IMPLEMENTASI *INTEGER LINEAR PROGRAMMING* DALAM SISTEM PENJADWALAN PERKULIAHAN BERBASIS WEB PADA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

(Skripsi)

Oleh

IQBAL AL HAFIDZU RAHMAN
2117051019

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

IMPLEMENTASI *INTEGER LINEAR PROGRAMMING* DALAM SISTEM PENJADWALAN PERKULIAHAN BERBASIS WEB PADA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

IQBAL AL HAFIDZU RAHMAN

Skripsi

Sebagai Salah Satu Syarat untuk Memperoleh Gelar
SARJANA KOMPUTER

Pada

Program Studi S1 Ilmu Komputer
Jurusan Ilmu Komputer

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

ABSTRAK

IMPLEMENTASI *INTEGER LINEAR PROGRAMMING* DALAM SISTEM PENJADWALAN PERKULIAHAN BERBASIS WEB PADA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

IQBAL AL HAFIDZU RAHMAN

Masalah penjadwalan perkuliahan merupakan salah satu permasalahan optimasi yang kompleks dan sering dikaji dalam bidang riset operasi (*Operations Research*). Salah satu pendekatan yang banyak digunakan untuk menyelesaikan permasalahan ini adalah *Integer Linear Programming* (ILP). Penelitian ini bertujuan untuk mengembangkan model optimasi penjadwalan perkuliahan berbasis ILP dengan menggunakan *library* Python PuLP. Penelitian dilakukan di Jurusan Ilmu Komputer Universitas Lampung pada Semester Ganjil Tahun Ajaran 2024/2025, dengan menggunakan data perkuliahan dari periode akademik 2023 Ganjil dan 2024 Genap. Seluruh aturan penyusunan jadwal diformulasikan ke dalam bentuk kendala matematis dan diimplementasikan dalam bentuk model optimasi. Model kemudian diselesaikan menggunakan fungsi *solver Coin-or Branch and Cut* (CBC) dan dianalisis dari segi validitas, kelayakan, dan efisiensi komputasi. Hasil pengujian menunjukkan bahwa model mampu menghasilkan jadwal tanpa pelanggaran kendala dengan status solusi optimal. Tiga kali pengujian terhadap data semester 2023 Ganjil menghasilkan waktu komputasi rata-rata 26,44 detik dengan penggunaan CPU antara 10,5% hingga 13,7%, memori sekitar 665 MB, dan total 264 jadwal yang terdiri dari 34 jadwal daring dan 230 jadwal luring. Dengan hasil ini, model yang dikembangkan terbukti efektif dan memiliki potensi untuk dikembangkan lebih lanjut dalam sistem penjadwalan akademik yang fleksibel dan terintegrasi.

Kata kunci: Penjadwalan Perkuliahan, *Integer Linear Programming* (ILP), PuLP, Optimasi, Riset Opera

ABSTRACT

IMPLEMENTATION OF INTEGER LINEAR PROGRAMMING IN A WEB-BASED COURSE SCHEDULING SYSTEM AT THE DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF LAMPUNG

By

IQBAL AL HAFIDZU RAHMAN

Course scheduling is one of the most complex and frequently studied optimization problems in the field of Operations Research. One widely used approach to solving this problem is Integer Linear Programming (ILP). This study aims to develop an ILP-based optimization model for course scheduling using the Python library PuLP. The research was conducted at the Department of Computer Science, University of Lampung, during the Odd Semester of the 2024/2025 Academic Year, utilizing course data from the 2023 Odd and 2024 Even academic periods. All scheduling rules were formulated into mathematical constraints and implemented in the optimization model. The model was then solved using the Coin-or Branch and Cut (CBC) solver and analyzed in terms of validity, feasibility, and computational efficiency. The test results show that the model was able to generate schedules without any constraint violations, with an optimal solution status. Three tests on data from the 2023 Odd Semester produced an average computation time of 26.44 seconds, CPU usage ranging from 10.5% to 13.7%, approximately 665 MB of memory usage, and a total of 264 schedules, consisting of 34 online and 230 offline schedules. These results demonstrate that the developed model is effective and has the potential to be further expanded into a flexible and integrated academic scheduling system.

Keywords*: Course Scheduling, Integer Linear Programming (ILP), PuLP, Optimization, Operations Research.*

Judul Skripsi	:	**IMPLEMENTASI *INTEGER LINEAR PROGRAMMING* DALAM SISTEM PENJADWALAN PERKULIAHAN BERBASIS WEB PADA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG**
Nama Mahasiswa	:	Iqbal Al Hafidzu Rahman
Nomor Pokok Mahasiswa	:	2117051019
Program Studi	:	S1 Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Febi Eka Febriansyah, M.T.
NIP. 19800219 200604 1 001

2. Ketua Jurusan Ilmu Komputer

Dwi Sakethi, S.Si., M.Kom.
NIP. 19680611199802 1 001

MENGESAHKAN

1. Tim Penguji

Ketua : **Febi Eka Febriansyah, M.T.** .....

Penguji Pembahas I : **Didik Kurniawan, S.Si.,M.T.** .....

Penguji Pembahas II : **Wartariyus, S.Kom., M.T.I.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.
NIP. 19711001 200501 1 002

Tanggal Lulus Ujian Skripsi: **23 Mei 2025**

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Iqbal Al Hafidzu Rahman

NPM : 2117051019

Menyatakan bahwa skripsi saya yang berjudul "**Implementasi *Integer Linear Programming* Dalam Sistem Penjadwalan Perkuliahan Berbasis Web Pada Ilmu Komputer Universitas Lampung**" merupakan karya saya sendiri dan bukan karya orang lain. Semua tulisan yang tertuang di skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil penjiplakan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 23 Mei 2025

Penulis,

Iqbal Al Hafidzu Rahman

NPM. 2117051019

RIWAYAT HIDUP

Penulis dilahirkan di Samarinda pada tanggal 22 Agustus 2003 sebagai anak pertama dari dua bersaudara, dari Bapak Fahmi Rahman. dan Ibu Rita Kadir, S.Pd. Penulis telah menyelesaikan pendidikan dasar di SD Negeri 1 Beringin Raya pada tahun 2015, pendidikan menengah pertama di SMP Negeri 4 Bandar Lampung pada tahun 2018, dan pendidikan menengah atas di SMA Negeri 7 Bandar Lampung pada tahun 2021. Perjalanan pendidikan penulis dilanjutkan dengan terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur Seleksi Nasional Masuk Perguruan Tinggi (SNMPTN) pada tahun 2021.

Selama menjadi mahasiswa, penulis aktif mengikuti beberapa kegiatan antara lain:

1. Asisten Praktikum mata kuliah Logika dan Sistem Operasi pada tahun 2022 dan 2023.
2. Kepala Bidang Keilmuan Himpunan Mahasiswa Jurusan Ilmu Komputer (HIMAKOM) Universitas Lampung pada tahun 2023.
3. MBKM Riset Independen di Jurusan Ilmu Komputer Universitas Lampung sebagai anggota penelitian dengan topik Penerapan *Convolutional Neural Networ*k untuk Pengenalan Wajah dalam Sistem Kehadiran Mahasiswa pada tahun 2023.
4. MBKM Magang Bersertifikat di Educourse.id melalui program MSIB *batch* 6 sebagai *Full Stack Engineer* pada tahun 2024.

MOTTO

“If you fail to plan, you plan to fail.”
(D. M.)

"No effort, no excuse."
(J. H. M.)

"Anti-algoritma, pro-rasa."
(N. A. F. H.)

"You were made to shine, go change the world!."
(T. A. P.)

“Don’t Cry because it’s over, Smile because it Happened.”
(Dr. Seuss)

PERSEMBAHAN

Alhamdulillahirabbil'alamin

Puji syukur kehadirat Allah Subhanahu Wa Ta'ala atas segala rahmat dan karunia-Nya sehingga skripsi ini dapat diselesaikan dengan sebaik-baiknya. Shalawat beriring salam selalu tercurahkan kepada junjungan Nabi Agung Muhammad Shallallahu 'Alaihi Wasallam.

Aku persembahkan karya ini kepada:

Kedua Orang Tuaku Tersayang
dan
Keluargaku Tercinta

Atas segala doa yang tak henti dipanjatkan, cinta yang tak pernah habis dibagikan, serta pengorbanan dan ketulusan dalam membimbing setiap langkah hidupku. Terima kasih telah menjadi sandaran, penyemangat, dan tempat pulang yang paling tulus sepanjang perjalanan ini.

Seluruh Keluarga Besar Ilmu Komputer 2021

Atas kebersamaan, semangat, dan dukungan yang tak ternilai sepanjang perjalanan perkuliahan ini.

Almamater Tercinta, Universitas Lampung dan Jurusan Ilmu Komputer

Tempat bernaung dan menimba ilmu untuk bekal kehidupan dunia dan akhirat.

SANWACANA

Puji syukur atas segala rahmat Allah SWT. Yang diberikan kepada penulis sehingga dapat menyelesaikan skripsi berjudul "IMPLEMENTASI *INTEGER LINEAR PROGRAMMING* DALAM SISTEM PENJADWALAN PERKULIAHAN BERBASIS WEB PADA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG". Tidak lupa shalawat dan salam senantiasa dicurahkan kepada Nabi Muhammad SAW.

Pada kesempatan ini, penulis menyampaikan terima kasih kepada semua pihak yang telah memberikan bantuan dan dukungan, baik dalam penyusunan skripsi ini maupun selama perjalanan perkuliahan penulis secara keseluruhan, yaitu:

1. Kepada kedua orang tua, Akan dan Ina, serta adik penulis, Sarah, yang selalu menjadi teladan dalam kebaikan, senantiasa memenuhi segala kebutuhan, dukungan, dan kepercayaan penuh terhadap setiap keputusan yang diambil penulis hingga saat ini.
2. Keluarga besar penulis, Ateh, Bunda, Mama Tua, Ina Batin, Kakak Tiara, dan lainnya yang tidak dapat disebutkan satu per satu, atas segala dukungan, doa, dan kasih sayang yang telah diberikan.
3. Bapak Dr. Eng. Heri Satria, S.Si., M.Si. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
4. Bapak Dwi Sakethi, S.Si., M.Kom. selaku Ketua Jurusan Ilmu Komputer Universitas Lampung.
5. Ibu Anie Rose Irawati, S.T., M.Cs. selaku Dosen Pembimbing Akademik selama penulis menempuh perkuliahan.
6. Bapak Febi Eka Febriansyah, M.T. selaku Dosen Pembimbing Utama yang senantiasa memberikan bimbingan, arahan, kritik, serta saran yang bermanfaat kepada penulis baik dalam penyusunan skripsi maupun hal-hal di luar skripsi.

7. Bapak Didik Kurniawan, S.Si., M.T. selaku Dosen Pembahas sekaligus sosok pembimbing yang senantiasa memberikan bimbingan, arahan, kritik, dan saran yang berharga, baik dalam penyusunan skripsi maupun perjalanan akademik penulis secara keseluruhan.
8. Bapak Wartariyus, S.Kom., M.T.I. selaku Dosen Pembahas yang telah memberikan kritik, saran, dan masukan dalam penyusunan skripsi serta telah memberikan peluang karier kepada penulis.
9. Bapak dan Ibu Dosen Jurusan Ilmu Komputer Universitas Lampung yang telah memberikan ilmu, pengetahuan, serta pengalaman terbaik selama penulis menjadi mahasiswa.
10. Jihan dan Cindy sebagai rekan penelitian penulis yang telah memberikan dukungan, kerja sama, serta ide-ide konstruktif yang sangat membantu dalam kelancaran dan keberhasilan proses penelitian ini.
11. Seluruh Staf Jurusan Ilmu Komputer, termasuk Ibu Ade Nora Maela, yang telah membantu penulis dari awal hingga akhir masa perkuliahan.
12. Teman-teman seperjuangan, Reza, Zidan, John, Keyvin, Hanif, Ferry, Satria, Wangsa, Shalahuddin, Abdi, Rimuru, Qolby, Farel, Bagas, Rafi, Tasya, Enjel, Zahra, Nisa, Nathan, Wirda, dan seluruh mahasiswa Jurusan Ilmu Komputer angkatan 2021 yang telah menemani dengan dukungan, motivasi, kekuatan, dan kebersamaan yang sangat berharga.
13. Teman-teman SMP, Arkan, Okta, Marco, Aul, Akhlis, Arjuna, Rio, Abyan, Puteranda, Abuzar, Toriq, Mei, Salsa, Monique, Caecaro, Aura, Rai, Danish, dan lainnya yang tidak dapat disebutkan satu per satu yang senantiasa berbagi pengalaman kepada penulis.
14. Teman-teman magang, Deana, Shanin, Daffa, Faldi, Nindya, Nara, Rena, Mawar, Indah, Oqsiana, Salsa, Dave, serta seluruh teman magang Educourse.id, beserta para mentor yang telah membimbing dan membersamai penulis dalam proses belajar, berbagi pengalaman, dan bekerja sama selama masa magang.
15. Teman-teman belajar, Aliya, Septia, Basid, Alika, Rendy yang selalu bertukar pengalaman serta berbagi ilmu dengan penulis di kopken.
16. Rekan-rekan kerja, Ms. Shindy, Pak Desfan, Ms. Vira, Bu Ika, Pak Abd, Ms. Maris, Ms. Yuni, Ms. Salwa, Ms. Erta, Ms. Adelia, Pak Arya, Pak Dana, Ms. Icha,

Pak Irfan, Pak Syahroni, Pak Agung, Pak Yud, Ms. Rosita, Ms. Trian, Ms. Winda, Ms. Ririn, dan lainnya yang tidak dapat disebutkan satu per satu yang senantiasa berbagi pengalaman kepada penulis.

17. Teman-teman perkeretaapian, Tesalonika, Rasti, dan lainnya yang tidak dapat disebutkan satu per satu yang senantiasa menyemangati penulis.

