



DISERTASI - ES 236601

**PENGEMBANGAN ALGORITMA
HIPER-HEURISTIK DALAM MENYELESAIKAN
PERMASALAHAN OPTIMASI PENJADWALAN
LINTAS DOMAIN**

**I GUSTI AGUNG PREMANANDA
7026211002**

**DOSEN PEMBIMBING
Dr. Ir. Aris Tjahyanto, M.Kom
Ahmad Muklason, S.Kom., M.Sc., Ph.D.**

**PROGRAM STUDI DOKTOR SISTEM INFORMASI
DEPARTEMEN SISTEM INFORMASI
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2025**

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama : I Gusti Agung Premananda / 7026211002
mahasiswa /
NRP
Program : S3 Sistem Informasi
studi
Dosen : Dr. Ir. Aris Tjahyanto, M.Kom /
Pembimbing 196503101991021001
/ NIP

dengan ini menyatakan bahwa Disertasi dengan judul
**"PENGEMBANGAN ALGORITMA HIPER-HEURISTIK
DALAM MENYELESAIKAN PERMASALAHAN
OPTIMASI PENJADWALAN LINTAS DOMAIN"** adalah
hasil karya sendiri , bersifat orisinal , dan ditulis dengan
mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan
pernyataan ini , maka saya
bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut
Teknologi Sepuluh Nopember.

Surabaya, 23 Januari 2025

Mengetahui

Dosen Pembimbing

Dr. Ir. Aris Tjahyanto, M.Kom
NIP. 196503101991021001

Mahasiswa



I Gusti Agung Premananda
NRP. 7026211002

Halaman ini sengaja dikosongkan.

LEMBAR PENGESAHAN DISERTASI

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Doktor (Dr)
di
Institut Teknologi Sepuluh Nopember

Oleh:
I Gusti Agung Premananda
NRP: 7026211002

Tanggal Ujian: 15 Januari 2025
Periode Wisuda ITS: Maret 2025

Disetujui oleh:
Pembimbing:

Dr. Ir. Aris Tjahyanto, M.Kom
NIP: 196503101991021001

Ahmad Mukhlason, S.Kom, M.Sc, Ph.D
NIP: 198203022009121009

Penguji:

Prof. Nur Aini Rakhmawati, S.Kom., M.Sc.Eng.,
Ph.D
NIP: 198201202005012001

Amalia Utamima, S.Kom, MBA, Ph.D
NIP: 198612132015042001

Komarudin, S.T., M.Eng., Ph.D.
NIP: 3175071502830017

Surabaya, 24 Januari 2025

Kepala Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas

LP/P/25/223



Prof. Dr. Wiwik Anggraeni, S.Si, M.Kom
NIP. 197601232001122002

Halaman ini sengaja dikosongkan.

PENGEMBANGAN ALGORITMA HIPER-HEURISTIK DALAM MENYELESAIKAN PERMASALAHAN OPTIMASI PENJADWALAN LINTAS DOMAIN

Nama Mahasiswa : I Gusti Agung Premananda
Promotor : Dr. Ir. Aris Tjahyanto, M.Kom
Co-Promotor : Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRAK

Permasalahan penjadwalan merupakan permasalahan yang sering ditemui dalam berbagai konteks nyata, seperti pada bidang pendidikan, olahraga dan transportasi. Permasalahan ini bertujuan untuk menjadwalkan sejumlah kegiatan dalam slot waktu yang tersedia dengan memastikan tidak ada pelanggaran terhadap *hard constraint* dan meminimalkan pelanggaran *soft constraint*. Karena kompleksitasnya, permasalahan penjadwalan telah diklasifikasikan sebagai permasalahan NP-Hard. Oleh karena itu, pengembangan algoritma heuristik menjadi pilihan utama untuk mendapatkan solusi yang praktis.

Penelitian terkini cenderung berfokus pada pengembangan algoritma yang spesifik terhadap satu studi kasus atau satu *benchmark*. Algoritma yang dikembangkan sering kali tidak dapat memberikan performa yang setara saat diterapkan pada studi kasus atau *benchmark* yang berbeda. Hal ini menyebabkan algoritma yang dikembangkan pada penelitian terdahulu sulit untuk diimplementasikan secara praktis. Kondisi ini mendorong perlunya pengembangan algoritma generik yang mampu menyelesaikan permasalahan penjadwalan lintas studi kasus, lintas dataset, atau bahkan lintas domain.

Penelitian ini mengembangkan algoritma berbasis metode hiper-heuristik untuk menghasilkan algoritma generik dalam menyelesaikan permasalahan penjadwalan lintas domain. Penelitian ini mengembangkan dua jenis algoritma yaitu Progressive Acceptance Iterated Local Search (PA-ILS) dan Adaptive

Threshold-Iterated Local Search (AT-ILS). Algoritma PA-ILS memfokuskan upaya pada penemuan solusi *feasible* melalui eksplorasi intensif dengan cara menerima seluruh solusi yang dihasilkan. Sementara itu, algoritma AT-ILS bertujuan meningkatkan kualitas solusi penjadwalan dan memiliki keunggulan utama dengan tidak membutuhkannya pengaturan parameter.

Kedua algoritma diuji menggunakan tiga *benchmark* dengan jenis permasalahan berbeda: penjadwalan ujian (*Toronto Benchmark*), penjadwalan mata kuliah (*International Timetabling Competition (ITC) 2019*) dan penjadwalan kompetisi olahraga (*ITC 2021*). Selain itu, kedua algoritma diterapkan pada penjadwalan sesi praktikum dan ujian praktikum dalam studi kasus nyata di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember.

Hasil pengujian menunjukkan bahwa kedua algoritma efektif dalam menangani permasalahan penjadwalan lintas domain. Pada algoritma PA-ILS, solusi *feasible* berhasil diperoleh dengan tingkat keberhasilan 97,4% dan mengungguli pendekatan lain, khususnya pada *benchmark* ITC 2021. Sementara itu, algoritma AT-ILS menunjukkan kinerja yang baik dan konsisten pada ketiga *benchmark* dengan berada pada peringkat di atas rata-rata dibandingkan penelitian terdahulu. Dalam studi kasus di Departemen Sistem Informasi, algoritma AT-ILS juga terbukti unggul dibandingkan dua algoritma pembanding, yaitu Hill Climbing dan Late Acceptance Hill Climbing.

Kata Kunci : Optimasi, Penjadwalan, Lintas Domain, Hiper-Heuristik

DEVELOPING HYPER-HEURISTIC ALGORITHM FOR SOLVING CROSS DOMAIN TIMETABLING OPTIMIZATION PROBLEMS

by : I Gusti Agung Premananda
Promotor : Dr. Ir. Aris Tjahyanto, M.Kom
Co-Promotor : Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRACT

Timetabling problems are frequently encountered in various real-world contexts, such as education, sports, and transportation. These problems aim to assign a set of activities into available time slots while ensuring that no hard constraints are violated and that soft constraint violations are minimized. Due to their complexity, timetabling problems have been classified as NP-Hard. Consequently, the development of heuristic algorithms has become a primary option for obtaining practical solutions.

Recent research tends to focus on developing algorithms tailored to a single case study or a specific benchmark. Often, such algorithms fail to deliver comparable performance when applied to different case studies or benchmarks. This limitation makes it challenging to implement previously developed algorithms in practical settings. Consequently, there is a growing need to develop generic algorithms capable of solving timetabling problems across different case studies, datasets, or even domains.

This study develops hyper-heuristic-based algorithms to produce generic solutions for cross-domain timetabling problems. Two types of algorithms are proposed: Progressive Acceptance Iterated Local Search (PA-ILS) and Adaptive Threshold-Iterated Local Search (AT-ILS). PA-ILS concentrates on finding feasible solutions through intensive exploration by accepting every solution generated. Meanwhile, AT-ILS aims to improve the quality of timetabling

solutions and has the major advantage of not requiring parameter tuning.

Both algorithms were tested using three different benchmarks: exam timetabling (Toronto Benchmark), course timetabling (International Timetabling Competition (ITC) 2019), and sports competition timetabling (ITC 2021). Additionally, they were applied to a real-world case study of laboratory session and practical exam timetabling in the Department of Information Systems, Institut Teknologi Sepuluh Nopember.

The test results show that both algorithms are effective in handling cross-domain timetabling problems. PA-ILS successfully obtained feasible solutions with a 97.4% success rate, outperforming other approaches, particularly on the ITC 2021 benchmark. Meanwhile, AT-ILS displayed consistent and robust performance across all three benchmarks, ranking above average compared to previous studies. In the Department of Information Systems case study, AT-ILS also outperformed two comparison algorithms: Hill Climbing and Late Acceptance Hill Climbing.

Keywords : Optimization, Timetabling, Cross-Domain, Hyper-Heuristic,

KATA PENGANTAR

Puji syukur atas karunia yang telah diberikan Ida Sang Hyang Widhi Wasa selama ini sehingga penulis mendapatkan kelancaran dalam menyelesaikan disertasi dengan judul: “*PENGEMBANGAN ALGORITMA HIPER-HEURISTIK DALAM MENYELESAIKAN PERMASALAHAN OPTIMASI PENJADWALAN LINTAS DOMAIN*”. Penggeraan disertasi ini tidak lepas dari bantuan, bimbingan, dukungan, dan doa dari berbagai pihak. Oleh karena itu, izinkan penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Orang tua penulis yaitu Bapak I Gusti Agung Putu Alit Wirawan dan Ibu Setyawati Hirawan yang telah memberikan dukungan dan doa agar bisa menyelesaikan disertasi ini.
2. Bapak Dr. Ir. Aris Tjahyanto, M.Kom. dan Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing yang telah meluangkan banyak waktu untuk membimbing serta mengarahkan penulis dalam penggeraan disertasi ini.
3. Ibu Prof. Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D, Amalia Utamima, S.Kom., MBA., Ph.D., dan Bapak Komarudin, S.T., M.Eng., Ph.D. selaku dosen penguji yang telah memberikan kritik dan saran yang membangun agar disertasi ini menjadi lebih baik.

Penulis juga menyadari penggeraan penelitian disertasi ini masih jauh dari kata sempurna. Oleh karena itu, penulis menerima pertanyaan, saran, dan kritik yang membangun untuk menjadi masukan penulis dan masukan penelitian selanjutnya. Semoga penelitian disertasi ini dapat memberikan manfaat bagi pembaca.

Surabaya, Januari 2025

Penulis

Daftar Isi

Lembar Orisinalitas	i
Lembar Pengesahan	iii
Abstrak	v
Abstract	vii
Kata Pengantar	ix
Daftar Isi	xii
Daftar Tabel	xvii
Daftar Gambar	xxi
Daftar Persamaan	xxv
Daftar Algoritma	xxvii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	7
1.3 Tujuan dan Manfaat Penelitian	8
1.4 Kontribusi dan Orisinalitas Penelitian	9
1.5 Batasan Penelitian	10
1.6 Publikasi	10
1.7 Sistematika Penulisan	11
2 KAJIAN PUSTAKA	13

2.1	Dasar Teori	13
2.1.1	Klasifikasi Permasalahan Komputasi dan Kompleksitas	14
2.1.1.1	Permasalahan P dan NP	14
2.1.1.2	Waktu Polinomial	15
2.1.1.3	NP-Complete	18
2.1.1.4	NP-Hard	19
2.1.1.5	Klasifikasi Permasalahan Penjadwalan sebagai NP-Complete dan NP-Hard	19
2.1.2	Metode Aproksimasi, Heuristik dan Metaheuristik	20
2.1.3	Permasalahan Penjadwalan	22
2.1.4	<i>Fitness Landscape</i>	24
2.1.4.1	Karakteristik <i>Fitness Landscape</i>	24
2.1.4.2	Implikasi <i>Landscape</i> terhadap Pengembangan Strategi <i>Move Acceptance</i>	26
2.1.5	Tingkatan Generalitas Algoritma	28
2.1.6	Hiper-heuristik	29
2.1.7	Algoritma ILS	31
2.2	Kajian Penelitian Terdahulu	32
2.2.1	<i>Benchmark</i> Toronto	34
2.2.2	<i>Benchmark</i> Socha	38
2.2.3	<i>Benchmark</i> ITC 2007 Track 1	41
2.2.4	<i>Benchmark</i> ITC 2007 Track 2	45
2.2.5	<i>Benchmark</i> ITC 2007 Track 3	48
2.2.6	<i>Benchmark</i> ITC 2019	50
2.2.7	<i>Benchmark</i> ITC 2021	52
2.2.8	Tingkat Generalitas dalam Algoritma Penjadwalan	54
2.2.9	Penelitian dengan Fokus Pencarian Solusi <i>Feasible</i>	58
2.3	Analisis Kebaruan Penelitian	59
3	METODOLOGI PENELITIAN	63
3.1	Identifikasi Permasalahan	64
3.2	Penentuan Kebutuhan Solusi	66
3.2.1	Kerangka Algoritma	66

3.2.2	Penentuan Kebutuhan	67
3.3	Perancangan dan Pengembangan Algoritma	67
3.3.1	Proses Pengembangan	68
3.3.2	Alat dan Teknik Pengembangan	69
3.4	Demonstrasi Algoritma	69
3.4.1	<i>Benchmark</i> dan Studi Kasus yang Digunakan untuk Demonstrasi . .	69
3.4.2	Lingkungan Demonstrasi	70
3.4.3	Proses Demonstrasi untuk Mencari Solusi Feasible	71
3.4.4	Proses Demonstrasi untuk Optimasi Solusi	71
3.5	Evaluasi Algoritma	72
3.5.1	Metrik Evaluasi	72
3.5.2	Metode Evaluasi	73
4	DESAIN DAN PENGEMBANGAN ALGORITMA	75
4.1	Rasionalitas Desain Algoritma	76
4.1.1	Keseimbangan antara Eksplorasi dan Eksloitasi	76
4.1.2	Penggunaan Pendekatan Hiper-Heuristik <i>Selection-Perturbation</i> . .	78
4.1.3	Pemilihan Algoritma ILS	79
4.1.4	Pendekatan Desain Algoritma PA-ILS dan AT-ILS	79
4.1.4.1	Pendekatan Desain Algoritma PA-ILS	80
4.1.4.2	Pendekatan Desain Algoritma AT-ILS	81
4.2	Desain Algoritma PA-ILS	83
4.2.1	Perhitungan Variabel <i>probabilityNonRandomTimeSlot</i> . .	84
4.2.2	Proses Menjadwalkan Kegiatan	87
4.2.3	Tahapan <i>Perturbation</i>	87
4.2.4	Tahapan <i>Local Search</i>	88
4.2.5	Parameter	90
4.3	Desain Algoritma AT-ILS	91
4.3.1	Desain Umum Algoritma AT-ILS	92
4.3.2	Fungsi dan Perhitungan Nilai Ambang Batas	92
4.3.3	Tahapan <i>Perturbation</i>	94
4.3.4	Tahapan <i>Local Search</i>	95
4.3.4.1	Fungsi <i>UpdateThresholdLocal</i>	97

4.3.4.2	Fungsi <i>ApplyHeuristicToAllEvents</i>	98
4.3.4.3	Fungsi <i>UpdateThresholdList</i>	100
4.3.5	Tahapan <i>Move Acceptance</i>	101
4.4	LLH	103
4.4.1	<i>Move</i>	103
4.4.2	<i>Swap</i>	104
4.4.3	<i>Kempe Chain</i>	105
4.5	Pengaturan Parameter	106
5	PERMASALAHAN PENJADWALAN UJIAN (<i>BENCHMARK TORONTO</i>)	113
5.1	Karakteristik Dataset	114
5.1.1	Karakteristik Dataset Secara Umum	114
5.1.2	Struktur Dataset	115
5.2	<i>Hard</i> dan <i>Soft Constraint</i>	115
5.3	Hasil Implementasi dan Analisis Algoritma	117
5.3.1	Hasil Tahapan Pencarian Solusi Feasible	117
5.3.2	Analisis Proses Algoritma PA-ILS	119
5.3.3	Hasil Tahapan Optimasi	119
5.3.4	Analisis Proses Algoritma AT-ILS	121
5.4	Perbandingan dengan Studi Sebelumnya	122
5.5	Kesimpulan	125
6	PERMASALAHAN PENJADWALAN MATA KULIAH (<i>BENCHMARK INTERNATIONAL TIMETABLING COMPETITION 2019</i>)	127
6.1	Karakteristik Dataset	128
6.1.1	Karakteristik Dataset Secara Umum	128
6.1.2	Struktur Dataset	130
6.2	<i>Hard</i> dan <i>Soft Constraint</i>	132
6.2.1	Batasan Slot Waktu	132
6.2.2	Batasan Ruangan	132
6.2.3	Batasan Mata Kuliah	133
6.2.4	Batasan Mahasiswa	134
6.2.5	Batasan Distribusi	135

6.3	Hasil Implementasi dan Analisis Algoritma	147
6.3.1	Hasil Tahapan Pencarian Solusi <i>Feasible</i>	148
6.3.2	Analisis Proses Algoritma PA-ILS	150
6.3.3	Hasil Tahapan Optimasi	151
6.3.4	Analisis Proses Algoritma AT-ILS	154
6.4	Perbandingan dengan Studi Sebelumnya	156
6.5	Kesimpulan	163
7	PERMASALAHAN PENJADWALAN KOMPETISI OLAH RAGA (BENCHMARK INTERNATIONAL TIMETABLING COMPETITION 2021)	165
7.1	Karakteristik Dataset	166
7.1.1	Karakteristik Dataset Secara Umum	166
7.1.2	Struktur Dataset	169
7.2	<i>Hard</i> dan <i>Soft Constraint</i>	170
7.3	Hasil Implementasi dan Analisis Algoritma	174
7.3.1	Hasil Tahapan Pencarian Solusi <i>Feasible</i>	175
7.3.2	Analisis Proses Algoritma PA-ILS	177
7.3.3	Hasil Tahapan Optimasi	179
7.3.4	Analisis Proses Algoritma AT-ILS	183
7.4	Perbandingan dengan Studi Sebelumnya	184
7.5	Kesimpulan	190
8	PERMASALAHAN PENJADWALAN SESI PRAKTIKUM DAN UJIAN PRAKTIKUM (STUDI KASUS DEPARTEMEN SISTEM INFORMASI ITS)	195
8.1	Karakteristik Dataset	196
8.1.1	Karakteristik Dataset Secara Umum	196
8.1.2	Struktur Dataset	197
8.2	Model Matematika	199
8.2.1	Variabel Keputusan	199
8.2.2	Fungsi Objektif	199
8.2.3	<i>Hard Constraint</i>	200
8.2.3.1	Penjadwalan Praktikum	200
8.2.3.2	Penjadwalan Ujian	202

8.2.4	<i>Soft Constraint</i>	204
8.2.4.1	Distribusi Beban Asisten Pengajar	204
8.2.4.2	Preferensi Waktu	206
8.3	Hasil Implementasi dan Analisis Algoritma	207
8.3.1	Hasil dan Analisis Tahapan Pencarian Solusi <i>Feasible</i>	207
8.3.2	Hasil dan Analisis Tahapan Optimasi	209
8.4	Perbandingan dengan Algoritma LAHC dan HC	213
8.5	Kesimpulan	215
9	KESIMPULAN DAN SARAN	217
9.1	Kesimpulan	219
9.2	Saran Penelitian Selanjutnya	219
DAFTAR PUSTAKA	221	
Lampiran A	237	
Lampiran B	245	
Lampiran C	253	
Lampiran D	263	
Lampiran E	267	

Daftar Tabel

2.2.1 Perbandingan Pemeringkatan Penelitian pada <i>Benchmark</i> Toronto	35
2.2.2 Perbandingan Pemeringkatan Penelitian pada <i>Benchmark</i> Socha	40
2.2.3 Perbandingan Pemeringkatan Penelitian pada <i>Benchmark</i> ITC 2007 Track 1	44
2.2.4 Perbandingan Penelitian pada <i>Benchmark</i> ITC 2007 Track 2	47
2.2.5 Perbandingan Penelitian pada <i>Benchmark</i> ITC 2007 Track 3	50
2.2.6 Perbandingan Penelitian pada <i>Benchmark</i> ITC 2019	52
2.2.7 Perbandingan Penelitian pada <i>Benchmark</i> ITC 2021	54
2.2.8 Perbandingan Penelitian pada Tingkatan Generalitas <i>Benchmark</i> dan Permasalahan	57
4.2.1 Daftar Parameter	90
4.5.1 Hasil Uji Coba Nilai Parameter <i>decreaseRate</i>	108
4.5.2 Hasil Uji Coba Nilai Parameter <i>constantFactor</i>	110
4.5.3 Daftar Nilai Parameter	112
5.1.1 Deskripsi Dataset pada <i>Benchmark</i> Toronto	114
5.3.1 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark</i> Toronto	118
5.3.2 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark</i> Toronto	121
5.4.1 Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada <i>Benchmark</i> Toronto	124
6.1.1 Deskripsi Dataset pada <i>Benchmark</i> ITC 2019 Kategori Early	129
6.1.2 Deskripsi Dataset pada <i>Benchmark</i> ITC 2019 Kategori Middle	129
6.1.3 Deskripsi Dataset pada <i>Benchmark</i> ITC 2019 Kategori Late	130
6.3.1 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark</i> ITC 2019 Kategori Early	149

6.3.2 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2019</i> Kategori Middle	149
6.3.3 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2019</i> Kategori Late	150
6.3.4 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2019</i> Kategori Early	153
6.3.5 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2019</i> Kategori Middle	153
6.3.6 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2019</i> Kategori Late	154
6.4.1 Perbandingan Jumlah Solusi <i>Feasible</i> pada Permasalahan ITC 2019	156
6.4.2 Perbandingan Hasil Peringkat Keseluruhan pada Permasalahan ITC 2019	159
6.4.3 Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada <i>Benchmark ITC 2019</i> Kategori Early	160
6.4.4 Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada <i>Benchmark ITC 2019</i> Kategori Middle	161
6.4.5 Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada <i>Benchmark ITC 2019</i> Kategori Late	162
7.1.1 Deskripsi Dataset pada <i>Benchmark ITC 2021</i> Kategori Early	167
7.1.2 Deskripsi Dataset pada <i>Benchmark ITC 2021</i> Kategori Middle	167
7.1.3 Deskripsi Dataset pada <i>Benchmark ITC 2021</i> Kategori Late	168
7.3.1 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2021</i> Kategori Early	175
7.3.2 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2021</i> Kategori Middle	176
7.3.3 Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2021</i> Kategori Late	177
7.3.4 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2021</i> Kategori Early	181
7.3.5 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2021</i> Kategori Middle	181
7.3.6 Hasil Uji Coba Tahapan Optimasi pada <i>Benchmark ITC 2021</i> Kategori Late	182
7.4.1 Perbandingan Solusi <i>Feasible</i> dengan Penelitian Sebelumnya Pada <i>Benchmark ITC 2021</i>	186
7.4.2 Perbandingan Peringkat Rata-Rata pada <i>Benchmark ITC 2021</i>	187

7.4.3	Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada <i>Benchmark ITC 2021 Kategori Early</i>	187
7.4.4	Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada <i>Benchmark ITC 2021 Kategori Middle</i>	188
7.4.5	Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada <i>Benchmark ITC 2021 Kategori Late</i>	189
8.1.1	Deskripsi Dataset untuk Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum	197
8.3.1	Hasil Uji Coba Tahapan Pencarian Solusi <i>Feasible</i> pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum	209
8.3.2	Hasil Uji Coba Tahapan Optimasi pada Permasalahan Penjadwalan Sesi Praktikum dan Penjadwalan Ujian Praktikum	211
8.4.1	Perbandingan Hasil dari Penelitian ini dengan Algoritma LAHC dan HC	214
A.1	Hasil Pemeringkatan Penelitian pada <i>Benchmark Toronto Bagian 1</i>	237
A.2	Hasil Pemeringkatan Penelitian pada <i>Benchmark Toronto Bagian 2</i>	238
A.3	Hasil Pemeringkatan Penelitian pada <i>Benchmark Toronto Bagian 3</i>	239
A.4	Hasil Pemeringkatan Penelitian pada <i>Benchmark Toronto Bagian 4</i>	241
A.5	Hasil Pemeringkatan Penelitian pada <i>Benchmark Toronto Bagian 5</i>	242
B.1	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Early Bagian 1</i>	245
B.2	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Early Bagian 2</i>	246
B.3	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Middle Bagian 1</i>	248
B.4	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Middle Bagian 2</i>	249
B.5	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Late Bagian 1</i>	250
B.6	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2019 Kategori Late Bagian 2</i>	251

C.1	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Early Bagian 1	253
C.2	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Early Bagian 2	255
C.3	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Middle Bagian 1	256
C.4	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Middle Bagian 2	258
C.5	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Late Bagian 1	259
C.6	Hasil Pemeringkatan Penelitian pada <i>Benchmark ITC 2021</i> Kategori Late Bagian 2	261
D.1	Preferensi Slot Waktu	264

Daftar Gambar

2.1.1 Perbandingan Kompleksitas Waktu Permasalahan Pengurutan dengan algoritma <i>Merge Sort</i> dan Permasalahan Penjadwalan	17
2.1.2 Contoh Perbandingan Kurva pada Landscape dengan Kondisi <i>Ruggedness</i> dan <i>smoothness</i>	26
2.1.3 Ilustrasi <i>local optima</i> , <i>plateaus</i> , dan <i>global optima</i>	27
2.1.4 Klasifikasi Metode Hiper-Heuristik (Burke et al., 2013)	30
2.1.5 Struktur Hiper-Heuristik (Choong et al., 2018)	31
2.2.1 Persebaran Pendekatan pada <i>Benchmark</i> Toronto	38
2.2.2 Persebaran Pendekatan pada <i>Benchmark</i> Socha	41
2.2.3 Persebaran Pendekatan pada <i>Benchmark</i> ITC 2007 Track 1	46
2.2.4 Persebaran Pendekatan pada <i>Benchmark</i> ITC 2007 Track 2	48
2.2.5 Persebaran Pendekatan pada <i>Benchmark</i> ITC 2007 Track 3	51
4.1.1 Ilustrasi Proses Pencarian dengan Tahapan Eksplorasi yang Berlebihan . .	77
4.1.2 Ilustrasi Proses Pencarian dengan Tahapan Eksplorasi yang Berlebihan . .	77
4.1.3 Ilustrasi dari Proses Optimasi Pada Algoritma Simulated Annealing . . .	81
4.1.4 Ilustrasi dari Proses Optimasi Pada Algoritma LAHC dan Threshold Acceptance	82
4.1.5 Ilustrasi dari Algoritma Simulated Annealing yang Tidak Menggunakan Nilai Parameter yang Tepat	83
4.1.6 Ilustrasi Solusi yang Menjauh dari Nilai Solusi Terbaik	84
4.2.1 Grafik Perubahan Nilai Variabel <i>probabilityNonRandomTimeSlot</i>	86
4.2.2 Ilustrasi dari Proses Penjadwalan Permainan antara Tim 1 vs Tim 2 . . .	88
4.4.1 Ilustrasi Operator <i>Move</i>	104
4.4.2 Ilustrasi Operator <i>Swap</i>	105

4.4.3 Ilustrasi Operator Kempe	106
4.5.1 Ilustrasi Perbandingan Nilai pada Parameter <i>constantFactor</i>	109
4.5.2 Ilustrasi Perbandingan Nilai pada Parameter <i>constantFactor</i>	110
5.1.1 Ilustrasi Struktur Dataset Toronto	115
5.3.1 Grafik Perubahan Solusi dalam Proses Pencarian Solusi <i>Feasible</i> pada <i>Benchmark</i> Toronto	119
5.3.2 Box plot Hasil Algoritma Optimasi pada Tiga Dataset: EAR83, CAR91, dan HEC92	122
5.3.3 Grafik Perubahan Solusi dalam Proses Optimasi pada <i>Benchmark</i> Toronto	123
5.3.4 Grafik Perubahan Solusi dalam Proses <i>Local Search</i> pada <i>Benchmark</i> Toronto	124
5.4.1 Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark</i> Toronto	126
6.1.1 Ilustrasi Elemen Problem dan Optimization pada Dataset ITC 2019	131
6.1.2 Ilustrasi Elemen Room pada Dataset ITC 2019	131
6.1.3 Ilustrasi Elemen Students pada Dataset ITC 2019	132
6.2.1 Ilustrasi Batasan Waktu pada Dataset ITC 2019	133
6.2.2 Ilustrasi Batasan Ruangan pada Dataset ITC 2019	133
6.2.3 Struktur Mata Kuliah pada Permasalahan ITC 2019	134
6.2.4 Ilustrasi Strukur Mata Kuliah pada Permasalahan ITC 2019	135
6.2.5 Ilustrasi Batasan SameStart	136
6.2.6 Ilustrasi Batasan SameTime	137
6.2.7 Ilustrasi Batasan DifferentTime	137
6.2.8 Ilustrasi Batasan SameDays	138
6.2.9 Ilustrasi Batasan DifferentDays	138
6.2.10 Ilustrasi Batasan SameWeeks	139
6.2.11 Ilustrasi Batasan DifferentWeeks	139
6.2.12 Ilustrasi Batasan Overlap	140
6.2.13 Ilustrasi Batasan NotOverlap	140
6.2.14 Ilustrasi Batasan SameRoom	141
6.2.15 Ilustrasi Batasan DifferentRoom	141

6.2.16 Ilustrasi Batasan SameAttendees	142
6.2.17 Ilustrasi Batasan Precedence	143
6.2.18 Ilustrasi Batasan WorkDay	143
6.2.19 Ilustrasi Batasan MinGap	144
6.2.20 Ilustrasi Batasan MaxDays	145
6.2.21 Ilustrasi Batasan MaxDayLoad	146
6.2.22 Ilustrasi Batasan MaxBreaks	146
6.2.23 Ilustrasi Batasan MaxBlock	147
6.3.1 Grafik Perubahan Solusi dalam Proses Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2019</i>	151
6.3.2 Box plot Hasil Algoritma Optimasi pada Tiga Dataset: agh-fal17, muni-pdf-spr16c, dan muni-fsp-spr17c	155
6.3.3 Grafik Perubahan Solusi dalam Proses Optimasi pada <i>Benchmark ITC 2019</i>	156
6.3.4 Grafik Perubahan Solusi dalam Proses Local Search pada <i>Benchmark ITC 2019</i>	158
6.4.1 Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC 2019</i> pada Kategori Early	159
6.4.2 Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC 2019</i> pada Kategori Middle	160
6.4.3 Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC 2019</i> pada Kategori Late	162
6.4.4 Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC 2019</i> pada Seluruh Kategori	164
7.1.1 Ilustrasi Struktur Elemen Resources Pada ITC 2021	170
7.1.2 Ilustrasi Struktur Elemen Constraints Pada ITC 2021	171
7.3.1 Grafik Perubahan Solusi dalam Proses Pencarian Solusi <i>Feasible</i> pada <i>Benchmark ITC 2021</i>	178
7.3.2 Box plot Hasil Algoritma Optimasi pada Tiga Dataset: Late 12, Early 12 dan Early 2	180
7.3.3 Grafik Perubahan Solusi dalam Proses Optimasi pada <i>Benchmark ITC 2021</i>	184
7.3.4 Grafik Perubahan Solusi dalam Proses Local Search pada <i>Benchmark ITC 2021</i>	185

7.4.1	Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC</i> 2021 pada Kategori Early	191
7.4.2	Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC</i> 2021 pada Kategori Middle	192
7.4.3	Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC</i> 2021 pada Kategori Late	192
7.4.4	Perbandingan Peringkat Rata-Rata pada Penelitian dengan <i>Benchmark ITC</i> 2021 pada Keseluruhan Kategori	193
8.1.1	Struktur Dataset Ruangan Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum	198
8.1.2	Struktur Dataset Mahasiswa Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum	198
8.1.3	Struktur Dataset Asisten Pengajar Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum	198
8.3.1	Grafik Perubahan Solusi dalam Proses Pencarian Solusi <i>Feasible</i> pada Permasalahan Penjadwalan Ujian Praktikum	209
8.3.2	Grafik Perubahan Solusi dalam Proses Pencarian Solusi <i>Feasible</i> pada Permasalahan Penjadwalan Sesi Praktikum	210
8.3.3	Grafik Perubahan Solusi dalam Proses Optimasi pada Permasalahan Penjadwalan Ujian Praktikum	211
8.3.4	Grafik Perubahan Solusi dalam Proses Optimasi pada Permasalahan Penjadwalan Sesi Praktikum	212
8.3.5	Grafik Perubahan Solusi dalam Proses <i>Local Search</i> pada Permasalahan Penjadwalan Ujian Praktikum	212
8.3.6	Grafik Perubahan Solusi dalam Proses <i>Local Search</i> pada Permasalahan Penjadwalan Sesi Praktikum	213
8.4.1	Grafik Pergerakan Solusi dalam Proses Optimasi pada Algoritma dalam Penelitian Ini, LAHC dan HC	215
D.1	Preferensi Slot Waktu	264
D.2	Hasil Jadwal dengan Algoritma AT-ILS	265
D.3	Hasil Jadwal dengan Pembentukan Manual	266

Daftar Persamaan

2.1	Persamaan Waktu Polinomial	15
4.1	Persamaan probabilityNonRandomTimeSlot	85
4.2	Persamaan decreasingValue	85
5.1	Persamaan <i>soft constraint</i> Toronto 1	116
5.2	Persamaan <i>soft constraint</i> Toronto 2	116
6.1	Persamaan SameStart ITC 2019	136
6.2	Persamaan SameTime ITC 2019	136
6.3	Persamaan DifferentTime ITC 2019	137
6.4	Persamaan SameDays ITC 2019	137
6.5	Persamaan DifferentDays ITC 2019	138
6.6	Persamaan SameWeeks ITC 2019	138
6.7	Persamaan DifferentWeeks ITC 2019	139
6.8	Persamaan Overlap ITC 2019	139
6.9	Persamaan NotOverlap ITC 2019	140
6.10	Persamaan SameRoom ITC 2019	141
6.11	Persamaan DifferentRoom ITC 2019	141
6.12	Persamaan SameAttendees ITC 2019	142
6.13	Persamaan Presedence ITC 2019	142
6.14	Persamaan WorkDay ITC 2019	143
6.15	Persamaan MinGap ITC 2019	144
6.16	Persamaan MaxDays ITC 2019	144
6.17	Persamaan MaxDayLoad 1 ITC 2019	145
6.18	Persamaan MaxDayLoad 2 ITC 2019	145
6.19	Persamaan MaxDayLoad 3 ITC 2019	145
6.20	Persamaan MaxBreaks 1 ITC 2019	146

6.21	Persamaan MaxBreaks 2 ITC 2019	146
6.22	Persamaan MaxBlock ITC 2019	147
8.1	Persamaan Fungsi Objektif	199
8.2	Persamaan Batasan Mahasiswa dijadwalkan Tepat Satu Kali	200
8.3	Persamaan Batasan Kapasitas Ruangan	201
8.4	Persamaan Batasan Ketersediaan Slot Waktu Mahasiswa	201
8.5	Persamaan Batasan Ketersediaan Slot Waktu Asisten Pengajar	201
8.6	Persamaan Batasan Sesi Penjadwalan	202
8.7	Persamaan Batasan Jumlah Asisten Pengajar	202
8.8	Persamaan Batasan Penugasan Mahasiswa pada Satu Asisten Pengajar dan Satu Sesi	202
8.9	Persamaan Batasan Kapasitas Asisten Pengajar	203
8.10	Persamaan Batasan Penjadwalan Sesi (Tepat Satu Kombinasi Ruang-Waktu) .	203
8.11	Persamaan Batasan Ketersediaan Slot Waktu Mahasiswa	203
8.12	Persamaan Batasan Ketersediaan Slot Waktu Asisten Pengajar	203
8.13	Persamaan Batasan Jumlah Asisten per Sesi	204
8.14	Persamaan Penalti Distribusi Asisten Pengajar (Ujian)	204
8.15	Persamaan Kendala Linear Penalti Distribusi Asisten Pengajar (Ujian)	205
8.16	Persamaan Jumlah Ideal Mahasiswa per Asisten Pengajar	205
8.17	Persamaan Penalti Distribusi Asisten Pengajar (Sesi Praktikum)	205
8.18	Persamaan Kendala Linear Penalti Distribusi Asisten Pengajar (Sesi Praktikum)	206
8.19	Persamaan Penalti Preferensi waktu	206

Daftar Algoritma

1	Algoritma Iterated Local Search	32
2	Algoritma PA-ILS	84
3	Algoritma PA-ILS:Tahapan <i>Perturbation</i>	88
5	Algoritma Prosedur Pengacakan	90
6	Algoritma AT-ILS	92
8	Algoritma AT-ILS:Tahapan <i>Perturbation</i>	95
9	Algoritma AT-ILS:Tahapan <i>Local Search</i>	97
10	Algoritma Pembaruan <i>Threshold Local</i>	98
11	Algoritma Penerapan Heuristik	100
12	Algoritma Pembaruan <i>Threshold List</i>	101
13	Algoritma AT-ILS:Tahapan <i>Move Acceptance</i>	103

Halaman ini sengaja dikosongkan.

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Permasalahan penjadwalan adalah permasalahan untuk menjadwalkan sejumlah kegiatan ke dalam periode waktu yang telah ditentukan. Menyelesaikan permasalahan ini secara manual seringkali memakan waktu yang sangat lama, sehingga muncul gagasan untuk mengotomatiskan proses tersebut (Schaerf, 1999). Upaya pertama dalam menyelesaikan permasalahan penjadwalan secara otomatis dilakukan oleh Gotlieb (1963) melalui penelitiannya yang mengembangkan metode untuk menjadwalkan kelas dan guru. Penelitian tersebut menjadi titik awal dalam pengembangan penelitian untuk permasalahan penjadwalan. Hingga saat ini penelitian penjadwalan telah mencakup beberapa bidang, seperti pada bidang pendidikan (Bellio et al., 2021; Song et al., 2018), olahraga (Kyngäs et al., 2017; Liang et al., 2021) dan transportasi (Jamili et al., 2012; Wu et al., 2015).

Dalam permasalahan penjadwalan, umumnya terdapat dua jenis batasan (kriteria) yang wajib dan diupayakan untuk dipenuhi, yaitu *hard constraint* dan *soft constraint* (Babaei et al., 2015; Tan et al., 2021). *Hard constraint* merupakan batasan yang wajib untuk dipenuhi. Contoh dari batasan ini adalah kapasitas ruangan. Jika terdapat ruangan dengan kapasitas 20 orang, maka suatu solusi penjadwalan tidak boleh menjadwalkan lebih dari 20 orang pada ruangan tersebut. Pelanggaran *hard constraint* menyebabkan solusi penjadwalan menjadi tidak memungkinkan untuk digunakan atau yang diistilahkan sebagai solusi yang tidak *feasible* (Chen et al., 2021). Oleh karena itu, *hard constraint* menjadi batasan mutlak yang tidak dapat dilanggar dalam menghasilkan solusi penjadwalan (Babaei et al., 2015).

Sementara itu, *soft constraint* adalah batasan yang diupayakan untuk dipenuhi, namun

memungkinkan untuk dilanggar (Pandey and Sharma, 2016). Pelanggaran terhadap *soft constraint* akan menyebabkan menurunnya kualitas dari solusi penjadwalan yang dihasilkan (Fong et al., 2015; Rezaeipanah et al., 2021). Sebagai contoh, *soft constraint* dalam permasalahan penjadwalan adalah preferensi waktu belajar. Misalnya, mayoritas mahasiswa tidak menyukai kelas yang dijadwalkan di pagi hari karena khawatir terlambat. Ketika memaksakan menjadwalkan kelas di pagi hari, maka kemungkinan untuk mahasiswa terlambat akan semakin besar. Pada akhirnya solusi penjadwalan ini dapat menurunkan kualitas pembelajaran. Beberapa penelitian juga telah menunjukkan pentingnya mengoptimalkan solusi penjadwalan dengan meminimalkan pelanggaran terhadap *soft constraint*. Penelitian oleh Köksalmış et al. (2014) menunjukkan bahwa penjadwalan otomatis yang mampu menghasilkan solusi penjadwalan yang lebih berkualitas terbukti dapat mengurangi kelelahan peserta ujian dan menekan risiko kecurangan. Penelitian oleh Sloan et al. (2020) dan Larabi-Marie-Sainte et al. (2021) juga menemukan bahwa ketidakcocokan jadwal dengan preferensi mahasiswa merupakan salah satu faktor yang menyebabkan mahasiswa tidak menghadiri kelas. Oleh karena itu, melakukan optimasi untuk mengurangi pelanggaran terhadap *soft constraint* menjadi penting untuk dilakukan dalam menghasilkan solusi penjadwalan.

Berdasarkan dua jenis batasan yang telah dijabarkan, permasalahan penjadwalan memiliki dua permasalahan utama yang harus diselesaikan. Permasalahan pertama adalah bagaimana menghasilkan solusi *feasible* dengan tidak melanggar satupun *hard constraint* yang ada. Permasalahan kedua adalah bagaimana mengoptimasi solusi penjadwalan dengan mengurangi jumlah pelanggaran terhadap *soft constraint*, namun tetap menjaga agar solusi *feasible*. Kedua permasalahan ini menjadi fokus utama pada berbagai penelitian penjadwalan (Abdelhalim and El Khayat, 2016; Bellio et al., 2021; Goh et al., 2020).

Dalam upaya penyelesaian kedua permasalahan penjadwalan, kebanyakan penelitian mengembangkan algoritma menggunakan metode heuristik. Hal ini disebabkan oleh permasalahan penjadwalan yang digolongkan sebagai permasalahan *Non-deterministic Polynomial Hard* (NP-Hard) (Bettinelli et al., 2015; Oude Vrielink et al., 2019). Permasalahan NP-Hard merupakan kelas permasalahan yang sulit untuk diselesaikan dalam waktu yang wajar (*polynomial time*) karena kompleksitasnya yang tumbuh secara eksponensial seiring dengan bertambahnya ukuran masalah (Babaei et al., 2015; Li et al., 2020). Sementara itu, penggunaan metode heuristik mampu menghasilkan solusi yang

cukup baik pada permasalahan yang tergolong dalam permasalahan NP-Hard dan dalam waktu yang wajar (Oude Vrielink et al., 2019; Thepphakorn and Pongcharoen, 2020). Hal ini yang menyebabkan banyak penelitian memilih menggunakan metode ini dalam menyelesaikan permasalahan penjadwalan (Premananda et al., 2024).

Meskipun metode heuristik telah berhasil menghasilkan solusi yang cukup baik dalam waktu yang wajar, metode ini memiliki keterbatasan, yaitu tidak adanya jaminan bahwa solusi yang dihasilkan merupakan solusi yang optimal. Keterbatasan ini menjadi ruang untuk mengembangkan algoritma baru yang mungkin dapat menghasilkan solusi yang lebih baik. Hal ini mendorong penelitian-penelitian dalam permasalahan penjadwalan berkembang dengan memunculkan berbagai pengembangan algoritma baru dengan harapan dapat menghasilkan hasil yang lebih baik dari penelitian sebelumnya (Al-Betar, 2021; Bellio et al., 2021; Rosati et al., 2022).

Pengembangan algoritma heuristik baru pada permasalahan penjadwalan umumnya berdasarkan pada tiga jenis pendekatan yaitu metaheuristik, hibrid metaheuristik dan hiper-heuristik (Premananda et al., 2024). Pendekatan metaheuristik merupakan pendekatan yang lebih general dibandingkan dengan pendekatan heuristik. Pendekatan heuristik umumnya ditujukan pada beberapa permasalahan spesifik (Dokeroglu et al., 2019; Hussain et al., 2019). Sementara pendekatan metaheuristik dapat diterapkan terhadap berbagai jenis permasalahan. Sebagai contoh algoritma Nearest Neighbor yang merupakan algoritma heuristik dapat menghasilkan solusi yang baik pada permasalahan *Traveling Salesman Problem* (TSP) dengan ukuran permasalahan kecil, namun menghasilkan solusi yang buruk jika diterapkan pada permasalahan TSP berukuran sedang atau besar dan tidak bisa diterapkan pada permasalahan optimasi lainnya seperti permasalahan penjadwalan (AlSalibi et al., 2013). Sementara itu algoritma Simulated Annealing yang merupakan algoritma metaheuristik dapat diterapkan pada berbagai jenis permasalahan optimasi seperti pada TSP, permasalahan penjadwalan, permasalahan *knapsack* dan permasalahan optimasi lainnya (Bellio et al., 2021; Dong et al., 2023; Moradi et al., 2022).

Pendekatan kedua yaitu hibrid metaheuristik merupakan pendekatan lanjutan dari pendekatan metaheuristik dengan menggabungkan beberapa algoritma metaheuristik dengan tujuan untuk dapat memaksimalkan kelebihan dan menutupi kekurangan dari

setiap algoritma yang dikombinasikan (Tan et al., 2021). Pendekatan ketiga yaitu hiper-heuristik. Pendekatan ini merupakan pengembangan dari pendekatan metaheuristik dengan lebih ditujukan untuk meningkatkan generalitas dari suatu algoritma (Drake et al., 2020). Pendekatan metaheuristik memang dapat diterapkan pada berbagai jenis permasalahan, namun solusi yang dihasilkan tidak konsisten antara satu permasalahan dengan permasalahan lainnya. Hal ini menyebabkan pendekatan hiper-heuristik dikembangkan untuk menyelesaikan hal tersebut dengan mengembangkan algoritma yang mampu menghasilkan solusi yang konsisten baik pada berbagai jenis permasalahan (Dokeroglu et al., 2024; Sánchez et al., 2020).

Dari penelitian-penelitian yang dipublikasikan, terdapat banyak penelitian yang menunjukkan hasil yang sangat baik dalam menghasilkan solusi *feasible* dan dalam mengoptimasi solusi penjadwalan (Premananda et al., 2024). Namun kebanyakan penelitian hanya mengembangkan algoritmanya pada suatu studi kasus yang spesifik. Saat algoritma yang sama diterapkan pada permasalahan penjadwalan yang berbeda, kecenderungannya tidak mendapatkan hasil yang sama baiknya seperti yang dilaporkan dalam penelitian (Pillay, 2016b). Oleh karena itu, untuk dapat menghasilkan solusi yang baik pada suatu studi kasus baru, diperlukan pemilihan algoritma yang sesuai beserta pengaturan ulang nilai parameter untuk menyesuaikan dengan permasalahan yang ada. Proses pemilihan dan pengaturan nilai parameter hanya dapat dilakukan oleh orang yang memiliki keahlian di bidang optimasi. Hal ini yang menyebabkan sulit untuk mengadaptasi algoritma yang telah dikembangkan dalam berbagai penelitian untuk penggunaan secara praktis (Pillay, 2016b). Hal ini menyebabkan pengembangan algoritma yang bersifat lebih generik dan mampu menghasilkan hasil yang konsisten baik pada permasalahan yang berbeda dengan tidak membutuhkan terlalu banyak perubahan dalam penerapannya menjadi penting untuk dikembangkan (Gümüş et al., 2023).

Dalam upaya pengembangan algoritma generik, pengembangan dibagi dalam beberapa tingkatan berdasarkan tingkat generalitasnya (Pillay, 2016b). Pada tingkatan pertama, algoritma mampu menghasilkan solusi yang konsisten baik pada sejumlah dataset yang tergolong dalam satu *benchmark* dan satu permasalahan yang sama. Selanjutnya, tingkatan kedua adalah tingkatan dimana algoritma mampu menghasilkan solusi yang konsisten baik pada dua *benchmark* atau lebih dalam jenis permasalahan yang sama. Terakhir, tingkatan ketiga yaitu algoritma mampu menghasilkan solusi yang konsisten baik

pada beberapa jenis permasalahan penjadwalan yang berbeda atau yang diistilahkan sebagai permasalahan lintas domain.

Berdasarkan survei yang dilakukan oleh Premananda et al. (2024), terdapat beberapa penelitian yang mampu menghasilkan solusi yang konsisten baik pada tingkatan generalitas pertama dan kedua. Pada tingkatan generalitas pertama, dalam bidang penjadwalan ujian, penelitian oleh Bellio et al. (2021) mampu menghasilkan solusi yang konsisten pada 12 dataset dalam *benchmark* Toronto dan unggul terhadap 26 penelitian lainnya. Penelitian lainnya oleh Bykov and Petrovic (2016) menghasilkan hasil yang konsisten baik pada 12 dataset dalam *benchmark International Timetabling Competition* (ITC) 2007 Track 1 dan unggul terhadap 17 penelitian lainnya. Pada tingkatan generalitas kedua, penelitian oleh Goh et al. (2020) dan Nagata (2018a) menunjukkan hasil yang konsisten baik pada dua jenis *benchmark*, yaitu Socha dan ITC 2007 Track 2, dalam menyelesaikan permasalahan penjadwalan mata kuliah *post-enrollment* dan menempati urutan teratas dibandingkan dengan penelitian lainnya.

Sementara itu, pada tingkatan generalitas ketiga, terdapat dua penelitian dengan hasil yang konsisten meskipun tidak sebaik hasil dari penelitian pada tingkatan generalitas pertama dan kedua. Penelitian oleh Fong et al. (2014a) menunjukkan konsistensi hasil pada dua jenis permasalahan. Pada permasalahan pertama, yaitu penjadwalan ujian menggunakan *benchmark* Toronto, penelitian ini menghasilkan solusi yang lebih baik dibandingkan dengan 20 penelitian lainnya dari 25 penelitian yang dibandingkan. Pada permasalahan kedua, yaitu penjadwalan mata kuliah menggunakan *benchmark* Socha, penelitian ini menunjukkan keunggulan dibandingkan dengan 6 penelitian dari 13 penelitian yang dilakukan perbandingan. Penelitian lain oleh Abdullah and Turabieh (2012) juga menghasilkan hasil yang baik pada dua jenis permasalahan. Pada permasalahan pertama, yaitu penjadwalan ujian menggunakan *benchmark* ITC 2007 Track 1, penelitian ini unggul terhadap 8 penelitian lainnya dari 16 penelitian yang dibandingkan. Pada permasalahan kedua, yaitu penjadwalan mata kuliah berbasis kurikulum menggunakan *benchmark* ITC 2007 Track 3, penelitian ini menghasilkan solusi yang lebih baik pada 5 penelitian dari 6 penelitian dilakukan perbandingan.

Walaupun kedua penelitian tersebut menunjukkan hasil yang konsisten, namun terdapat beberapa keterbatasan. Pertama, kedua penelitian ini hanya diuji pada dua jenis

permasalahan penjadwalan, yaitu penjadwalan ujian dan penjadwalan mata kuliah. Hal ini membatasi bukti generalitas algoritma dalam menyelesaikan permasalahan penjadwalan yang lebih beragam, yang juga relevan dalam konteks penjadwalan. Sebagai contoh, terdapat jenis permasalahan penjadwalan lain, seperti penjadwalan kompetisi olahraga, yang memiliki karakteristik dan kendala yang berbeda jauh dengan penjadwalan pada bidang edukasi.

Kedua, penelitian ini menggunakan dataset lama, seperti *benchmark* Toronto yang dipublikasikan pada tahun 1996, *benchmark* Socha yang dipublikasikan pada tahun 2003 dan *benchmark* ITC 2007 yang dipublikasikan pada tahun 2007. Ketiga *benchmark* tersebut memiliki perbedaan signifikan terutama pada jumlah jenis batasan terhadap permasalahan penjadwalan saat ini. Sebagai contoh salah satu permasalahan penjadwalan mata kuliah terbaru yaitu ITC 2019 yang mengumpulkan dataset yang diambil dari permasalahan nyata, memiliki jumlah batasan sebanyak 18 jenis. Sementara itu penjadwalan mata kuliah pada *benchmark* socha hanya memiliki lima jenis batasan dan pada ITC 2007 hanya memiliki 7 dan 8 jenis batasan. Ukuran permasalahan juga menunjukkan perbedaan signifikan. ITC 2019 pada dataset terbesarnya memiliki 8,813 kelas dan 38,437 mahasiswa yang harus dijadwalkan. Sementara pada Socha hanya memiliki 400 kelas dan 400 mahasiswa dan pada ITC 2007 hanya 400 kelas dan 1000 mahasiswa. Oleh karena itu, hasil kedua penelitian tersebut tidak mencerminkan kompleksitas dan kendala permasalahan penjadwalan saat ini.

Berdasarkan pentingnya pengembangan algoritma generik untuk menyelesaikan permasalahan penjadwalan lintas domain, penelitian ini mengembangkan algoritma generik yang dirancang untuk menangani berbagai jenis permasalahan penjadwalan. Algoritma ini dikembangkan menggunakan metode hiper-heuristik dengan pengembangan berdasarkan bentuk dasar dari algoritma *Iterated Local Search*. Metode hiper-heuristik yang merupakan metode untuk meningkatkan generalitas algoritma merupakan metode yang sesuai dengan tujuan dari pengembangan algoritma ini. Sementara itu algoritma ILS memiliki struktur yang sederhana dan tidak terlalu membutuhkan pengaturan nilai parameter. Hal ini menyebabkan algoritma ILS sesuai untuk digunakan sebagai dasar pengembangan dalam penelitian ini. Hasil pengembangan algoritma diuji coba pada berbagai jenis permasalahan penjadwalan, seperti penjadwalan ujian dengan *benchmark* Toronto, penjadwalan mata kuliah dengan *benchmark* ITC 2019, penjadwalan kompetisi

olahraga dengan *benchmark* ITC 2021, serta dua permasalahan penjadwalan pada Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS). Pemilihan studi kasus ini bertujuan untuk membuktikan generalitas algoritma yang dikembangkan, dengan menguji pada berbagai jenis dan ukuran permasalahan.

1.2 Perumusan Masalah

Mengembangkan algoritma untuk menyelesaikan permasalahan penjadwalan lintas domain menjadi tantangan penting karena kebutuhan praktis yang belum terpenuhi (Pillay, 2016b). Sejauh ini, belum ada penelitian yang secara efektif mampu mengatasi permasalahan penjadwalan lintas domain (Premananda et al., 2024). Dalam menyelesaikan permasalahan penjadwalan lintas domain, terdapat dua tantangan utama yang juga merupakan tantangan umum dalam permasalahan penjadwalan. Tantangan pertama adalah bagaimana mengembangkan algoritma yang dapat menghasilkan solusi penjadwalan yang *feasible* tanpa melanggar *hard constraint* (Babaei et al., 2015; Chen et al., 2021). Tantangan kedua adalah bagaimana mengoptimasi solusi penjadwalan namun tetap mempertahankan kelayakan solusi (*feasible*) (Pandey and Sharma, 2016). Kedua tantangan ini menjadi lebih sulit dalam konteks penjadwalan lintas domain, karena terdapat variasi jenis dan ukuran permasalahan.

Secara teori, salah satu metode yang menjanjikan adalah pengembangan algoritma menggunakan metode hiper-heuristik. Metode ini dirancang untuk menghasilkan algoritma generik dan dapat diterapkan pada berbagai jenis permasalahan (Drake et al., 2020). Namun, dalam penerapannya pada permasalahan penjadwalan, pengembangan algoritma hiper-heuristik untuk permasalahan penjadwalan lintas domain masih menghadapi banyak tantangan. Hingga saat ini, belum ada keberhasilan signifikan dalam menyelesaikan permasalahan penjadwalan lintas domain menggunakan metode ini (Premananda et al., 2024).

Selain itu, dalam menyelesaikan permasalahan lintas domain, tujuan dari algoritma yang dikembangkan tidak hanya sekadar mampu menghasilkan solusi *feasible* dengan kualitas solusi yang baik. Solusi yang dihasilkan juga harus konsisten berkualitas baik antar dataset dan jenis permasalahan yang berbeda. Untuk mengukur hal tersebut, hasil pengembangan algoritma perlu dibandingkan dengan hasil dari penelitian lainnya. Hal ini

penting karena, pada permasalahan NP-Hard seperti permasalahan penjadwalan, solusi optimal dari suatu studi kasus umumnya tidak dapat diketahui dengan pasti. Oleh karena itu, perbandingan dengan hasil dari penelitian lain menjadi salah satu cara untuk mengevaluasi performa dari algoritma yang dikembangkan.

Berdasarkan uraian tersebut, penelitian ini merumuskan tiga rumusan masalah sebagai berikut:

- Bagaimana mengembangkan algoritma dengan metode hiper-heuristik untuk menghasilkan solusi *feasible* pada permasalahan penjadwalan lintas domain?
- Bagaimana mengembangkan algoritma dengan metode hiper-heuristik untuk mengoptimasi solusi pada permasalahan penjadwalan lintas domain?
- Bagaimana performa algoritma yang dikembangkan terhadap berbagai jenis permasalahan, jika dibandingkan dengan penelitian sebelumnya?

1.3 Tujuan dan Manfaat Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah dijabarkan, penelitian memiliki tiga tujuan yaitu :

- Mengembangkan algoritma hiper-heuristik yang mampu menghasilkan solusi penjadwalan yang *feasible* pada permasalahan penjadwalan lintas domain.
- Mengembangkan algoritma hiper-heuristik yang mampu mengoptimasi solusi penjadwalan pada permasalahan penjadwalan lintas domain.
- Mengevaluasi performa algoritma yang dikembangkan dengan membandingkan hasil penelitian dengan hasil dari penelitian lain pada beberapa *benchmark* dalam permasalahan penjadwalan.

Sementara itu, penelitian ini diharapkan mampu menghasilkan dua manfaat:

- Secara teoritis: Penelitian ini diharapkan dapat menyumbangkan pengetahuan baru dalam pengembangan algoritma untuk permasalahan penjadwalan lintas domain dan menghasilkan solusi baru pada beberapa dataset dalam *benchmark* permasalahan penjadwalan. Selain itu penelitian ini juga menyediakan *benchmark* dataset baru pada dua permasalahan nyata. Terakhir, penelitian ini menyediakan data

perbandingan performa antara penelitian pada beberapa *benchmark* yang berguna untuk digunakan dalam penelitian kedepannya.

- Secara Praktis: Penelitian ini diharapkan menghasilkan algoritma yang dapat diterapkan di berbagai jenis kasus permasalahan nyata, seperti pada penjadwalan ujian, mata kuliah, kompetisi olahraga dan permasalahan penjadwalan lainnya dengan performa yang konsisten dan hasil yang baik tanpa memerlukan banyak penyesuaian dengan studi kasus yang ada.

1.4 Kontribusi dan Orisinalitas Penelitian

Penelitian ini memberikan beberapa kontribusi utama:

1. Mengembangkan algoritma baru menggunakan metode hiper-heuristik berbasis algoritma ILS untuk menghasilkan solusi *feasible* pada permasalahan penjadwalan lintas domain.
2. Mengembangkan algoritma baru menggunakan metode hiper-heuristik berbasis algoritma ILS untuk mengoptimasi solusi penjadwalan pada berbagai jenis permasalahan lintas domain.
3. Menghasilkan solusi baru yang kompetitif pada tiga *benchmark*, yaitu Toronto (penjadwalan ujian), ITC 2019 (penjadwalan mata kuliah) dan ITC 2021 (penjadwalan kompetisi olahraga).
4. Mengembangkan model matematis dan dataset *benchmark* untuk permasalahan penjadwalan sesi praktikum dan ujian praktikum di Departemen Sistem Informasi ITS

Orisinalitas dari penelitian ini terletak pada pengembangan dua algoritma baru berbasis algoritma ILS dengan metode hiper-heuristik, yang masing-masing dirancang untuk menghasilkan solusi *feasible* dan mengoptimasi solusi penjadwalan lintas domain. Kedua algoritma ini menawarkan pendekatan yang lebih generik dan fleksibel, serta dapat diterapkan pada berbagai jenis dataset dan ukuran masalah.

1.5 Batasan Penelitian

Untuk memfokuskan permasalahan dalam penelitian ini, terdapat batasan penelitian yang berupa jumlah jenis studi kasus yang digunakan dan juga limitasi eksperimen untuk setiap studi kasus. Batasan tersebut terdiri dari:

1. Algoritma yang diuji coba pada enam studi kasus yang terdiri dari:
 - (a) Permasalahan penjadwalan ujian menggunakan *benchmark* Toronto yang terdiri dari 13 dataset yang tersedia.
 - (b) Permasalahan penjadwalan mata kuliah menggunakan 30 dataset pada *benchmark* ITC 2019.
 - (c) Permasalahan penjadwalan kompetisi olah raga menggunakan 45 dataset pada *benchmark* ITC 2021.
 - (d) Permasalahan penjadwalan pada departemen Sistem Informasi ITS yang terdiri dari :
 - Permasalahan penjadwalan sesi praktikum dengan dua dataset dari semester 1 tahun 2023 dan semester 2 tahun 2024.
 - Permasalahan penjadwalan ujian praktikum dengan dua dataset dari semester 1 tahun 2023 dan semester 2 tahun 2024.
2. Jumlah uji coba dalam tahapan mencari solusi *feasible* maupun tahapan optimasi dilakukan sebanyak 10 kali pada setiap dataset. Untuk tahapan mencari solusi *feasible*, durasi maksimal setiap uji coba dibatasi 120 jam. Sementara pada tahapan optimasi, tidak ada durasi tetap yang membatasi. Namun ketika tidak ada peningkatan solusi selama 10 jam, maka proses optimasi diakhiri.

1.6 Publikasi

Berikut adalah daftar publikasi yang telah berhasil diterbitkan selama proses penelitian ini berlangsung:

1. I. G. A. Premananda, A. Tjahyanto and A. Muklason, "Hybrid Whale Optimization Algorithm for Solving Timetabling Problems of ITC 2019," 2022 IEEE International

- Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Malang, Indonesia, 2022, pp. 317-322, doi: 10.1109/CyberneticsCom55287.2022.9865647.
2. I. G. Agung Premananda, A. Tjahyanto and A. Mukhlason, "Design Science Research Methodology and Its Application to Developing a New Timetabling Algorithm," 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Malang, Indonesia, 2022, pp. 433-438, doi: 10.1109/CyberneticsCom55287.2022.9865661.
 3. Premananda, I.G.A., Tjahyanto, A. and Mukhlason, A. Efficient iterated local search based metaheuristic approach for solving sports timetabling problems of International Timetabling Competition 2021. Ann Oper Res (2024). <https://doi.org/10.1007/s10479-024-06285-x>
 4. I. Gusti Agung Premananda, A. Tjahyanto and A. Muklason, "Timetabling Problems and the Effort Toward Generic Algorithms: A Comprehensive Survey," in IEEE Access, vol. 12, pp. 143854-143868, 2024, doi: 10.1109/ACCESS.2024.3463721.

1.7 Sistematika Penulisan

Sistematika penulisan dokumen disertasi ini dibagi menjadi sembilan bab yang terstruktur sebagai berikut:

Bab 1 merupakan pengantar yang memberikan gambaran umum tentang penelitian ini. Bab ini mencakup latar belakang penelitian, perumusan masalah, tujuan dan manfaat penelitian, kontribusi serta orisinalitas penelitian, batasan penelitian, publikasi dan sistematika penulisan dokumen.

Bab 2 berfokus pada landasan teori yang mendukung penelitian ini. Selain membahas teori dasar, bab ini juga mengulas penelitian terdahulu dengan membaginya ke dalam tiga bagian, yaitu: (1) penelitian terkait tujuh *benchmark* yang membahas permasalahan penjadwalan, (2) pembahasan mengenai tingkat generalitas algoritma pada penelitian sebelumnya dan (3) pengembangan algoritma yang secara khusus diarahkan pada pencarian solusi *feasible*. Terakhir, bab ini menyajikan analisis kebaruan penelitian berdasarkan hasil kajian literatur.

Bab 3 menjelaskan metodologi penelitian, yang terdiri dari lima tahapan utama: identifikasi permasalahan, penentuan kebutuhan solusi, perancangan dan pengembangan algoritma, demonstrasi algoritma, serta evaluasi algoritma.

Bab 4 membahas secara mendalam tentang desain dan pengembangan algoritma. Pembahasan dimulai dengan penjelasan rasionalitas desain algoritma yang dikembangkan, diikuti dengan detail desain algoritma untuk pencarian solusi *feasible* dan optimasi solusi. Selain itu, bab ini juga menjelaskan Low-Level Heuristic yang digunakan serta pengaturan parameter untuk algoritma pencarian solusi *feasible*.

Bab 5 memaparkan implementasi algoritma pada permasalahan penjadwalan ujian menggunakan *benchmark* Toronto. Pembahasan mencakup deskripsi karakteristik dataset, penjelasan *hard constraint* dan *soft constraint*, hasil implementasi algoritma, serta analisis kinerja algoritma dalam menyelesaikan permasalahan. Selain itu, hasil implementasi algoritma dibandingkan dengan penelitian sebelumnya. Bab ini ditutup dengan kesimpulan dari implementasi yang dilakukan.

Bab 6, 7 dan 8 membahas implementasi algoritma pada tiga jenis permasalahan lainnya. Bab 6 membahas implementasi pada penjadwalan mata kuliah menggunakan *benchmark* ITC 2019. Bab 7 membahas implementasi pada penjadwalan kompetisi olahraga menggunakan *benchmark* ITC 2021. Bab 8 membahas implementasi pada dua permasalahan nyata, yaitu penjadwalan sesi praktikum dan penjadwalan ujian praktikum di Departemen Sistem Informasi ITS. Ketiga bab ini memiliki struktur pembahasan yang serupa dengan Bab kelima.

Bab 9 menyajikan kesimpulan dan saran untuk penelitian selanjutnya. Bagian kesimpulan merangkum hasil implementasi pada keempat jenis permasalahan dan menjawab apakah algoritma yang dikembangkan mampu menyelesaikan permasalahan penjadwalan lintas domain. Bagian saran memaparkan keterbatasan penelitian ini serta potensi pengembangan lebih lanjut berdasarkan hasil yang telah diperoleh.

BAB 2

KAJIAN PUSTAKA

Bab ini membahas tiga topik yang menjadi dasar pengembangan penelitian. Bagian pertama menjelaskan pustaka yang digunakan sebagai landasan ilmiah penelitian dalam pengembangan algoritma penjadwalan lintas domain. Selanjutnya, bagian kedua mengulas penelitian terdahulu yang relevan dengan permasalahan penjadwalan, terutama pada penelitian yang menggunakan *benchmark*, penelitian terkait generalitas algoritma, serta penelitian yang berfokus pada tahapan pencarian solusi *feasible*. Terakhir, bagian ketiga membahas analisis kebaruan penelitian ini berdasarkan kajian terhadap hasil penelitian sebelumnya. Ketiga topik tersebut dijelaskan lebih rinci secara berurutan pada Subbab 2.1, 2.2 dan 2.3.

2.1 Dasar Teori

Penelitian ini bertujuan untuk mengembangkan algoritma hiper-heuristik berbasis *Iterated Local Search* (ILS) untuk menyelesaikan permasalahan penjadwalan lintas domain. Untuk menjalankan penelitian ini, diperlukan pemahaman terhadap teori-teori dasar terkait permasalahan penjadwalan, tingkat kompleksitasnya, serta pendekatan-pendekatan yang dapat digunakan untuk menghasilkan solusi yang *feasible* dan optimal.

Secara rinci, terdapat tujuh teori dasar yang menjadi landasan penelitian ini. Teori pertama adalah klasifikasi permasalahan komputasi, yang digunakan untuk memahami kompleksitas permasalahan penjadwalan dari perspektif komputasi. Teori kedua membahas metode aproksimasi dan metaheuristik, yang menjelaskan alasan metode aproksimasi dianggap sesuai untuk menyelesaikan permasalahan penjadwalan.

Selanjutnya, teori ketiga berfokus pada permasalahan penjadwalan, yang menguraikan secara lebih mendalam karakteristik dan tantangan yang terkait dengan permasalahan penjadwalan. Teori keempat adalah teori yang membahas bagaimana ruang solusi pencarian dalam permasalahan optimasi. Teori kelima adalah tingkatan generalitas algoritma, yang memberikan pemahaman tentang tingkat generalitas, terutama dalam konteks permasalahan penjadwalan lintas domain. Berikutnya, teori keenam membahas hiper-heuristik, yaitu metode pengembangan algoritma yang bertujuan menghasilkan algoritma yang lebih generik. Terakhir, teori ketujuh adalah algoritma ILS, yang diperlukan untuk memahami bentuk dasar dan prinsip kerja dari algoritma tersebut. Secara rinci, keenam teori ini dibahas secara berurutan pada Subbab 2.1.1 hingga 2.1.7.

2.1.1 Klasifikasi Permasalahan Komputasi dan Kompleksitas

Dalam pembahasan teori klasifikasi permasalahan komputasi, terutama untuk memahami kompleksitas permasalahan penjadwalan, terdapat dua aspek utama yang perlu dipahami. Aspek pertama adalah perbedaan antara kategori permasalahan *polinomial* (P) dan *non-deterministic polinomial* (NP), yang dilihat dari kemampuan algoritma untuk menyelesaikan permasalahan dalam waktu polinomial. Aspek kedua berkaitan dengan kategori lanjutan dari NP, yaitu *Non-deterministic Polynomial Complete* (NP-Complete) dan *Non-deterministic Polynomial Hard* (NP-Hard), termasuk bagaimana permasalahan penjadwalan dapat diklasifikasikan ke dalam kategori NP-Complete atau NP-Hard. Kedua aspek ini dijelaskan secara rinci secara berurutan pada Subbab 2.1.1.1 hingga 2.1.1.5.

2.1.1.1 Permasalahan P dan NP

Dalam teori kompleksitas komputasi, masalah dapat dikategorikan ke dalam dua kelas besar: P dan NP (Goldreich, 2010; Sipser, 1996). Kelas P mencakup masalah-masalah yang dapat diselesaikan oleh algoritma dalam waktu polinomial. Ini berarti bahwa waktu komputasi untuk menyelesaikan masalah tersebut tumbuh secara polinomial dengan bertambahnya ukuran input (Penjelasan detail waktu polinomial terdapat pada Subbab 2.1.1.2). Sebagai contoh, permasalahan mengurutan data dengan algoritma *Merge Sort* atau *Quick Sort* dapat diselesaikan dalam waktu polinomial.

Di sisi lain, kelas NP mencakup permasalahan di mana kebenaran solusi dapat diverifikasi dalam waktu polinomial, meskipun mungkin tidak ada algoritma yang dapat

menemukan solusi tersebut dalam waktu polinomial. Salah satu contoh dari masalah NP adalah permasalahan penjadwalan, khususnya ketika tujuan utamanya hanya untuk menemukan solusi *feasible* dengan memenuhi semua *hard constraint*. Dalam masalah ini, terdapat sejumlah batasan, seperti memastikan tidak ada dua ujian yang dijadwalkan pada waktu yang sama untuk mahasiswa yang sama, serta memastikan setiap ujian dijadwalkan dalam ruang yang sesuai. Meskipun menemukan solusi *feasible* secara langsung tetap menjadi tantangan karena banyaknya kemungkinan kombinasi jadwal yang harus diperiksa, namun ketika sebuah solusi diberikan, solusi tersebut dapat diverifikasi dalam waktu polinomial. Hal ini dikarenakan verifikasi dilakukan dengan hanya melakukan pengecekan satu per satu untuk menentukan apakah ada konflik atau tidak. Oleh karena itu, masalah penjadwalan berada dalam kelas NP, di mana solusi sulit ditemukan, tetapi mudah diverifikasi (Mühlenthaler and Mühlenthaler, 2015).