18. Siswa-siswa yang penulis banggakan, Naila, Khoirunisa, Yesaya, Alya, Fahri, Seli, Clara, Faisol, Sakha, Naura, dan lainnya yang tidak dapat disebutkan satu per satu yang senantiasa semangat belajar bersama penulis.

19. Teman-teman KKN, Joey, Ikhsan, Arasso, Nina, Tere, Widri, serta warga Desa Donomulyo atas kerja sama dan kebersamaan yang terjalin selama masa tinggal dan pelaksanaan kegiatan di desa.

20. Seluruh pihak yang telah membantu penulis secara langsung maupun tidak langsung, atas dukungannya dalam menyelesaikan skripsi dan perkuliahan.

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan skripsi ini. Akan tetapi, penulis berharap skripsi ini dapat membawa manfaat dan keberkahan bagi perkembangan ilmu pengetahuan terutama bagi civitas Ilmu Komputer Universitas Lampung.

Bandar Lampung, 24 Juni 2025

Iqbal Al Hafidzu Rahman
NPM. 2117051019

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I
PENDAHULUAN

1.1. Latar Belakang

Penjadwalan adalah salah satu aspek krusial dalam pengelolaan waktu dan sumber daya di berbagai bidang, termasuk pendidikan. Penjadwalan diciptakan untuk memastikan bahwa berbagai kegiatan dapat berlangsung secara terorganisir, teratur, dan efisien. Dalam konteks pendidikan, penjadwalan perkuliahan berfungsi untuk mengatur jadwal kelas, dosen, dan ruang perkuliahan secara optimal, sehingga semua pihak yang terlibat dapat menjalankan kegiatan akademik dengan lancar dan efektif (Anam et al., 2021).

Awalnya, penjadwalan dilakukan secara manual, di mana pengelola harus menyusun jadwal secara fisik, sering kali menggunakan kertas, papan tulis, atau *spreadsheet* sederhana (Irvin & Brown, 1999). Penjadwalan manual ini memerlukan banyak waktu dan tenaga, serta rawan kesalahan manusia, seperti tumpang tindih jadwal atau kekurangan alokasi waktu yang tepat. Meskipun metode ini memungkinkan fleksibilitas tertentu, namun tingkat kesalahan dan efisiensinya dapat mengganggu kelancaran proses pendidikan.

Dengan perkembangan teknologi, sistem penjadwalan mulai diadopsi dalam bentuk digital, di mana data di-*input* secara manual ke dalam perangkat lunak tertentu (Varela et al., 2004). Sistem ini memungkinkan pengelola untuk lebih mudah menyusun jadwal, menyimpan data, dan melakukan perubahan secara lebih cepat dibandingkan dengan metode manual sepenuhnya. Namun, meskipun teknologi ini memberikan beberapa kemudahan, *input* secara manual masih membutuhkan intervensi manusia yang signifikan, dan kesalahan tetap bisa

terjadi, terutama jika data tidak diperiksa secara cermat. Selain itu, sistem penjadwalan digital manual masih menghadapi tantangan besar dalam memenuhi berbagai kendala (*constraint*) yang ada. Misalnya, jadwal kelas sering kali berbenturan dengan ketersediaan ruangan yang terbatas, atau kebutuhan akan kelas laboratorium untuk mata kuliah praktikum yang hanya bisa dilaksanakan di ruang khusus dengan fasilitas tertentu.

Untuk mengatasi permasalahan penjadwalan yang kompleks dan beragam, muncul kebutuhan akan perencanaan jadwal otomatis yang dapat meminimalkan konflik dan meningkatkan efisiensi. Dalam penjadwalan perkuliahan, berbagai kendala seperti ketersediaan dosen, kapasitas ruangan, dan kebutuhan fasilitas khusus harus diperhitungkan dengan tepat. Salah satu metode optimasi yang dapat digunakan untuk mengatasi masalah ini adalah *Integer Linear Programming* (ILP). ILP bekerja dengan memformulasikan masalah penjadwalan sebagai serangkaian persamaan dan pertidaksamaan linear, yang kemudian dioptimalkan menggunakan pemecah (*solver*) khusus seperti melalui penggunaan metode *branch-and-bound* atau *branch-and-cut* (Trilling et al., 2006). Dengan metode ini, ILP dapat menangani berbagai kendala sekaligus, seperti mencegah bentrokan jadwal dan memastikan alokasi ruang yang efisien.

Penggunaan *Integer Linear Programming* (ILP) adalah pilihan yang sangat tepat karena kemampuannya dalam menyelesaikan masalah optimasi yang kompleks secara otomatis dan akurat (Trauth & Woolsey, 1969). Dengan ILP, pemanfaatan sumber daya seperti ruangan dan waktu dapat dioptimalkan secara maksimal, serta konflik penjadwalan dapat diselesaikan tanpa perlu intervensi manual. Hal ini menjadikan ILP sangat sesuai untuk sistem penjadwalan perkuliahan yang membutuhkan ketepatan tinggi dan fleksibilitas dalam memenuhi berbagai kendala (*constraint*).

Keunggulan ILP terletak pada kemampuannya dalam memberikan solusi yang optimal secara matematis, berbeda dengan pendekatan menggunakan metode lain seperti *Constraint Satisfaction Problem* (CSP) atau Algoritma Genetik. ILP memastikan solusi yang diberikan bersifat optimal karena memecahkan masalah secara determinatif, sedangkan CSP lebih fokus pada pencarian solusi yang

feasible tanpa menjamin keoptimalan. Sementara itu, Algoritma Genetik bekerja dengan pendekatan heuristik yang sering kali tidak menjamin hasil yang optimal dan memerlukan waktu lebih lama untuk mencapai solusi yang mendekati optimal. Oleh karena itu, ILP menjadi pilihan yang lebih unggul dalam hal keakuratan dan efisiensi waktu, terutama pada sistem penjadwalan yang melibatkan banyak kendala (*constraint*) yang saling terkait.

1.2. Rumusan Masalah

Berdasarkan permasalahan yang telah diuraikan pada latar belakang, rumusan masalah pada penelitian ini adalah bagaimana mengimplementasikan metode optimasi *Integer Linear Programming* (ILP) untuk membuat *web service* yang menangani pembuatan penjadwalan otomatis dalam aplikasi penjadwalan otomatis berbasis web di Jurusan Ilmu Komputer Universitas Lampung.

1.3. Batasan Masalah

Adapun batasan masalah yang ditetapkan dalam penelitian ini sebagai berikut.

a. Penelitian ini berfokus pada formulasi model *Integer Linear Programming* menggunakan *library* PuLP pada bahasa pemrograman Python.
b. Pemecah (*solver*) yang digunakan pada penelitian ini akan menerapkan metode *branch-and-cut*.
c. Penelitian tidak mencakup penjelasan mengenai cara kerja *solver* dalam mendapatkan solusi.
d. Model yang telah dibuat akan diaplikasikan ke dalam *web service* khusus yang dibangun menggunakan *framework* FastAPI yang akan menangani pembuatan jadwal secara otomatis.
e. Penelitian ini akan dilakukan dengan menggunakan data penjadwalan Jurusan Ilmu Komputer Universitas Lampung pada T.A. 2023/2024.

f. Data yang digunakan dalam penelitian ini diperoleh dari observasi dan wawancara.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah mengimplementasikan metode optimasi *Integer Linear Programming* (ILP) ke dalam *web service* yang menangani penjadwalan otomatis dalam aplikasi penjadwalan otomatis berbasis web di Jurusan Ilmu Komputer Universitas Lampung.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah sebagai berikut.

a. Penelitian ini akan meningkatkan efisiensi dalam penyusunan jadwal perkuliahan melalui implementasi dan integrasi *Integer Linear Programming* (ILP) ke dalam aplikasi berbasis web yang mampu menyusun jadwal secara otomatis, sehingga mengurangi beban kerja manual dan mempercepat proses penyusunan.
b. Dengan penerapan *Integer Linear Programming* (ILP), maka jadwal yang dibuat akan mengoptimalkan penggunaan sumber daya seperti ruangan dan waktu, serta membantu menghindari bentrokan jadwal yang sering terjadi.
c. Penelitian ini akan menyediakan referensi yang bermanfaat bagi peneliti lain dalam penerapan *Integer Linear Programming* (ILP) untuk menyelesaikan masalah penjadwalan.

BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian terdahulu merupakan penelitian yang telah dilakukan sebelumnya. Penelitian yang terkait dalam penggunaan *Integer Linear Programming* untuk pembagian penjadwalan sebagai bentuk kajian dan dasar yang memperkuat rangka penyusunan dengan penelitian yang dilakukan dapat dilihat pada tabel 1 di bawah:

Tabel 1. Penelitian Terdahulu.

No.	Peneliti	Judul Penelitian	Hasil Penelitian
1	(Safitri et al., 2021)	Optimasi Penjadwalan Perawat Menggunakan Integer Linear Programming (Studi Kasus: RS. Aulia Hospital Pekanbaru)	Model *Integer Linear Programming* pada penjadwalan perawat
2	(Livia & Oktiarso, 2017)	Penjadwalan Untuk Meminimalkan Total Tardiness Dengan Metode Integer Linear Programming	Model ILP yang meminimalkan dan menghasilkan total tardiness lebih kecil daripada penjadwalan FCFS
3	(Chanida Leelayutto , 2019)	Classroom Scheduling Problem with an integer linear program	Pembuatan model ILP menggunakan perangkat lunak CPLEX Studio (versi 12.6.3) dan Python (versi 3.7) untuk menangani masalah penjadwalan

2.1.1. Penjadwalan Untuk Meminimalkan Total Tardiness Dengan Metode Integer Linear Programming

Penjadwalan produksi merupakan elemen penting bagi perusahaan untuk memastikan permintaan konsumen terpenuhi tepat waktu. PT Mitra Mulia Makmur saat ini menerapkan metode penjadwalan *First Come First Serve* (FCFS) dalam proses produksinya untuk menghasilkan produk kontainer plastik. Namun, metode ini belum cukup efektif dan efisien karena masih terdapat permintaan konsumen yang terlambat diselesaikan. Keterlambatan ini disebabkan oleh waktu pengerjaan yang lebih lama dari batas waktu yang telah ditentukan. Untuk mengatasi masalah ini, penelitian ini menerapkan metode *Integer Linear Programming* (ILP) dengan tujuan meminimalkan total keterlambatan (*total tardiness*) dalam proses produksi.

Penelitian ini melalui beberapa tahapan utama, dimulai dengan pembuatan formulasi matematis dari masalah penjadwalan, dilanjutkan dengan penerjemahan formulasi tersebut ke dalam *software* LINGO 16.0, dan diakhiri dengan pembentukan tabel penjadwalan baru. Metode ILP dipilih karena kemampuannya dalam menghasilkan solusi yang optimal dengan mempertimbangkan kendala yang ada. Berdasarkan hasil penelitian, penggunaan ILP terbukti mampu mengurangi total keterlambatan secara signifikan dibandingkan dengan metode FCFS. Selama tahun 2016, jumlah pekerjaan yang terlambat berkurang dari 10 *job* dengan total keterlambatan 32.292,6 menit pada penjadwalan FCFS, menjadi hanya 5 *job* dengan total keterlambatan 20.003,33 menit menggunakan metode ILP.

Penelitian ini relevan dengan penelitian yang sedang ditulis karena melibatkan tahapan yang serupa, yaitu formulasi matematis, penerapan model matematis ke dalam *software*, dan penerapan pada skenario dunia nyata. Tahapan-tahapan ini sangat penting dalam mengembangkan model penjadwalan yang dapat digunakan untuk meningkatkan efisiensi proses produksi. Penggunaan ILP dalam penelitian ini memperkuat penerapan teori optimasi dalam konteks masalah nyata, yang

sejalan dengan fokus penelitian terhadap penggunaan teknik optimasi untuk menghasilkan solusi yang lebih efektif.

2.1.2. Optimasi Penjadwalan Perawat Menggunakan Integer Linear Programming (Studi Kasus: RS. Aulia Hospital Pekanbaru)

Penelitian ini didasarkan pada penelitian sebelumnya yang berjudul "*Nurse Scheduling Model with the Work Sift and Work Location"* oleh (Harlina et al., 2019). Penelitian ini kemudian dikembangkan lebih lanjut dengan kasus yang berbeda, yaitu membahas penjadwalan perawat berdasarkan *shift* kerja, serta melakukan perbandingan antara penjadwalan manual dan penjadwalan yang dihasilkan menggunakan *software* Lingo di RS Aulia Hospital Pekanbaru. Pendekatan yang digunakan adalah *Integer Linear Programming* (ILP) untuk mengoptimalkan jadwal perawat di tiga ruangan utama, yaitu ruang perawatan, ruang IGD, dan ruang ICU. Tujuan utamanya adalah meminimalkan jumlah *shift* yang dijadwalkan bagi perawat sekaligus memastikan bahwa setiap *shift* memiliki jumlah perawat yang memadai sesuai dengan standar operasional rumah sakit. Penelitian ini juga menerapkan beberapa asumsi dan kendala seperti: setiap perawat tidak boleh bekerja lebih dari satu *shift* per hari, perawat harus bekerja minimal 22 hari dalam satu bulan, dan *shift* kerja diatur sedemikian rupa untuk menghindari kelelahan berlebih.

Penelitian ini sangat relevan dengan penjadwalan otomatis yang menggunakan metode optimasi *Integer Linear Programming* untuk menangani masalah optimasi penjadwalan, dengan mempertimbangkan beberapa kendala spesifik. Pada penelitian di RS Aulia Hospital Pekanbaru, kendala seperti perawat yang tidak boleh bekerja lebih dari satu *shift* dalam satu hari, dan perawat yang harus bekerja minimal 22 hari dalam satu bulan, mirip dengan kendala dalam penjadwalan perkuliahan, seperti dosen yang tidak boleh mengajar dua sesi yang bertabrakan, serta ruangan yang tidak boleh digunakan untuk lebih dari satu kelas dalam waktu yang sama.

Penelitian ini juga menunjukkan bahwa penggunaan ILP lebih efektif dibandingkan metode manual dalam menghasilkan jadwal yang optimal dan sesuai dengan kebutuhan, baik itu di lingkungan rumah sakit maupun di institusi pendidikan. Hal ini menggarisbawahi fleksibilitas dan kemampuan ILP dalam menangani masalah penjadwalan di berbagai konteks, serta potensinya untuk meningkatkan efisiensi dan penggunaan sumber daya secara optimal.

2.1.3. Classroom Scheduling Problem With an Integer Linear Program

Proyek penelitian ini berfokus pada penyelesaian masalah penjadwalan kelas menggunakan pendekatan *Integer Linear Programming* (ILP). Masalah penjadwalan kelas merupakan tantangan besar bagi banyak universitas, karena metode manual dalam membuat jadwal seringkali menghasilkan inefisiensi. Proyek ini bertujuan untuk mengoptimalkan alokasi ruang kelas, memastikan kapasitas ruangan sesuai dengan jumlah mahasiswa, serta menjadwalkan dosen secara efisien, sambil mematuhi berbagai kendala, seperti ketersediaan ruang kelas dan *slot* waktu yang terbatas.

Proyek ini secara khusus menggunakan data dari tahun 2019, yang disediakan oleh Kantor Registrasi dan Departemen Matematika dan Ilmu Komputer di Universitas Chulalongkorn. Fokusnya adalah pada mata kuliah di dalam Departemen Matematika dan Ilmu Komputer, dan menjadwalkan kelas-kelas yang berlangsung antara pukul 08.00 hingga 17.00.

Penelitian ini melibatkan pembuatan model menggunakan perangkat lunak CPLEX Studio (versi 12.6.3) dan Python (versi 3.7) untuk menangani masalah penjadwalan. Langkah-langkah utama dalam proyek ini meliputi tinjauan literatur untuk mengidentifikasi kendala yang relevan, membangun model ILP, serta menggunakan CPLEX Studio untuk memprogram dan menyelesaikan model tersebut. Tujuan dari penelitian ini adalah untuk menunjukkan bagaimana ILP dapat digunakan untuk menghasilkan jadwal kelas yang optimal, yang mengatasi

masalah sumber daya dan anggaran, sekaligus meningkatkan efisiensi penjadwalan.

2.2. Uraian Landasan Teori

Berikut ini adalah beberapa teori yang berkaitan dengan penelitian yang sedang dilakukan, yang memberikan landasan konseptual dan mendukung pemahaman lebih mendalam mengenai topik yang dibahas.

2.2.1. Penjadwalan

Penjadwalan adalah proses pengorganisasian aktivitas, tugas, atau peristiwa yang memperhatikan prioritas dan/atau kendala sumber daya yang ada (Herroelen, 2005). Dalam konteks akademik atau industri, penjadwalan sering kali melibatkan pengaturan waktu untuk kelas, pertemuan, produksi, atau pengiriman layanan berdasarkan berbagai kendala yang ada, seperti ketersediaan sumber daya atau prioritas tugas. Tujuan utama dari penjadwalan adalah memastikan bahwa setiap aktivitas dapat dilakukan tepat waktu dan sesuai dengan kapasitas yang tersedia. Menurut (Muhammad Asrar Amir, 2017), Penjadwalan yang baik dapat memaksimalkan pemenuhan dari kendala-kendala yang ada, untuk menyusun sebuah jadwal yang optimal dibutuhkan teknik optimasi yang dapat mempertimbangkan berbagai aspek yang ada.

Pentingnya penjadwalan terletak pada kemampuannya untuk mengoptimalkan penggunaan sumber daya dan meminimalkan konflik atau ketidakpastian dalam pelaksanaan tugas. Dalam skala besar, seperti penjadwalan produksi atau pengajaran, penjadwalan yang efektif dapat meningkatkan efisiensi operasional, mengurangi biaya, serta meningkatkan kepuasan pihak-pihak yang terlibat. Dalam lingkungan pendidikan, misalnya, penjadwalan yang baik memastikan bahwa kelas, dosen, dan ruang belajar dialokasikan dengan tepat, sehingga tidak ada waktu atau sumber daya yang terbuang. Demikian juga, dalam lingkungan bisnis,

penjadwalan yang tepat dapat membantu perusahaan menghindari keterlambatan produksi dan memastikan bahwa proyek dapat diselesaikan sesuai jadwal.

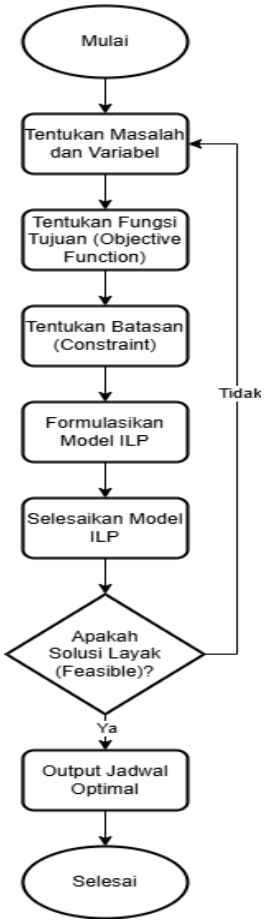
Perkembangan penjadwalan telah berubah seiring waktu. Pada awalnya, penjadwalan dilakukan secara manual dengan menggunakan papan tulis atau kertas, yang memerlukan waktu dan tenaga kerja yang besar untuk mengelola perubahan dan memperbarui jadwal. Seiring dengan kemajuan teknologi, penjadwalan mulai dilakukan menggunakan perangkat lunak *spreadsheet* seperti Microsoft Excel (Irvin & Brown, 1999), yang memungkinkan perhitungan lebih mudah dan penyimpanan data secara digital. Namun, penggunaan Excel masih memerlukan *input* dan kontrol manual yang rentan terhadap kesalahan. Saat ini, perkembangan teknologi lebih lanjut telah membawa penjadwalan ke tingkat otomatisasi, dengan sistem otomatis yang dapat menggunakan algoritma untuk menghasilkan jadwal optimal secara cepat, berdasarkan berbagai faktor dan kendala yang telah ditentukan sebelumnya. Sistem penjadwalan otomatis ini tidak hanya menghemat waktu tetapi juga meningkatkan akurasi dan efisiensi, terutama dalam skenario kompleks seperti penjadwalan di institusi pendidikan atau industri besar.

2.2.2. *Integer Linear Programming*

*Integer Linear Programming (*ILP) adalah metode optimasi matematis yang menyelesaikan masalah dengan variabel keputusan berupa bilangan bulat. Metode ini bekerja dengan memaksimalkan atau meminimalkan fungsi tujuan yang dibatasi oleh sejumlah kendala dalam bentuk persamaan dan pertidaksamaan linear (Graver, 1975). ILP merupakan suatu *linear programming* dengan variabel keputusannya berupa bilangan bulat (integer), sehingga pada bentuk umum *linear programming* terdapat tambahan syarat bahwa variabel keputusannya harus bilangan bulat (Basriati, 2018). Metode ini bertujuan untuk memaksimalkan atau meminimalkan suatu fungsi tujuan, yang dibatasi oleh serangkaian kendala dalam bentuk persamaan atau pertidaksamaan linear. ILP sering digunakan dalam situasi di mana solusi harus berbentuk diskrit atau terstruktur, seperti penjadwalan,

alokasi sumber daya, dan optimasi dalam sistem logistik. Metode ini memungkinkan formulasi masalah kompleks menjadi lebih sistematis, sehingga dapat menghasilkan solusi optimal yang memenuhi semua kendala yang telah ditentukan.

Jurnal "*On the Foundations of Linear and Integer Linear Programming I*" membahas prinsip dasar dan teori yang mendasari pemrograman linear (LP) dan pemrograman bilangan bulat (ILP) . Menurut jurnal tersebut, LP fokus pada variabel keputusan yang bersifat kontinu, sedangkan ILP menangani masalah di mana variabel keputusan harus berupa bilangan bulat. Pendekatan ini menjadi sangat penting dalam konteks masalah yang memerlukan solusi diskrit, seperti penjadwalan, alokasi sumber daya, dan perencanaan produksi. *Flowchart* untuk pengimplementasian ILP pada kasus penjadwalan dapat dilihat pada gambar berikut.



Gambar 1. *Flowchart* implementasi ILP pada kasus penjadwalan.

2.2.2.1. Variabel Keputusan (*Decision Variables*)

Decision variables atau variabel keputusan adalah komponen inti dalam model *Integer Linear Programming* (ILP), yang direpresentasikan sebagai elemen-elemen dalam matriks dan berfungsi sebagai parameter keputusan (Serafini, 2005). Variabel-variabel ini biasanya dalam bentuk bilangan bulat, dan mereka menentukan hasil akhir dari solusi. Setiap variabel keputusan mencerminkan aspek yang ingin dioptimalkan, seperti alokasi sumber daya, penjadwalan, atau distribusi barang, yang akan memaksimalkan atau meminimalkan fungsi tujuan dengan tetap mematuhi kendala yang ada.

Dalam konteks ILP, variabel keputusan memiliki peran penting dalam menggambarkan keputusan yang perlu diambil. Misalnya, dalam penjadwalan, variabel keputusan dapat menggambarkan apakah seorang dosen mengajar suatu kelas di *slot* waktu tertentu atau tidak (misalnya, 1 jika mengajar dan 0 jika tidak). Keputusan ini diatur melalui variabel bilangan bulat dan akan digunakan dalam proses optimasi untuk mencari solusi terbaik.

Salah satu kategori umum dari variabel keputusan adalah variabel biner, yang hanya dapat mengambil dua nilai, biasanya 0 atau 1. Variabel biner sering digunakan untuk keputusan ya atau tidak, seperti apakah suatu kelas akan ditempatkan di ruangan tertentu pada waktu tertentu. Variabel ini sangat berguna dalam masalah-masalah yang melibatkan keputusan diskrit, di mana hasilnya hanya bisa dalam dua keadaan.

Selain variabel biner, ILP juga menggunakan variabel integer umum, yang dapat mengambil nilai bilangan bulat positif atau negatif. Variabel ini lebih fleksibel dan bisa digunakan untuk menggambarkan keputusan yang melibatkan kuantitas, seperti jumlah barang yang harus diproduksi, berapa sesi yang harus dijadwalkan dalam satu hari, atau berapa sumber daya yang dialokasikan ke suatu proyek. Variabel integer umum ini penting ketika keputusan yang melibatkan perhitungan kuantitatif diperlukan.

2.2.2.2. Fungsi Objektif (*Objective Function*)

Objective function atau fungsi objektif dalam *Integer Linear Programming* (ILP) adalah ekspresi matematis yang menggambarkan tujuan dari masalah optimasi (Tantawy, 2014). Fungsi ini merepresentasikan nilai yang harus diminimalkan atau dimaksimalkan, dan akan dipengaruhi oleh variabel keputusan dalam model. Fungsi objektif adalah inti dari setiap model ILP, karena memberikan arahan jelas tentang apa yang ingin dicapai dalam proses optimasi, seperti meminimalkan biaya, memaksimalkan keuntungan, atau mengoptimalkan penggunaan sumber daya.

Dalam ILP, fungsi tujuan biasanya ditulis dalam bentuk persamaan linier yang melibatkan variabel keputusan dan parameter. Sebagai contoh, dalam masalah penjadwalan, fungsi tujuan mungkin ingin meminimalkan total biaya operasional, seperti mengurangi waktu menganggur ruangan atau mengoptimalkan alokasi tenaga pengajar. Fungsi ini dirumuskan dengan menggunakan variabel keputusan yang relevan, seperti apakah suatu kelas dijadwalkan di ruangan tertentu atau berapa sesi yang dijadwalkan pada hari tertentu.

Setiap fungsi objektif dikaitkan erat dengan variabel keputusan, karena nilai dari fungsi tujuan tergantung pada nilai variabel-variabel ini. Fungsi tujuan berperan dalam mengarahkan pencarian solusi, di mana algoritma optimasi ILP akan mencoba menemukan kombinasi nilai variabel yang menghasilkan hasil optimal sesuai dengan kriteria yang ditentukan.

Fungsi tujuan juga sering dibatasi oleh berbagai kendala (*constraints*) yang menentukan ruang solusi. Misalnya, meskipun fungsi tujuan mungkin ingin meminimalkan biaya, kendala seperti ketersediaan ruangan, kapasitas kelas, atau ketersediaan dosen harus tetap dipenuhi. Dengan kata lain, ILP akan mencari nilai minimal atau maksimal dari fungsi tujuan hanya dalam batas-batas yang diperbolehkan oleh kendala (*constraint*) yang ditentukan.

Dalam beberapa masalah yang lebih kompleks, *multiple objective functions* dapat digunakan, yang memungkinkan untuk mengoptimalkan beberapa tujuan secara

bersamaan. Sebagai contoh, dalam sistem penjadwalan, mungkin perlu meminimalkan biaya sekaligus memaksimalkan kepuasan pengajar atau mahasiswa. Pendekatan ini memerlukan penyeimbangan antara berbagai tujuan, di mana prioritas tertentu diberikan kepada salah satu atau beberapa fungsi tujuan.

Sebagai contoh lain, dalam *Multiobjective Decision Making* (MODM), sering kali terdapat lebih dari satu tujuan yang ingin dicapai, yang biasanya saling bertentangan dan tidak sebanding. Model MODM melibatkan vektor variabel keputusan, fungsi tujuan, dan kendala. Misalnya, untuk sebuah perusahaan yang berorientasi pada keuntungan, selain menghasilkan uang, perusahaan tersebut juga ingin mengembangkan produk baru, menyediakan keamanan kerja bagi karyawan, dan melayani masyarakat. Para manajer ingin memuaskan para pemegang saham sekaligus menikmati gaji yang tinggi dan tunjangan lainnya, sementara karyawan ingin meningkatkan pendapatan dan manfaat mereka. Ketika keputusan harus diambil, misalnya tentang proyek investasi, beberapa tujuan ini saling melengkapi sementara yang lainnya saling bertentangan.