2.1.1.2 Waktu Polinomial

Waktu polinomial adalah waktu komputasi di mana jumlah langkah yang diperlukan oleh sebuah algoritma untuk menyelesaikan masalah bertambah sesuai dengan fungsi polinomial berdasarkan input dari ukuran permasalahan. Secara umum, algoritma dikatakan memiliki waktu polinomial jika kompleksitas waktunya dapat diekspresikan dalam bentuk:

$$T(n) = O(n^k) \quad (2.1)$$

di mana:

- $T(n)$ adalah jumlah langkah yang dibutuhkan oleh algoritma untuk menyelesaikan masalah dengan ukuran input n ,
- $O(n^k)$ adalah notasi "Big-O" yang merepresentasikan batas atas kompleksitas waktu,
- n adalah ukuran input (jumlah data yang diproses),
- k adalah konstanta yang menentukan derajat polinomial.

Algoritma dengan waktu polinomial dianggap efisien karena pertumbuhan waktu

komputasinya masih dapat dikelola meskipun ukuran input membesar. Hal ini berbeda dengan permasalahan yang pertumbuhan waktu komputasinya eksponensial $O(2^n)$, di mana waktu komputasi tumbuh sangat cepat seiring dengan bertambahnya ukuran input permasalahan (Goldreich, 2010).

Sebagai contoh, algoritma *Merge Sort* adalah contoh algoritma yang dapat menyelesaikan masalah pengurutan dalam waktu polinomial. *Merge Sort* menggunakan pendekatan *divide and conquer* untuk mengurutkan elemen-elemen dalam sebuah array. Algoritma ini bekerja dengan cara membagi array menjadi dua bagian secara rekursif, kemudian menggabungkan dua bagian yang sudah diurutkan.

Kompleksitas waktu dari *Merge Sort* adalah $O(n \log n)$. Penjelasan detailnya sebagai berikut:

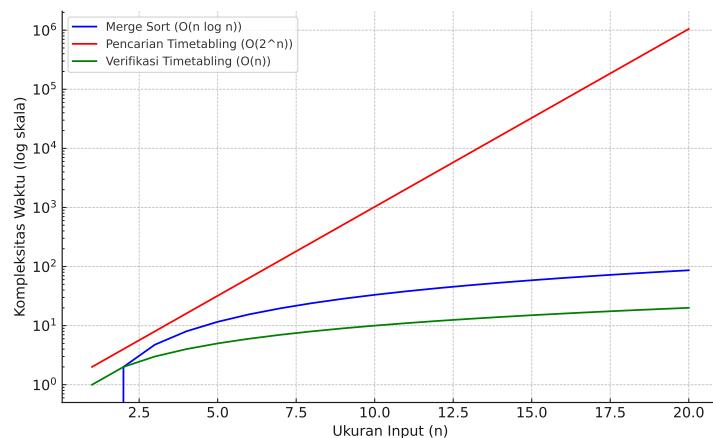
- Pembagian Array: Pada setiap tahap rekursi, array dibagi menjadi dua bagian, yang menghasilkan $\log n$ tahap rekursi.
- Penggabungan Subarray: Pada setiap tahap rekursi, penggabungan dua subarray membutuhkan waktu $O(n)$ untuk memproses seluruh elemen.

Jadi, total kompleksitas waktu dari *Merge Sort* adalah $O(n \log n)$, yang termasuk dalam waktu polinomial.

Sementara itu, banyak varian masalah penjadwalan (misalnya penjadwalan ujian dan mata kuliah) dikategorikan ke dalam kelas masalah NP. Untuk menemukan solusi *feasible* secara langsung merupakan tugas yang sangat sulit dan tidak dapat diselesaikan dalam waktu polinomial. Jumlah kemungkinan kombinasi jadwal yang harus diperiksa bertambah secara eksponensial seiring dengan bertambahnya jumlah kegiatan dan batasan. Oleh karena itu, pencarian solusi memerlukan waktu eksponensial $O(2^n)$, yang membuat masalah ini tidak dapat diselesaikan dalam waktu polinomial.

Di sisi lain, verifikasi solusi untuk masalah penjadwalan dapat dilakukan dengan mudah dengan memeriksa setiap elemen jadwal satu per satu, guna memastikan tidak ada konflik (misalnya, dua ujian yang dijadwalkan pada waktu yang sama untuk mahasiswa yang sama). Proses verifikasi ini dapat dilakukan dalam waktu $O(n)$, sehingga masalah ini masih termasuk dalam kelas NP, di mana verifikasi solusi dapat dilakukan dalam waktu polinomial meskipun pencarian solusinya memerlukan waktu eksponensial.

Untuk menunjukkan perbedaan kompleksitas antara permasalahan yang dapat diselesaikan dalam waktu polinomial dan permasalahan yang termasuk dalam kelas NP, Gambar 2.1.1 memperlihatkan perbandingan peningkatan waktu berdasarkan peningkatan ukuran permasalahan. Grafik ini menggunakan skala logaritmik pada sumbu y untuk memperjelas perbedaan pertumbuhan waktu antara permasalahan dengan waktu penyelesaian polinomial dan non-polinomial.



Gambar 2.1.1: Perbandingan Kompleksitas Waktu Permasalahan Pengurutan dengan algoritma *Merge Sort* dan Permasalahan Penjadwalan

Pada grafik ini, *Merge Sort* yang memiliki kompleksitas waktu $O(n \log n)$ digambarkan dengan garis biru, yang memperlihatkan pertumbuhan waktu yang relatif lambat meskipun ukuran input bertambah. Sebaliknya, garis merah menggambarkan pertumbuhan permasalahan terhadap waktu pada permasalahan penjadwalan. Permasalahan penjadwalan yang termasuk dalam kelas NP dengan kompleksitas waktu $O(2^n)$ tumbuh secara eksponensial seiring bertambahnya ukuran input. Hal ini menunjukkan betapa cepatnya waktu komputasi bertambah untuk masalah dalam kelas NP. Garis hijau pada grafik tersebut menunjukkan kompleksitas waktu pada verifikasi solusi penjadwalan, yang dapat dilakukan dalam waktu polinomial $O(n)$. Hal ini memperlihatkan bahwa meskipun solusi penjadwalan sulit ditemukan, verifikasi solusi penjadwalan dapat dilakukan dengan cepat. Grafik ini mengilustrasikan dengan jelas perbedaan signifikan antara masalah polinomial dan masalah dalam kelas NP.

2.1.1.3 NP-Complete

Kelas masalah NP-Complete pertama kali muncul dalam teori kompleksitas komputasi pada awal 1970-an. Stephen Cook, seorang ahli komputer, memperkenalkan konsep ini dalam makalahnya yang berjudul "The Complexity of Theorem-Proving Procedures" pada tahun 1971 (Cook, 1971). Dalam makalah tersebut, Cook mengusulkan bahwa terdapat suatu kelas khusus dari masalah NP yang sangat sulit dipecahkan, tetapi memiliki karakteristik yang memungkinkan solusinya diverifikasi dengan mudah dalam waktu polinomial, yaitu kelas NP-Complete.

Cook memperkenalkan masalah *Boolean Satisfiability Problem* atau SAT sebagai masalah pertama yang terbukti sebagai permasalahan NP-Complete. SAT adalah masalah di mana diberikan suatu formula logika proposisional dalam bentuk Boolean dan tujuannya adalah untuk menentukan apakah terdapat pengaturan nilai kebenaran (*true* atau *false*) untuk setiap variabel dalam formula tersebut agar formula menjadi benar secara keseluruhan. Cook menjelaskan untuk membuktikan suatu permasalahan termasuk dalam kategori NP-Complete dibutuhkan dua kriteria pembuktian. Pertama, permasalahan yang ingin dibuktikan sebagai permasalahan NP-Complete harus terbukti sebagai permasalahan NP. Kedua, suatu permasalahan yang sudah terbukti sebagai NP-Complete seperti SAT harus bisa di reduksi menjadi permasalahan yang ingin dibuktikan dalam waktu polinomial.

Setelah penemuan Cook, Richard Karp memperluas konsep ini pada tahun 1972 dalam makalahnya "*Reducibility Among Combinatorial Problems*" (Karp, 2010). Karp menunjukkan bahwa banyak masalah kombinatorial lainnya, seperti *Traveling Salesman Problem* (TSP) dan *Knapsack Problem*, juga termasuk dalam NP-Complete dengan menggunakan reduksi polinomial, baik dari SAT maupun dari masalah NP-Complete lainnya. Ia memperkenalkan teknik sistematis untuk membuktikan *NP-Completeness* dan mengidentifikasi 21 masalah kombinatorial yang termasuk dalam kategori NP-Complete. Sejak saat itu, reduksi polinomial menjadi standar dalam membuktikan bahwa suatu masalah termasuk dalam kelas NP-Complete.

2.1.1.4 NP-Hard

Permasalahan dalam kelas NP-Hard mencakup permasalahan yang setidaknya sama sulitnya dengan masalah-masalah tersulit dalam kelas NP. Untuk membuktikan bahwa suatu permasalahan termasuk dalam kelas NP-Hard, diperlukan proses reduksi dari suatu masalah NP-Complete ke permasalahan tersebut dalam waktu polinomial. Pemilihan masalah dalam kelas NP-Complete dilakukan karena permasalahan ini sudah terbukti merupakan permasalahan tersulit dalam kelas NP. Ketika salah satu permasalahan dalam kelas NP-Complete bisa direduksi ke permasalahan yang ingin dibuktikan, maka permasalahan tersebut jelas paling tidak sama sulitnya dengan permasalahan tersulit dalam kelas NP dan dapat dikategorikan sebagai NP-Hard (Arora and Barak, 2009; Garey and Johnson, 1979).

Dalam kelas NP-Hard memungkinkan suatu permasalahan merupakan permasalahan NP. Hal ini ditentukan dari waktu validasi solusi yang dihasilkan. Jika proses validasi dapat dilakukan dalam waktu polinomial, maka permasalahan tersebut tergolong pada kelas permasalahan NP atau lebih tepatnya permasalahan tersebut adalah NP-Complete. Namun jika proses validasi tidak dapat dilakukan dalam waktu polinomial, maka permasalahan tersebut hanya tergolong dalam kelas NP-Hard dan bukan tergolong kelas permasalahan NP dan NP-Complete (Arora and Barak, 2009; Garey and Johnson, 1979).

2.1.1.5 Klasifikasi Permasalahan Penjadwalan sebagai NP-Complete dan NP-Hard

Permasalahan penjadwalan dapat diklasifikasikan sebagai NP-Complete karena memenuhi dua kriteria utama yang harus dipenuhi oleh masalah NP-Complete (Cooper and Kingston, 1996).

- 1. Termasuk dalam kelas NP :** Seperti yang telah dijabarkan pada Subbab 2.1.1.1 dan 2.1.1.2, menghasilkan solusi penjadwalan membutuhkan waktu eksponensial, namun untuk memverifikasi solusinya hanya membutuhkan waktu polinomial. Hal ini menunjukkan permasalahan penjadwalan dapat digolongkan sebagai permasalahan NP.
- 2. Permasalahan NP-Complete Dapat Direduksi Menjadi Permasalahan Penjadwalan dalam Waktu Polinomial:** Permasalahan penjadwalan dapat diformulasikan sebagai permasalahan *graph-coloring*. Permasalahan *graph-coloring*

telah dibuktikan sebagai permasalahan NP-Complete (Karp, 2010). Hal ini yang menyebabkan permasalahan penjadwalan memenuhi syarat kedua sebagai permasalahan NP-Complete.

Berdasarkan kriteria kedua, maka permasalahan penjadwalan dapat dibuktikan sebagai permasalahan yang paling tidak sama sulitnya dengan permasalahan NP yang ada karena permasalahan NP-Complete dapat di reduksi kedalam permasalahan penjadwalan. Hal ini menunjukkan bahwa permasalahan penjadwalan tergolong juga dalam kelas permasalahan NP-Hard.

Dalam permasalahan penjadwalan terdapat dua tujuan yaitu mencari solusi *feasible* dan mengoptimasi solusi. Untuk tujuan mencari solusi *feasible*, permasalahan penjadwalan dikategorikan sebagai NP dan NP-Complete, karena hasil dari solusi penjadwalan dapat divalidasi apakah solusi tersebut *feasible* atau tidak dalam waktu polinomial. Sementara itu, permasalahan penjadwalan yang bertujuan untuk mengoptimasi solusi hanya dapat dikategorikan sebagai permasalahan NP-Hard. Hal ini dikarenakan untuk dapat memvalidasi apakah solusi yang dihasilkan merupakan solusi paling optimal atau tidak membutuhkan waktu eksponensial. Dengan demikian, permasalahan penjadwalan dalam mengoptimasi solusi tidak bisa dikategorikan sebagai permasalahan NP dan NP-Complete.

2.1.2 Metode Aproksimasi, Heuristik dan Metaheuristik

Metode aproksimasi adalah metode yang digunakan untuk menemukan solusi yang mendekati optimal pada masalah yang sangat kompleks, terutama untuk masalah-masalah dalam kelas NP, NP-Complete dan NP-Hard. Pada jenis permasalahan tersebut, pencarian solusi optimal sering kali tidak mungkin dicapai dalam waktu komputasi yang efisien, karena ruang solusi yang bertambah secara eksponensial seiring bertambahnya ukuran permasalahan. Metode aproksimasi menawarkan kemampuan untuk menghasilkan solusi yang "cukup baik" dengan waktu komputasi yang lebih dapat diterima walaupun tidak bisa memastikan apakah solusi yang dihasilkan merupakan solusi optimal. Hal ini yang menyebabkan metode aproksimasi lebih banyak digunakan terutama dalam penerapannya secara praktikal (Vazirani, 1997; Williamson and Shmoys, 2011).

Di antara metode aproksimasi yang ada, kebanyakan pengembangan dilakukan terhadap metode heuristik dan metaheuristik. Metode heuristik merupakan metode yang

menggunakan aturan sederhana untuk menemukan solusi dengan cepat tanpa memeriksa seluruh ruang solusi. Heuristik berfokus pada bagian-bagian ruang solusi yang dianggap paling menjanjikan, sehingga waktu pencarian dapat diminimalkan. Meskipun solusi yang dihasilkan mungkin tidak optimal, pendekatan ini sering kali menghasilkan hasil yang cukup memadai dalam waktu singkat. Sebagai contoh, dalam masalah penjadwalan ujian, algoritma *Greedy* dapat digunakan dengan mengalokasikan waktu dan ruang ujian berdasarkan urutan kebutuhan ruang terbesar terlebih dahulu. Dengan strategi ini, ujian yang membutuhkan kapasitas ruang lebih besar dijadwalkan lebih awal, sehingga peluang mendapatkan ruang yang sesuai lebih tinggi. Pendekatan ini mungkin dapat menghasilkan jadwal yang layak dalam waktu singkat, meskipun belum tentu optimal (Blum and Roli, 2001; Franke et al., 2007; Vince, 2002).

Pendekatan heuristik umumnya memiliki kekurangan dimana hanya bekerja untuk beberapa jenis dan ukuran permasalahan. Sebagai contoh, algoritma *Greedy* mungkin hanya bekerja dengan baik untuk menghasilkan solusi *feasible* pada permasalahan penjadwalan ukuran kecil dan memiliki jumlah batasan yang minim. Namun saat diterapkan terhadap permasalahan penjadwalan ukuran besar dengan jumlah batasan yang banyak, algoritma ini tidak mampu untuk menghasilkan solusi *feasible*. Algoritma *Greedy* menunjukkan hasil yang sama pada permasalahan lainnya seperti pada permasalahan TSP (Bang-Jensen et al., 2004). Hal ini menunjukkan algoritma heuristik seperti algoritma *Greedy* hanya cocok diterapkan pada beberapa kasus spesifik saja (Blum and Roli, 2001).

Sementara itu, metode metaheuristik adalah pengembangan dari metode heuristik yang dirancang untuk memiliki tingkat generalisasi yang lebih baik dengan kemampuan untuk menghindari terjebak pada kondisi lokal optimal dengan strategi memperluas ruang pencarian. Metaheuristik menggunakan strategi pencarian yang lebih kompleks, yang memungkinkan algoritma menjelajahi ruang solusi yang lebih luas dan meningkatkan kemungkinan untuk menghasilkan solusi yang mendekati solusi optimal. Misalnya, Simulated Annealing menggunakan mekanisme penerimaan solusi yang lebih buruk dengan probabilitas tertentu pada awal proses pencarian, yang kemudian berkurang seiring waktu, untuk mendorong pencarian di area solusi yang lebih luas. Dengan pengembangan strategi pada metode metaheuristik, algoritma dapat diterapkan lebih general pada berbagai jenis permasalahan optimasi dengan berbagai ukuran permasalahan (Gendreau et al., 2010; Hussain et al., 2019).

Dalam metode metaheuristik, terdapat dua pendekatan, yaitu metaheuristik berbasis populasi, selanjutnya disebut sebagai metaheuristik (P), dan metaheuristik berbasis solusi tunggal, selanjutnya disebut sebagai metaheuristik (S). Metaheuristik (P) merupakan pendekatan yang menggunakan beberapa solusi dalam proses pencarian untuk mendapatkan solusi yang lebih baik, seperti pada algoritma Genetic yang menggunakan lebih dari satu solusi dan melakukan kombinasi antar solusi (*crossover*) untuk menghasilkan solusi baru. Sebaliknya, pada metaheuristik (S), pencarian solusi berfokus hanya pada satu solusi, seperti pada algoritma Simulated Annealing (Premananda et al., 2024; Tan et al., 2021).

Selain itu, kelebihan dan kekurangan dari berbagai algoritma dalam pendekatan metaheuristik menyebabkan munculnya metode baru yaitu hibrida metaheuristik. Metode ini menggabungkan dua atau lebih algoritma metaheuristik dengan tujuan memanfaatkan kelebihan masing-masing algoritma dan menutupi kekurangannya. Umumnya, metode ini memiliki dua pendekatan, yaitu menggabungkan algoritma berbasis populasi dan solusi tunggal, selanjutnya disebut sebagai hibrida (P+S), atau menggabungkan dua atau lebih algoritma solusi tunggal, selanjutnya disebut sebagai hibrida (S+S) (Premananda et al., 2024; Tan et al., 2021).

2.1.3 Permasalahan Penjadwalan

Permasalahan penjadwalan adalah permasalahan untuk mengalokasikan sumber daya terbatas—seperti waktu, ruang dan tenaga kerja—ke dalam serangkaian kegiatan atau acara yang memiliki keterbatasan atau batasan tertentu. Tujuan utama dari penjadwalan adalah untuk menghasilkan suatu jadwal yang memenuhi seluruh batasan yang ada, baik yang bersifat wajib (*hard constraint*) maupun preferensial (*soft constraint*), dalam rangka mengoptimalkan penggunaan sumber daya yang ada (Babaei et al., 2015; Chen et al., 2021; Rezaeipanah et al., 2021; Tan et al., 2021).

Permasalahan penjadwalan ditemukan dalam berbagai konteks, seperti penjadwalan ujian dan mata kuliah di universitas dan penjadwalan kompetisi olahraga. Meskipun konteksnya berbeda, sebagian besar permasalahan penjadwalan memiliki karakteristik umum berupa keterbatasan sumber daya dan kebutuhan untuk menghindari konflik antara kegiatan yang dijadwalkan (Liang et al., 2021; Song et al., 2018; Wu et al., 2015).

- **Penjadwalan Ujian:** Permasalahan ini Mengatur waktu dan lokasi ujian bagi sekelompok besar mahasiswa, dengan tujuan memastikan bahwa mahasiswa yang mengambil lebih dari satu mata kuliah tidak memiliki bentrokan dalam jadwal ujiannya. *Hard constraint* dalam penjadwalan ujian meliputi ketentuan bahwa tidak ada dua ujian yang dijadwalkan pada waktu dan ruang yang sama. *Soft constraint* mungkin mencakup pengurangan jumlah sesi ujian berturut-turut bagi seorang mahasiswa (Bellio et al., 2021; Bykov and Petrovic, 2016; Leite et al., 2018).
- **Penjadwalan Mata Kuliah:** Penjadwalan mata kuliah melibatkan penempatan jadwal mata kuliah di ruang kelas dan waktu tertentu dalam satu periode akademik. Terdapat dua pendekatan utama dalam penjadwalan kursus: *Post-Enrolment Course Timetabling* dan *Curriculum-Based Course Timetabling*. Pada pendekatan *Post-Enrolment*, mahasiswa telah memilih mata kuliah yang ingin mereka ambil sebelum jadwal disusun, sehingga penjadwalan dilakukan dengan mempertimbangkan pilihan individual mahasiswa. Dalam hal ini, batasan penjadwalan mencakup penghindaran bentrokan antara mata kuliah yang dipilih oleh mahasiswa yang sama. Beberapa *hard constraint* mungkin termasuk bahwa tidak ada dua mata kuliah yang diambil oleh mahasiswa yang sama dapat dijadwalkan pada waktu yang sama, serta keterbatasan kapasitas ruangan. Sementara itu, *soft constraint* bisa meliputi preferensi waktu bagi pengajar atau mahasiswa. Sebaliknya, pada pendekatan *Curriculum-Based*, jadwal disusun berdasarkan kurikulum yang telah ditetapkan, di mana setiap jurusan atau program memiliki set mata kuliah yang wajib diambil oleh semua mahasiswa di program tersebut. Dalam pendekatan ini, *hard constraint* mencakup ketentuan bahwa tidak boleh ada dua mata kuliah yang dijadwalkan pada waktu yang sama untuk satu program studi. *Soft constraint* bisa mencakup preferensi penempatan ruang yang dekat atau waktu yang lebih nyaman bagi mahasiswa dan pengajar (Goh et al., 2020; Nagata, 2018b; Soria-Alcaraz et al., 2016).
- **Penjadwalan Kompetisi Olahraga:** Penjadwalan ini mengatur jadwal pertandingan antar tim di liga atau turnamen, dengan tujuan untuk menghindari konflik lokasi dan memastikan pemerataan jadwal pertandingan. *Hard constraint* pada penjadwalan olahraga dapat mencakup ketentuan tidak boleh ada dua pertandingan yang berlangsung di stadion yang sama pada waktu yang sama dan tim tidak boleh

bermain lebih dari satu pertandingan pada waktu yang sama. *Soft constraint* mungkin mencakup preferensi untuk menghindari pertandingan berturut-turut bagi tim tertentu atau mengurangi perjalanan antar lokasi bagi tim-tim yang harus berpartisipasi dalam berbagai lokasi pertandingan (Jamili et al., 2012; Van Bulck and Goossens, 2023; Wu et al., 2015).

2.1.4 *Fitness Landscape*

Fitness landscape merupakan representasi menyeluruh dari ruang solusi dan nilai kualitas (*fitness* atau *cost*) untuk setiap titik solusinya. Dalam masalah optimasi diskrit seperti penjadwalan, setiap jadwal dapat dianggap sebagai satu titik pada *landscape* tersebut. Apabila penjadwalan dipahami sebagai penempatan sekumpulan kelas, ruang, dan waktu, maka setiap kombinasi pemetaan yang valid akan memiliki nilai *fitness* tertentu—misalnya jumlah konflik atau tingkat ketidakterpenuhan *hard* maupun *soft constraints* (Najaran, 2021; Reeves, 1999; Vassilev et al., 2003).

Pemahaman mengenai bentuk *landscape* menjadi penting karena secara langsung memengaruhi tingkat kesulitan pencarian dalam memperoleh solusi optimal. *Landscape* yang “rumit” (misalnya, memiliki banyak *local optima* dan struktur berliku atau *rugged*) cenderung membuat algoritma sederhana mudah terjebak pada solusi *suboptimal*. Sebaliknya, *landscape* yang relatif “mulus” lebih mudah ditelusuri oleh algoritma *local search*. Dengan Memahami karakteristik *landscape*, mulai dari derajat kekasaran (*ruggedness*), keberadaan *plateau* dan potensi terjebak dalam kondisi *local optima*, akan membantu dalam proses pengembangan strategi pencarian yang lebih efektif (Zou et al., 2022).

Pada bagian ini membahas dua hal utama berkaitan dengan *fitness landscape* yaitu karakteristik dari *fitness landscape* yang terdiri dari *ruggedness*, *smoothness*, *plateaus*, *local optima* dan *global optima* yang dibahas pada Subbab 2.1.4.1. Selanjutnya, pembahasan implikasi *landscape* terhadap pengembangan strategi algoritma terutama pada tahapan *move acceptance* dibahas pada Subbab 2.1.4.2.

2.1.4.1 Karakteristik *Fitness Landscape*

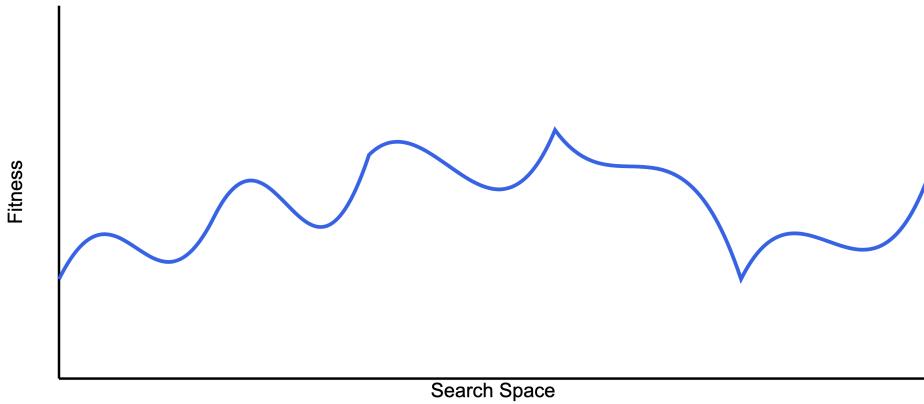
Terdapat lima istilah utama yang dapat digunakan untuk menggambarkan karakteristik *fitness landscape*. Istilah pertama adalah *ruggedness*, yang mengacu pada derajat

“kekasaran” suatu *landscape* optimasi. Semakin tinggi nilai *ruggedness*, semakin kompleks bentuk *landscape* tersebut, ditandai oleh keberadaan banyak puncak, lembah, dan *local optima*. Kondisi ini kerap menyebabkan pencarian solusi terjebak pada titik suboptimal (Zou et al., 2022). Gambar 2.1.2a menampilkan contoh *landscape* dengan *ruggedness* tinggi, yang tampak dari banyaknya puncak dan lembah, dan mengindikasikan risiko terperangkap pada solusi lokal yang kurang optimal.

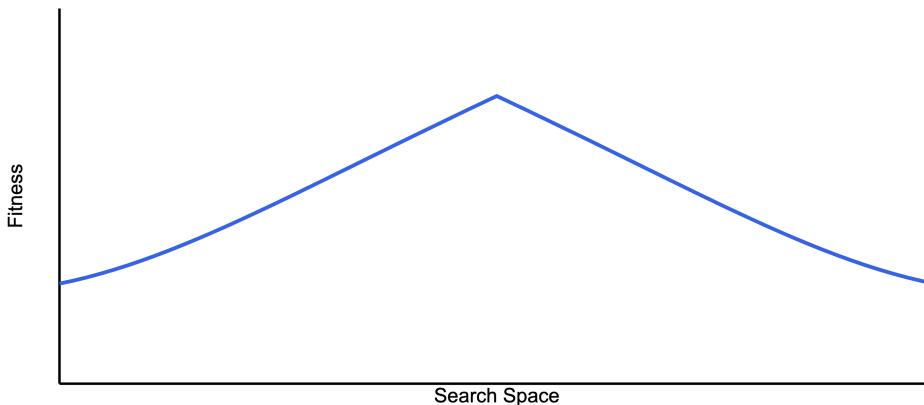
Sebaliknya, istilah kedua, yaitu *smoothness*, menunjukkan tingkat kemulusan perubahan *fitness* ketika berpindah dari satu solusi ke solusi lain. Suatu *landscape* yang relatif halus dapat membuat algoritma pencarian lebih stabil dan mengurangi frekuensi terjebaknya pencarian di *local optima*. Namun demikian, *smoothness* yang berlebihan dapat menghambat peningkatan *fitness*, karena perubahan di antara solusi cenderung kecil dan bertahap (Malan and Engelbrecht, 2013). Gambar 2.1.2b memperlihatkan *landscape* dengan satu puncak dominan, yang menandakan tingkat *smoothness* tinggi sehingga memudahkan proses pencarian solusi global.

Selain dua karakteristik di atas, terdapat tiga istilah lain yang turut memengaruhi pemahaman terhadap *fitness landscape*, yaitu *local optima*, *plateaus*, dan *global optima*. *Local optima* adalah titik di mana tidak tersedia perbaikan *fitness* yang dapat segera dicapai hanya dengan melakukan perubahan kecil (misalnya menukar posisi dua elemen) pada suatu solusi. Sementara itu, *plateaus* merupakan wilayah di mana beberapa solusi memiliki nilai *fitness* yang sama, sehingga perubahan pada solusi di dalam *plateau* tidak otomatis meningkatkan maupun menurunkan *fitness*. Kondisi *plateaus* sering kali menyebabkan proses pencarian berada pada keadaan statis dan kurang terstimulasi untuk menjelajahi ruang solusi lain (Czogalla and Fink, 2012). Terakhir, *global optima* merupakan solusi terbaik secara keseluruhan di dalam ruang solusi, yang berarti tidak ada solusi lain dengan nilai *fitness* yang lebih tinggi. Upaya untuk mencapai *global optima* umumnya menjadi tujuan utama dalam berbagai metode pencarian (Zou et al., 2022).

Gambar 2.1.3 mengilustrasikan *plateau*, *local optima*, serta *global optima*. Pada contoh tersebut, *local optima* (titik berwarna merah) merupakan puncak lokal dengan nilai yang selalu lebih rendah dibandingkan *global optima* (titik berwarna hijau), yang berfungsi sebagai puncak tertinggi. Selain itu, *plateau* yang digambarkan oleh garis berwarna kuning menunjukkan bahwa nilai *fitness* pada rentang tersebut relatif tetap dan tidak mengalami



(a) *Landscape* dengan Kondisi *Ruggedness*



(b) *Landscape* dengan Kondisi *Smoothness*

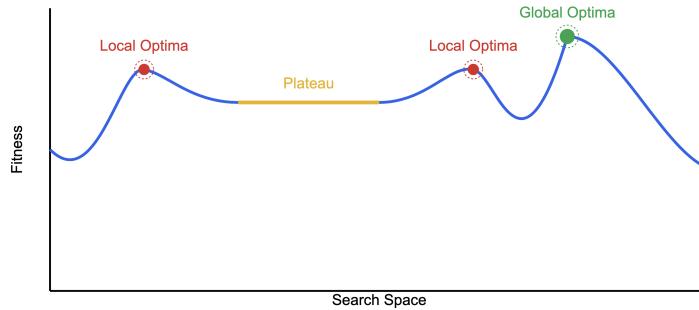
Gambar 2.1.2: Contoh Perbandingan Kurva pada *Landscape* dengan Kondisi *Ruggedness* dan *smoothness*

perubahan signifikan.

2.1.4.2 Implikasi *Landscape* terhadap Pengembangan Strategi *Move Acceptance*

Pendekatan pengembangan algoritma yang menitikberatkan pada *move acceptance* menjadikan mekanisme penerimaan (atau penolakan) solusi baru sebagai kunci untuk menjelajahi *landscape* secara lebih efektif. Dalam sebuah *landscape* yang kompleks—dengan beragam *local optima* dan kemungkinan terbentuknya *plateau*—strategi penerimaan yang tepat akan memungkinkan pencarian:

- Melepaskan diri dari jebakan *local optima*.
- Menjelajahi wilayah baru yang berpotensi mengandung solusi lebih baik.



Gambar 2.1.3: Contoh Kurva *Fitness* yang Menunjukkan Beberapa *Local Minima*, Area *Plateau*, dan Satu *Global Minimum*.

- Memperdalam perbaikan solusi saat berada di area yang menjanjikan (eksplorasi).

Selama proses pencarian, perubahan pada solusi dapat meningkatkan, mempertahankan, atau bahkan menurunkan nilai *fitness*. Apabila mekanisme penerimaan hanya mengizinkan solusi dengan nilai *fitness* lebih baik, pencarian cenderung cepat terjebak pada *local optima*. Namun, jika aturan penerimaan terlalu longgar terhadap solusi yang lebih buruk, pencarian dikhawatirkan justru berlarut-larut di wilayah yang tidak menjanjikan (Basseur and Goëffon, 2015; Najaran, 2021).

Dalam konteks pengembangan algoritma optimasi dan penjelajahan *fitness landscape*, terdapat dua aspek penting yang perlu dijaga, yakni eksplorasi (memperluas ruang pencarian) dan eksplorasi (memanfaatkan solusi yang sudah baik). *Move acceptance* memegang peran krusial sebagai mekanisme yang menyeimbangkan kedua aspek tersebut:

- 1. Memfasilitasi Eksplorasi:** Memberikan toleransi untuk menerima solusi yang tidak selalu lebih baik. Hal ini membuka peluang bagi algoritma untuk menelusuri lintasan baru dalam *landscape*, daripada berhenti pada solusi yang terlihat stabil namun berpotensi belum optimal.
- 2. Memperkuat Eksplorasi:** Menekan penerimaan terhadap solusi yang jauh lebih buruk dan mengarahkan pencarian agar fokus menyempurnakan solusi yang sudah menjanjikan, sehingga nilai *fitness* terus meningkat secara bertahap.

Penetapan tingkat toleransi atau aturan penerimaan tertentu—seperti berbasis *threshold*, probabilitas, maupun kombinasi keduanya—sangat menentukan sejauh mana algoritma sanggup bertahan dalam upaya memperbaiki kualitas solusi, sekaligus tetap menjaga akses

ke wilayah lain di ruang solusi (Malan and Engelbrecht, 2013).

2.1.5 Tingkatan Generalitas Algoritma

Salah satu tantangan dalam penelitian penjadwalan adalah menghasilkan suatu algoritma yang berkinerja baik pada sekumpulan dataset dengan permasalahan yang sama belum tentu mampu menghasilkan hasil yang sama baiknya saat diterapkan pada kumpulan dataset yang berbeda dalam permasalahan yang sama atau bahkan pada jenis permasalahan yang berbeda. Untuk mengatasi hal tersebut maka penting untuk menghasilkan algoritma dengan yang memiliki tingkatan generalitas yang baik (Pillay, 2016b).

Pillay (2016b) membagi tingkatan generalitas dari suatu algoritma pada tiga kategori yang terdiri dari:

1. **Generalitas Tingkat Dataset** Tingkatan ini mengacu pada kemampuan algoritma untuk secara konsisten berkinerja baik di semua dataset dalam satu *benchmark* tertentu. Misalnya, dalam penjadwalan ujian, algoritma yang dikembangkan diharapkan dapat menghasilkan solusi yang efektif dan konsisten pada seluruh dataset dalam satu *benchmark*. Sebagai contoh, *benchmark* Toronto memiliki 13 dataset. Maka algoritma yang dikembangkan untuk memenuhi tingkat generalitas ini harus mampu menghasilkan hasil yang konsisten baik terhadap 13 dataset tersebut.
2. **Generalitas Tingkat Set atau Benchmark:** Tingkatan ini menilai kemampuan algoritma untuk berkinerja efektif dan konsisten di berbagai *benchmark* untuk jenis masalah yang sama. Sebagai contoh, algoritma yang dikembangkan untuk memenuhi tingkat generalitas ini mampu menghasilkan solusi yang baik dan konsisten pada dua *benchmark* dalam permasalahan penjadwalan ujian seperti pada *benchmark International Timetabling Competition (ITC) 2007 Track 1* dan *benchmark* Toronto.
3. **Generalitas Tingkat Jenis Masalah (Cross Domain):** Tingkat generalisasi ini menilai apakah algoritma dapat digeneralisasi untuk diterapkan di berbagai jenis masalah penjadwalan, tidak terbatas pada satu *set* masalah tertentu. Sebagai contoh, algoritma yang dikembangkan untuk memenuhi tingkatan generalitas ini harus menghasilkan solusi yang baik dan konsisten pada permasalahan ujian, mata kuliah dan kompetisi olahraga.

2.1.6 Hiper-heuristik

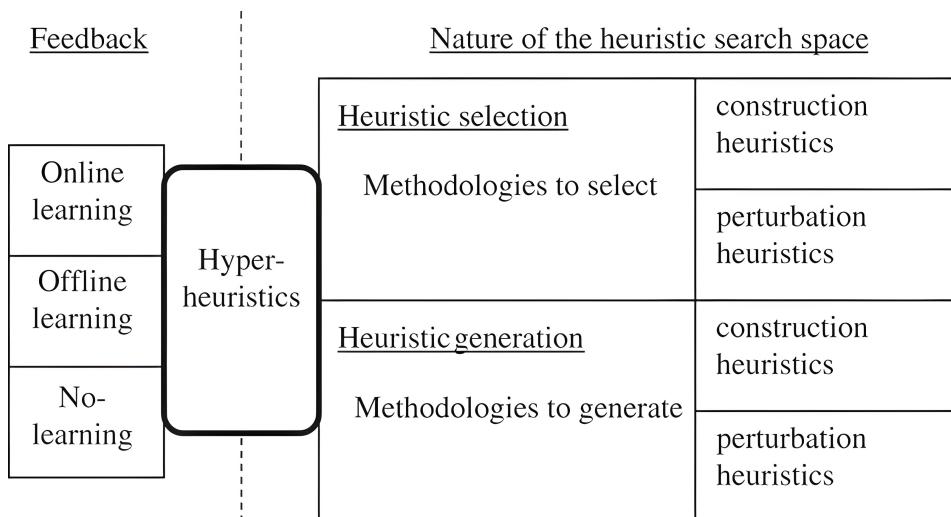
Metode metaheuristik merupakan metode yang umum digunakan dalam permasalahan yang termasuk dalam kelas NP-Complete dan NP-Hard. Hal ini disebabkan oleh kemampuan metaheuristik sebagai metode *high-level* yang dapat diterapkan pada berbagai jenis masalah. Walaupun metode metaheuristik dapat diterapkan pada berbagai permasalahan, namun solusi yang dihasilkan tidak konsisten. Pengembangan algoritma dengan metode metaheuristik umumnya dikembangkan secara spesifik dari segi desain strategi dan pengaturan parameter pada satu studi kasus atau pada satu *benchmark* tertentu. Akibatnya, sulit bagi metode ini untuk menghasilkan solusi yang konsisten baik saat diterapkan pada permasalahan yang berbeda (Pillay, 2016b).

Untuk mengatasi kekurangan ini, maka dikembangkan metode baru yang dikenal sebagai metode hiper-heuristik. Metode ini bertujuan membangun algoritma yang lebih generik, dengan salah satu tujuannya adalah mampu menangani berbagai jenis permasalahan. Secara formal, hiper-heuristik didefinisikan sebagai metode pencarian atau mekanisme pembelajaran yang bertujuan memilih atau membentuk heuristik guna memecahkan permasalahan komputasional (Burke et al., 2013, 2003; Pillay, 2016b).

Hiper-heuristik dapat diklasifikasikan berdasarkan beberapa kriteria, seperti yang ditunjukkan pada Gambar 2.1.4. Klasifikasi ini dapat dilihat dari dua dimensi. Dimensi pertama adalah klasifikasi berdasarkan sumber informasi umpan balik yang digunakan dalam pembelajaran, yang terbagi menjadi tiga jenis: *online*, *offline* dan tanpa umpan balik. Pada proses pembelajaran *online*, umpan balik diperoleh secara langsung selama proses pencarian solusi berlangsung. Sebaliknya, dalam pembelajaran *offline*, proses belajar dilakukan dengan menerapkan *preliminary run* atau proses awal sebelum proses pencarian solusi dimulai (Burke et al., 2013, 2003; Pillay, 2016b).

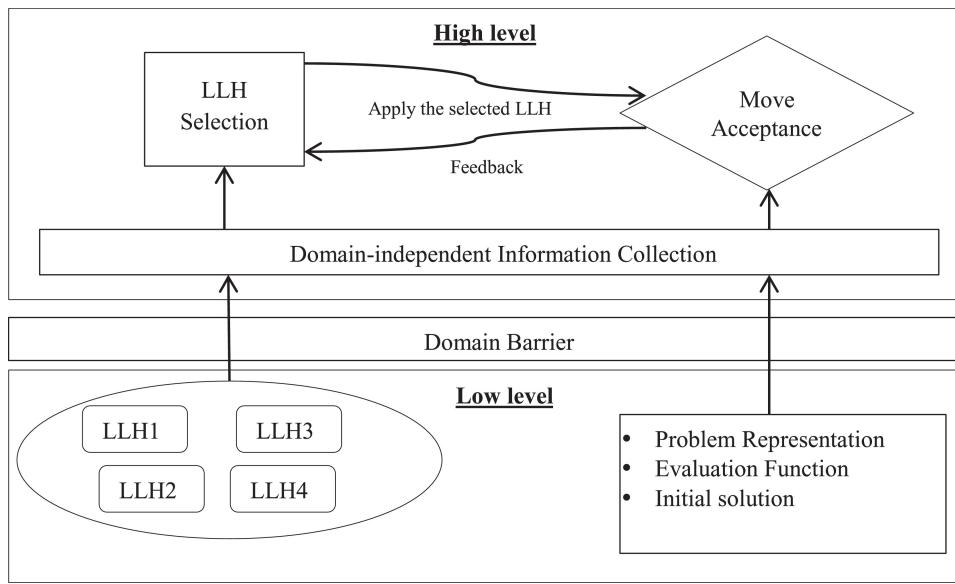
Dimensi kedua dalam klasifikasi hiper-heuristik adalah ruang pencarian heuristik, yang membagi hiper-heuristik menjadi dua metodologi utama: metodologi pemilihan heuristik (*heuristic-selection*) dan metodologi pembentukan heuristik (*heuristic-construction*). Kedua metodologi ini juga terbagi menjadi dua pendekatan yang berbeda. Pendekatan pertama adalah *construction-heuristic*, yang membangun solusi dari awal pada setiap iterasi. Pendekatan kedua adalah *perturbation-heuristic*, di mana solusi yang sudah *feasible* dimodifikasi untuk mencapai solusi yang lebih optimal. Dengan

demikian, terdapat empat jenis pendekatan dalam metode hiper-heuristik, yaitu hiper-heuristik dengan *selection-perturbation*, selanjutnya disebut sebagai hiper heuristik (S-P), hiper-heuristik dengan *selection-construction*, selanjutnya disebut sebagai hiper heuristik (S-C), hiper-heuristik dengan *generation-perturbation*, selanjutnya disebut sebagai hiper heuristik (G-P), serta hiper-heuristik dengan *generation-construction*, selanjutnya disebut sebagai hiper heuristik (G-C) (Burke et al., 2013, 2003; Pillay, 2016b).



Gambar 2.1.4: Klasifikasi Metode Hiper-Heuristik (Burke et al., 2013)

Secara struktur, metode hiper-heuristik terdiri dari dua bagian utama, yaitu *High-Level Heuristic* (HLH) dan *Low-Level Heuristic* (LLH). Bagian HLH terdiri dari dua komponen: strategi pemilihan LLH dan strategi *move acceptance*. Sedangkan pada bagian LLH, terdapat sekumpulan LLH yang dapat dipilih dan digunakan oleh HLH. Proses pencarian solusi dalam hiper-heuristik dimulai dengan melakukan pemilihan LLH berdasarkan strategi yang telah ditetapkan. LLH yang dipilih kemudian diterapkan pada solusi saat ini untuk menghasilkan solusi baru. Setelah itu, strategi *move acceptance* akan menentukan apakah solusi baru tersebut diterima atau tidak. Keputusan yang diambil oleh komponen *move acceptance* ini menjadi sumber informasi bagi mekanisme pemilihan LLH untuk melakukan pembelajaran. Proses ini terus berulang hingga mencapai batas iterasi atau tujuan optimasi. Gambar 2.1.5 menunjukkan struktur dari metode hiper-heuristik (Choong et al., 2018).



Gambar 2.1.5: Struktur Hiper-Heuristik (Choong et al., 2018)

2.1.7 Algoritma ILS

Algoritma ILS adalah salah satu algoritma yang dikembangkan dengan metode metaheuristik. Algoritma 1 menunjukkan pseudocode dari algoritma ILS. Algoritma ILS didesain dengan menerapkan perulangan terhadap tiga tahapan yaitu *perturbation*, *local search* dan *move acceptance*. Secara detail, tiga tahapan dalam algoritma ILS dijabarkan sebagai berikut:

- **Tahapan Perturbation:** Tahapan ini bertujuan untuk melakukan eksplorasi solusi. Hal ini dilakukan sebagai upaya untuk menghindari solusi terjebak dalam kondisi *local optima*. Pada tahapan ini solusi akan di modifikasi dengan menerapkan LLH. Umumnya solusi baru yang dihasilkan akan diterima tanpa memperhatikan apakah solusi yang dihasilkan lebih baik atau lebih buruk. Hal ini untuk mendukung tujuan eksplorasi solusi pada tahapan ini (Soria-Alcaraz et al., 2016).
- **Tahapan Local Search:** Tahapan ini bertujuan untuk melakukan eksloitasi solusi. Pada tahapan ini, solusi akan di modifikasi dengan menerapkan LLH. Solusi baru umumnya lebih sering diterima ketika menghasilkan solusi yang lebih baik. Secara detail, penerimaan solusi bergantung pada strategi yang diterapkan. Sebagai contoh jika menerapkan strategi *hill-climbing* pada tahapan ini, maka hanya solusi yang lebih baik atau sama dengan solusi sebelumnya yang diterima. Hasil dari tahapan ini

diharapkan mampu menemukan solusi lebih baik dari iterasi sebelumnya (Goh et al., 2020; Soria-Alcaraz et al., 2017).

- **Tahap Move Acceptance:** Tahapan ini bertujuan untuk memutuskan solusi yang telah dihasilkan pada tahapan *local search* akan diterima dan digunakan pada iterasi selanjutnya atau tidak. Kriteria penerimaan bergantung pada strategi yang diterapkan. Umumnya strategi penerimaan mengadopsi strategi dari algoritma metaheuristik seperti algoritma Simulated Annealing (Goh et al., 2020; Song et al., 2018).

Algoritma 1 ILS

```
1: procedure ITERATEDLOCALSEARCH(initialSolution)
2:   while termination condition not met do
3:      $s' \leftarrow \text{Perturbation}(s)$ 
4:      $s'' \leftarrow \text{LocalSearch}(s')$ 
5:     if  $f(s'') < f(s^*)$  then                                 $\triangleright$  Update best solution
6:        $s^* \leftarrow s''$ 
7:     end if
8:     if AcceptanceCriterion( $s, s''$ ) then
9:        $s \leftarrow s''$ 
10:    end if
11:   end while
12:   return  $s^*$ 
13: end procedure
```

2.2 Kajian Penelitian Terdahulu

Penelitian pada bidang pengembangan algoritma untuk permasalahan penjadwalan telah berkembang dengan munculnya berbagai macam pengembangan algoritma dan dataset permasalahan. Seiring perkembangannya, beberapa kumpulan dataset sering digunakan oleh peneliti untuk menguji algoritma yang mereka kembangkan dan membandingkan hasilnya dengan penelitian lain. Kumpulan dataset tersebut saat ini digunakan sebagai *benchmark* oleh para peneliti, seperti pada *benchmark* Toronto, Socha dan ITC 2007, untuk mengevaluasi algoritma yang dikembangkan. Penggunaan *benchmark* dalam pengembangan algoritma penjadwalan menjadi penting karena permasalahan penjadwalan merupakan permasalahan NP-Hard yang sulit untuk mengetahui solusi optimal dari suatu permasalahan. Oleh karena itu, menggunakan *benchmark* untuk mengukur kinerja algoritma dengan membandingkan hasil penelitian

dengan hasil dari penelitian terdahulu pada *benchmark* yang sama merupakan cara yang paling umum digunakan dalam mengevaluasi algoritma yang dikembangkan.

Dalam proses perbandingan antar penelitian, umumnya evaluasi dilakukan terhadap dua metrik utama, yaitu:

- **Feasibilitas:** Metrik ini digunakan untuk mengevaluasi kemampuan algoritma dalam menghasilkan solusi *feasible*. Evaluasi dilakukan dengan membandingkan jumlah dataset yang berhasil menghasilkan solusi *feasible*. Dataset yang tidak menghasilkan solusi *feasible* tidak disertakan dalam proses perbandingan, karena solusi yang tidak *feasible* tidak dapat digunakan. Sebagai ilustrasi, solusi penjadwalan yang menempatkan 30 siswa dalam sebuah ruangan dengan kapasitas 20 orang tidak dapat digunakan. Demikian pula, solusi lain yang menempatkan 27 siswa dalam ruangan dengan kapasitas yang sama juga tidak dapat digunakan, meskipun jumlah siswa yang dijadwalkan lebih sedikit. Oleh karena itu, perbedaan jumlah *hard constraint* yang dilanggar pada solusi yang tidak *feasible* tidak memiliki pengaruh, karena solusi tersebut tetap tidak dapat digunakan.
- **Kualitas Solusi:** Metrik kualitas solusi digunakan untuk mengevaluasi kemampuan algoritma dalam memenuhi *soft constraints*—termasuk *fairness*, preferensi pengguna, serta aspek-aspek lain yang relevan—sehingga jadwal yang dihasilkan dapat memenuhi berbagai kriteria kualitas yang diinginkan. Pada metrik ini, penilaian biasanya dilakukan dengan menggunakan rata-rata, solusi terbaik, serta standar deviasi atau koefisien variasi. Pendekatan ini bertujuan untuk mengevaluasi kemampuan algoritma dalam menghasilkan solusi terbaik sekaligus menilai konsistensinya. Selain itu, kualitas solusi hanya akan dievaluasi apabila solusi yang dihasilkan berada dalam kondisi *feasible*.

Selain kedua metrik tersebut, terdapat juga metrik penilaian lain, yaitu generalitas. Metrik ini menilai kemampuan suatu algoritma ketika diuji pada dataset dan permasalahan yang berbeda. Perbandingan dilakukan dengan melihat hasil dari feasibilitas dan kualitas solusi yang dihasilkan pada berbagai dataset dan permasalahan.

Dari perkembangan penelitian sebelumnya, terdapat 98 penelitian dari tahun 2012 hingga tahun 2022 yang menggunakan berbagai *benchmark* untuk mengevaluasi algoritma yang dikembangkan. Terdapat tujuh *benchmark* yang banyak digunakan peneliti, yaitu

Toronto, Socha, ITC 2007 (Track 1, Track 2, dan Track 3), ITC 2019, dan ITC 2021. Untuk mendeskripsikan perkembangan penelitian terdahulu pada ke tujuh dataset tersebut, Subbab 2.2.1 hingga Subbab 2.2.7 akan membandingkan hasil dari setiap penelitian dengan melakukan pemeringkatan hasil solusi pada setiap dataset. Pada *benchmark* Toronto dan ITC 2007 pembahasan pengembangan penelitian difokuskan pada sisi optimasi solusi, karena untuk menghasilkan solusi *feasible* bukan menjadi permasalahan yang sulit. Sementara pada ITC 2019 dan ITC 2021, pembahasan melibatkan hasil solusi *feasible* dan juga hasil optimasi solusi. Selanjutnya, Subbab 2.2.8 melakukan perbandingan terhadap tingkatan generalitas dari penelitian yang ada berdasarkan pemeringkatan yang telah dihasilkan pada beberapa Subbab sebelumnya.

Meskipun fokus utama sebagian besar penelitian lebih kepada pada optimasi solusi, namun terdapat permasalahan yang memiliki kesulitan dalam menghasilkan solusi *feasible*, khususnya pada dataset yang memiliki *hard constraints* kompleks. Pada Subbab 2.2.9 membahas penelitian yang berfokus pada permasalahan yang penjadwalan untuk menghasilkan solusi *feasible*. Pembahasan ini menjadi relevan terutama dalam konteks penyelesaian permasalahan penjadwalan terkini, seperti pada *benchmark* ITC 2019 dan ITC 2021, yang dikenal memiliki tingkat kesulitan tinggi dalam menghasilkan solusi *feasible*.

2.2.1 Benchmark Toronto

Benchmark Toronto diperkenalkan oleh Carter et al. (1996) yang mengangkat permasalahan penjadwalan ujian tanpa memperhitungkan kapasitas ruangan (*uncapacitated*). *Benchmark* ini terdiri dari 13 dataset *real-world* yang diambil dari berbagai universitas, dengan sejumlah batasan yang disederhanakan. Pada permasalahan ini, terdapat *hard constraint* yang memastikan bahwa tidak ada siswa yang dijadwalkan untuk mengikuti lebih dari satu ujian pada waktu yang sama. Selain itu, *soft constraint* diterapkan untuk mengoptimalkan waktu istirahat antar ujian pada hari yang sama untuk setiap siswa (Fong et al., 2014a).

Tabel 2.2.1 menampilkan hasil pemeringkatan pada penelitian-penelitian yang menggunakan *benchmark* Toronto. Hasil dari pemeringkatan menunjukkan pengembangan algoritma dengan pendekatan metaheuristik (S) menghasilkan hasil yang sangat baik, seperti yang ditunjukkan dalam studi oleh Bellio et al. (2021) dengan pengembangan

algoritma Simulated Annealing dan studi oleh Burke and Bykov (2016) dengan pengembangan algoritma Great Deluge. Kedua penelitian ini menempati peringkat pertama dan ketiga. Pada pendekatan hibrida, penelitian oleh Leite et al. (2018) yang menggabungkan algoritma Cellular Memetics dengan Threshold Acceptance dan berada pada peringkat kedua. Sementara itu, pendekatan hiper-heuristik (S-P) yang diterapkan oleh Demeester et al. (2012) juga menggunakan algoritma Simulated Annealing dan menempati peringkat keempat. Namun, tidak semua penelitian dengan pendekatan metaheuristik (S) dan hibrida menunjukkan hasil optimal. Sebagai contoh, studi oleh Al-Betar (2021), Pais and Amaral (2012), dan Li et al. (2015) yang menggunakan pendekatan yang sama berada pada peringkat 17, 18, dan 21.

Sementara itu, penelitian yang menggunakan pendekatan lain seperti metaheuristik (P) dan hiper-heuristik (G-P), hanya menghasilkan peringkat di sekitar rata-rata atau median. Tiga studi yang menggunakan hiper-heuristik (S-C) bahkan menunjukkan hasil yang kurang memuaskan, menempati peringkat ke-16, ke-19, dan ke-25. Selain itu, satu studi dengan pendekatan hiper-heuristik (G-C) oleh Pillay and Özcan (2019) berada pada peringkat 23.

Tabel 2.2.1: Perbandingan Pemeringkatan Penelitian pada *Benchmark* Toronto

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
1	Bellio et al. (2021)	Metaheuristik (S)	Simulated Annealing	1,76
2	Leite et al. (2018)	Hibrida (P+S)	Cellular Memetic + Threshold Acceptance	2,73
3	Burke and Bykov (2016)	Metaheuristik (S)	Great Deluge	2,88
4	Demeester et al. (2012)	Hiper-heuristik (S-P)	Simulated Annealing	4,73
5	Mandal et al. (2020)	Hibrida (Other)	Great Deluge + Constructive Method	7,91

(Lanjutan) Perbandingan Pemeringkatan Penelitian pada *Benchmark* Toronto

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
6	Alzaqebah and Abdullah (2015)	Hibrida (P+S)	Bee Colony Optimization + Simulated Annealing + Late Acceptance Hill-Climbing	8,58
7	Fong et al. (2014a)	Hibrida (P+S)	Artificial Bee Colony + Great Deluge	8,81
8	Muklason (2017)	Hiper-heuristik (S-P)	Hyflex	9,69
9	Pillay (2016a)	Metaheuristik (P)	Developmental Approach	10,45
10	Alzaqebah and Abdullah (2014)	Hibrida (P+S)	Artificial Bee Colony + Late Acceptance Hill-Climbing	10,95
11	Sabar, Ayob, Kendall and Qu (2012)	Metaheuristik (P)	Honey-Bee Mating Optimization	11,11
12	Abdullah and Alzaqebah (2013)	Hibrida (P+S)	Bee Algorithm + Simulated Annealing + Late Acceptance Hill Climbing	11,95
13	Al-Betar et al. (2014)	Metaheuristik (P)	Harmony Search	11,95
14	Sabar et al. (2013)	Hiper-heuristik (G-P)	Backus Naur Form Grammar + Genetic Algorithm	13,00
15	Aldeeb et al. (2021)	Hibrida (P+S)	Intelligent Water Drops + Local Search Optimizer	13,62
16	Qu et al. (2015)	Hiper-heuristik (S-C)	Estimation Distribution	15,65
17	Al-Betar (2021)	Metaheuristik (S)	B-hill Climbing	16,25

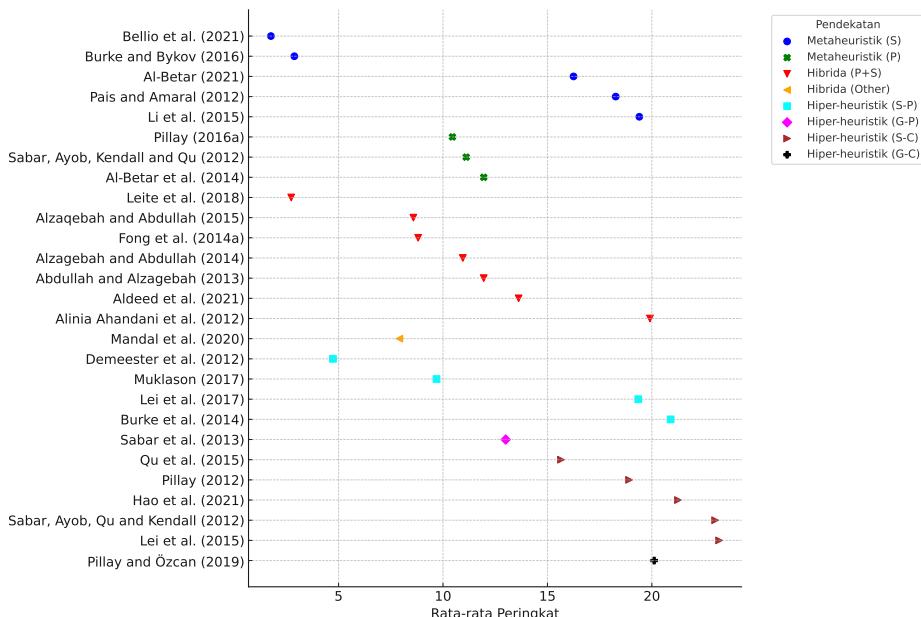
(Lanjutan) Perbandingan Pemeringkatan Penelitian pada *Benchmark* Toronto

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
18	Pais and Amaral (2012)	Metaheuristik (S)	Tabu Search	18,27
19	Pillay (2012)	Hiper-heuristik (S-C)	Evolutionary Algorithm	18,91
20	Lei et al. (2017)	Hiper-heuristik (S-P)	Memetic Algorithm	19,35
21	Li et al. (2015)	Metaheuristik (S)	Ruin and Recreate	19,40
22	Alinia Ahandani et al. (2012)	Hibrida (P+S)	Discrete Particle Swarm Optimization + Hill Climbing	19,91
23	Pillay and Özcan (2019)	Hiper-heuristik (G-C)	Arithmetic and Hierarchical Heuristics	20,11
24	Burke et al. (2014)	Hiper-heuristik (S-P)	Adaptive Selection Heuristic	20,90
25	Hao et al. (2021)	Hiper-heuristik (S-C)	Evolutionary Multitasking Optimization	21,25
26	Sabar, Ayob, Qu and Kendall (2012)	Hiper-heuristik (S-C)	Graph Coloring	23,03
27	Lei et al. (2015)	Hiper-heuristik (S-C)	Memetic Algorithm + Hill Climbing	23,22

Gambar 2.2.1 menampilkan grafik persebaran pendekatan dari 27 penelitian. Setiap pendekatan direpresentasikan dengan simbol dan warna yang berbeda. Sumbu y menunjukkan nama studi, sementara sumbu x mewakili peringkat mereka. Grafik ini menunjukkan bahwa beberapa studi dengan pendekatan metaheuristik (S), hibrida (P+S), dan hiper-heuristik (P-S) mencapai performa yang superior dibandingkan pendekatan

lainnya. Namun, ditemukan hasil yang bervariasi di antara studi dengan pendekatan serupa yang kemungkinan disebabkan oleh perbedaan dalam pengembangan algoritma pada setiap penelitian.

Dari perspektif algoritma, Simulated Annealing dan Great Deluge menunjukkan kinerja yang baik pada *benchmark* ini. Simulated Annealing, yang diterapkan dalam penelitian oleh Bellio et al. (2021), Demeester et al. (2012), Alzaqebah and Abdullah (2015), dan Abdullah and Alzaqebah (2013), menempati peringkat 1, 4, 6, dan 12 secara berturut-turut. Pada algoritma Great Deluge, penelitian oleh Burke and Bykov (2016), Mandal et al. (2020), dan Fong et al. (2014a) berhasil meraih peringkat 3, 5, dan 7.



Gambar 2.2.1: Persebaran Pendekatan pada *Benchmark* Toronto

2.2.2 *Benchmark Socha*

Benchmark Socha, diperkenalkan oleh Socha et al. (2003), merupakan kumpulan dataset yang dikembangkan dengan generator yang dibuat oleh Ben Paechter. Dataset ini terdiri dari 12 dataset, namun 11 di antaranya yang paling sering digunakan dalam penelitian. *Benchmark* ini dirancang untuk menangani masalah penjadwalan mata kuliah tipe *post-enrollment*, dengan beberapa *hard constraint* sebagai berikut:

- Tidak ada mahasiswa yang dijadwalkan untuk mengikuti dua atau lebih mata kuliah dalam waktu bersamaan.

- Setiap mata kuliah harus ditempatkan pada ruang yang sesuai dengan kebutuhan.
- Kapasitas setiap ruang yang digunakan harus mencukupi jumlah mahasiswa yang mengikuti mata kuliah tersebut.
- Satu ruang hanya dapat digunakan untuk satu mata kuliah pada setiap slot waktu.

Permasalahan ini juga memiliki tiga *soft constraint* yang terdiri dari:

- Tidak menjadwalkan mata kuliah pada slot terakhir setiap harinya.
- Siswa tidak memiliki lebih dari dua mata kuliah berturut-turut.
- Jadwal siswa tidak hanya terdiri dari satu mata kuliah dalam satu hari.

Tabel 2.2.2 menyajikan hasil pemeringkatan pada 14 penelitian yang menggunakan *benchmark* Socha. Sementara itu, Gambar 2.2.2 menampilkan persebaran antara penelitian yang dikelompokkan berdasarkan pendekatan yang digunakan. Pendekatan metaheuristik (S) memberikan hasil yang baik, dengan menempati empat peringkat teratas. Tiga dari empat studi teratas ditulis oleh peneliti yang sama, yaitu oleh Goh et al. (2017, 2019, 2020), yang melakukan penyempurnaan terhadap algoritma Simulated Annealing. Pada peringkat ke 4, penelitian oleh Nagata (2018b) memperkenalkan algoritma baru bernama Random Partial Neighbourhood Search.

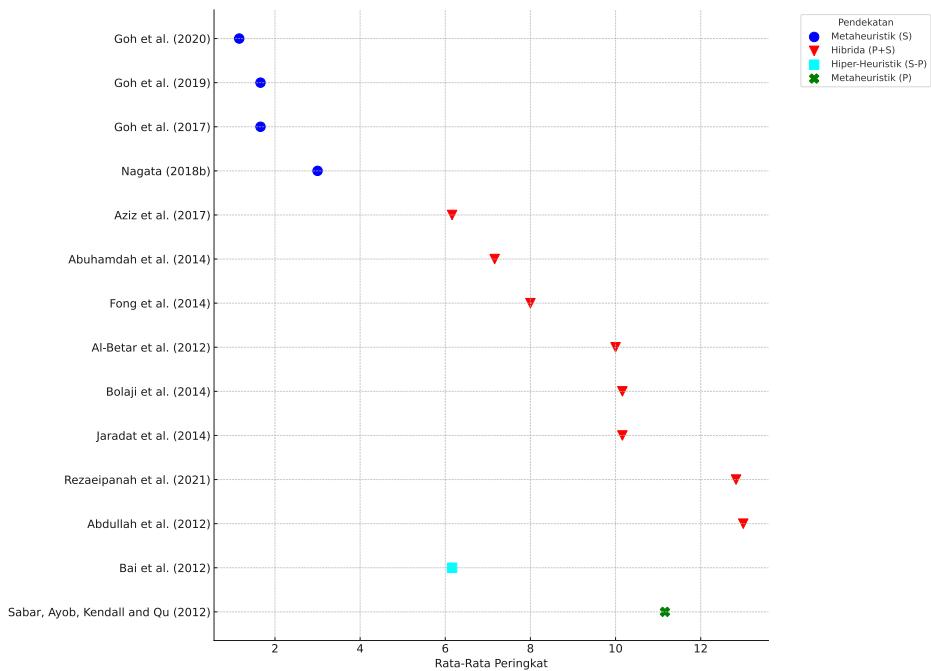
Pendekatan lainnya, terutama hibrida (P+S), hanya menunjukkan kinerja yang moderat. Meskipun hibrida adalah salah satu pendekatan yang paling sering digunakan, secara keseluruhan hasilnya tidak menunjukkan kinerja yang baik. Satu studi hibrida, Aziz et al. (2017), berhasil menempati peringkat kelima. Namun terdapat kesenjangan performa yang jelas dibandingkan penelitian yang menggunakan pendekatan metaheuristik (S). Hal ini ditunjukkan dengan adanya jarak berdasarkan perbedaan rata-rata peringkat yang ditampilkan pada Gambar 2.2.2. Sebagian besar studi dengan pendekatan hibrida lainnya justru menempati peringkat di bawah rata-rata. Pendekatan lain seperti metaheuristik (P) oleh Sabar, Ayob, Kendall and Qu (2012) dan hiper-heuristik (S-P) oleh Bai et al. (2012) menunjukkan performa yang kurang baik dengan berada pada peringkat ke-11 dan ke-6.

Dari sisi algoritma, Simulated Annealing kembali terbukti sangat efektif, dengan menempati peringkat 1, 2, 3, dan 6. Sementara pada algoritma Great Deluge, yang sebelumnya menunjukkan kinerja baik pada *benchmark* Toronto, tidak menunjukkan hasil

yang serupa pada *benchmark* Socha. Penelitian dengan algoritma ini hanya menempati peringkat ke-8 dan ke-14. Algoritma lain seperti Random Partial Neighbourhood Search, menunjukkan potensi yang baik dengan berada pada peringkat 4.

Tabel 2.2.2: Perbandingan Pemeringkatan Penelitian pada *Benchmark* Socha

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
1	Goh et al. (2020)	Metaheuristik (S)	Simulated Annealing	1,16
2	Goh et al. (2019)	Metaheuristik (S)	Simulated Annealing	1,66
3	Goh et al. (2017)	Metaheuristik (S)	Simulated Annealing	1,66
4	Nagata (2018b)	Metaheuristik (S)	Random Partial Neighbourhood Search	3,00
5	Aziz et al. (2017)	Hibrida (P+S)	Honey-Bee Mating Optimization + Adaptive Guided Variable Neighbourhood Search	6,16
6	Bai et al. (2012)	Hiper-heuristik (S-P)	Simulated Annealing	6,16
7	Abuhamda et al. (2014)	Hibrida (P+S)	Gravitational Emulation + MPCA-ARDA	7,16
8	Fong et al. (2014a)	Hibrida (P+S)	Artificial Bee Colony + Great Deluge	8,00
9	Al-Betar et al. (2012)	Hibrida (P+S)	Harmony Search + Hill Climbing	10,00
10	Bolaji et al. (2014)	Hibrida (P+S)	Artificial Bee Colony + Hill Climbing Optimizer	10,16
11	Jaradat et al. (2014)	Hibrida (P+S)	Scatter Search + Hill Climbing + ILS	10,16



Gambar 2.2.2: Persebaran Pendekatan pada *Benchmark Socha*

(Lanjutan) Perbandingan Pemeringkatan Penelitian pada *Benchmark Socha*

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
12	Sabar, Ayob, Kendall and Qu (2012)	Metaheuristik (P)	Honey-Bee Mating Optimization	11,16
13	Rezaeipanah et al. (2021)	Hibrida (P+S)	Genetic Algorithms + Local Search	12,83
14	Abdullah et al. (2012)	Hibrida (P+S)	Electromagnetism-like Mechanism + Great Deluge	13,00

2.2.3 *Benchmark ITC 2007 Track 1*

Benchmark ITC 2007 Track 1 merupakan *benchmark* dengan kumpulan dataset untuk permasalahan penjadwalan ujian. *Benchmark* ini memperkenalkan tingkat kompleksitas lebih tinggi dibandingkan *benchmark* Toronto dengan menambahkan beberapa batasan baru yang diadaptasi dari permasalahan nyata. Terdapat lima *hard constraint* yang dijabarkan

sebagai berikut:

- Tidak ada siswa yang dijadwalkan mengikuti lebih dari satu ujian pada waktu yang sama.
- Kapasitas ruang ujian tidak boleh terlampaui dalam setiap sesi ujian.
- Durasi ujian harus sama atau lebih pendek dari durasi periode yang tersedia.
- Semua batasan terkait periode harus dipenuhi.
- Semua batasan terkait ruangan harus dipenuhi.

Selain itu, *benchmark* ini juga memperkenalkan tujuh *soft constraint* yang dijabarkan sebagai berikut:

- Siswa tidak mengikuti dua ujian berturut-turut.
- Siswa tidak mengikuti lebih dari satu ujian per hari.
- Jadwal ujian siswa tersebar merata pada beberapa periode.
- Ujian dengan durasi berbeda tidak dijadwalkan pada periode yang sama.
- Ujian untuk kelas dengan peserta yang banyak dijadwalkan lebih awal dalam sesi ujian.
- Batasan tambahan terkait periode.
- Batasan tambahan terkait ruangan.

Benchmark ini terdiri dari 12 dataset dengan 8 dataset yang dipublikasikan. Berbeda dari dua *benchmark* sebelumnya, *benchmark* ITC 2007 memiliki batasan waktu eksekusi yang ditentukan oleh perangkat lunak resmi berdasarkan spesifikasi perangkat yang digunakan.

Tabel 2.2.3 menyajikan hasil perbandingan dari 17 studi pada *benchmark* ini. Pendekatan metaheuristik (S) kembali menunjukkan kinerja yang baik, sebagaimana terlihat pada dua *benchmark* sebelumnya. Dua studi teratas menggunakan pendekatan ini, yaitu penelitian oleh Bykov and Petrovic (2016) dengan algoritma Step Counting Hill Climbing dan Burke and Bykov (2016) dengan algoritma Great Deluge. Dua studi lainnya yang menggunakan metaheuristik (S) juga mencapai hasil yang baik, dengan berada pada peringkat ke-5 dan ke-7. Pada kategori pendekatan hibrida, hanya satu studi dari Gogos

et al. (2012) yang menggabungkan dua pendekatan berbasis solusi tunggal, yaitu Steepest Descent dan Simulated Annealing. Penelitian ini berhasil mencapai peringkat ke-6. Sebaliknya, hibrida yang menggabungkan algoritma berbasis populasi dan solusi tunggal tidak menunjukkan hasil yang optimal, dengan peringkat di bawah rata-rata. Pendekatan hibrida lainnya oleh Mandal et al. (2020) juga menunjukkan hasil yang kurang memuaskan, berada pada peringkat ke-11.

Sementara itu pada pendekatan hiper-heuristik, studi oleh Sabar et al. (2013) yang menggunakan hiper-heuristik (G-P) dengan kombinasi Backus Naur Form Grammar dan Genetic Algorithm menghasilkan hasil yang baik dengan menduduki peringkat ke-3. Studi lain dengan hiper-heuristik (S-P) oleh Sabar and Kendall (2015) yang menggunakan Gene Expression Programming juga menunjukkan hasil yang baik dengan meraih peringkat ke-4. Namun, dua studi lain dengan pendekatan serupa oleh Demeester et al. (2012) dan Burke et al. (2014) menunjukkan hasil yang kurang memuaskan dengan berada pada peringkat ke-16 dan ke-17. Pendekatan hiper-heuristik (S-C) juga tidak memberikan hasil optimal, dengan berada pada peringkat ke-14 dan ke-15.

Gambar 2.2.3 menunjukkan grafik persebaran penelitian berdasarkan pendekatan yang digunakan pada *benchmark* ini. Terlihat dominasi pendekatan metaheuristik (S) yang konsisten dengan hasil pada *benchmark* Toronto dan Socha. Beberapa pendekatan lain seperti hiper-heuristik (S-P) dan hiper-heuristik (G-P) menunjukkan kinerja baik, namun belum menunjukkan hasil yang konsisten, dengan beberapa studi menunjukkan hasil yang kurang baik pada *benchmark* ini maupun *benchmark* lainnya. Pendekatan hibrida, terutama hibrida (P+S), juga tidak menunjukkan kinerja optimal seperti pada *benchmark* Socha. Namun, pendekatan hibrida (S+S) yang menggabungkan dua pendekatan berbasis solusi tunggal memberikan hasil lebih baik dibandingkan semua studi hibrida (P+S). Hal ini mengarah kepada kesimpulan bahwa penggunaan algoritma berbasis solusi tunggal cocok untuk permasalahan penjadwalan.

Dari sisi algoritma, berbagai algoritma dikembangkan pada *benchmark* ini. Meskipun Simulated Annealing mendominasi dua *benchmark* sebelumnya, performanya di sini kurang menonjol, dengan berada pada peringkat ke-5, ke-6, dan ke-13. Great Deluge menunjukkan hasil yang baik di peringkat ke-2. Algoritma baru, Step Counting Hill Climbing, yang diperkenalkan oleh Bykov and Petrovic (2016), menunjukkan potensi

dengan meraih peringkat teratas pada *benchmark* ini.

Tabel 2.2.3: Perbandingan Pemeringkatan Penelitian pada *Benchmark* ITC 2007 Track 1

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
1	Bykov and Petrovic (2016)	Metaheuristik (S)	Step Counting Hill Climbing	1,50
2	Burke and Bykov (2016)	Metaheuristik (S)	Great Deluge	1,87
3	Sabar et al. (2015)	Hiper-heuristik (S-P)	Gene Expression Programming	4,12
4	Sabar et al. (2013)	Hiper-heuristik (G-P)	Backus Naur Form Grammar + Genetic Algorithm	4,75
5	Battistutta et al. (2017)	Metaheuristik (S)	Simulated Annealing	5,62
6	Gogos et al. (2012)	Hibrida (S+S)	Steepest Descent + Simulated Annealing	6,00
7	Leite et al. (2019)	Metaheuristik (S)	Simulated Annealing	6,50
8	Abdullah and Turabieh (2012)	Hibrida (P+S)	Memetic + Tabu Search	6,87
9	Alzaqebah and Abdullah (2015)	Hibrida (P+S)	Bee Colony Optimization + Simulated Annealing + Late Acceptance Hill-Climbing	9,25
10	Abdullah and Alzaqebah (2013)	Hibrida (P+S)	Bee Algorithm + Late Acceptance Hill Climbing + Simulated Annealing	10,75
11	Mandal et al. (2020)	Hibrida (other)	Great Deluge + Constructive Method	10,87

(Lanjutan) Perbandingan Pemeringkatan Penelitian pada *Benchmark ITC 2007 Track 1*

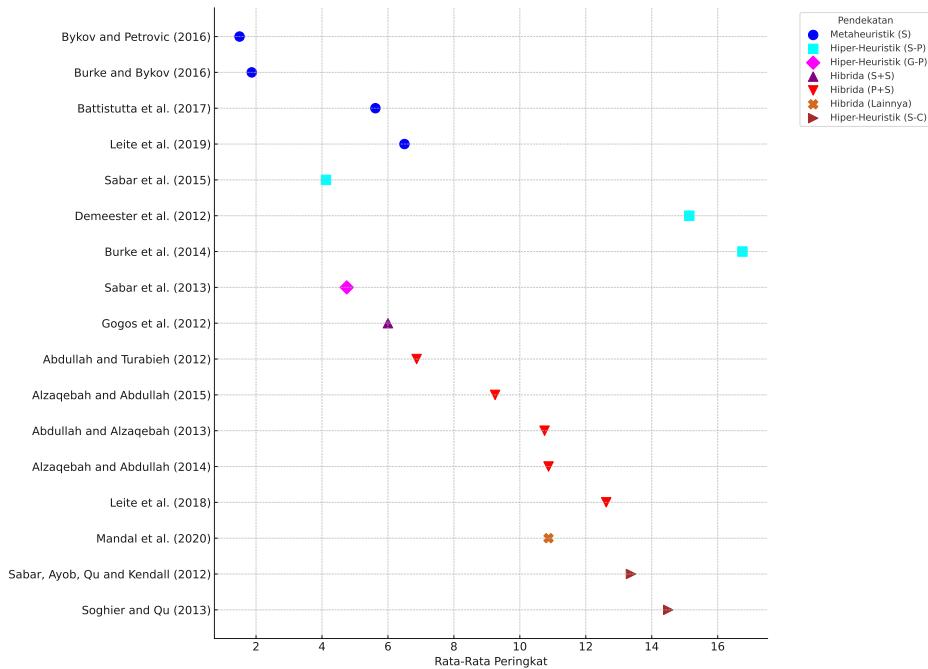
No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
12	Alzaqebah and Abdullah (2014)	Hibrida (P+S)	Artificial Bee Colony + Late Acceptance Hill-Climbing	10,87
13	Leite et al. (2018)	Hibrida (P+S)	Cellular Memetic + Threshold Acceptance	12,62
14	Sabar, Ayob, Qu and Kendall (2012)	Hiper-heuristik (S-C)	Graph Coloring	13,37
15	Soghier and Qu (2013)	Hiper-heuristik (S-C)	Iterative Adaptive	14,50
16	Demeester et al. (2012)	Hiper-heuristik (S-P)	Simulated Annealing	15,14
17	Burke et al. (2014)	Hiper-heuristik (S-P)	Adaptive Selection Heuristic	16,75

2.2.4 *Benchmark ITC 2007 Track 2*

Benchmark ITC 2007 Track 2 merupakan dataset kompetisi yang membahas permasalahan penjadwalan mata kuliah dengan tipe *post-enrollment* (Lewis, 2007). *Benchmark* ini memiliki *soft constraint* yang sama dengan *benchmark Socha*. Sementara pada *hard constraint*, *benchmark* ini menggunakan *hard constraint* yang sama dengan *benchmark Socha* namun dengan penambahan dua *hard constraint* baru yang terdiri dari:

- Mata kuliah harus dialokasikan hanya pada slot waktu yang ditentukan sebagai “tersedia” untuk mata kuliah tersebut.
- Jika diperlukan, mata kuliah harus dijadwalkan dalam urutan yang benar dalam satu minggu.

Benchmark ini memiliki 24 dataset. Sebanyak 11 studi telah menguji algoritma yang



Gambar 2.2.3: Persebaran Pendekatan pada *Benchmark ITC 2007 Track 1*

dikembangkan dengan *benchmark* ini, dan hasil pemeringkatan ditampilkan pada Tabel 2.2.4.

Pendekatan metaheuristik (S) menunjukkan kinerja sangat baik, dengan lima studi yang menempati posisi lima besar. Namun, satu studi oleh Lewis (2012) mendapat peringkat terendah. Hal ini kemungkinan disebabkan oleh pendekatan yang berbeda dalam pengembangan algoritma, yaitu dengan membagi fase menjadi tiga bagian yang dibatasi oleh waktu program berjalan. Pendekatan lainnya, seperti hibrida (P+S), hiper-heuristik (G-S) dan hiper-heuristik (P-S), hanya mencapai di antara hasil rata-rata atau kurang memuaskan.

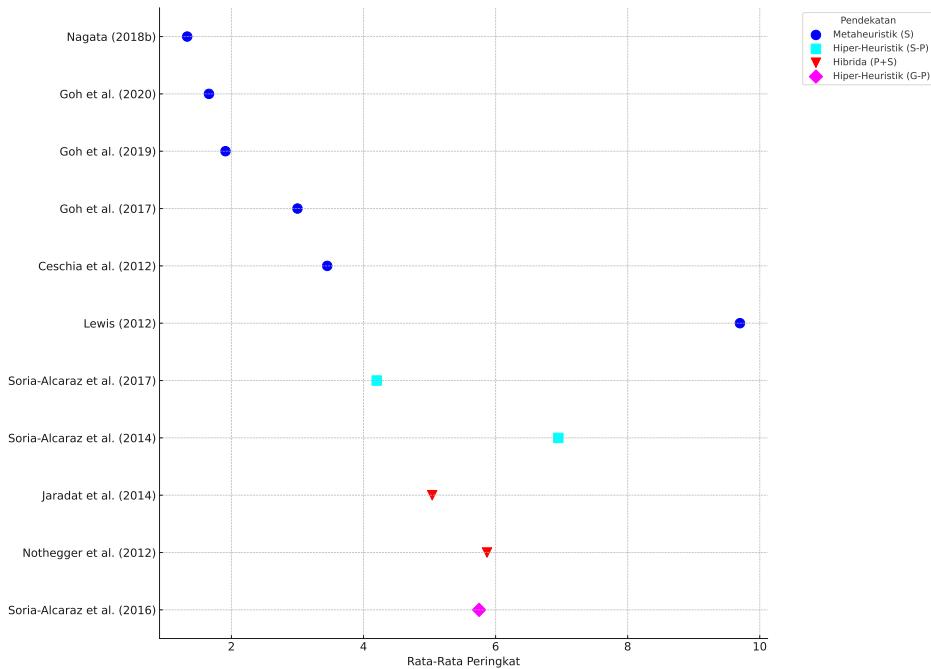
Gambar 2.2.4 menunjukkan distribusi persebaran pendekatan yang digunakan pada *benchmark* ini. Gambar ini memperlihatkan dominasi pendekatan metaheuristik (S) yang konsisten lebih unggul dibandingkan pendekatan lainnya, sejalan dengan temuan pada tiga *benchmark* sebelumnya. Tren ini menunjukkan bahwa pendekatan berbasis solusi tunggal lebih efektif untuk permasalahan penjadwalan. Sebaliknya, pendekatan lainnya tidak menunjukkan tren positif, dengan hasil peringkat yang berada di sekitar rata-rata.

Dari segi algoritma, Simulated Annealing kembali menunjukkan kinerja yang baik, dengan berada pada peringkat antara ke-2 hingga ke-5, kecuali satu studi yang menempati

peringkat terakhir. Algoritma Random Partial Neighborhood Search juga menunjukkan kinerja baik, sebagaimana yang ditemukan pada *benchmark* Socha. Sementara itu algoritma lain yaitu ILS, yang digunakan dalam tiga penelitian, hanya menghasilkan performa di sekitar rata-rata.

Tabel 2.2.4: Perbandingan Penelitian pada *Benchmark* ITC 2007 Track 2

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
1	Nagata (2018b)	Metaheuristik (S)	Random Partial Neighbourhood Search	1,33
2	Goh et al. (2020)	Metaheuristik (S)	Simulated Annealing	1,66
3	Goh et al. (2019)	Metaheuristik (S)	Simulated Annealing	1,91
4	Goh et al. (2017)	Metaheuristik (S)	Simulated Annealing	3,00
5	Ceschia et al. (2012)	Metaheuristik (S)	Simulated Annealing	3,45
6	Soria-Alcaraz et al. (2017)	Hiper-heuristik (S-P)	ILS	4,20
7	Jaradat et al. (2014)	Hibrida (P+S)	Scatter Search + Hill Climbing + ILS	5,04
8	Soria-Alcaraz et al. (2016)	Hiper-heuristik (G-P)	ILS	5,75
9	Nothegger et al. (2012)	Hibrida (P+S)	Ant Colony Optimization + Simulated Annealing	5,87
10	Soria-Alcaraz et al. (2014)	Hiper-heuristik (S-P)	ILS	6,95
11	Lewis (2012)	Metaheuristik (S)	Simulated Annealing	9,70



Gambar 2.2.4: Persebaran Pendekatan pada *Benchmark ITC 2007 Track 2*

2.2.5 *Benchmark ITC 2007 Track 3*

Benchmark ITC 2007 Track 3 adalah kumpulan dataset yang mengangkat permasalahan penjadwalan mata kuliah berbasis kurikulum yang diambil dari studi kasus nyata di Universitas Udine (Di Gaspero et al., 2007). Masalah ini mencakup empat *hard constraint* dan empat *soft constraint* (Soria-Alcaraz et al., 2016). Empat *hard constraint* yang digunakan terdiri dari :

- Semua perkuliahan harus dijadwalkan, dengan masing-masing dialokasikan pada periode yang berbeda.
- Dua perkuliahan tidak boleh dijadwalkan dalam ruang yang sama pada periode yang sama.
- Perkuliahan dari mata kuliah dalam kurikulum yang sama atau yang diajarkan oleh dosen yang sama harus dijadwalkan pada periode berbeda.
- Perkuliahan harus dijadwalkan sesuai dengan ketersediaan waktu dan persyaratan pengajar.

Sementara itu, empat *soft constraint* yang berlaku terdiri dari:

- Semua siswa diupayakan untuk bisa menghadiri perkuliahan (tanpa konflik waktu atau keterbatasan kapasitas ruangan).
- Perkuliahan untuk setiap mata kuliah harus tersebar di atas jumlah minimum hari yang disyaratkan.
- Perkuliahan dalam kurikulum yang sama sebaiknya dijadwalkan secara berurutan.
- Semua perkuliahan untuk satu mata kuliah sebaiknya dijadwalkan di ruang yang sama.

Benchmark ini memiliki 21 dataset dan mengikuti aturan batasan waktu yang sama seperti pada *benchmark* ITC 2007 Track 1 dan Track 2. Sebanyak 7 studi telah menguji algoritma yang dikembangkan menggunakan *benchmark* ini, dengan hasil perbandingan ditampilkan pada Tabel 2.2.5.

Pada *benchmark* ini, pendekatan hiper-heuristik (G-P) yang digunakan oleh Soria-Alcaraz et al. (2016) mengungguli pendekatan lain dengan memperoleh peringkat teratas di seluruh dataset. Sementara itu pada pendekatan hibrida, studi oleh Abdullah and Turabieh (2012) yang menggunakan strategi hibrida (P+S) menunjukkan kinerja yang baik, dengan berada pada peringkat kedua. Namun, studi lain dengan pendekatan serupa oleh Abdullah et al. (2012) hanya mencapai peringkat keenam. Pendekatan hibrida lainnya yaitu hibrida (S+S) memberikan hasil rata-rata, menempati peringkat ke-4 dan ke-5. Sementara pada pendekatan metaheuristik (S) menunjukkan hasil yang kurang dominan, dengan menempati peringkat ke-3 dan ke-7.

Distribusi persebaran penelitian berdasarkan pendekatan yang digunakan pada *benchmark* ini ditampilkan dalam Gambar 2.2.5. Meskipun tidak ada pendekatan yang menunjukkan dominasi mutlak pada *benchmark* ini, hasil ini masih mencerminkan tren yang sama seperti pada *benchmark* sebelumnya. Pendekatan hiper-heuristik (G-P) yang menempati peringkat teratas, dikembangkan menggunakan strategi berbasis solusi tunggal. Hal ini memperkuat bukti efektivitas pendekatan solusi tunggal dalam masalah penjadwalan. Pendekatan metaheuristik (S) juga berkinerja baik dengan menempati peringkat ketiga. Hal ini semakin mendukung kesimpulan bahwa algoritma berbasis solusi tunggal sangat cocok untuk permasalahan penjadwalan.

Dari segi kinerja algoritma, Simulated Annealing dan Great Deluge, yang

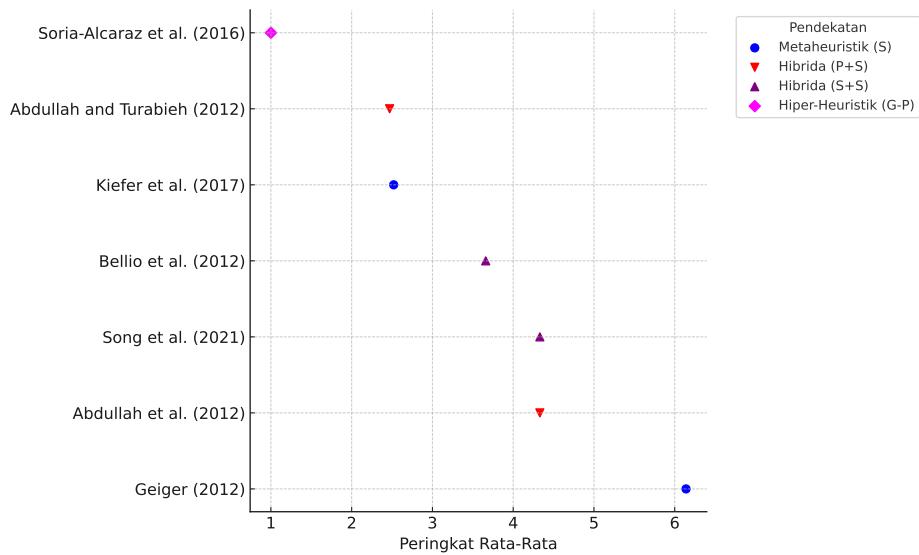
menunjukkan kinerja baik pada *benchmark* sebelumnya, hanya menunjukkan kinerja rata-rata pada *benchmark* ini. Algoritma Simulated Annealing menempati peringkat ke-4 dan ke-5. Sementara algoritma Great Deluge berada di peringkat ke-6. Algoritma ILS, yang sebelumnya menghasilkan performa moderat pada *benchmark* ITC 2007 Track 2, menunjukkan hasil yang lebih baik pada *benchmark* ini. Tabu Search juga menunjukkan kinerja yang baik dengan menempati peringkat ke-2 dan ke-3.

Tabel 2.2.5: Perbandingan Penelitian pada *Benchmark* ITC 2007 Track 3

No	Penelitian	Pendekatan	Algoritma	Peringkat Rata-Rata
1	Soria-Alcaraz et al. (2016)	Hiper-heuristik (G-P)	ILS	1,00
2	Abdullah and Turabieh (2012)	Hibrida (P+S)	Memetic + Tabu Search	2,47
3	Kiefer et al. (2017)	Metaheuristik (S)	Adaptive Large Neighbourhood Search	2,52
4	Bellio et al. (2012)	Hibrida (S+S)	Simulated Annealing + Dynamic Tabu Search	3,66
5	Song et al. (2021)	Hibrida (S+S)	Simulated Annealing	4,33
6	Abdullah et al. (2012)	Hibrida (P+S)	Electromagnetism-like Mechanism + Great Deluge	4,33
7	Geiger (2012)	Metaheuristik (S)	Threshold Acceptance	6,14

2.2.6 *Benchmark* ITC 2019

Benchmark ITC 2019 adalah *benchmark* terbaru yang diambil dari kasus nyata dalam penjadwalan mata kuliah berbasis kurikulum. *Benchmark* ini mencakup 30 dataset dan memiliki 19 jenis batasan yang dapat diterapkan sebagai *hard* atau *soft constraint* bergantung pada permasalahan pada setiap dataset. Deskripsi lengkap permasalahan ini dibahas pada Bab 6. Banyaknya jumlah batasan serta ukuran dataset yang lebih besar



Gambar 2.2.5: Persebaran Pendekatan pada *Benchmark* ITC 2007 Track 3

dibandingkan *benchmark* sebelumnya menyebabkan pencarian solusi *feasible* dan proses optimasi menjadi sangat menantang.

Tabel 2.2.6 menampilkan hasil peringkat dan persentase dataset dengan solusi *feasible* pada *benchmark* ITC 2019. Terdapat tiga metode yang dikembangkan oleh penelitian sebelumnya. Metode pertama adalah metode metaheuristik. Dalam metode ini terdapat dua pendekatan yaitu metaheuristik (S) yang digunakan dalam studi oleh Sylejmani et al. (2022) dan Premananda and Muklason (2021), serta pendekatan hibrida (P+S) oleh Premananda, Tjahyanto and Muklason (2022). Hasil dari berbagai pendekatan dalam metode metaheuristik ini hanya menghasilkan peringkat 3 ,4 dan 6. Namun tiga penelitian dengan metode ini menunjukkan hasil yang baik dalam aspek menemukan solusi *feasible*, dengan hasil ketiga penelitian mampu menemukan seluruh solusi *feasible*. Selain itu, pendekatan solusi tunggal dan algoritma Simulated Annealing yang diterapkan dalam penelitian oleh Sylejmani et al. (2022) menunjukkan hasil yang baik pada *benchmark* ini.

Metode kedua adalah metode *matheuristic*, yaitu kombinasi antara metode eksak dan heuristik. Metode ini menunjukkan hasil yang bervariasi. Studi oleh Mikkelsen and Holm (2022) dan Rappos et al. (2022) menunjukkan kinerja yang sangat baik dengan berada pada peringkat 1 dan 2. Sementara studi lainnya oleh Lemos et al. (2020) dan Lemos et al. (2021) menunjukkan hasil yang kurang memuaskan dengan berada pada peringkat 5 dan 8. Pendekatan ketiga adalah metode eksak dengan menggunakan Mixed Integer Programming

(MIP). Hasil dari penggunaan metode ini tidak memuaskan, dengan menempati peringkat ketujuh. Secara keseluruhan, metode eksak campuran seperti *matheuristic* maupun metode eksak murni menunjukkan hasil yang tidak konsisten dari sisi feasibilitas. Sebagai contoh, meskipun Rappos et al. (2022) menempati peringkat kedua, terdapat satu dataset yang tidak berhasil ditemukan solusi *feasible*. Sementara itu, studi oleh Holm et al. (2022) dan Lemos et al. (2020) menghasilkan solusi *feasible* hanya untuk 10 dan 20 dataset.

Tabel 2.2.6: Perbandingan Penelitian pada *Benchmark ITC 2019*

No	Penelitian	Pendekatan	Peringkat Rata-Rata	Persentase Feasibilitas
1	Mikkelsen and Holm (2022)	<i>Matheuristic</i>	1,11	100%
2	Rappos et al. (2022)	<i>Matheuristic</i>	2,66	97%
3	Sylejmani et al. (2022)	Metaheuristik (S)	3,36	100%
4	Premananda, Tjahyanto and Muklason (2022)	Hibrida(P+S)	4,66	100%
5	Lemos et al. (2021)	<i>Matheuristic</i>	5,18	100%
6	Premananda and Muklason (2021)	Metaheuristik (S)	5,86	100%
7	Holm et al. (2022)	MIP	6,20	33%
8	Lemos et al. (2020)	<i>Matheuristic</i>	6,93	67%

2.2.7 *Benchmark ITC 2021*

Benchmark ITC 2021 adalah *benchmark* terbaru yang mengangkat permasalahan penjadwalan dalam kompetisi olahraga. Permasalahan ini terdiri dari 45 dataset buatan dan memiliki sembilan jenis batasan yang dapat diklasifikasikan sebagai *hard* atau *soft constraint*, bergantung pada permasalahan dalam setiap dataset. Penjelasan detail mengenai permasalahan ini dapat dilihat pada Bab 7. Meskipun jumlah jenis batasan lebih sedikit dibandingkan batasan pada *benchmark ITC 2019*, permasalahan pada *benchmark ITC 2021* sangat menantang terutama dalam menghasilkan solusi *feasible*. Dari 14 penelitian yang ditinjau, hanya satu yang berhasil menghasilkan solusi *feasible* untuk

seluruh dataset (Van Bulck and Goossens, 2023).

Hasil pemeringkatan dari 14 penelitian ditampilkan dalam Tabel 2.2.7 dengan mendeskripsikan pendekatan yang digunakan, rata-rata peringkat dan persentase keberhasilan menemukan solusi *feasible*. Dari 14 penelitian, tidak semua penelitian melaporkan hasil optimasinya. Terdapat enam penelitian yang hanya melaporkan hasil dalam menemukan solusi *feasible*, sehingga enam penelitian tersebut tidak diikutkan dalam pemeringkatan.

Dari hasil ini, metode eksak menjadi metode yang paling dominan. Sebanyak 7 dari 14 penelitian menggunakan metode eksak dengan pendekatan seperti Integer Linear Programming (ILP), Mixed Integer Linear Programming (MILP), MIP dan Pseudo-Boolean Optimization (PBO). Penggunaan metode eksak menghasilkan hasil yang bervariasi. Beberapa penelitian menunjukkan hasil yang sangat baik, seperti Lamas-Fernandez et al. (2021) yang berhasil menghasilkan solusi *feasible* untuk seluruh dataset dan menempati peringkat pertama dalam hasil keseluruhan. Sementara penelitian lain dengan metode eksak, menunjukkan hasil yang kurang baik seperti pada penelitian oleh Lester (2022), yang hanya menghasilkan 73% solusi *feasible* dan menempati peringkat terakhir dalam hasil keseluruhan.

Pendekatan kedua yang banyak digunakan adalah *hybrid matheuristic*, dengan tiga penelitian yang menggunakan pendekatan ini. Hasil dari pendekatan ini juga bervariasi. Penelitian oleh Fonseca and Toffolo (2022) menunjukkan hasil yang baik dalam aspek optimasi dan feasibilitas, sedangkan penelitian oleh Phillips et al. (2021) dan Dimitras et al. (2022) menunjukkan hasil yang kurang memuaskan, terutama dalam aspek optimasi.

Pada metode heuristik, hanya terdapat dua pendekatan, yaitu metaheuristik (S) dan hibrida (S+S). Pendekatan metaheuristik (S) yang digunakan oleh Rosati et al. (2022) menunjukkan hasil yang sangat baik, dengan peringkat kedua dalam hasil keseluruhan dan dengan persentase memperoleh solusi *feasible* mencapai 97%. Sementara itu, penelitian lain yang menggunakan pendekatan hibrida (S+S) oleh Hutama and Muklason (2021) menunjukkan hasil yang kurang memuaskan. Penelitian ini berfokus pada pengembangan algoritma untuk menghasilkan solusi *feasible*, namun hanya menghasilkan solusi *feasible* dengan persentase keberhasilan sebesar 42%.

Tabel 2.2.7: Perbandingan Penelitian pada *Benchmark* ITC 2021

No	Penelitian	Pendekatan	Peringkat Rata-Rata	Persentase Feasibilitas
1	Lamas-Fernandez et al. (2021)	ILP	1,40	100%
2	Rosati et al. (2022)	Metaheuristik (S)	2,42	97%
3	Fonseca and Toffolo (2022)	<i>Matheuristic</i>	3,92	82%
4	Berthold et al. (2021)	MILP	4,23	88%
5	Phillips et al. (2021)	<i>Matheuristic</i>	5,02	82%
6	Dimitsas et al. (2022)	<i>Matheuristic</i>	5,10	82%
7	Subba and Stordal (2021)	MILP	6,55	88%
8	Lester (2022)	PBO	7,34	73%
9	Giorgio Sartor & Bjørnar Luteberget	-	-	88%
10	van Doornmalen et al. (2021)	MIP	-	84%
11	Sumin and Rodin (2021)	MILP	-	82%
12	Arbaoui Taha et al.	-	-	68%
13	Swarup Ghadiali	PBO	-	57%
14	Hutama and Muklason (2021)	Hibrida (S+S)	-	42%

2.2.8 Tingkat Generalitas dalam Algoritma Penjadwalan

Pengembangan algoritma yang dapat diterapkan secara umum untuk berbagai permasalahan penjadwalan merupakan tujuan penting. Sebagaimana dibahas pada Subbab 2.1.5, terdapat tiga tingkatan generalitas. Tingkat pertama berkaitan dengan algoritma yang dirancang khusus untuk dataset tertentu dalam satu *benchmark*. Sebagian besar penelitian telah berhasil mencapai tingkat generalitas ini, khususnya penelitian-penelitian yang dianalisis dalam kajian ini. Penelitian-penelitian tersebut tidak hanya mampu menghasilkan solusi yang *feasible*, tetapi juga mengoptimalkan *soft constraint* di seluruh dataset dalam *benchmark* yang sama.

Sementara itu, pada tingkatan generalitas kedua yaitu antara *benchmark* dan tingkatan

generalitas ketiga yaitu antara jenis permasalahan, hasil penelitian terdahulu tidak menunjukkan hasil sebaik pada tingkatan generalitas pertama. Hal ini menyebabkan pada subbab ini memfokuskan pembahasan perbandingan antara penelitian terdahulu dalam tingkatan generalitas kedua dan ketiga. Tabel 2.2.8 menyajikan deskripsi rinci dari 20 studi, mencakup *benchmark* yang digunakan, peringkat yang diperoleh, serta informasi apakah parameter yang sama diterapkan pada dua *benchmark* atau permasalahan yang berbeda menggunakan nilai parameter yang sama atau tidak.

Pada tingkat generalitas kedua, terdapat dua kombinasi *benchmark* yang berbeda. Kombinasi pertama berfokus pada penjadwalan ujian menggunakan *benchmark* Toronto dan ITC 2007 Track 1 yang digunakan oleh 10 penelitian. Studi oleh Burke and Bykov (2016) menunjukkan kinerja yang baik pada kedua *benchmark*, dengan mendapatkan peringkat ke-3 dari 26 studi pada *benchmark* Toronto dan peringkat ke-2 dari 17 studi pada *benchmark* ITC 2007 Track 1. Namun, penelitian ini menggunakan pengaturan parameter yang berbeda untuk kedua *benchmark* tersebut. Sebaliknya, tiga studi lain oleh Abdullah and Alzaqebah (2013), Alzaqebah and Abdullah (2014) dan Alzaqebah and Abdullah (2015) menunjukkan hasil yang konsisten dengan berada pada peringkat rata-rata, namun menggunakan nilai parameter yang sama. Di sisi lain, tiga studi lainnya menunjukkan hasil yang tidak konsisten, yaitu menghasilkan solusi yang baik pada satu *benchmark* tetapi kurang baik pada *benchmark* lainnya. Sebagai contoh, studi oleh Leite et al. (2018) menempati peringkat ke-2 pada *benchmark* Toronto tetapi hanya menempati peringkat ke-13 pada *benchmark* ITC 2007 Track 1. Pola serupa juga ditemukan pada studi oleh Demeester et al. (2012), Mandal et al. (2020) dan Sabar et al. (2013). Terakhir, studi oleh Sabar, Ayob, Qu and Kendall (2012) dan Burke et al. (2014) menunjukkan kinerja yang kurang baik pada kedua *benchmark*.

Pada kombinasi kedua, terdapat lima studi yang berfokus pada permasalahan penjadwalan mata kuliah dengan tipe *post-enrollment* menggunakan *benchmark* Socha dan ITC 2007 Track 2. Studi oleh Goh et al. (2020) menunjukkan kinerja yang baik pada kedua *benchmark*. Pada setiap *benchmark*, penelitian ini menggunakan nilai parameter yang berbeda, namun nilai parameter ini ditentukan secara otomatis melalui prosedur awal sebelum optimasi dijalankan. Hal ini menyebabkan penelitian tersebut termasuk pada penelitian yang memiliki tingkat generalitas yang baik. Dua studi lain oleh Goh et al. (2019) dan Goh et al. (2017), yang ditulis oleh peneliti yang sama dengan metode serupa,

juga menunjukkan hasil yang konsisten dan positif pada kedua *benchmark*. Penelitian oleh Nagata (2018b) juga memberikan hasil yang baik pada kedua *benchmark*. Namun penelitian ini menggunakan pengaturan parameter yang berbeda. Sementara itu studi lainnya oleh Jaradat et al. (2014) tidak menunjukkan kinerja yang baik pada kedua *benchmark*.

Untuk tingkat generalitas ketiga, terdapat tiga kombinasi jenis permasalahan dan *benchmark* yang berbeda. Kombinasi pertama berfokus pada penjadwalan ujian dan mata kuliah dengan tipe *post-enrollment* dengan menggunakan *benchmark* Toronto dan Socha. Terdapat dua studi yang menggunakan kombinasi ini. Kedua studi tersebut menggunakan nilai parameter yang sama untuk kedua jenis permasalahan. Studi pertama oleh Fong et al. (2014a) menunjukkan kinerja yang baik, dengan mendapatkan peringkat ke-6 dari 26 studi pada *benchmark* Toronto dan peringkat ke-8 dari 14 studi pada *benchmark* Socha. Studi kedua oleh Sabar, Ayob, Kendall and Qu (2012) memiliki hasil yang sedikit lebih rendah, yaitu peringkat ke-11 pada *benchmark* Toronto dan peringkat ke-12 pada *benchmark* Socha.

Kombinasi kedua berfokus pada penjadwalan ujian dan mata kuliah berbasis kurikulum dengan menggunakan *benchmark* ITC 2007 Track 1 dan Track 3. Pada kombinasi ini, hanya terdapat satu studi oleh Abdullah and Turabieh (2012), yang menunjukkan kinerja yang baik, dengan berada pada peringkat ke-8 dari 17 studi untuk ITC 2007 Track 1 dan peringkat ke-2 dari 7 studi untuk ITC 2007 Track 3. Kombinasi ketiga berfokus pada penjadwalan mata kuliah dengan tipe *post-enrollment* dan penjadwalan mata kuliah berbasis kurikulum. Terdapat dua penelitian yang menggunakan kombinasi ini. Studi pertama oleh Abdullah et al. (2012) menggunakan *benchmark* Socha dan ITC 2007 Track 3. Hasil yang ditunjukkan oleh studi ini tidak terlalu baik, dengan peringkat berada pada peringkat terakhir untuk *benchmark* Socha dan peringkat ke-6 dari 7 studi pada *benchmark* ITC 2007 Track 3. Studi kedua oleh Soria-Alcaraz et al. (2017), menggunakan kombinasi *benchmark* yang berbeda yaitu *benchmark* ITC 2007 Track 2 dan Track 3. Hasil studi ini menunjukkan hasil yang tidak konsisten, dengan peringkat ke-8 dari 12 studi pada *benchmark* ITC 2007 Track 2, namun berhasil meraih peringkat teratas pada *benchmark* ITC 2007 Track 3, menempati peringkat pertama dari 7 studi.

Tabel 2.2.8: Perbandingan Penelitian pada Tingkatan Generalitas *Benchmark* dan Permasalahan

Penelitian	<i>Benchmark</i>	Peringkat	Total Penelitian	Nilai Parameter General
Burke and Bykov (2016)	Carter	3	26	Tidak
	ITC 2007 Track 1	2	17	
Leite et al. (2018)	Carter	2	26	Tidak
	ITC 2007 Track 1	13	17	
Demeester et al. (2012)	Carter	4	26	Tidak
	ITC 2007 Track 1	16	17	
Mandal et al. (2020)	Carter	5	26	Tidak
	ITC 2007 Track 1	11	17	
Alzaqebah and Abdullah (2015)	Carter	7	26	Ya
	ITC 2007 Track 1	9	17	
Alzaqebah and Abdullah (2014)	Carter	9	26	Ya
	ITC 2007 Track 1	12	17	
Sabar, Ayob, Qu and Kendall (2012)	Carter	26	26	Tidak
	ITC 2007 Track 1	14	17	
Abdullah and Alzaqebah (2013)	Carter	12	26	Ya
	ITC 2007 Track 1	10	17	
Burke et al. (2014)	Carter	22	26	Ya
	ITC 2007 Track 1	17	17	
Sabar et al. (2013)	Carter	14	26	Ya
	ITC 2007 Track 1	4	17	
Fong et al. (2014a)	Carter	6	26	Ya
	Socha	8	14	
Sabar, Ayob, Kendall and Qu (2012)	Carter	11	25	Ya
	Socha	12	14	
Goh et al. (2020)	Socha	1	14	Tidak
	ITC 2007 Track 2	2	11	

(Lanjutan) Perbandingan Penelitian pada Tingkatan Generalitas *Benchmark* dan Permasalahan

Penelitian	<i>Benchmark</i>	Peringkat	Total Penelitian	Nilai Parameter General
Goh et al. (2019)	Socha	2	14	Ya
	ITC 2007 Track 2	3	11	
Goh et al. (2017)	Socha	3	14	Ya
	ITC 2007 Track 2	4	11	
Nagata (2018b)	Socha	4	14	Tidak
	ITC 2007 Track 2	1	11	
Jaradat et al. (2014)	Socha	11	14	Ya
	ITC 2007 Track 2	7	11	
Abdullah et al. (2012)	Socha	14	14	Ya
	ITC 2007 Track 3	6	7	
Abdullah and Turabieh (2012)	ITC 2007 Track 1	8	17	Ya
	ITC 2007 Track 3	2	7	
Soria-Alcaraz et al. (2016)	ITC 2007 Track 2	8	12	Ya
	ITC 2007 Track 3	1	7	

2.2.9 Penelitian dengan Fokus Pencarian Solusi *Feasible*

Dalam permasalahan penjadwalan, sebelum dipublikasikannya *benchmark* ITC 2019 dan ITC 2021, terdapat satu *benchmark* yang diperkenalkan oleh Lewis and Paechter (2007) yang memiliki permasalahan menantang dalam menghasilkan solusi *feasible*. *Benchmark* ini mengangkat permasalahan penjadwalan mata kuliah yang terdiri dari 60 dataset. Berdasarkan penelitian sebelumnya, terdapat dua studi yang menunjukkan hasil yang sangat baik.

Penelitian pertama oleh Song et al. (2018) mengembangkan algoritma berbasis ILS dan menggabungkannya dengan Simulated Annealing. Algoritma Simulated Annealing diterapkan pada proses *local search* dalam algoritma ILS. Penelitian ini menunjukkan hasil yang sangat baik, dengan berhasil menghasilkan solusi *feasible* pada 58 dari 60 dataset.

Penelitian lainnya dengan hasil yang juga baik dilakukan oleh Goh et al. (2020). Penelitian ini menggabungkan algoritma Tabu Search Sampling Perturbation dengan algoritma ILS. Penggabungan ini dilakukan dengan menjalankan kedua algoritma secara bergantian hingga solusi *feasible* ditemukan. Penelitian ini berhasil menghasilkan solusi *feasible* pada 57 dari 60 dataset.

2.3 Analisis Kebaruan Penelitian

Subbab 2.2 telah menghasilkan analisis dari penelitian sebelumnya dengan melakukan pemeringkatan terhadap setiap penelitian yang memiliki kesamaan penggunaan *benchmark* dan melakukan analisis terhadap tingkatan generalitas. Dari hasil tersebut dapat disimpulkan bahwa pada tingkatan generalitas pertama yaitu menghasilkan solusi yang konsisten baik pada satu jenis permasalahan dalam satu jenis *benchmark*, penelitian terdahulu telah menunjukkan hasil yang sangat baik. Semua penelitian yang dianalisis menguji algoritma yang dikembangkan pada beberapa dataset dalam satu *benchmark* yang sama. Beberapa penelitian menghasilkan hasil yang sangat baik dengan mengungguli penelitian lainnya seperti Bellio et al. (2021), Goh et al. (2020) dan Bykov and Petrovic (2016). Secara keseluruhan penelitian pada tingkatan generalitas pertama sudah menunjukkan hasil yang sangat baik.

Pada tingkatan generalitas kedua, yaitu menghasilkan solusi yang konsisten baik pada dua atau lebih *benchmark* dengan jenis permasalahan yang sama, beberapa penelitian telah menunjukkan hasil yang baik. Penelitian oleh Goh et al. (2020) menghasilkan hasil yang sangat baik dengan mencapai peringkat teratas pada dua *benchmark* pada permasalahan penjadwalan mata kuliah dengan tipe *post-enrolment*. Studi lainnya Alzaqebah and Abdullah (2014) dan Alzaqebah and Abdullah (2015), meski tidak sebaik hasil dari studi Goh et al. (2020), keduanya melaporkan hasil yang konsisten baik pada dua *benchmark* untuk permasalahan penjadwalan ujian. Sementara itu hasil pada tingkat generalitas ketiga yaitu mampu menghasilkan solusi yang konsisten baik pada permasalahan lintas domain, terdapat dua studi yang menunjukkan hasil yang konsisten yaitu Abdullah and Turabieh (2012) dan Fong et al. (2014a). Kedua studi berhasil menghasilkan peringkat di atas rata-rata pada dua jenis permasalahan berbeda.

Dari hasil tersebut, terdapat beberapa keterbatasan yang menjadi dasar pengembangan

penelitian ini. Keterbatasan pertama adalah jenis permasalahan yang digunakan. Hasil penelitian pada tingkatan generalitas ketiga hanya teruji pada dua jenis permasalahan yaitu permasalahan penjadwalan ujian dan penjadwalan mata kuliah. Sementara dalam penjadwalan tidak hanya ada penjadwalan mata kuliah dan ujian namun terdapat jenis penjadwalan lainnya seperti penjadwalan dalam kompetisi olah raga yang memiliki model permasalahan yang jauh berbeda dengan penjadwalan mata kuliah dan ujian. Oleh karena itu, diperlukan pengujian yang lebih beragam untuk membuktikan keberhasilan dalam tingkat generalitas ini.

Keterbatasan kedua adalah terdapat pada *benchmark* yang digunakan. Penelitian oleh Fong et al. (2014b) menggunakan *benchmark* Toronto yang dipublikasikan pada tahun 1996 dan *benchmark* Socha yang dipublikasikan pada tahun 2004. Sementara penelitian oleh Abdullah and Turabieh (2012) menggunakan *benchmark* yang dipublikasikan pada tahun 2007. Penggunaan *benchmark* yang lawas memungkinkan algoritma tersebut tidak relevan untuk diterapkan pada permasalahan saat ini. Sebagai contoh salah satu permasalahan penjadwalan mata kuliah terbaru yaitu ITC 2019 yang menggunakan *real-world* dataset, memiliki 19 jenis batasan. Sementara itu penjadwalan mata kuliah pada *benchmark* Socha hanya memiliki lima jenis batasan dan pada *benchmark* ITC 2007 hanya memiliki 7 hingga 8 jenis batasan. Besaran permasalahan juga menunjukkan perbedaan signifikan. ITC 2019 pada dataset terbesarnya memiliki 8,813 kelas dan 38,437 mahasiswa yang harus dijadwalkan. Sementara pada *benchmark* Socha hanya memiliki 400 kelas dan 400 mahasiswa. Pada *benchmark* ITC 2007 hanya 400 kelas dan 1000 mahasiswa. Oleh karena itu, hasil kedua penelitian tersebut tidak mencerminkan kompleksitas dan kendala permasalahan penjadwalan saat ini.

Dari keterbatasan tersebut maka penelitian ini mengembangkan penelitian dengan fokus mengembangkan kebaruan pada algoritma penjadwalan untuk menghasilkan solusi *feasible* dan mengoptimasi solusi yang ditujukan pada jenis permasalahan yang lebih beragam seperti pada *benchmark* Toronto, yang merupakan *benchmark* klasik, *benchmark* terbaru dalam permasalahan penjadwalan mata kuliah yaitu *benchmark* ITC 2019 dan *benchmark* terbaru pada permasalahan penjadwalan kompetisi olahraga yaitu *benchmark* ITC 2021 yang belum pernah diuji pada penelitian sebelumnya dalam konteks generalitas. Selain itu untuk meningkatkan pembuktian generalitas, permasalahan penjadwalan baru berdasarkan studi kasus nyata pada Departemen Sistem Informasi ITS juga digunakan

dalam pengujian algoritma. Dengan pengujian yang beragam, algoritma yang dikembangkan memiliki bukti yang lebih untuk dapat menunjukkan bahwa algoritma yang dikembangkan memiliki kemampuan dalam menyelesaikan permasalahan penjadwalan lintas domain.

BAB 3

METODOLOGI PENELITIAN

Pengembangan algoritma generik untuk menyelesaikan permasalahan penjadwalan lintas domain merupakan permasalahan yang kompleks. Permasalahan ini tidak hanya menuntut untuk menghasilkan solusi yang *feasible* tanpa melanggar *hard constraint*, tetapi juga untuk mampu mengurangi jumlah pelanggaran *soft constraint* untuk meningkatkan kualitas jadwal yang dihasilkan. Selain itu, algoritma yang dikembangkan diharapkan dapat menyelesaikan berbagai variasi permasalahan penjadwalan dengan hasil yang baik dan konsisten. Untuk melakukan penelitian tersebut, diperlukan pemilihan metodologis yang tepat.

Metodologi *Design Science Research* (DSR) merupakan metodologi yang cocok untuk digunakan dalam penelitian ini karena beberapa alasan. Pertama, DSR mendukung penelitian yang berfokus pada pengembangan artefak, seperti pengembangan algoritma, yang bertujuan untuk memecahkan masalah praktis sekaligus memberikan kontribusi teoretis. Selain itu, pendekatan DSR memungkinkan adanya iterasi antara teori dan praktik, sehingga memastikan bahwa artefak yang dihasilkan tidak hanya efektif secara fungsional tetapi juga memiliki dasar ilmiah yang kuat. Dengan mempertimbangkan kesesuaian ini, metodologi DSR dipilih sebagai metodologi yang digunakan dalam penelitian ini.

Mengacu pada kerangka DSR yang dikemukakan oleh Johannesson and Perjons (2014), penelitian ini dikembangkan melalui tahapan-tahapan berikut:

1. Identifikasi Masalah: Tahapan ini melakukan analisis mendalam terhadap generalitas algoritma dalam permasalahan penjadwalan untuk dapat menemukan kesenjangan yang akan dipenuhi dalam penelitian ini.

2. Penentuan Tujuan Solusi: Tahapan ini menetapkan kriteria dan tujuan yang harus dicapai oleh algoritma yang dikembangkan, baik dari sisi praktis maupun teoretis.
3. Desain dan Pengembangan Algoritma: Tahapan ini mengembangkan algoritma menggunakan metode hiper-heuristik berbasis algoritma *Iterated Local Search* (ILS), yang diharapkan mampu menangani berbagai jenis permasalahan penjadwalan.
4. Demonstrasi: Tahapan ini mengimplementasikan algoritma yang dikembangkan pada berbagai studi kasus, seperti penjadwalan ujian, mata kuliah, kompetisi olahraga, dan permasalahan penjadwalan pada Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS), untuk menunjukkan bagaimana kemampuan algoritma saat diimplementasikan pada beberapa jenis permasalahan yang berbeda.
5. Evaluasi: Tahapan ini menilai kinerja algoritma dengan melakukan perbandingan terhadap penelitian terdahulu dan pada beberapa algoritma pembanding untuk melihat tingkat generalitas pada algoritma yang dikembangkan.

Penjelasan lebih detail terkait lima tahapan tersebut disajikan secara berurutan pada Subbab 3.1 hingga 3.5.

3.1 Identifikasi Permasalahan

Proses identifikasi permasalahan dalam penelitian ini berfokus pada generalitas dari suatu algoritma dalam menyelesaikan permasalahan penjadwalan. Generalitas dalam permasalahan penjadwalan mencakup kemampuan algoritma untuk diterapkan pada berbagai dataset, *benchmark*, dan permasalahan lintas domain dengan menghasilkan performa yang konsisten. Untuk memahami permasalahan ini, penelitian diawali dengan kajian literatur yang sistematis, dengan pendekatan sebagai berikut:

- Pengumpulan Literatur: Pencarian literatur dilakukan menggunakan kata kunci umum yaitu “timetabling” dengan rentang tahun dari tahun 2012 hingga 2022. Tujuan dari tahapan ini untuk mendapatkan penelitian-penelitian yang berkembang sejak tahun 2012. Selain itu tahapan ini juga bertujuan untuk menemukan permasalahan dan *benchmark* yang sering digunakan dalam rentang waktu tersebut.
- Seleksi Berdasarkan Penggunaan *Benchmark*: Proses melakukan identifikasi

permasalahan membutuhkan pemahaman secara menyeluruh terkait dengan efektifitas algoritma dalam menyelesaikan permasalahan penjadwalan. Untuk bisa melakukan analisis tersebut, dibutuhkan proses evaluasi dari algoritma yang telah dikembangkan dalam penelitian sebelumnya. Proses evaluasi dapat dilakukan dengan membandingkan hasil dari setiap algoritma yang menggunakan *benchmark* yang sama. Untuk melaksanakan proses evaluasi ini, literatur yang telah dikumpulkan diseleksi dengan fokus untuk mencari literatur yang menguji coba algoritma yang dikembangkan pada *benchmark* yang umum digunakan oleh para peneliti.

- Pemeringkatan pada setiap Benchmark: Untuk mengidentifikasi algoritma yang memiliki performa terbaik, pada setiap *benchmark* dilakukan pemeringkatan berdasarkan hasil solusi terbaik yang dilaporkan pada setiap penelitian. Pendekatan ini bertujuan untuk memberikan gambaran bagaimana performa dari setiap metode, pendekatan, dan algoritma yang dikembangkan. Selain itu hasil ini dapat menunjukkan pendekatan dan algoritma seperti apa yang memiliki potensi untuk dikembangkan kedepannya.
- Perbandingan Hasil Pemeringkatan Antar-*Benchmark*: Perbandingan hasil perangkingan juga dilakukan pada penelitian yang mengevaluasi algoritmanya terhadap dua *benchmark* atau lebih. Hal ini bertujuan untuk mengevaluasi konsistensi dari setiap penelitian dalam aspek tingkatan generalitas. Dari hasil ini dapat menunjukkan bagaimana perkembangan penelitian dalam mengembangkan algoritma generik dan kesenjangan apa saja yang masih ada dalam ranah penelitian penjadwalan lintas domain.

Tahapan ini menemukan tujuh *benchmark* yang umum digunakan dan 98 penelitian yang relevan. Hasil dari analisis menunjukkan adanya kesenjangan pada tingkat generalitas dalam permasalahan penjadwalan lintas domain. Penelitian terdahulu hanya bisa membuktikan kemampuan algoritmanya pada dua jenis permasalahan dengan *benchmark* lawas, sehingga perlu pembuktian lebih lanjut untuk menunjukkan keberhasilannya dalam tingkat generalitas penjadwalan lintas domian. Hasil analisis lengkap telah dijabarkan pada Bab 2 dalam Subbab 2.2.

3.2 Penentuan Kebutuhan Solusi

Penentuan kebutuhan solusi dalam penelitian ini bertujuan untuk mengidentifikasi pendekatan dan spesifikasi yang diperlukan agar algoritma yang dikembangkan mampu menangani permasalahan penjadwalan lintas domain. Untuk mencapai tujuan tersebut, pengembangan dilakukan menjadi dua algoritma terpisah. Algoritma pertama ditujukan untuk menghasilkan solusi *feasible*. Sementara algoritma kedua ditujukan untuk mengoptimalkan solusi.

3.2.1 Kerangka Algoritma

Untuk memastikan pengembangan algoritma yang efektif, metode *hiper-heuristik* dipilih sebagai kerangka utama, dengan pendekatan *selection-perturbation*. Selain itu pengembangan dilakukan dengan mengambil bentuk dasar dari algoritma ILS. Setiap elemen dalam pendekatan ini dipilih berdasarkan pertimbangan sebagai berikut:

- **Hiper-heuristik:** Metode ini dipilih karena fokusnya pada pengembangan algoritma yang lebih generik dan fleksibel, cocok untuk diterapkan dalam pengembangan algoritma generik.
- **Pendekatan Selection-Perturbation:** Pendekatan ini dipilih karena terbukti lebih efektif dibandingkan pendekatan lain dalam kerangka hiper-heuristik, terutama dalam menghasilkan solusi yang berkualitas dalam beberapa jenis permasalahan.
- **Algoritma ILS:** Algoritma ILS digunakan sebagai algoritma dasar karena karakteristiknya yang sederhana dan tidak membutuhkan pengaturan parameter, sehingga mudah diadaptasi untuk berbagai jenis permasalahan. Selain itu, ILS adalah algoritma berbasis solusi tunggal yang mengombinasikan tahapan eksplorasi dan eksploitasi secara berulang. Pendekatan ini telah terbukti berhasil dalam beberapa penelitian sebelumnya.

Pada algoritma optimasi, algoritma *Threshold Acceptance* diterapkan dalam tahap pencarian lokal untuk meningkatkan kualitas solusi secara bertahap. Algoritma ini memungkinkan untuk menerima solusi yang kurang optimal dalam beberapa iterasi. Strategi ini diharapkan mampu meningkatkan kualitas solusi secara keseluruhan dan menghindari terjebak dalam kondisi *local optima*. Rasionalitas di balik pemilihan metode

hiper-heuristik, pendekatan *selection-perturbation* dan ILS dijelaskan lebih mendetail pada Bab 4.

3.2.2 Penentuan Kebutuhan

Untuk memastikan kedua algoritma mampu memenuhi tujuan dari penelitian, beberapa spesifikasi kebutuhan telah diidentifikasi sebagai berikut:

- **Algoritma untuk Menghasilkan Solusi *Feasible*:**
 - **Pengelolaan *Hard Constraint*:** Algoritma mampu menghasilkan solusi yang memenuhi seluruh *hard constraint*.
 - **Adaptabilitas dan Konsistensi:** Algoritma mampu diterapkan pada berbagai jenis permasalahan penjadwalan dan dapat menunjukkan hasil yang konsisten.
 - **Minimasi Pengaturan Parameter:** Algoritma didesain dengan mengurangi ketergantungan pada proses pengaturan parameter.
- **Algoritma Optimasi:**
 - **Pengoptimalan *Soft Constraint*:** Algoritma mampu meminimalkan pelanggaran *soft constraint* untuk meningkatkan kualitas solusi, namun tetap mempertahankan kondisi solusi yang *feasible*.
 - **Adaptabilitas dan Konsistensi:** Algoritma mampu menghasilkan solusi berkualitas baik di berbagai domain penjadwalan.
 - **Minimasi Pengaturan Parameter:** Algoritma optimasi didesain dengan mengurangi ketergantungan pada proses pengaturan parameter.

Dengan persyaratan ini, diharapkan bahwa kedua algoritma dapat memberikan solusi yang adaptif dan efektif pada berbagai jenis permasalahan tanpa memerlukan pengaturan parameter yang kompleks.

3.3 Perancangan dan Pengembangan Algoritma

Bagian ini menguraikan pendekatan yang diambil dalam pengembangan dua algoritma, yaitu algoritma untuk menghasilkan solusi *feasible* dan algoritma untuk mengoptimasi solusi. Proses pengembangan dirancang secara iteratif, dengan setiap

algoritma disempurnakan melalui beberapa tahap yang saling berkesinambungan untuk memastikan bahwa algoritma memenuhi kebutuhan yang telah diidentifikasi pada Subbab 3.2.2.

3.3.1 Proses Pengembangan

Pengembangan algoritma dilakukan melalui tiga tahap utama yang dirancang untuk mengoptimalkan performa di berbagai domain permasalahan penjadwalan. Ketiga tahapan tersebut dijabarkan sebagai berikut:

- **Studi Algoritma pada Penelitian Sebelumnya:** Langkah awal dalam pengembangan algoritma adalah mempelajari algoritma-algoritma yang terbukti memiliki performa yang baik dari penelitian terdahulu. Hal ini dilakukan untuk memahami pendekatan dan teknik yang terbukti efektif.
- **Desain, Implementasi, dan Pengujian Awal:** Berdasarkan hasil studi tersebut, konsep desain algoritma dikembangkan. Selanjutnya konsep tersebut diimplementasikan menggunakan bahasa Java. Pengujian awal dilakukan pada *benchmark* yang berbeda untuk kedua algoritma:
 - **Algoritma untuk Menghasilkan Solusi Feasible:** Pengujian awal dilakukan pada *benchmark International Timetabling Competition (ITC) 2021*. *Benchmark* ini digunakan sebagai pengujian awal karena memiliki kompleksitas tinggi untuk menghasilkan solusi *feasible* namun waktu eksekusinya relatif cepat.
 - **Algoritma Optimasi:** Pengujian awal dilakukan pada *benchmark* Toronto. Hal ini disebabkan *benchmark* Toronto memiliki batasan yang minimal sehingga dapat mengevaluasi kemampuan algoritma dengan lebih baik. Selain itu waktu eksekusi untuk *benchmark* ini relatif cepat.

Untuk setiap dataset dalam pengujian awal, waktu eksekusi dibatasi hingga 10 jam. Jika algoritma tidak mencapai hasil yang memadai, proses pengembangan akan dimulai ulang dari tahapan awal.

- **Evaluasi Iteratif:** Jika algoritma menunjukkan hasil yang memuaskan pada pengujian awal, algoritma diterapkan pada *benchmark* lainnya untuk memastikan

generalitas dan adaptabilitas. Jika hasil yang diperoleh konsisten baik, proses pengembangan dianggap selesai. Namun, jika hasil tidak memadai, evaluasi dilakukan untuk menentukan area yang perlu perbaikan dan pengembangan dilakukan kembali berdasarkan hasil evaluasi tersebut.

3.3.2 Alat dan Teknik Pengembangan

Algoritma dikembangkan menggunakan bahasa pemrograman Java. Bahasa pemrograman Java memiliki kemampuannya dalam penanganan data dan stabilitas untuk proses komputasi besar. Hal ini yang menjadi alasan pemilihan bahasa pemrograman Java. Selain itu, beberapa pustaka dan framework pendukung dalam Java yang relevan digunakan untuk implementasi algoritma.

3.4 Demonstrasi Algoritma

Bagian ini menjelaskan pendekatan yang digunakan untuk mendemonstrasikan algoritma yang telah dikembangkan. Demonstrasi berfokus pada pengujian kemampuan algoritma dalam menangani permasalahan penjadwalan lintas domain, menggunakan berbagai *benchmark*, dan studi kasus yang memiliki kompleksitas dan karakteristik permasalahan yang bervariasi.

3.4.1 *Benchmark* dan Studi Kasus yang Digunakan untuk Demonstrasi

Untuk menguji generalitas algoritma dalam menyelesaikan berbagai jenis permasalahan penjadwalan, beberapa *benchmark*, dan satu studi kasus digunakan dalam proses implementasi algoritma yang terdiri dari:

- **Benchmark Toronto:** *Benchmark* ini merupakan *benchmark* klasik dalam penjadwalan ujian, yang bersifat sederhana, dan banyak digunakan dalam penelitian terdahulu. *Benchmark* Toronto dipilih karena representasinya terhadap permasalahan penjadwalan ujian yang umum, sehingga menjadi acuan penting untuk mengukur performa algoritma pada permasalahan penjadwalan ujian sederhana.
- **Benchmark ITC 2019:** *Benchmark* ini merupakan *benchmark* terbaru untuk permasalahan penjadwalan mata kuliah. *Benchmark* ini menggunakan dataset dari permasalahan nyata dengan tingkat kompleksitas yang lebih tinggi dan ukuran

permasalahan yang lebih besar dibandingkan *benchmark* lainnya. *Benchmark* ITC 2019 dipilih untuk mewakili permasalahan penjadwalan mata kuliah yang nyata, rumit, dan memiliki ukuran permasalahan yang besar. Hal ini penting untuk menunjukkan variasi uji coba dalam menunjukkan tingkat generalitas dari algoritma yang dikembangkan.

- **Benchmark ITC 2021:** *Benchmark* ini merupakan *benchmark* terbaru dalam permasalahan penjadwalan kompetisi olahraga. *Benchmark* ini memiliki permasalahan yang berbeda secara signifikan dari *benchmark* Toronto dan ITC 2019. Pemilihan menggunakan *benchmark* ITC 2021 untuk menunjukkan variasi jenis permasalahan yang diuji coba.
- **Permasalahan pada Departemen Sistem Informasi:** Selain menggunakan *benchmark*, algoritma diuji terhadap permasalahan penjadwalan nyata dalam Departemen Sistem Informasi ITS. Terdapat dua permasalahan yang terdiri dari:
 - *Penjadwalan Ujian Praktikum:* Permasalahan ini mengangkat bentuk penjadwalan baru yang mengharuskan setiap mahasiswa diawasi oleh satu orang dalam proses ujian.
 - *Penjadwalan Sesi Praktikum:* Permasalahan ini merupakan permasalahan untuk menjadwalkan sesi praktikum antara mahasiswa dan asisten pengajar.

Kedua permasalahan ini mengangkat bentuk permasalahan dengan ketersediaan slot waktu yang sangat minim untuk menjadwalkan ujian dan juga sesi praktikum. Penggunaan dua permasalahan ini pada pengujian algoritma untuk menambah jenis permasalahan yang mampu diselesaikan oleh algoritma yang dikembangkan dan juga untuk mengilustrasikan bagaimana algoritma diterapkan pada permasalahan yang ada dalam lingkungan Departemen Sistem Informasi ITS.

3.4.2 Lingkungan Demonstrasi

Pada penelitian ini kedua algoritma diimplementasikan menggunakan bahasa pemrograman Java dan dijalankan melalui lingkungan pengembangan IntelliJ IDEA. Uji coba dilakukan pada perangkat komputer pribadi dengan spesifikasi prosesor AMD 3.8 GHz dan RAM 32 GB, menggunakan sistem operasi Windows.

3.4.3 Proses Demonstrasi untuk Mencari Solusi Feasible

Pada proses demonstrasi algoritma dalam menghasilkan solusi *feasible*, pengujian dilakukan dengan ketentuan sebagai berikut:

- **Durasi Pengujian:** Algoritma dijalankan selama maksimal 120 jam per dataset untuk mencapai solusi *feasible*. Jika pengujian telah melewati durasi 120 jam, maka dalam pengujian tersebut algoritma dianggap gagal menghasilkan solusi *feasible*.
- **Jumlah Eksperimen:** Pengujian dilakukan sebanyak 10 kali untuk setiap dataset.

3.4.4 Proses Demonstrasi untuk Optimasi Solusi

Pada proses demonstrasi algoritma dalam tahapan mengoptimasi solusi, pengujian dilakukan dengan ketentuan sebagai berikut:

- **Pemilihan Solusi Awal:** Algoritma untuk proses optimasi dijalankan menggunakan solusi yang dipilih secara acak dari solusi *feasible* yang dihasilkan pada tahapan sebelumnya. Hal ini dilakukan untuk menguji algoritma dalam kondisi solusi awal yang bervariasi.
- **Pengujian Awal:** Pengujian awal dilakukan untuk melihat konsistensi algoritma. Setiap dataset dioptimasi menggunakan durasi pengujian yang berbeda. Waktu optimasi disesuaikan dengan ukuran dataset. Secara detail durasi pengujian dijabarkan sebagai berikut:
 - *Benchmark* Toronto : 10 jam
 - *Benchmark* ITC 2019 : 20 jam
 - *Benchmark* ITC 2021 : 10 jam
 - Penjadwalan Sesi Praktikum dan Ujian : 10 menit
- **Pengujian Lanjutan:** Solusi terbaik dari tahapan pengujian awal dioptimasi kembali. Pengujian ini akan dihentikan ketika tidak ada solusi terbaik baru yang berhasil ditemukan dalam waktu 10 jam terakhir. Pengujian lanjutan hanya dilakukan pada pengujian terhadap *benchmark*.

3.5 Evaluasi Algoritma

Bagian ini menjelaskan pendekatan evaluasi untuk mengukur performa dari kedua algoritma yang telah dikembangkan. Subbab 3.5.1 menjelaskan metrik yang digunakan untuk proses evaluasi. Sedangkan Subbab 3.5.2 menjelaskan metode evaluasi yang digunakan.

3.5.1 Metrik Evaluasi

Untuk mengevaluasi hasil dari tahapan menghasilkan solusi *feasible*, digunakan dua metrik utama, yaitu:

- **Persentase Keberhasilan:** Metrik ini dihitung dengan membagi jumlah solusi *feasible* yang berhasil dihasilkan dengan jumlah total percobaan (sepuluh kali uji coba). Hasil metrik ini digunakan untuk mengevaluasi efektivitas algoritma yang dikembangkan.
- **Waktu Eksekusi:** Metrik ini mengukur durasi waktu tercepat, terlama, dan rata-rata yang diperlukan untuk menghasilkan solusi *feasible*. Metrik ini memberikan gambaran mengenai efisiensi algoritma dalam konteks aplikasinya secara praktis.

Sementara itu, tahap optimasi dievaluasi menggunakan lima metrik yang didasarkan pada nilai penalti yang didapatkan dari terhadap pelanggaran *soft constraint* pada solusi yang dihasilkan. Kelima metriks yang digunakan dijabarkan sebagai berikut:

- **Solusi Rata-Rata:** Nilai penalti rata-rata yang diperoleh dari 10 kali uji coba. Semakin kecil nilai rata-rata, semakin baik kualitas solusi secara umum.
- **Solusi Terbaik:** Nilai penalti terendah (minimum) yang berhasil ditemukan selama 10 kali uji coba. Angka ini menunjukkan solusi dengan jumlah pelanggaran *soft constraint* paling sedikit.
- **Solusi Terburuk:** Nilai penalti tertinggi (maksimum) yang muncul dari 10 kali uji coba. Nilai ini memberikan gambaran tentang kemungkinan performa terendah algoritma.
- **Koefisien Variasi:** Mengukur konsistensi solusi dari 10 kali uji coba. Koefisien variasi yang rendah menandakan bahwa algoritma cenderung menghasilkan solusi

dengan penalti yang relatif seragam, sedangkan nilai yang tinggi menunjukkan variabilitas performa algoritma.

- **Waktu Eksekusi:** Metrik ini mengukur durasi proses pada tahapan optimasi lanjutan. Metriks ini digunakan untuk memberikan gambaran penerapan algoritma dalam aplikasinya secara praktis.

3.5.2 Metode Evaluasi

Evaluasi performa algoritma dilakukan dengan memperhatikan hasil pada setiap *benchmark* dan juga hasil secara keseluruhan untuk menilai generalitas algoritma yang dikembangkan. Secara detail, metode evaluasi dijabarkan sebagai berikut:

- **Perbandingan Hasil dengan Penelitian dan Algoritma Lainnya:**

- Untuk uji coba menggunakan *benchmark*, hasil penelitian ini dibandingkan dengan studi terdahulu. Proses perbandingan dilakukan dengan melakukan pemeringkatan pada setiap dataset. Hasil dari pemeringkatan pada setiap dataset digunakan untuk menghasilkan peringkat rata-rata yang digunakan sebagai hasil akhir untuk menilai performa dari algoritma yang dikembangkan.

Dalam proses pemeringkatan, jika ditemukan peringkat yang sama antara dua atau lebih studi maka pemeringkatan akan mengambil nilai rata-rata peringkat. Sebagai contoh jika terdapat tiga studi dengan peringkat 4, maka pemeringkatan akan dihitung sebagai berikut $\frac{(4+5+6)}{3} = 5$. Hal ini ditujukan untuk memberikan hasil pemeringkatan yang lebih adil.

- Untuk uji coba studi kasus departemen Sistem Informasi ITS, algoritma yang dikembangkan dibandingkan dengan dua algoritma dasar yaitu *Hill-Climbing* (HC) dan *Late Acceptance Hill Climbing* (LAHC). HC dipilih karena algoritma ini tidak memerlukan penyesuaian parameter dan sering digunakan sebagai *baseline* dalam evaluasi algoritma. Sedangkan LAHC dipilih karena memiliki satu nilai pengaturan parameter dan algoritma ini menunjukkan hasil yang baik pada beberapa penelitian yang berkaitan dengan optimasi (Burke and Bykov, 2017).

- **Analisis Generalitas:** Untuk mengevaluasi aspek generalitas, performa algoritma

dianalisis berdasarkan konsistensi hasil yang baik (memiliki peringkat di atas rata-rata) pada berbagai *benchmark*. Konsistensi ini menunjukkan bahwa algoritma dapat mempertahankan performa optimal di berbagai jenis permasalahan tanpa melakukan perubahan yang signifikan.

BAB 4

DESAIN DAN PENGEMBANGAN ALGORITMA

Bab ini menguraikan desain dan pengembangan dari dua algoritma utama untuk menyelesaikan permasalahan penjadwalan. Algoritma pertama merupakan algoritma yang ditujukan untuk mampu menghasilkan solusi *feasible* dengan memenuhi semua *hard constraint* dari suatu permasalahan penjadwalan. Sementara itu, algoritma kedua merupakan algoritma yang bertujuan untuk mengoptimasi solusi *feasible* yang telah dihasilkan oleh algoritma pertama. Proses optimasi dijalankan dengan mengurangi jumlah pelanggaran *soft constraint* dalam batas waktu yang ditentukan dan tetap menjaga solusi dalam keadaan *feasible*.

Kedua algoritma dikembangkan berdasarkan pendekatan hiper-heuristik *selection-perturbation*, dengan menggunakan bentuk dasar dari algoritma Iterated Local Search (ILS). Pada algoritma untuk mencari solusi *feasible*, pengembangan dilakukan dengan mendesain algoritma untuk dapat terus menjalankan proses pencarian tanpa adanya penolakan terhadap solusi baru. Berdasarkan ide tersebut algoritma ini dinamakan sebagai algoritma Progressive Acceptance Iterated Local Search (PA-ILS). Sementara itu pada algoritma untuk proses optimasi, algoritma didesain dengan menggabungkan strategi nilai ambang batas (*threshold*) dengan algoritma ILS. Kelebihan utama pada algoritma ini ada pada aspek generalitasnya dengan nilai *threshold* yang mampu beradaptasi dengan permasalahan yang ada, sehingga tidak membutuhkan pengaturan nilai parameter. Berdasarkan karakteristik tersebut, algoritma ini dinamakan Adaptive Threshold-Iterated Local Search (AT-ILS).