Masalah ini dapat diformulasikan dalam bentuk *Multiobjective Linear Programming* (MOLP) yang melibatkan fungsi tujuan linear yang dimaksimalkan atau diminimalkan dengan sejumlah kendala linier. Bentuk standar dari masalah MOLP dapat dinyatakan sebagai berikut: $max\, f(x) = Cx, s.t.\ x \in S = \{x \in Rn: Ax \leq b, x \geq 0\}$, di mana C adalah matriks fungsi tujuan berukuran $k \times n, A$ adalah matriks kendala berukuran $m \times n, b$ adalah vektor kendala berukuran m, dan x adalah vektor variabel keputusan berukuran n. Dalam hal ini, pemecahan masalah tersebut jarang menghasilkan solusi tunggal yang unik, sehingga pengambil keputusan diharapkan untuk memilih solusi dari sekumpulan solusi yang efisien sebagai alternatif (Roostaee et al., 2012). Contoh di atas menunjukkan bahwa dalam penerapan ILP atau MODM, pengambilan keputusan harus mempertimbangkan berbagai tujuan dan kendala yang ada untuk mencapai solusi optimal yang memenuhi kriteria yang telah ditentukan.

Secara keseluruhan, fungsi objektif berfungsi sebagai kompas dalam model ILP, yang mengarahkan pencarian solusi optimal berdasarkan kriteria yang telah ditentukan. Dengan merumuskan fungsi objektif yang tepat, ILP dapat

memecahkan masalah optimasi yang kompleks dan memberikan hasil yang sesuai dengan tujuan praktis yang diinginkan, baik itu dalam konteks penjadwalan, alokasi sumber daya, atau masalah lainnya.

2.2.2.3. Kendala (*Constraints*)

Constraints atau kendala dalam *Integer Linear Programming* (ILP) adalah aturan atau kendala yang membatasi ruang solusi yang dapat dipilih untuk mencapai solusi optimal. *Constraints* memainkan peran penting dalam memastikan bahwa solusi yang dihasilkan oleh ILP sesuai dengan kondisi atau kendala dunia nyata, seperti ketersediaan sumber daya, kendala waktu, atau kapasitas. Dalam ILP, *constraints* dirumuskan sebagai persamaan atau pertidaksamaan linier yang melibatkan variabel keputusan.

Setiap *constraint* mendefinisikan suatu kendala spesifik yang harus dipenuhi oleh solusi. Misalnya, dalam masalah penjadwalan, *constraint* dapat menggambarkan bahwa seorang dosen tidak boleh mengajar dua kelas pada waktu yang sama, atau kapasitas ruangan tidak boleh melebihi jumlah maksimum mahasiswa yang diizinkan.

Sebagai contoh, dalam model *Integer Programming* (IP), *constraints* sering digunakan untuk memastikan bahwa jadwal atau solusi yang dihasilkan tidak memiliki konflik. Salah satu jenis *constraint* yang umum adalah *uniqueness constraint*, yang memastikan tidak ada konflik dalam jadwal. Misalnya, setiap anggota staf pengajar hanya boleh mengajar satu mata kuliah, satu kelompok siswa, dan satu ruangan pada waktu yang sama. *Constraint* ini dapat dinyatakan sebagai berikut: $\forall i \in I, \forall j \in J, \forall l \in L_i, \sum_{k \in K_l} \sum_{m \in M_{kl}} \sum_{n \in N_{mk}} x_{i,j,k,l,m,n} \leq 1,$ di mana i adalah anggota staf pengajar, j adalah kelompok siswa, l adalah ruangan, dan variabel keputusan $(x_{i,j,k,l,m,n})$ menunjukkan apakah kombinasi tertentu dari pengajar, siswa, dan ruangan sedang digunakan (Daskalaki et al., 2004).

Constraints berfungsi untuk menyempitkan ruang solusi, sehingga hanya solusi yang memenuhi semua kendala yang valid dan dipertimbangkan sebagai solusi potensial. Dalam banyak kasus, *constraints* dapat membuat masalah optimasi lebih sulit karena membatasi kebebasan dalam memilih solusi, tetapi hal ini diperlukan untuk memastikan bahwa hasil optimasi dapat diterapkan secara praktis.

2.2.3. *Branch-and-Cut*

Branch-and-cut adalah metode algoritmik yang efektif digunakan untuk menyelesaikan masalah optimasi, terutama yang melibatkan variabel bilangan bulat dan kendala kompleks. Metode ini menggabungkan pendekatan *branch-and-bound* dan *cutting planes* (J. Mitchell, 2011).

Metode *branch-and-bound* adalah teknik algoritma yang umum digunakan untuk memecahkan masalah pemrograman diskrit, seperti pemrograman integer dan optimasi kombinatorial (Mitten, 1970). Metode ini bekerja dengan membagi ruang solusi menjadi sub-masalah yang lebih kecil (*branching*) dan menghitung batas atas dan bawah (*bounding)* untuk mengevaluasi dan mempersempit ruang solusi. Pendekatan ini bertujuan untuk mengidentifikasi solusi optimal secara efisien dengan mengesampingkan sub-masalah yang tidak memenuhi syarat. Dengan menggunakan teknik ini, sub-masalah yang tidak dapat memberikan solusi yang lebih baik dari yang saat ini terbaik dapat dieliminasi, sehingga memungkinkan pencarian solusi optimal secara efisien tanpa harus melakukan pencarian secara menyeluruh. Pendekatan ini sering diterapkan sebagai teknik yang efisien untuk menangani masalah optimasi yang kompleks (Boyd & Mattingley, 2010).

Cutting planes adalah teknik optimasi yang digunakan untuk menyelesaikan berbagai masalah dalam program linear dan integer. Teknik ini bekerja dengan secara bertahap memperbaiki solusi dari himpunan *feasible* atau fungsi objektif melalui penggunaan ketaksamaan linear yang disebut *cutting planes*. Teknik ini

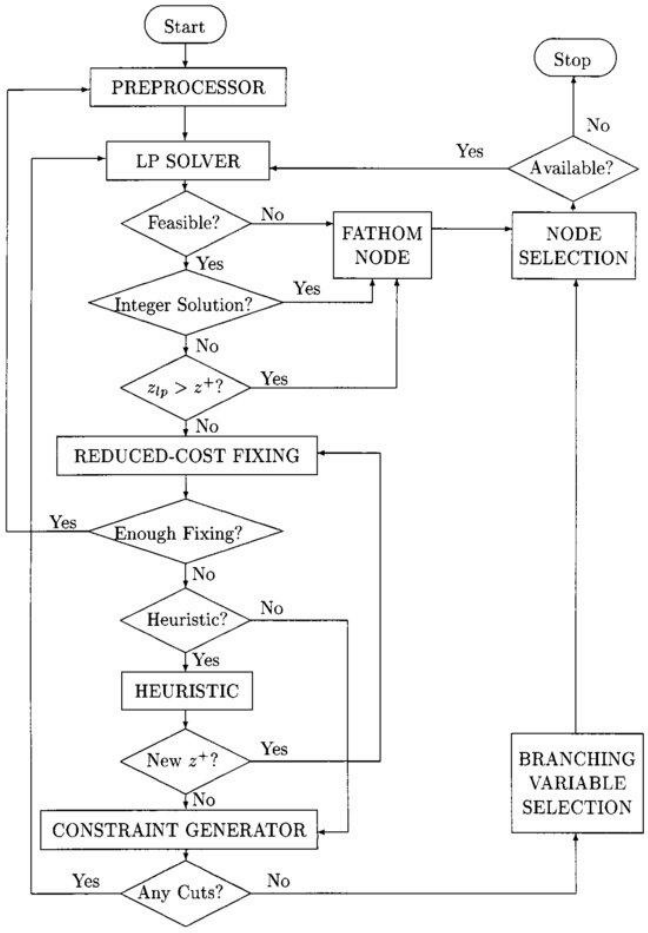
bekerja dengan menambahkan kendala tambahan, yang dikenal sebagai *cutting planes*, ke dalam relaksasi linier dari masalah bilangan bulat. Relaksasi linier ini dipecahkan terlebih dahulu, namun sering menghasilkan solusi yang tidak memenuhi syarat bilangan bulat. Dengan menambahkan *cutting planes*, bagian dari ruang solusi yang mengandung solusi tidak valid dipotong, sehingga mendekatkan solusi ke hasil bilangan bulat (Ceria et al., 1998).

Dalam prosesnya, *branch-and-cut* membagi ruang solusi menjadi sub-masalah yang lebih kecil (*branching*) sambil menambahkan ketaksamaan linear (*cutting planes*) untuk mempersempit ruang solusi yang perlu dieksplorasi. Kombinasi dari kedua teknik ini memungkinkan *branch-and-cut* untuk secara efisien mengidentifikasi solusi optimal dengan memfokuskan pencarian hanya pada area yang relevan dalam ruang solusi.

Dalam proses *branching*, masalah awal dipecah menjadi beberapa sub-masalah dengan membuat cabang berdasarkan pengaturan variabel tertentu, terutama yang berhubungan dengan kendala komplementaritas. Sementara itu, *cutting planes* adalah kendala tambahan yang ditambahkan untuk menghilangkan solusi yang tidak valid tanpa mempengaruhi solusi optimal.

Dalam jurnal "*Solving Linear Programs with Complementarity Constraints using Branch-and-Cu*t", metode ini dijelaskan sebagai pendekatan yang efisien untuk menangani LPCC dengan memanfaatkan kemampuan *branch-and-cut* dalam menangani kombinasi aspek linier dan komplementaritas (Yu et al., 2019). Jurnal ini juga menjelaskan bagaimana pengenalan kendala komplementaritas secara bertahap melalui proses *branching* dan penambahan *cutting planes* memungkinkan penyempitan ruang solusi hingga solusi optimal tercapai.

Dalam penelitian ini, *solver* yang digunakan menerapkan prinsip *branch-and-cut* karena menggabungkan pendekatan *branch-and-bound* dan *cutting planes*, yang secara efektif mengintegrasikan eksplorasi ruang solusi dengan *branching* dan perbaikan solusi melalui *cutting planes*. *Flowchart* implementasi metode *Branch-and-cut* pada kasus penjadwalan dapat dilihat pada gambar 2 berikut.



[a] z_{lp} = LP objective value; z^+ = objective value of best integer solution

Gambar 2. *Flowchart Branch-and-Cut (Bixby & Lee, 1996).*

2.2.4. *Feasibility Problem*

Feasibility problem atau masalah keterlaksanaan adalah jenis permasalahan yang bertujuan untuk menemukan setidaknya satu solusi yang memenuhi seluruh kendala atau kendala (*constraint*) yang diberikan, tanpa mempertimbangkan fungsi objektif yang harus dioptimalkan (Censor & Zaknoon, 2018).

Dalam sebuah model optimasi *Integer Linear Programming* (ILP), *feasibility* dari setiap *constraint* maupun keseluruhan *constraint* dapat diuji menggunakan *Helly's Theorem* dan *Monotonicity*. Berikut adalah penjelasan mengenai *Helly's Theorem* dan *Monotonicity*.

2.2.4.1. *Helly's Theorem* (Teorema Helly)

Teorema Helly adalah salah satu hasil fundamental dalam geometri konveks yang menyatakan bahwa jika kita memiliki sekumpulan himpunan konveks dalam ruang Ed (ruang berdimensi d), dan setiap d+1 himpunan dalam kumpulan tersebut memiliki setidaknya satu titik yang sama, maka seluruh himpunan juga memiliki satu titik yang sama (ada titik yang berada di dalam semua himpunan tersebut) (Amenta, 1993).

Dalam konteks *Integer Linear Programming* (ILP), Teorema Helly dapat digunakan untuk membuktikan bahwa:

a. Jika sekumpulan *constraint* linier *feasible* (ada solusi),
b. Maka *subset* dari *constraint* tersebut juga *feasible*, selama irisan non-kosong tetap terjaga.

Hal ini berguna untuk menguji model ILP, karena kita bisa membuktikan bahwa menambahkan *constraint* tidak akan tiba-tiba membuat sistem menjadi *infeasible* (tidak ada solusi) selama masih memenuhi properti irisan dari Teorema Helly.

2.2.4.2. *Monotonicity* (Monotonisitas)

Monotonicity dalam potongan program mengacu pada program dengan perilaku yang tidak meningkat atau tidak menurun, yang dapat dianalisis secara tepat menggunakan interpretasi abstrak (Campion et al., 2024).

Dalam konteks *Integer Linear Programming* (ILP), *monotonicity* dapat diartikan sebagai sifat tidak berkurangnya atau tidak bertambahnya *feasibility* ketika *constraint* dikurangi. Dalam ILP, jika suatu sistem *constraint feasible*, maka penghapusan beberapa *constraint* tidak akan membuatnya menjadi *infeasible*, karena himpunan solusi yang memenuhi *constraint* menjadi lebih besar atau tetap sama.

2.2.5. Analisis Kompleksitas Polinomial

Teori kompleksitas polinomial adalah studi tentang *computability* waktu polinomial, menjembatani kesenjangan antara pendekatan abstrak dan konkret terhadap algoritma, serta menyediakan landasan bagi kompleksitas komputasi masalah praktis (Ko, 1991).

Pendekatan ini menjadi fundamental dalam menganalisis efisiensi algoritma, terutama dalam menentukan apakah suatu masalah dapat diselesaikan dalam waktu yang wajar seiring dengan bertambahnya ukuran masukan. Dalam konteks *Integer Linear Programming* (ILP), kompleksitas polinomial berperan dalam mengevaluasi sejauh mana suatu formulasi model dapat diselesaikan dalam skala besar tanpa mengalami lonjakan eksponensial dalam waktu komputasi.

Meskipun ILP secara umum tergolong NP-Hard, pengembangan algoritma berbasis relaksasi linier, *branch-and-bound*, dan *cutting plane* telah memungkinkan penyelesaian masalah dengan ukuran menengah hingga besar dalam waktu yang lebih efisien. Studi kompleksitas ini juga menjadi dasar dalam memilih metode optimasi yang tepat, seperti apakah suatu masalah lebih cocok diselesaikan dengan formulasi polinomial yang lebih sederhana atau memerlukan pendekatan heuristik untuk mendapatkan solusi mendekati optimal dalam waktu yang lebih singkat.

Dalam analisis algoritma, kompleksitas waktu diekspresikan menggunakan notasi Big-O, yang menyatakan batas atas pertumbuhan waktu eksekusi sebagai fungsi dari ukuran masukan n. Jika waktu eksekusi suatu algoritma dapat dinyatakan sebagai fungsi polinomial, maka kompleksitasnya termasuk dalam kelas P (*Polynomial Time Complexity*), yang berarti dapat diselesaikan dalam waktu yang dapat diprediksi secara polinomial terhadap ukuran masukan. Secara empiris, model polinomial sering digunakan untuk memodelkan waktu eksekusi algoritma berdasarkan hasil pengujian pada berbagai ukuran masukan. Model ini dinyatakan dalam bentuk: $T(n) = k \cdot n^p$.

Dalam model polinomial yang digunakan untuk menganalisis waktu eksekusi algoritma, $T(n)$ merepresentasikan waktu eksekusi sebagai fungsi dari ukuran masukan n. Nilai ini menunjukkan bagaimana perubahan jumlah data yang diproses memengaruhi durasi penyelesaian suatu masalah. Konstanta k dalam model tersebut bergantung pada berbagai faktor, seperti arsitektur perangkat keras yang digunakan, efisiensi implementasi algoritma, serta *overhead* sistem yang terjadi selama proses komputasi. Sementara itu, eksponen p menentukan laju pertumbuhan waktu eksekusi terhadap peningkatan ukuran masukan, yang mencerminkan tingkat kompleksitas algoritma dalam menyelesaikan perhitungan.

2.2.6. *Web Service*

Web service adalah Sebuah aplikasi perangkat lunak yang diidentifikasi melalui URI, dengan antarmuka dan mekanisme komunikasinya dapat didefinisikan, dijelaskan, dan ditemukan dalam bentuk artefak XML (Ferris & Farrell, 2003). *Web service* berkomunikasi dengan layanan lain menggunakan pesan berbasis XML yang dikirimkan melalui protokol web seperti HTTP atau SOAP (Tsalgatidou et al., 2002).

Dalam penelitian ini, *web service* yang akan dikembangkan berfungsi untuk men-*generate* jadwal secara otomatis. Layanan ini akan mengolah data terkait ketersediaan dosen, ruang kelas, dan *slot* waktu yang tersedia, serta menerapkan berbagai aturan dan kendala penjadwalan yang berlaku. *Web service* ini akan berjalan secara terpisah namun tetap terhubung dengan komponen lain dalam sistem melalui mekanisme pertukaran data berbasis API. Layanan ini dapat menerima permintaan (*request*) dalam format XML atau JSON, memproses permintaan tersebut, dan memberikan respons dalam format yang sama.

Pendekatan ini memungkinkan pengembangan yang modular, di mana setiap layanan web dapat diuji, di-*deploy*, dan dikelola secara terpisah dari komponen lain dalam sistem. Dengan menggunakan *web service*, penjadwalan otomatis dapat diintegrasikan dengan sistem lain yang memerlukan data terkait, serta

memberikan fleksibilitas dalam hal komunikasi dan keterhubungan antar sistem yang berbeda.

2.2.7. Python

Python adalah bahasa pemrograman tingkat tinggi yang diinterpretasikan, berorientasi objek, dan memiliki semantik dinamis (Python Software Foundation, 2024). Dikembangkan pertama kali oleh Guido van Rossum dan dirilis pada tahun 1991, Python dikenal karena sintaks-nya yang sederhana dan mudah dibaca, menjadikannya pilihan ideal untuk pemula sekaligus pengembang berpengalaman. Python memiliki ekosistem *library* dan *framework* yang sangat kaya, seperti Django dan Flask untuk pengembangan web, NumPy dan Pandas untuk analisis data, serta TensorFlow dan PyTorch untuk pembelajaran mesin.

Python juga digunakan dalam berbagai aplikasi ilmiah dan rekayasa, berkat kemampuannya dalam menangani perhitungan matematis yang kompleks dan analisis data skala besar. Selain itu, Python populer dalam pengembangan otomatisasi sistem dan skrip, memungkinkan pengembang untuk menulis kode yang efisien dengan sedikit *boilerplate*.

Dalam konteks penjadwalan, Python juga banyak digunakan untuk menyelesaikan masalah penjadwalan menggunakan *Integer Linear Programming* (ILP) (Santos & Toffolo, 2020). Paket seperti Python-MIP dan PuLP memudahkan pengembangan model ILP untuk mengoptimalkan penjadwalan, baik untuk penjadwalan kelas, penjadwalan tenaga kerja, atau bahkan penjadwalan produksi di industri. Python memungkinkan integrasi dengan *solver* canggih seperti COIN-OR dan Gurobi, yang memanfaatkan algoritma branch-and-cut untuk menyelesaikan masalah penjadwalan yang kompleks secara efisien. Penggunaan ILP dalam Python memberikan solusi optimal untuk berbagai jenis penjadwalan, memastikan alokasi sumber daya seperti waktu, ruang, dan tenaga kerja secara optimal, dengan memperhitungkan berbagai kendala yang relevan.

2.2.8. FastAPI

FastAPI adalah *framework* web modern berbasis Python yang memungkinkan pengembangan aplikasi API dengan performa tinggi dan efisiensi tinggi (Chen, 2023). *Framework* ini dirancang untuk membuat aplikasi yang cepat dan mudah dipelihara dengan pendekatan asinkron, serta mendukung tipe anotasi Python untuk validasi *input* otomatis. Dikembangkan oleh Sebastián Ramírez, FastAPI dikenal karena kemampuannya untuk membangun aplikasi API dengan cepat, serta menawarkan dukungan bawaan untuk OpenAPI dan JSON Schema.

Dengan arsitektur asinkron yang dioptimalkan untuk kinerja, FastAPI mempermudah pengembangan API yang cepat, *scalable*, dan aman. *Framework* ini mampu menangani sejumlah besar permintaan secara efisien, menjadikannya pilihan ideal untuk aplikasi skala besar. Selain itu, FastAPI secara otomatis menghasilkan dokumentasi interaktif untuk API menggunakan Swagger UI dan Redoc, yang sangat memudahkan pengembang dalam menguji dan memelihara API.

Dalam konteks penelitian ini, FastAPI akan digunakan sebagai kerangka kerja untuk *web service* yang khusus menangani pembuatan penjadwalan otomatis. Dengan memanfaatkan keunggulan kinerja dan kemudahan penggunaan yang ditawarkan oleh FastAPI, *web service* ini diharapkan dapat mengelola dan menghasilkan jadwal secara efisien, memastikan respons yang cepat terhadap permintaan pengguna dan mendukung integrasi yang baik dengan komponen lain dalam sistem penjadwalan.

2.2.9. PuLP

PuLP adalah sebuah kerangka pemrograman linier dan pemrograman bilangan bulat campuran dalam Python (pulp documentation team, 2009). PuLP dilisensikan di bawah lisensi BSD yang dimodifikasi. Tujuan dari PuLP adalah untuk memungkinkan seorang praktisi atau *programmer* Riset Operasional (OR)

untuk mengungkapkan model Pemrograman Linier (LP) dan Pemrograman Bilangan Bulat (IP) dalam Python dengan cara yang mirip dengan notasi matematis konvensional. PuLP juga akan menyelesaikan masalah-masalah ini menggunakan berbagai pemecah LP, baik yang gratis maupun yang berbayar. Pulp memodelkan LP dengan cara yang alami dan sesuai dengan Python (S. Mitchell, 2009).

PuLP dikembangkan untuk memudahkan pemrograman dan pemecahan masalah optimasi linear dengan cara yang sederhana dan intuitif. *Library* ini mendukung berbagai *solver* optimasi, seperti CBC, GLPK, dan Gurobi, serta menyediakan antarmuka pemrograman yang fleksibel untuk membuat, memecahkan, dan menganalisis model optimasi.

PuLP memungkinkan pengembang untuk mendefinisikan fungsi tujuan (*objective function*), serta kendala (*constraints*) dalam bentuk persamaan atau pertidaksamaan linear. Dalam konteks *Integer Linear Programming* (ILP), PuLP memberikan kemampuan untuk menangani masalah optimasi di mana beberapa atau semua variabel harus berupa bilangan bulat, yang sangat berguna dalam berbagai aplikasi seperti penjadwalan, alokasi sumber daya, dan logistik.

Berikut adalah contoh penerapan PuLP untuk studi kasus penjadwalan perkuliahan sederhana:

1. Mendeklarasikan Model

   Untuk Mendeklarasikan model, hal yang pertama kali harus dilakukan adalah mengimpor *library* PuLP, lalu mendeklarasikan variabel model dengan menggunakan pulp.LpProblem seperti gambar 3 di bawah ini.

```
# Membuat masalah ILP
import pulp
model = pulp.LpProblem("Class_Scheduling_Problem", pulp.LpMinimize)
```

Gambar 3. Mendeklarasikan Model

2. Mendefinisikan Variabel Keputusan (*Decision Variables*)

Untuk mendefinisikan variabel keputusan dapat menggunakan *syntax* pulp.LpVariable seperti pada gambar 4 di bawah ini.

```
# Variabel biner
X_dhsp = pulp.LpVariable.dicts("X", (lecturers, days, sessions, rooms), cat='Binary')
Y_ahsp = pulp.LpVariable.dicts("Y", (assistants, days, sessions, rooms), cat='Binary')
K_khs = pulp.LpVariable.dicts("K", (classes, days, sessions), cat='Binary')
R_rhs = pulp.LpVariable.dicts("R", (rooms, days, sessions), cat='Binary')
```

Gambar 4. Mendefinisikan variabel keputusan

3. Mendefinisikan Fungsi Objektif (*Objective Function*)

Untuk mendefinisikan fungsi objektif dapat menggunakan *syntax* pulp.LpSum seperti pada gambar 5 di bawah ini.

```
# Objective Function: meminimalkan penggunaan ruangan yang tidak efisien
model += pulp.lpSum([R_rhs[r][h][s] for r in rooms for h in days for s in sessions])
```

Gambar 5. Mendefinisikan fungsi objektif

4. Mendefinisikan Kendala (*Constraint*)

Pendeklarasian kendala atau *constraints* dilakukan dengan cara menggunakan operator += terhadap variabel model, lalu menuliskan persamaan atau pertidaksamaan linearnya dalam bahasa pemrograman Python seperti pada gambar 6 di bawah.

```
# 1. Dosen tidak boleh mengajar lebih dari satu kelas di waktu yang sama
for d in lecturers:
    for h in days:
        for s in sessions:
            model += pulp.lpSum([X_dhsp[d][h][s][r] for r in rooms]) <= 1

# 2. Asisten dosen tidak boleh mengajar lebih dari satu kelas di waktu yang sama
for a in assistants:
    for h in days:
        for s in sessions:
            model += pulp.lpSum([Y_ahsp[a][h][s][r] for r in rooms]) <= 1

# 3. Asisten dosen tidak boleh mengajar pada waktu yang sama dengan jadwal perkuliahannya
# (asumsi kelas asisten dosen diketahui dalam variabel 'assistant_classes')
assistant_classes = {'Assistant1': 'Class1', 'Assistant2': 'Class2', 'Assistant3': 'Class3'}
for a in assistants:
    for h in days:
        for s in sessions:
            model += Y_ahsp[a][h][s][rooms[0]] <= 1 - K_khs[assistant_classes[a]][h][s]
```

Gambar 6. Mendefinisikan kendala (*constraint*).

2.2.10. API (*Application Programming Interface*)

API (*Application Programming Interface*) adalah antarmuka yang memungkinkan berbagai aplikasi atau layanan perangkat lunak untuk saling berkomunikasi (Efuntade & Efuntade, 2023). Dengan API, sebuah sistem dapat menyediakan akses terbatas ke fungsionalitasnya, sehingga aplikasi lain dapat menggunakan fitur atau data tersebut tanpa perlu mengetahui cara kerja internal sistem tersebut. API berperan penting dalam integrasi antar sistem, di mana pertukaran data dan layanan menjadi lebih efisien dan terstruktur. API memudahkan pengembang untuk membangun aplikasi yang dapat berkolaborasi dengan layanan pihak ketiga atau sistem eksternal dengan aman dan terkontrol.

REST API (*Representational State Transfer*) adalah salah satu bentuk API yang mengikuti prinsip REST, yaitu gaya arsitektur yang menggunakan protokol HTTP untuk mengakses dan mengelola sumber daya di web. REST API memanfaatkan metode HTTP standar seperti GET, POST, PUT, dan DELETE untuk berinteraksi dengan data (Segura et al., 2018). Setiap sumber daya di REST API diidentifikasi oleh URL, dan data biasanya dikirim atau diterima dalam format ringan seperti JSON atau XML. Kesederhanaan, fleksibilitas, serta kemampuannya untuk diintegrasikan dengan berbagai platform membuat REST API menjadi pilihan utama dalam pengembangan aplikasi web dan *mobile*.

BAB III
METODOLOGI PENELITIAN

3.1. Waktu dan Tempat Penelitian

Penelitian ini dilakukan di Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung yang beralamat di jalan Prof. Dr. Ir. Brojonegoro No.1, Kelurahan Gedung Meneng, Kecamatan Rajabasa, Kota Bandar Lampung. Penelitian ini dilakukan pada Semester Ganjil dan Genap TA 2024/2025.

3.2. Perangkat Penelitian

Penelitian ini menggunakan dua jenis, yaitu perangkat keras dan perangkat lunak. Berikut merupakan spesifikasi alat yang digunakan selama penelitian.