Untuk penjelasan lebih detail dari kedua algoritma tersebut, Subbab 4.1, 4.2, dan 4.3 menjabarkan rasionalitas dan desain pengembangan algoritma. Selain itu, pembahasan *Low Level Heuristic* (LLH) yang digunakan oleh kedua algortima dibahas pada Subbab

4.4. Terakhir, pembahasan proses pengaturan nilai parameter untuk algoritma PA-ILS dijabarkan dalam Subbab 4.5.

4.1 Rasionalitas Desain Algoritma

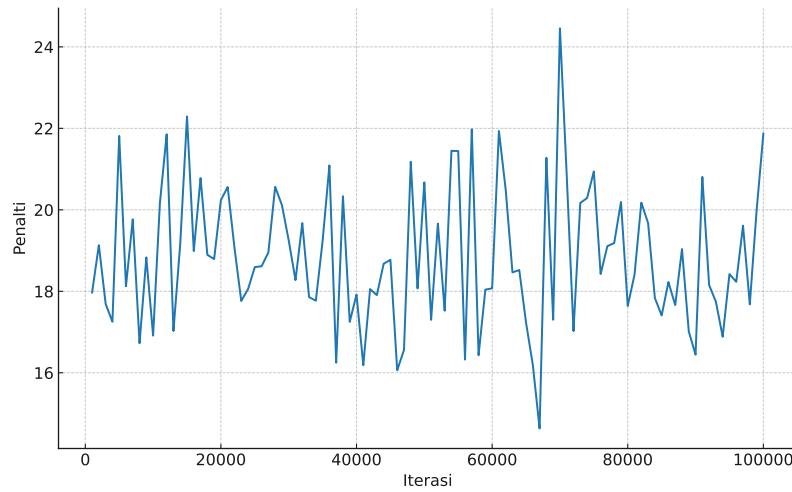
Bagian ini menjelaskan rasionalitas dari desain algoritma yang dikembangkan. Subbab 4.1.1 menjelaskan pentingnya keseimbangan eksplorasi dan eksploitasi yang menjadi ide dasar dari pengembangan algoritma ini. Selanjutnya, Subbab 4.1.2 dan 4.1.3 menjelaskan alasan pemilihan metode dan pendekatan hiper-heuristik *Selection-Perturbation* serta pemilihan algoritma ILS yang digunakan sebagai bentuk dasar dalam pengembangan algoritma. Terakhir, Subbab 4.1.4 menjelaskan pendekatan dalam mendesain algoritma yang dikembangkan pada penelitian ini.

4.1.1 Keseimbangan antara Eksplorasi dan Eksploitasi

Dalam mengembangkan algoritma untuk permasalahan optimasi, dibutuhkan keseimbangan antara proses eksplorasi dan eksploitasi dalam proses pencarian solusi. Keseimbangan ini tidak berarti bahwa jumlah proses eksplorasi dan eksploitasi harus sama, tetapi kedua proses tersebut mampu menyesuaikan dengan karakteristik permasalahan.

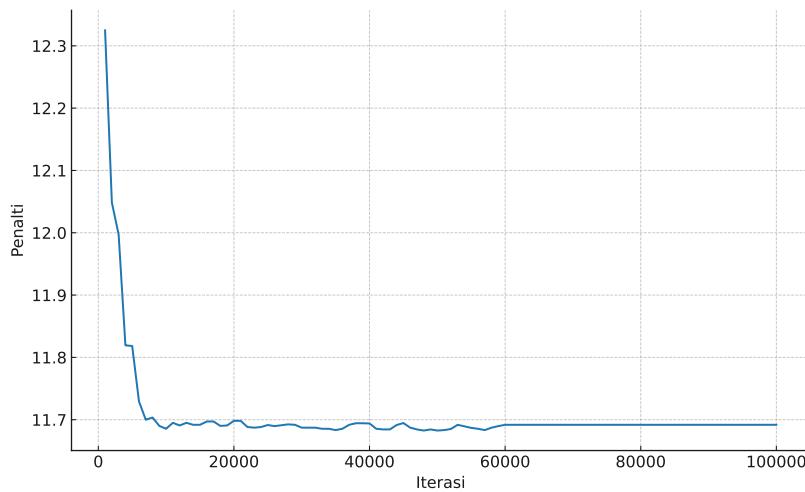
Algoritma yang terlalu banyak melakukan eksplorasi dapat menghambat konvergensi solusi menuju hasil yang lebih optimal. Hasilnya solusi akan terjebak dalam rentang tertentu tanpa peningkatan signifikan. Sebagai ilustrasi, Gambar 4.1.1 menunjukkan proses pencarian solusi pada algoritma Threshold Acceptance dengan nilai ambang batas yang terlalu besar, yang menyebabkan proses eksplorasi menjadi berlebihan. Akibatnya, pencarian solusi menjadi stagnan dalam suatu rentang tertentu dan sulit diarahkan menuju hasil yang lebih baik. Hal ini terjadi karena saat algoritma mencoba menurunkan nilai penalti, proses eksplorasi yang berlebihan sering kali menyebabkan nilai penalti kembali meningkat, sehingga pencarian solusi hanya berfluktuasi dalam rentang tertentu tanpa mencapai perbaikan signifikan.

Sebaliknya, algoritma yang terlalu banyak melakukan eksploitasi cenderung terjebak pada kondisi yang disebut *local optima*. Kondisi ini terjadi ketika algoritma tidak mengizinkan penerimaan solusi yang sedikit lebih buruk untuk eksplorasi ruang pencarian. Namun, algoritma tidak mampu menemukan solusi yang lebih baik. Hal ini menyebabkan



Gambar 4.1.1: Ilustrasi Proses Pencarian dengan Tahapan Eksplorasi yang Berlebihan

solusi berhenti pada titik tertentu hingga akhir proses pencarian. Gambar 4.1.2 menunjukkan contoh algoritma Threshold Acceptance dengan nilai ambang batas yang terlalu kecil, yang menyebabkan minimnya proses eksplorasi. Pada ilustrasi ini, solusi tidak bergerak sejak iterasi sekitar 60000. Algoritma tidak dapat menemukan solusi yang lebih baik, namun algoritma juga tidak mengizinkan penerimaan solusi yang sedikit lebih buruk untuk keluar dari titik tersebut. Hasilnya proses pencarian terjebak dalam kondisi *local optima*.



Gambar 4.1.2: Ilustrasi Proses Pencarian dengan Tahapan Eksplorasi yang Berlebihan

4.1.2 Penggunaan Pendekatan Hiper-Heuristik *Selection-Perturbation*

Berdasarkan hasil pemeringkatan penelitian terdahulu pada Subbab 2.2, pendekatan metaheuristik (S) menunjukkan hasil paling baik dan konsisten. Dalam permasalahan penjadwalan, pendekatan metaheuristik (S) dimulai dengan memodifikasi satu solusi menggunakan salah satu *neighborhood operator* yang umumnya dipilih secara acak. Solusi baru yang dihasilkan kemudian dievaluasi untuk menentukan apakah solusi tersebut akan diterima. Jika diterima, solusi ini disimpan dan digunakan dalam iterasi berikutnya. Jika tidak, solusi tersebut dibuang dan solusi sebelumnya tetap dipertahankan.

Secara umum, pendekatan metaheuristik (S) dalam menyelesaikan permasalahan penjadwalan tidak berbeda secara signifikan dari pendekatan hiper-heuristik (S-P). Pendekatan hiper-heuristik (S-P) memiliki tiga komponen utama, yaitu:

- kumpulan LLH: kumpulan dari LLH yang dapat digunakan untuk memodifikasi solusi.
- LLH *selection*: strategi dalam memilih LLH pada setiap iterasi.
- *move acceptance*: strategi untuk menentukan penerimaan solusi.

Ketiga komponen pada hiper-heuristik (S-P) pada dasarnya menyerupai komponen pada pendekatan metaheuristik (S). Pada pendekatan hiper-heuristik (S-P), komponen *move acceptance* umumnya menggunakan algoritma yang sama dengan algoritma pada pendekatan metaheuristik (S) seperti Simulated Annealing, ILS, dan Great Deluge (Demeester et al., 2012; Soria-Alcaraz et al., 2016). Sementara itu pada komponen LLH *selection*, pemilihan LLH dapat dilakukan dengan beberapa strategi tertentu atau bisa dilakukan secara acak (Pillay, 2016b). Penggunaan strategi pemilihan LLH secara acak merupakan strategi yang serupa dengan pendekatan metaheuristik (S), yang juga cenderung memilih *neighborhood operator* secara acak. Istilah LLH dan *neighborhood operator* dalam algoritma penjadwalan umumnya merujuk pada konsep yang sama yaitu metode untuk melakukan perubahan solusi. Hal ini menunjukkan pendekatan metaheuristik (S) dan hiper-heuristik (S-P) pada dasarnya menjalankan proses serupa dalam konteks penjadwalan.

Namun, secara konseptual terdapat perbedaan antara pendekatan metaheuristik (S) dan hiper-heuristik (S-P). Pada pendekatan metaheuristik (S), pengembangan desain

algoritma dilakukan secara menyeluruh. Sementara pada pendekatan hiper-heuristik (S-P), desain algoritma dibagi menjadi tiga komponen utama seperti yang telah dijabarkan sebelumnya. Struktur desain algoritma yang dibagi menjadi tiga komponen memberikan keunggulan berupa kemudahan pemahaman serta fleksibilitas yang lebih tinggi dalam pengembangan lebih lanjut. Berdasarkan pertimbangan tersebut, penelitian ini memilih untuk mengembangkan desain algoritma menggunakan pendekatan hiper-heuristik (S-P).

4.1.3 Pemilihan Algoritma ILS

Algoritma ILS menjalankan tiga tahapan utama secara berulang, yaitu *perturbation*, *local search*, dan *move acceptance*. Algoritma ILS sebenarnya lebih tepat disebut sebagai sebuah kerangka kerja, karena algoritma ini hanya menyediakan struktur dasar berupa tiga tahapan tersebut tanpa menjelaskan secara rinci bagaimana masing-masing tahapan, seperti *perturbation* dan *local search*, harus dijalankan atau strategi apa yang diterapkan dalam *move acceptance*. Oleh karena itu, algoritma ILS sangat cocok dijadikan dasar dalam pengembangan algoritma yang dikembangkan dalam penelitian ini.

Hasil penelitian sebelumnya menunjukkan bahwa pengembangan penelitian menggunakan algoritma ILS menunjukkan hasil yang menjanjikan. Dalam aspek pencarian solusi *feasible*, studi oleh Song et al. (2018) dan Goh et al. (2020) menunjukkan bahwa algoritma ini dapat menghasilkan solusi terbaik dibandingkan dengan penelitian lainnya. Sementara itu, pada aspek optimasi, penelitian oleh Soria-Alcaraz et al. (2016) menunjukkan hasil yang sangat menjanjikan pada *benchmark International Timetabling Competition (ITC) 2007 Track 3* dengan memperoleh peringkat terbaik dari 7 penelitian. Temuan-temuan ini mendukung pemilihan ILS sebagai dasar pengembangan algoritma dalam penelitian ini, yang memungkinkan untuk menghasilkan algoritma yang efektif dalam menghasilkan solusi *feasible* dan juga mengoptimasi solusi.

4.1.4 Pendekatan Desain Algoritma PA-ILS dan AT-ILS

Dalam permasalahan menemukan solusi *feasible*, algoritma memiliki tujuan untuk menghasilkan solusi dengan tidak ada satupun pelanggaran terhadap *hard constraint*. Tujuan dalam permasalahan mencari solusi *feasible* merupakan tujuan yang jelas dan dapat diukur. Sementara itu, pada proses optimasi, tujuan utamanya adalah meminimalkan jumlah pelanggaran *soft constraint*. Namun, nilai akhir yang menjadi tujuan dalam proses

optimasi tidak diketahui akibat dari permasalahan penjadwalan yang tergolong sebagai permasalahan NP-Hard. Hal ini menyebabkan tidak adanya nilai yang jelas untuk digunakan sebagai target dari proses optimasi.

Pada proses pencarian solusi *feasible*, solusi yang tidak *feasible* umumnya tidak digunakan dan tidak dipertimbangkan lebih lanjut. Sebagai contoh, jika terdapat dua solusi yang tidak *feasible*, solusi pertama gagal menjadwalkan tiga mahasiswa dan solusi kedua gagal menjadwalkan enam mahasiswa, maka umumnya kedua solusi tersebut akan diabaikan dan kembali mencoba untuk menemukan solusi yang *feasible*. Sebaliknya, dalam proses optimasi, semua solusi yang dihasilkan tetap memungkinkan untuk dipertimbangkan dan digunakan karena solusi tersebut *feasible*.

Perbedaan ini mempengaruhi pendekatan yang digunakan dalam mengembangkan algoritma PA-ILS dan AT-ILS. Algoritma PA-ILS didesain untuk mampu mencapai sasaran yang konkret yaitu menemukan solusi dengan tidak ada satupun pelanggaran terhadap *hard constraint*. Sementara algoritma AT-ILS didesain untuk lebih fleksibel dengan mampu mengurangi pelanggaran terhadap *soft constraint* tanpa memiliki nilai target yang pasti.

4.1.4.1 Pendekatan Desain Algoritma PA-ILS

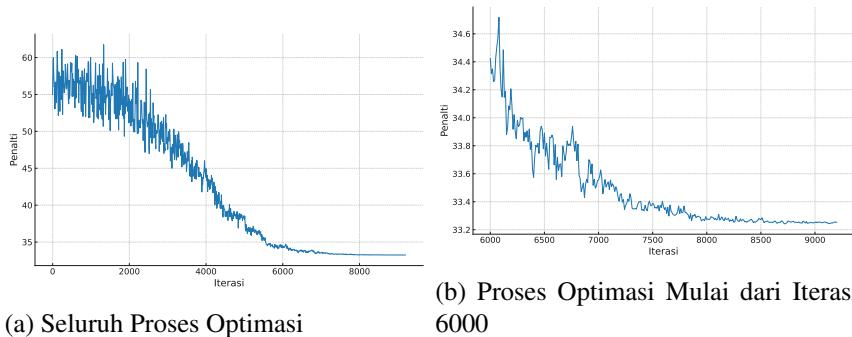
Dalam mengembangkan algoritma PA-ILS untuk mencari solusi *feasible*, algoritma didesain dengan strategi pencarian yang tidak terpaku pada satu titik solusi yang baik. Hal ini diakibatkan solusi yang tidak *feasible* tidak dapat digunakan walaupun hanya beberapa kegiatan saja yang memiliki konflik atau belum terjadwalkan. Berdasarkan hal tersebut, algoritma PA-ILS didesain untuk melakukan eksplorasi secara kontinu dalam proses pencarian solusi dan tidak terpaku pada satu solusi yang baik namun tidak *feasible*.

Untuk menerapkan strategi eksplorasi yang kontinu, algoritma dikembangkan dengan hanya menggunakan tahapan *perturbation* dan *local search*, tanpa menggunakan tahapan *move acceptance*. Dengan menghilangkan tahapan *move acceptance*, setiap solusi yang dihasilkan dari *perturbation* dan *local search* selalu diterima. Sementara itu, untuk mengontrol agar perubahan solusi tidak terlalu menjauhi dari target, perubahan solusi pada tahap *perturbation* tidak dilakukan secara acak sepenuhnya. Strategi pada tahap ini memungkinkan menjalankan dua strategi yang berbeda. Strategi pertama memilih slot

waktu secara acak seperti tahapan *perturbation* pada umumnya. Strategi kedua memilih slot waktu dengan konflik paling sedikit. Pemilihan ini didasarkan pada nilai probabilitas yang akan dijelaskan lebih lanjut pada bagian 4.2 dan 4.2.3. Dengan desain strategi ini, tahapan *perturbation* lebih sering menghasilkan solusi yang mendekati kondisi *feasible*, meskipun sesekali solusi dapat bergerak agak jauh untuk mencegah pencarian terjebak di tempat yang sama.

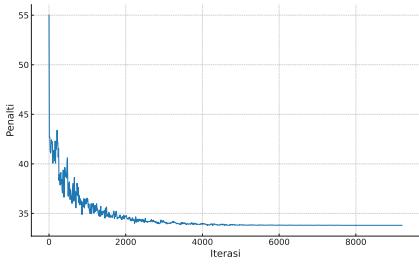
4.1.4.2 Pendekatan Desain Algoritma AT-ILS

Penelitian sebelumnya menunjukkan bahwa algoritma Simulated Annealing merupakan algoritma yang menunjukkan hasil yang paling menjanjikan pada berbagai jenis *benchmark*. Untuk mengetahui mengapa algoritma Simulated Annealing mampu menghasilkan hasil yang baik, penelitian ini melakukan pengujian awal untuk memahami kelebihan dari algoritma Simulated Annealing. Pengujian dilakukan dengan melihat bagaimana proses pencarian pada algoritma Simulated Annealing dan juga pada algoritma lainnya yaitu Late Acceptance Hill Climbing (LAHC) dan Threshold Acceptance. Gambar 4.1.3 menunjukkan ilustrasi proses optimasi dari algoritma Simulated Annealing dan Gambar 4.1.4 menunjukkan ilustrasi proses optimasi pada algoritma LAHC dan Threshold Acceptance.

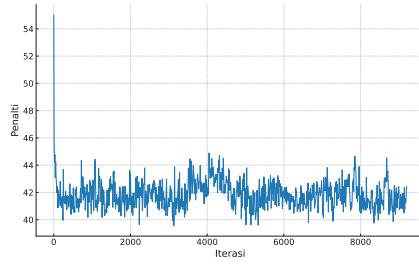


Gambar 4.1.3: Ilustrasi dari Proses Optimasi Pada Algoritma Simulated Annealing

Dari ilustrasi ini, algoritma Simulated Annealing memiliki proses eksplorasi dan eksploitasi yang lebih berimbang. Dalam setiap rentang iterasi, proses eksplorasi dan eksploitasi dilakukan namun solusi akan ditekan untuk menurun secara perlahan. Pada Gambar 4.1.3b menunjukkan bahwa dari iterasi ke-6000 hingga akhir proses pencarian, proses eksplorasi dan eksploitasi tetap dilakukan namun solusi tetap terus ditekan turun secara perlahan.



(a) LAHC



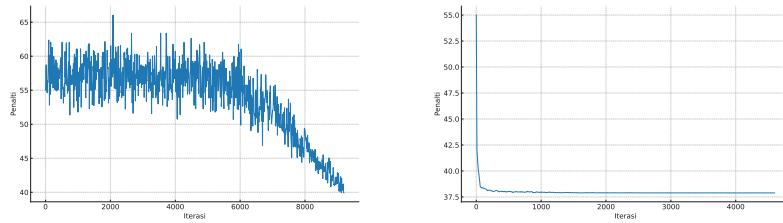
(b) *Threshold Acceptance*

Gambar 4.1.4: Ilustrasi dari Proses Optimasi Pada Algoritma LAHC dan Threshold Acceptance

Berbeda dengan algoritma Simulated Annealing, algoritma LAHC melakukan penekanan penurunan solusi yang lebih besar pada awal iterasi dan terus menekan solusi hingga akhir iterasi. Walaupun pada algoritma LAHC proses eksplorasi dan eksloitasi tetap ada pada setiap rentang iterasi, namun proses ini jelas lebih didominasi oleh proses eksloitasi dengan solusi yang turun sangat cepat dibandingkan algoritma Simulated Annealing. Sementara pada algoritma Threshold Acceptance, penekanan solusi di awal iterasi sangat cepat menunjukkan dominasi proses eksloitasi. Namun pada saat mencapai nilai tertentu, solusi hanya bergerak pada rentang penalti tertentu. Hal ini diakibatkan dari proses eksplorasi yang terlalu banyak sehingga solusi tidak dapat ditekan turun.

Dari perbandingan ini, algoritma Simulated Annealing menunjukkan strategi yang lebih seimbang dalam menjalankan eksplorasi dan eksloitasi. Namun permasalahan pada algoritma Simulated Annealing ada pada dua nilai parameter yaitu *temperature* dan *cooling rate* yang harus dilakukan pengaturan nilai dengan tepat agar bisa memberikan hasil yang maksimal. Ketidaktepatan nilai parameter akan menyebabkan algoritma terlalu banyak melakukan proses eksplorasi yang berakibat solusi terjebak pada rentang tertentu seperti yang diilustrasikan pada Gambar 4.1.5a. Sebaliknya, ketika algoritma menjalankan terlalu banyak menjalankan proses eksloitasi, solusi akan terjebak dalam kondisi *local optima* seperti yang diilustrasikan pada Gambar 4.1.5b.

Berdasarkan hal tersebut, algoritma AT-ILS didesain dengan mengupayakan strategi eksplorasi dan eksloitasi yang seimbang menyerupai algoritma Simulated Annealing. Proses eksplorasi dan eksloitasi didesain agar mampu berjalan bergantian dan secara keseluruhan mampu menekan solusi untuk mengurangi penalti secara perlahan. Algoritma ini didesain dengan tidak membutuhkan satupun pengaturan nilai parameter. Hal ini



(a) Terlalu Banyak Explorasi Solusi

(b) Terlalu Banyak Exploitasi Solusi

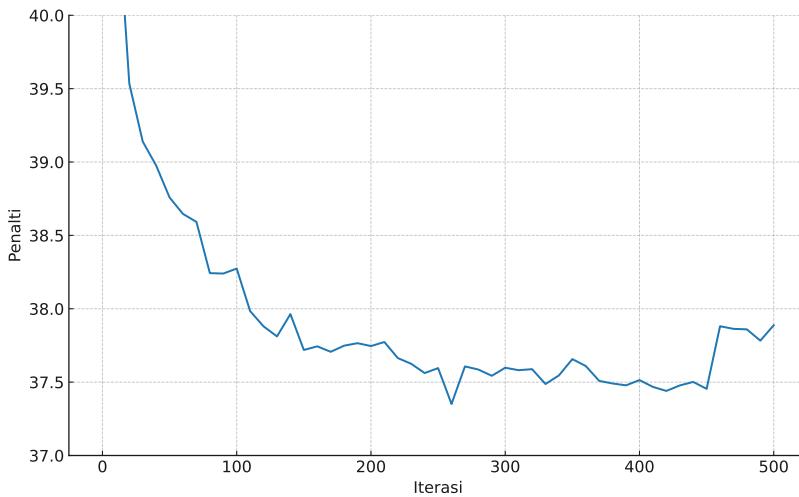
Gambar 4.1.5: Ilustrasi dari Algoritma Simulated Annealing yang Tidak Menggunakan Nilai Parameter yang Tepat

memperbaiki kekurangan pada algoritma Simulated Annealing dan juga meningkatkan tingkat generalitas dari algoritma yang dikembangkan.

Selain itu, dari pengembangan yang sudah dilakukan pada penelitian sebelumnya (Premananda, Muklason and Hutama, 2022; Premananda, Tjahyanto and Muklason, 2022), terdapat kemungkinan pencarian solusi bergerak ke arah yang menjauhi dari solusi terbaik yang berhasil ditemukan. Pencarian solusi juga memungkinkan tidak pernah kembali kearah solusi terbaik yang pernah ditemukan hingga akhir iterasi. Sebagai contoh Gambar 4.1.6 mengilustrasikan bagaimana nilai terbaik didapatkan berkisar pada iterasi ke-250, namun setelah itu proses pencarian tidak berhasil mendekati nilai terbaik yang ditemukan. Bahkan diakhir iterasi, solusi semakin bergerak menjauhi solusi terbaik. Berdasarkan hal tersebut, algoritma didesain untuk dapat kembali pada titik solusi terbaik ditemukan. Hal ini untuk mencegah algoritma bergerak terlalu jauh dari solusi terbaik dan diharapkan algortima bisa menghasilkan solusi terbaik baru.

4.2 Desain Algoritma PA-ILS

Desain algoritma PA-ILS ditunjukkan oleh Algoritma 2. Untuk menghasilkan proses pencarian solusi yang terus berjalan, algoritma didesain hanya memiliki dua tahapan utama, yaitu tahapan *perturbation* dan *local search*. Tahapan *perturbation* menghasilkan solusi baru yang memungkinkan untuk menghasilkan solusi yang lebih buruk. Sebaliknya, tahapan *local search* bertujuan untuk menghasilkan solusi yang lebih baik atau setidaknya setara dengan solusi sebelumnya. Suatu solusi dianggap lebih baik jika dapat menjadwalkan lebih banyak kegiatan tanpa melanggar satupun *hard constraint*. Sementara solusi baru dikategorikan lebih buruk jika jumlah kegiatan yang berhasil dijadwalkan lebih



Gambar 4.1.6: Ilustrasi Solusi yang Menjauh dari Nilai Solusi Terbaik

sedikit dari solusi sebelumnya. Dengan desain ini, algoritma selalu menerima solusi baru yang dihasilkan oleh tahapan *perturbation* dan *local search*.

Algoritma PA-ILS dikembangkan dengan selalu menerima solusi yang dihasilkan oleh tahapan *perturbation* dan *local search*. Untuk memastikan proses pencarian solusi tidak terlalu eksploratif, satu variabel dikembangkan yang bernama `probabilityNonRandomTimeSlot` yang dihitung melalui fungsi `calculateProbability`. Variabel ini bertujuan untuk membatasi eksplorasi pada tahapan *perturbation*. Variabel ini akan mengatur pemilihan slot waktu yang dapat digunakan berdasarkan jumlah konflik yang dihasilkan.

Penjelasan detail mengenai aspek-aspek utama pada algoritma PA-ILS dijabarkan pada subbab-subbab berikutnya. Subbab 4.2.1 membahas perhitungan nilai variabel `probabilityNonRandomTimeSlot`. Subbab 4.2.2 membahas proses penjadwalan kegiatan dan tahapan untuk menghapus konflik yang mungkin terjadi. Selanjutnya, tahapan *perturbation* dan *local search* dijabarkan pada Subbab 4.2.3 dan 4.2.4. Terakhir, pembahasan parameter yang terdapat pada algoritma PA-ILS dijabarkan pada Subbab 4.2.5

4.2.1 Perhitungan Variabel `probabilityNonRandomTimeSlot`

Variabel `probabilityNonRandomTimeSlot` merupakan variabel yang mengatur nilai probabilitas, sehingga nilai variabel ini berada diantara 0 hingga 1. Nilai

Algoritma 2 PA-ILS

```
1: procedure SEARCHFEASIBLESOLUTION( decreasingValue, constantFactor,  
    limitUpdateProbabilityNonRandomTimeSlot, limitStuck, limitShuffle )  
2:   calculateProbability(decreasingValue, constantFactor)  
3:   iteration ← 0  
4:   while not allEventScheduled() do  
5:     perturbationPhase(probabilityNonRandomTimeSlot)  
6:     localSearchPhase(limitStuck, limitShuffle)  
7:     if iteration MOD limitUpdateProbabilityNonRandomTimeSlot = 0 then  
8:       calculateProbability(decreasingValue, constantFactor)  
9:     end if  
10:    iteration ← iteration +1  
11:   end while  
12: end procedure
```

dari variabel dihitung dengan mengurangi nilai 1 dengan `decreasingValue`, seperti pada Persamaan 4.1. Nilai `decreasingValue` diperoleh dengan mengalikan nilai `decreasingValue` saat ini dengan `constantFactor`, sebagaimana diperlihatkan pada Persamaan 4.2. Ketiga variabel—`probabilityNonRandomTimeSlot`, `decreasingValue`, dan `constantFactor`—dibatasi dalam rentang 0 hingga 1.

Kedua persamaan ini dieksekusi ketika metode `calculateProbability` dipanggil. Pertama, Persamaan 4.2 dijalankan untuk menghitung nilai terbaru dari `decreasingValue`. Selanjutnya, Persamaan 4.1 digunakan untuk menghitung `probabilityNonRandomTimeSlot`. Pendekatan ini dirancang agar nilai `probabilityNonRandomTimeSlot` secara bertahap meningkat, sehingga meningkatkan kemungkinan pemilihan slot waktu dengan konflik minimal. Jika nilai `probabilityNonRandomTimeSlot` melebihi 0,99, prosedur pengulangan diaktifkan untuk mengatur ulang variabel `decreasingValue` ke nilai awal.

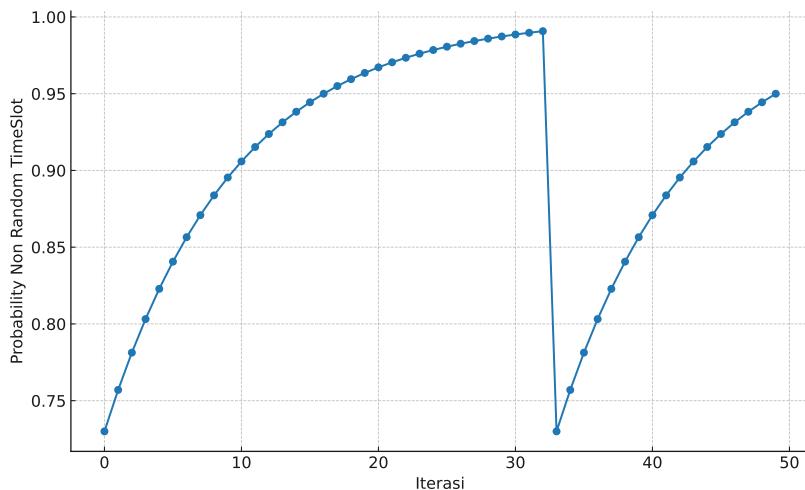
$$probabilityNonRandomTimeSlot = 1 - decreasingValue \quad (4.1)$$

$$decreasingValue = decreasingValue \cdot constantFactor \quad (4.2)$$

Untuk memahami cara kerja dari kedua persamaan, Gambar 4.2.1 mengilustrasikan perubahan nilai variabel `probabilityNonRandomTimeSlot` seiring dengan pertambahan iterasi. Dalam contoh ini, nilai awal `decreasingValue` ditetapkan sebesar 0,3 dan `constantFactor` sebesar 0,9, dengan asumsi bahwa fungsi `calculateProbability` dipanggil pada setiap iterasi.

Pada iterasi ke-0, persamaan 4.2 dijalankan dengan mengalikan `decreasingValue` (0,3) dan `constantFactor` (0,9), sehingga dihasilkan nilai 0,27. Nilai ini kemudian digunakan sebagai `decreasingValue` untuk persamaan 4.1, sekaligus menjadi nilai awal pada iterasi berikutnya. Setelah itu, persamaan 4.1 dieksekusi dengan mengurangi 1 dengan 0,27, menghasilkan nilai `probabilityNonRandomTimeSlot` sebesar 0,73. Proses perhitungan ini berulang pada setiap iterasi dan menyebabkan nilai `probabilityNonRandomTimeSlot` terus meningkat.

Ketika hasil persamaan pertama melebihi 0,99—sebagaimana ditunjukkan pada iterasi ke-32—kedua persamaan dihitung ulang dengan menggunakan nilai awal `decreasingValue` sebesar 0,3. Akibatnya, `probabilityNonRandomTimeSlot` akan memiliki nilai 0,73 sama dengan nilai pada iterasi ke-0.



Gambar 4.2.1: Grafik Perubahan Nilai Variabel `probabilityNonRandomTimeSlot`

4.2.2 Proses Menjadwalkan Kegiatan

Pada tahapan *local search* dan *perturbation*, proses penjadwalan kegiatan dilakukan dengan terlebih dahulu menghapus kegiatan-kegiatan yang berkonflik dengan kegiatan yang akan dijadwalkan akibat adanya pelanggaran terhadap *hard constraint*. Setelah tidak ada konflik yang tersisa, kegiatan tersebut kemudian dijadwalkan. Untuk menggambarkan proses ini, Gambar 4.2.2 menunjukkan contoh penjadwalan kegiatan (dalam ilustrasi ini berupa penjadwalan kompetisi olahraga) dengan menampilkan beberapa tahapan yang terlibat.

Dalam contoh ilustrasi ini, pertandingan antara Tim 1 dan Tim 2 akan dijadwalkan pada slot waktu ke-1. Namun, pada slot waktu ke-1, telah dijadwalkan pertandingan antara Tim 1 dan Tim 3. Kondisi ini menimbulkan pelanggaran terhadap *hard constraint*, yang mensyaratkan bahwa satu tim hanya dapat bermain satu kali dalam satu slot waktu. Oleh karena itu, pertandingan antara Tim 1 dan Tim 3 harus dihapus dari slot waktu tersebut. Gambar 4.2.2a mengilustrasikan kondisi sebelum dan sesudah penyelesaian konflik ini.

Selain itu, penjadwalan pertandingan antara Tim 1 dan Tim 2 juga melanggar batasan *hard constraint* lain. Dalam contoh ini, *hard constraint* membatasi hanya satu tim yang dapat bermain tandang pada slot waktu ke-1, dengan pilihan antara Tim 2 atau Tim 5. Akibatnya, untuk mengatasi konflik ini, diperlukan untuk menghapus pertandingan antara Tim 4 dan Tim 5 dari slot waktu ke-1. Gambar 4.2.2b mengilustrasikan kondisi sebelum dan sesudah penyelesaian konflik kedua ini. Setelah semua konflik terselesaikan, pertandingan antara Tim 1 dan Tim 2 berhasil dijadwalkan pada slot waktu ke-1, seperti yang ditunjukkan pada Gambar 4.2.2c.

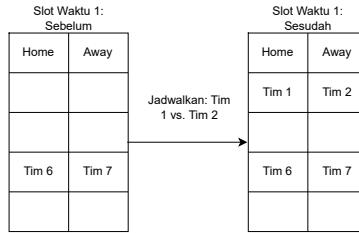
4.2.3 Tahapan *Perturbation*

Tahapan *perturbation* didesain seperti yang ditampilkan pada Algoritma 3. Tahapan *perturbation* dimulai dengan memilih secara acak kegiatan yang belum dijadwalkan. Selanjutnya, dilakukan perbandingan antara nilai acak yang dihasilkan (dalam rentang 0 hingga 1) dengan nilai variabel `probabilityNonRandomTimeSlot`. Jika nilai acak tersebut lebih kecil dari `probabilityNonRandomTimeSlot`, strategi yang digunakan adalah menjadwalkan kegiatan pada slot waktu dengan konflik paling sedikit. Jika terdapat beberapa slot waktu dengan jumlah konflik minimum yang sama, slot waktu



(a) Hapus Konflik Pertama

(b) Hapus Konflik Kedua



(c) Menjadwalkan Permainan antara Tim 1 vs Tim 2

Gambar 4.2.2: Ilustrasi dari Proses Penjadwalan Permainan antara Tim 1 vs Tim 2

tersebut akan dipilih secara acak. Sebaliknya, jika nilai acak lebih besar atau sama dengan `probabilityNonRandomTimeSlot`, maka slot waktu akan dipilih secara acak tanpa mempertimbangkan jumlah konflik.

Algoritma 3 Perturbation Phase

```

1: procedure PERTURBATIONPHASE(probabilityNonRandomTimeSlot)
2:    $x \leftarrow \text{getRandomUnscheduledEvent}()$ 
3:    $u \leftarrow \text{getRandomFloat}()$                                  $\triangleright$  returns a value in  $[0, 1)$ 
4:   if  $u < \text{probabilityNonRandomTimeSlot}$  then
5:      $y \leftarrow \text{getBestTimeSlot}(x)$ 
6:   else
7:      $y \leftarrow \text{getRandomTimeSlot}(x)$ 
8:   end if
9:    $x \leftarrow \text{scheduleEvent}(x, y)$ 
10: end procedure

```

4.2.4 Tahapan Local Search

Tahapan *local search* ditunjukkan pada Algoritma 4. Tahapan ini terdiri dari proses iteratif yang mencakup dua langkah utama. Langkah pertama bertujuan untuk menjadwalkan semua kegiatan yang belum terjadwal. Sebuah kegiatan memenuhi syarat untuk dijadwalkan pada tahapan ini jika tersedia slot waktu yang tidak memiliki konflik

dengan kegiatan lain atau slot waktu dengan konflik maksimal dengan satu kegiatan yang telah terjadwal. Dengan demikian, jumlah kegiatan yang dijadwalkan akan tetap atau meningkat. Langkah kedua adalah menjalankan proses pengacakan dengan menerapkan LLH pada solusi saat ini. Namun penerapan LLH harus tetap menjaga jumlah kegiatan yang dijadwalkan agar tidak berubah. Tujuan utama proses pengacakan ini adalah untuk meningkatkan peluang keberhasilan dalam menjadwalkan kegiatan yang belum terjadwal pada iterasi berikutnya.

Tahapan *local search* akan berakhir ketika salah satu dari kondisi terpenuhi. Kondisi pertama adalah ketika variabel `stuck` melebihi ambang batas yang ditentukan, yaitu sebesar nilai pada variabel `limitStuck`. Kondisi kedua terjadi ketika tidak ada peningkatan dalam jumlah kegiatan yang berhasil dijadwalkan dan proses pengacakan gagal menghasilkan solusi baru.

Algoritma 4 Local Search Phase

```

1: procedure LOCALSEARCHPHASE(limitStuck, limitShuffle)
2:   currentBest  $\leftarrow$  numberUnscheduledEvent()
3:   stuck  $\leftarrow$  0
4:   succeedShuffle  $\leftarrow$  true
5:   improvement  $\leftarrow$  false
6:   while (stuck  $\leq$  limitStuck) AND (succeedShuffle OR improvement) do
7:     improvement  $\leftarrow$  false
8:     shuffleUnscheduledEvent(x, y)
9:     for each unscheduledEvent in getUnscheduledEvents() do
10:      timeSlot  $\leftarrow$  getBestTimeSlot(unscheduledEvent)
11:      if calculateConflict(unscheduledEvent, timeSlot) < 2 then
12:        scheduleEvent(unscheduledEvent, timeSlot)
13:        improvement  $\leftarrow$  true
14:      end if
15:    end for
16:    if currentBest < numberUnscheduledEvent() then
17:      stuck  $\leftarrow$  0
18:      currentBest  $\leftarrow$  numberUnscheduledEvent()
19:    else
20:      stuck  $\leftarrow$  stuck + 1
21:    end if
22:    succeedShuffle  $\leftarrow$  shuffle(limitShuffle)
23:   end while
24: end procedure

```

Dalam proses pengacakan, tahapan yang dilakukan mengikuti langkah-langkah pada Algoritma 5. Proses pengacakan dimulai dengan memilih suatu kegiatan secara acak,

kemudian menerapkan LLH yang juga dipilih secara acak. Jika proses ini berhasil, maka prosedur pengacakan dianggap selesai. Namun, jika belum berhasil, proses akan diulang hingga jumlah iterasi mencapai batas yang ditentukan oleh variabel `limitShuffle`.

Algoritma 5 Shuffle Procedure

```

1: procedure SHUFFLE(limitShuffle)
2:   succeedShuffle  $\leftarrow$  false
3:   for  $i \leftarrow 1$  to limitShuffle do
4:      $x \leftarrow \text{getRandomUnscheduledEvent}()$ 
5:     succeedShuffle  $\leftarrow \text{doLLH}(\text{selectRandomLLH}(), x)$ 
6:     if succeedShuffle then
7:       return true
8:     end if
9:   end for
10:  return false
11: end procedure
```

4.2.5 Parameter

Berdasarkan desain algoritma yang telah dijelaskan, terdapat lima parameter yang berperan pada algoritma PA-ILS, seperti yang ditampilkan pada Tabel 4.2.1. Parameter `decreasingValue` dan `constantFactor` mempengaruhi intensitas tahapan eksploitasi dalam proses pencarian solusi. Kedua parameter ini memiliki rentang nilai antara lebih dari 0 hingga kurang dari 1.

Sementara itu, tiga parameter lainnya, berfungsi sebagai batasan untuk berbagai prosedur dalam algoritma. Parameter `limitStuck` dan `limitShuffle` berperan dalam membatasi prosedur pada tahapan *local search*. Parameter `limitUpdateProbabilityNonRandomTimeSlot` menentukan kapan probabilitas akan diperbarui melalui fungsi `calculateProbability`. Parameter `limitStuck` digunakan untuk menentukan kapan tahapan *local search* akan berakhir, sementara `limitShuffle` membatasi jumlah iterasi dalam proses pengacakan pada solusi.

Tabel 4.2.1: Daftar Parameter

Parameter	Rentang Nilai
<code>decreasingValue</code>	(0, 1)
<code>constantFactor</code>	(0, 1)
<code>limitUpdateProbabilityNonRandomTimeSlot</code>	> 0 (bilangan bulat positif)

(Lanjutan) Daftar Parameter

Parameter	Rentang Nilai
limitStuck	> 0 (bilangan bulat positif)
limitShuffle	> 0 (bilangan bulat positif)

4.3 Desain Algoritma AT-ILS

Pada desain algoritma AT-ILS, terdapat tiga aspek pengembangan utama yang menjadi kunci dari algoritma ini. Pengembangan pertama adalah penggunaan strategi nilai ambang batas pada tiga tahapan algoritma ILS yaitu *perturbation*, *local search*, dan *move acceptance*. Pada tahapan *perturbation*, nilai ambang batas akan menjadi batas seberapa besar solusi buruk yang memungkinkan untuk diterima.

Pada tahapan *local search*, penggunaan nilai ambang batas ditujukan untuk dapat menghasilkan proses pencarian solusi yang memiliki keseimbangan tahapan eksplorasi dan eksploitasi yang menyerupai algoritma Simulated Annealing. Penerapan nilai ambang batas memungkinkan proses pencarian menghasilkan solusi yang lebih buruk selama masih dibawah nilai ambang batas. Namun secara keseluruhan, proses pencarian pada tahapan *local search* akan mampu menurunkan nilai penalti seiring dengan nilai ambang batas yang akan terus menurun.

Sementara pada tahapan *move acceptance*, nilai ambang batas bekerja untuk memberikan peluang menerima solusi yang lebih buruk. Nilai ambang batas menjadi batasan seberapa buruk nilai solusi yang masih bisa diterima dalam tahapan *move acceptance*.

Selanjutnya, pengembangan kedua adalah desain strategi kembali ke solusi terbaik. Strategi ini diterapkan terhadap tahapan *move acceptance*. Ketika solusi baru tidak diterima pada tahapan ini, strategi ini memungkinkan untuk menggunakan solusi terbaik sebagai solusi yang digunakan dalam tahapan iterasi selanjutnya. Hal ini didesain untuk mengatasi permasalahan pencarian solusi yang memungkinkan untuk menjauhi solusi terbaik hingga akhir iterasi.

Pengembangan ketiga adalah algoritma AT-ILS didesain tanpa membutuhkan

pengaturan nilai parameter. Hal ini ditujukan untuk mengatasi kelemahan Algoritma Simulated Annealing yang sensitif terhadap perubahan nilai parameter. Strategi ini dikembangkan dengan mengembangkan daftar nilai ambang batas yang berisi selisih nilai solusi saat dilakukan penerapan LLH.

Untuk menjelaskan desain pengembangan algoritma AT-ILS secara detail, Subbab 4.3.1 menjelaskan struktur algoritma ini secara umum. Selanjutnya, Subbab 4.3.2 menjelaskan bagaimana konsep dari daftar nilai ambang batas dan proses melakukan perhitungan nilai ambang batas. Subbab 4.3.3, 4.3.4 dan 4.3.5 menjelaskan ketiga tahapan utama dalam algoritma AT-ILS.

4.3.1 Desain Umum Algoritma AT-ILS

Desain algoritma AT-ILS dikembangkan dengan menggunakan tiga tahapan dalam algoritma ILS sebagai struktur dasar. Secara keseluruhan, Algoritma 6 menunjukkan desain dari algoritma AT-ILS. Algoritma didesain menerima dua input. Input pertama berupa solusi awal yang *feasible* (`initialSol`), yang disimpan pada variabel `currentSol`. Input kedua berupa `eventList`, yang berisi daftar kegiatan dalam solusi awal. Selain itu, terdapat dua variabel yang diinisiasi yaitu `currentSol` dan `bestSol`. Kedua variabel diinisiasi dengan nilai input `initialSol`.

Tahapan selanjutnya, algoritma ini melakukan iterasi hingga waktu yang telah ditentukan tercapai. Dalam setiap iterasi, tiga tahapan utama dijalankan yaitu tahapan *perturbation*, *local search*, dan *move acceptance*. Selain itu dua fungsi lainnya yaitu untuk menghitung nilai dari variabel `thresholdList` dan `thresholdValue` diperbarui disetiap awal iterasi.

4.3.2 Fungsi dan Perhitungan Nilai Ambang Batas

Pada algoritma AT-ILS, terdapat dua variabel yang berkaitan dengan nilai ambang batas yaitu `thresholdList` dan `thresholdValue`. Variabel `thresholdValue` merupakan variabel yang digunakan sebagai nilai ambang batas pada tiga tahapan dalam algoritma AT-ILS. Nilai dari variabel `thresholdValue` didapatkan dengan menghitung nilai rata-rata dari nilai yang terdapat pada variabel `thresholdList`. Dalam proses penerapannya, daftar nilai pada variabel `thresholdList` akan dihapus secara perlahan

Algoritma 6 AT-ILS

```
1: procedure OPTIMIZESOLUTION(initialSol,eventList)
2:   currentSol  $\leftarrow$  initialSol
3:   bestSol  $\leftarrow$  currentSol
4:   while within duration of running time do
5:     thresholdList  $\leftarrow$  calculateThreshold(currentSol,eventList)
6:     thresholdValue  $\leftarrow$  updateThreshold(thresholdList)
7:     newSol  $\leftarrow$  currentSol
8:     perturbationPhase(newSol,eventList,thresholdValue)
9:     localSearchPhase(newSol,eventList,thresholdList)
10:    moveAcceptancePhase(newSol,currentSol,bestSol)
11:   end while
12: end procedure
```

mulai dari nilai terbesar. Hal ini menyebabkan nilai pada variabel *thresholdValue* perlahan akan berkurang. Strategi ini ditujukan untuk mengarahkan solusi pada konvergensi terutama dalam tahapan *local search*.

Pengembangan variabel *thresholdList* ditujukan untuk mengetahui perkiraan jangkauan solusi yang dapat berubah saat LLH diterapkan terhadap solusi terkini. Setiap permasalahan memungkinkan memiliki jangkauan yang berbeda walaupun diterapkan LLH yang sama. Selain itu, pada dataset yang sama, kondisi solusi saat ini juga mampu mempengaruhi jangkauan yang bisa dihasilkan saat LLH diterapkan. Sebagai contoh, penerapan LLH dapat menghasilkan solusi baru dengan perbedaan nilai penalti yang besar ketika nilai penalti dari solusi saat ini masih tinggi. Namun ketika nilai penalti dari solusi saat ini sudah jauh lebih rendah, maka memungkinkan untuk menghasilkan solusi baru dengan perbedaan penalti yang lebih kecil. Hal ini mungkin berlaku sebaliknya, bergantung dari jenis dan studi kasus yang akan diselesaikan. Oleh karena itu, menggunakan variabel *thresholdList* untuk menghitung nilai ambang batas dapat menghasilkan nilai ambang batas yang lebih adaptif terhadap permasalahan dan studi kasus yang akan diselesaikan.

Secara detail, proses penghitungan nilai variabel *thresholdList* dengan menjalankan fungsi *calculateThreshold* ditunjukkan pada algoritma 7. Fungsi *calculateThreshold* membutuhkan dua input berupa solusi saat ini (*currentSol*) dan daftar kegiatan yang dijadwalkan (*eventList*). Selanjutnya terdapat dua variabel yang perlu dideklarasikan dan diinisialisasi. Pertama, variabel *currentPenalty* dideklarasikan dan diinisialisasi dengan menghitung nilai penalti saat ini. Kedua, variabel

`thresholdList` diinisiasi menjadi variabel dengan daftar nilai yang kosong.

Selanjutnya, proses perulangan dijalankan dengan menerapkan LLH pada seluruh daftar kegiatan. Jika hasil dari penerapan LLH ini *feasible* dan memiliki nilai penalti yang berbeda dari solusi saat ini, maka perbedaan tersebut dicatat dalam variabel `thresholdList`. Perubahan yang dilakukan selalu dikembalikan ke kondisi awal pada setiap iterasi agar tidak mempengaruhi proses optimasi. Jika `thresholdList` tetap kosong setelah perulangan, algoritma akan mengulang proses ini hingga `thresholdList` memiliki nilai. Setelah perulangan selesai, `thresholdList` diurutkan dan hasil akhirnya dikembalikan kepada pemanggil fungsi.

Algoritma 7 Calculate Threshold

```
1: procedure CALCULATETHRESHOLD(currentSolution, eventList)
2:   currentPenalty  $\leftarrow$  calculatePenalty(currentSolution)
3:   thresholdList  $\leftarrow$  an empty list
4:   while thresholdList is empty do
5:     for each event in eventList do
6:       chosenLLH  $\leftarrow$  selectRandomLLH()
7:       doLLH(chosenLLH, event)
8:       if checkFeasibility(currentSolution) is true then
9:         newPenalty  $\leftarrow$  calculatePenalty(currentSolution)
10:        if newPenalty  $\neq$  currentPenalty then
11:          difference  $\leftarrow$  |currentPenalty – newPenalty|
12:          append(thresholdList, difference)
13:        end if
14:      end if
15:      undoChanges(event)
16:    end for
17:  end while
18:  sort(thresholdList)
19:  return thresholdList
20: end procedure
```

4.3.3 Tahapan Perturbation

Tahapan *perturbation* merupakan tahapan yang bertujuan untuk mengeksplorasi solusi. Pada tahapan ini, proses eksplorasi didesain seperti yang ditampilkan pada Algoritma 8. Tahapan ini dimulai dengan menerima input berupa solusi baru saat ini (`newSol`), daftar kegiatan (`eventList`) dan nilai ambang batas (`thresholdValue`). Langkah pertama adalah mendeklarasikan variabel `perturbation` dengan nilai awal `true`, yang menandakan bahwa tahapan *perturbation* sedang berjalan.

Selanjutnya, variabel `currentPenalty` dideklarasikan dan diinisialisasi dengan nilai penalti dari solusi pada variabel `newSol`. Algoritma kemudian memasuki perulangan di mana, pada setiap iterasi, LLH diterapkan pada kegiatan yang dipilih secara acak dari kegiatan dalam variabel `eventList`. Solusi baru diterima jika solusi tersebut *feasible* tanpa mempertimbangkan nilai penaltinya. Jika solusi tidak *feasible*, maka solusi dikembalikan ke kondisi awal menggunakan fungsi `undoChanges`.

Tahapan *perturbation* akan berakhir ketika nilai penalti dari solusi saat ini (`newPenalty`) melebihi nilai `currentPenalty` ditambah `thresholdValue`. Variabel *perturbation* diubah menjadi `false` yang menyatakan proses *perturbation* berakhir. Penggunaan variabel `thresholdValue` ditujukan untuk memastikan dampak dari proses eksplorasi solusi tidak terlalu kecil. Hal ini diharapkan mampu menghasilkan solusi baru yang lebih baik saat diterapkan tahapan *local search*.

Algoritma 8 Perturbation Phase

```

1: procedure PERTURBATIONPHASE(newSol, eventList, thresholdValue)
2:   perturbation  $\leftarrow$  true
3:   currentPenalty  $\leftarrow$  calculatePenalty()
4:   while perturbation do
5:     chosenLLH  $\leftarrow$  selectRandomLLH()
6:     doLLH(chosenLLH, event.get(random))
7:     if checkFeasibility() then
8:       newPenalty  $\leftarrow$  calculatePenalty()
9:       if newPenalty  $>$  currentPenalty + thresholdValue then
10:        perturbation  $\leftarrow$  false
11:      end if
12:    else
13:      undoChanges()
14:    end if
15:   end while
16: end procedure

```

4.3.4 Tahapan Local Search

Tahapan *local search* merupakan tahapan yang berfokus untuk mengeksplorasi solusi untuk dapat menghasilkan solusi terbaik baru. Tahapan ini didesain seperti yang ditunjukkan pada Algoritma 9. Tahapan ini dimulai dengan menerima input berupa solusi baru dari tahapan *perturbation* (`newSol`), daftar kegiatan (`eventList`), nilai ambang batas (`thresholdValue`), serta daftar nilai ambang batas (`thresholdList`). Selanjutnya, algoritma melakukan deklarasi dan inisialisasi beberapa variabel sebagai

berikut:

- `localSearch`: variabel ini berfungsi untuk menandai kapan tahapan *local search* akan berakhir. Variabel ini diinisialisasi dengan nilai awal `true`.
- `currentPenalty`: variabel untuk mencatat nilai penalti solusi saat ini. Variabel ini diinisialisasi dengan menghitung nilai penalti dari `newSol`.
- `localBest`: variabel ini berfungsi untuk mencatat nilai penalti terbaik yang ditemukan selama tahapan *local search*. Variabel ini diinisialisasi menggunakan nilai dari variabel `currentPenalty`.
- `thresholdLocal`: variabel ini digunakan untuk membatasi penerimaan solusi dalam tahapan *local search*. Variabel ini diinisialisasi dengan nilai dari `currentPenalty`.
- `improve`: variabel ini menandakan apakah terdapat peningkatan solusi selama penerapan fungsi `ApplyHeuristicToAllEvents`. Ketika variabel ini bernilai `true` maka menandakan terjadinya peningkatan solusi. Sebaliknya, ketika variabel ini bernilai `false`, maka manandakan tidak adanya peningkatan solusi. Variabel ini diinisialisasi dengan nilai awal `false`.
- `improveLocalBest`: variabel ini menandakan apakah terjadi peningkatan pada variabel `localBest` selama penerapan fungsi `ApplyHeuristicToAllEvents`. Ketika variabel ini bernilai `true` maka menandakan terjadinya peningkatan solusi pada variabel `localBest`. Sebaliknya, ketika variabel ini bernilai `false` manandakan tidak adanya peningkatan solusi pada variabel `localBest`. Variabel ini diinisialisasi dengan nilai awal `false`.
- `amountRemoved`: variabel ini menentukan jumlah elemen yang akan dihapus dari `eventList` dalam fungsi `UpdateThresholdList`. Variabel ini diinisialisasi dengan nilai 1. Nilai pada variabel ini akan bertambah seiring berjalannya tahapan *local search*. Hal ini bertujuan untuk meningkatkan secara lebih masif penghapusan pada variabel `eventList` yang berdampak pada semakin kecilnya nilai ambang batas pada variabel `thresholdValue`.

Setelah deklarasi dan inisialisasi variabel, tahapan *local search* dimulai melalui proses iteratif. Dalam proses ini, Terdapat tiga langkah utama. Langkah pertama adalah

pembaruan nilai variabel `thresholdLocal` melalui pemanggilan fungsi `updateThresholdLocal`. Detail mengenai proses pembaruan ini dijelaskan pada Subbab 4.3.4.1. Langkah kedua adalah menerapkan perubahan pada solusi dengan menjalankan fungsi `ApplyHeuristicToAllEvents`. Dalam fungsi ini, perubahan yang diterima tidak boleh melebihi batas nilai yang ditentukan oleh variabel `thresholdLocal`. Penjelasan lebih lanjut tentang fungsi ini dijelaskan pada Subbab 4.3.4.2.

Langkah ketiga, algoritma memperbarui nilai variabel `thresholdList` dan `thresholdValue` dengan menjalankan fungsi `UpdateThresholdList`. Fungsi ini juga memeriksa apakah tahapan *local search* akan berakhir atau dilanjutkan. Penjelasan lengkap mengenai fungsi ini dijabarkan pada Subbab 4.3.4.1. Setelah iterasi berakhir, solusi terkini akan dikembalikan menjadi solusi terbaik yang ditemukan dalam tahapan *local search* dengan menjalankan fungsi `useLocalBest`.

4.3.4.1 Fungsi *UpdateThresholdLocal*

Dalam tahapan *local search*, variabel `thresholdLocal` merupakan variabel yang menjadi nilai ambang batas dalam proses penerimaan solusi baru. Variabel `thresholdLocal` dihitung dengan menjalankan fungsi `UpdateThresholdLocal` yang melibatkan variabel `thresholdList` dan `thresholdValue`. Fungsi ini didesain seperti yang ditampilkan pada Algoritma 10.

Proses dimulai dengan memeriksa apakah variabel `thresholdList` kosong atau tidak. Jika variabel `thresholdList` kosong, maka variabel `thresholdLocal` akan diatur ke nilai penalti saat ini. Jika variabel `thresholdList` tidak kosong, maka langkah berikutnya adalah mengecek apakah hasil pengurangan variabel `thresholdLocal` dengan nilai pada indeks pertama dari variabel `thresholdList` tetap lebih besar dari nilai penalti saat ini. Jika benar, maka variabel `thresholdLocal` diperbarui dengan nilai pengurangan tersebut. Namun, jika hasilnya sama atau lebih kecil dari nilai penalti saat ini, maka variabel `thresholdLocal` akan diatur sama dengan nilai penalti saat ini. Penggunaan nilai indeks pertama dari variabel `thresholdList` bertujuan untuk memberikan ruang pada solusi untuk melakukan eksplorasi.

Selain itu, terdapat pengecekan tambahan untuk memastikan solusi tidak terjebak

Algoritma 9 Local Search Phase

```
1: procedure LOCALSEARCHPHASE(newSol, eventList, thresholdValue, thresholdList)
2:   localSearch  $\leftarrow$  true
3:   currentPenalty  $\leftarrow$  calculatePenalty(newSol)
4:   localBest  $\leftarrow$  currentPenalty
5:   thresholdLocal  $\leftarrow$  currentPenalty
6:   improve  $\leftarrow$  false
7:   improveLocalBest  $\leftarrow$  false
8:   amountRemoved  $\leftarrow$  1
9:   while localSearch do
10:    thresholdLocal  $\leftarrow$  UpdateThresholdLocal(thresholdLocal,
11:                                              thresholdList, currentPenalty, thresholdValue, improve)
12:    improveLocalBest  $\leftarrow$  ApplyHeuristicToAllEvents(
13:      eventList, thresholdLocal,
14:      currentPenalty, localBest)
15:    if NOT improve then
16:      localSearch  $\leftarrow$  UpdateThresholdList(thresholdList,
17:                                              amountRemoved, improveLocalBest,
18:                                              localBest, currentPenalty)
19:    end if
20:   end while
21:   useLocalBest()
22: end procedure
```

dalam kondisi *local optima*. Jika solusi tidak mengalami peningkatan dari iterasi sebelumnya, maka nilai dari variabel `thresholdLocal` akan ditambah dengan nilai dari variabel `thresholdValue`. Jika terjadi peningkatan, maka variabel `improve` diatur menjadi `false` untuk melakukan pengecekan peningkatan pada iterasi berikutnya.

4.3.4.2 Fungsi *ApplyHeuristicToAllEvents*

Bagian ini bertujuan untuk meningkatkan solusi dalam setiap iterasinya. Proses dalam fungsi ini ditunjukkan pada Algoritma 11. Tahapan dimulai dengan mengacak urutan dalam variabel `eventList` dan menginisialisasi variabel `improveLocalBest` dengan nilai `false`. Langkah selanjutnya menerapkan LLH yang dipilih secara acak pada setiap kegiatan dalam `eventList` untuk menghasilkan solusi baru.

Setiap solusi baru yang dihasilkan akan dilakukan pengecekan untuk memutuskan apakah solusi baru akan digunakan atau tidak. Jika solusi yang dihasilkan *feasible*, maka dilanjutkan dengan pengecekan nilai penalti. Jika solusi tidak *feasible*, solusi akan dikembalikan ke kondisi sebelumnya melalui fungsi `undoChanges`. Pada pengecekan

Algoritma 10 Update Threshold Local

```
1: procedure UPDATETHRESHOLDLOCAL(thresholdLocal, thresholdList,  
   currentPenalty, thresholdValue, improve)  
2:   if thresholdList is not empty then  
3:     firstElement  $\leftarrow$  the first (or smallest) element of thresholdList  
4:     if (thresholdLocal – firstElement) > currentPenalty then  
5:       thresholdLocal  $\leftarrow$  thresholdLocal – firstElement  
6:     else  
7:       thresholdLocal  $\leftarrow$  currentPenalty  
8:     end if  
9:   else  
10:    thresholdLocal  $\leftarrow$  currentPenalty  
11:   end if  
12:   if not improve then  
13:     thresholdLocal  $\leftarrow$  thresholdLocal + thresholdValue  
14:   else  
15:     improve  $\leftarrow$  false  
16:   end if  
17:   return thresholdLocal  
18: end procedure
```

nilai penalti, terdapat dua persyaratan yang menggunakan logika OR, yang berarti solusi akan diterima jika salah satu persyaratan terpenuhi. Persyaratan pertama adalah solusi baru memiliki nilai penalti yang lebih kecil dari nilai pada variabel *thresholdLocal*. Persyaratan kedua adalah nilai penalti solusi baru sama dengan nilai pada variabel *thresholdLocal* dan nilai penalti pada variabel *currentPenalty* juga sama dengan nilai variabel *thresholdLocal*. Jika solusi diterima dan memiliki nilai lebih kecil dari variabel *thresholdLocal*, maka variabel *improve* akan diubah menjadi true untuk menunjukan adanya peningkatan solusi. Selain itu, variabel *improveLocalBest* akan diperbarui jika ditemukan solusi terbaik lokal yang baru.

Penggunaan dua persyaratan untuk penerimaan solusi dari sisi nilai penalti ini bertujuan untuk mencegah solusi yang sudah meningkat kembali ke nilai penalti yang sama dengan nilai *thresholdLocal*. Dalam fungsi ini, nilai ambang batas tidak mengalami perubahan nilai. Jika syarat penerimaan hanya diterapkan pada solusi baru yang lebih kecil atau sama dengan ambang batas, terdapat kemungkinan solusi kembali ke nilai yang sama dengan nilai ambang batas. Sebagai contoh, LLH swap diterapkan pada kegiatan 1 dengan kegiatan 3 dan mampu menghasilkan nilai penalti yang lebih baik, sehingga solusi ini diterima. Namun saat LLH swap diterapkan pada kegiatan 3 dan 1

dalam beberapa iterasi berikutnya, terdapat kemungkinan akan menghasilkan nilai penalti yang sama sebelum LLH *swap* diterapkan pada kegiatan 1 dengan kegiatan 3. LLH *swap* yang awalnya menukar kan kegiatan 1 dengan kegiatan 3 pada iterasi berikutnya kembali menukarkan kegiatan 3 dan 1 yang menghasilkan tidak adanya perubahan solusi. Untuk menghindari kondisi tersebut, dua syarat penerimaan yang telah dijelaskan sebelumnya digunakan.

Algoritma 11 Apply Heuristic to Events

```

1: procedure APPLYHEURISTICTOEVENTS(eventList, thresholdLocal, currentPenalty,
   localBest)
2:   shuffle(eventList)
3:   improveLocalBest  $\leftarrow$  false
4:   for each event in eventList do
5:     chosenLLH  $\leftarrow$  selectRandomLLH()
6:     doLLH(chosenLLH, event)
7:     if isFeasible() then
8:       newPenalty  $\leftarrow$  calculatePenalty()
9:       if (newPenalty < thresholdLocal) OR ((newPenalty = thresholdLocal)
   AND (currentPenalty = thresholdLocal)) then
10:        currentPenalty  $\leftarrow$  newPenalty
11:        if currentPenalty < thresholdLocal then
12:          improve  $\leftarrow$  true
13:        end if
14:        if currentPenalty < localBest then
15:          localBest  $\leftarrow$  currentPenalty
16:          saveBestLocalSol()
17:          improveLocalBest  $\leftarrow$  true
18:        end if
19:        else
20:          undoChanges(event)
21:        end if
22:        else
23:          undoChanges(event)
24:        end if
25:      end for
26:      return improveLocalBest
27: end procedure

```

4.3.4.3 Fungsi *UpdateThresholdList*

Bagian ini merupakan fungsi untuk memperbarui nilai dari *thresholdList*. Perbaruan dilakukan dengan mengurangi jumlah elemen pada *thresholdList* untuk menurunkan nilai *thresholdValue*. Pengurangan ini bertujuan untuk mengurangi

eksplorasi dan memfokuskan proses pada tahapan eksplorasi sebelum solusi akhir diserahkan pada tahapan *move acceptance*. Algoritma 12 menunjukkan desain dari proses ini. Pada tahapan pertama, proses pengecekan terhadap jumlah elemen dalam `thresholdList` dilakukan. Jika `thresholdList` tidak memiliki satupun elemen, maka tidak ada elemen yang dapat dihapus. Hal ini menandakan akhir dari tahapan *local search*. Jika hal ini terjadi, fungsi `UpdateThresholdList` akan mengembalikan nilai `false`. Jika `thresholdList` masih memiliki elemen, maka proses penghapusan diterapkan.

Proses penghapusan dimulai dengan mengecek apakah terdapat peningkatan pada `localBest` pada tahapan `ApplyHeuristicToAllEvents` yang ditandai dalam variabel `improveLocalBest`. Jika ada, maka jumlah penghapusan akan diubah menjadi 1 dan variabel `improveLocalBest` diatur menjadi `false` untuk digunakan pada iterasi berikutnya. Jika tidak ada peningkatan, maka jumlah penghapusan akan ditingkatkan dengan menambah nilai 1 pada variabel `amountRemoved`. Tahap selanjutnya adalah menghapus elemen dari `thresholdList` sebanyak nilai `amountRemoved`. Penghapusan dimulai dari nilai terbesar yang terdapat pada variabel `thresholdList`.

Setelah penghapusan selesai, solusi mungkin akan dikembalikan ke kondisi `localBest` berdasarkan persyaratan berikut. Jika nilai penalti solusi saat ini lebih besar dari `localBest` ditambah salah satu nilai dari kuartil pertama pada `thresholdList`, maka solusi saat ini akan diatur kembali ke solusi terbaik lokal.

Langkah terakhir adalah memperbarui `thresholdValue`. Jika `thresholdList` masih memiliki elemen setelah proses penghapusan, maka nilai `thresholdValue` diperbarui melalui fungsi `updateThreshold`. Jika tidak ada elemen yang tersisa, maka `thresholdValue` akan diatur menjadi 0.

4.3.5 Tahapan *Move Acceptance*

Tahapan *move acceptance* bertujuan untuk memutuskan apakah solusi baru akan digunakan pada iterasi berikutnya. Algoritma 13 menunjukkan desain dari tahapan ini. Pertama, tiga solusi — solusi saat ini, solusi baru dan solusi terbaik — akan dihitung nilai penaltinya. Selanjutnya, jika nilai penalti solusi baru lebih kecil atau sama dengan penalti

Algoritma 12 Update Threshold List

```
1: procedure      UPDATETHRESHOLDLIST(thresholdList,           amountRemoved,
   improveLocalBest, localBest, currentPenalty)
2:   if thresholdList is empty then
3:     return false                                ▷ Indicate the end of the local search phase
4:   else
5:     if improveLocalBest then
6:       amountRemoved ← 1
7:       improveLocalBest ← false
8:     else
9:       amountRemoved ← amountRemoved + 1
10:    end if
11:    for k ← 1 to amountRemoved do
12:      if thresholdList is empty then
13:        break                                 ▷ No more thresholds to remove
14:      end if
15:      removeLastElement(thresholdList)
16:    end for
17:    if (thresholdList is not empty) AND (currentPenalty > localBest +  
getRandomQuartil1(thresholdList)) then
18:      useLocalBest()
19:      currentPenalty ← localBest
20:    end if
21:    if thresholdList has more than one element then
22:      thresholdValue ← updateThreshold(thresholdList)
23:    else
24:      thresholdValue ← 0
25:    end if
26:  end if
27:  return true                               ▷ Continue the local search
28: end procedure
```

solusi saat ini, maka solusi baru akan menggantikan solusi saat ini dan solusi tersebut digunakan pada iterasi berikutnya. Selain itu, dilakukan pengecekan apakah nilai penalti solusi baru lebih kecil daripada solusi terbaik. Jika iya, maka solusi baru juga akan menggantikan solusi terbaik.

Sementara itu, jika solusi baru memiliki penalti lebih tinggi daripada solusi saat ini, maka dijalankan proses pengecekan berikutnya. Sebelum melakukan pengecekan lebih lanjut, `thresholdList` yang telah kosong setelah tahapan *local search* akan diisi ulang dengan memanggil fungsi `calculateThreshold`. Selanjutnya, dalam pengecekan kedua, jika nilai penalti solusi baru lebih kecil dari solusi terbaik yang ditambahkan dengan salah satu nilai kuartil pertama dari `thresholdList` yang diambil secara acak, maka solusi baru akan digunakan sebagai solusi saat ini. Sebaliknya, jika syarat ini tidak terpenuhi, solusi terbaik akan digunakan untuk iterasi berikutnya. Penggunaan solusi terbaik dibandingkan solusi saat ini bertujuan agar proses pencarian tidak semakin menjauh dari kondisi terbaik yang sudah ditemukan.

4.4 LLH

Dalam pengembangan suatu algoritma heuristik, LLH berfungsi sebagai operator untuk merubah solusi. Pada desain algoritma ini, tiga jenis LLH yang digunakan adalah *move*, *swap* dan *kempe chain*. Pemilihan ketiga LLH tersebut didasarkan pada tujuan algoritma, yaitu untuk menguji tingkat generalitasnya. Oleh karena itu, LLH yang diterapkan merupakan LLH yang sederhana dan cukup fleksibel untuk diterapkan pada berbagai jenis permasalahan penjadwalan. Penjelasan masing-masing LLH dijabarkan dalam Subbab 4.4.1, 4.4.2, dan 4.4.3.

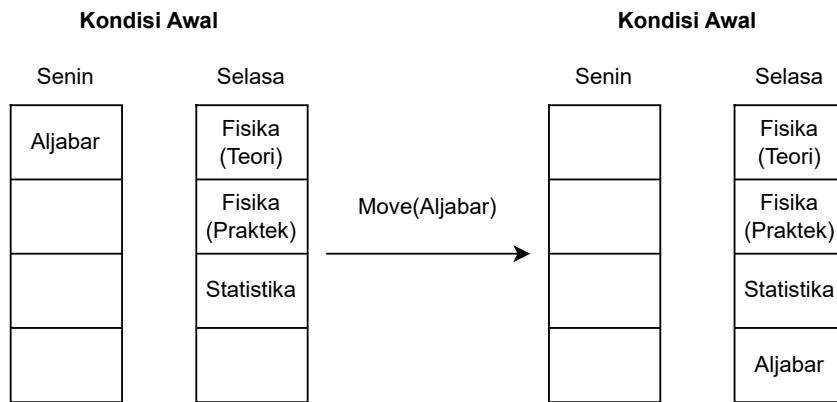
4.4.1 *Move*

LLH *move* adalah operator yang memindahkan jadwal dari satu kegiatan ke slot waktu lain. Pada kasus penjadwalan mata kuliah, operator ini memungkinkan digunakan untuk memindahkan jadwal dari suatu kelas ke slot waktu lain. Gambar 4.4.1 memperlihatkan ilustrasi LLH *move* yang diterapkan pada untuk memindahkan kelas Aljabar. Pada kondisi awal, kelas Aljabar berada pada slot waktu ke-1 di hari Senin. Dengan menggunakan operator *move*, jadwal kelas Aljabar dipindahkan ke slot waktu ke-4 di hari Selasa,

Algoritma 13 Move Acceptance

```
1: procedure MOVEACCEPTANCE(newSol, currentSol, bestSol, eventList)
2:   newPenalty  $\leftarrow$  calculatePenalty(newSol)
3:   currentPenalty  $\leftarrow$  calculatePenalty(currentSol)
4:   bestPenalty  $\leftarrow$  calculatePenalty(bestSol)
5:   if currentPenalty  $\geq$  newPenalty then
6:     currentSol  $\leftarrow$  newSol
7:     if newPenalty < bestPenalty then
8:       bestSol  $\leftarrow$  newSol
9:     end if
10:   else
11:     thresholdList  $\leftarrow$  calculateThreshold(bestSol, eventList)
12:     if newPenalty < bestPenalty + GetRandomFromTopQuartile(thresholdList)
then
13:       currentSol  $\leftarrow$  newSol
14:     else
15:       useBestSol()
16:     end if
17:   end if
18: end procedure
```

sehingga slot waktu ke-1 hari Senin menjadi kosong, sementara slot waktu ke-4 di hari Selasa kini terisi oleh kelas Aljabar.

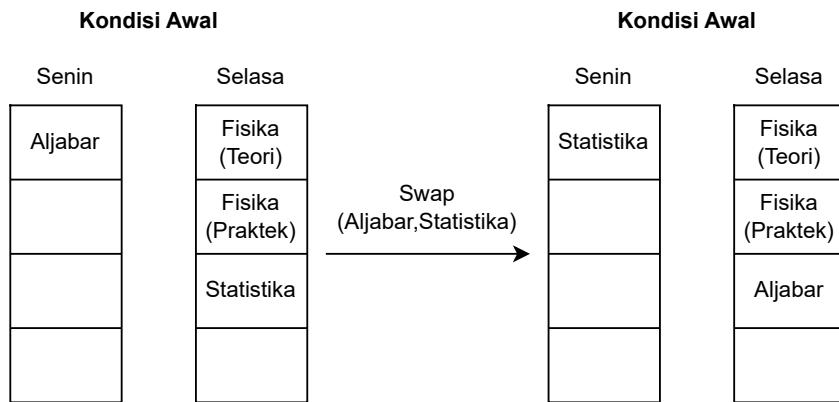


Gambar 4.4.1: Ilustrasi Operator *Move*

4.4.2 Swap

LLH *swap* adalah operator yang menukar jadwal antara dua kegiatan. Pada kasus penjadwalan, operator *swap* dapat diterapkan seperti pada ilustrasi dalam Gambar 4.4.2. Pada ilustrasi ini LLH *swap* diterapkan untuk menukar jadwal kelas Aljabar di slot waktu

ke-1 pada hari Senin dengan kelas Statistika yang berada di slot waktu ke-3 pada hari Selasa. Hasil dari penerapan operator ini kelas Aljabar berada pada slot waktu ke-3 di hari Selasa dan kelas Statistika berada pada slot waktu ke-1 di hari Senin.



Gambar 4.4.2: Ilustrasi Operator *Swap*

4.4.3 *Kempe Chain*

LLH *kempe chain* adalah operator yang memperluas konsep *swap* atau *move* dengan cara mengelompokkan kegiatan yang saling terkait dan kemudian memindahkan seluruh kelompok ini untuk memenuhi batasan penjadwalan. Gambar 4.4.3 menunjukkan ilustrasi penggunaan operator *kempe chain*. Dalam contoh ini kondisi penjadwalan awal mata kuliah sebagai berikut:

- Kelas Aljabar dijadwalkan pada hari Senin, Slot ke-1.
- Kelas Fisika Teori dijadwalkan pada hari Selasa, Slot ke-1.
- Kelas Praktikum Fisika dijadwalkan pada hari Selasa, Slot ke-2.

Dalam contoh ini terdapat *hard constraint* berupa kelas Fisika Teori dan Praktikum Fisika harus dijadwalkan pada hari yang sama. Ketika kita ingin menukar jadwal kelas Aljabar dengan kelas Fisika Teori, pertukaran ini akan menyebabkan kondisi tidak *feasible*, karena kelas Fisika Teori (yang berpindah ke Senin) akan terpisah dari kelas Praktikum Fisika, yang seharusnya berada pada hari yang sama. Untuk menyelesaikan masalah ini, kita menerapkan LLH *kempe chain* dengan langkah berikut:

1. Menukar Kelas Aljabar dan Kelas Fisika Teori. Kelas Aljabar dipindahkan ke hari

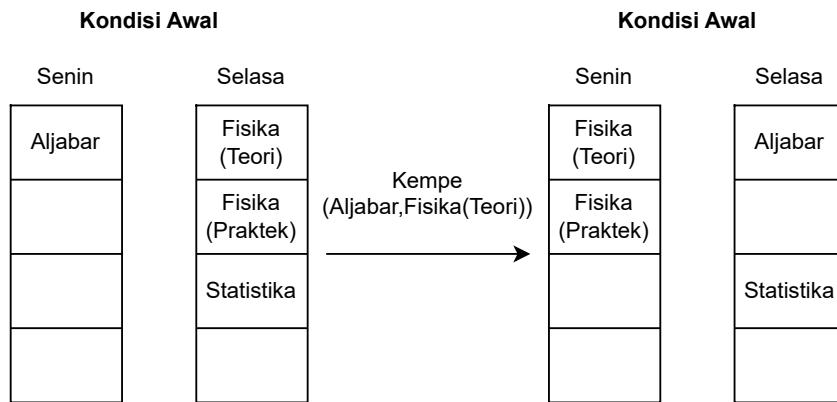
Selasa pada slot ke-1 dan kelas Fisika Teori dipindahkan ke hari Senin pada slot waktu ke-1.

2. Karena kelas Fisika Teori memiliki ketergantungan dengan kelas Praktikum Fisika, maka kelas Praktikum Fisika juga perlu dipindahkan untuk tetap berada di hari yang sama dengan kelas Fisika Teori. Maka kelas Praktikum Fisika dipindahkan ke hari Senin pada slot waktu ke-2.

Hasil akhir setelah penerapan LLH *kempe chain*:

- Kelas Aljabar dipindahkan ke hari Selasa, Slot 1.
- Kelas Fisika Teori dipindahkan ke hari Senin, Slot 1.
- Kelas Praktikum Fisika dipindahkan ke hari Senin, Slot 2.

Dengan demikian, operator kempe chain ini mempertahankan keterkaitan antara Fisika Teori dan Praktikum Fisika, sehingga tetap memenuhi batasan jadwal yang mengharuskan kedua kelas tersebut berada pada hari yang sama. Penggunaan kempe chain efektif untuk mengatasi ketergantungan antar kegiatan, terutama dalam situasi penjadwalan kompleks yang melibatkan beberapa batasan terkait.



Gambar 4.4.3: Ilustrasi Operator Kempe

4.5 Pengaturan Parameter

Berdasarkan penjelasan pada Subbab 4.2.5, algoritma PA-JLS memiliki lima parameter. Bagian ini akan menentukan nilai dari kelima parameter yang akan digunakan. Penentuan

nilai ini dilakukan secara sederhana berdasarkan percobaan pada beberapa nilai dan juga analisis dari strategi yang dikembangkan. Percobaan ini tidak ditujukan untuk menemukan nilai parameter paling optimal, melainkan hanya untuk menentukan nilai parameter yang cukup baik karena pengujian algoritma ditujukan dengan fokus generalitas.

Untuk mewujudkan hal tersebut, uji coba hanya dijalankan pada *benchmark* ITC 2021 dengan memilih tiga dataset secara acak yang terdiri dari Early 3, Middle 10, dan Late 1. Proses pengujian dijalankan pada tiga dataset dengan pengujian sebanyak sepuluh kali untuk setiap dataset pada setiap nilai parameter yang diuji. Metrik tingkat keberhasilan, rata-rata waktu untuk menemukan solusi *feasible* dan konsistensi waktu dalam bentuk Standar Deviasi (SD) digunakan sebagai penentu nilai yang akan digunakan.

Percobaan pertama dilakukan terhadap parameter `decreasingValue`. Parameter ini menentukan pemilihan strategi dalam tahapan *perturbation*. Nilai parameter ini akan menjadi batas bawah dari probabilitas dalam memilih strategi penjadwalan (lihat Subbab 4.2). Semakin besar nilai `decreasingValue` maka semakin besar faktor acak dalam menentukan strategi yang digunakan. Semakin kecil nilai `decreasingValue`, maka semakin besar probabilitas untuk memilih strategi pemilihan slot waktu secara acak. Variabel ini memiliki rentang nilai 0 hingga 1. Pada percobaan ini lima nilai diuji coba yang terdiri dari 0, 0,3, 0,6, 0,9, dan 1. Sementara untuk parameter lain diterapkan nilai yang sama yakni parameter `constantFactor` akan diisi dengan nilai 0,5, `limitUpdateProbabilityNonRandomSlot` dan `limitShuffle` dengan nilai 100, dan `limitStuck` dengan nilai 10.

Tabel 4.5.1 menunjukkan hasil dari percobaan ini. Tabel menampilkan rata-rata dan standar deviasi dari waktu yang dibutuhkan untuk menemukan solusi *feasible*. Sementara pada tingkat keberhasilan menghasilkan solusi *feasible*, seluruh ujicoba pada ketiga dataset berhasil menghasilkan solusi *feasible*, sehingga tidak dicantumkan dalam tabel. Dari hasil ini, nilai `decreaseRate` = 0,3 merupakan pilihan paling optimal karena memberikan keseimbangan antara performa dan konsistensi di semua dataset. Meskipun `decreaseRate` = 0,6 menunjukkan rata-rata terendah untuk dataset Middle 10, namun variabilitasnya lebih tinggi, terutama pada Late 1. Sebaliknya, `decreaseRate` = 0,3 memiliki nilai rata-rata yang kompetitif serta standar deviasi terendah di semua dataset, yang menunjukkan hasil yang konsisten dan stabil. Nilai lainnya, yakni 0,0,9, dan 1,

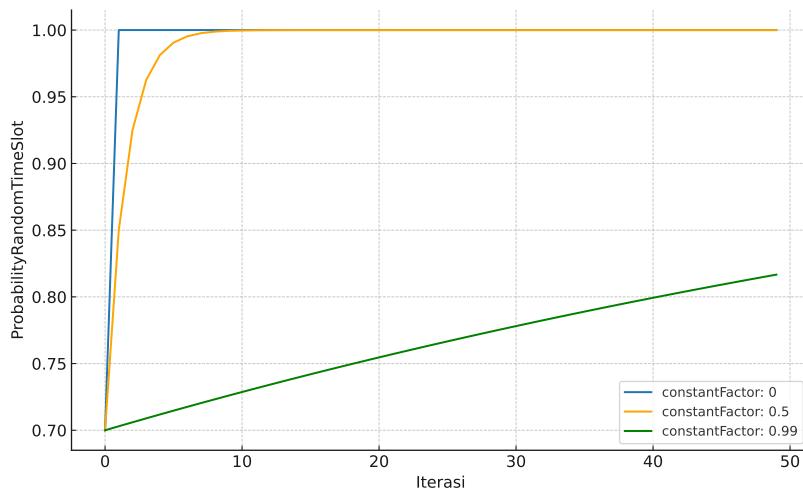
menunjukkan waktu rata-rata dan standar deviasi yang lebih tinggi. Berdasarkan hasil dan analisis tersebut, nilai 0,3 dipilih untuk digunakan pada uji coba selanjutnya.

Tabel 4.5.1: Hasil Uji Coba Nilai Parameter *decreaseRate*

Nilai	Dataset	Rata-rata	SD
0	Early 3	0,11	0,16
	Middle 10	10,10	7,03
	Late 1	7,05	2,81
0,3	Early 3	0,05	0,02
	Middle 10	6,53	3,92
	Late 1	5,69	2,83
0,6	Early 3	0,12	0,08
	Middle 10	4,31	3,32
	Late 1	5,85	4,89
0,9	Early 3	0,04	0,01
	Middle 10	9,90	5,25
	Late 1	10,52	5,67
1	Early 3	0,043	0,02
	Middle 10	7,68	6,95
	Late 1	4,63	2,56

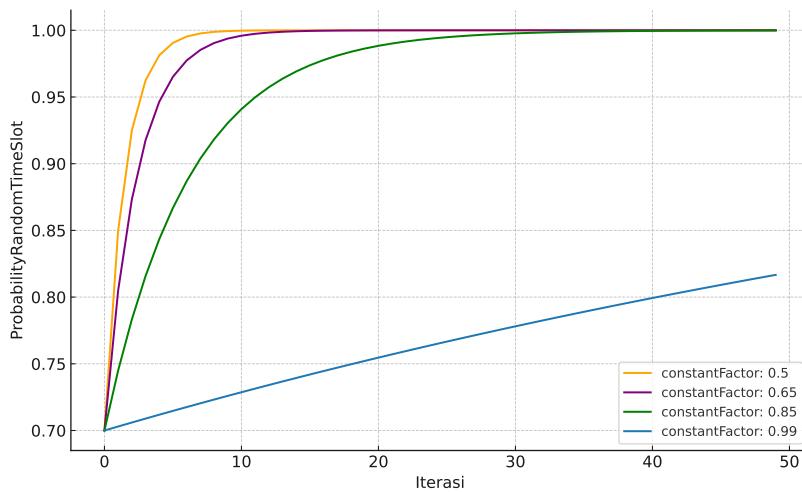
Uji coba berikutnya pada parameter `constantFactor`. Dalam pengembangan algoritma untuk mencari solusi *feasible*, probabilitas pemilihan strategi dalam tahapan *perturbation* tidaklah konstan melainkan bergerak secara perlahan menuju nilai mendekati satu. Parameter `constantFactor` yang menjadi parameter yang menentukan seberapa cepat pergerakan tersebut. Nilai parameter `constantFactor` yang lebih kecil menandakan pergerakan yang semakin cepat dan sebaliknya nilai parameter `constantFactor` yang lebih besar menandakan pergerakan yang semakin lambat. Gambar 4.5.1 menunjukkan bagaimana efek dari nilai parameter `constantFactor` terhadap probabilitas dalam pemilihan slot waktu. Berdasarkan hasil nilai `decreasingValue` 0,3, maka nilai probabilitas akan berkisar antara 0,7 sampai 1 (lihat persamaan 4.2). Penggunaan nilai `constantFactor` 0 akan menyebabkan nilai probabilitas langsung

mendekati nilai 1 pada iterasi berikutnya. Hal ini tentu tidak sesuai dengan strategi yang diharapkan karena strategi yang dikembangkan ingin melakukan perubahan nilai probabilitas secara perlahan. Sementara pada nilai 0,5, perubahan terjadi lebih perlahan dimana nilai probabilitas mendekati 1 baru diperoleh setelah beberapa iterasi. Sementara penggunaan nilai 0,99 menghasilkan pergerakan yang sangat pelan bahkan dalam ilustrasi ini setelah 50 iterasi, masih belum mencapai nilai probabilitas mendekati 1. Berdasarkan ilustrasi ini, maka uji coba pada variabel `constraintFactor` diuji coba dengan nilai 0,5, 0,65, 0,85, dan 0,99.



Gambar 4.5.1: Ilustrasi Perbandingan Nilai pada Parameter `constantFactor`

Hasil dari uji coba ditampilkan pada Tabel 4.5.2. Pada hasil ini seluruh uji coba menghasilkan solusi *feasible*. Dari segi waktu rata-rata menemukan solusi *feasible* dan konsistensinya, nilai 0,85 menunjukkan hasil yang paling optimal, sehingga nilai ini akan digunakan dalam tahapan berikutnya. Nilai 0,85 menjadi nilai yang paling baik dikarenakan nilai ini memiliki pergerakan yang berada di antara nilai 0,5 dan 0,99. Gambar 4.5.2 menunjukkan perbandingan pergerakan nilai probabilitas pada nilai `constraintFactor` yang diuji. Dari ilustrasi ini, nilai 0,85 menunjukkan pergerakan yang lebih seimbang dibandingkan nilai lainnya dimana nilai ini menghasilkan perubahan nilai probabilitas yang perlahan, namun tidak terlalu lama untuk mencapai nilai mendekati satu seperti pada nilai 0,99.



Gambar 4.5.2: Ilustrasi Perbandingan Nilai pada Parameter *constantFactor*

Tabel 4.5.2: Hasil Uji Coba Nilai Parameter *constantFactor*

Nilai decreasingRate	Dataset	Rata-rata	SD
0,50	Early 3	0,05	0,02
	Middle 10	6,53	3,92
	Late 1	5,69	2,83
0,65	Early 3	0,10	0,14
	Middle 10	7,42	5,60
	Late 1	5,52	2,99
0,85	Early 3	0,04	0,01
	Middle 10	6,58	5,34
	Late 1	5,42	3,91
0,99	Early 3	0,03	0,03
	Middle 10	10,58	7,50
	Late 1	12,31	10,39

Uji coba selanjutnya dilakukan pada tiga parameter lainnya yaitu parameter *limitUpdateProbabilityNonRandomSlot*, *limitShuffle*, dan *limitStuck*. Pada ketiga parameter ini hanya metrik tingkat keberhasilan solusi yang

dapat digunakan. Sementara waktu menemukan solusi tidak dapat digunakan sebagai perbandingan karena semakin besar nilai ketiga parameter tersebut akan menyebabkan waktu menemukan solusi yang lebih lama. Namun nilai yang lebih besar akan memberikan kesempatan proses pencarian solusi yang lebih detail dalam beberapa tahapan. Parameter `limitUpdateProbabilityNonRandomSlot` menentukan berapa batas iterasi solusi tidak mengalami peningkatan sebelum dilakukan perubahan probabilitas. Semakin besar nilai `limitUpdateProbabilityNonRandomSlot` akan memberikan kesempatan pencarian solusi yang lebih banyak pada setiap nilai probabilitas. Hal ini juga berlaku pada `limitShuffle` yang menentukan batas kegagalan pada proses pengacakan solusi dan `limitStuck` membatasi jumlah batas tidak meningkatnya solusi pada tahapan *local search*.

Uji coba pada ketiga parameter dijalankan dengan beberapa nilai. Parameter `limitUpdateProbabilityNonRandomSlot` diuji dengan nilai 100, 1000, 10000, dan 100000. Parameter `limitShuffle` diuji pada nilai 100, 1000, dan 10000. Nilai `limitStuck` diuji dengan nilai 10, 20, dan 100. Hasil uji coba menemukan seluruh uji coba menghasilkan solusi *feasible*. Pada parameter `limitUpdateProbabilityNonRandomSlot`, nilai 100000 dipilih untuk digunakan dengan pertimbangan untuk memberikan kesempatan lebih dalam proses pencarian terutama untuk permasalahan yang sulit seperti dataset Middle 2 dan Middle 3 pada *benchmark* ITC 2021. Sementara nilai `limitShuffle` dipilih menggunakan nilai 100. Hal ini dikarenakan dalam strategi yang dikembangkan, proses pengacakan solusi akan dihentikan ketika ada satu saja solusi yang berhasil diacak. Dari pengujian sangat jarang menemukan solusi yang gagal diacak dalam 100 kali percobaan. Namun dengan pertimbangan untuk memberikan kesempatan lebih ketika proses pengacakan mengalami banyak kegagalan, maka nilai tersebut dipilih untuk digunakan. Sementara pada nilai `limitStuck`, nilai 20 digunakan dengan tujuan untuk memberikan kesempatan lebih dalam tahapan *local search*. Nilai 100 tidak dipilih karena nilai ini memberikan peningkatan waktu yang signifikan dalam menghasilkan solusi *feasible*. Secara keseluruhan nilai parameter yang akan digunakan pada lima parameter untuk uji coba pada tiga *benchmark* dan satu studi kasus ditampilkan dalam Tabel 4.5.3.

Tabel 4.5.3: Daftar Nilai Parameter

Parameter	Nilai
decreasingValue	0,30
constantFactor	0,85
limitUpdateProbabilityNonRandomTimeSlot	100000
limitStuck	20
limitShuffle	100

BAB 5

PERMASALAHAN PENJADWALAN UJIAN (*BENCHMARK TORONTO*)

Toronto *Benchmark* merupakan kumpulan dataset yang mengangkat permasalahan penjadwalan ujian tanpa memperhatikan kapasitas ruangan (*uncapacited exam timetabling*). *Benchmark* ini dipublikasikan pada tahun 1996 oleh Carter et al. (1996) dengan mengembangkan dataset dari permasalahan nyata melalui sejumlah penyederhanaan. Terdapat 13 dataset dalam *benchmark* ini yang terdiri dari 3 sekolah menengah atas di Kanada, 5 universitas di Kanada, 1 universitas di Amerika, 1 universitas di Inggris, dan 1 universitas di Timur Tengah. Permasalahan utama dalam *benchmark* ini adalah menjadwalkan ujian pada slot waktu yang tersedia dengan mempertimbangkan *hard constraint* yang harus dipenuhi serta *soft constraint* yang diupayakan untuk diminimalkan, sebagaimana pada permasalahan penjadwalan umumnya. Saat ini, *benchmark* Toronto telah banyak digunakan oleh peneliti dan menjadi salah satu standar untuk menguji kemampuan dari algoritma baru yang dikembangkan terutama untuk permasalahan penjadwalan ujian.

Bab ini membahas hasil penerapan dari algoritma Progressive Acceptance Iterated Local Search (PA-ILS) dan Adaptive Threshold-Iterated Local Search (AT-ILS) untuk menyelesaikan permasalahan pada *benchmark* Toronto. Subbab 5.1 membahas karakteristik dan struktur dari dataset. Selanjutnya, Subbab 5.2 membahas batasan berupa *hard* dan *soft constraint* yang harus dipenuhi dalam permasalahan ini. Pada Subbab 5.3, hasil dari penerapan algoritma PA-ILS dalam menghasilkan *feasible* dan algoritma AT-ILS dalam mengoptimasi solusi dijabarkan beserta analisis terhadap kesesuaian pengembangan strategi dan hasil implementasinya. Subbab 5.4 membahas perbandingan hasil dari penerapan kedua algoritma terhadap penelitian terdahulu. Terakhir, Subbab 5.5

menyimpulkan hasil dari uji coba pada *benchmark* Toronto dan penilaian terhadap generalitas dari algoritma yang dikembangkan.

5.1 Karakteristik Dataset

Subbab ini membahas karakteristik dataset dari *benchmark* Toronto. Subbab 5.1.1 membahas karakteristik dataset secara umum dengan menyajikan deskripsi dari setiap dataset. Subbab 5.1.2 membahas struktur dataset pada *benchmark* Toronto.

5.1.1 Karakteristik Dataset Secara Umum

Benchmark Toronto terdiri dari 13 dataset. Dataset ini dapat diunduh melalui tautan berikut: <https://people.cs.nott.ac.uk/pszrq/data.htm>. Karakteristik dari setiap dataset ditampilkan pada Tabel 5.1.1, dengan menyajikan deskripsi dari jumlah ujian, jumlah murid atau mahasiswa, jumlah peserta ujian, dan slot waktu yang tersedia.

Dari tabel ini, dapat dilihat *benchmark* Toronto menyajikan dataset yang beragam. Beberapa dataset, seperti CAR91, CAR92, PUR 93 dan UTA92, memiliki jumlah ujian, murid atau mahasiswa, dan peserta ujian yang relatif besar, yang mencerminkan tingkat kompleksitas yang lebih tinggi dalam penyusunan jadwal. Sebaliknya, dataset seperti STA83 dan EAR83 memiliki jumlah ujian, murid atau mahasiswa, dan peserta ujian yang lebih sedikit yang mungkin menjadikan permasalahan ini lebih mudah. Sementara dataset lainnya memiliki karakteristik yang beragam dan berada pada rentang antara dataset besar dan kecil.

Tabel 5.1.1: Deskripsi Dataset pada *Benchmark* Toronto

Dataset	Ujian	Murid/Mahasiswa	Peserta Ujian	Slot
CAR91	682	16925	56242	35
CAR92	543	18419	55189	32
EAR83	189	1108	8057	24
HEC92	80	2823	10625	18
KFU93	461	5349	25113	20
LSE91	381	2726	10918	18
PUR93	2419	30029	120681	42

(Lanjutan) Deskripsi Dataset pada *Benchmark* Toronto

Dataset	Ujian	Murid/Mahasiswa	Peserta Ujian	Slot
RYE92	482	11483	45051	23
STA83	138	549	5417	13
TRE92	261	4360	14901	23
UTA92	638	21329	59144	35
UTE92	184	2750	11793	10
YOR83	180	919	66002	21

5.1.2 Struktur Dataset

Pada *benchmark* Toronto, dataset disajikan dalam struktur yang sangat sederhana berupa tabel dua dimensi. Setiap baris mewakili satu peserta ujian dan isi dari baris tersebut menunjukkan ID ujian yang diambil oleh peserta tersebut, dengan setiap ID ujian dipisahkan oleh spasi. Gambar 5.1.1 menunjukkan format dataset pada permasalahan Toronto. Pada contoh yang diberikan, terdapat 3 peserta ujian. Peserta ujian pertama mengikuti ujian dengan ID 0170, peserta ujian kedua mengikuti ujian dengan ID 0156, dan peserta ujian ketiga mengikuti ujian dengan ID 0154 dan 0156.

```
0170
0156
0154 0156
```

Gambar 5.1.1: Ilustrasi Struktur Dataset Toronto

5.2 Hard dan Soft Constraint

Batasan pada *benchmark* Toronto tergolong sederhana dengan hanya memiliki satu *hard constraint* dan satu *soft constraint*. Pada *hard constraint*, permasalahan ini tidak memperbolehkan ada peserta ujian yang dijadwalkan untuk mengikuti dua atau lebih pada waktu yang sama. Sebagai contoh, jika seorang peserta mengikuti ujian dengan ID 0001 dan 0002, kedua ujian tersebut harus dijadwalkan pada waktu yang berbeda.

Sementara itu, *soft constraint* dalam permasalahan ini bertujuan untuk memberikan jeda waktu antar ujian yang diikuti oleh peserta. Penerapan penalti untuk pelanggaran jeda

waktu dihitung berdasarkan formula berikut:

$$P = \frac{\sum_{i=1}^n \sum_{j=i+1}^n w_{|s_i - s_j|} \cdot x_{ij}}{N} \quad (5.1)$$

dengan:

- P : Total penalti rata-rata
- n : Jumlah ujian
- N : Jumlah total peserta ujian
- s_i : Slot waktu ujian ke- i
- $w_{|s_i - s_j|}$: Bobot penalti berdasarkan jeda waktu $|s_i - s_j|$, yang nilainya adalah:

$$w_{|s_i - s_j|} = \begin{cases} 16, & \text{jika } |s_i - s_j| = 0 \\ 8, & \text{jika } |s_i - s_j| = 1 \\ 4, & \text{jika } |s_i - s_j| = 2 \\ 2, & \text{jika } |s_i - s_j| = 3 \\ 1, & \text{jika } |s_i - s_j| = 4 \\ 0, & \text{jika } |s_i - s_j| > 4 \end{cases} \quad (5.2)$$

- x_{ij} : Jumlah peserta yang mengikuti kedua ujian i dan j .

Dalam formula ini, penalti dihitung dengan mengevaluasi jarak antar setiap pasangan ujian, mengalikan bobot penalti dengan jumlah peserta yang mengikuti kedua ujian tersebut, kemudian menjumlahkan seluruh nilai penalti. Nilai total penalti dibagi dengan jumlah seluruh peserta ujian untuk mendapatkan penalti rata-rata.

Sebagai ilustrasi, misalkan suatu permasalahan memiliki peserta dengan ID 1 dan 2 yang mengikuti ujian dengan ID 0001, 0002, dan 0003, yang dijadwalkan sebagai berikut:

- ujian 0001 pada slot 1
- ujian 0002 pada slot 2

- ujian 0003 pada slot 3

Maka perhitungan penalti adalah sebagai berikut:

- Untuk ujian 0001 dan 0002, karena tidak ada jeda waktu, dikenakan penalti sebesar 16. Karena terdapat dua peserta, total penalti menjadi $16 \times 2 = 32$.
- Untuk ujian 0002 dan 0003, karena tidak ada jeda waktu, dikenakan penalti sebesar 16. Karena terdapat dua peserta, total penalti menjadi $16 \times 2 = 32$.
- Untuk ujian 0001 dan 0003, karena jeda antar ujian hanya 1 slot waktu, dikenakan penalti sebesar 8. Karena terdapat dua peserta, total penalti menjadi $8 \times 2 = 16$.
- Total penalti yang diberikan dalam permasalahan ini adalah $32 + 32 + 16 = 78$. Hasil ini kemudian dibagi dengan jumlah peserta ujian, yaitu 2, sehingga nilai akhir penalti adalah $\frac{78}{2} = 39$.

5.3 Hasil Implementasi dan Analisis Algoritma

Pada *benchmark* Toronto, implementasi algoritma dimulai dengan tahap pencarian solusi *feasible* untuk setiap dataset. Setelah solusi *feasible* ditemukan, proses optimasi dijalankan dengan menggunakan solusi *feasible* tersebut sebagai input. Dalam kedua proses tersebut, tiga Low-Level Heuristic (LLH), yaitu *move*, *swap*, dan *kempe chain*, digunakan untuk menyelesaikan permasalahan ini. Bagian 5.3.1 dan 5.3.2 membahas hasil penerapan dan analisis dari tahapan mencari solusi *feasible*. Sementara bagian 5.3.3 dan 5.3.4 membahas hasil penerapan dan analisis dari tahapan optimasi.

5.3.1 Hasil Tahapan Pencarian Solusi Feasible

Pada bagian ini, algoritma pencarian solusi *feasible* diterapkan pada 13 dataset dalam *benchmark* Toronto. Hasil dari penerapan algoritma PA-ILS disajikan dalam Tabel 5.3.1. Berdasarkan hasil uji coba, algoritma PA-ILS mampu menemukan solusi *feasible* pada seluruh dataset dengan tingkat keberhasilan 100%. Hal ini berarti dari 10 percobaan yang dilakukan, algoritma berhasil menghasilkan solusi *feasible* sebanyak 10 kali.

Proses pencarian solusi menunjukkan efisiensi yang tinggi, di mana lima dataset (EAR83, HEC92, STA83, UTE92, dan YOR83) dapat diselesaikan dengan rata-rata waktu eksekusi kurang dari 1 detik. Empat dataset lainnya (KFU93, LSE91, RYE92, dan TRE92)

ditemukan dalam waktu yang relatif singkat, yakni di bawah 20 detik. Tiga dataset lainnya dengan ukuran yang lebih besar, yaitu CAR91, CAR92, dan UTA92, membutuhkan waktu yang lebih lama untuk menemukan solusi feasible, yaitu berkisar antara 12 menit hingga 38 menit. Sementara dataset PUR93 yang merupakan dataset terbesar membutuhkan waktu dengan rata-rata 24 jam untuk menemukan solusi *feasible*. Durasi yang lebih lama ini diakibatkan oleh kompleksitas dan ukuran dataset tersebut.

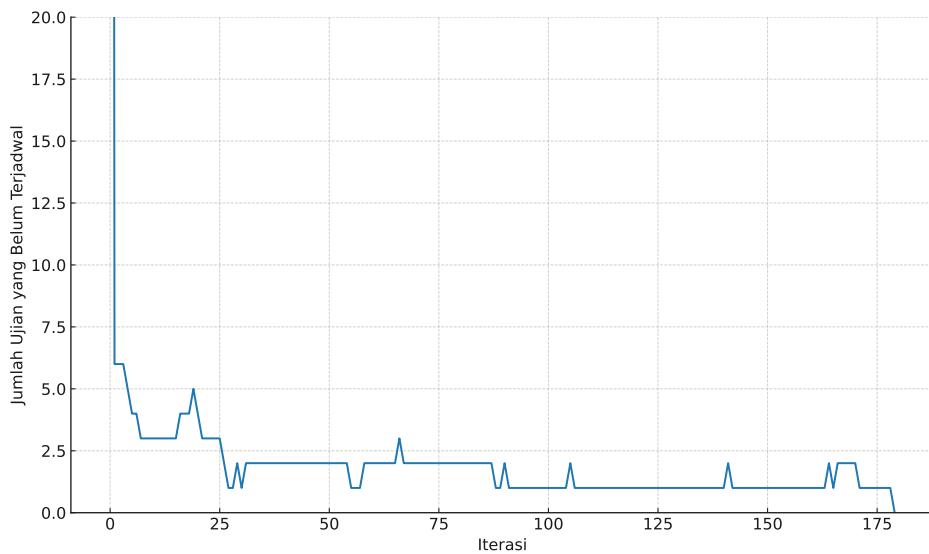
Dari hasil ini, dapat disimpulkan waktu yang dibutuhkan untuk menemukan solusi *feasible* tergolong cepat, dengan sebagian besar dataset dapat diselesaikan dalam waktu beberapa detik. Kecepatan ini sesuai dengan kebutuhan praktis dalam penjadwalan ujian, yang umumnya dilakukan beberapa hari atau minggu sebelum pelaksanaan. Selain itu, waktu pencarian solusi yang cepat memberikan fleksibilitas lebih untuk mengalokasikan waktu tambahan pada tahap optimasi solusi.

Tabel 5.3.1: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark* Toronto

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
CAR91	692,4	296,7	1162,9	100%
CAR92	2168,7	1025,5	4629,5	100%
EAR83	0,7	0,3	1,4	100%
HEC92	0,1	0,1	0,2	100%
KFU93	19,2	6,2	51,9	100%
LSE91	9,4	8,6	17,2	100%
PUR93	88434,1	59183,0	153757,0	100%
RYE92	16,5	8,6	25,1	100%
STA83	0,1	0,1	0,2	100%
TRE92	1,5	0,6	2,5	100%
UTA92	2269,3	981,1	4200,4	100%
UTE92	0,3	0,1	0,8	100%
YOR83	0,3	0,1	0,6	100%

5.3.2 Analisis Proses Algoritma PA-ILS

Untuk menggambarkan proses pencarian solusi *feasible*, Gambar 5.3.1 menampilkan visualisasi proses pencarian pada dataset EAR83. Grafik tersebut menunjukkan bahwa algoritma mampu menghindari kondisi *local optima* dengan menerima beberapa solusi yang lebih buruk selama iterasi, seperti pada iterasi ke-50 hingga ke-70 yang menerima solusi dengan dua hingga tiga ujian yang belum terjadwalkan. Meskipun demikian, algoritma tetap menunjukkan konvergensi menuju solusi yang lebih baik dengan memanfaatkan mekanisme eksplorasi, sebagaimana terlihat dari stabilisasi solusi pada iterasi ke-25 ke atas. Dengan demikian, pergerakan solusi tetap terkendali dan berkisar pada 1 hingga 3 ujian yang belum terjadwalkan.



Gambar 5.3.1: Grafik Perubahan Solusi dalam Proses Pencarian Solusi *Feasible* pada Benchmark Toronto

5.3.3 Hasil Tahapan Optimasi

Pada bagian ini, algoritma optimasi diterapkan pada 13 dataset, dengan menggunakan input dari solusi *feasible* yang ditemukan pada tahapan sebelumnya. Solusi *feasible* yang digunakan sebagai input awal dipilih secara acak untuk mengevaluasi konsistensi algoritma dalam mengoptimasi solusi dari kondisi awal yang beragam. Proses optimasi terdiri dari dua tahapan. Pada tahapan pertama, uji coba dilakukan dengan menerapkan algoritma optimasi sebanyak 10 kali pada setiap dataset, dengan masing-masing uji coba berlangsung selama 10 jam. Tahapan ini ditujukan untuk melihat konsistensi dari algoritma dalam

mengoptimasi solusi pada batasan waktu tertentu. Selanjutnya, tahapan kedua bertujuan untuk menghasilkan solusi terbaik yang digunakan sebagai perbandingan dengan penelitian lainnya. Tahapan ini akan mengoptimasi kembali solusi terbaik yang dihasilkan pada tahapan pertama selama 10 jam. Jika ditemukan solusi yang lebih baik, maka proses ini diulang. Namun, jika tidak ada peningkatan, maka proses optimasi dihentikan.

Hasil optimasi disajikan dalam Tabel 5.3.2. Untuk uji coba tahapan pertama, tabel menampilkan rata-rata penalti, solusi terbaik, solusi terburuk, serta Koefisien Variasi (KV). Sementara untuk tahapan kedua, tabel menampilkan solusi terbaik dan juga waktu yang dibutuhkan untuk mencapainya. Dari hasil tersebut, sebagian besar dataset menunjukkan hasil yang sangat konsisten, dengan sembilan dataset (CAR92, EAR83, KFU93, PUR93, RYE92, STA83, TRE92, UTE92, YOR83) memiliki nilai KV di bawah 1%. Tiga dataset lainnya, yaitu CAR91 dan LSE91, memiliki tingkat konsistensi yang cukup baik dengan nilai KV antara 1-2%. Satu-satunya dataset dengan tingkat konsistensi yang lebih buruk adalah HEC92, yang memiliki nilai KV sebesar 6,11

Gambar 5.3.2 menyajikan grafik box plot untuk tiga dataset, yaitu EAR83 (nilai KV di bawah 1%), CAR91 (nilai KV antara 1-2%), dan HEC92 (nilai KV di atas 2%). Dari grafik tersebut, terlihat bahwa algoritma menghasilkan hasil yang relatif konsisten, meskipun terdapat beberapa outlier pada dataset CAR91 dan HEC92. Pada dataset HEC92, outlier tersebut cukup jauh dari persebaran data. Jika outlier pada dataset HEC92 dihilangkan, nilai KV dapat meningkat secara signifikan menjadi 0,46%, yang menunjukkan tingkat konsistensi yang sangat baik.

Sementara itu, dalam aspek menghasilkan solusi terbaik, menjalankan algoritma dalam waktu 10 jam mampu menghasilkan solusi terbaik pada beberapa dataset, seperti HEC92, RYE92, STA83, TRE92, UTE92, dan YOR83. Hal ini ditunjukkan dengan tidak adanya peningkatan solusi ketika tahapan kedua dijalankan. Sementara pada beberapa dataset lainnya membutuhkan waktu optimasi yang lebih lama. Dataset KFU93 dan LSE91 memerlukan waktu optimasi selama 20 jam untuk mendapatkan hasil terbaik. Dataset EAR83 membutuhkan waktu lebih lama dibandingkan dataset lainnya, yakni hingga 30 jam untuk menemukan solusi terbaik, meskipun ukuran dataset ini tidak berbeda signifikan dengan dataset lain yang selesai dalam 10-20 jam. Dataset CAR91, CAR92, dan UTA92 memerlukan waktu hingga 50 jam untuk mendapatkan solusi terbaik, yang disebabkan

oleh ukuran dataset yang lebih besar dibandingkan dataset lainnya. Sementara untuk dataset PUR93 yang merupakan dataset terbesar membutuhkan waktu yang jauh lebih lama dengan membutuhkan waktu hingga 100 jam.

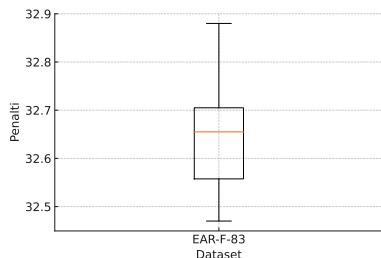
Tabel 5.3.2: Hasil Uji Coba Tahapan Optimasi pada *Benchmark* Toronto

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
CAR91	4,82	4,72	4,99	1,45	4,48	50 jam
CAR92	4,03	3,98	4,09	0,74	3,87	50 jam
EAR83	32,65	32,47	32,88	0,34	32,40	30 jam
HEC92	10,32	10,05	12,12	6,11	10,05	10 jam
KFU93	13,08	12,97	13,23	0,69	12,94	20 jam
LSE91	10,10	9,89	10,30	1,29	9,85	20 jam
PUR93	6,45	6,31	6,54	0,98	4,56	100 jam
RYE92	8,29	8,19	8,41	0,84	8,19	10 jam
STA83	157,03	157,03	157,03	0,00	157,03	10 jam
TRE92	7,91	7,81	7,98	0,63	8,19	10 jam
UTA92	3,36	3,29	3,44	1,19	8,17	50 jam
UTE92	24,82	24,77	24,84	0,08	24,77	10 jam
YOR83	34,93	34,59	35,17	0,46	34,59	10 jam

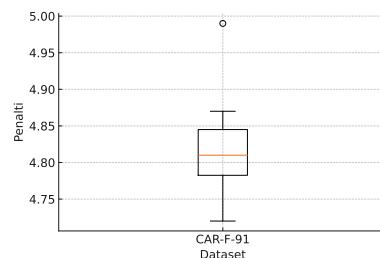
5.3.4 Analisis Proses Algoritma AT-ILS

Untuk memahami bagaimana algoritma AT-ILS bekerja dalam mengoptimasi solusi, Gambar 5.3.3 dan 5.3.4 menyajikan visualisasi dari proses optimasi. Gambar 5.3.3 memvisualisasikan pergerakan solusi pada setiap iterasi. Dari gambar ini, terlihat bahwa algoritma AT-ILS memungkinkan penerimaan solusi yang lebih buruk. Namun, jika dalam beberapa iterasi berikutnya tidak ditemukan solusi yang lebih baik, solusi akan dikembalikan ke kondisi solusi terbaik. Strategi ini menunjukkan bagaimana algoritma AT-ILS mampu mempertahankan konvergensi sambil tetap melakukan eksplorasi untuk menghindari kondisi *local optima*.

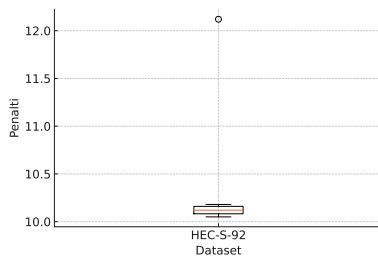
Sementara itu, Gambar 5.3.4 menampilkan proses pergerakan solusi pada tahapan *local search*. Garis biru mengindikasikan perubahan nilai solusi, sedangkan garis kuning



(a) EAR83



(b) CAR91



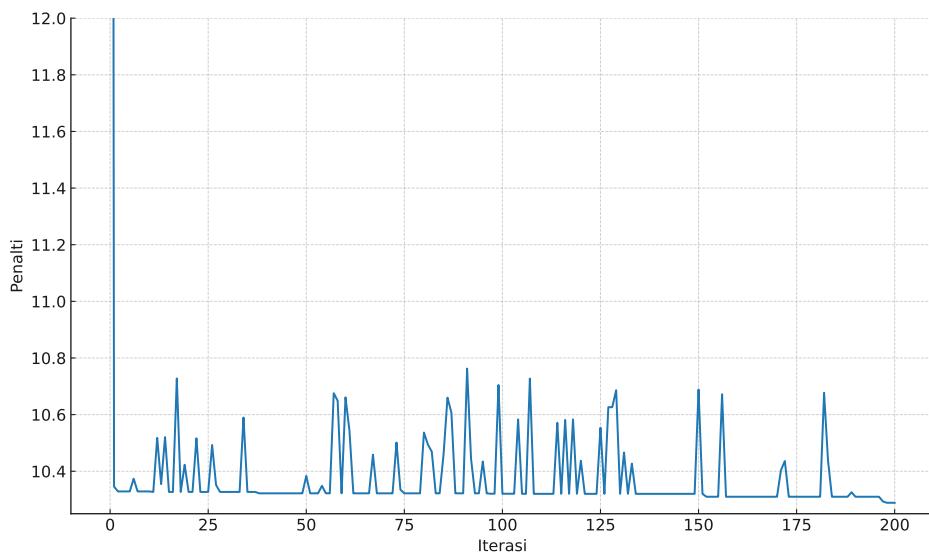
(c) HEC92

Gambar 5.3.2: Box plot Hasil Algoritma Optimasi pada Tiga Dataset: EAR83, CAR91, dan HEC92

merepresentasikan nilai ambang batas. Dari gambar ini, dapat dilihat bahwa nilai ambang batas berperan dalam menurunkan nilai penalti solusi sehingga pencarian dapat mengarah pada konvergensi. Sementara itu, nilai ambang batas dapat dinaikkan ketika tidak ada pergerakan solusi setelah LLH diterapkan pada seluruh ujian yang belum terjadwalkan. Hal ini bertujuan untuk mencegah algoritma terjebak dalam kondisi *local optima*. Sementara itu, pergerakan solusi yang naik turun menunjukkan strategi algoritma dalam menjaga keseimbangan antara proses eksplorasi dan eksloitasi, dengan harapan dapat mengarahkan algoritma untuk menghasilkan solusi yang lebih baik.

5.4 Perbandingan dengan Studi Sebelumnya

Pada bagian ini menyajikan perbandingan hasil implementasi pada *benchmark* Toronto dengan hasil dari 27 penelitian sebelumnya. Perbandingan ini dilakukan untuk dapat mengevaluasi performa dari algoritma yang telah dikembangkan terhadap penelitian sebelumnya. Perbandingan dilakukan berdasarkan nilai dari solusi terbaik yang dihasilkan dari setiap studi. Hal ini dilakukan mengingat sebagian besar penelitian sebelumnya hanya menyajikan nilai dari solusi terbaik yang berhasil diperoleh. Proses perbandingan ini dilakukan dengan memberikan peringkat pada setiap dataset, kemudian menghitung



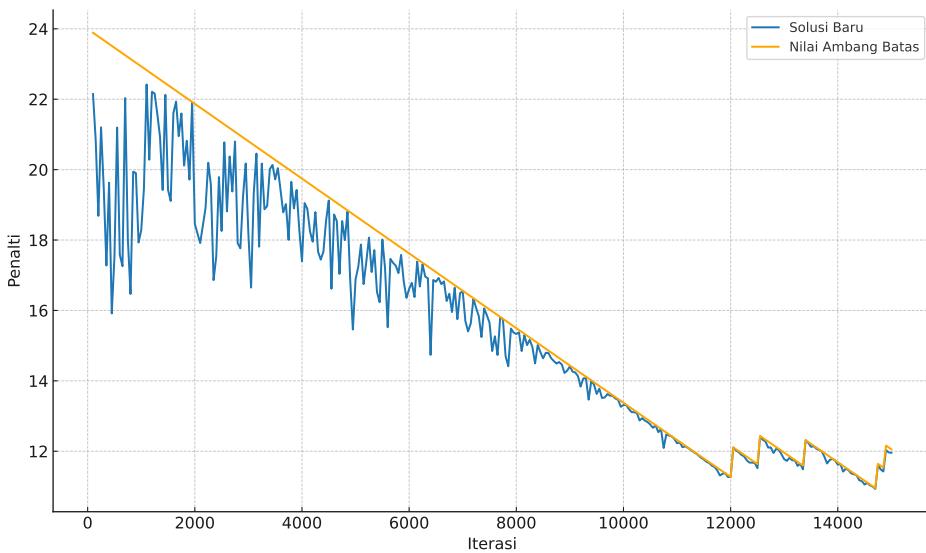
Gambar 5.3.3: Grafik Perubahan Solusi dalam Proses Optimasi pada *Benchmark* Toronto

rata-rata peringkat sebagai hasil akhir, seperti yang telah dijelaskan pada Subbab 3.5.

Hasil perbandingan disajikan pada Gambar 5.4, dengan detail perbandingan terhadap lima penelitian terbaik sebelumnya ditampilkan pada Tabel 5.4.1, serta hasil pemeringkatan dan solusi terbaik dari setiap studi pada Lampiran A. Berdasarkan hasil perbandingan, algoritma yang dikembangkan menunjukkan performa yang kompetitif, meraih peringkat keempat, mengungguli 24 penelitian lainnya. Selain itu, algoritma yang dikembangkan dalam penelitian ini berhasil menghasilkan solusi terbaik baru pada dataset Ear-f-83-I, yang sebelumnya dihasilkan oleh penelitian Bellio et al. (2021).

Hasil perbandingan ini juga menunjukkan bahwa algoritma yang dikembangkan memiliki tingkat konsistensi yang baik. Hal ini terlihat dari hasil peringkat untuk setiap dataset yang relatif stabil, tanpa adanya hasil peringkat yang memiliki perbedaan signifikan. Dari Lampiran A, dapat dilihat bahwa peringkat dari hasil penelitian berada pada rentang 4 sampai 5,5 untuk tujuh dataset yang menunjukkan konsistensi yang tinggi. Pada lima dataset lainnya memiliki peringkat yang tidak terlalu jauh, 3 dataset dengan peringkat 1 sampai 3 dan 2 dataset yaitu Car-f-92-I dan Uta-s-92-I dengan peringkat 7 dan 10.

Dari hasil perbandingan ini mengindikasikan performa yang menjanjikan, terutama pada algoritma AT-ILS. Algoritma optimasi yang dikembangkan mampu mengungguli 24 penelitian lainnya. Hal ini menunjukkan tingkat generalitas yang baik pada tingkat



Gambar 5.3.4: Grafik Perubahan Solusi dalam Proses *Local Search* pada *Benchmark* Toronto

pertama, yaitu antar dataset. Selain itu algoritma dikembangkan tanpa memerlukan pengaturan nilai parameter dan dalam pengujian ini menggunakan LLH yang tidak mengkhusus pada permasalahan dalam *benchmark* Toronto. Hal ini menjadikan kelebihan yang menunjukkan generalitas dari algoritma yang dikembangkan.

Sementara itu, pada algoritma PA-ILS, hasil uji coba menunjukkan bahwa algoritma ini bekerja dengan baik dengan menemukan seluruh solusi *feasible* dan dengan tingkat keberhasilan 100%. Namun hasil ini belum bisa membuktikan kemampuan dari algoritma PA-ILS, karena permasalahan pada *benchmark* Toronto tidak menyajikan permasalahan yang sulit dalam menghasilkan solusi *feasible*. Hal ini dibuktikan dengan seluruh penelitian sebelumnya mampu menemukan solusi *feasible* pada hampir seluruh dataset.

Tabel 5.4.1: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark* Toronto

<i>Dataset</i>	Penelitian ini	Bellio et al. (2021)	Leite et al. (2018)	Burke and Bykov (2016)	Demeester et al. (2012)	Mandal et al. (2020)
CAR91	4,48	4,24	4,31	4,32	4,52	4,58
CAR92	3,87	3,64	3,68	3,67	3,78	3,82
EAR83	32,40	32,42	32,48	32,62	32,49	32,48

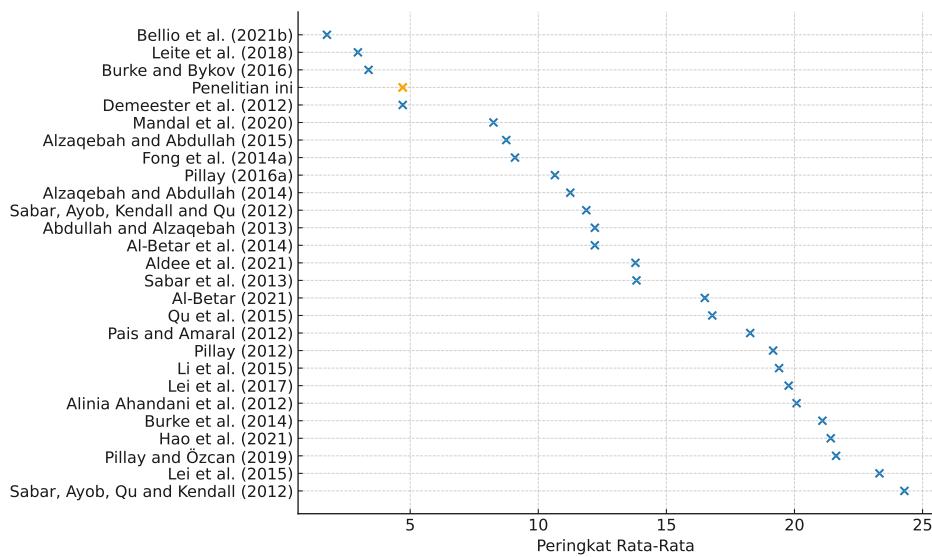
(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark* Toronto

<i>Dataset</i>	Penelitian ini	Bellio et al. (2021)	Leite et al. (2018)	Burke and Bykov (2016)	Demeester et al. (2012)	Mandal et al. (2020)
HEC92	10,05	10,03	10,03	10,06	10,03	10,32
KFU93	12,93	12,81	12,81	12,80	12,90	13,34
LSE91	9,85	9,78	9,78	9,78	10,04	10,24
PUR93	4,56	4,22	4,14	3,88	5,00	-
RYE92	8,19	7,84	7,89	7,91	8,05	9,79
STA83	157,03	157,03	157,03	157,03	157,03	157,03
TRE92	7,69	7,59	7,66	7,64	7,69	7,72
UTA92	3,17	2,95	3,01	2,98	3,13	3,13
UTE92	24,77	24,76	24,80	24,78	24,77	25,28
YOR83	34,59	34,40	34,45	34,71	34,64	35,46
Rata-Rata Peringkat	4,76	1,88	2,92	3,23	5,23	8,70

5.5 Kesimpulan

Pada bab ini, algoritma yang dikembangkan dalam penelitian ini diterapkan pada permasalahan penjadwalan ujian menggunakan *benchmark* Toronto. Permasalahan pada *benchmark* Toronto merupakan permasalahan klasik dengan satu *hard constraint* dan satu *soft constraint*, yang membuatnya relatif sederhana dibandingkan dengan permasalahan penjadwalan lainnya. *Benchmark* ini merupakan salah satu *benchmark* yang paling umum digunakan untuk menguji algoritma dalam permasalahan penjadwalan.

Penerapan algoritma pencarian solusi *feasible* menunjukkan hasil yang baik dengan seluruh dataset berhasil ditemukan solusi *feasible* dengan tingkat keberhasilan 100%. Sementara itu, pada algoritma optimasi, hasil uji coba menunjukkan performa yang sangat baik dengan meraih peringkat keempat dari 27 penelitian terdahulu. Selain itu, algoritma ini juga mampu menghasilkan solusi terbaik baru pada dataset EAR83.



Gambar 5.4.1: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* Toronto

Hasil menunjukkan bahwa pada tingkat generalitas pertama, algoritma yang dikembangkan mampu bekerja dengan baik, khususnya dalam melakukan optimasi. Ketiadaan kebutuhan untuk mengatur parameter tambahan turut memperkuat generalitas algoritma ini. Temuan tersebut mengindikasikan potensi yang menjanjikan bagi algoritma yang telah dikembangkan. Untuk mengevaluasi performa algoritma pada tingkat generalitas yang lebih tinggi, pengujian selanjutnya dilakukan pada permasalahan penjadwalan mata kuliah menggunakan *benchmark* International Timetabling Competition 2019, yang dibahas pada Bab 6.

BAB 6

PERMASALAHAN PENJADWALAN MATA KULIAH (BENCHMARK INTERNATIONAL TIMETABLING COMPETITION 2019)

Benchmark International Timetabling Competition (ITC) 2019 merupakan kompetisi yang berfokus pada permasalahan penjadwalan mata kuliah. Kompetisi ini menghadirkan kumpulan dataset terbaru yang diambil dari permasalahan nyata di berbagai universitas dan negara, seperti Purdue University di Amerika Serikat, Masaryk University di Republik Ceko, AGH University of Science and Technology di Polandia, dan Istanbul Kultur University di Turki. Pengumpulan dataset dari berbagai universitas dan negara bertujuan untuk menciptakan kumpulan data yang beragam dengan karakteristik yang bervariasi.

Pada permasalahan dalam *benchmark* ITC 2019, terdapat dua hal utama yang perlu dijadwalkan yaitu kelas dan mahasiswa. Selain itu, terdapat dua tujuan dalam *benchmark* ITC 2019 yaitu menghasilkan solusi *feasible* dengan tanpa melanggar *hard constraint* yang ada, serta meminimalkan jumlah pelanggaran *soft constraint*. Penjadwalan kelas dalam kompetisi ini cukup kompleks karena setiap kelas memiliki jumlah ruangan dan slot waktu yang sangat terbatas. Selain itu, terdapat 19 jenis batasan distribusi yang mungkin harus dipenuhi oleh setiap kelas. Di sisi lain, penjadwalan mahasiswa relatif lebih sederhana, karena hanya perlu disesuaikan dengan struktur mata kuliah yang diambil masing-masing mahasiswa.

Bab ini membahas hasil penerapan algoritma Progressive Acceptance Iterated Local Search (PA-ILS) dan Adaptive Threshold-Iterated Local Search (AT-ILS) untuk menyelesaikan permasalahan dalam *benchmark* ITC 2019. Pembahasan dimulai pada Subbab 6.1, dengan menyajikan karakteristik dari 30 dataset dan format penulisan pada

setiap dataset. Selanjutnya pada Subbab 6.2, membahas *hard* dan *soft constraint* dari permasalahan dalam *benchmark* ITC 2019. Subbab 6.3 membahas hasil implementasi dan analisis dari penerapan algoritma PA-ILS dan AT-ILS. Pada Subbab 6.4, hasil implementasi pada kedua algoritma dibandingkan dengan penelitian terdahulu untuk mengevaluasi kinerja algoritma. Subbab terakhir yaitu Subbab 6.5 memberikan kesimpulan dari hasil uji coba pada *benchmark* ITC 2019.

6.1 Karakteristik Dataset

6.1.1 Karakteristik Dataset Secara Umum

Dalam permasalahan pada *benchmark* ITC 2019, terdapat 30 dataset yang dipublikasikan dalam tiga kategori, yaitu Early, Middle, dan Late. Setiap kategori berisi 10 dataset, dengan perbedaan utama terletak pada waktu publikasinya. Kategori Early dipublikasikan paling awal, diikuti dengan kategori Middle dan kategori Late dipublikasikan paling akhir. Seluruh dataset dapat diunduh melalui situs web resmi <https://www.itc2019.org/instances/all>. Karakteristik dari setiap dataset ditampilkan pada Tabel 6.1.1, 6.1.2, dan 6.1.3 dengan mencakup jumlah kelas, ruangan, murid, serta distribusi *hard* dan *soft constraint*.

Pada aspek jumlah kelas yang harus dijadwalkan, terdapat variasi yang signifikan diantara ke-30 dataset. Sebagai contoh, dataset yach-fal17 yang merupakan dataset dengan jumlah kelas terkecil, memiliki 417 kelas yang harus dijadwalkan. Sementara pada dataset pu-proj-fal19, yang merupakan dataset terbesar, memiliki 8.813 kelas yang harus dijadwalkan.

Sementara itu, jumlah ruangan pada setiap dataset juga sangat bervariasi. Namun jumlah ruangan tidak selalu sebanding dengan jumlah kelas. Sebagai contoh, dataset muni-fi-spr17 memiliki lebih banyak kelas dibandingkan lums-spr18, yaitu 516 berbanding 487, namun hanya menyediakan 35 ruangan, jauh lebih sedikit dibandingkan dengan jumlah ruangan pada dataset lums-spr18 yaitu sebanyak 73 ruangan. Ketidakseimbangan antara jumlah kelas yang harus dijadwalkan dan ketersediaan ruangan ini menunjukkan permasalahan yang lebih menantang seperti pada dataset muni-fi-spr17 karena keterbatasan sumber daya yang tersedia untuk memenuhi kebutuhan penjadwalan kelas.

Dari segi jumlah mahasiswa, variasi antar dataset juga cukup besar. Beberapa dataset, seperti tg-fal17 dan lums-spr18, tidak melibatkan penjadwalan mahasiswa. Sementara itu, dataset lain menunjukkan perbedaan jumlah mahasiswa yang signifikan, mulai dari dataset dengan jumlah mahasiswa paling sedikit, yaitu 395 pada dataset muni-fsp-spr17c, hingga pada dataset dengan jumlah mahasiswa terbanyak, yaitu 38.437 pada dataset pu-proj-fal19.

Perbedaan terakhir yaitu pada jumlah batasan distribusi. *Benchmark ITC 2019* menghardirkan dataset dengan variasi jumlah batasan distribusi antar dataset dalam bentuk *hard* dan *soft constraint*. Dataset pada *benchmark ITC 2019* umumnya memiliki lebih banyak batasan distribusi berupa *hard constraint* dibandingkan dengan *soft constraint*. Hal ini yang mengindikasikan tantangan besar dalam menghasilkan solusi *feasible*. Selain itu, proses optimasi juga akan menjadi cukup kompleks, karena solusi yang tidak *feasible* akan lebih sering ditemukan dalam proses optimasi.

Tabel 6.1.1: Deskripsi Dataset pada *Benchmark ITC 2019* Kategori Early

Dataset	Kelas	Ruang	Mahasiswa	Distribusi	
				<i>Hard Constraint</i>	<i>Soft Constraint</i>
agh-fis-spr17	1239	80	1641	820	400
agh-ggis-spr17	1859	44	2116	2202	488
bet-fal17	983	62	3018	861	390
iku-fal17	2641	214	0	2237	665
mary-spr17	882	90	3666	3151	796
muni-fi-spr16	575	35	1543	645	95
muni-fsp-spr17	561	44	865	331	69
muni-pdf-spr16c	2526	70	2938	1456	570
pu-llr-spr17	1001	75	27018	416	218
tg-fal17	711	15	0	459	42

Tabel 6.1.2: Deskripsi Dataset pada *Benchmark ITC 2019* Kategori Middle

Dataset	Kelas	Ruang	Mahasiswa	Distribusi	
				<i>Hard Constraint</i>	<i>Soft Constraint</i>
agh-ggos-spr17	1144	84	2254	1181	507

(Lanjutan) Deskripsi Dataset pada *Benchmark* ITC 2019 Kategori Middle

Dataset	Kelas	Ruang	Mahasiswa	Distribusi	
				<i>Hard Constraint</i>	<i>Soft Constraint</i>
agh-h-spr17	460	39	1988	288	111
lums-spr18	487	73	0	449	69
muni-fi-spr17	516	35	1469	639	60
muni-fspss-spr17c	650	29	395	562	147
muni-pdf-spr16	1515	83	3443	579	433
nbi-spr18	782	67	2293	585	11
pu-d5-spr17	1061	84	13497	1262	273
pu-proj-fal19	8813	768	38437	6399	1398
yach-fal17	417	28	821	529	116

Tabel 6.1.3: Deskripsi Dataset pada *Benchmark* ITC 2019 Kategori Late

Dataset	Kelas	Ruang	Mahasiswa	Distribusi	
				<i>Hard Constraint</i>	<i>Soft Constraint</i>
agh-fal17	5081	327	6925	4836	2318
bet-spr18	1083	63	2921	1004	414
iku-spr18	2782	208	0	2833	655
lums-fal17	502	73	0	521	76
mary-fal18	951	93	5051	349	164
muni-fi-fal17	535	36	1685	635	152
muni-fspssx-fal17	1623	33	1152	1070	289
muni-pdfx-fal17	3717	86	5651	2433	1068
pu-d9-fal19	2798	224	35213	2039	707
tg-spr18	676	18	0	376	50

6.1.2 Struktur Dataset

ITC 2019 menyajikan datasetnya dalam struktur XML. Setiap dataset dimulai dengan elemen `Problem` yang menyajikan nama dataset, jumlah hari dalam seminggu yang dapat digunakan, jumlah slot waktu dalam sehari, dan jumlah minggu dalam permasalahan. Di

dalam elemen `Problem` terdapat sub-elemen `Optimization`. Elemen ini menjelaskan bobot pelanggaran dari segi waktu, ruangan, distribusi, dan mahasiswa. Setiap dataset memiliki bobot yang bervariasi. Gambar 6.1.1 menunjukkan ilustrasi format kedua elemen ini. Pada contoh, disajikan dataset bernama "tg-fal17" dengan jumlah hari yang dapat digunakan dalam seminggu sebanyak 7, jumlah slot waktu dalam sehari sebanyak 288 slot waktu, dan penjadwalan berlangsung selama 14 minggu. Dalam contoh ini, bobot penalti untuk pelanggaran slot waktu adalah 2, ruangan 1, distribusi 20, dan mahasiswa 0. Bobot penalti ini digunakan terhadap pelanggaran *soft constraint*. Sebagai ilustrasi, jika terdapat 2 pelanggaran slot waktu, 10 pelanggaran ruangan, dan 2 pelanggaran distribusi, maka total penalti pada dataset ini adalah $2 \times 2 + 10 \times 1 + 2 \times 20 = 54$.

```
<problem name="tg-fal17" nrDays="7" slotsPerDay="288" nrWeeks="14">
  <optimization time="2" room="1" distribution="20" student="0"/>
```

Gambar 6.1.1: Ilustrasi Elemen Problem dan Optimization pada Dataset ITC 2019

Sub-elemen berikutnya yang terdapat dalam elemen `Problem` adalah sub-elemen `Room`. Sub-elemen ini menjelaskan ruangan-ruangan yang terdapat dalam permasalahan ini. Setiap ruangan dapat memiliki data jarak waktu ke ruangan lainnya serta data waktu ketidaktersediaan ruangan. Gambar 6.1.2 menampilkan ilustrasi format data ruangan. Pada ilustrasi ini, ruangan dengan id 1 memiliki jarak sebesar 1 slot waktu untuk menuju ke ruangan 20 hingga 23. Selain itu, ruangan ini tidak tersedia pada hari ke-3 mulai dari slot 192 hingga 18 slot setelahnya (baru dapat digunakan kembali mulai dari slot 210) dan ketidaktersediaan ini hanya terjadi pada minggu ke-5.

```
<room id="1" capacity="40">
  <travel room="20" value="1"/>
  <travel room="21" value="1"/>
  <travel room="22" value="1"/>
  <travel room="23" value="1"/>
  <unavailable days="0010000" start="192" length="18"
    weeks="0000100000000000"/>
</room>
```

Gambar 6.1.2: Ilustrasi Elemen Room pada Dataset ITC 2019

Sub-elemen berikutnya adalah `Course` dan `Distribution`. Sub-elemen `Course` berisi konfigurasi mata kuliah serta daftar kelas yang tersedia. Sedangkan sub-elemen `Distribution` menjabarkan batasan distribusi yang ada dalam permasalahan ini.

Karena kedua elemen ini berkaitan dengan *hard* dan *soft constraint*, penjelasan lebih detail akan dibahas pada Subbab 6.2.

Sub-elemen terakhir adalah `Students`. Sub-elemen ini menjabarkan mahasiswa yang harus dijadwalkan dalam permasalahan ini beserta mata kuliah yang mereka ambil. Gambar 6.1.3 menunjukkan ilustrasi dari data pada elemen `Students`. Pada contoh ini, mahasiswa dengan id 1 mengambil 3 mata kuliah, yaitu mata kuliah dengan id 17, 165, dan 24.

```
<student id="1">
  <course id="17"/>
  <course id="165"/>
  <course id="24"/>
</student>
```

Gambar 6.1.3: Ilustrasi Elemen `Students` pada Dataset ITC 2019

6.2 Hard dan Soft Constraint

Pada permasalahan dalam *benchmark* ITC 2019, terdapat lima jenis batasan utama: batasan slot waktu, ruangan, mata kuliah, mahasiswa, dan distribusi. Batasan mata kuliah berfungsi sebagai *hard constraint* dalam penjadwalan mahasiswa. Sementara itu, batasan lainnya dapat berfungsi sebagai *hard* atau *soft constraint*, bergantung pada karakteristik dari setiap dataset.

6.2.1 Batasan Slot Waktu

Pada permasalahan dalam *benchmark* ITC 2019, setiap kelas memiliki daftar slot waktu yang dapat digunakan untuk penjadwalan. Selain itu, setiap slot waktu juga dapat memiliki nilai penalti jika digunakan, yang menunjukkan preferensi untuk menghindari slot tersebut. Gambar 6.2.1 menunjukkan ilustrasi dari batasan slot waktu. Pada kelas dengan ID 730, hanya terdapat 7 slot waktu yang memungkinkan untuk digunakan dalam penjadwalan kelas tersebut. Sebagai contoh, jika kelas ini dijadwalkan pada slot waktu pertama, maka akan dikenakan penalti sebesar 5.

6.2.2 Batasan Ruangan

Batasan serupa dengan batasan slot waktu juga terdapat pada ruangan. Setiap kelas memiliki daftar ruangan yang dapat digunakan. Selain itu setiap ruangan memungkinkan

```

<class id="730" limit="48">
    <time days="0000010" start="96" length="27"
        weeks="00010000" penalty="5"/>
    <time days="0000001" start="96" length="27"
        weeks="00010000" penalty="0"/>
    <time days="0000010" start="106" length="27"
        weeks="00010000" penalty="0"/>
    <time days="0000001" start="106" length="27"
        weeks="00010000" penalty="0"/>
    <time days="0000010" start="116" length="27"
        weeks="00010000" penalty="0"/>
    <time days="0000001" start="116" length="27"
        weeks="00010000" penalty="0"/>
    <time days="0000010" start="126" length="27"
        weeks="00010000" penalty="0"/>
</class>

```

Gambar 6.2.1: Ilustrasi Batasan Waktu pada Dataset ITC 2019

untuk memiliki penalti jika digunakan. Hal ini menunjukkan preferensi tertentu terhadap ruangan tersebut. Gambar 6.2.2 mengilustrasikan batasan ruangan. Pada ilustrasi ini, untuk kelas dengan ID 730, hanya dapat dijadwalkan pada tiga ruangan yaitu ruangan dengan id 11, 23 dan 24. Jika memilih menjadwalkan pada ruangan dengan ID 11 atau 23, maka akan dikenakan penalti sebesar 4.

```

<class id="730" limit="48">
    <room id="11" penalty="4"/>
    <room id="23" penalty="4"/>
    <room id="24" penalty="0"/>
</class>

```

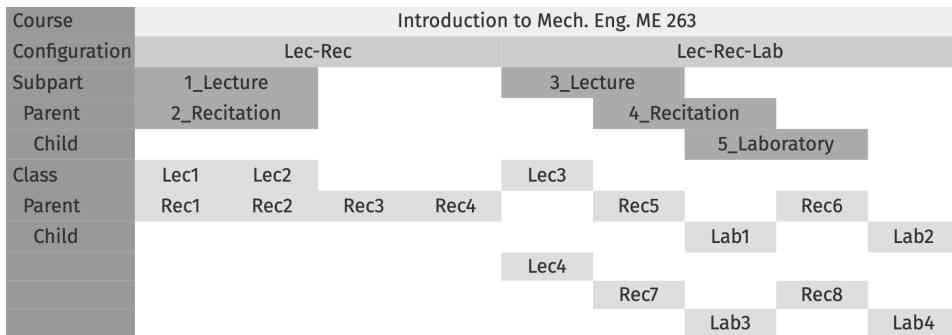
Gambar 6.2.2: Ilustrasi Batasan Ruangan pada Dataset ITC 2019

6.2.3 Batasan Mata Kuliah

Batasan mata kuliah mengatur kombinasi kelas apa saja yang dapat diambil pada setiap mata kuliah. Batasan ini berfungsi saat proses penjadwalan mahasiswa. Gambar 6.2.3 menunjukkan struktur dari mata kuliah pada permasalahan ini. Suatu mata kuliah memiliki satu atau lebih konfigurasi dimana setiap mahasiswa akan dijadwalkan pada salah satu konfigurasi. Setiap konfigurasi akan memiliki satu atau lebih *subpart*. Pada setiap *subpart* akan memiliki satu atau lebih kelas. Setiap mahasiswa harus didaftarkan tepat pada satu kelas pada setiap *subpartnya*. Selain itu setiap kelas memungkinkan untuk memiliki kelas *parent*. Artinya jika kita menjadwalkan mahasiswa yang memiliki kelas

parent, maka mahasiswa tersebut harus juga dijadwalkan pada kelas *parent*-nya.