3.2.1. Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan dalam penelitian ini adalah:

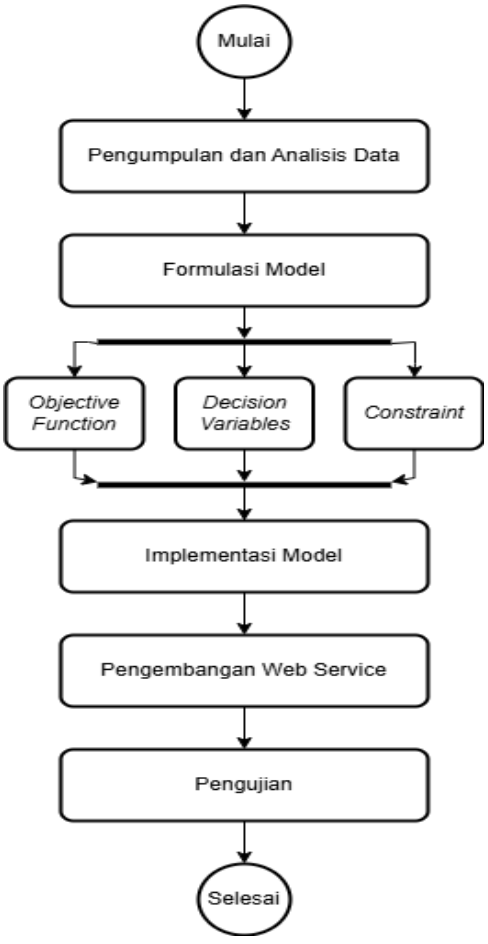
a. Sistem Operasi Windows 11 Home 64-bit
b. Visual Studio Code
c. Python 3.11.11
d. Postman
e. GitHub
f. Web Browser Google Chrome dan Microsoft Edge

3.2.2. Perangkat Keras (*Hardware*)

Perangkat yang digunakan dalam penelitian ini merupakan laptop Acer Nitro 5 AN515-57 yang diproduksi oleh Acer, dilengkapi dengan prosesor Intel Core i5-11400H, GPU NVIDIA GeForce RTX 3050 dengan kapasitas 4GB, serta memori utama sebesar 16GB DDR4 berkecepatan 2933 MT/s. Media penyimpanannya menggunakan SSD M.2 NVMe Gen3x4 berkapasitas 512GB.

3.3. Tahapan Penelitian

Penelitian ini dilakukan melalui lima tahapan utama, yaitu pengumpulan data, formulasi model, implementasi model, pengembangan *web service*, dan pengujian. Alur metode penelitian ini dapat dilihat pada gambar 7 di bawah.



Gambar 7. Tahapan Penelitian

3.3.1. Pengumpulan Data

Tahapan pengumpulan data bertujuan untuk mengumpulkan informasi yang diperlukan guna mendukung proses formulasi model dan implementasi solusi penjadwalan perkuliahan menggunakan *Integer Linear Programming* (ILP). Data yang akan digunakan dalam penelitian ini didapatkan dari dua sumber utama, yaitu wawancara dan observasi.

3.3.1.1. Wawancara

Wawancara dilakukan dengan para pemangku kepentingan yang terlibat langsung dalam proses penyusunan jadwal perkuliahan. Dalam konteks penelitian ini, wawancara dilakukan dengan Sekretaris Jurusan dan Badan Khusus yang bertanggung jawab atas penyusunan jadwal di Jurusan Ilmu Komputer Universitas Lampung. Hasil wawancara ini bertujuan untuk mengidentifikasi *constraint* (kendala) yang akan digunakan dalam model *Integer Linear Programming* (ILP) untuk penjadwalan perkuliahan, mencakup aturan umum serta aturan spesifik yang harus dipenuhi dalam proses optimasi penjadwalan tersebut.

Hasil wawancara mengungkap beberapa poin penting terkait kendala dan kebutuhan dalam pengembangan sistem penjadwalan otomatis di Jurusan Ilmu Komputer. Pertama, keterbatasan jumlah ruangan menjadi masalah utama, terutama saat terjadi peningkatan jumlah mahasiswa, sehingga sebagian kelas harus dilaksanakan secara daring untuk mengatasi keterbatasan kapasitas ruangan. Kedua, proses penjadwalan semakin kompleks dengan adanya kombinasi peran dosen penanggung jawab (PJ), dosen anggota, dan asisten dosen, di mana jadwal mereka tidak boleh tumpang tindih atau berada pada sesi yang sama untuk mata kuliah yang berbeda. Hal ini memerlukan pengaturan tambahan untuk memastikan kepatuhan terhadap aturan internal jurusan. Terakhir, wawancara juga membantu mengidentifikasi berbagai kendala, baik aturan umum maupun khusus, yang harus

dipenuhi dalam sistem penjadwalan untuk menciptakan solusi yang efektif dan sesuai dengan kebutuhan akademik.

Informasi dari wawancara ini akan menjadi dasar perancangan sistem penjadwalan otomatis, khususnya dalam menentukan *constraint* yang harus dipenuhi untuk menghasilkan jadwal yang sesuai dengan kebutuhan dan kondisi aktual di jurusan.

3.3.1.2. Observasi

Pada tahap ini, data dari Sistem Informasi Akademik (SIAKAD) Universitas Lampung akan dikumpulkan secara manual menggunakan akun mahasiswa dari program studi S1 Ilmu Komputer dan D3 Manajemen Informatika. Informasi jadwal perkuliahan yang mencakup kurikulum, periode akademik, mata kuliah, program studi pengampu, nama kelas, dosen pengajar, jadwal mingguan, serta kapasitas kelas akan diambil dari menu "Perkuliahan" > "Kelas & Jadwal" > "Kelas Kuliah," yang dapat diakses melalui URL https://siakadu.unila.ac.id/siakad/list_kelas. Pada menu tersebut, filter "Periode Akademik" akan diatur pada nilai 2023 Ganjil dan 2023 Genap, serta filter "Kurikulum" akan diatur pada kurikulum 2020. Data jadwal yang melibatkan kelas MBKM tidak akan digunakan. Informasi mengenai semester setiap kelas akan diambil melalui menu "Perkuliahan" > "MK & Kurikulum" > "Kurikulum Prodi" yang tersedia pada URL https://siakadu.unila.ac.id/siakad/set_kurikulum.

a. Data Dosen

Tabel 2. Data Dosen Pengampu Mata Kuliah.

Jenis Dosen	Jumlah
Dosen Jurusan Ilmu Komputer	22
Dosen Lain	16
Total	38

Data jumlah dosen yang mengampu mata kuliah di Jurusan Ilmu Komputer telah dikumpulkan dan disajikan dalam Tabel 2. Di tabel ini, dapat dilihat perincian jumlah dosen berdasarkan kategori yang ada, dengan total keseluruhan 38 dosen. Untuk detail data, termasuk nama dan informasi lain dari masing-masing dosen, dapat dilihat pada Lampiran 2.

b. Data Mata Kuliah

Tabel 3. Data Mata Kuliah Semester Ganjil.

Program Studi	Semester	Kelompok	Jumlah Mata Kuliah
S1 Ilmu Komputer	1	Wajib	13
		Peminatan	0
	3	Wajib	8
		Peminatan	4
	5	Wajib	6
		Peminatan	5
	7	Wajib	1
		Peminatan	7
D3 Manajemen Informatika	1	Wajib	12
		Peminatan	0
	3	Wajib	9
		Peminatan	0
	5	Wajib	6
		Peminatan	0
Total			**71**

Data mengenai jumlah mata kuliah yang diselenggarakan pada semester ganjil ditampilkan dalam Tabel 3. Tabel ini mengklasifikasikan mata kuliah berdasarkan program studi (S1 Ilmu Komputer dan D3 Manajemen Informatika), semester (1, 3, 5, dan 7), serta kelompok mata kuliah, yaitu mata kuliah wajib dan mata kuliah peminatan. Dari tabel tersebut, terlihat total 71 mata kuliah ditawarkan pada semester ganjil. Perincian lengkap mengenai setiap mata kuliah, termasuk kode mata kuliah, nama mata kuliah, jumlah SKS, dan nama dosen pengampu, disajikan dalam Lampiran 3 dan 5.

Tabel 4. Data Mata Kuliah Semester Genap.

Program Studi	Semester	Kelompok	Jumlah Mata Kuliah
S1 Ilmu Komputer	2	Wajib	9
		Peminatan	0
	4	Wajib	6
		Peminatan	4
	6	Wajib	1
		Peminatan	7
D3 Manajemen Informatika	2	Wajib	9
		Peminatan	0
	4	Wajib	8
		Peminatan	0
Total			**44**

Data mengenai jumlah mata kuliah yang diselenggarakan pada semester genap ditampilkan dalam Tabel 4. Tabel ini mengklasifikasikan mata kuliah berdasarkan program studi (S1 Ilmu Komputer dan D3 Manajemen Informatika), semester (2, 4, dan 6), serta kelompok mata kuliah, yaitu mata kuliah wajib dan mata kuliah peminatan. Dari tabel tersebut, terlihat total 44 mata kuliah ditawarkan pada semester genap. Perincian lengkap mengenai setiap mata kuliah, termasuk kode mata kuliah, nama mata kuliah, jumlah SKS, dan nama dosen pengampu, disajikan dalam Lampiran 4 dan 6.

c. Data Kelas

Tabel 5. Data Kelas Semester Ganjil.

Program Studi	Kelompok	Jumlah Kelas
S1 Ilmu Komputer	Wajib	88
	Peminatan	25
D3 Manajemen Informatika	Wajib	28
	Peminatan	0
Total		**141**

Data jumlah kelas yang diselenggarakan pada semester ganjil ditampilkan dalam Tabel 5. Tabel ini mengklasifikasikan jumlah kelas berdasarkan program studi (S1 Ilmu Komputer dan D3 Manajemen Informatika) dan kelompok mata kuliah, yaitu wajib dan peminatan. Total kelas yang dibuka pada semester ganjil berjumlah 141 kelas. Informasi detail mengenai setiap kelas, termasuk nama kelas, kapasitas kelas, dan informasi lain yang relevan, disajikan dalam Lampiran 7 dan 9.

Tabel 6. Data Kelas Semester Genap.

Program Studi	Kelompok	Jumlah Kelas
S1 Ilmu Komputer	Wajib	59
	Peminatan	23
D3 Manajemen Informatika	Wajib	18
	Peminatan	0
Total		**100**

Data jumlah kelas yang diselenggarakan pada semester genap ditampilkan dalam Tabel 6. Tabel ini mengklasifikasikan jumlah kelas berdasarkan program studi (S1 Ilmu Komputer dan D3 Manajemen Informatika) dan kelompok mata kuliah, yaitu wajib dan peminatan. Total kelas yang dibuka pada semester genap berjumlah 100 kelas. Informasi detail mengenai setiap kelas, termasuk nama kelas, kapasitas kelas, dan informasi lain yang relevan, disajikan dalam Lampiran 8 dan 10.

d. Data Ruangan

Tabel 7. Data Ruangan.

Ruangan	Kapasitas
GIK L1 A	50
GIK L1 B	50
GIK L1 C	100
GIK L2	100
LAB R1	25
LAB R2	25
LAB R3	25
LAB R4	20
LAB RPL	20
MIPA T L1 A	80
MIPA T L1 B	80

Data lengkap mengenai kapasitas ruangan yang digunakan untuk kegiatan perkuliahan disajikan dalam Tabel 7. Tabel ini mencantumkan nama ruangan dan kapasitas masing-masing ruangan yang tersedia di Jurusan Ilmu Komputer.

e. Data Sesi Perkuliahan

Tabel 8. Data Sesi Perkuliahan.

Sesi	Mulai	Selesai
1	7:30	9:10
2	9:20	11:00
3	11:10	12:50
4	13:30	15:10
5	15:30	17:00

Data sesi perkuliahan yang disajikan dalam Tabel 8 memberikan informasi mengenai alokasi waktu untuk setiap sesi perkuliahan yang berlaku di Jurusan Ilmu Komputer T.A. 2023/2024.

3.3.2. Formulasi Model

Pada tahap formulasi model, fokus utama adalah mengubah data primer dan sekunder yang telah dikumpulkan menjadi bentuk persamaan atau pertidaksamaan linear. Persamaan ini kemudian akan diolah menjadi struktur yang dapat dipahami dan diproses oleh program Python. Proses ini melibatkan beberapa langkah penting untuk memastikan data diubah menjadi model matematis yang siap digunakan dalam implementasi kode Python.

Setelah melakukan pengumpulan data, ditemukan beberapa permasalahan dalam penjadwalan perkuliahan yang perlu diperhatikan. Pertama, diperlukan upaya untuk meminimalkan penggunaan ruangan dengan kapasitas yang jauh lebih besar dari jumlah mahasiswa pada kelas tertentu serta mengurangi sesi berturut-turut yang dijadwalkan untuk dosen. Setiap pertemuan harus dijadwalkan tepat satu kali, dan tiap ruangan tidak boleh digunakan oleh lebih dari satu kelas pada waktu yang sama. Untuk memastikan efisiensi penggunaan ruangan, kapasitas ruangan harus lebih besar atau sama dengan jumlah mahasiswa pada suatu kelas. Selain itu, tiap dosen hanya diperbolehkan mengajar maksimal tiga sesi dalam sehari.

Khusus pada hari Jumat, perkuliahan hanya dapat dijadwalkan pada sesi 1, 2, 4, dan 5. Untuk kelas dengan nama dan semester yang sama, perkuliahan hanya boleh dijadwalkan sekali dalam hari dan sesi yang sama. Sementara itu, kelas praktikum harus dijadwalkan di dua ruangan yang berbeda namun tetap pada hari dan sesi yang sama. Selain itu, tiap dosen tidak diperbolehkan mengajar lebih dari satu kelas dalam waktu yang bersamaan.