Sebagai ilustrasi, gambar 6.2.4 menunjukkan contoh dari struktur mata kuliah ME-263. Mata kuliah ini memiliki dua konfigurasi. Konfigurasi pertama memiliki dua *subpart*, sedangkan pada konfigurasi kedua memiliki 3 *subpart*. Sebagai contoh jika menjadwalkan mahasiswa pada mata kuliah ini, mahasiswa bisa dijadwalkan pada konfigurasi satu atau dua. Jika memilih konfigurasi satu, maka mahasiswa bisa dijadwalkan pada salah satu kelas dalam *subpart* "1_Lecture" dan salah satu kelas pada *subpart* "2_Recitation". Jika mahasiswa dijadwalkan pada konfigurasi dua, maka mahasiswa bisa dijadwalkan pada salah satu kelas dalam *subpart* "3_Lecture", salah satu kelas pada *subpart* "4_Recitation" dan salah satu kelas pada *subpart* "5_Laboratory". Namun perlu diperhatikan bahwa kelas dalam *subpart* "3_Lecture" merupakan kelas parent dari kelas pada *subpart* "4_Recitation", sehingga saat pemilihan kelas pada *subpart* "4_Recitation" bergantung pada kelas yang dipilih dalam *subpart* "3_Lecture". Sebagai contoh jika pada *subpart* "3_Lecture" memilih kelas "Lec3" maka pada *subpart* "4_Recitation" harus memilih antara kelas "Rec5" atau "Rec6". Hal ini juga berlaku pada *subpart* berikutnya yaitu *subpart* "5_Laboratory". Jika pada *subpart* "4_Recitation" memilih kelas "Rec5" maka pada *subpart* "5_Laboratory" harus memilih kelas "Lab1" karena kelas parent dari "Lab1" adalah "Rec5".



Gambar 6.2.3: Struktur Mata Kuliah pada Permasalahan ITC 2019

6.2.4 Batasan Mahasiswa

Pada batasan mahasiswa, *hard constraint* mengharuskan mahasiswa dijadwalkan sesuai dengan struktur mata kuliah yang diambil, serta tidak melanggar kapasitas ruangan. Sementara itu, pada bagian *soft constraint*, terdapat penalti yang dihitung berdasarkan jumlah kelas yang mengalami konflik untuk setiap mahasiswa. Sebagai contoh jika seorang mahasiswa dijadwalkan pada dua kelas dalam waktu yang bersamaan maka solusi

```

<course id="ME_263">
    <config id="1">
        <subpart id="1_Lecture">
            <class id="Lec1" limit="100"/>
            <class id="Lec2" limit="100"/>
        </subpart>
        <subpart id="2_Recitation">
            <class id="Rec1" limit="50"/>
            <class id="Rec2" limit="50"/>
            <class id="Rec3" limit="50"/>
            <class id="Rec4" limit="50"/>
        </subpart>
    </config>
    <config id="2">
        <subpart id="3_Lecture">
            <class id="Lec3" limit="100"/>
            <class id="Lec4" limit="100"/>
        </subpart>
        <subpart id="4_Recitation">
            <class id="Rec5" parent="Lec3" limit="50"/>
            <class id="Rec6" parent="Lec3" limit="50"/>
            <class id="Rec7" parent="Lec4" limit="50"/>
            <class id="Rec8" parent="Lec4" limit="50"/>
        </subpart>
        <subpart id="5_Laboratory">
            <class id="Lab1" parent="Rec5" limit="50"/>
            <class id="Lab2" parent="Rec6" limit="50"/>
            <class id="Lab3" parent="Rec7" limit="50"/>
            <class id="Lab4" parent="Rec8" limit="50"/>
        </subpart>
    </config>
</course>

```

Gambar 6.2.4: Ilustrasi Strukur Mata Kuliah pada Permasalahan ITC 2019

ini akan dikenakan penalti sebesar 1.

6.2.5 Batasan Distribusi

Pada batasan distribusi, terdapat 19 jenis batasan yang diterapkan secara beragam pada setiap dataset. Batasan ini dapat berfungsi sebagai *hard* maupun *soft constraint*, dan berperan dalam mengatur distribusi antar kelas dalam setiap permasalahan. Ke-19 batasan ini terdiri dari:

1. **SameStart:** Kelas yang termasuk dalam batasan ini harus dimulai pada slot waktu

yang sama. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$C_i.start = C_j.start \quad \text{dengan } i, j = 0, \dots, n \quad (6.1)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan SameStart. Gambar 6.2.5 menunjukkan contoh implementasi dari batasan SameStart. Pada contoh ini, kelas dengan ID "2332" dan kelas dengan ID "2273" harus dijadwalkan untuk memulai kelas pada slot waktu yang sama.

```
<distribution type="SameStart" required="true">
  <class id="2332"/>
  <class id="2273"/>
</distribution>
```

Gambar 6.2.5: Ilustrasi Batasan SameStart

2. **SameTime:** Kelas yang termasuk dalam batasan ini harus memiliki waktu mulai dan selesai yang sama jika durasi kelas tersebut sama. Jika durasi kelas berbeda, maka kelas dengan durasi lebih singkat harus mulai bersamaan atau setelah kelas dengan durasi lebih panjang dan harus selesai sebelum atau bersamaan dengan kelas yang berdurasi lebih panjang. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$(C_i.start \leq C_j.start \wedge C_j.end \leq C_i.end) \vee (C_j.start \leq C_i.start \wedge C_i.end \leq C_j.end) \quad \text{dengan } i, j = 0, \dots, n \quad (6.2)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan SameTime. Gambar 6.2.6 menunjukkan contoh implementasi dari batasan SameTime. Pada contoh ini, kelas dengan ID "2758" dan kelas dengan ID "2759" harus memenuhi aturan tersebut.

3. **DifferentTime:** Batasan ini merupakan kebalikan dari batasan SameTime, di mana kelas-kelas yang termasuk dalam batasan ini tidak boleh memiliki waktu yang

```

<distribution type="SameTime" required="true">
  <class id="2758"/>
  <class id="2759"/>
</distribution>

```

Gambar 6.2.6: Ilustrasi Batasan SameTime

tumpang tindih. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$(C_i.end \leq C_j.start) \vee (C_j.end \leq C_i.start) \quad \text{dengan } i, j = 0, \dots, n \quad (6.3)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan DifferentTime. Gambar 6.2.7 menunjukkan contoh implementasi dari batasan DifferentTime. Pada contoh ini, kelas dengan ID "1343" dan kelas dengan ID "1344" harus dijadwalkan pada waktu yang berbeda, tanpa tumpang tindih antara kedua kelas tersebut.

```

<distribution type="DifferentTime" required="true">
  <class id="1343"/>
  <class id="1344"/>
</distribution>

```

Gambar 6.2.7: Ilustrasi Batasan DifferentTime

4. **SameDays:** Batasan ini mensyaratkan bahwa kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada hari yang sama. Jika salah satu kelas memiliki jumlah hari yang lebih sedikit dalam satu minggu, maka kelas tersebut harus dijadwalkan sebagai subset dari kelas yang memiliki jumlah hari lebih banyak. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$((C_i.days \text{ or } C_j.days) = C_i.days) \vee ((C_i.days \text{ or } C_j.days) = C_j.days) \quad \text{dengan } i, j = 0, \dots, n \quad (6.4)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan SameDays. Gambar 6.2.8 menunjukkan contoh implementasi dari batasan SameDays. Pada contoh ini, kelas dengan ID "91" dan kelas dengan ID "92" harus

dijadwalkan pada hari yang sama.

```
<distribution type="SameDays" penalty="4">
<class id="91"/>
<class id="92"/>
</distribution>
```

Gambar 6.2.8: Ilustrasi Batasan SameDays

5. **DifferentDays:** Batasan ini merupakan kebalikan dari batasan SameDays, di mana kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada hari yang berbeda. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$(C_i.days \text{ and } C_j.days) = 0 \quad \text{dengan } i, j = 0, \dots, n \quad (6.5)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan DifferentDays. Gambar 6.2.9 menunjukkan contoh implementasi dari batasan DifferentDays. Pada contoh ini, kelas dengan ID "311" dan kelas dengan ID "290" harus dijadwalkan pada hari yang berbeda.

```
<distribution type="DifferentDays" required="true">
<class id="311"/>
<class id="290"/>
</distribution>
```

Gambar 6.2.9: Ilustrasi Batasan DifferentDays

6. **SameWeeks:** Batasan ini mirip dengan batasan SameDays, namun berlaku pada tingkatan minggu. Kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada minggu yang sama. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$((C_i.weeks \text{ or } C_j.weeks) = C_i.weeks) \vee ((C_i.weeks \text{ or } C_j.weeks) = C_j.weeks) \quad (6.6)$$

dengan $i, j = 0, \dots, n$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam

batasan SameWeeks. Gambar 6.2.10 menunjukkan contoh implementasi dari batasan SameWeeks. Pada contoh ini, kelas dengan ID "428" dan kelas dengan ID "452" harus dijadwalkan pada minggu yang sama.

```
<distribution type="SameWeeks" required="true">
<class id="428"/>
<class id="452"/>
</distribution>
```

Gambar 6.2.10: Ilustrasi Batasan SameWeeks

7. **DifferentWeeks:** Batasan ini merupakan kebalikan dari batasan SameWeeks, di mana kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada minggu yang berbeda. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$(C_i.weeks \text{ and } C_j.weeks) = 0 \quad \text{dengan } i, j = 0, \dots, n \quad (6.7)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan DifferentWeeks. Gambar 6.2.11 menunjukkan contoh implementasi dari batasan DifferentWeeks. Pada contoh ini, kelas dengan ID "184," "183," dan "185" harus dijadwalkan pada minggu yang berbeda.

```
<distribution type="DifferentWeeks" required="true">
<class id="184"/>
<class id="183"/>
<class id="185"/>
</distribution>
```

Gambar 6.2.11: Ilustrasi Batasan DifferentWeeks

8. **Overlap:** Kelas-kelas yang termasuk dalam batasan ini harus memiliki overlap dari segi slot waktu, hari, dan minggu. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$\begin{aligned} & (C_j.start < C_i.end) \wedge (C_i.start < C_j.end) \wedge \\ & ((C_i.days \text{ and } C_j.days) \neq 0) \wedge ((C_i.weeks \text{ and } C_j.weeks) \neq 0) \quad (6.8) \\ & \text{dengan } i, j = 0, \dots, n \end{aligned}$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan Overlap. Gambar 6.2.12 menunjukkan contoh implementasi dari batasan Overlap. Pada contoh ini, kelas dengan ID "115," "116," "145," dan "775" harus memiliki overlap dari segi waktu, hari, dan minggu.

```
<distribution type="Overlap" required="true">
  <class id="115"/>
  <class id="116"/>
  <class id="145"/>
  <class id="775"/>
</distribution>
```

Gambar 6.2.12: Ilustrasi Batasan Overlap

9. **NotOverlap:** Batasan ini merupakan kebalikan dari batasan Overlap, di mana kelas-kelas yang termasuk dalam batasan ini tidak boleh memiliki overlap dari segi slot waktu, hari, dan minggu. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$(C_i.end \leq C_j.start) \vee (C_j.end \leq C_i.start) \vee \\ ((C_i.days \text{ and } C_j.days) = 0) \vee ((C_i.weeks \text{ and } C_j.weeks) = 0) \quad (6.9)$$

dengan $i, j = 0, \dots, n$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan NotOverlap. Gambar 6.2.13 menunjukkan contoh implementasi dari batasan NotOverlap. Pada contoh ini, kelas dengan ID "32" dan kelas dengan ID "31" harus dijadwalkan tanpa overlap dari segi waktu, hari, dan minggu.

```
<distribution type="NotOverlap" required="true">
  <class id="32"/>
  <class id="31"/>
</distribution>
```

Gambar 6.2.13: Ilustrasi Batasan NotOverlap

10. **SameRoom:** Kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada ruangan yang sama. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$C_i.room = C_j.room \quad \text{dengan } i, j = 0, \dots, n \quad (6.10)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan SameRoom. Gambar 6.2.14 menunjukkan contoh implementasi dari batasan SameRoom. Pada contoh ini, kelas dengan ID "151," "152," dan "165" harus dijadwalkan pada ruangan yang sama.

```
<distribution type="SameRoom" penalty="1">
<class id="151"/>
<class id="152"/>
<class id="165"/>
</distribution>
```

Gambar 6.2.14: Ilustrasi Batasan SameRoom

11. **DifferentRoom:** Batasan ini merupakan kebalikan dari batasan SameRoom, di mana kelas-kelas yang termasuk dalam batasan ini harus dijadwalkan pada ruangan yang berbeda. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$C_i.room \neq C_j.room \quad \text{dengan } i, j = 0, \dots, n \quad (6.11)$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan DifferentRoom. Gambar 6.2.15 menunjukkan contoh implementasi dari batasan DifferentRoom. Pada contoh ini, kelas dengan ID "385," "388," dan "438" harus dijadwalkan pada ruangan yang berbeda.

```
<distribution type="DifferentRoom" penalty="4">
<class id="385"/>
<class id="388"/>
<class id="438"/>
</distribution>
```

Gambar 6.2.15: Ilustrasi Batasan DifferentRoom

12. **SameAttendees:** Kelas-kelas yang termasuk dalam batasan ini harus memungkinkan untuk diambil oleh satu murid secara bersamaan. Artinya, kelas-kelas tersebut tidak boleh dijadwalkan secara overlap dan harus memperhitungkan waktu perjalanan dari satu kelas ke kelas lainnya. Secara matematis, batasan ini dapat dirumuskan sebagai

berikut:

$$\begin{aligned}
 & (C_i.end + C_i.room.travel[C_j.room] \leq C_j.start) \vee \\
 & (C_j.end + C_j.room.travel[C_i.room] \leq C_i.start) \vee \\
 & ((C_i.days \text{ and } C_j.days) = 0) \vee ((C_i.weeks \text{ and } C_j.weeks) = 0) \\
 & \quad \text{dengan } i, j = 0, \dots, n
 \end{aligned} \tag{6.12}$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan SameAttendees. Gambar 6.2.16 menunjukkan contoh implementasi dari batasan SameAttendees. Pada contoh ini, kelas dengan ID "22" dan kelas dengan ID "21" harus diatur sedemikian rupa sehingga memungkinkan seorang murid menghadiri keduanya.

```

<distribution type="SameAttendees" required="true">
  <class id="22"/>
  <class id="21"/>
</distribution>

```

Gambar 6.2.16: Ilustrasi Batasan SameAttendees

13. **Precedence:** Kelas-kelas yang termasuk dalam batasan ini harus diurutkan penjadwalannya sesuai dengan urutan yang ditentukan, berdasarkan pertemuan pertama di setiap kelas. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$\begin{aligned}
 & (first(C_i.weeks) < first(C_j.weeks)) \vee [(first(C_i.weeks) \\
 & = first(C_j.weeks)) \wedge [(first(C_i.days) < first(C_j.days)) \\
 & \vee ((first(C_i.days) = first(C_j.days)) \wedge (C_i.end \leq C_j.start))]] \\
 & \quad i, j = 0, \dots, n \quad i < j
 \end{aligned} \tag{6.13}$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan Precedence. Gambar 6.2.17 menunjukkan contoh implementasi dari batasan Precedence. Pada contoh ini, kelas dengan ID "1786" harus dijadwalkan sebelum kelas dengan ID "1788" dalam pertemuan pertama mereka.

```

<distribution type="Precedence" required="true">
  <class id="1786"/>
  <class id="1788"/>
</distribution>

```

Gambar 6.2.17: Ilustrasi Batasan Precedence

14. **WorkDay:** Batasan ini mengatur kelas-kelas yang tercantum agar waktu antara dimulainya satu kelas dan berakhirnya kelas lain yang terjadi pada hari yang sama tidak melebihi S slot waktu. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$\begin{aligned}
 ((C_i.days \text{ and } C_j.days) = 0) \vee ((C_i.weeks \text{ and } C_j.weeks) = 0) \vee \\
 (\max(C_i.end, C_j.end) - \min(C_i.start, C_j.start) \leq S) \quad (6.14) \\
 \text{dengan } i, j = 0, \dots, n
 \end{aligned}$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan WorkDay. Gambar 6.2.18 menunjukkan contoh implementasi dari batasan WorkDay. Pada contoh ini, kelas dengan ID "248" dan kelas dengan ID "249" harus memenuhi syarat WorkDay di mana waktu antara kelas-kelas ini pada hari yang sama tidak boleh melebihi 48 slot waktu. Sebagai contoh jika kelas ID "248" dijadwalkan pada slot ke 5 maka kelas ID "249" harus dijadwalkan paling lambat kelas tersebut berakhir di slot 53.

```

<distribution type="WorkDay (48)" required="true">
  <class id="248"/>
  <class id="249"/>
</distribution>

```

Gambar 6.2.18: Ilustrasi Batasan WorkDay

15. **MinGap:** Kelas-kelas yang termasuk dalam batasan ini, jika dijadwalkan pada hari yang sama, harus memiliki jarak minimal sejumlah G slot waktu. Secara matematis,

batasan ini dapat dirumuskan sebagai berikut:

$$\begin{aligned}
 & ((C_i.days \text{ and } C_j.days) = 0) \vee ((C_i.weeks \text{ and } C_j.weeks) = 0) \vee \\
 & (C_i.end + G \leq C_j.start) \vee (C_j.end + G \leq C_i.start) \quad (6.15) \\
 & \text{dengan } i, j = 0, \dots, n
 \end{aligned}$$

di mana C_i merupakan kelas ke- i dan n adalah jumlah kelas yang terdapat dalam batasan MinGap. Gambar 6.2.19 menunjukkan contoh implementasi dari batasan MinGap. Pada contoh ini, kelas dengan ID "269" dan kelas dengan ID "252" harus memiliki jarak minimal 20 slot waktu jika dijadwalkan pada hari yang sama. Sebagai contoh jika kedua kelas dijadwalkan pada hari yang sama dan jika ID "269" dijadwalkan pada slot ke 2 dan berakhir pada slot ke 5, maka kelas ID "252" paling cepat dijadwalkan pada slot ke 25.

```

<distribution type="MinGap(20)" required="true">
  <class id="269"/>
  <class id="252"/>
</distribution>

```

Gambar 6.2.19: Ilustrasi Batasan MinGap

16. **MaxDays:** Kelas-kelas yang termasuk dalam batasan ini tidak dapat terpisah lebih dari D hari. Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$\text{countNonzeroBits}(C_1.days \text{ or } C_2.days \text{ or } \dots \text{ or } C_n.days) \leq D \quad (6.16)$$

di mana n adalah jumlah kelas dalam batasan MaxDays, dan fungsi `countNonzeroBits` akan mengembalikan jumlah bit yang tidak bernilai 0. Gambar 6.2.20 menunjukkan contoh implementasi dari batasan MaxDays. Pada contoh ini, kelas dengan ID "446," "305," dan "304" tidak boleh terpisah lebih dari 2 hari. Jika ID "446" dijadwalkan hari senin, maka ID "305" hanya bisa dijadwalkan antara hari Sabtu, Minggu, Senin, Selasa dan Rabu. Jika ID "305" dijadwalkan hari minggu, maka pilihan hari untuk ID "304" hanya tersisa antara Sabtu, Minggu, Senin dan Selasa.

```

<distribution type="MaxDays (2)" required="true">
  <class id="446"/>
  <class id="305"/>
  <class id="304"/>
</distribution>

```

Gambar 6.2.20: Ilustrasi Batasan MaxDays

17. **MaxDayLoad:** Kelas-kelas yang termasuk dalam batasan ini tidak boleh dijadwalkan pada hari yang sama jika total jam belajar melebihi jumlah slot waktu S . Secara matematis, batasan ini dapat dirumuskan sebagai berikut:

$$\text{DayLoad}(d, w) \leq S \quad (6.17)$$

dengan:

$$\text{DayLoad}(d, w, P) = \sum_i C_i \cdot \text{length} | ((C_i.\text{days} \& 2^d) \neq 0 \wedge (C_i.\text{weeks} \& 2^w) \neq 0) \quad (6.18)$$

di mana C_i merupakan kelas ke- i dan nilai 2^d dan 2^w adalah bit string yang tidak bernilai 0 pada posisi d atau w . Untuk menghitung penalti jika batasan ini bersifat soft constraint, digunakan persamaan berikut:

$$\frac{\text{penalty} \times \sum_{w,d} \max(\text{DayLoad}(d, w) - S, 0)}{\text{nrWeeks}} \quad (6.19)$$

Gambar 6.2.21 menunjukkan contoh implementasi dari batasan ini. Pada contoh ini, batasan mengatur agar total jam belajar pada hari yang sama untuk ketiga kelas dengan ID "446," "305," dan "304" tidak melebihi 72 jam. Sebagai ilustrasi, jika ketiga kelas tersebut masing-masing memiliki durasi 30 slot waktu, maka hanya dua kelas yang boleh dijadwalkan pada hari yang sama untuk memenuhi batasan ini.

18. **MaxBreaks:** Batasan ini membatasi jumlah istirahat berdasarkan nilai yang ditentukan oleh R . Dalam konteks ini, suatu kondisi dianggap sebagai "istirahat"

```

<distribution type="MaxDayLoad(72)" required="true">
  <class id="446"/>
  <class id="305"/>
  <class id="304"/>
</distribution>

```

Gambar 6.2.21: Ilustrasi Batasan MaxDayLoad

jika terdapat jarak antar kelas yang melebihi S slot waktu. Model matematis dari batasan ini dapat dilihat pada persamaan berikut:

$$\text{MergeBlocks} \left(\{(C.start, C.end) \mid (C.days \& 2^d) \neq 0 \wedge (C.weeks \& 2^w) \neq 0\} \right) \leq R + 1 \quad (6.20)$$

Di mana nilai 2^d dan 2^w merupakan bit string yang tidak bernilai 0 pada posisi d atau w . Fungsi `MergeBlocks` menggabungkan secara rekursif blok-blok B dari dua blok B_a dan B_b yang diidentifikasi oleh slot awal dan akhir yang overlap atau yang terpisah kurang dari atau sama dengan S slot waktu, hingga tidak ada lagi blok yang dapat disatukan. Model matematis fungsi ini dapat dilihat pada persamaan berikut:

$$(B_a.end + S > B_b.start) \wedge (B_b.end + S > B_a.start) \implies \\ (B.start = \min(B_a.start, B_b.start)) \wedge (B.end = \max(B_a.end, B_b.end)) \quad (6.21)$$

Gambar 6.2.22 mengilustrasikan batasan ini. Pada contoh ini, tidak boleh ada "istirahat" antara kelas dengan ID 840 dan 841, di mana kondisi istirahat diidentifikasi jika jarak antar kedua kelas lebih dari 72 slot waktu.

```

<distribution type="MaxBreaks(0, 72)" required="true">
  <class id="841"/>
  <class id="840"/>
</distribution>

```

Gambar 6.2.22: Ilustrasi Batasan MaxBreaks

19. **MaxBlock:** Batasan ini memberikan pembatasan terhadap panjang sebuah blok agar tidak melebihi M slot waktu. Sekelompok kelas dikatakan berada dalam satu blok

jika kelas-kelas tersebut dijadwalkan pada hari dan minggu yang sama, di mana jarak antar dua kelas tidak lebih dari S slot waktu. Model matematis dari batasan ini dapat dirumuskan dengan persamaan berikut:

$$\max \left(\left\{ B.\text{end} - B.\text{start} \mid B \in \text{MergeBlocks}(\{(C.\text{start}, C.\text{end}) \mid (C.\text{days} \& 2^d) \neq 0 \wedge (C.\text{weeks} \& 2^w) \neq 0\}) \right\} \right) \leq M \quad (6.22)$$

Gambar 6.2.23 mengilustrasikan batasan ini. Pada contoh ini, suatu blok tidak boleh memiliki panjang lebih dari 40 slot waktu. Selain itu, dua kelas atau lebih dikatakan berada dalam satu blok jika memiliki jarak maksimal 9 slot waktu. Sebagai ilustrasi, jika kelas dengan ID 666 dijadwalkan mulai slot 1 hingga slot 20, dan kelas dengan ID 1436 dijadwalkan mulai slot 23 hingga slot 45, kedua kelas ini berada dalam satu blok. Namun, solusi ini melanggar batasan panjang maksimal blok, karena panjang blok ini adalah 45 slot waktu, sedangkan batas maksimal yang diizinkan oleh batasan ini adalah 40 slot waktu.

```
<distribution type="MaxBlock(40, 9)" penalty="4">
<class id="666"/>
<class id="1436"/>
<class id="1657"/>
<class id="1639"/>
<class id="667"/>
</distribution>
```

Gambar 6.2.23: Ilustrasi Batasan MaxBlock

6.3 Hasil Implementasi dan Analisis Algoritma

Pada *benchmark* ITC 2019, implementasi algoritma dimulai dengan tahapan pencarian solusi *feasible* untuk setiap dataset. Setelah solusi *feasible* ditemukan, proses dilanjutkan dengan tahapan optimasi solusi. Tahapan ini menggunakan input dari hasil solusi *feasible* yang dipilih secara acak. Hasil dari tahapan pencarian solusi *feasible* dan optimasi divalidasi dengan mengunggah solusi ke situs web resmi validator

(<https://www.itc2019.org/validator>). Proses ini bertujuan untuk memastikan bahwa solusi yang dihasilkan memenuhi kriteria *feasible* dan nilai penalti yang dihitung oleh algoritma sesuai dengan hasil perhitungan validator.

Dalam proses pencarian solusi *feasible* dan optimasi, hanya Low-Level Heuristic (LLH) *move* yang memungkinkan untuk diterapkan. Hal ini dikarenakan karakteristik dari permasalahan ITC 2019 yang sangat membatasi jumlah slot waktu yang tersedia untuk setiap mata kuliah. Hal ini menyebabkan sangat kecilnya kemungkinan untuk melakukan pertukaran slot waktu antara dua kelas. Hanya menggunakan satu LLH *move* mengganggu tahapan *perturbation* dalam tahapan mencari solusi *feasible*. Solusi yang dihasilkan dari tahapan *perturbation* hanya menghasilkan perubahan solusi yang minim. Hal ini akan mengganggu strategi yang dijalankan dan memungkinkan untuk terjebak dalam kondisi *local optima*. Untuk memperbaiki hal tersebut, maka LLH *move* dimungkinkan untuk dijalankan sebanyak satu atau dua kali dalam tahapan *perturbation* dengan ditentukan secara acak.

Hasil dari tahapan pencarian solusi *feasible* disajikan pada Subbab 6.3.1. Sementara itu, Subbab 6.3.2 membahas analisis dari kinerja algoritma PA-ILS dalam menghasilkan solusi *feasible*. Hasil implementasi algoritma AT-ILS dalam proses optimasi dan analisis kinerja algoritma disajikan pada Subbab 6.3.3 dan 6.3.4.

6.3.1 Hasil Tahapan Pencarian Solusi *Feasible*

Pada bagian ini, pencarian solusi *feasible* dilakukan terhadap 30 dataset. Hasil uji coba dari tahapan ini ditampilkan dalam Tabel 6.3.1, 6.3.2, dan 6.3.3. Setiap tabel menyajikan waktu rata-rata, waktu tercepat, waktu terlambat, serta persentase keberhasilan dalam menemukan solusi *feasible*.

Dari hasil yang diperoleh, algoritma PA-ILS menunjukkan performa yang sangat baik dengan mampu memperoleh solusi *feasible* pada seluruh dataset dengan persentase keberhasilan 100% (10 solusi *feasible* dari 10 percobaan). Hal ini mengindikasikan kemampuan algoritma dalam menghasilkan solusi *feasible* secara konsisten pada *benchmark* ITC 2019 yang memiliki kompleksitas lebih tinggi dibandingkan dengan *benchmark* Toronto.

Dalam aspek waktu, algoritma ini juga berjalan dengan efisien. Sepuluh dataset dapat

ditemukan solusi *feasible* dalam waktu rata-rata pencarian kurang dari 1 menit, dan delapan belas dataset dalam waktu rata-rata kurang dari 1 jam. Dua dataset lainnya, yaitu *agh-fal17* dan *pu-proj-fal19*, membutuhkan waktu yang lebih lama, masing-masing sebesar rata-rata 3,2 jam dan 8,1 jam. Secara keseluruhan, permasalahan ini memerlukan waktu yang lebih lama untuk menemukan solusi *feasible* dibandingkan dengan *benchmark* sebelumnya, yaitu Toronto. Namun, hal ini dapat dipahami mengingat dataset ITC 2019 memiliki ukuran dan kompleksitas yang jauh lebih tinggi. Selain itu, waktu rata-rata yang dibutuhkan untuk menghasilkan solusi *feasible* masih dalam durasi yang wajar, sehingga algoritma ini sangat memungkinkan untuk diterapkan dalam aplikasi permasalahan nyata.

Tabel 6.3.1: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark* ITC 2019 Kategori Early

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
agh-fis-spr17	1683,7	1245	1868	100%
agh-ggis-spr17	583,5	213	2374	100%
bet-fal17	2485,0	1458	3738	100%
iku-fal17	1288,4	853	2041	100%
mary-spr17	39,9	15	100	100%
muni-fi-spr16	8,2	5	14	100%
muni-fsp-spr17	8,0	3	22	100%
muni-pdf-spr16c	335,5	240	501	100%
pu-llr-spr17	36,5	17	69	100%
tg-fal17	161,7	78	259	100%

Tabel 6.3.2: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark* ITC 2019 Kategori Middle

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
agh-ggos-spr17	388,1	251	704	100%
agh-h-spr17	807,8	223	2362	100%
lums-spr18	71,6	30	193	100%

(Lanjutan) Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2019* Kategori Middle

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
muni-fi-spr17	14,9	6	56	100%
muni-fspss-spr17c	45,0	21	98	100%
muni-pdf-spr16	46,6	31	75	100%
nbi-spr18	41,3	8	75	100%
pu-d5-spr17	61,1	29	132	100%
pu-proj-fal19	29175,0	13751	48295	100%
yach-fal17	349,0	68	843	100%

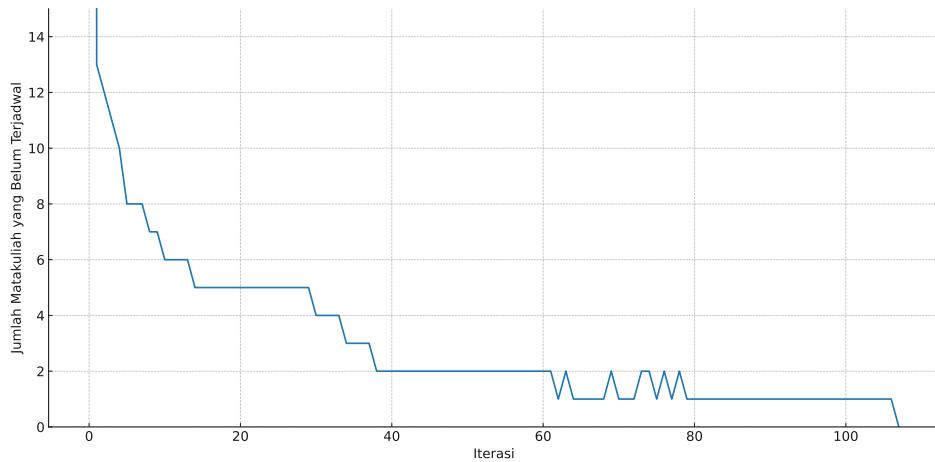
Tabel 6.3.3: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2019* Kategori Late

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
agh-fal17	11805,5	5270	19764	100%
bet-spr18	1246,7	416	3275	100%
iku-spr18	3098,3	1219	5450	100%
lums-fal17	274,5	134	648	100%
mary-fal18	15,2	7	41	100%
muni-fi-fal17	11,1	7	19	100%
muni-fspssx-fal17	184,7	79	320	100%
muni-pdfx-fal17	5717,7	1071	26732	100%
pu-d9-fal19	500,6	211	787	100%
tg-spr18	167,6	43	302	100%

6.3.2 Analisis Proses Algoritma PA-ILS

Untuk menggambarkan bagaimana algoritma bekerja dalam mencari solusi *feasible*, Gambar 6.3.1 menyajikan visualisasi perubahan jumlah mata kuliah yang belum terjadwalkan pada setiap iterasinya. Ilustrasi ini menunjukkan bahwa algoritma berjalan sesuai dengan strategi yang diterapkan. Algoritma memungkinkan penerimaan solusi yang

kurang optimal seperti yang ditunjukan pada iterasi ke-60 hingga ke-80, namun tetap mengarahkan proses pencarian menuju konvergensi. Hasil ini konsisten dengan uji coba pada *benchmark* Toronto, meskipun terdapat perbedaan pada tahap awal iterasi di mana pada kasus ITC 2019 jumlah mata kuliah yang belum terjadwalkan berkurang secara lebih perlahan dibandingkan dengan *benchmark* Toronto.



Gambar 6.3.1: Grafik Perubahan Solusi dalam Proses Pencarian Solusi *Feasible* pada *Benchmark* ITC 2019

6.3.3 Hasil Tahapan Optimasi

Pada bagian ini, solusi *feasible* yang telah dihasilkan pada tahapan sebelumnya dioptimasi untuk menurunkan nilai penalti. Proses optimasi dijalankan dalam dua tahapan. Tahapan pertama melakukan optimasi sebanyak 10 kali pada setiap dataset dengan batas waktu setiap uji coba adalah 20 jam. Pada tahapan ini, pemilihan solusi *feasible* dilakukan secara acak dari 10 solusi *feasible* yang dihasilkan dalam tahapan sebelumnya. Sementara itu, tahapan kedua melakukan optimasi dengan tujuan untuk mencari solusi terbaik. Tahapan ini dijalankan dengan mengoptimasi kembali hasil terbaik yang diperoleh pada tahapan pertama. Tahapan ini mengevaluasi solusi yang dihasilkan setiap 10 jam. Jika dalam waktu tersebut tidak menghasilkan solusi terbaik baru, maka tahapan ini akan berakhir.

Hasil dari kedua tahapan optimasi disajikan dalam Tabel 6.3.4, 6.3.5, dan 6.3.6. Pada proses optimasi pertama, tabel menampilkan nilai penalti rata-rata, terbaik, terburuk, serta nilai Koefisien Variasi (KV) untuk melihat konsistensi dari algoritma AT-ILS. Dari hasil ini, terdapat 10 dataset dengan nilai KV di bawah 3%, 13 dataset dengan rentang nilai KV

3%-10%, dan 7 dataset dengan nilai KV di atas 10%. Jika dibandingkan dengan *benchmark* sebelumnya, hasil ini menunjukkan penurunan konsistensi, di mana pada *benchmark* Toronto mendapatkan nilai KV di bawah 7%. Namun, permasalahan ini memiliki tingkat kompleksitas yang lebih tinggi dari segi ukuran permasalahan dan jumlah batasan. Selain itu, slot waktu yang dapat digunakan pada setiap kelas sangat terbatas, tidak seperti pada *benchmark* Toronto yang memungkinkan penjadwalan pada slot waktu manapun. Kondisi tersebut menurunkan fleksibilitas algoritma, sehingga menghasilkan variasi yang lebih besar dalam hasil optimasi.

Untuk melihat ilustrasi lebih detail dari keberagaman solusi yang dihasilkan, Gambar 6.3.2 menampilkan box plot dari dataset *agh-fall17* yang merupakan dataset dengan hasil paling konsisten dengan nilai KV 1,4%, dataset *muni-pdf-spr16c* sebagai dataset dengan tingkat konsistensi rata-rata dengan nilai KV rata-rata 8,97%, dan dataset *muni-fsp-spr17c* yang merupakan dataset dengan hasil paling tidak konsisten dengan nilai KV 52,23%. Dari gambar ini dapat dilihat bahwa pada dataset *agh-fall17* menunjukkan variasi data yang lebih konsisten, ditunjukkan dari ukuran box yang lebih kecil. Sementara itu, *muni-pdf-spr16c* menghasilkan ukuran box yang lebih besar dibandingkan dengan *agh-fall17*, namun tidak sebesar pada *muni-fsp-spr17c*. Dari contoh tiga dataset ini, penyebab nilai KV yang besar memang diakibatkan dari variasi hasil yang beragam, bukan dari adanya *outlier*.

Sementara itu, pada tahapan kedua yang disajikan dalam kolom Hasil Terbaik dan Waktu Eksekusi, seluruh dataset memerlukan tambahan waktu optimasi untuk menemukan solusi terbaiknya. Secara keseluruhan, jika digabungkan dengan tahapan pertama, waktu proses optimasi berkisar antara 30 hingga 250 jam. Hal ini berbeda dengan hasil pada *benchmark* Toronto, di mana hanya beberapa dataset saja yang memerlukan penambahan waktu untuk menemukan solusi terbaiknya. Namun, perlu diingat bahwa ukuran dataset pada permasalahan ITC 2019 jauh lebih besar dibandingkan dengan dataset Toronto. Seperti yang ditampilkan pada bagian 6.1.1, dataset yang digunakan jauh lebih besar dan pada permasalahan ITC 2019 terdapat dua variabel keputusan, yakni kelas dan mahasiswa, tidak seperti pada *benchmark* Toronto yang hanya melibatkan kelas saja. Oleh karena itu, proses optimasi membutuhkan waktu yang lebih lama.

Dari sisi praktikal, waktu yang dibutuhkan untuk mencari solusi terbaik masih dalam rentang yang wajar. Pada umumnya, dalam menjadwalkan kelas di tingkat universitas

memiliki jeda waktu yang cenderung fleksibel, yaitu dalam beberapa hari hingga beberapa minggu bergantung pada kompleksitas permasalahan. Pada hasil ini, waktu yang diperoleh antara 30 hingga 250 jam atau 1,25 hingga 10 hari. Durasi ini masih dalam rentang beberapa hari dan kurang dari 2 minggu. Hasil ini menunjukkan bahwa algoritma ini masih layak untuk digunakan secara praktikal dengan rentang waktu yang dibutuhkan masih dalam rentang waktu yang umum untuk menghasilkan solusi penjadwalan.

Tabel 6.3.4: Hasil Uji Coba Tahapan Optimasi pada *Benchmark ITC 2019* Kategori Early

Dataset	Waktu Eksekusi (20 Jam)				Waktu Eksekusi (>20 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
agh-fis-spr17	8166,9	7974	8378	2,03	7780	60 jam
agh-ggis-spr17	125805,7	120513	137824	5,60	89081	210 jam
bet-fal17	319668,3	313480	334155	2,41	309109	150 jam
iku-fal17	65267,0	61626	69118	4,01	58373	230 jam
mary-spr17	22017,1	20047	23570	7,35	17880	80 jam
muni-fi-spr16	7148,2	6934	7293	2,00	6859	70 jam
muni-fsp-spr17	13128,6	9037	14042	11,32	8949	50 jam
muni-pdf-spr16c	276993,8	247765	317224	8,97	87813	140 jam
pu-llr-spr17	30713,3	28110	33091	5,02	27729	100 jam
tg-fal17	9386,9	7978	10736	11,71	7818	70 jam

Tabel 6.3.5: Hasil Uji Coba Tahapan Optimasi pada *Benchmark ITC 2019* Kategori Middle

Dataset	Waktu Eksekusi (20 Jam)				Waktu Eksekusi (>20 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
agh-ggos-spr17	17755,0	16065	19639	7,77	15905	50 jam
agh-h-spr17	30704,7	29987	32372	2,31	29315	70 jam
lums-spr18	158,5	132	169	6,79	128	30 jam
muni-fi-spr17	5776,3	5532	5854	1,58	5381	70 jam

(Lanjutan) Hasil Uji Coba Tahapan Optimasi pada *Benchmark* ITC 2019 Kategori Middle

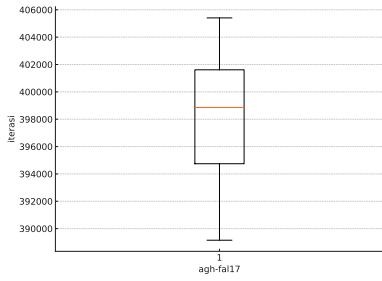
Dataset	Waktu Eksekusi (20 Jam)				Waktu Eksekusi (>20 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
muni-fsp-spr17c	75020,2	41181	129807	52,23	13614	60 jam
muni-pdf-spr16	68739,5	55596	113982	30,72	52565	60 jam
nbi-spr18	38688,2	37744	39406	1,63	32782	150 jam
pu-d5-spr17	22269,2	22549	22710	1,60	21250	50 jam
pu-proj-fal19	411570,6	78928	456745	28,46	237544	220 jam
yach-fal17	2416,1	2136	3052	11,05	1949	90 jam

Tabel 6.3.6: Hasil Uji Coba Tahapan Optimasi pada *Benchmark* ITC 2019 Kategori Late

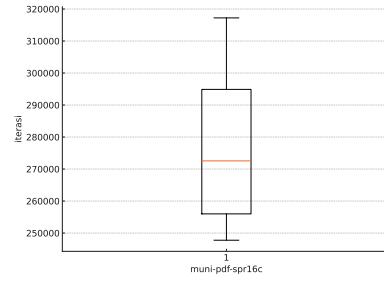
Dataset	Waktu Eksekusi (20 Jam)				Waktu Eksekusi (>20 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
agh-fal17	397942,3	389151	405408	1,40	183741	250 jam
bet-spr18	2373705,2	367814	384830	2,05	365852	120 jam
iku-spr18	98226,1	96844	100573	1,68	86121	250 jam
lums-fal17	603,0	539	641	4,71	531	50 jam
mary-fal18	7843,1	7393	8466	4,89	7215	50 jam
muni-fi-fal17	6873,7	6380	7177	3,76	6100	60 jam
muni-fsp-sx-fal17	490615	297978	650174	28,25	65020	140 jam
muni-pdfx-fal17	467283,4	431016	536445	6,04	190830	160 jam
pu-d9-fal19	320696,2	300315	333283	3,20	101744	250 jam
tg-spr18	27562,8	26018	28516	3,39	24498	50 jam

6.3.4 Analisis Proses Algoritma AT-ILS

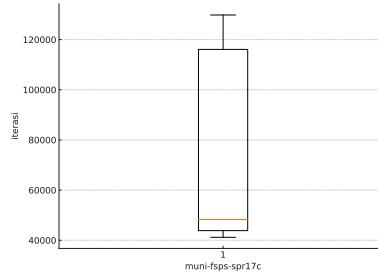
Untuk melihat bagaimana algoritma AT-ILS bekerja pada *benchmark* ITC 2019, Gambar 6.3.3 dan 6.3.1 menyajikan visualisasi dari tahapan optimasi secara keseluruhan



(a) agh-fal17



(b) muni-pdf-spr16c

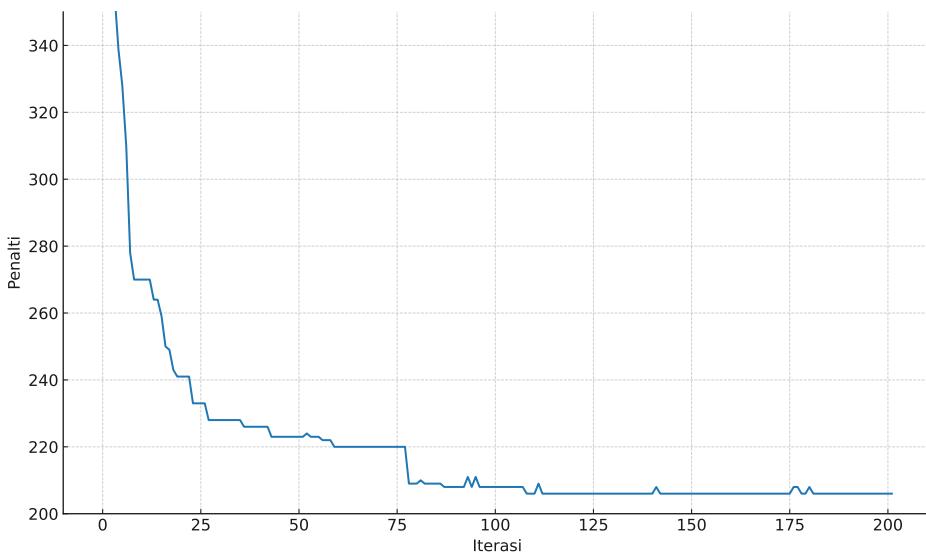


(c) muni-fsp-spr17c

Gambar 6.3.2: Box plot Hasil Algoritma Optimasi pada Tiga Dataset: agh-fal17, muni-pdf-spr16c, dan muni-fsp-spr17c

dan tahapan *local search*. Pada tahapan optimasi secara keseluruhan, hasil visualisasi menunjukkan penurunan nilai penalti yang menunjukkan konvergensi, namun tetap memungkinkan penerimaan solusi yang kurang optimal. Penerimaan solusi yang kurang optimal ini akan dikembalikan ke kondisi solusi terbaik jika dalam beberapa iterasi berikutnya tidak mampu menemukan solusi yang lebih baik. Misalnya, pada iterasi ke-175 ke atas, terjadi penerimaan solusi yang kurang optimal, namun pada iterasi berikutnya kondisi solusi terbaik dikembalikan. Hal ini mengindikasikan bahwa strategi algoritma yang diterapkan berjalan sesuai dengan yang diharapkan, yaitu algoritma mempertahankan arah konvergensi sambil memungkinkan penerimaan solusi yang kurang optimal untuk sementara waktu.

Sementara itu, pada tahapan *local search*, Gambar 6.3.1 menunjukkan bahwa perubahan solusi pada setiap iterasinya memungkinkan algoritma untuk terus melakukan eksplorasi. Namun, secara keseluruhan, tahapan ini tetap fokus pada eksplorasi dengan nilai ambang batas yang terus menekan solusi terbaru ke arah konvergensi. Dari visualisasi ini juga dapat dilihat bahwa nilai ambang batas dapat dinaikkan untuk menghindari kondisi *local optima*, seperti yang terjadi pada iterasi berkisar 200 hingga 300.



Gambar 6.3.3: Grafik Perubahan Solusi dalam Proses Optimasi pada *Benchmark ITC 2019*

6.4 Perbandingan dengan Studi Sebelumnya

Untuk mengevaluasi performa kedua algoritma yang dikembangkan dalam penelitian ini, hasil dari implementasi algoritma dibandingkan dengan delapan penelitian lainnya. Perbandingan pertama dilakukan dari sisi feasibilitas. Tabel 6.4.1 menyajikan perbandingan jumlah solusi *feasible* yang berhasil diperoleh dibandingkan dengan penelitian sebelumnya. Dari hasil ini, kebanyakan penelitian mampu menghasilkan seluruh solusi *feasible* seperti yang dihasilkan dalam penelitian ini. Sementara itu, terdapat tiga penelitian yang gagal menghasilkan solusi *feasible* pada seluruh dataset. Penelitian oleh Rappos et al. (2022) gagal menghasilkan solusi *feasible* pada satu dataset (agh-fal17). Sementara itu dua penelitian lainnya memiliki hasil yang jauh lebih buruk dengan penelitian oleh Lemos et al. (2020) menghasilkan 20 dataset dengan solusi *feasible*, dan penelitian oleh Holm et al. (2022) menghasilkan 10 dataset dengan solusi *feasible*. Dalam faktor persentase keberhasilan, tidak ada satupun studi yang melaporkan hal tersebut, sehingga perbandingan dari faktor persentase keberhasilan tidak dapat dilakukan.

Tabel 6.4.1: Perbandingan Jumlah Solusi *Feasible* pada Permasalahan ITC 2019

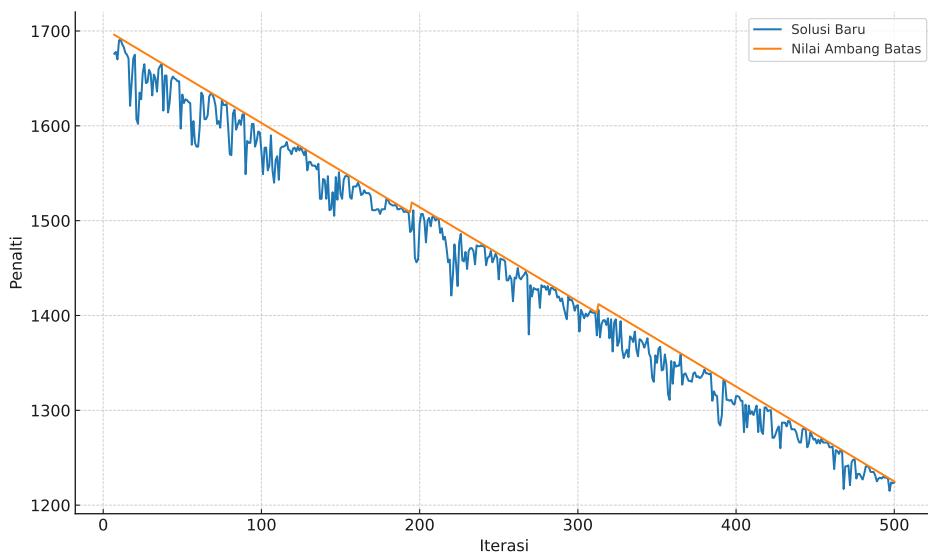
Author	Early	Middle	Late	Total
Penelitian ini	10	10	10	30
Mikkelsen and Holm (2022)	10	10	10	30

(Lanjutan) Perbandingan Jumlah Solusi *Feasible* pada Permasalahan ITC 2019

Author	Early	Middle	Late	Total
Sylejmani et al. (2022)	10	10	10	30
Premananda, Tjahyanto and Muklason (2022)	10	10	10	30
Lemos et al. (2021)	10	10	10	30
Premananda and Muklason (2021)	10	10	10	30
Rappos et al. (2022)	10	10	9	29
Lemos et al. (2020)	8	8	4	20
Holm et al. (2022)	4	4	2	10

Selanjutnya, perbandingan kedua dilakukan dengan penambahan aspek optimasi. Pada perbandingan ini dilakukan pemeringkatan dari hasil pada setiap dataset. Tabel 6.4.2 menampilkan hasil perbandingan rata-rata peringkat dari kategori Early, Middle, Late, dan pemeringkatan secara keseluruhan dengan delapan penelitian terdahulu. Gambar 6.4.1, 6.4.2, 6.4.3, dan 6.4.4 menunjukkan persebaran perbandingan peringkat antar penelitian. Hasil perbandingan menunjukkan algoritma AT-ILS mampu menghasilkan hasil yang kompetitif dengan berada pada peringkat empat dan unggul dari lima penelitian lainnya. Hasil ini merupakan hasil yang positif mengingat bahwa algoritma yang dikembangkan ditujukan untuk generalitas dan dalam proses optimasi, algoritma yang dikembangkan tidak membutuhkan pengaturan parameter. Dengan pengembangan tersebut, algoritma ini masih mampu memperoleh peringkat di atas rata-rata. Sementara penelitian yang mendapatkan peringkat 1 hingga 3 mengembangkan algoritmanya secara spesifik yang ditujukan khusus pada permasalahan ITC 2019. Sebagai perbandingan, penelitian oleh Sylejmani et al. (2022) mengembangkan algoritma Simulated Annealing yang memiliki 30 variabel yang perlu dilakukan pengaturan nilai parameter agar sesuai dengan *benchmark* ITC 2019.

Untuk melihat lebih detail dari hasil perbandingan, Tabel 6.4.3, 6.4.4, dan 6.4.5 menunjukkan perbandingan nilai penalti dari setiap dataset pada 5 penelitian dengan posisi peringkat teratas. Hasil lengkap pemeringkatan dapat dilihat pada Lampiran B. Hasil perbandingan ini menunjukkan hasil dari penelitian oleh Mikkelsen and Holm (2022) yang



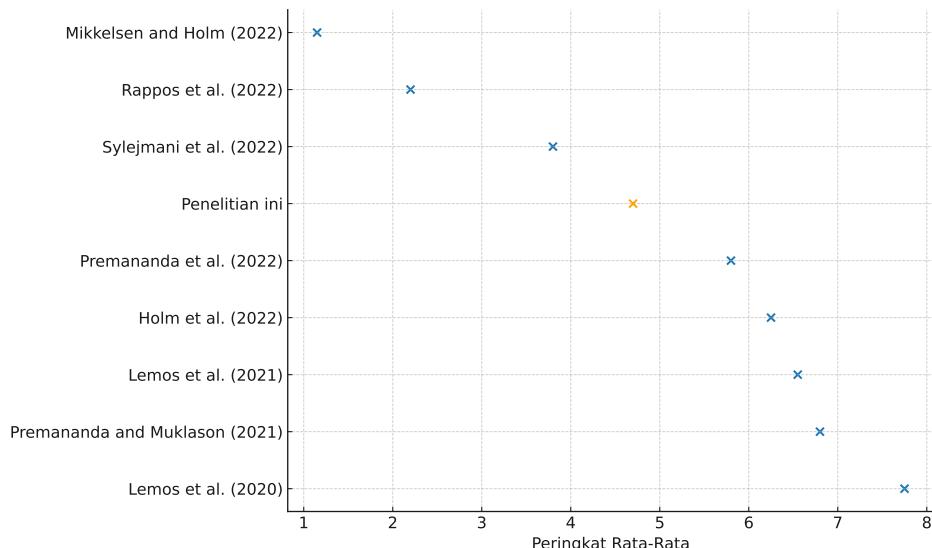
Gambar 6.3.4: Grafik Perubahan Solusi dalam Proses Local Search pada *Benchmark ITC 2019*

mengembangkan *matheuristic* dengan penggabungan Mix Integer Programming dan algoritma Fix-and-Optimize mendominasi pada seluruh dataset. Jika dibandingkan dengan hasil dari penelitian ini, pada beberapa dataset penelitian ini menghasilkan solusi dengan penalti yang tidak terpaut terlalu jauh, seperti pada dataset *bet-fal17*, *agh-h-spr17*, *lums-spr*, *pu-d5-spr17*, *pu-proj-fal19*, dan *bet-spr18*. Namun, pada dataset lainnya, hasil penelitian ini menunjukkan ketertinggalan yang lumayan jauh seperti pada dataset *muni-fsps-spr17*, *muni-fsps-spr17c*, dan *muni-fspsx-fal17*.

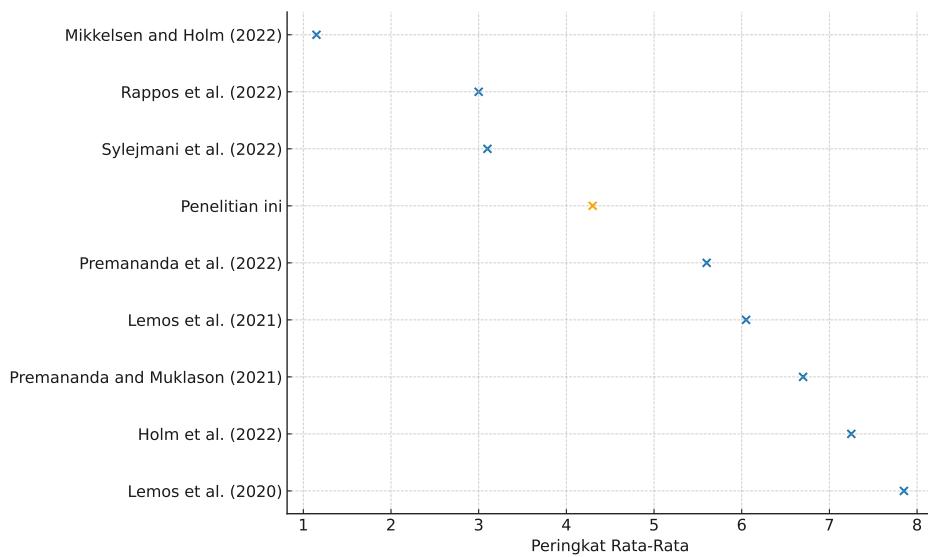
Sementara itu, jika dibandingkan dengan penelitian dengan metode serupa seperti oleh Sylejmani et al. (2022) yang mengembangkan algoritma Simulated Annealing, kebanyakan solusi yang dihasilkan dalam penelitian ini memiliki penalti yang tidak terlalu jauh dari hasil penelitian Sylejmani et al. (2022). Bahkan, pada dua dataset penelitian ini mampu menghasilkan hasil yang lebih baik, yaitu pada dataset *pu-proj-fal19* dan *agh-fal17*. Sementara jika dibandingkan dengan penelitian oleh Premananda, Tjahyanto and Muklason (2022) yang mengembangkan algoritma hibrida Whale Optimization dan penelitian oleh Premananda and Muklason (2021) yang mengembangkan algoritma Great Deluge, penelitian ini menghasilkan solusi yang lebih baik pada seluruh dataset.

Tabel 6.4.2: Perbandingan Hasil Peringkat Keseluruhan pada Permasalahan ITC 2019

Peneliti	Early	Middle	Late	Rata-rata Peringkat
Mikkelsen and Holm (2022)	1,15	1,15	1,05	1,11
Rappos et al. (2022)	2,20	3,00	3,10	2,76
Sylejmani et al. (2022)	3,80	3,10	3,50	3,46
Penelitian ini	4,70	4,30	4,30	4,43
Premananda, Tjahyanto and Muklason (2022)	5,80	5,60	5,60	5,66
Lemos et al. (2021)	6,55	6,05	4,95	5,85
Premananda and Muklason (2021)	6,80	6,70	7,10	6,86
Holm et al. (2022)	6,25	7,25	7,30	6,93
Lemos et al. (2020)	7,75	7,85	8,10	7,90



Gambar 6.4.1: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2019 pada Kategori Early



Gambar 6.4.2: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2019 pada Kategori Middle

Tabel 6.4.3: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada *Benchmark* ITC 2019 Kategori Early

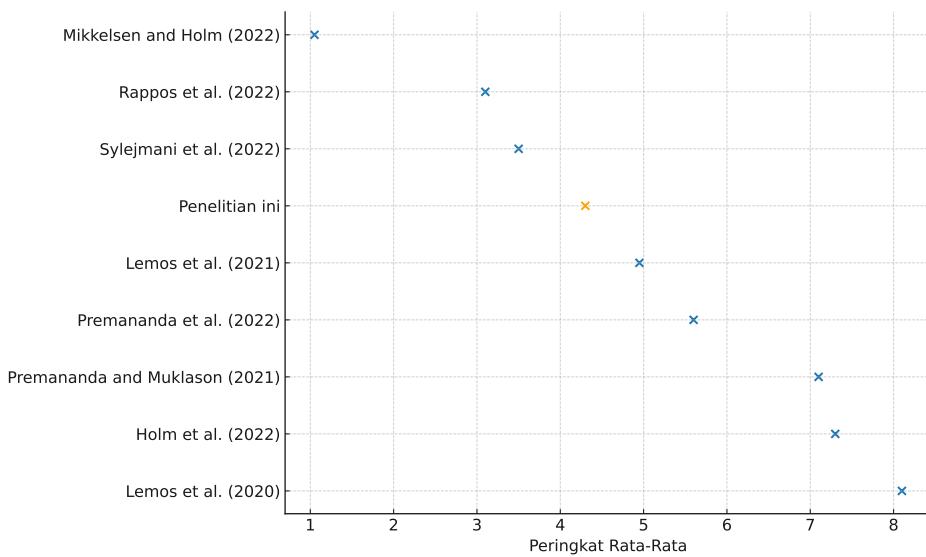
Dataset	Penelitian ini	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Premananda, Tjahyanto and Muklason (2022)
agh-fis-spr17	7780	3094	4557	6799	9345
agh-ggis-spr17	89081	35147	36616	77932	92556
bet-fal17	309109	290127	295427	299205	320360
iku-fal17	58373	18989	26840	50613	86908
mary-spr17	17880	14922	15021	15894	27525
muni-fi-spr16	6859	3764	3844	5006	7698
muni-fsp-spr17	8949	868	883	1938	16161
muni-pdf-spr16c	87813	35731	37487	58206	140026
pu-llr-spr17	27729	10038	13385	16874	35767
tg-fal17	7818	4215	4215	8044	11285

(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada *Benchmark ITC 2019* Kategori Early

<i>Dataset</i>	Penelitian ini	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Premananda, Tjahyanto and Muklason (2022)
Rata-Rata Peringkat	4,70	1,15	2,20	3,80	5,80

Tabel 6.4.4: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada *Benchmark ITC 2019* Kategori Middle

<i>Dataset</i>	Penelitian ini	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Premananda, Tjahyanto and Muklason (2022)
agh-ggos-spr17	15905	3237	6320	9328	18985
agh-h-spr17	29315	21559	26159	25081	30893
lums-spr18	128	95	114	107	147
muni-fi-spr17	5381	3796	4289	4692	7127
muni-fsps-spr17c	13614	2780	3303	9222	31534
muni-pdf-spr16	52565	19320	24318	40074	77907
nbi-spr18	32782	18014	19055	26517	53254
pu-d5-spr17	21250	15842	18813	19440	24815
pu-proj-fal19	237544	178135	561194	237909	466488
yach-fal17	1949	1410	1844	1727	4206
Rata-Rata Peringkat	4,30	1,15	3,00	3,10	5,60



Gambar 6.4.3: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2019 pada Kategori Late

Tabel 6.4.5: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada *Benchmark* ITC 2019 Kategori Late

Dataset	Penelitian ini	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Premananda, Tjahyanto and Muklason (2022)
agh-fal17	183741	140194	-	184030	215516
bet-spr18	365852	350410	360057	360437	376676
iku-spr18	86121	25863	36711	85969	111952
lums-fal17	531	349	386	486	717
mary-fal18	7215	4331	5637	7199	11194
muni-fi-fal17	6100	3129	3794	4712	7776
muni-fspsx-fal17	65020	12390	33001	41933	129419
muni-pdfx-fal17	190830	84703	151464	159203	281751
pu-d9-fal19	101744	39251	134009	82757	201122
tg-spr18	24498	12704	12856	15992	31710

(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Empat Studi Sebelumnya pada *Benchmark* ITC 2019 Kategori Late

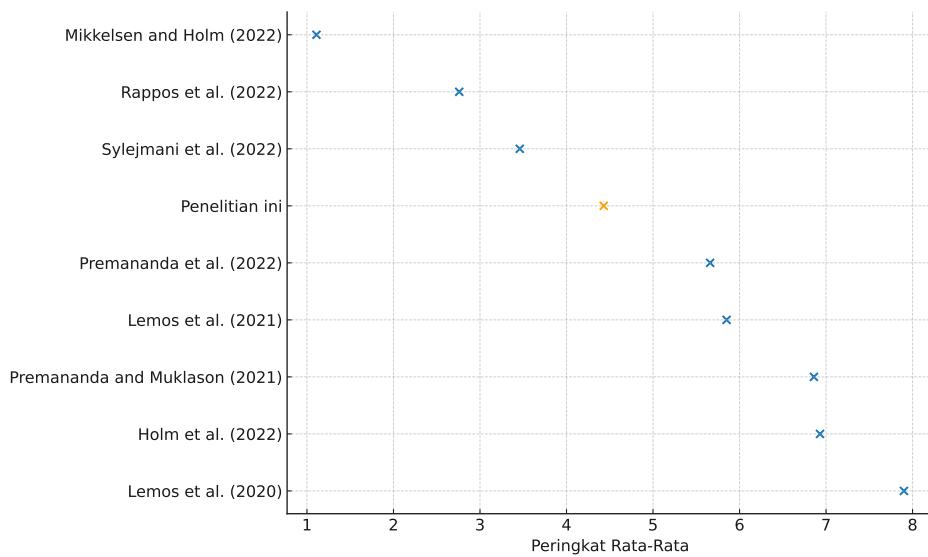
<i>Dataset</i>	Penelitian ini	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Premananda, Tjahyanto and Muklason (2022)
Rata-Rata Peringkat	4,30	1,05	3,10	3,50	5,60

6.5 Kesimpulan

Bab ini melakukan implementasi algoritma yang telah dikembangkan pada permasalahan kedua, yaitu permasalahan penjadwalan mata kuliah menggunakan *benchmark* ITC 2019. *Benchmark* ITC 2019 mengangkat permasalahan dengan ukuran yang jauh lebih besar dan juga lebih kompleks dibandingkan dengan permasalahan pada *benchmark* Toronto. Pemilihan menguji algoritma pada *benchmark* ITC 2019 bertujuan untuk memberikan pengujian yang lebih variatif terhadap algoritma yang dikembangkan guna menguji tingkat generalitasnya.

Pengujian algoritma PA-ILS menunjukkan hasil yang sangat baik dengan mampu menghasilkan solusi *feasible* pada seluruh dataset dengan tingkat keberhasilan 100%. Sementara itu, pada tahapan optimasi, algoritma AT-ILS mampu menghasilkan solusi yang kompetitif dengan menempati peringkat ke-4 dari 9 penelitian. Hasil dari uji coba ini menunjukkan bahwa algoritma yang dikembangkan memiliki potensi yang baik dalam menyelesaikan permasalahan penjadwalan lintas domain.

Pada tahapan pencarian solusi *feasible*, hasil uji coba menunjukkan dengan jelas bahwa algoritma yang dikembangkan memiliki performa dengan sangat baik, dimana dalam uji coba pada dua permasalahan yang berbeda, algoritma ini selalu berhasil menghasilkan solusi *feasible*. Sedangkan pada tahapan optimasi, meskipun tidak menghasilkan hasil terbaik dibandingkan penelitian sebelumnya, algoritma yang dikembangkan mampu memberikan solusi yang baik ketika dibandingkan dengan penelitian sebelumnya, dengan



Gambar 6.4.4: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2019 pada Seluruh Kategori

berada pada peringkat di atas rata-rata. Hal ini sudah menunjukkan hasil yang baik karena algoritma yang dikembangkan difokuskan pada sisi generalitas bukan berfokus untuk meraih hasil terbaik pada satu permasalahan saja. Untuk memvalidasi lebih lanjut generalitas dari kedua algoritma ini, uji coba dilanjutkan pada permasalahan yang berbeda yaitu pada permasalahan penjadwalan kompetisi olahraga pada Bab 7.

BAB 7

PERMASALAHAN PENJADWALAN KOMPETISI OLAH RAGA (*BENCHMARK INTERNATIONAL TIMETABLING COMPETITION 2021*)

International Timetabling Competition (ITC) 2021 merupakan kompetisi yang bertujuan untuk menyelesaikan permasalahan penjadwalan turnamen olahraga. Kompetisi ini menghadirkan dataset yang bersifat artifisial, namun dirancang untuk mencakup tantangan yang kompleks dan realistik, serta menampilkan variasi masalah antar dataset (Van Bulck and Goossens, 2023). Dataset pada *benchmark* ITC 2021 terdiri dari tim-tim yang berjumlah antara 16 hingga 20, yang mengikuti format kompetisi *double round-robin*. Dalam format ini, setiap tim bertanding melawan semua tim lain dalam dua ronde, dengan lokasi pertandingan yang berbeda pada setiap ronde. Sebagai contoh, jika pada ronde pertama tim 1 bermain melawan tim 2 di kandang tim 1, maka pada ronde kedua pertandingan yang sama kembali digelar namun dengan lokasi di kandang tim 2.

Pada ITC 2021, permasalahan utama adalah bagaimana mengalokasikan pertandingan antara dua tim ke dalam slot waktu yang tersedia, di mana jumlah slot waktu sama dengan jumlah pertandingan yang harus dijadwalkan. Dalam menjadwalkan pertandingan, terdapat dua tujuan utama. Tujuan pertama adalah menemukan solusi yang *feasible* tanpa melanggar *hard constraint* yang ditetapkan. Mencari solusi *feasible* pada *benchmark* ITC 2021 merupakan tantangan yang sulit. Hal ini dibuktikan dengan hanya satu penelitian oleh Lamas-Fernandez et al. (2021) yang berhasil menemukan semua solusi *feasible*. Tujuan kedua adalah meminimalkan pelanggaran terhadap *soft constraint*. Hal ini juga menjadi tantangan tersendiri mengingat banyaknya batasan *soft constraint* yang harus dipenuhi serta perlunya menjaga solusi tetap *feasible*.

Bab ini membahas hasil penerapan algoritma Progressive Acceptance Iterated Local Search (PA-ILS) dan Adaptive Threshold-Iterated Local Search (AT-ILS) dalam menyelesaikan permasalahan pada *benchmark* ITC 2021. Pembahasan pertama dalam subbab ini adalah penjabaran karakteristik dari 45 dataset dan format dataset yang dijelaskan pada Subbab 7.1. Selanjutnya, Subbab 7.2 membahas *hard* dan *soft constraint* yang terdapat dalam *benchmark* ITC 2021. Subbab 7.3 membahas hasil dari implementasi pada *benchmark* ITC 2021 dan analisis dari kinerja kedua algoritma. Setelah itu, hasil dari implementasi dibandingkan dengan penelitian terdahulu untuk menilai performa dari kedua algoritma yang dikembangkan. Perbandingan ini disajikan pada Subbab 7.4. Terakhir, Subbab 7.5 menyajikan kesimpulan dari hasil implementasi kedua algoritma.

7.1 Karakteristik Dataset

7.1.1 Karakteristik Dataset Secara Umum

Pada ITC 2021, terdapat 45 dataset yang terbagi dalam 3 kategori: Early, Middle, dan Late, dengan masing-masing kategori terdiri dari 15 dataset. Perbedaan antara ketiga kategori hanya didasarkan pada waktu publikasi dataset tersebut. Seluruh dataset dapat diunduh melalui <https://www.robinxval.ugent.be/ITC2021/instances.php>. Tabel 7.1.1, 7.1.2, dan 7.1.3 menampilkan karakteristik dari setiap dataset yang terdiri dari jumlah tim, slot waktu, *hard constraint*, dan *soft constraint*.

Berdasarkan tabel tersebut, setiap kategori memiliki komposisi tim dan slot waktu yang sama. Dataset 1-3 terdiri dari 16 tim dan 30 slot waktu, dataset 4-9 terdiri dari 18 tim dan 34 slot waktu, dan dataset 10-15 terdiri dari 20 tim dengan 38 slot waktu. Namun, jumlah *hard* dan *soft constraint* bervariasi pada setiap dataset. Beberapa dataset memiliki jumlah *hard constraint* yang cukup tinggi, yaitu di atas 200, seperti Early 1, Early 5, Early 11, Middle 2, Middle 3, Middle 13, Late 1, Late 2, Late 10, dan Late 12. Jumlah *hard constraint* yang tinggi ini berkorelasi dengan tingkat kesulitan dalam menemukan solusi *feasible*, sebagaimana dibuktikan dalam penelitian sebelumnya yang menunjukkan sebagian besar dataset ini memiliki tantangan tinggi untuk menemukan solusi *feasible*.

Sementara pada jumlah *soft constraint*, *benchmark* ITC 2021 juga menghadirkan dataset dengan jumlah *soft constraint* yang bervariasi. Beberapa dataset memiliki *soft constraint* yang jauh lebih tinggi dibandingkan dengan dataset lainnya, seperti Early 7,

Early 8, Early 15, Middle 2, Middle 3, dan Late 14. Sebaliknya, terdapat juga dataset dengan jumlah *soft constraint* yang rendah, seperti pada dataset Early 4, Early 12, Early 14, Late 4, Late 6, dan Late 15. Jumlah *soft constraint* yang banyak tidak selalu menunjukkan bahwa masalah tersebut sulit dioptimasi. Sebaliknya, hal ini dapat memberikan fleksibilitas lebih dalam melakukan optimasi. Sementara itu, pada dataset dengan jumlah *soft constraint* yang rendah, optimasi mungkin menjadi lebih sederhana karena penalti yang perlu diperhatikan lebih sedikit. Namun, hal ini juga bisa menghadirkan tantangan, mengingat ruang solusi yang menjadi lebih luas, berpotensi menyebabkan terbentuknya *plateau* yang menyulitkan algoritma keluar dari kondisi *local optima*.

Tabel 7.1.1: Deskripsi Dataset pada *Benchmark ITC 2021* Kategori Early

<i>Dataset</i>	Tim	Slot Waktu	<i>Hard Constraint</i>	<i>Soft Constraint</i>
1	16	30	83	113
2	16	30	53	114
3	16	30	148	165
4	18	34	164	33
5	18	34	207	256
6	18	34	192	206
7	18	34	175	539
8	18	34	70	525
9	18	34	90	102
10	20	38	246	395
11	20	38	246	488
12	20	38	179	35
13	20	38	100	175
14	20	38	56	56
15	20	38	187	604

Tabel 7.1.2: Deskripsi Dataset pada *Benchmark ITC 2021* Kategori Middle

<i>Dataset</i>	Tim	Slot Waktu	<i>Hard Constraint</i>	<i>Soft Constraint</i>
1	16	30	144	373

(Lanjutan) Deskripsi Dataset pada *Benchmark ITC 2021* Kategori Middle

<i>Dataset</i>	Tim	Slot Waktu	<i>Hard Constraint</i>	<i>Soft Constraint</i>
2	16	30	246	611
3	16	30	237	595
4	18	34	97	168
5	18	34	150	164
6	18	34	162	154
7	18	34	141	121
8	18	34	62	197
9	18	34	94	164
10	20	38	198	351
11	20	38	176	436
12	20	38	63	73
13	20	38	219	108
14	20	38	63	498
15	20	38	95	133

Tabel 7.1.3: Deskripsi Dataset pada *Benchmark ITC 2021* Kategori Late

<i>Dataset</i>	Tim	Slot Waktu	<i>Hard Constraint</i>	<i>Soft Constraint</i>
1	16	30	235	344
2	16	30	246	457
3	16	30	127	113
4	18	34	96	34
5	18	34	176	133
6	18	34	163	34
7	18	34	126	137
8	18	34	110	181
9	18	34	102	152
10	20	38	233	247
11	20	38	52	92
12	20	38	244	389

(Lanjutan) Deskripsi Dataset pada *Benchmark ITC 2021* Kategori Late

<i>Dataset</i>	Tim	Slot Waktu	<i>Hard Constraint</i>	<i>Soft Constraint</i>
13	20	38	169	119
14	20	38	116	603
15	20	38	51	41

7.1.2 Struktur Dataset

Pada ITC 2021, dataset disajikan dalam format XML. Setiap dataset memiliki elemen `MetaData`, `Structure`, `ObjectiveFunction`, `Resources`, dan `Constraints`. Elemen `MetaData`, `Structure`, dan `ObjectiveFunction` menyediakan informasi mengenai dataset tersebut, sementara elemen `Resources` dan `Constraints` merupakan inti dari permasalahan yang dijabarkan dalam dataset.

Elemen `Resources` terdiri dari tiga sub-elemen, yaitu `Leagues`, `Teams`, dan `Slots`. Gambar 7.1.1 menampilkan contoh isi dari elemen `Resources`. Sub-elemen `League` hanya mencakup `id` dan nama kompetisi. Sub-elemen `Team` menyajikan data mengenai tim-tim yang terlibat dalam permasalahan ini. Sedangkan sub-elemen `Slot` menyediakan informasi mengenai slot yang tersedia. Sebagai ilustrasi, contoh pada Gambar 7.1.1 menunjukkan bahwa dataset ini mencakup 6 tim dan 10 slot yang tersedia.

Elemen `Constraints` berisi daftar *hard* dan *soft constraint* dalam dataset. Gambar 7.1.2 memperlihatkan ilustrasi dari elemen ini. Elemen ini memiliki sub-elemen yang terdiri dari kategori batasan yang ada. Setiap kategori batasan mencakup daftar batasan yang berlaku dalam permasalahan ini. Sebagai contoh pada Gambar 7.1.2, dataset ini memiliki lima jenis batasan, yaitu `CapacityConstraints`, `GameConstraints`, `BreakConstraints`, `FairnessConstraints` dan `SeparationConstraints` masing-masing dengan satu batasan. Peran setiap batasan sebagai *hard* atau *soft constraint* ditandai oleh atribut `type` pada masing-masing batasan. Pembahasan lebih lanjut mengenai batasan ini disajikan pada Subbab 7.2.

```

<Resources>
    <Leagues>
        <league id="0" name="League_0"/>
    </Leagues>
    <Teams>
        <team id="0" league="0" name="Team_0"/>
        <team id="1" league="0" name="Team_1"/>
        <team id="2" league="0" name="Team_2"/>
        <team id="3" league="0" name="Team_3"/>
        <team id="4" league="0" name="Team_4"/>
        <team id="5" league="0" name="Team_5"/>
    </Teams>
    <Slots>
        <slot id="0" name="Slot_0"/>
        <slot id="1" name="Slot_1"/>
        <slot id="2" name="Slot_2"/>
        <slot id="3" name="Slot_3"/>
        <slot id="4" name="Slot_4"/>
        <slot id="5" name="Slot_5"/>
        <slot id="6" name="Slot_6"/>
        <slot id="7" name="Slot_7"/>
        <slot id="8" name="Slot_8"/>
        <slot id="9" name="Slot_9"/>
    </Slots>
</Resources>

```

Gambar 7.1.1: Ilustrasi Struktur Elemen Resources Pada ITC 2021

7.2 Hard dan Soft Constraint

Pada ITC 2021, terdapat lima jenis batasan yang dibentuk berdasarkan lima aspek, yaitu kapasitas, pertandingan, waktu istirahat, keadilan jumlah bermain kandang, dan jarak pertandingan. Batasan kapasitas, pertandingan, dan waktu istirahat dapat digunakan sebagai *hard* atau *soft constraint* yang bervariasi pada setiap dataset. Sementara itu, batasan keadilan jumlah bermain kandang dan jarak pertandingan hanya digunakan sebagai *soft constraint*. Detail dari kelima jenis batasan dijabarkan sebagai berikut:

1. **Kapasitas (CA):** Batasan ini dirancang untuk mengatur jumlah bermain satu atau beberapa tim terhadap lokasi pertandingan, tim lain dan slot waktu. Kategori batasan CA memiliki empat jenis variasi batasan:
 - CA1: Batasan ini membatasi jumlah total pertandingan kandang atau tandang yang dimainkan oleh tim tertentu dalam satu set slot waktu yang ditentukan. Sebagai contoh, pada Gambar 7.1.2 baris ke-3, batasan ini menetapkan bahwa

```

1 <Constraints>
2   <CapacityConstraints>
3     <CA1 max="1" mode="H" slots="9;6;7" teams="0" type="HARD"/>
4     <CA2 max="1" min="0" model="HA" mode2="GLOBAL" penalty="5"
5       slots="1;3;5;6;7;8;9" teams1="4" teams2="1;0;5;2;3"
6       type="SOFT"/>
7     <CA3 intp="4" max="2" min="0" model="A" mode2="SLOTS"
8       penalty="5" teams1="2" teams2="0;3;5;4;1" type="SOFT"/>
9     <CA4 max="1" min="0" model="H" mode2="GLOBAL" penalty="1"
10       slots="2;3;1;0" teams1="4;3;1;2" teams2="4;3;1;2"
11       type="HARD"/>
12   </CapacityConstraints>
13   <GameConstraints>
14     <GA1 max="0" meetings="2,3;" min="0" penalty="1" slots="5"
15       type="SOFT"/>
16   </GameConstraints>
17   <BreakConstraints>
18     <BR1 intp="0" model="LEQ" mode2="HA" penalty="5" slots="4"
19       teams="0" type="SOFT"/>
20     <BR2 intp="16" homeMode="HA" mode2="LEQ" penalty="1"
21       slots="9;2;3;4;5;6;7;8;1;0" teams="2;0;3;5;4;1" type="HARD"/>
22   </BreakConstraints>
23   <FairnessConstraints>
24     <FA2 intp="2" mode="H" penalty="10"
25       slots="1;2;3;4;5;6;7;8;9;0" teams="2;0;1;4;3;5" type="SOFT"/>
26   <FairnessConstraints/>
27   <SeparationConstraints>
28     <SE1 model="SLOTS" min="10" penalty="10" teams="2;0;3;5;4;1"
29       type="SOFT"/>
30   </SeparationConstraints>
31 </Constraints>

```

Gambar 7.1.2: Ilustrasi Struktur Elemen Constraints Pada ITC 2021

tim 0 hanya boleh bermain kandang maksimal 1 kali secara keseluruhan di slot waktu ke-9, ke-6, dan ke-7. Mode "H" (Home) menunjukkan bahwa batasan ini hanya berlaku untuk pertandingan kandang. Artinya, jika tim 0 memainkan pertandingan kandang pada slot ke-6 dan ke-9, maka batasan ini dilanggar karena jumlah total pertandingan kandang dalam slot-slot tersebut melebihi batas yang diperbolehkan.

- CA2: Batasan ini membatasi jumlah total pertandingan antara tim dari dua kelompok dalam keseluruhan slot waktu yang ditentukan. Sebagai contoh, pada Gambar 7.1.2 baris ke-4 hingga baris ke-6, batasan ini mengatur bahwa tim 4 hanya boleh bertanding maksimal satu kali melawan tim-tim dalam teams2 (yaitu tim 1, 0, 5, 2, dan 3) dalam keseluruhan slot waktu yang

disebutkan (slot ke-1, ke-3, ke-5, ke-6, ke-7, ke-8, dan ke-9). Mode mode1="HA" menunjukkan bahwa pembatasan ini berlaku untuk pertandingan kandang dan tandang. Dengan kata lain, pertandingan antara tim 4 dan tim-tim dalam teams2 hanya boleh terjadi sekali di antara semua slot yang disebutkan secara keseluruhan.

- CA3: Batasan ini membatasi jumlah pertandingan yang dimainkan oleh suatu tim dalam mode tertentu (kandang atau tandang) dalam jangka slot waktu tertentu. Sebagai contoh, pada Gambar 7.1.2 baris ke-7 dan baris ke-8, tim 2 hanya boleh memainkan maksimal dua pertandingan tandang melawan tim-tim yang disebutkan dalam jangka empat slot waktu (intp="4"). Mode mode1="A" menunjukkan bahwa pembatasan ini hanya berlaku untuk pertandingan tandang. Artinya, batasan ini akan mengecek setiap 4 slot waktu (slot waktu 1-4, 2-5, 3-6, dan seterusnya) dan memastikan bahwa tim 2 hanya bermain maksimal 2 kali dalam mode tandang melawan tim 0, 3, 5, 4, dan 1. Misalnya, jika tim 2 melawan tim 0 pada slot 1, melawan tim 5 pada slot 2, dan melawan tim 4 pada slot 3, dan semua pertandingan dilakukan dalam mode tandang, maka batasan ini dilanggar karena dalam 4 slot waktu, tim 2 bermain 3 kali melawan tim yang tercantum dalam batasan ini.
- CA4: Batasan ini membatasi jumlah pertandingan antara dua kelompok tim dalam satu set slot waktu. Terdapat dua mode: mode "*Global*" yang menghitung jumlah pertandingan secara keseluruhan dalam slot yang ditentukan, dan mode "*Every*" yang menghitung jumlah pertandingan secara individual pada setiap slot. Sebagai contoh, pada Gambar 7.1.2 baris ke-9 hingga baris ke-11, batasan ini menetapkan bahwa tim-tim dalam teams1 dapat bertanding kandang maksimal satu kali melawan tim-tim dalam teams2 di seluruh slot waktu yang ditentukan. Karena mode yang digunakan adalah "*Global*", maka perhitungan dilakukan secara keseluruhan pada slot yang terdaftar. Jika terdapat pertandingan antara tim 1 dan tim 4 pada slot 2 dan tim 2 melawan tim 3 pada slot 3, maka jadwal ini telah melanggar batasan ini karena jumlah pertandingan dalam slot yang disebutkan melebihi batas maksimal 1.

2. **Pertandingan (GA):** Pada batasan ini, hanya terdapat satu jenis batasan yang disebut GA1, yang berfungsi untuk mengatur jumlah pertemuan antara dua tim tertentu dalam slot waktu yang ditentukan. Sebagai contoh, pada Gambar 7.1.2 baris ke-14 dan baris ke-15, batasan ini menetapkan bahwa tim 2 dan tim 3 tidak boleh bertemu pada slot waktu ke-5.
3. **Waktu Istirahat (BR):** Batasan ini membatasi jumlah istirahat untuk satu atau beberapa tim. Istirahat didefinisikan sebagai kondisi di mana tidak ada perubahan mode bertanding antara dua slot waktu berurutan. Sebagai contoh, jika tim 1 bermain kandang pada slot 2 dan 3, maka pada slot 3 tim 1 dikategorikan sedang beristirahat. Hal ini juga berlaku jika suatu tim memainkan pertandingan tandang secara berurutan. Batasan istirahat memiliki dua jenis batasan:
 - BR1: Batasan ini dirancang untuk membatasi jumlah istirahat bagi tim tertentu dalam mode pertandingan tertentu pada slot waktu tertentu. Sebagai contoh, pada Gambar 7.1.2 baris ke-18 dan baris ke-19, batasan ini menetapkan bahwa tim dengan indeks 0 tidak boleh memiliki istirahat pada slot waktu ke-4 (`intp="0"` menunjukkan jumlah istirahat yang diperbolehkan). Mode "HA" menunjukkan batasan ini berlaku untuk bermain kandang dan tandang. Artinya, tim 0 tidak boleh memiliki mode pertandingan yang sama antara pertandingan di slot waktu ke-3 dan ke-4.
 - BR2: Batasan ini dirancang untuk mengatur jumlah istirahat dalam jadwal pertandingan bagi sekelompok tim pada rentang slot waktu tertentu. Sebagai contoh, pada konfigurasi pada Gambar 7.1.2 baris ke-20 dan baris ke-21, batasan ini menetapkan bahwa tim-tim dengan indeks 2, 0, 3, 5, 4, dan 1 tidak boleh memiliki lebih dari 16 istirahat di seluruh slot waktu yang disebutkan, dihitung secara keseluruhan untuk semua tim dan slot yang tercantum.
4. **Keadilan Jumlah Bermain Kandang (FA):** Pada batasan ini hanya terdapat satu jenis batasan, yaitu FA2, yang memastikan keseimbangan jumlah pertandingan kandang bagi tim-tim tertentu selama rentang slot waktu yang ditentukan.

Sebagai contoh, pada batasan pada Gambar 7.1.2 baris ke-24 dan baris ke-25, batasan ini menetapkan bahwa jumlah pertandingan kandang yang dimainkan oleh tim-tim yang disebutkan dalam `teams` pada slot waktu yang disebutkan dalam `slots` tidak

lebih dari 2. Sebagai contoh, jika tim 2 bermain kandang pada slot waktu ke-1, dan tim 1 bermain kandang pada slot waktu ke-1 hingga ke-4, maka batasan ini dilanggar karena perbedaan jumlah pertandingan kandang antara tim-tim tersebut melebihi 2.

5. **Jarak Pertandingan (SE):** Pada batasan ini hanya terdapat satu jenis batasan, yaitu SE1, yang memastikan adanya jarak minimum antara pertemuan pertama dan kedua antara dua tim dalam satu musim. Sebagai contoh, pada Gambar 7.1.2 baris ke-28 dan baris ke-29, batasan ini menetapkan bahwa tim-tim yang disebutkan harus memiliki jarak minimal 10 slot waktu antara pertemuan pertama dan kedua mereka. Jika tim 2 bertemu tim 0 di slot waktu ke-1 (dengan tim 2 sebagai tuan rumah), maka pertemuan berikutnya antara tim 0 dan tim 2 (dengan status tuan rumah berganti) hanya dapat dijadwalkan paling cepat pada slot waktu 11.

7.3 Hasil Implementasi dan Analisis Algoritma

Pada *benchmark* ITC 2021, implementasi algoritma dimulai dengan tahap pencarian solusi *feasible* untuk setiap dataset. Setelah solusi *feasible* ditemukan, proses dilanjutkan dengan tahapan optimasi solusi. Tahapan optimasi menggunakan input dari hasil solusi *feasible* yang dipilih secara acak. Hasil dari tahapan pencarian solusi *feasible* dan optimasi divalidasi dengan mengunggah solusi ke situs web resmi validator (<http://robinxval.ugent.be/ITC2021/validator.php>). Proses ini bertujuan untuk memastikan bahwa solusi yang dihasilkan memenuhi kriteria *feasible* dan nilai penalti yang dihitung oleh algoritma sesuai dengan hasil perhitungan validator.

Penggunaan Low-Level Heuristic (LLH) pada tahapan pencarian solusi *feasible* menerapkan tiga jenis LLH yaitu *move*, *swap* dan *kempe chain*. Sementara pada tahapan optimasi, hanya LLH *swap* dan *kempe chain* yang digunakan. LLH *move* tidak dapat digunakan dalam tahapan optimasi karena jumlah pertandingan yang harus dijadwalkan sama dengan jumlah slot waktu yang tersedia. Pada kondisi *feasible*, seluruh slot waktu pasti sudah diisi dengan satu pertandingan, sehingga tidak memungkinkan untuk memindahkan satu pertandingan ke slot waktu yang masih kosong.

Hasil dari tahapan pencarian solusi *feasible* disajikan pada Subbab 7.3.1 dengan analisis algoritma PA-ILS yang disajikan dalam Subbab 7.3.2. Sementara hasil dan analisis dari tahapan optimasi disajikan pada Subbab 7.3.3 dan 7.3.4.

7.3.1 Hasil Tahapan Pencarian Solusi *Feasible*

Pada bagian ini, algoritma PA-ILS diterapkan pada tahapan pencarian solusi *feasible* pada 45 dataset. Hasil implementasi disajikan pada tabel 7.3.1 dengan menampilkan waktu rata-rata, tercepat dan terlama dalam memperoleh solusi *feasible* serta menampilkan persentase keberhasilan dari 10 kali uji coba. Hasil ini menunjukan algoritma yang dikembangkan mampu menghasilkan solusi *feasible* pada 42 dataset dengan persentase keberhasilan 100%. Pada 20 dataset, algoritma mampu menemukan solusi *feasible* dengan rata-rata waktu kurang dari 1 menit dan pada lainnya 14 dataset dengan rata-rata waktu kurang dari 15 menit. Sementara itu, delapan dataset lainnya membutuhkan waktu yang lebih lama dengan rentang waktu 86 hingga 1691 menit.

Untuk tiga dataset lainnya yaitu Middle 2, Middle 3, dan Late 2, algoritma mampu menghasilkan solusi *feasible* namun dengan persentase keberhasilan yang lebih kecil. Pada dataset Middle 2, hanya satu solusi *feasible* yang diperoleh dari sepuluh kali percobaan dengan durasi yang jauh lebih lama (sekitar 113 jam) dibandingkan dengan dataset lainnya. Untuk memastikan bahwa temuan satu solusi *feasible* tersebut bukan kebetulan, maka penelitian ini mengulang uji coba sebanyak 10 kali pada dataset tersebut. Percobaan kedua ini menghasilkan hasil yang sama yaitu hanya menemukan satu solusi *feasible* dengan durasi sekitar 60 jam. Sementara itu, pada dataset Middle 3, algoritma menunjukkan kinerja yang sedikit lebih baik dengan menghasilkan dua solusi *feasible* dari sepuluh percobaan, dengan waktu rata-rata sekitar 60 jam. Pada dataset Late 2, algoritma menunjukkan kinerja yang lebih baik, menghasilkan tiga solusi *feasible* dari sepuluh percobaan, dengan waktu rata-rata sekitar 72 jam.

Tabel 7.3.1: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada Benchmark ITC 2021 Kategori Early

Dataset	Waktu (Detik)			Persentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
1	9	5	20	100%
2	94	26	197	100%
3	4	2	13	100%
4	771	214	1368	100%
5	101449	6511	270198	100%

(Lanjutan) Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2021* Kategori Early

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
6	759	141	3420	100%
7	37665	1096	176080	100%
8	3	2	4	100%
9	3	2	4	100%
10	76004	7609	253681	100%
11	42315	12676	167789	100%
12	45	11	86	100%
13	130	25	328	100%
14	7	3	22	100%
15	465	139	874	100%

Tabel 7.3.2: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2021* Kategori Middle

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
1	5114	472	23255	100%
2	310796	215520	406073	10%
3	215389	183895	246883	20%
4	7	4	8	100%
5	4	3	5	100%
6	102	43	172	100%
7	81	40	188	100%
8	6	4	10	100%
9	5	4	7	100%
10	432	175	634	100%
11	126	25	208	100%
12	5	2	9	100%
13	62	20	179	100%

(Lanjutan) Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2021* Kategori Middle

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
14	395	38	1081	100%
15	3	2	6	100%

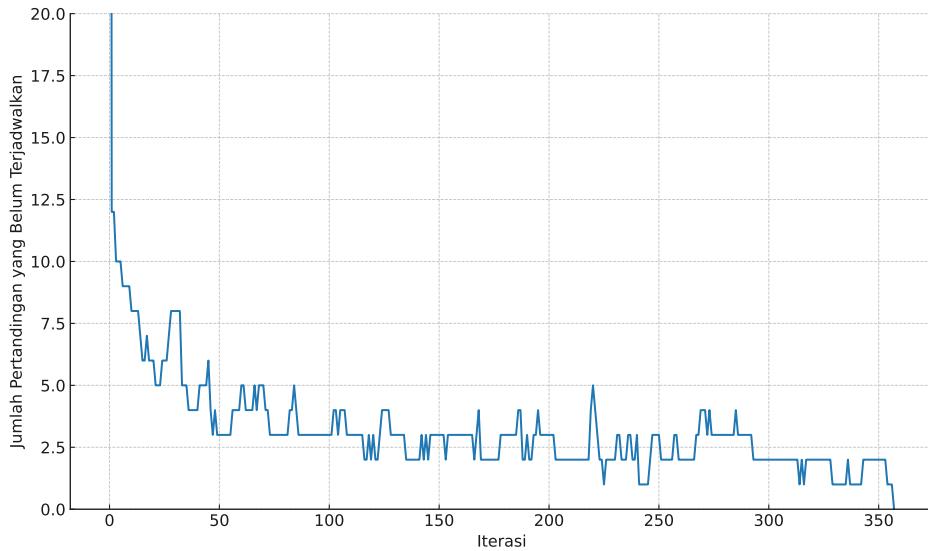
Tabel 7.3.3: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada *Benchmark ITC 2021* Kategori Late

Dataset	Waktu (Detik)			Percentase Solusi <i>Feasible</i>
	Rata-Rata	Minimum	Maksimum	
1	554	162	2338	100%
2	259691	227688	311365	30%
3	3	3	4	100%
4	3	2	4	100%
5	42527	5235	148768	100%
6	9	5	17	100%
7	142	61	491	100%
8	3	2	7	100%
9	10	7	12	100%
10	19632	1977	43941	100%
11	20	7	44	100%
12	26178	1766	53887	100%
13	217	44	449	100%
14	19	7	48	100%
15	3	2	4	100%

7.3.2 Analisis Proses Algoritma PA-ILS

Untuk melihat bagaimana algoritma PA-ILS bekerja dalam proses mencari solusi *feasible* pada *benchmark ITC 2021*, Gambar 7.3.1 menyajikan visualisasi dari pergerakan solusi pada setiap iterasinya berdasarkan jumlah pertandingan yang belum berhasil

dijadwalkan. Visualisasi ini menunjukkan algoritma masih bekerja sesuai dengan strategi yang dikembangkan. Algoritma mampu mengarahkan proses pencarian menuju konvergensi. Selain itu, algoritma juga menjaga aspek eksplorasi dengan memungkinkan penerimaan solusi yang kurang optimal.



Gambar 7.3.1: Grafik Perubahan Solusi dalam Proses Pencarian Solusi *Feasible* pada *Benchmark ITC 2021*

Dari Visualisasi tersebut, dapat dilihat bahwa eksplorasi berjalan lebih masif jika dibandingkan dengan *benchmark* Toronto maupun ITC 2019. Proses eksplorasi dalam tahapan *perturbation* mengharuskan penerimaan solusi meskipun dalam kondisi yang lebih buruk saat mencoba menjadwalkan sebuah pertandingan. Solusi yang kurang optimal ini akan dicoba ditingkatkan kembali dalam tahapan *local search*. Hasil dari tahapan *local search* tidak menjamin akan menghasilkan solusi yang sama atau lebih baik dari solusi yang dihasilkan pada iterasi sebelumnya. Dengan demikian, hasil akhir dari tahapan *local search* memungkinkan untuk menghasilkan solusi yang kurang optimal dalam beberapa iterasi.

Jika dibandingkan dengan *benchmark* Toronto dan ITC 2019, kedua *benchmark* tersebut menunjukkan cukup banyak menghasilkan solusi yang sama dalam beberapa iterasi (lihat Gambar 5.3.1 dan 6.3.1). Hal ini menunjukan perbedaan karakteristik permasalahan ITC 2021 dengan dua permasalahan lainnya. ITC 2021 memiliki jumlah slot waktu yang sama persis dengan jumlah pertandingan yang dijadwalkan. Ketika melakukan penjadwalan suatu pertandingan, sangat memungkinkan untuk menghasilkan konflik yang

lebih banyak dengan pertandingan lainnya yang sudah dijadwalkan. Berbeda dengan permasalahan pada *benchmark* Toronto dan ITC 2019 yang memiliki jumlah slot waktu lebih banyak dibandingkan dengan jumlah ujian atau mata kuliah yang ingin dijadwalkan. Hal ini menyebabkan fleksibilitas yang lebih tinggi dalam menjadwalkan suatu ujian atau mata kuliah dan menghasilkan kemungkinan konflik yang lebih sedikit.

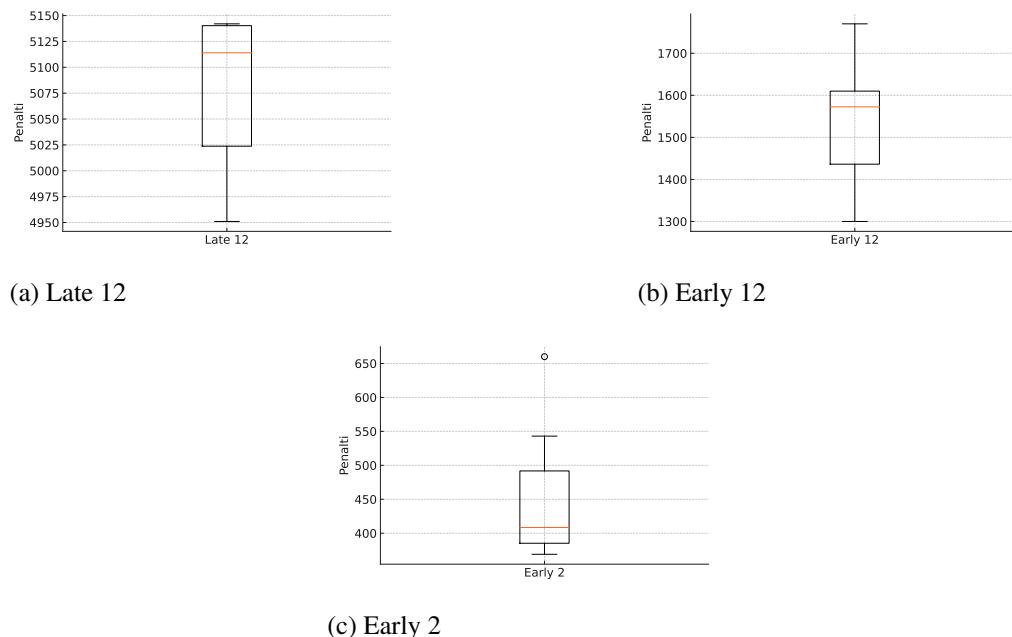
7.3.3 Hasil Tahapan Optimasi

Pada bagian ini, solusi *feasible* yang telah dihasilkan pada tahapan sebelumnya dioptimasi untuk menurunkan nilai penalti. Proses optimasi dijalankan dalam dua tahapan. Tahapan pertama melakukan optimasi sebanyak 10 kali pada setiap dataset dengan batas waktu setiap uji coba adalah 10 jam. Pada tahapan ini, pemilihan solusi *feasible* dilakukan secara acak dari 10 solusi *feasible* yang dihasilkan dalam tahapan sebelumnya. Sementara itu, tahapan kedua melakukan optimasi dengan tujuan untuk mencari solusi terbaik. Tahapan ini dijalankan dengan mengoptimasi kembali hasil terbaik yang diperoleh pada tahapan pertama. Tahapan ini mengevaluasi solusi yang dihasilkan setiap 10 jam. Jika dalam waktu tersebut tidak menghasilkan solusi terbaik baru, maka tahapan ini akan berakhir.

Hasil optimasi pada kedua tahapan ditampilkan pada Tabel 7.3.4, 7.3.5 dan 7.3.6. Pada tahapan pertama, tabel menampilkan nilai penalti rata-rata, terbaik, terburuk, serta nilai Koefisien Variasi (KV). Hasil implementasi menunjukkan konsistensi yang relatif baik. Berdasarkan nilai KV, 38 dataset memiliki nilai KV dibawah 10% dan 6 dataset dengan rentang 10 hingga 15%. Hanya satu dataset yang memiliki nilai KV yang buruk yaitu Early 2 dengan nilai KV sebesar 21,03%. Hasil ini jelas lebih tinggi dibandingkan dengan hasil uji coba pada *benchmark* Toronto namun sedikit lebih baik dari hasil uji coba pada *benchmark* ITC 2019. Hal ini dikarenakan kompleksitas yang berbeda jauh antara *benchmark* Toronto dengan *benchmark* ITC 2019 dan ITC 2021. Pada *benchmark* ITC 2021 memiliki banyak *hard* dan *soft constraint* yang membatasi proses pencarian. Selain itu pada permasalahan ini juga memiliki fleksibilitas slot waktu yang sangat terbatas dengan jumlah pertandingan dan slot waktu yang sama.

Untuk melihat lebih detail dari keberagaman solusi yang dihasilkan, Gambar 7.3.2 menampilkan box plot dari dataset Late 12 yang merupakan dataset dengan hasil konsistensi yang baik dengan nilai KV 1,37%, dataset Early 12 sebagai dataset dengan

tingkat konsistensi sedang dengan nilai KV 9,84%, dan dataset Early 2 yang merupakan dataset dengan hasil paling tidak konsisten dengan nilai KV 21,03%. Dari Gambar 7.3.2c, nilai KV yang buruk pada Early 2 dihasilkan dari adanya satu *outlier*. Jika *outlier* tersebut dihilangkan, maka dataset Early 2 memiliki nilai KV yang lebih baik yaitu 14,51%. Sementara pada dataset Early 12, Gambar 7.3.2b menunjukkan bentuk box plot menunjukkan persebaran nilai penalti yang cukup seimbang tanpa adanya nilai ekstrem yang memengaruhi hasil perhitungan nilai KV secara signifikan. Sementara pada dataset Early 2, walaupun menyajikan bentuk box plot yang menunjukkan kebanyakan variasi data berada dibawah rata-rata, namun box plot ini berada pada rentang penalti yang kecil yang menunjukkan hasil yang konsisten dari dataset ini.



Gambar 7.3.2: Box plot Hasil Algoritma Optimasi pada Tiga Dataset: Late 12, Early 12 dan Early 2

Sementara itu pada hasil optimasi tahapan kedua, kebanyakan dataset berhasil ditemukan solusi terbaik baru dengan rentang durasi antara 20 hingga 60 jam. Namun terdapat beberapa dataset seperti Early 5, Early 7 dan Early 8 yang tidak berhasil menghasilkan solusi terbaik baru. Dari hasil ini menunjukkan bahwa algoritma yang dikembangkan menunjukkan efisiensi yang baik terutama dari sisi praktikal. Pada pembentukan solusi penjadwalan dalam kompetisi olah raga, umumnya memiliki waktu untuk membentuk penjadwalan dalam beberapa minggu. Dari hasil ini menunjukkan hasil

terbaik paling lama dihasilkan dalam 60 jam. Hal ini menunjukan algoritma ini dapat digunakan secara praktikal dari sisi waktu eksekusi.

Tabel 7.3.4: Hasil Uji Coba Tahapan Optimasi pada *Benchmark ITC 2021* Kategori Early

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
1	583,1	474	690	10,29	388	60 jam
2	448,6	369	660	21,03	354	20 jam
3	1242,9	1175	1297	3,52	1161	20 jam
4	1711,1	1367	1974	10,98	1359	20 jam
5	4184,8	4008	4560	4,07	4008	10 jam
6	5445,3	4903	5864	6,59	4276	20 jam
7	6129,6	5826	6431	4,20	5826	10 jam
8	1713,1	1612	1776	2,70	1612	10 jam
9	976,2	863	1048	5,94	848	20 jam
10	4622,7	4296	5138	5,46	4296	10 jam
11	5366,3	5006	5723	4,10	4290	20 jam
12	1546,5	1300	1770	9,48	1300	10 jam
13	742,8	659	893	12,08	659	10 jam
14	951,9	837	1034	6,65	837	10 jam
15	5646,9	5442	5963	2,89	5335	20 jam

Tabel 7.3.5: Hasil Uji Coba Tahapan Optimasi pada *Benchmark ITC 2021* Kategori Middle

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
1	6502,3	6240	6920	3,98	6240	10 jam
2	7143,2	7115	7256	0,83	7115	10 jam
3	8694,8	8648	8726	0,46	8648	10 jam

(Lanjutan) Hasil Uji Coba Tahapan Optimasi pada *Benchmark* ITC 2021 Kategori Middle

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
4	32,2	26	37	12,55	21	40 jam
5	742,3	658	825	8,14	653	20 jam
6	3007,5	2430	3375	12,33	2130	50 jam
7	2502,0	2258	2998	8,63	2237	20 jam
8	469,0	407	533	8,67	302	50 jam
9	1513,0	1355	1625	5,44	1195	20 jam
10	2089,7	1869	2231	6,12	1869	20 jam
11	3473,4	3383	3618	2,28	3237	30 jam
12	1447,9	1385	1527	3,11	1354	20 jam
13	608,7	546	687	7,14	521	20 jam
14	2523,7	2297	2703	5,63	2216	40 jam
15	1209,9	1139	1318	5,02	1099	30 jam

Tabel 7.3.6: Hasil Uji Coba Tahapan Optimasi pada *Benchmark* ITC 2021 Kategori Late

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
1	3225,9	2804	3630	8,81	2804	10 jam
2	6403,8	6270	6493	1,80	6270	10 jam
3	2728,1	2564	2854	3,54	2564	10 jam
4	0,0	0	0	0	0	10 jam
5	2589,7	2503	2654	1,57	2503	10 jam
6	1243,7	1171	1349	4,50	1130	20 jam
7	2849,0	2255	3394	13,35	2255	10 jam
8	1251,0	1168	1367	6,28	1138	20 jam

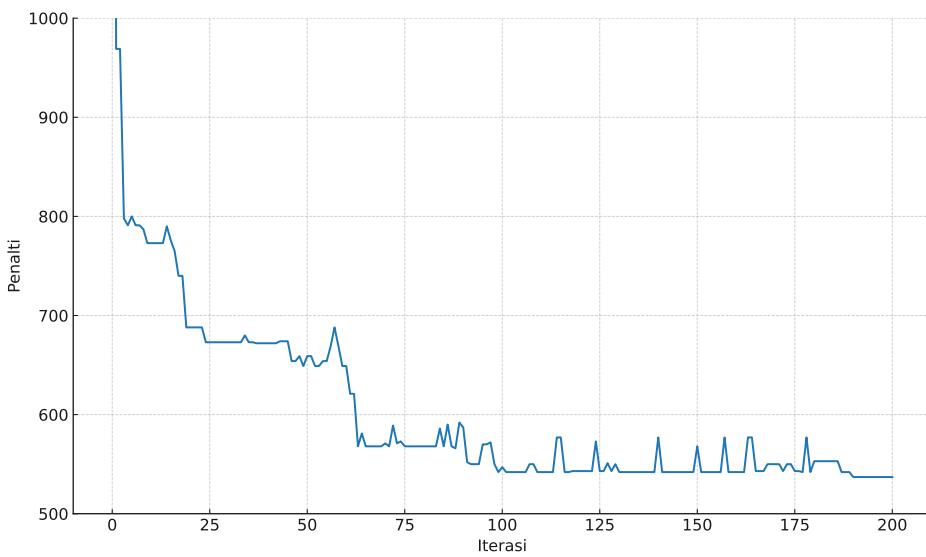
(Lanjutan) Hasil Uji Coba Tahapan Optimasi pada *Benchmark* ITC 2021 Kategori Late

Dataset	Waktu Eksekusi (10 Jam)				Waktu Eksekusi (>10 Jam)	
	Rata-rata	Minimum	Maksimum	KV (%)	Hasil Terbaik	Waktu Eksekusi
9	1885,9	1584	2068	7,71	1574	20 jam
10	3265,7	3024	3665	5,31	2964	20 jam
11	367,4	326	396	6,77	321	20 jam
12	5082,7	4951	5142	1,37	4951	10 jam
13	2279,4	1960	2555	8,13	1960	10 jam
14	2252,2	2160	2355	2,86	2160	10 jam
15	1021,0	900	1115	6,12	870	10 jam

7.3.4 Analisis Proses Algoritma AT-ILS

Untuk melihat bagaimana algoritma AT-ILS bekerja dalam mengoptimasi *benchmark* ITC 2021, Gambar 7.3.3 memvisualisasikan pergerakan solusi pada setiap iterasinya. Dari hasil visualisasi ini, dapat dilihat bahwa algoritma bekerja sesuai dengan strategi yang dikembangkan. Algoritma mampu mengarahkan solusi menuju kondisi konvergensi namun tetap memungkinkan penerimaan solusi yang kurang optimal untuk menjaga eksplorasi solusi. Selain itu algoritma juga mengembalikan solusi pada hasil terbaik ketika dalam beberapa iterasi tidak dapat menemukan solusi terbaik baru.

Pada tahapan *local search*, seperti yang ditunjukkan pada Gambar 7.3.4, algoritma melakukan proses pencarian dengan menerima solusi yang lebih baik dan lebih buruk secara bergantian. Namun, penggunaan ambang batas menekan solusi untuk mencapai konvergensi. Nilai ambang batas memungkinkan kenaikan untuk memastikan algoritma tidak terjebak dalam kondisi *local optima*. Secara keseluruhan, seluruh proses optimasi berjalan sesuai dengan strategi yang dikembangkan.

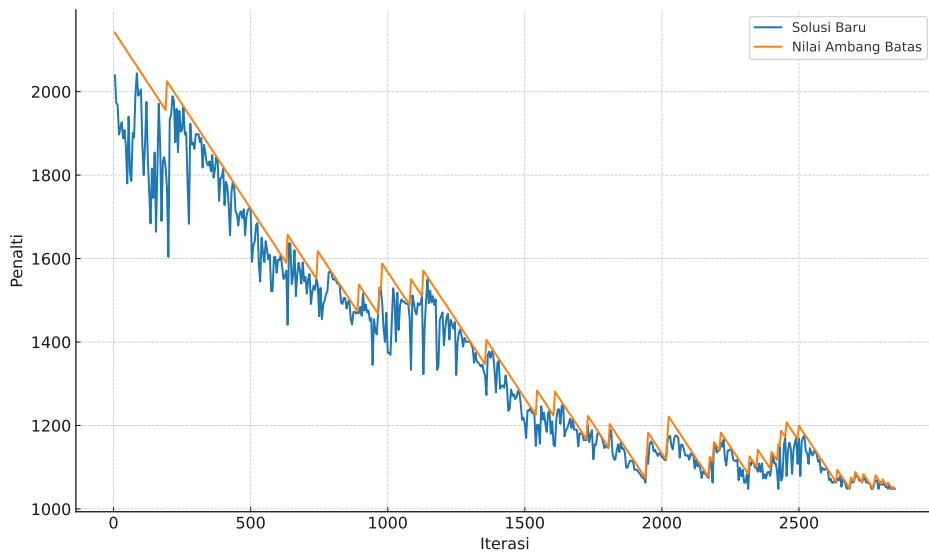


Gambar 7.3.3: Grafik Perubahan Solusi dalam Proses Optimasi pada *Benchmark ITC 2021*

7.4 Perbandingan dengan Studi Sebelumnya

Untuk menilai bagaimana performa algoritma yang dikembangkan, hasil dari penelitian ini dibandingkan dengan hasil dari penelitian sebelumnya. Beberapa penelitian hanya menampilkan hasil feasibilitas dalam publikasinya dan tidak melakukan proses optimasi, sehingga perbandingan dilakukan dalam dua tahapan dengan jumlah penelitian yang berbeda. Tahapan pertama dilakukan dari sisi feasibilitas dengan melakukan perbandingan terhadap 14 penelitian lainnya. Tahapan kedua membandingkan hasil keseluruhan dengan delapan penelitian sebelumnya.

Pada tahapan pertama, Tabel 7.4.1 menyajikan hasil perbandingan pada aspek feasibilitas dengan menampilkan jumlah solusi *feasible* pada setiap kategori dan total keseluruhan jumlah solusi *feasible*. Beberapa penelitian hanya menyajikan total solusi *feasible* tanpa membagi per kategori, sehingga untuk penelitian tersebut diberikan tanda “-” pada kolom kategori Early, Middle, dan Late. Dari hasil ini, jelas terlihat bahwa hasil dari penelitian ini mampu unggul dibandingkan 13 penelitian lainnya dan juga menyamai hasil terbaik pada penelitian oleh Lamas-Fernandez et al. (2021). Hasil ini menunjukkan efektivitas algoritma yang dikembangkan untuk permasalahan yang kompleks seperti pada *benchmark* ITC 2021. Penelitian lainnya terlihat kesulitan dalam menghasilkan solusi *feasible*, ditunjukan dengan 12 penelitian yang menghasilkan solusi *feasible* hanya pada 19



Gambar 7.3.4: Grafik Perubahan Solusi dalam Proses *Local Search* pada Benchmark ITC 2021

hingga 40 dataset.

Hanya dua penelitian sebelumnya yang menghasilkan hasil yang sangat baik, yakni Lamas-Fernandez et al. (2021) dan Rosati et al. (2022). Penelitian oleh Lamas-Fernandez et al. (2021) mampu menghasilkan solusi *feasible* pada seluruh dataset. Namun, metode yang digunakan adalah ILP (Integer Linear Programming), dimana metode ini merupakan metode *exact* yang dikembangkan secara spesifik untuk satu permasalahan. Pada penelitian ini juga tidak dijelaskan bagaimana persentase keberhasilannya. Sementara itu, penelitian oleh Rosati et al. (2022) menghasilkan solusi *feasible* pada 44 dataset. Penelitian ini menampilkan persentase keberhasilan, dimana hasilnya 36 penelitian menghasilkan persentase keberhasilan 100%, dan secara keseluruhan menghasilkan persentase keberhasilan sebesar 83%. Hasil ini jelas menunjukkan bahwa algoritma yang dikembangkan dalam penelitian ini lebih efektif dalam menghasilkan solusi *feasible* dengan persentase keberhasilan 100% pada 42 dataset dan secara keseluruhan mampu menghasilkan persentase keberhasilan sebesar 94,7%.

Selanjutnya, pada tahapan kedua, hasil perbandingan ditampilkan dalam Tabel 7.4.2 dan distribusi pemeringkatan ditampilkan dalam grafik *scatter plot* pada Gambar 7.4.1, 7.4.2, 7.4.3, dan 7.4.4. Dari hasil ini, penelitian ini menunjukkan performa yang cukup baik secara keseluruhan dengan menempati peringkat ke-4 dari 9 penelitian. Selain itu,

secara konsistensi, algoritma ini menunjukkan hasil yang baik dimana dari ketiga kategori, hasil peringkat rata-rata yang dihasilkan cukup stabil dengan rentang 4,4 hingga 4,8.

Tabel 7.4.1: Perbandingan Solusi *Feasible* dengan Penelitian Sebelumnya Pada *Benchmark ITC 2021*

Peneliti	Early	Middle	Late	Total
Penelitian Ini	15	15	15	45
Lamas-Fernandez et al. (2021)	15	15	15	45
Rosati et al. (2022)	15	14	15	44
Berthold et al. (2021)	13	13	14	40
Subba and Stordal (2021)	13	13	14	40
Giorgio Sartor & Bjørnar Luteberget	-	-	-	40
van Doornmalen et al. (2021)	-	-	-	38
Sumin and Rodin (2021)	-	-	-	37
Fonseca and Toffolo (2022)	12	13	12	37
Phillips et al. (2021)	12	13	12	37
Dimitsas et al. (2022)	12	12	13	37
Lester (2022)	11	11	11	33
Arbaoui Taha et al.	-	-	-	31
Swarup Ghadiali	-	-	-	26
Hutama and Muklason (2021)	6	5	8	19

Untuk melihat detail perbandingan dari setiap dataset, Tabel 7.4.3, 7.4.4, dan 7.4.5 menyajikan perbandingan nilai penalti dari setiap solusi pada lima posisi teratas. Hasil perbandingan lengkap dapat dilihat pada Lampiran C. Dari hasil ini, dapat dilihat bahwa penelitian ini berhasil menghasilkan solusi terbaik baru pada dua dataset, yaitu Middle 2 dan 3. Selain itu, jika dibandingkan dengan penelitian terbaik oleh Lamas-Fernandez et al. (2021), terdapat beberapa dataset yang menghasilkan solusi dengan nilai penalti yang tidak terlalu jauh seperti pada Early 1, 11, Middle 7, 15, Late 2, 3, 4, 5, 6, dan 13. Namun, pada dataset lainnya seperti Early 4, 9, 12, 13, 14, Middle 4, 8, 9, 13, Late 9, dan 15, hasil dari penelitian ini masih memiliki perbedaan penalti yang signifikan.

Jika dibandingkan dengan penelitian dengan metode yang sama, yaitu pada penelitian

oleh Rosati et al. (2022) yang mengembangkan algoritma Simulated Annealing, secara keseluruhan perbedaan dihasilkan tidak signifikan dari penelitian oleh Lamas-Fernandez et al. (2021). Bahkan pada beberapa dataset seperti Early 1, 4, 10, Middle 2, 3, dan Late 13, penelitian ini mampu menghasilkan solusi yang lebih baik. Selain itu perlu diperhatikan, algoritma yang dikembangkan oleh Rosati et al. (2022) merupakan algoritma yang sangat spesifik dikembangkan terhadap permasalahan ITC 2021. Hal ini ditunjukkan dengan adanya 22 nilai parameter yang harus ditemukan nilai yang sesuai dengan permasalahan ini, serta pengembangan 6 LLH yang juga ditujukan khusus terhadap permasalahan ini.

Tabel 7.4.2: Perbandingan Peringkat Rata-Rata pada *Benchmark* ITC 2021

Peneliti	Early	Middle	Late	Total
Lamas-Fernandez et al. (2021)	1,20	1,60	1,63	1,47
Rosati et al. (2022)	2,63	2,66	2,33	2,54
Fonseca and Toffolo (2022)	4,16	4,03	4,66	4,28
Penelitian Ini	4,80	4,73	4,80	4,77
Berthold et al. (2021)	5,10	4,83	4,40	4,77
Phillips et al. (2021)	5,56	5,43	5,96	5,65
Dimitsas et al. (2022)	5,56	5,53	5,96	5,68
Subba and Stordal (2021)	7,66	7,73	6,96	7,45
Lester (2022)	8,30	8,43	8,26	8,33

Tabel 7.4.3: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark* ITC 2021 Kategori Early

Dataset	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
1	388	362	423	421	804
2	354	222	318	309	402
3	1161	1052	1068	1146	1246
4	1359	536	556	-	764

(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark ITC 2021* Kategori Early

<i>Dataset</i>	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
5	4008	3127	4117	-	-
6	4276	3714	3927	4088	5506
7	5826	4763	5205	6434	6881
8	1612	1114	1051	1064	1409
9	848	108	132	538	122
10	4296	3400	4986	-	-
11	4920	4436	4526	5127	6843
12	1300	510	1010	890	1025
13	659	121	173	331	360
14	837	47	63	84	25
15	5335	3368	3556	4196	4616
Rata-Rata Peringkat	4,80	1,20	2,63	4,16	5,10

Tabel 7.4.4: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark ITC 2021* Kategori Middle

<i>Dataset</i>	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
1	6240	5177	5657	-	-
2	7115	7381	-	-	-
3	8648	9800	9542	9837	9700
4	21	7	16	7	7
5	629	494	510	543	413
6	2130	1275	1701	1630	2270

(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark ITC* 2021 Kategori Middle

<i>Dataset</i>	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
7	2237	2049	2203	2394	2991
8	302	129	136	200	174
9	1195	450	640	1050	810
10	1869	1250	1357	1537	1813
11	3237	2608	2696	2798	3367
12	1354	923	950	1007	1538
13	521	282	362	430	1051
14	2216	1323	1172	1682	1679
15	1099	965	985	1089	1614
Rata-Rata Peringkat	4,73	1,60	2,67	4,03	4,83

Tabel 7.4.5: Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark ITC* 2021 Kategori Late

<i>Dataset</i>	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
1	2804	1969	2021	2068	2073
2	6270	5400	5715	5525	-
3	2564	2369	2457	3069	2474
4	0	0	0	0	0
5	2503	2218	2341	-	-
6	1130	923	930	1212	1082
7	2255	1652	1765	2525	2333
8	1138	934	997	1454	1165

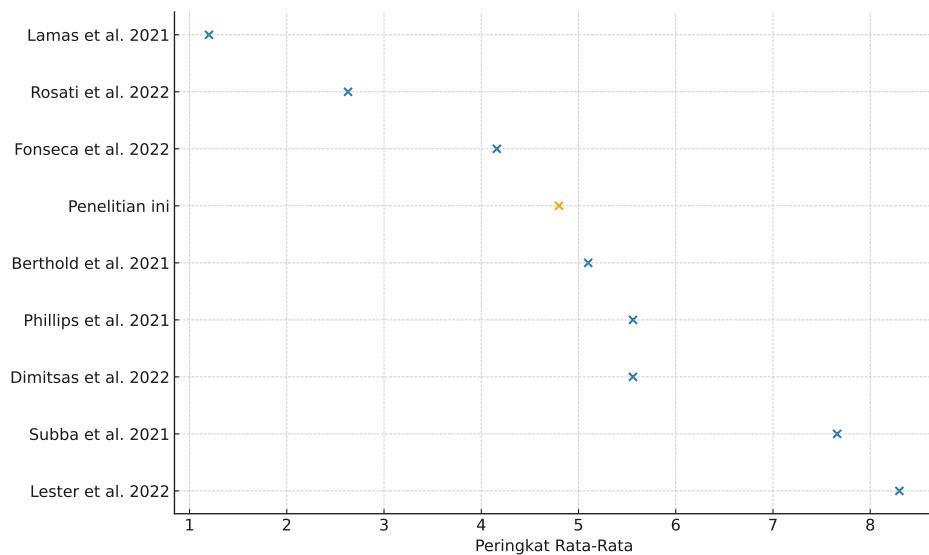
(Lanjutan) Perbandingan Hasil Algoritma pada Setiap Dataset dengan Lima Studi Sebelumnya pada *Benchmark ITC 2021* Kategori Late

<i>Dataset</i>	Penelitian ini	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Berthold et al. (2021)
9	1574	563	715	790	1219
10	2964	2031	2571	2544	-
11	321	226	207	305	361
12	4951	3912	3944	5669	4786
13	1960	2110	1868	3877	1820
14	2160	1363	1202	1433	1562
15	870	40	60	40	160
Rata-Rata Peringkat	4,80	1,63	2,33	4,40	4,67

7.5 Kesimpulan

Bab ini menguji algoritma yang dikembangkan pada permasalahan penjadwalan kompetisi olahraga menggunakan *benchmark ITC 2021*. Pengujian ini bertujuan untuk menguji generalitas algoritma pada permasalahan yang jauh berbeda dengan pengujian pada dua jenis permasalahan sebelumnya, yaitu penjadwalan ujian dan mata kuliah. ITC 2021 menyajikan permasalahan penjadwalan kompetisi olahraga dengan batasan yang jauh berbeda dibandingkan dengan dua permasalahan sebelumnya, serta fleksibilitas slot waktu yang jauh lebih sedikit dengan jumlah pertandingan yang dijadwalkan sama persis dengan slot waktu yang tersedia.

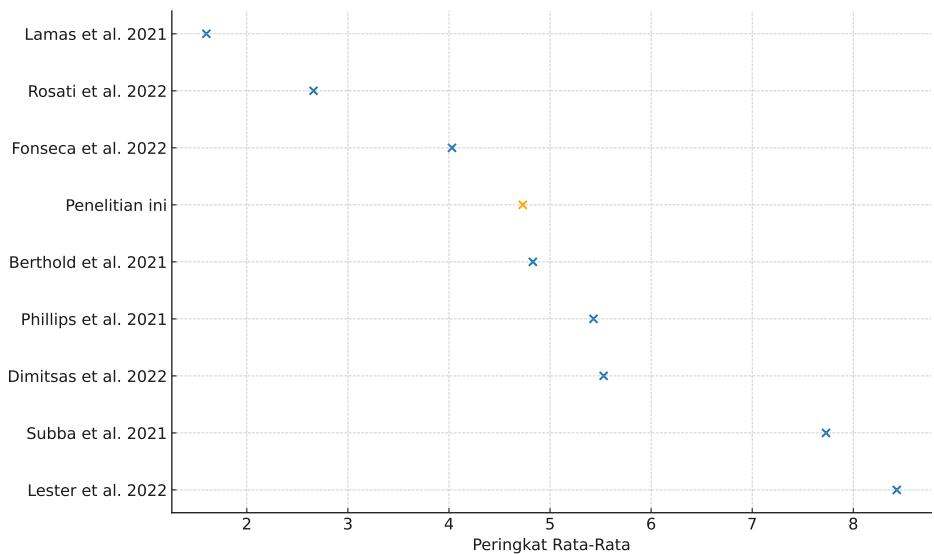
Hasil dari uji coba menunjukkan performa yang baik. Pada tahapan pencarian solusi *feasible*, seluruh dataset dapat ditemukan solusi *feasible* dengan tingkat keberhasilan secara keseluruhan mencapai 94,7%. Hasil ini jauh lebih baik dibandingkan dengan kebanyakan penelitian sebelumnya. Sementara itu, pada perbandingan secara keseluruhan dengan memasukkan faktor optimasi, menempatkan hasil penelitian ini pada peringkat ke-4 dari 9 penelitian. Hal ini menunjukkan algoritma yang dikembangkan mampu



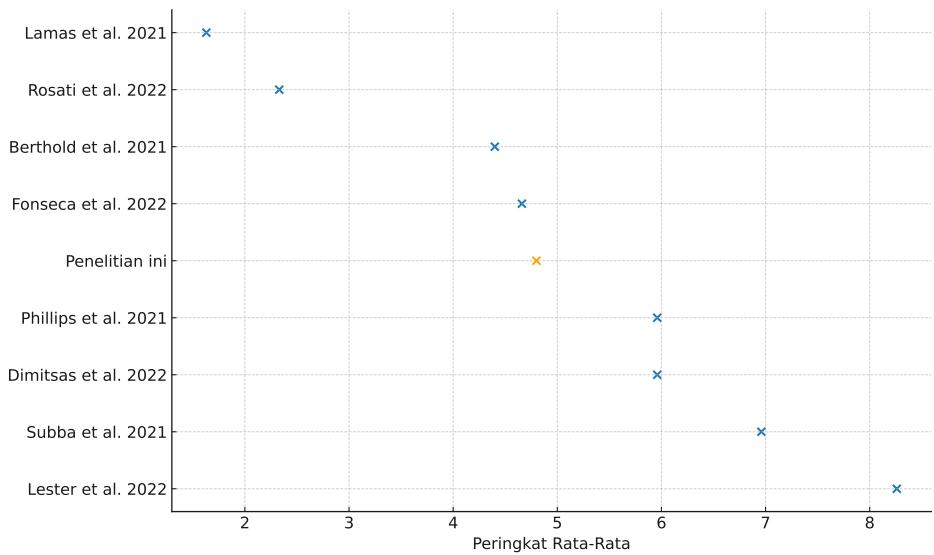
Gambar 7.4.1: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2021 pada Kategori Early

menghasilkan hasil yang relatif kompetitif dengan berada pada peringkat diatas rata-rata.

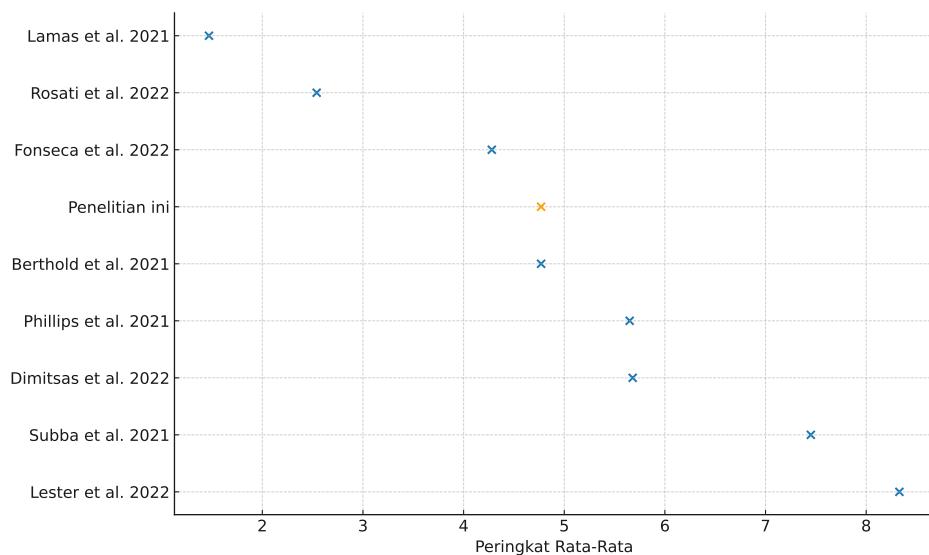
Secara keseluruhan, uji coba ini menunjukkan kemampuan yang baik dari algoritma yang dikembangkan dalam menyelesaikan permasalahan penjadwalan lintas domain. Tiga jenis permasalahan dengan tiga *benchmark* yang memiliki puluhan dataset telah diuji. Hasil keseluruhan menunjukkan bahwa algoritma ini mampu menghasilkan hasil yang konsisten di atas rata-rata dibandingkan dengan penelitian lainnya. Tingkat konsistensi antara dataset juga menunjukkan hasil yang baik dari keseluruhan uji coba, dengan mayoritas dataset yang diuji menghasilkan peringkat yang konsisten. Untuk memvalidasi lebih lanjut, proses pengujian dilanjutkan pada dua studi kasus nyata dengan permasalahan penjadwalan sesi praktikum dan ujian praktikum yang dibahas pada Bab 8.



Gambar 7.4.2: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark ITC 2021* pada Kategori Middle



Gambar 7.4.3: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark ITC 2021* pada Kategori Late



Gambar 7.4.4: Perbandingan Peringkat Rata-Rata pada Penelitian dengan *Benchmark* ITC 2021 pada Keseluruhan Kategori

BAB 8

PERMASALAHAN PENJADWALAN SESI PRAKTIKUM DAN UJIAN PRAKTIKUM (STUDI KASUS DEPARTEMEN SISTEM INFORMASI ITS)

Pada Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS), terdapat permasalahan penjadwalan yang sering dijumpai dalam kegiatan pembelajaran. Salah satu permasalahan yang ada adalah penjadwalan sesi praktikum dan ujian praktikum. Hal ini terjadi pada mata kuliah yang membutuhkan sesi praktikum seperti mata kuliah pemrograman. Saat ini, proses penjadwalan untuk sesi praktikum dan ujian praktikum dijadwalkan secara manual. Hal ini menghadirkan permasalahan secara efektifitas dan efisiensi. Permasalahan pertama adalah waktu dan sumber daya. Penjadwalan secara manual membutuhkan waktu yang lama. Selain itu penjadwalan manual membutuhkan sumber daya berupa orang yang mampu melakukan proses penjadwalan. Permasalahan kedua berkaitan dengan solusi yang dihasilkan. Karena proses penjadwalan manual yang membutuhkan waktu yang lama, proses penjadwalan hanya bisa berfokus untuk dapat menghasilkan solusi yang bisa digunakan (*feasible*). Namun tidak mempertimbangkan apakah jadwal yang dibuat sesuai dengan preferensi asisten pengajar dan mahasiswa dan juga distribusi beban asisten pengajar. Berdasarkan kondisi tersebut, maka diperlukan solusi untuk dapat menjadwalkan kedua permasalahan ini secara otomatis.

Bab ini membahas penerapan algoritma Progressive Acceptance Iterated Local Search (PA-ILS) dan Adaptive Threshold-Iterated Local Search (AT-ILS) untuk menyelesaikan permasalahan penjadwalan sesi praktikum dan ujian praktikum. Pembahasan dimulai pada Subbab 8.1 dengan menjabarkan karakteristik dataset dan format dataset. Selanjutnya, Subbab 8.2 membahas model matematika yang dibentuk dari permasalahan ini. Subbab

8.3 memaparkan hasil dari implementasi algoritma terhadap dua jenis permasalahan ini. Selanjutnya, hasil implementasi dibandingkan dengan algoritma Late Acceptance Hill Climbing (LAHC) dan Hill Climbing (HC) pada Subbab 8.4. Terakhir, Subbab 8.5 menyajikan kesimpulan dari hasil implementasi ini.

8.1 Karakteristik Dataset

8.1.1 Karakteristik Dataset Secara Umum

Studi kasus ini mencakup dua dataset dengan dataset pertama berasal dari semester kedua tahun 2023, sedangkan dataset kedua berasal dari semester pertama tahun 2024. Tabel 8.1.1 menjabarkan deskripsi karakteristik dari kedua dataset ini yang terdiri dari jumlah mahasiswa, asisten pengajar, sesi praktikum/ujian, slot waktu dan ruangan yang tersedia. Selain itu, terdapat data mengenai kapasitas rata-rata ruangan dan persentase rata-rata dari ketersediaan slot waktu dari mahasiswa dan asisten pengajar. Pada data kapasitas ruangan, data hanya ditampilkan pada permasalahan penjadwalan sesi praktikum. Hal ini disebabkan pada permasalahan penjadwalan ujian praktikum, kapasitas ruangan ditentukan oleh jumlah asisten pengajar yang dijadwalkan dalam setiap sesi ujian. Hal ini akan dijelaskan lebih lanjut pada Subbab 8.2.

Perbandingan karakteristik dari kedua dataset menunjukkan tantangan yang berbeda. Untuk penjadwalan sesi praktikum, Dataset 1 memiliki jumlah mahasiswa yang lebih sedikit, jumlah slot waktu yang lebih banyak, dan jumlah ruangan yang lebih banyak dibandingkan Dataset 2. Namun, Dataset 1 memiliki persentase ketersediaan slot waktu yang lebih rendah untuk mahasiswa dan asisten pengajar. Hal ini menunjukkan lebih terbatasnya pergerakan solusi dalam mencari solusi *feasible* dan dalam proses optimasi. Sebaliknya, Dataset 2 memiliki jumlah mahasiswa dan asisten pengajar yang lebih tinggi namun jumlah slot waktu dan ruangan yang lebih sedikit. Tetapi, Dataset 2 memiliki persentase ketersediaan slot waktu yang lebih tinggi untuk mahasiswa dan asisten pengajar. Hal ini memberikan fleksibilitas yang lebih tinggi dalam proses menghasilkan solusi *feasible* dan proses optimasi.

Untuk permasalahan penjadwalan ujian praktikum, pada dasarnya memiliki karakteristik yang serupa karena ujian praktikum ini dilaksanakan terhadap mahasiswa yang mengikuti sesi praktikum. Namun pada permasalahan penjadwalan ujian, persentase

ketersediaan slot waktu lebih rendah dibandingkan dengan pada permasalahan penjadwalan. Hal ini akan menjadi tantangan, karena fleksibilitas dalam pencarian solusi *feasible* dan optimasi akan berkurang. Selain itu, kapasitas ruangan yang dinamis yang ditentukan oleh jumlah asisten pengajar, menjadi tantangan yang berbeda dengan permasalahan penjadwalan sesi praktikum.

Tabel 8.1.1: Deskripsi Dataset untuk Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum

Karakteristik	Dataset 1		Dataset 2	
	Praktikum	Ujian	Praktikum	Ujian
Jumlah Mahasiswa	217	217	233	233
Jumlah Asisten Pengajar	12	12	13	13
Jumlah Sesi Praktikum / Sesi Ujian	6	6	7	7
Jumlah Slot Waktu	54	54	33	33
Jumlah Ruangan	6	6	4	4
Kapasitas Rata-Rata Ruangan	45	-	52,5	-
Persentase Ketersediaan Slot Waktu Mahasiswa	34,17%	12,65%	47,96%	29,06%
Persentase Ketersediaan Slot Waktu Asisten Pengajar	49,16%	12,96%	47,69%	26,8%

8.1.2 Struktur Dataset

Pada permasalahan ini, dataset dirumuskan dalam format csv. Setiap dataset memiliki tiga file yang terdiri dari data ruangan, mahasiswa dan asisten pengajar. Gambar 8.1.1 menyajikan penggambaran format dari dataset ini. Pada contoh ini disajikan format dari data ruangan. Kolom dua sampai empat menyajikan data id ruangan, nama ruangan dan juga kapasitasnya. Sementara 20 kolom berikutnya menyajikan ketersediaan ruangan pada setiap slot waktunya. Jumlah 20 slot waktu didapatkan dari setiap hari yang memiliki 4 slot dan waktu pembelajaran hanya bisa dilakukan pada hari senin hingga jumat (5 hari), sehingga secara keseluruhan terdapat 20 slot waktu. Pada setiap slot waktu, jika terdapat angka satu artinya ruangan tersebut dapat digunakan pada slot waktu tersebut. Sebaliknya jika terdapat angka 0 artinya ruangan tersebut tidak dapat digunakan pada slot waktu tersebut.

Sementara itu pada mahasiswa dan asisten pengajar memiliki format yang sama, beberapa kolom awal menyajikan deskripsi dari setiap mahasiswa dan asisten pengajar seperti nama dan id. Sementara 20 kolom berikutnya menyajikan data ketersediaan slot waktu mereka dengan format yang sama seperti pada ketersediaan slot waktu dalam data ruangan. Gambar 8.1.2 dan 8.1.3 menunjukkan contoh dari format data mahasiswa dan asisten pengajar.

No	Room ID	Room Name	Room Capacity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1101	Class Room	35	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0	1	0	1	1	1
2	1102	Class Room	35	1	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1
3	STU 1 (APTER)	Applied Program Lab	50	1	0	0	1	0	0	0	0	1	0	1	1	1	1	0	1	0	0	1	1
4	STU 2 (PRO)	Programming Lab	50	1	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1
5	STU 1 (APTER)	Jason Ho	50	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1
6	STU 2 (PRO)	Programming Lab	50	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1

Gambar 8.1.1: Struktur Dataset Ruangan Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum

No	Student ID	Full Name	Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	5026221001	Viqi Alvianto	A	0	0	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	1	1
2	5026221002	Shabrina Nur Ihsani	A	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1
3	5026221003	Belva Talitha Dwiyanti	A	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	1	0	0	1	1
4	5026221004	Celine Auriel	A	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1
5	5026221005	Jason Ho	A	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1
6	5026221028	Muhammad Robikhul Zaki Zain	A	1	0	0	1	0	1	0	0	0	0	0	1	1	1	1	0	1	0	1	1
7	5026221029	Diva Ardelia Alyadrus	A	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1
8	5026221030	Fajhri Ramadhan A.	A	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0	1	1	0	1	1
9	5026221031	Talitha Firyal Ghina Nuha	A	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1
10	5026221032	Fadillah Nur Laili	A	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1	0	0	1	1	1

Gambar 8.1.2: Struktur Dataset Mahasiswa Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum

No	Student ID	Full Name	codeboard	Init	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	521940000011	Muhammad Fajrul Alam Ulin Nuha	fajrulalam	FA	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
2	5026201011	Muhammad Arif Nuriman	muhammadarif011	AN	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1
3	5026201117	Naufal Rafiawan Basara	naufalbasara	RB	0	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	1	1	1	1
4	5026201124	Aulia Cisatra	cisatra	AC	0	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1	1	0	1	1
5	5026201134	Liefrian Satrio Sim	lsatriosim	LS	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1	1	1
6	5026201138	Muhammad Dimas Adjianto	donmai	DA	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	1	1	1
7	5026211012	Inggrit Rismauli Siahaan	inggritsiahaan	IR	1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0	0	0	1	1
8	5026211029	Ahmadian Daffa Yudistira	ahmadhiandaffa	DY	0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0	1	0	1	0
9	5026211082	Ribka Devina Margaretha	ribkadevinamargaretha	RD	1	0	0	0	1	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1
10	5026211116	Tasya Putri Aliya	tasyaputrialiya	TP	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1
11	5026211118	Fahrul Ramadhan Putra	5026211118fahrulputra	FR	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1
12	5026211187	Andaru Pratama	andarupratama	AP	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1

Gambar 8.1.3: Struktur Dataset Asisten Pengajar Pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum

8.2 Model Matematika

8.2.1 Variabel Keputusan

Pada permasalahan penjadwalan sesi praktikum dan ujian praktikum, terdapat empat variabel keputusan. Secara spesifik, keempat variabel keputusan tersebut dapat definisikan sebagai berikut:

- $w_{ik} \in \{0, 1\}$: Bernilai 1 jika mahasiswa i mengikuti sesi praktikum k , dan 0 jika tidak. Variabel ini hanya digunakan pada permasalahan penjadwalan praktikum.
- $x_{ijk} \in \{0, 1\}$: Bernilai 1 jika mahasiswa i diawasi oleh asisten j dalam sesi ujian k , dan 0 jika tidak.. Variabel ini hanya digunakan pada permasalahan penjadwalan ujian praktikum.
- $y_{ktr} \in \{0, 1\}$: Bernilai 1 jika sesi k dijadwalkan di ruangan r pada slot waktu t , dan 0 jika tidak. Variabel ini digunakan pada permasalahan penjadwalan sesi praktikum dan ujian praktikum.
- $z_{jk} \in \{0, 1\}$: Bernilai 1 jika asisten pengajar j ditugaskan pada sesi k , dan 0 jika tidak. Variabel ini digunakan pada permasalahan penjadwalan sesi praktikum dan ujian praktikum.