Dari sisi penggunaan ruangan, terdapat aturan khusus di mana ruang laboratorium hanya diperbolehkan untuk pertemuan praktikum, sedangkan ruang kelas hanya boleh digunakan untuk pertemuan teori dan responsi. Hal ini bertujuan untuk memastikan pemanfaatan fasilitas yang lebih optimal sesuai dengan kebutuhan masing-masing jenis perkuliahan.

Permasalahan ini kemudian akan dirumuskan sebagai masalah optimasi menggunakan *Integer Linear Programming* (ILP). Dalam perumusan ini, setiap

permasalahan akan dikonversi menjadi variabel keputusan sebagai variabel utama, fungsi objektif yang bertujuan untuk memaksimalkan atau meminimalkan suatu nilai, serta kendala yang harus dipenuhi.

3.3.2.1. Variabel Keputusan (*Decision Variables*)

Pada tahap ini, variabel keputusan (*decision variables*) yang akan digunakan dalam model ILP diidentifikasi. Variabel ini digunakan untuk memformulasikan persamaan linear yang menggambarkan kondisi dalam penjadwalan perkuliahan.

Data dari basis data yang akan diproses oleh model terdiri dari lima variabel utama, yaitu:

1. *classLecturers*: merupakan himpunan pasangan kelas dan dosen yang menjadi basis utama atau unit dalam setiap pertemuan, direpresentasikan dengan *i*.
2. *rooms*: merupakan himpunan ruangan yang tersedia dalam sebuah jurusan, direpresentasikan dengan *r*.
3. *days*: merupakan himpunan hari yang tersedia dalam satu minggu, direpresentasikan dengan *d*.
4. *sessions*: merupakan himpunan sesi perkuliahan yang tersedia dalam satu hari, direpresentasikan dengan *s*.
5. *lecturers*: merupakan himpunan dosen, direpresentasikan dengan *l*.

Berdasarkan permasalahan dalam model optimasi serta variabel-variabel yang harus diproses oleh model, maka dirumuskan dua variabel keputusan yang menjadi dasar utama dalam perumusan model.

a. Jadwal (x_*irds*): Variabel biner yang menunjukkan apakah pertemuan i dijadwalkan di ruangan r, pada hari d, di sesi s. Bernilai 1 jika iya dan 0 jika tidak. Variabel keputusan ini merupakan variabel keputusan utama yang akan menjadi unit untuk satu pertemuan di dalam penjadwalan, variabel keputusan ini juga yang nantinya akan disimpan ke dalam *database* sebagai

sebuah pertemuan. Persamaan untuk variabel keputusan ini adalah sebagai berikut.

$x_{i,r,d,s} = 1$ jika i di ruang r pada hari d di sesi s, 0 jika tidak

b. Dosen (y_*lds*): Variabel biner yang menunjukkan apakah dosen l ada di hari d pada sesi s. Bernilai 1 jika iya dan 0 jika tidak. Variabel keputusan ini merupakan variabel keputusan sekunder yang akan digunakan untuk mengoptimalkan jadwal mengajar dosen. Persamaan untuk variabel keputusan ini adalah sebagai berikut.

$y_{l,d,s} = 1$ jika dosen l di hari d pada sesi s, 0 jika tidak

3.3.2.2. Fungsi Objektif (*Objective Function*)

Objective function atau fungsi objektif dalam model *Integer Linear Programming* (ILP) digunakan untuk meminimalkan atau memaksimalkan suatu nilai yang diinginkan dalam penjadwalan perkuliahan. Fungsi tujuan ini dirancang untuk mengoptimalkan alokasi sumber daya sehingga proses perkuliahan berjalan lebih efektif dan efisien.

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, terdapat dua aspek yang tidak bersifat wajib tetapi perlu dioptimalkan, yaitu meminimalkan penggunaan ruangan dengan kapasitas yang jauh lebih besar dari jumlah mahasiswa dalam suatu kelas dan mengurangi sesi perkuliahan yang dijadwalkan secara berturut-turut untuk seorang dosen. Oleh karena itu, kedua aspek ini dimasukkan ke dalam fungsi objektif sebagai faktor yang tidak bersifat wajib tetapi tetap diperhitungkan dalam optimasi. Dengan demikian, fungsi objektif untuk model optimasi dirumuskan sebagai berikut.

Meminimalkan penggunaan ruangan dengan kapasitas yang jauh lebih besar dari jumlah mahasiswa pada kelas tertentu dan meminimalkan sesi yang berturut-turut untuk dosen. Di bawah ini adalah persamaan untuk fungsi objektif tersebut.

Minimize $\sum_{i,r,d,s} x_{i,r,d,s}(C_r - C_i) + \sum_{l,d,s} y_{l,d,s}$

3.3.2.3. Constraint

Constraint dalam ILP adalah kendala atau aturan yang harus dipenuhi dalam model penjadwalan. *Constraint* ini diformulasikan dalam bentuk persamaan atau pertidaksamaan linear. *Constraint* atau kendala dalam penelitian ini adalah sebagai berikut:

a. Setiap pertemuan harus dijadwalkan tepat satu kali.

$$\sum_{r,d,s} x_{i,r,d,s} = 1 \quad \forall i$$

b. Tiap ruangan tidak boleh digunakan oleh lebih dari satu kelas pada waktu yang sama.

$$\sum_{i} x_{i,r,d,s} \leq 1 \quad \forall r, d, s$$

c. Untuk menggunakan ruangan kelas, kapasitas ruangan harus lebih besar atau sama dengan kapasitas mahasiswa pada suatu kelas.

$$x_{i,r,d,s}\, C_i \leq C_{rK} \quad \forall i, r, d, s \; jika \; r = Kelas$$

d. Tiap dosen hanya boleh mengajar maksimal tiga sesi dalam sehari.

$$\sum_{i,r,s} x_{i,r,d,s} \leq 3 \quad \forall l, d \; jika \; i = Teori$$

e. Tidak boleh ada pertemuan pada sesi ketiga di hari Jumat.

$$x_{i,r,d,s} = 0 \quad \forall i, r \; jika \; d = \text{Jumat dan } s = 3$$

f. Untuk kelas teori dan responsi dengan nama kelas yang sama dan juga semester yang sama hanya boleh dijadwalkan tepat satu kali pada hari dan sesi yang sama.

$$\sum_{r} x_{i,r,d,s} = \sum_{r} x_{j,r,d,s} \quad \forall i,j \in \{\text{Bentrok}\}, d, s \; jika \; i \neq j$$

g. Untuk kelas praktikum dijadwalkan pada hari dan sesi yang sama.

$$\sum_{r} x_{ip,r,d,s} = \sum_{r} x_{jp,r,d,s} \;\; \forall ip, jp \in \{\text{Grup}\}, d, s$$

h. Untuk kelas praktikum dijadwalkan pada dua ruangan yang berbeda di hari dan sesi yang sama.

$$x_{ip,r,d,s} + x_{jp,r,d,s} \leq 1 \quad \forall ip, jp \in \{\text{Grup}\}, r, d, s$$

i. Untuk kelas praktikum yang dilakukan secara daring, maka kedua pertemuannya harus dilaksanakan di dalam kelas dengan tipe daring pula.

$x_{ip,r_1,d,s} + x_{jp,r_2,d,s} \leq 1 \; \forall ip, jp \in \{\text{Grup}\}$ r_1=Online, r_2≠Online, d, s

j. Untuk kelas praktikum yang dilakukan menggunakan ruangan lab, maka kedua pertemuannya harus dilaksanakan di dalam kelas dengan tipe ruangan lab pula.

$x_{ip,r_1,d,s} + x_{jp,r_2,d,s} \leq 1 \; \forall ip, jp \in \{\text{Grup}\}$ r_1=Lab, r_2≠Lab, d, s

k. Tiap dosen tidak boleh mengajar lebih dari satu kelas pada waktu yang sama.

$$\sum_{i,r} x_{i,r,d,s} \leq 1 \quad \forall l, d, s \text{ jika dosen dalam } i = l \text{ dan } i = Teori$$

l. Ruangan lab hanya dapat digunakan untuk pertemuan praktikum.

$x_{i,r,d,s} = 0 \quad \forall i, r$ jika r = Lab dan $i \neq$ Praktikum

m. Ruangan kelas hanya dapat digunakan untuk pertemuan teori dan responsi.

$x_{i,r,d,s} = 0 \quad \forall i, r$ jika r = Kelas dan $i \neq$ Teori atau Responsi

n. Pertemuan mata kuliah wajib teori tidak boleh dilakukan secara daring.

$x_{i,r,d,s} = 0 \quad \forall i, r$ jika r = Online dan i mata kuliah wajib

3.3.3. Implementasi Model

Dalam tahap implementasi ILP menggunakan PuLP, pertama-tama kita perlu membuat objek model dengan menggunakan fungsi LpProblem(), yang digunakan untuk mendefinisikan masalah optimasi yang akan diselesaikan, termasuk fungsi objektif dan kendalanya. Pada langkah ini, model akan dideklarasikan sebagai masalah minimalisasi, seperti dalam kasus penjadwalan kelas. Setelah itu, variabel keputusan dideklarasikan menggunakan LpVariable(), di mana variabel-variabel ini akan digunakan untuk menunjukkan keputusan biner, misalnya apakah dosen mengajar pada hari tertentu, sesi tertentu, di ruangan tertentu, atau apakah ruangan digunakan pada waktu yang ditentukan. Variabel keputusan ini biasanya didefinisikan sebagai variabel biner dengan nilai 1 jika kondisi terpenuhi (misalnya, dosen mengajar di sesi tersebut) dan 0 jika tidak.

Setelah mendeklarasikan variabel, fungsi objektif kemudian ditambahkan ke model berdasarkan fungsi objektif yang sudah diformulasikan sebelumnya. Fungsi objektif ini dirumuskan dengan menggunakan metode lpSum() untuk menjumlahkan penalti terhadap sesi dosen yang berurutan dan penalti terhadap penggunaan ruangan yang terlalu besar.

Selanjutnya, *constraint* atau kendala yang telah diformulasikan sebelumnya diterapkan ke model dengan menambahkan persamaan atau pertidaksamaan ke dalam model menggunakan operator +=. *Constraint* ini memastikan bahwa jadwal yang dihasilkan memenuhi aturan yang telah ditentukan, seperti satu dosen hanya dapat mengajar satu kelas pada satu waktu dan setiap ruangan hanya dapat digunakan oleh satu kelas pada satu sesi.

Setelah semua variabel dan *constraint* dideklarasikan, solver dari PuLP seperti PULP_CBC_CMD() digunakan untuk menganalisis model. Metode solve() kemudian dipanggil pada model untuk menjalankan *solver*, yang akan mencari kombinasi nilai variabel yang memenuhi semua *constraint* dan meminimalkan fungsi objektif yang telah ditentukan.

3.3.4. Pengembangan *Web Service*

Pengembangan *web service* dalam penelitian ini akan menggunakan *framework* FastAPI, dipilih karena kemampuannya yang mudah diintegrasikan dengan berbagai modul eksternal dan kemampuannya menangani permintaan secara asinkron. *Web service* ini dirancang untuk berfungsi sebagai modul terpisah yang akan menjalankan fungsi pembuatan jadwal otomatis pada Aplikasi Penjadwalan Perkuliahan Otomatis Berbasis Web dengan model *Integer Linear Programming* (ILP) yang sudah dikembangkan sebelumnya.

Web service ini akan menyediakan *endpoint* yang menerima data dari aplikasi penjadwalan. Data tersebut akan diproses menggunakan model yang telah dibuat, dan hasil pemrosesan tersebut yang berbentuk jadwal akan dikembalikan ke aplikasi dalam format JSON, sehingga memudahkan aplikasi untuk menampilkan jadwal perkuliahan yang telah dioptimalkan.

3.3.5. Pengujian

Pengujian dilakukan untuk memastikan bahwa sistem yang dikembangkan berfungsi dengan benar dan sesuai dengan kebutuhan permasalahan penjadwalan kuliah. Pengujian akan dibagi menjadi empat, yaitu pengujian solusi, pengujian sumber daya dan komputasi, pengujian *web service*, serta validasi solusi.

3.3.5.1. Pengujian Solusi (*Feasibility Testing*)

Pengujian solusi ILP dengan pendekatan *monotonicity* dilakukan dengan memeriksa bagaimana *feasibility* berubah saat *constraint* dikurangi. Jika solusi dengan seluruh *constraint* terpenuhi (status OPTIMAL atau FEASIBLE), maka setiap *subset* dari *constraint* juga seharusnya *feasible*, sesuai dengan prinsip *monotonicity*. Sebaliknya, jika suatu kombinasi *constraint* menghasilkan status

INFEASIBLE, maka ada kemungkinan *constraint* tertentu terlalu ketat atau tidak konsisten.

Pendekatan ini memungkinkan evaluasi model secara bertahap dengan menghapus atau menyesuaikan *constraint* untuk mengidentifikasi batas minimum *constraint* yang masih menjaga *feasibility*. Proses ini dilakukan secara iteratif hingga ditemukan solusi yang tetap *feasible* dalam berbagai kombinasi *constraint*, memastikan model tetap stabil dan sesuai dengan kriteria yang diinginkan.

3.3.5.2. Pengujian Sumber Daya dan Komputasi

Pengujian sumber daya dan komputasi dilakukan untuk memastikan bahwa sistem penjadwalan perkuliahan yang dikembangkan dapat menyelesaikan proses optimasi secara efisien dan memenuhi semua kendala yang telah ditentukan. Pengujian ini mencakup pengukuran waktu eksekusi, penggunaan CPU, serta konsumsi memori selama proses penjadwalan berlangsung. Model ILP dievaluasi dengan berbagai skenario, termasuk variasi ukuran data masukan, untuk menguji performa sistem. Hasil pengujian dicatat dan dianalisis guna memastikan bahwa algoritma yang digunakan mampu menghasilkan solusi optimal dalam waktu yang wajar serta tetap berada dalam batas sumber daya yang tersedia.

3.3.5.3. Pengujian *Web Service*

Pengujian *web service* dilakukan untuk memastikan semua *endpoint* yang dikembangkan berfungsi dengan baik dan sesuai dengan kebutuhan sistem penjadwalan perkuliahan. Pengujian ini menggunakan aplikasi Postman untuk mengirimkan berbagai permintaan HTTP ke setiap *endpoint* yang tersedia. Setiap permintaan diuji berdasarkan skenario yang telah dirancang, termasuk pengujian terhadap berbagai parameter yang relevan. Respons yang diterima dari *web service* akan dicatat dan dibandingkan dengan hasil yang diharapkan guna

memastikan bahwa seluruh fungsionalitas berjalan sesuai dengan kebutuhan. Skenario untuk pengujian *web service* ini dapat dilihat pada tabel 9 di bawah.

Tabel 9. Skenario Pengujian *Web Service.*

Endpoint	**Kasus Uji**	**Respon yang Diharapkan**
/api/generate-schedule	Mengirimkan data lengkap dan valid pada *endpoint* dengan parameter yang nilainya valid	Data jadwal perkuliahan
/api/generate-schedule	Mengirimkan data dengan *ClassLecturer* kosong	Tidak ada hasil jadwal perkuliahan
/api/generate-schedule	Mengirimkan data dengan *ScheduleDay* kosong	Tidak ada hasil jadwal perkuliahan
/api/generate-schedule	Mengirimkan data dengan *ScheduleSession* kosong	Tidak ada hasil jadwal perkuliahan
/api/generate-schedule	Mengirimkan data dengan *Room* kosong	Tidak ada hasil jadwal perkuliahan
/api/generate-schedule	Mengirimkan data ke *endpoint* tanpa parameter	*Error* data tidak dapat diproses (422)
/api/generate-schedule	Mengirimkan data ke *endpoint* dengan parameter tanpa nilai	*Error* data tidak dapat diproses (422)
/api/generate-schedule	Mengirimkan data ke *endpoint* dengan parameter yang nilainya tidak ditemukan	Tidak ada hasil jadwal perkuliahan
/api/generate-schedule	Mengirimkan data ke *endpoint* dengan parameter yang tipe datanya tidak sesuai	*Error* data tidak dapat diproses (422)
/api/generate-schedule	Mengirimkan *request* dengan metode yang salah	*Error* metode tidak diizinkan (405)

3.3.5.4. Validasi Solusi

Validasi solusi dilakukan sebagai langkah untuk memastikan bahwa hasil yang diperoleh dari proses optimasi telah sesuai dengan seluruh kendala yang telah ditentukan dalam model. Proses ini bertujuan untuk memverifikasi apakah setiap jadwal yang dihasilkan benar-benar mencerminkan aturan dan ketentuan yang berlaku. Apabila ditemukan ketidaksesuaian dalam jadwal yang dihasilkan, hal tersebut dapat menjadi indikator adanya kekeliruan dalam perumusan model atau proses penyelesaian oleh fungsi *solver.*

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian, pengujian, dan analisis yang telah diselesaikan, dapat diidentifikasi beberapa kesimpulan sebagai berikut:

1. Penelitian ini telah berhasil mengimplementasikan sebuah model optimasi menggunakan pendekatan *Integer Linear Programming* ke dalam sebuah *web service* yang dirancang untuk menyelesaikan permasalahan penjadwalan perkuliahan secara efektif dan efisien. Dengan dua studi kasus data asli yang digunakan dalam perkuliahan di jurusan Ilmu Komputer Universitas Lampung didapatkan hasil yang selalu optimal (*feasible*) dan juga dengan waktu eksekusi rata-rata yaitu 0,10 detik per-pertemuan.

2. Penggunaan sumber daya dalam model optimasi tergolong rendah, dengan konsumsi memori rata-rata sebesar 617,65 MB dan pemanfaatan CPU dalam kisaran 10–15%. Evaluasi dilakukan pada dua skenario dengan 264 dan 206 unit data pertemuan, dalam ruang solusi yang mencakup total 312 unit yang tersedia. Hasil pengujian menunjukkan bahwa waktu komputasi dan penggunaan sumber daya meningkat dalam skala polinomial terhadap jumlah unit data. Berdasarkan analisis empiris, kompleksitas waktu eksekusi model diperkirakan berada dalam orde $O(n^{2.38})$, yang konsisten dengan sifat permasalahan *Integer Linear Programming* (ILP).

3. Penelitian ini telah berhasil mengembangkan layanan web dengan kerangka kerja FastAPI yang memungkinkan integrasi model *Integer Linear Programming* (ILP) dengan aplikasi klien. Layanan ini menyediakan berbagai *endpoint* yang dapat digunakan untuk mengajukan permintaan optimasi jadwal berdasarkan parameter tertentu. Dengan pendekatan ini, sistem dapat menangani proses penjadwalan secara efisien dan memastikan solusi optimal sesuai dengan batasan dan fungsi objektif yang telah ditetapkan dalam model ILP.

5.2. Saran

Walaupun penelitian ini telah berhasil mencapai tujuan yang telah ditetapkan, masih terdapat beberapa peluang untuk pengembangan lebih lanjut yang dapat dieksplorasi. Beberapa aspek tertentu dalam sistem yang telah dirancang masih memiliki potensi untuk disempurnakan guna meningkatkan efisiensi, akurasi, serta fleksibilitas dalam penerapannya. Oleh karena itu, berbagai strategi dan pendekatan tambahan dapat dipertimbangkan untuk memperbaiki dan memperluas cakupan penelitian ini di masa mendatang.

1. Dalam model optimasi *Integer Linear Programming* yang telah dikembangkan masih dapat ditingkatkan dengan menambahkan atau menyesuaikan beberapa kendala agar lebih relevan seiring dengan perkembangan waktu. Selain itu, aspek opsional juga dapat dimasukkan ke dalam fungsi objektif, seperti preferensi individu atau kebutuhan tambahan lainnya, untuk meningkatkan fleksibilitas dan akurasi solusi yang dihasilkan.
2. Mengembangkan model penjadwalan asisten untuk kelas praktikum dan responsi yang terintegrasi dengan pendekatan *Integer Linear Programming* (ILP). Integrasi ini bertujuan untuk mengoptimalkan penjadwalan secara keseluruhan setelah data perkuliahan asisten dosen tersedia, sehingga menghasilkan jadwal akhir yang lebih efisien.

3. Menambahkan *constraints* pada tingkat basis data untuk mencegah terjadinya inkonsistensi atau kesalahan data yang mungkin lolos dari validasi pada tingkat aplikasi atau model. Dengan penerapan *constraint* seperti *unique constraint*, dan *check constraint*, sistem dapat memastikan integritas data tetap terjaga, sehingga hasil penjadwalan yang dihasilkan lebih andal dan bebas dari konflik logika akibat data yang tidak valid.

4. Menambahkan fitur konfigurasi model yang memungkinkan perubahan parameter dan kendala secara fleksibel tanpa perlu mengubah apa pun pada kode sumber. Hal ini bertujuan untuk meningkatkan efektivitas dan efisiensi serta fleksibilitas sistem dalam menghasilkan jadwal yang sesuai dengan kebutuhan yang paling relevan.

DAFTAR PUSTAKA

Amenta, N. (1993). *Helly Theorems and Generalized Linear Programming*. https://doi.org/https://doi.org/10.1145/160985.161000

Anam, K., Asyhar, B., Saddhono, K., & Setyawan, B. (2021). E-SIP: Website-Based Scheduling Information System to Increase the Effectivity of Lecturer's Performance and Learning Process. *Ingénierie Des Systèmes d Inf.*, *26*, 265–273. https://doi.org/10.18280/isi.260303

Basriati. (2018). Integer Linear Programming Dengan Pendekatan Metode Cutting Plane Dan Branch And Bound Untuk Optimasi Produksi Tahu. *Jurnal Sains Matematika Dan Statistika*, *4*(2).

Bixby, R. E., & Lee, E. K. (1996). *Solving a Truck Dispatching Scheduling Problem Using Branch-and-Cut* (Vol. 46, Issue 3).

Boyd, S., & Mattingley, J. (2010). *Branch and Bound Methods*.

Campion, M., Dalla Preda, M., Giacobazzi, R., & Urban, C. (2024). Monotonicity and the Precision of Program Analysis. *Proceedings of the ACM on Programming Languages*, *8*, 1629–1662. https://doi.org/10.1145/3632897

Censor, Y., & Zaknoon, M. (2018). *Algorithms and Convergence Results of Projection Methods for Inconsistent Feasibility Problems: A Review*. http://arxiv.org/abs/1802.07529

Ceria, S., Cordier, C., Marchand, H., & Wolsey, L. A. (1998). Cutting planes for integer programs with general integer variables. *Mathematical Programming*, *81*(2), 201–214. https://doi.org/10.1007/BF01581105

Chanida Leelayutto. (2019). *Classroom Scheduling Problem with an integer linear program*.

Chen, J. (2023). *Model Algorithm Research based on Python Fast API*.

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, *153*(1), 117–135. https://doi.org/10.1016/S0377-2217(03)00103-6

Efuntade, O. O., & Efuntade, A. O. (2023). Application Programming Interface (API) And Management of Web-Based Accounting Information System

(AIS): Security of Transaction Processing System, General Ledger and Financial Reporting System. *Journal of Accounting and Financial Management*, *9*(6), 1–18. https://doi.org/10.56201/jafm.v9.no6.2023.pg1.18

Ferris, C., & Farrell, J. (2003). What are web services? *Communications of the ACM*. http://d.web.umkc.edu/di5x7/output/Paper%20Critique%20-%20Web%20Services.pdf

Graver, J. E. (1975). On the Foundations of Linear and Integer Linear Programming I*. In *Mathematical Programming* (Vol. 8). North-Holland Publishing Company.

Herroelen, W. (2005). *Project Scheduling-Theory and Practice*.

Irvin, S., & Brown, H. (1999). Self-scheduling with Microsoft Excel. *Nursing Economic*, *17 4*, 201–206.

Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Birkhäuser Boston. https://doi.org/10.1007/978-1-4684-6802-1

Livia, C. Y., & Oktiarso, T. (2017). Penjadwalan Untuk Memininimalkan Total Tardiness Dengan Metode Integer Linear Programming. *Jurnal Teknik Industri*, *18*(2), 127–137. https://doi.org/10.22219/jtiumm.vol18.no2.127-137

Mitchell, J. (2011). *Branch and Cut*. https://doi.org/10.1002/9780470400531.EORMS0117

Mitchell, S. (2009). An Introduction to pulp for Python Programmers. In *The Python Papers Monograph* (Vol. 1). http://ojs.pythonpapers.org/index.php/tppm

Mitten, L. G. (1970). Branch-and-Bound Methods: General Formulation and Properties. *Oper. Res.*, *18*, 24–34. https://doi.org/10.1287/opre.18.1.24

Muhammad Asrar Amir. (2017). *Optimasi Penjadwalan Perawat Menggunakan Gabungan Integer Linear Programming dan Variable Neighborhood Search. Studi Kasus Instalasi Gawat Darurat Rumah Sakit Ibnu Sina Makassar*.

pulp documentation team. (2009). *Optimization with PuLP*. Https://Coin-or.Github.Io/Pulp/.

Python Software Foundation. (2024). *Python Documentation*. Https://Www.Python.Org/Doc/Essays/Blurb/.

Roostaee, R., Izadikhah, M., & Hosseinzadeh Lotfi, F. (2012). An interactive procedure to solve multi-objective decision-making problem: An improvment to STEM method. *Journal of Applied Mathematics*, *2012*. https://doi.org/10.1155/2012/324712

Safitri, E., Basriati, S., Eka Putri Program Studi Matematika, R., Sains dan Teknologi, F., Sultan Syarif Kasim Riau Jl Soebrantas No, U. H., Baru, S., Korespondensi, I., & Eka Putri, R. (2021). *Optimasi Penjadwalan Perawat Menggunakan Integer Linear Programming (Studi Kasus: RS. Aulia Hospital Pekanbaru)*. *10*(1), 45–56. https://doi.org/10.14421/fourier.2021.101.45-56

Santos, H. G., & Toffolo, T. A. M. (2020). *Mixed Integer Linear Programming with Python*.

Segura, S., Parejo, J. A., Troya, J., & Ruiz-Cortés, A. (2018). Metamorphic Testing of RESTful Web APIs. *IEEE Transactions on Software Engineering*, *44*, 1083–1099. https://doi.org/10.1109/TSE.2017.2764464

Serafini, P. (2005). Linear programming with variable matrix entries. *Oper. Res. Lett.*, *33*, 165–170. https://doi.org/10.1016/j.orl.2004.04.011

Tantawy, S. (2014). A New Procedure for Solving Integer Linear Programming Problems. *Arabian Journal for Science and Engineering*, *39*, 5265–5269. https://doi.org/10.1007/s13369-014-1079-6

Trauth, C. A., & Woolsey, R. E. (1969). Integer Linear Programming: A Study in Computational Efficiency. *Management Science*, *15*(9), 481–493. https://doi.org/10.1287/mnsc.15.9.481

Trilling, L., Guinet, A., & Magny, D. Le. (2006). NURSE SCHEDULING USING INTEGER LINEAR PROGRAMMING AND CONSTRAINT PROGRAMMING. *IFAC Proceedings Volumes*, *39*(3), 671–676. https://doi.org/https://doi.org/10.3182/20060517-3-FR-2903.00340

Tsalgatidou, A., Benatallah, B., & Casati, F. (2002). *Distributed and Parallel Databases* (Vol. 12).

Varela, L., Silva, S., & Aparício, J. N. (2004). *Scheduling Decision-Making Using Web Service*. 169–176. https://doi.org/10.5220/0001399601690176

Yu, B., Mitchell, J. E., & Pang, J.-S. (2019). Solving linear programs with complementarity constraints using branch-and-cut. *Mathematical Programming Computation*, *11*(2), 267–310. https://doi.org/10.1007/s12532-018-0149-2