8.2.2 Fungsi Objektif

Fungsi objektif dalam model ini dirancang untuk meminimalkan total penalti yang dihasilkan dari pelanggaran dua jenis *soft constraint* yang telah dijabarkan. Dalam proses optimasi, terdapat kemungkinan untuk menerima solusi yang tidak *feasible* namun dibatasi hanya pada kondisi belum terjadwalkannya sejumlah mahasiswa atau asisten pengajar. Untuk memastikan solusi yang tidak *feasible* cenderung tidak dipilih untuk diterima, maka pada fungsi tujuan, nilai penalti ditambahkan dengan jumlah mahasiswa dan asisten pengajar yang belum terjadwalkan dan dikalikan dengan bobot sebesar 100. Secara keseluruhan fungsi tujuan diformulasikan pada persamaan 8.1.

$$\text{Minimize} \quad \text{Penalti}_{\text{distribusi}} + \text{Penalti}_{\text{waktu}} + 100 \times \left(\sum_i T_i + \sum_j S_j \right) \quad (8.1)$$

dimana:

- Penalti_{distribusi}: Penalty yang terkait dengan ketidakseimbangan distribusi mahasiswa pada sesi praktikum atau asisten pengajar pada ujian praktikum.
- Penalti_{waktu}: Penalty yang terkait dengan preferensi waktu tertentu, di mana slot waktu tertentu dihindari sesuai masukan dari mahasiswa dan asisten pengajar.
- $\sum_i T_i$: merupakan jumlah asisten pengajar yang belum dijadwalkan.
- $\sum_i S_i$: merupakan jumlah mahasiswa yang belum dijadwalkan.

8.2.3 Hard Constraint

Permasalahan penjadwalan sesi praktikum dan ujian praktikum memiliki masing-masing lima *hard constraint* yang harus dipenuhi. Pada permasalahan penjadwalan sesi praktikum, kelima *hard constraint* dijabarkan pada Subbab 8.2.3.1. Sementara itu, lima *hard constraint* pada permasalahan penjadwalan ujian praktikum dijabarkan pada Subbab 8.2.3.2.

8.2.3.1 Penjadwalan Praktikum

1. **Setiap Mahasiswa Dijadwalkan Tepat Satu Kali** : Batasan ini mengatur untuk memastikan setiap mahasiswa hanya dijadwalkan tepat satu kali pada salah satu sesi yang tersedia. Secara matematis batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_k w_{ik} = 1, \quad \forall i. \tag{8.2}$$

di mana $\sum_k w_{ik}$ merupakan jumlah dari mahasiswa i yang dijadwalkan pada setiap sesi. Jumlah tersebut mengharuskan tepat satu kali yang menandakan mahasiswa i tepat dijadwalkan satu kali.

2. **Kapasitas Ruang** : Batasan ini berfungsi untuk memastikan bahwa jumlah mahasiswa yang dijadwalkan pada suatu sesi tidak melebihi kapasitas total ruang yang digunakan untuk sesi tersebut. Secara matematis, batasan ini dapat dirumuskan

sebagai berikut:

$$\sum_i w_{ik} \leq \sum_{t,r} (\text{Cap}(r)) y_{ktr}, \quad \forall k. \quad (8.3)$$

di mana $\sum_i w_{ik}$ menyatakan total mahasiswa yang terjadwal pada sesi k , $\text{Cap}(r)$ adalah kapasitas ruangan r , dan y_{ktr} merupakan variabel biner yang bernilai 1 jika ruangan r digunakan pada waktu t di sesi k , serta 0 jika tidak digunakan.

3. **Ketersediaan Slot Waktu :** Batasan ini bertujuan untuk memastikan bahwa penjadwalan hanya dilakukan jika mahasiswa maupun asisten pengajar memiliki ketersediaan waktu pada slot yang akan dijadwalkan. Secara matematis dapat dinyatakan dengan dua persamaan berikut:

$$w_{ik} \leq \sum_{t,r} (a_{i,t}) y_{ktr}, \quad \forall i, k. \quad (8.4)$$

$$z_{jk} \leq \sum_{t,r} (b_{j,t}) y_{ktr}, \quad \forall j, k. \quad (8.5)$$

di mana:

- w_{ik} menyatakan penjadwalan mahasiswa i pada sesi k .
- z_{jk} menyatakan penjadwalan asisten pengajar j pada sesi k .
- $a_{i,t}$ bernilai 1 jika mahasiswa i tersedia pada slot waktu t , dan 0 jika tidak.
- $b_{j,t}$ bernilai 1 jika asisten pengajar j tersedia pada slot waktu t , dan 0 jika tidak.
- y_{ktr} adalah variabel biner yang bernilai 1 jika ruangan r digunakan pada waktu t di sesi k , dan 0 jika tidak.

4. **Penjadwalan Sesi (Tepat Satu Ruang-Waktu per Sesi Praktikum) :** Batasan ini mengatur agar setiap sesi praktikum k hanya ditempatkan pada satu kombinasi ruang-waktu (ruang r dan slot waktu t) yang tersedia. Secara matematis, batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_{t,r} y_{ktr} = 1, \quad \forall k. \quad (8.6)$$

Di mana $\sum_{t,r} y_{ktr}$ merupakan jumlah penempatan sesi praktikum k pada seluruh kombinasi ruang-waktu, dan nilai totalnya harus tepat satu.

5. **Batasan Penugasan Asisten (Minimal 2 Per Sesi)** : Batasan ini mengatur agar setiap sesi praktikum k memiliki paling sedikit dua asisten pengajar yang ditugaskan. Secara matematis, batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_j z_{jk} \geq 2, \quad \forall k. \quad (8.7)$$

Di mana $\sum_j z_{jk}$ merupakan jumlah asisten pengajar j yang ditugaskan pada sesi k .

8.2.3.2 Penjadwalan Ujian

1. **Setiap mahasiswa ditempatkan pada tepat satu asisten pengajar dan satu sesi**: Batasan ini memastikan bahwa setiap mahasiswa i hanya ditempatkan pada tepat satu kombinasi asisten pengajar j dan sesi k . Secara matematis, batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_{j,k} x_{ijk} = 1, \quad \forall i. \quad (8.8)$$

Di mana $\sum_{j,k} x_{ijk}$ merepresentasikan jumlah penugasan mahasiswa i kepada seluruh asisten pengajar j dan sesi k , yang harus bernilai tepat satu.

2. **Batas kapasitas asisten pengajar**: Batasan ini memastikan bahwa setiap asisten pengajar j pada sesi k tidak mengampu lebih dari 10 mahasiswa. Secara matematis, batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_i x_{ijk} \leq 10, \quad \forall j,k. \quad (8.9)$$

Di mana $\sum_i x_{ijk}$ menunjukkan total jumlah mahasiswa i yang ditugaskan kepada asisten pengajar j pada sesi k , dan nilai totalnya harus tidak melebihi 10.

3. Penjadwalan Sesi (Tepat Satu Kombinasi Ruang-Waktu)

Batasan ini mengatur agar setiap sesi k harus dijadwalkan tepat satu kali, yaitu pada salah satu kombinasi ruang (r) dan waktu (t). Secara matematis, batasan tersebut dapat dirumuskan sebagai berikut:

$$\sum_{t,r} y_{ktr} = 1, \quad \forall k. \quad (8.10)$$

Di mana $\sum_{t,r} y_{ktr}$ menyatakan penjadwalan sesi k pada berbagai ruang (r) dan slot waktu (t). Batasan ini memastikan bahwa setiap sesi k harus dijadwalkan tepat satu kali.

4. Ketersediaan Mahasiswa dan Asisten Pengajar

Batasan ini mengatur bahwa penjadwalan mahasiswa i yang diawasi oleh asisten pengajar j pada sesi k hanya memungkinkan jika mahasiswa i serta asisten pengajar j tersedia pada slot waktu sesi k dilaksanakan. Secara matematis, hal tersebut dinyatakan dengan:

$$x_{ijk} \leq \sum_{t,r} (a_{i,t}) y_{ktr}, \quad \forall i, j, k, \quad (8.11)$$

$$x_{ijk} \leq \sum_{t,r} (b_{j,t}) y_{ktr}, \quad \forall i, j, k. \quad (8.12)$$

Di mana:

- $a_{i,t} = 1$ jika mahasiswa i tersedia pada slot waktu t , dan 0 jika tidak.
- $b_{j,t} = 1$ jika asisten pengajar j tersedia pada slot waktu t , dan 0 jika tidak.
- $y_{ktr} = 1$ jika sesi k dijadwalkan di slot waktu t dan ruang r , dan 0 jika tidak.

5. Batas Jumlah Asisten per Sesi

Batasan ini mengatur jumlah maksimal asisten pengajar yang boleh mengawasi ujian untuk setiap sesi k . Secara matematis, batasan tersebut dapat dinyatakan sebagai:

$$\sum_j z_{jk} \leq 4, \quad \forall k. \quad (8.13)$$

Di mana z_{jk} merupakan variabel yang bernilai 1 jika asisten j mengawasi ujian di sesi k , dan 0 jika tidak. Batasan ini memastikan bahwa jumlah asisten dalam setiap sesi k tidak melebihi 4.

8.2.4 Soft Constraint

Pada permasalahan penjadwalan sesi praktikum dan ujian praktikum, terdapat dua *soft constraint* yang mengatur keadilan beban kerja bagi asisten pengajar dan mengatur preferensi waktu yang lebih disukai oleh mahasiswa dan asisten pengajar. Kedua batasan tersebut dijabarkan pada Subbab 8.2.4.1 dan 8.2.4.2.

8.2.4.1 Distribusi Beban Asisten Pengajar

Dalam penjadwalan ujian praktikum dan sesi praktikum, penting untuk memastikan distribusi beban kerja yang adil bagi asisten pengajar. Selain itu, pemberian beban pengajaran yang adil terutama pada sesi praktikum akan memberikan dampak sesi praktikum yang baik bagi mahasiswa.

Pada permasalahan ujian, batasan dikembangkan untuk memastikan bahwa jumlah mahasiswa yang diawasi oleh setiap asisten pengajar mendekati jumlah ideal, untuk membagi beban kerja secara merata. Untuk mewujudkan hal tersebut maka batasan diformulasikan dalam persamaan 8.14 dan 8.15.

$$\text{Penalty}_{\text{distribusi}} = \sum_j 2p_j, \quad (8.14)$$

dengan kendala linear:

$$p_j \geq S_j - I, \quad p_j \geq 0, \quad \forall j. \quad (8.15)$$

Di mana:

- S_j menyatakan total mahasiswa yang diawasi oleh asisten j .
- I adalah jumlah ideal mahasiswa per asisten pengajar, dihitung sebagai:

$$I = \left\lfloor \frac{N_{\text{total}}}{M_{\text{total}}} \right\rfloor \quad (8.16)$$

dengan N_{total} adalah total jumlah mahasiswa dan M_{total} adalah total jumlah asisten pengajar.

Variabel p_j merepresentasikan kelebihan beban pengawasan asisten j dibandingkan jumlah ideal I . Jika $S_j \leq I$, maka p_j akan bernilai 0 karena kendala linear $p_j \geq 0$ dan $p_j \geq S_j - I$. Namun, jika jumlah mahasiswa yang diawasi melebihi I , maka p_j akan sama dengan $S_j - I$, sehingga selisih tersebut terhitung sebagai penalti. Nilai penalti total kemudian dijumlahkan dengan faktor pengali 2. Oleh karena itu, penalti hanya muncul ketika terjadi kelebihan beban pengawasan dan bernilai 0 jika seorang asisten mengawasi jumlah mahasiswa yang sama atau di bawah nilai ideal I .

Sementara pada permasalahan penjadwalan sesi praktikum, batasan yang serupa juga diterapkan. Namun karena dalam setiap sesi praktikum memungkinkan ada jumlah asisten pengajar yang berbeda sehingga batasan diformulasikan seperti pada persamaan 8.17 dan 8.18.

$$\text{Penalti}_{\text{distribusi}} = \sum_k 2 p_k, \quad (8.17)$$

dengan kendala linear:

$$p_k \geq S_k - I \times (TA_k), \quad p_k \geq 0, \quad \forall k. \quad (8.18)$$

Di mana:

- S_k adalah jumlah mahasiswa di sesi praktikum k .
- TA_k adalah jumlah asisten pengajar yang ditugaskan pada sesi tersebut.
- I adalah jumlah ideal mahasiswa per asisten pengajar, dihitung sebagai dengan rumus pada persamaan 8.16.

Jika jumlah mahasiswa S_k dalam sesi k melebihi kapasitas ideal ($I \times TA_j$), maka penalti diterapkan sebesar selisih tersebut dikalikan faktor penalti. Hal ini bertujuan untuk mendorong distribusi mahasiswa yang lebih merata di antara sesi praktikum, sehingga setiap asisten pengajar dapat memberikan bimbingan yang optimal.

8.2.4.2 Preferensi Waktu

Berdasarkan masukan dari mahasiswa dan asisten pengajar, terdapat slot waktu yang se bisa mungkin dihindari. Slot waktu pertama yang dimulai di pagi hari tidak disukai karena banyak mahasiswa dan asisten pengajar yang terlambat ketika sesi dilaksanakan pada slot waktu tersebut. Sementara itu, slot waktu keempat yang dimulai di sore hari juga kurang disukai karena kecenderungan merasa lelah atau mengantuk saat sesi berlangsung.

Untuk mengakomodasi preferensi ini, model penjadwalan memberikan penalti pada sesi yang dijadwalkan pada slot waktu yang kurang disukai. Formulasi kendala preferensi waktu dijabarkan dalam persamaan 8.19.

$$\text{Penalti}_{\text{waktu}} = \sum_{l,s} P_l \cdot (T_t + S_s) \cdot z_{ls} \quad (8.19)$$

Di mana:

- z_{ls} adalah variabel biner yang bernilai 1 jika sesi s dijadwalkan pada slot waktu l , dan 0 jika tidak.
- P_l adalah penalti yang terkait dengan penjadwalan sesi pada slot waktu l .
- S_s adalah jumlah mahasiswa yang terlibat dalam sesi s .

- T_s adalah jumlah asisten pengajar yang terlibat dalam sesi s .

Penalty P_l ditetapkan berdasarkan preferensi terhadap slot waktu tertentu yang telah dikumpulkan melalui diskusi dengan mahasiswa, asisten pengajar dan dosen. Slot waktu yang lebih disukai tidak memiliki penalti ($P_2 = 0$, $P_3 = 0$), sedangkan slot waktu yang kurang disukai memiliki penalti lebih tinggi ($P_1 = 1$, $P_4 = 2$). Dengan mengalikan penalti dengan jumlah mahasiswa dan asisten pengajar dalam sesi tersebut (S_s), model mempertimbangkan dampak preferensi waktu terhadap jumlah mahasiswa yang terlibat.

Sebagai contoh, jika sesi ke lima memiliki $S_s = 30$ mahasiswa dan $T_s = 2$ asisten pengajar dan dijadwalkan pada slot waktu $l = 4$, yang memiliki penalti $P_4 = 2$. Maka kontribusi penalti untuk sesi tersebut adalah:

$$\text{Penalty}_{\text{waktu}} = P_4 \cdot (T_s + S_s) \cdot z_{4,5} = 2 \times (2 + 30) \times 1 = 64$$

8.3 Hasil Implementasi dan Analisis Algoritma

Pada bagian ini, algoritma yang dikembangkan dalam penelitian ini diimplementasikan pada permasalahan penjadwalan sesi praktikum dan ujian praktikum. Proses implementasi dilakukan melalui dua tahapan yaitu tahapan pertama untuk mencari solusi *feasible* dan tahapan kedua untuk mengoptimasi solusi. Kedua tahapan dijabarkan secara detail pada Subbab 8.3.1 dan 8.3.2

Pada kedua tahapan, Low-Level Heuristic (LLH) *move* dan *swap* digunakan dalam proses pencarian. Sementara LLH *kempe chain* tidak digunakan karena keterbatasan ketersediaan slot waktu antara mahasiswa yang mengakibatkan tidak adanya rantai konflik antara mahasiswa yang menyebabkan LLH *kempe chain* akan bekerja menyerupai LLH *swap*.

8.3.1 Hasil dan Analisis Tahapan Pencarian Solusi *Feasible*

Pada tahapan ini, proses pencarian solusi *feasible* diterapkan pada permasalahan penjadwalan sesi praktikum dan ujian praktikum. Hasil dari penerapan disajikan dalam Tabel 8.3.1. Hasil ini menunjukkan kemampuan yang baik dari algoritma PA-ILS dengan mampu menemukan solusi *feasible* dengan persentase keberhasilan sebesar 100%.

Sementara itu dari sisi durasi untuk menemukan solusi *feasible*, pada tiga dataset yaitu dataset 1 dan 2 pada permasalahan penjadwalan sesi praktikum dan dataset 2 pada permasalahan ujian praktikum, algoritma PA-ILS mampu menemukan solusi *feasible* dengan rata-rata durasi dibawah 3 detik. Sementara pada dataset 1 dalam permasalahan penjadwalan ujian membutuhkan waktu yang lebih lama dengan rata-rata 101,63 detik atau berkisar 1,6 menit.

Hasil ini menunjukkan waktu yang relatif cepat untuk menemukan solusi *feasible*. Namun hal ini tidak mengindikasikan kedua permasalahan ini merupakan permasalahan yang mudah, melainkan disebabkan oleh ukuran permasalahan. Jika dibandingkan dengan *benchmark* Toronto, salah satu dataset terkecil pada *benchmark* tersebut menjadwalkan 80 ujian dan 1108 mahasiswa. Sementara pada permasalahan ini, dataset terbesar hanya menjadwalkan 7 sesi praktikum/ujian dengan 233 mahasiswa.

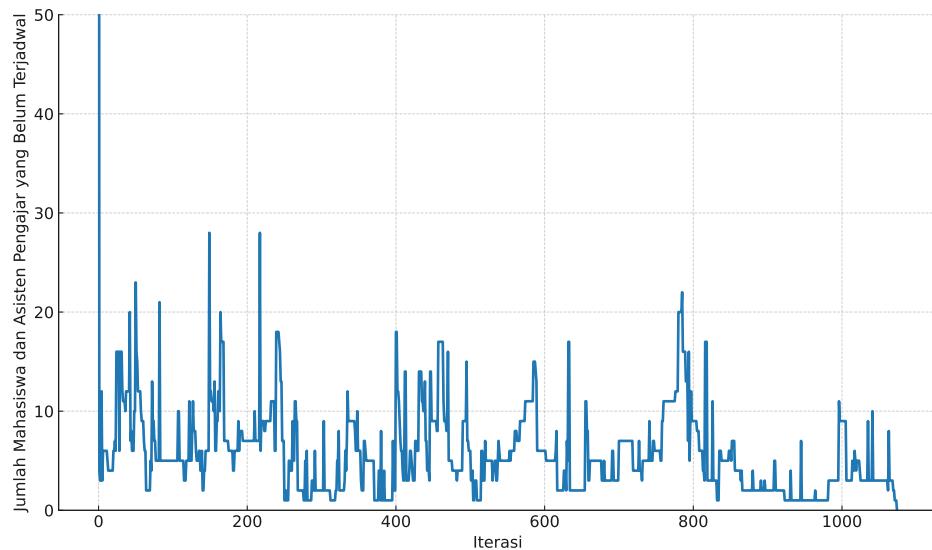
Untuk melihat proses pencarian solusi *feasible*, Gambar 8.3.1 dan 8.3.2 menvisualisasikan perubahan solusi pada setiap iterasinya. Dari hasil visualisasi, kedua permasalahan ini bukan merupakan permasalahan yang mudah terutama pada permasalahan penjadwalan ujian praktikum. Hal ini dapat dilihat pada jumlah iterasi yang dibutuhkan untuk menemukan solusi *feasible*. Jika dibandingkan dengan tiga *benchmark* sebelumnya, pergerakan solusi pada permasalahan ini lebih masif dan membutuhkan lebih dari 1000 iterasi untuk dapat menemukan solusi *feasible*. Sementara tiga *benchmark* sebelumnya membutuhkan 100-350 iterasi. Hal ini menunjukan bahwa kedua permasalahan ini bukan merupakan permasalahan yang mudah dalam menghasilkan solusi *feasible*.

Dari sisi algoritma, hasil ini menunjukan bahwa algoritma PA-ILS mampu bekerja dengan baik. Kedua ilustrasi menunjukan bahwa algoritma memungkinkan melakukan eksplorasi namun tetap mampu mempertahankan konvergensi. Hasil visualisasi menunjukan pergerakan solusi yang tidak terlalu jauh dari tujuan (seluruh mahasiswa dan asisten pengajar terjadwal). Ketika ada pergerakan yang sedikit menjauh dari tujuan seperti mendekati angka 30 dalam iterasi antara 150-200 pada Gambar 8.3.1, algoritma akan menekan pencarian untuk kembali ke arah eksplorasi. Algoritma ini juga menunjukan proses pencarian berjalan efektif, tidak terjebak dalam kondisi *local optima* dan tidak terlalu eksploratif dan mampu menghasilkan 100% tingkat keberhasilan pada

kedua permasalahan ini.

Tabel 8.3.1: Hasil Uji Coba Tahapan Pencarian Solusi *Feasible* pada Permasalahan Penjadwalan Sesi Praktikum dan Ujian Praktikum

Permasalahan	Dataset	Waktu (detik)			Percentase Solusi <i>Feasible</i>
		Rata-Rata	Minimum	Maksimum	
Ujian	Dataset 1	101,63	21,07	295,98	100%
	Dataset 2	1,29	0,06	5,58	100%
Praktikum	Dataset 1	2,23	0,08	7,20	100%
	Dataset 2	0,47	0,15	1,14	100%

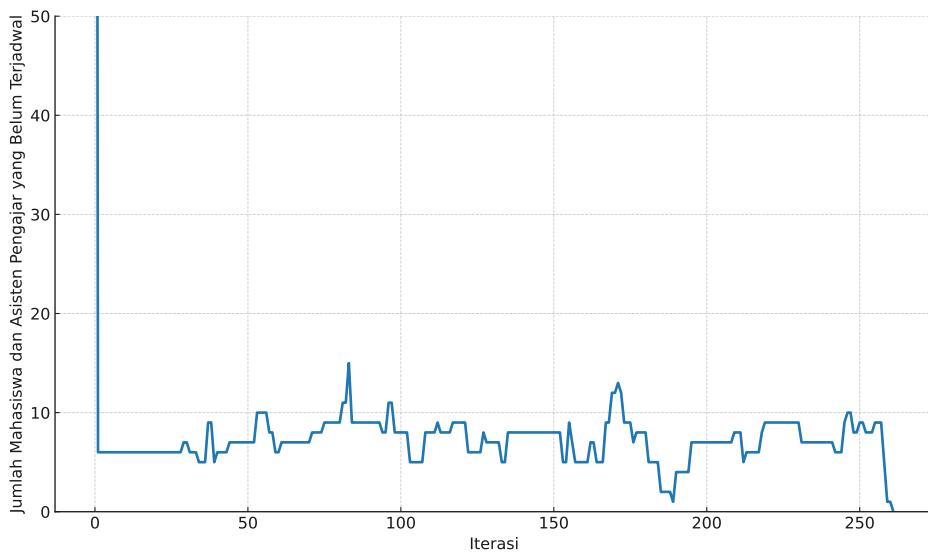


Gambar 8.3.1: Grafik Perubahan Solusi dalam Proses Pencarian Solusi *Feasible* pada Permasalahan Penjadwalan Ujian Praktikum

8.3.2 Hasil dan Analisis Tahapan Optimasi

Pada bagian ini, hasil solusi *feasible* dilanjutkan pada tahapan optimasi. Pada studi kasus ini, proses optimasi dijalankan dengan mengoptimasi sebanyak 10 kali untuk setiap dataset dengan durasi optimasi selama 10 menit. Penggunaan solusi *feasible* sebagai input dilakukan dengan pemilihan acak dari 10 solusi yang telah dihasilkan pada tahapan sebelumnya.

Dalam proses optimasi, karena permasalahan ini memiliki keterbatasan slot waktu yang sangat sedikit, maka dalam tahapan *perturbation* dan *local search* algoritma dimungkinkan



Gambar 8.3.2: Grafik Perubahan Solusi dalam Proses Pencarian Solusi *Feasible* pada Permasalahan Penjadwalan Sesi Praktikum

untuk menerima solusi yang tidak *feasible*, namun hanya untuk kondisi mahasiswa dan asisten pengajar yang belum dijadwalkan. Dalam tahapan *move acceptance* hanya solusi *feasible* yang memungkinkan untuk diterima.

Hasil dari tahapan optimasi ditampilkan pada Tabel 8.3.2. Hasil ini menunjukkan konsistensi algoritma yang tidak terlalu baik dengan rentang nilai Koefisien Variasi (KV) antara 22% hingga 55%. Hal ini disebabkan oleh permasalahan yang memiliki slot waktu yang sangat terbatas, sehingga proses optimasi lebih sering menghasilkan solusi yang tidak *feasible* dan tidak menghasilkan perubahan yang signifikan dari kondisi solusi awal. Hal ini menyebabkan hasil yang didapatkan lebih banyak dipengaruhi oleh kondisi solusi awal.

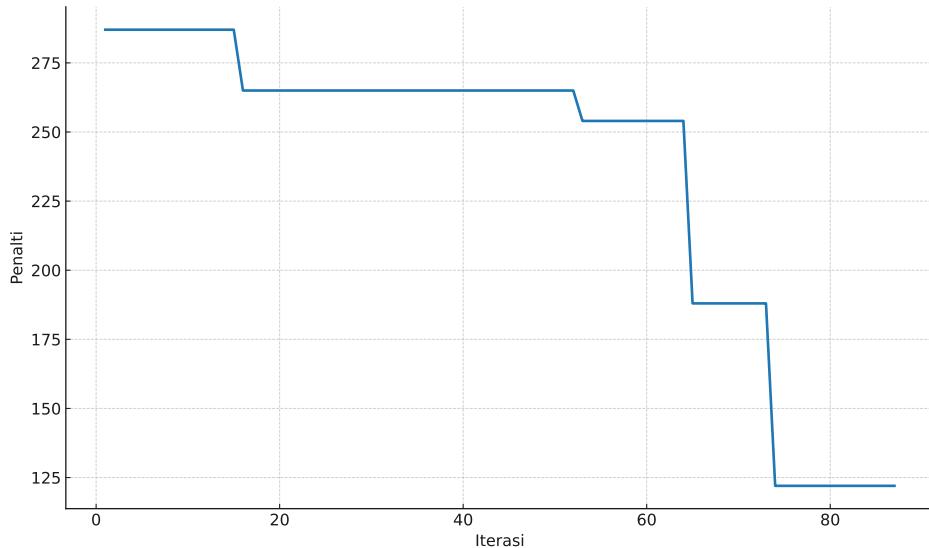
Untuk memahami lebih detail, Gambar 8.3.3 memvisualisasikan proses optimasi pada permasalahan penjadwalan ujian praktikum dan Gambar 8.3.4 memvisualisasikan proses optimasi pada permasalahan penjadwalan sesi praktikum. Hasil visualisasi ini menunjukkan bahwa pada tahapan *move acceptance* jarang menerima solusi baru dengan ditunjukkan banyaknya solusi yang sama dalam beberapa iterasi. Selain itu, eksplorasi yang terjadi antar iterasinya juga minim. Dalam visualisasi ini hanya ada satu eksplorasi yang ditemukan yaitu pada Gambar 8.3.4 pada awal iterasi. Hal ini mengindikasikan permasalahan ini sulit untuk menghasilkan solusi *feasible* dan juga strategi eksplorasi tidak berjalan dengan baik, yang kemungkinan disebabkan oleh banyaknya solusi baru yang

dihadarkan tidak *feasible* dan juga memiliki nilai penalti yang terlalu jauh dari nilai penalti pada iterasi sebelumnya.

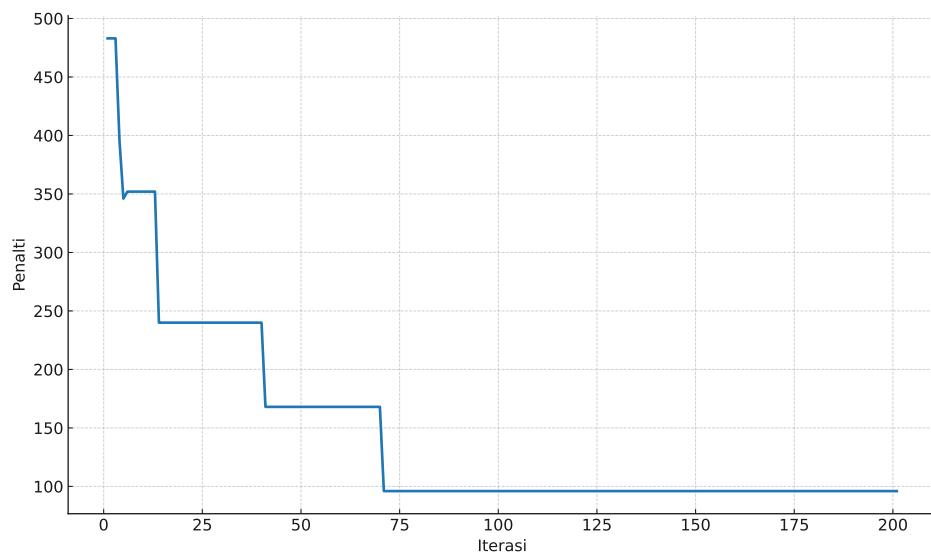
Sementara pada proses yang terjadi dalam tahapan *local search*, Gambar 8.3.5 memvisualisasikan proses tahapan *local search* pada permasalahan penjadwalan ujian praktikum dan Gambar 8.3.6 memvisualisasikan untuk permasalahan penjadwalan sesi praktikum. Hasil visualisasi ini menunjukkan tahapan *local search* telah bekerja sesuai dengan strategi yang dibentuk. Proses pencarian solusi yang bergerak di antara tahapan eksplorasi dan eksploitasi solusi namun tetap diarahkan menuju konvergensi melalui nilai ambang batas.

Tabel 8.3.2: Hasil Uji Coba Tahapan Optimasi pada Permasalahan Penjadwalan Sesi Praktikum dan Penjadwalan Ujian Praktikum

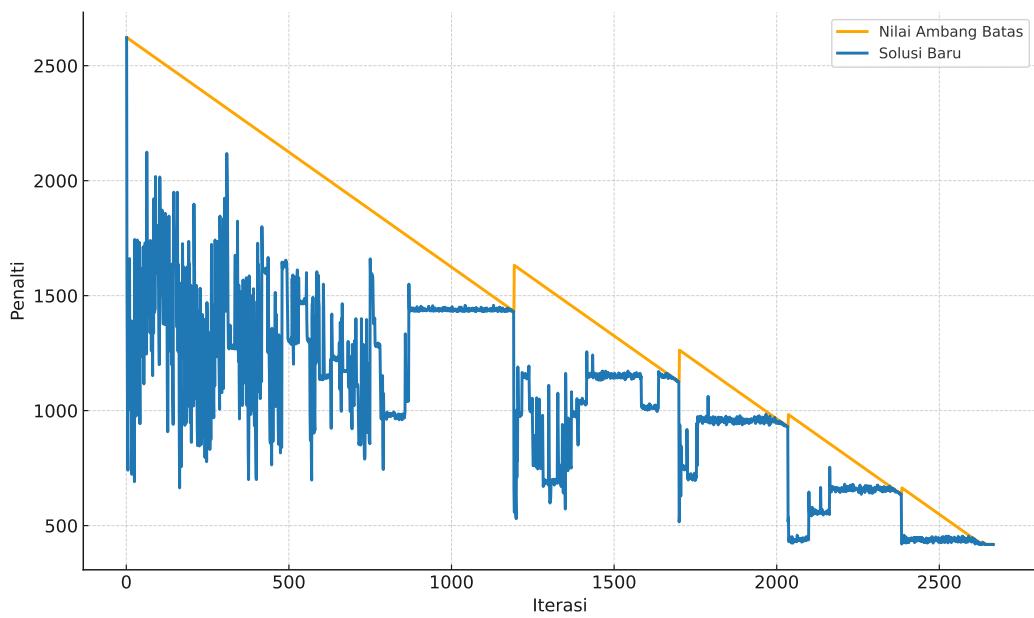
Permasalahan	Dataset	Rata-rata	Minimum	Maksimum	KV (%)
Ujian Praktikum	Dataset 1	160,6	130	218	22
	Dataset 2	121,1	39	188	44
Sesi Praktikum	Dataset 1	111,2	74	194	39
	Dataset 2	63,8	14	130	55



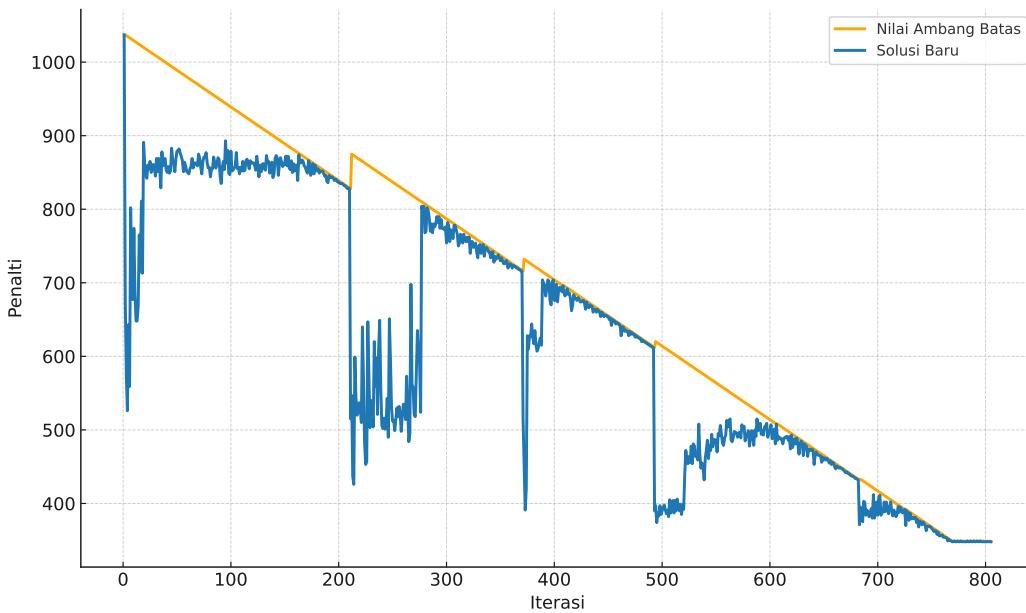
Gambar 8.3.3: Grafik Perubahan Solusi dalam Proses Optimasi pada Permasalahan Penjadwalan Ujian Praktikum



Gambar 8.3.4: Grafik Perubahan Solusi dalam Proses Optimasi pada Permasalahan Penjadwalan Sesi Praktikum



Gambar 8.3.5: Grafik Perubahan Solusi dalam Proses *Local Search* pada Permasalahan Penjadwalan Ujian Praktikum



Gambar 8.3.6: Grafik Perubahan Solusi dalam Proses *Local Search* pada Permasalahan Penjadwalan Sesi Praktikum

8.4 Perbandingan dengan Algoritma LAHC dan HC

Untuk dapat menilai bagaimana performa dari algoritma yang dikembangkan untuk menyelesaikan dua permasalahan ini, maka hasil penelitian dibandingkan dengan hasil penerapan kedua permasalahan ini pada dua algoritma pembanding yaitu LAHC dan HC. Kedua algoritma pembanding diimplementasikan dengan metode dan LLH yang sama. Pada algoritma LAHC terdapat satu nilai parameter (jumlah penalti solusi terdahulu yang bisa disimpan) yang membutuhkan proses pengaturan parameter. Tiga nilai diujicoba yaitu 1000, 2500 dan 5000. Hasilnya nilai 5000 menghasilkan solusi yang paling baik walaupun tidak menunjukkan hasil yang signifikan. Berdasarkan hasil tersebut, nilai 5000 dipergunakan pada algoritma LAHC. Selain itu algoritma LAHC dan HC diterapkan dengan tidak harus selalu menerima solusi *feasible*, namun solusi *feasible* terbaik yang ditemukan disimpan dan digunakan sebagai hasil akhir. Hal ini diakibatkan dari permasalahan yang memiliki jumlah slot waktu yang terbatas, sehingga jika menerapkan strategi hanya menerima solusi *feasible* pada setiap iterasinya, kedua algoritma sering tidak mampu untuk menghasilkan solusi terbaik baru. Terakhir, daftar solusi yang disimpan oleh algoritma LAHC umumnya menggunakan solusi awal. Namun dalam proses uji coba, jika menerapkan strategi tersebut, algoritma LAHC sering kali tidak mampu menghasilkan

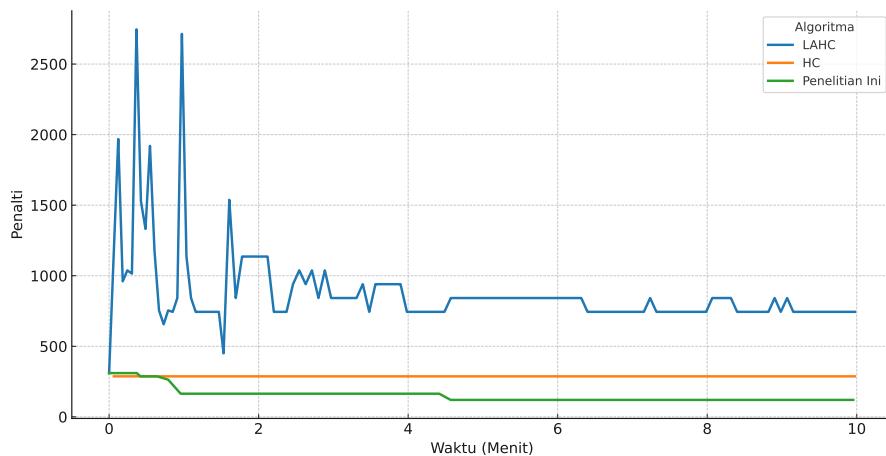
solusi baru karena sudah terjebak dalam kondisi *local optima*. Untuk mengatasi hal tersebut, daftar solusi diisi dengan nilai penalti dari solusi awal yang dikalikan dengan 10 dengan harapan algoritma mampu menjalankan proses optimasi dengan lebih baik.

Hasil dari perbandingan ditampilkan pada Tabel 8.4.1. Dari hasil ini menunjukan algoritma AT-ILS unggul terhadap algoritma LAHC dan HC dari metrik rata-rata dan solusi terbaik. Algoritma AT-ILS mampu menghasilkan solusi yang lebih baik dengan rentang persentase 10-77% pada hasil rata-rata dan 0-93% pada hasil terbaik. Sementara untuk hasil konsistensi, algoritma AT-ILS menghasilkan hasil konsistensi yang lebih buruk dari pada algoritma LAHC dan HC. Namun solusi yang dihasilkan memiliki jarak yang jauh dengan kedua algoritma pembanding. Hal ini ditunjukkan hasil terburuk dari penelitian ini memiliki solusi yang mendekati hasil rata-rata dari algoritma LAHC dan HC. Bahkan pada Dataset 2 untuk kedua permasalahan, hasil terburuk dari penelitian ini lebih baik dari pada hasil rata-rata dari pada algoritma LAHC dan HC. Hal ini menunjukkan hasil konsistensi pada algoritma LAHC dan HC, tidak menjadikan faktor yang menunjukkan kedua algoritma tersebut lebih baik karena perbedaan yang signifikan antara hasil algoritma AT-ILS dengan hasil dari algoritma LAHC dan HC.

Tabel 8.4.1: Perbandingan Hasil dari Penelitian ini dengan Algoritma LAHC dan HC

Permasalahan	Dataset	Algoritma	Rata-rata	Minimum	Maksimum	KV (%)
Ujian Praktikum	Dataset 1	AT-ILS	160,6	130	218	22
		LAHC	202,8	130	252	24
		HC	178,4	130	218	21
	Dataset 2	AT-ILS	121,1	39	188	44
		LAHC	203,0	98	326	42
		HC	273,8	111	353	26
Sesi Praktikum	Dataset 1	AT-ILS	111,2	74	194	39
		LAHC	180,2	114	234	23
		HC	173,2	156	194	10
	Dataset 2	AT-ILS	63,8	14	130	55
		LAHC	130,3	58	215	48
		HC	284,7	204	348	18

Untuk melihat perbedaan proses optimasi antara ketiga algoritma, Gambar 8.4.1 menunjukkan visualisasi dari proses optimasi pada ketiga algoritma menggunakan dataset dan solusi awal yang sama. Algoritma AT-ILS menunjukkan beberapa perubahan solusi dalam proses optimasi. Sementara pada algoritma HC, proses pencarian sudah terjebak pada kondisi *local optima* sejak awal iterasi. Sementara pada algoritma LAHC, pergerakan solusi yang besar terjadi di awal iterasi. Namun upaya algoritma dalam mengarahkan solusi ke arah konvergensi tidak berjalan dengan baik. Pada contoh visualisasi ini solusi terjebak pada rentang penalti 800-1000 hingga akhir iterasi. Pada algoritma AT-ILS, proses *perturbation* dan *local search* berperan untuk melakukan eksplorasi solusi. Sementara *move acceptance* memastikan konvergensi dan feasibilitas solusi terjaga. Hal ini yang menyebabkan hasil yang lebih baik pada penelitian ini.



Gambar 8.4.1: Grafik Pergerakan Solusi dalam Proses Optimasi pada Algoritma dalam Penelitian Ini, LAHC dan HC

8.5 Kesimpulan

Bab ini menyajikan uji coba terakhir pada permasalahan nyata penjadwalan sesi praktikum dan ujian praktikum dalam Departemen Sistem Informasi ITS. Kedua permasalahan ini memiliki tantangan yang berbeda dengan tiga *benchmark* sebelumnya. Permasalahan ini menghadirkan permasalahan penjadwalan dengan ukuran permasalahan kecil namun sulit untuk menemukan solusi *feasible* akibat dari keterbatasan slot waktu yang ada.

Hasil uji coba dari dua permasalahan ini menunjukan hasil yang baik dalam tahapan menghasilkan solusi *feasible* dan tahapan mengoptimasi solusi. Pada tahapan menghasilkan solusi *feasible*, algoritma yang dikembangkan mampu menghasilkan solusi *feasible* dengan persentase keberhasilan 100%. Sementara dalam tahapan optimasi, algoritma yang dikembangkan mampu menghasilkan solusi yang unggul dari algoritma LAHC dan HC secara signifikan dengan perbedaan mencapai 93%.

Uji coba ini merupakan uji coba terakhir dalam penelitian ini. Hasil ini semakin menegaskan kemampuan algoritma yang dikembangkan dalam menyelesaikan permasalahan penjadwalan lintas domain. Permasalahan ini yang menyajikan karakteristik yang berbeda dengan uji coba sebelumnya. Namun algoritma yang dikembangkan tetap mampu menghasilkan solusi yang baik, dalam aspek menghasilkan solusi *feasible* dan dalam aspek optimasi.

BAB 9

KESIMPULAN DAN SARAN

Penelitian ini memberikan kontribusi dalam pengembangan algoritma untuk menyelesaikan permasalahan penjadwalan lintas domain. Dengan memanfaatkan pendekatan hiper-heuristik, algoritma yang dikembangkan, seperti dijelaskan pada Bab 4, secara konsisten mampu menghasilkan solusi *feasible* serta mengoptimasi solusi penjadwalan. Pengembangan algoritma ini didasarkan pada kerangka dasar *Iterated Local Search* (ILS) dan dirancang melalui dua pendekatan utama: menghasilkan solusi *feasible* dan mengoptimasi solusi tersebut.

Pada aspek pengembangan algoritma untuk menghasilkan solusi *feasible*, penelitian ini berfokus pada peningkatan eksplorasi ruang solusi. Algoritma yang diusulkan, yaitu algoritma Progressive Acceptance Iterated Local Search (PA-ILS), terdiri atas dua tahapan utama, yaitu *perturbation* dan *local search*. Setiap solusi yang dihasilkan dari kedua tahapan tersebut selalu diterima, sehingga menjadikan pendekatan ini dinamakan *progressive acceptance*. Pada tahap *perturbation*, algoritma PA-ILS memilih slot waktu secara acak atau berdasarkan probabilitas slot dengan nilai konflik paling rendah. Sementara itu, tahap *local search* memastikan bahwa solusi baru selalu sama baiknya atau lebih baik dibandingkan solusi sebelumnya. Selain itu, strategi pengacakan diterapkan untuk meningkatkan peluang menemukan solusi yang lebih baik pada iterasi selanjutnya.

Untuk pengembangan algoritma optimasi, penelitian ini menawarkan inovasi berupa menghilangkan kebutuhan pengaturan parameter melalui penerapan konsep nilai ambang batas. Algoritma yang dikembangkan, algoritma Adaptive Threshold-Iterated Local Search (AT-ILS), menerapkan nilai ambang batas pada tiga tahapan: *perturbation*, *local search*, dan *move acceptance*. Pada tahap *perturbation*, solusi diubah hingga mencapai nilai tertentu sesuai ambang batas yang ditetapkan. Tahap *local search* menggunakan nilai

ambang batas dinamis untuk mengarahkan pencarian menuju konvergensi. Adapun tahap *move acceptance* memungkinkan penerimaan solusi dengan penalti lebih tinggi secara probabilistik, sambil tetap menjaga peluang untuk kembali ke solusi terbaik yang telah ditemukan.

Uji coba algoritma dilakukan pada tiga *benchmark* dan dua studi kasus nyata. Pada *benchmark* Toronto (Bab 5), algoritma PA-ILS meraih tingkat keberhasilan 100% dalam menemukan solusi *feasible*. Dari segi optimasi, algoritma AT-ILS menempati peringkat ke-4 di antara 27 penelitian sebelumnya, serta berhasil mencatatkan solusi terbaik baru untuk dataset EAR83.

Selanjutnya, pengujian pada *benchmark* International Timetabling Competition (ITC) 2019 (Bab 6) menghasilkan temuan serupa. Algoritma PA-ILS mampu memproduksi solusi *feasible* pada seluruh dataset dengan tingkat keberhasilan 100%. Dalam perbandingan dengan sembilan penelitian terdahulu, algoritma AT-ILS menunjukkan performa optimasi yang baik dengan menempati posisi keempat.

Pada *benchmark* ITC 2021 (Bab 7), kedua algoritma kembali menunjukkan performa yang unggul. Dalam pencarian solusi *feasible*, algoritma PA-ILS berhasil menemukan solusi *feasible* pada seluruh dataset dengan persentase keberhasilan 94,7%, yang merupakan pencapaian terbaik hingga saat ini. Dalam aspek optimasi, penelitian ini menempati peringkat ke-4 dari 9 penelitian, serta mencatatkan solusi terbaik baru pada dua dataset, yakni Middle 2 dan Middle 3.

Pengujian pada permasalahan nyata di Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (Bab 8) juga mengonfirmasi performa yang baik dari kedua algoritma. Algoritma PA-ILS dan AT-ILS sukses menyelesaikan dua jenis permasalahan penjadwalan, yaitu penjadwalan sesi praktikum dan penjadwalan ujian praktikum. Algoritma PA-ILS terbukti memiliki kemampuan yang baik dalam menghasilkan solusi *feasible*, sementara algoritma AT-ILS menunjukkan keunggulan dalam hal optimasi. Dibandingkan dengan algoritma Late Acceptance Hill Climbing dan Hill Climbing, algoritma yang diusulkan mampu menyediakan solusi dengan kualitas hingga 93% lebih baik.

9.1 Kesimpulan

Secara keseluruhan, algoritma yang dikembangkan dalam penelitian ini memperlihatkan kemampuan menjanjikan dalam menyelesaikan permasalahan penjadwalan lintas domain. Dari 91 *dataset* yang diuji, algoritma PA-ILS berhasil menghasilkan solusi *feasible* dengan tingkat keberhasilan 97,4%. Bila dibandingkan penelitian terdahulu, hasil algoritma PA-ILS dan AT-ILS secara konsisten berada di atas rata-rata dan bahkan menghasilkan solusi terbaik baru pada tiga *dataset*. Selain itu, dalam studi kasus nyata, algoritma AT-ILS membuktikan keunggulannya dibandingkan algoritma pembanding. Hasil-hasil tersebut menegaskan kemampuan algoritma yang dikembangkan dalam mengatasi permasalahan penjadwalan lintas domain.

9.2 Saran Penelitian Selanjutnya

Berdasarkan hasil penelitian ini, terdapat beberapa peluang penelitian yang dapat dilakukan di masa depan:

- **Uji Kemampuan Solusi *Feasible* pada Algoritma AT-ILS:** Meskipun algoritma AT-ILS telah menunjukkan hasil yang baik dan keunggulan dengan tidak memerlukan pengaturan parameter, namun algoritma ini belum pernah diuji dalam upaya untuk mencari solusi *feasible*. Penelitian selanjutnya dapat menguji algoritma ini dalam proses pencarian solusi *feasible* untuk mengevaluasi kemampuannya dalam menghasilkan solusi *feasible*.
- **Pengujian pada Permasalahan Lain:** Penelitian ini telah menguji algoritma pada tiga *benchmark* dan satu studi kasus. Walaupun pengujian cukup beragam, penelitian selanjutnya dapat menguji algoritma ini pada permasalahan yang belum dieksplorasi, seperti penjadwalan tingkat SMA pada *benchmark* ITC 2011, penjadwalan perawat di rumah sakit, atau bahkan permasalahan di luar penjadwalan, seperti Traveling Salesman Problem, Vehicle Routing Problem, atau Knapsack Problem.
- **Pengembangan Strategi Optimasi:** Walaupun algoritma AT-ILS menghasilkan hasil di atas rata-rata terhadap penelitian terdahulu, namun masih terdapat ruang untuk melakukan peningkatan. Penelitian ini lebih berfokus terhadap pengembangan

strategi *move acceptance* dan belum melakukan eksplorasi terhadap strategi LLH *selection* dan juga LLH. Penelitian berikutnya dapat mencoba mengembangkan kedua aspek tersebut untuk mampu meningkatkan kualitas solusi namun tetap menjaga generalitas algoritma.

DAFTAR PUSTAKA

- Abdelhalim, E. A. and El Khayat, G. A. (2016), ‘A utilization-based genetic algorithm for solving the university timetabling problem (uga)’, *Alexandria Engineering Journal* **55**(2), 1395–1409.
- URL:** <https://www.sciencedirect.com/science/article/pii/S1110016816000703>
- Abdullah, S. and Alzaqebah, M. (2013), ‘A hybrid self-adaptive bees algorithm for examination timetabling problems’, *Applied Soft Computing* **13**(8), 3608–3620.
- URL:** <https://linkinghub.elsevier.com/retrieve/pii/S1568494613001294>
- Abdullah, S. and Turabieh, H. (2012), ‘On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems’, *Information Sciences* **191**, 146–168.
- URL:** <https://linkinghub.elsevier.com/retrieve/pii/S0020025511006670>
- Abdullah, S., Turabieh, H., McCollum, B. and McMullan, P. (2012), ‘A hybrid metaheuristic approach to the university course timetabling problem’, *Journal of Heuristics* **18**(1), 1–23.
- URL:** <https://link.springer.com/10.1007/s10732-010-9154-y>
- Abuhamadah, A., Ayob, M., Kendall, G. and Sabar, N. R. (2014), ‘Population based Local Search for university course timetabling problems’, *Applied Intelligence* **40**(1), 44–53.
- URL:** <http://link.springer.com/10.1007/s10489-013-0444-6>
- Al-Betar, M. A. (2021), ‘A β -hill climbing optimizer for examination timetabling problem’, *Journal of Ambient Intelligence and Humanized Computing* **12**(1), 653–666.
- URL:** <https://link.springer.com/10.1007/s12652-020-02047-2>
- Al-Betar, M. A., Khader, A. T. and Doush, I. A. (2014), ‘Memetic techniques for

examination timetabling', *Annals of Operations Research* **218**(1), 23–50.

URL: <http://link.springer.com/10.1007/s10479-013-1500-7>

Al-Betar, M. A., Khader, A. T. and Zaman, M. (2012), 'University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42**(5), 664–681.

URL: <http://ieeexplore.ieee.org/document/6135818/>

Aldeeb, B. A., Azmi Al-Betar, M., Md Norwawi, N., Alissa, K. A., Alsmadi, M. K., Hazaymeh, A. A. and Alzaqebah, M. (2021), 'Hybrid intelligent water Drops algorithm for examination timetabling problem', *Journal of King Saud University - Computer and Information Sciences* .

URL: <https://linkinghub.elsevier.com/retrieve/pii/S1319157821001634>

Alinia Ahandani, M., Vakil Baghmisheh, M. T., Badamchi Zadeh, M. A. and Ghaemi, S. (2012), 'Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem', *Swarm and Evolutionary Computation* **7**, 21–34.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S2210650212000417>

AlSalibi, B. A., Jelodar, M. B. and Venkat, I. (2013), 'A comparative study between the nearest neighbor and genetic algorithms: A revisit to the traveling salesman problem', *International Journal of Computer Science and Electronics Engineering (IJCSEE)* **1**(1), 110–123.

Alzaqebah, M. and Abdullah, S. (2014), 'An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling', *Journal of Scheduling* **17**(3), 249–262.

URL: <http://link.springer.com/10.1007/s10951-013-0352-y>

Alzaqebah, M. and Abdullah, S. (2015), 'Hybrid bee colony optimization for examination timetabling problems', *Computers & Operations Research* **54**, 142–154.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305054814002524>

Arora, S. and Barak, B. (2009), *Computational complexity: a modern approach*, Cambridge University Press.

Aziz, R. A., Ayob, M., Othman, Z., Ahmad, Z. and Sabar, N. R. (2017), 'An adaptive guided

variable neighborhood search based on honey-bee mating optimization algorithm for the course timetabling problem', *Soft Computing* **21**(22), 6755–6765.

URL: <http://link.springer.com/10.1007/s00500-016-2225-8>

Babaei, H., Karimpour, J. and Hadidi, A. (2015), 'A survey of approaches for university course timetabling problem', *Computers & Industrial Engineering* **86**, 43–59. Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems.

URL: <https://www.sciencedirect.com/science/article/pii/S0360835214003714>

Bai, R., Blazewicz, J., Burke, E. K., Kendall, G. and McCollum, B. (2012), 'A simulated annealing hyper-heuristic methodology for flexible decision support', *4OR* **10**(1), 43–66.

URL: <http://link.springer.com/10.1007/s10288-011-0182-8>

Bang-Jensen, J., Gutin, G. and Yeo, A. (2004), 'When the greedy algorithm fails', *Discrete optimization* **1**(2), 121–127.

Basseur, M. and Goëffon, A. (2015), 'Climbing combinatorial fitness landscapes', *Applied Soft Computing* **30**, 688–704.

URL: <https://www.sciencedirect.com/science/article/pii/S156849461500068X>

Battistutta, M., Schaeufle, A. and Urlı, T. (2017), 'Feature-based tuning of single-stage simulated annealing for examination timetabling', *Annals of Operations Research* **252**(2), 239–254.

URL: <http://link.springer.com/10.1007/s10479-015-2061-8>

Bellio, R., Ceschia, S., Di Gaspero, L. and Schaeufle, A. (2021), 'Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling', *Computers & Operations Research* **132**, 105300.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305054821000927>

Bellio, R., Di Gaspero, L. and Schaeufle, A. (2012), 'Design and statistical analysis of a hybrid local search algorithm for course timetabling', *Journal of Scheduling* **15**(1), 49–61.

URL: <http://link.springer.com/10.1007/s10951-011-0224-2>

Berthold, T., Koch, T. and Shinano, Y. (2021), Milp. try. repeat., in 'Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT', Vol. 2.

- Bettinelli, A., Cacchiani, V., Roberti, R. and Toth, P. (2015), ‘An overview of curriculum-based course timetabling’, *Top* **23**, 313–349.
- Blum, C. and Roli, A. (2001), ‘Metaheuristics in combinatorial optimization: Overview and conceptual comparison’, *ACM Comput. Surv.* **35**, 268–308.
- Bolaji, A. L., Khader, A. T., Al-Betar, M. A. and Awadallah, M. A. (2014), ‘University course timetabling using hybridized artificial bee colony with hill climbing optimizer’, *Journal of Computational Science* **5**(5), 809–818.
- URL:** <https://linkinghub.elsevier.com/retrieve/pii/S1877750314000441>
- Burke, E. K. and Bykov, Y. (2016), ‘An Adaptive Flex-Deluge Approach to University Exam Timetabling’, *INFORMS Journal on Computing* **28**(4), 781–794.
- URL:** <http://pubsonline.informs.org/doi/10.1287/ijoc.2015.0680>
- Burke, E. K. and Bykov, Y. (2017), ‘The late acceptance hill-climbing heuristic’, *European Journal of Operational Research* **258**(1), 70–78.
- Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. and Qu, R. (2013), ‘Hyper-heuristics: A survey of the state of the art’.
- Burke, E. K., Qu, R. and Soghier, A. (2014), ‘Adaptive selection of heuristics for improving exam timetables’, *Annals of Operations Research* **218**(1), 129–145.
- URL:** <http://link.springer.com/10.1007/s10479-012-1140-3>
- Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg, S. (2003), Hyper-heuristics: An emerging direction in modern search technology, in ‘Handbook of metaheuristics’, Springer, pp. 457–474.
- Bykov, Y. and Petrovic, S. (2016), ‘A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling’, *Journal of Scheduling* **19**(4), 479–492.
- URL:** <http://link.springer.com/10.1007/s10951-016-0469-x>
- Carter, M. W., Laporte, G. and Lee, S. Y. (1996), ‘Examination timetabling: Algorithmic strategies and applications’, *Journal of the Operational Research Society* **47**(3).
- Ceschia, S., Di Gaspero, L. and Schaerf, A. (2012), ‘Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling

- problem', *Computers & Operations Research* **39**(7), 1615–1624.
- URL:** <https://linkinghub.elsevier.com/retrieve/pii/S0305054811002759>
- Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R. and Kendall, G. (2021), 'A survey of university course timetabling problem: Perspectives, trends and opportunities', *IEEE Access* **9**, 106515–106529.
- Choong, S. S., Wong, L.-P. and Lim, C. P. (2018), 'Automatic design of hyper-heuristic based on reinforcement learning', *Information Sciences* **436**, 89–107.
- Cook, S. A. (1971), The complexity of theorem-proving procedures, in 'Proceedings of the Third Annual ACM Symposium on Theory of Computing', STOC '71, Association for Computing Machinery, New York, NY, USA, p. 151–158.
- URL:** <https://doi.org/10.1145/800157.805047>
- Cooper, T. B. and Kingston, J. H. (1996), The complexity of timetable construction problems, in 'Practice and Theory of Automated Timetabling: First International Conference Edinburgh, UK, August 29–September 1, 1995 Selected Papers 1', Springer, pp. 281–295.
- Czogalla, J. and Fink, A. (2012), 'Fitness landscape analysis for the no-wait flow-shop scheduling problem', *Journal of Heuristics* **18**, 25–51.
- Demeester, P., Bilgin, B., De Causmaecker, P. and Vanden Berghe, G. (2012), 'A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice', *Journal of Scheduling* **15**(1), 83–103.
- URL:** <http://link.springer.com/10.1007/s10951-011-0258-5>
- Di Gaspero, L., McCollum, B. and Schaerf, A. (2007), The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3), Technical report, Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1. 0, Queen's~....
- Dimitzas, A., Gogos, C., Valouxis, C., Tzallas, A. and Alefragis, P. (2022), A pragmatic approach for solving the sports scheduling problem, in 'In Proc. 13th Int. Conf. Pract. Theory Autom. Timetabling', Vol. 3, pp. 195–207.
- Dokeroglu, T., Kucukyilmaz, T. and Talbi, E.-G. (2024), 'Hyper-heuristics: A survey and

taxonomy’, *Computers & Industrial Engineering* **187**, 109815.

URL: <https://www.sciencedirect.com/science/article/pii/S0360835223008392>

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. and Cosar, A. (2019), ‘A survey on new generation metaheuristic algorithms’, *Computers & Industrial Engineering* **137**, 106040.

URL: <https://www.sciencedirect.com/science/article/pii/S0360835219304991>

Dong, X., Lin, Q., Shen, F., Guo, Q. and Li, Q. (2023), ‘A novel hybrid simulated annealing algorithm for colored bottleneck traveling salesman problem’, *Swarm and Evolutionary Computation* **83**, 101406.

URL: <https://www.sciencedirect.com/science/article/pii/S2210650223001797>

Drake, J. H., Kheiri, A., Özcan, E. and Burke, E. K. (2020), ‘Recent advances in selection hyper-heuristics’, *European Journal of Operational Research* **285**(2), 405–428.

URL: <https://www.sciencedirect.com/science/article/pii/S0377221719306526>

Fong, C. W., Asmuni, H. and McCollum, B. (2015), ‘A hybrid swarm-based approach to university timetabling’, *IEEE Transactions on Evolutionary Computation* **19**(6), 870–884.

Fong, C. W., Asmuni, H., McCollum, B., McMullan, P. and Omatu, S. (2014a), ‘A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems’, *Information Sciences* **283**, 1–21.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0020025514005994>

Fong, C. W., Asmuni, H., McCollum, B., McMullan, P. and Omatu, S. (2014b), ‘A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems’, *Information Sciences* **283**, 1–21.

Fonseca, G. H. and Toffolo, T. A. (2022), ‘A fix-and-optimize heuristic for the itc2021 sports timetabling problem’, *Journal of Scheduling* **25**(3), 273–286.

Franke, C., Lepping, J. and Schwiegelshohn, U. (2007), Greedy scheduling with complex objectives, in ‘2007 IEEE Symposium on Computational Intelligence in Scheduling’, IEEE, pp. 113–120.

Garey, M. and Johnson, D. (1979), *Computers and Intractability: A Guide to the Theory of*

NP-completeness, Mathematical Sciences Series, Freeman.

URL: <https://books.google.co.id/books?id=fjxGAQAAIAAJ>

Geiger, M. J. (2012), ‘Applying the threshold accepting metaheuristic to curriculum based course timetabling’, *Annals of Operations Research* **194**(1), 189–202.

URL: <http://link.springer.com/10.1007/s10479-010-0703-4>

Gendreau, M., Potvin, J.-Y. et al. (2010), *Handbook of metaheuristics*, Vol. 2, Springer.

Gogos, C., Alefragis, P. and Housos, E. (2012), ‘An improved multi-staged algorithmic process for the solution of the examination timetabling problem’, *Annals of Operations Research* **194**(1), 203–221.

URL: <http://link.springer.com/10.1007/s10479-010-0712-3>

Goh, S. L., Kendall, G. and Sabar, N. R. (2017), ‘Improved local search approaches to solve the post enrolment course timetabling problem’, *European Journal of Operational Research* **261**(1), 17–29.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221717300759>

Goh, S. L., Kendall, G. and Sabar, N. R. (2019), ‘Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem’, *Journal of the Operational Research Society* **70**(6), 873–888.

URL: <https://www.tandfonline.com/doi/full/10.1080/01605682.2018.1468862>

Goh, S. L., Kendall, G., Sabar, N. R. and Abdullah, S. (2020), ‘An effective hybrid local search approach for the post enrolment course timetabling problem’, *OPSEARCH* **57**(4), 1131–1163.

URL: <https://link.springer.com/10.1007/s12597-020-00444-x>

Goldreich, O. (2010), *P, NP, and NP-Completeness: The basics of computational complexity*, Cambridge University Press.

Gotlieb, C. (1963), The construction of class-teacher timetables, in ‘IFIP congress’, Vol. 62, pp. 73–77.

Gümüş, D. B., Özcan, E., Atkin, J. and Drake, J. H. (2023), ‘An investigation of f-race training strategies for cross domain optimisation with memetic algorithms’, *Information Sciences* **619**, 153–171.

Hao, X., Qu, R. and Liu, J. (2021), ‘A Unified Framework of Graph-Based Evolutionary Multitasking Hyper-Heuristic’, *IEEE Transactions on Evolutionary Computation* **25**(1), 35–47.

URL: <https://ieeexplore.ieee.org/document/9084121/>

Holm, D. S., Mikkelsen, R., Sørensen, M. and Stidsen, T. J. (2022), ‘A graph-based mip formulation of the international timetabling competition 2019’, *Journal of Scheduling* **25**.

Hussain, K., Mohd Salleh, M. N., Cheng, S. and Shi, Y. (2019), ‘Metaheuristic research: a comprehensive survey’, *Artificial intelligence review* **52**, 2191–2233.

Hutama, R. R. and Muklason, A. (2021), ‘Pembentukan solusi awal international timetabling competition 2021’, *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)* **8**(4), 1939–1944.

Jamili, A., Shafia, M. A., Sadjadi, S. J. and Tavakkoli-Moghaddam, R. (2012), ‘Solving a periodic single-track train timetabling problem by an efficient hybrid algorithm’, *Engineering Applications of Artificial Intelligence* **25**(4), 793–800.

Jaradat, G., Ayob, M. and Ahmad, Z. (2014), ‘On the performance of Scatter Search for post-enrolment course timetabling problems’, *Journal of Combinatorial Optimization* **27**(3), 417–439.

URL: <http://link.springer.com/10.1007/s10878-012-9521-8>

Johannesson, P. and Perjons, E. (2014), *An introduction to design science*, Vol. 10, Springer.

Karp, R. M. (2010), *Reducibility among combinatorial problems*, Springer.

Kiefer, A., Hartl, R. F. and Schnell, A. (2017), ‘Adaptive large neighborhood search for the curriculum-based course timetabling problem’, *Annals of Operations Research* **252**(2), 255–282.

URL: <http://link.springer.com/10.1007/s10479-016-2151-2>

Köksalmaş, E., Garcia, C. and Rabadi, G. (2014), ‘The optimal exam experience: a timetabling approach to prevent student cheating and fatigue’, *International Journal of Operational Research* **21**(3), 263–278.

Kyngäs, J., Nurmi, K., Kyngäs, N., Lilley, G., Salter, T. and Goossens, D. (2017),

‘Scheduling the australian football league’, *Journal of the Operational Research Society* **68**(8), 973–982.

Lamas-Fernandez, C., Martinez-Sykora, A. and Potts, C. N. (2021), Scheduling double round-robin sports tournaments, in ‘Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT’, Vol. 2.

Larabi-Marie-Sainte, S., Jan, R., Al-Matouq, A. and Alabduhadi, S. (2021), ‘The impact of timetable on student’s absences and performance’, *Plos one* **16**(6), e0253256.

Lei, Y., Gong, M., Jiao, L., Shi, J. and Zhou, Y. (2017), ‘An adaptive coevolutionary memetic algorithm for examination timetabling problems’, *International Journal of Bio-Inspired Computation* **10**(4), 248.

URL: <http://www.inderscience.com/link.php?id=87918>

Lei, Y., Gong, M., Jiao, L. and Zuo, Y. (2015), ‘A memetic algorithm based on hyper-heuristics for examination timetabling problems’, *International Journal of Intelligent Computing and Cybernetics* **8**(2), 139–151.

URL: <https://www.emerald.com/insight/content/doi/10.1108/IJICC-02-2015-0005/full/html>

Leite, N., Fernandes, C. M., Melício, F. and Rosa, A. C. (2018), ‘A cellular memetic algorithm for the examination timetabling problem’, *Computers & Operations Research* **94**, 118–138.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305054818300455>

Leite, N., Melício, F. and Rosa, A. C. (2019), ‘A fast simulated annealing algorithm for the examination timetabling problem’, *Expert Systems with Applications* **122**, 137–151.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0957417418308169>

Lemos, A., Monteiro, P. T. and Lynce, I. (2020), ‘Itc 2019: University course timetabling with maxsat’, *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling (PATAT) Volume 1*.

Lemos, A., Monteiro, P. T. and Lynce, I. (2021), ‘Introducing unicort: an iterative university course timetabling tool with maxsat’, *Journal of Scheduling* .

URL: <https://link.springer.com/10.1007/s10951-021-00695-6>

Lester, M. M. (2022), ‘Pseudo-boolean optimisation for robinx sports timetabling’, *Journal of Scheduling* **25**(3), 287–299.

Lewis, R. (2007), ‘A survey of metaheuristic-based techniques for University Timetabling problems’, *OR Spectrum* **30**(1), 167–190.

URL: <http://link.springer.com/10.1007/s00291-007-0097-0>

Lewis, R. (2012), ‘A time-dependent metaheuristic algorithm for post enrolment-based course timetabling’, *Annals of Operations Research* **194**(1), 273–289.

URL: <http://link.springer.com/10.1007/s10479-010-0696-z>

Lewis, R. and Paechter, B. (2007), ‘Finding feasible timetables using group-based operators’, *IEEE Transactions on Evolutionary Computation* **11**(3), 397–413.

Li, J., Bai, R., Shen, Y. and Qu, R. (2015), ‘Search with evolutionary ruin and stochastic rebuild: A theoretic framework and a case study on exam timetabling’, *European Journal of Operational Research* **242**(3), 798–806.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221714009060>

Li, W., Ding, Y., Yang, Y., Sherratt, R. S., Park, J. H. and Wang, J. (2020), ‘Parameterized algorithms of fundamental np-hard problems: a survey’, *Human-centric Computing and Information Sciences* **10**, 1–24.

Liang, Y.-C., Lin, Y.-Y., Chen, A. H.-L. and Chen, W.-S. (2021), ‘Variable neighborhood search for major league baseball scheduling problem’, *Sustainability* **13**(7), 4000.

Malan, K. M. and Engelbrecht, A. P. (2013), ‘A survey of techniques for characterising fitness landscapes and some possible ways forward’, *Information Sciences* **241**, 148–163.

URL: <https://www.sciencedirect.com/science/article/pii/S0020025513003125>

Mandal, A. K., Kahar, M. N. M. and Kendall, G. (2020), ‘Addressing Examination Timetabling Problem Using a Partial Exams Approach in Constructive and Improvement’, *Computation* **8**(2), 46.

URL: <https://www.mdpi.com/2079-3197/8/2/46>

Mikkelsen, R. and Holm, D. S. (2022), ‘A parallelized matheuristic for the international timetabling competition 2019’, *Journal of Scheduling* **25**.

Moradi, N., Kayvanfar, V. and Rafiee, M. (2022), ‘An efficient population-based

simulated annealing algorithm for 0–1 knapsack problem’, *Engineering with Computers* **38**(3), 2771–2790.

Mühlenthaler, M. and Mühlenthaler, M. (2015), ‘The university course timetabling problem’, *Fairness in Academic Course Timetabling* pp. 11–73.

Muklason, A. (2017), Hyper-heuristics and fairness in examination timetabling problems, PhD thesis, University of Nottingham.

Nagata, Y. (2018a), ‘Random partial neighborhood search for the post-enrollment course timetabling problem’, *Computers & Operations Research* **90**, 84–96.

Nagata, Y. (2018b), ‘Random partial neighborhood search for the post-enrollment course timetabling problem’, *Computers & Operations Research* **90**, 84–96.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305054817302447>

Najaran, M. H. T. (2021), ‘How to exploit fitness landscape properties of timetabling problem: A new operator for quantum evolutionary algorithm’, *Expert Systems with Applications* **168**, 114211.

URL: <https://www.sciencedirect.com/science/article/pii/S0957417420309386>

Nothegger, C., Mayer, A., Chwatal, A. and Raidl, G. R. (2012), ‘Solving the post enrolment course timetabling problem by ant colony optimization’, *Annals of Operations Research* **194**(1), 325–339.

URL: <http://link.springer.com/10.1007/s10479-012-1078-5>

Oude Vrielink, R. A., Jansen, E., Hans, E. W. and van Hillegersberg, J. (2019), ‘Practices in timetabling in higher education institutions: a systematic review’, *Annals of operations research* **275**(1), 145–160.

Pais, T. C. and Amaral, P. (2012), ‘Managing the tabu list length using a fuzzy inference system: an application to examination timetabling’, *Annals of Operations Research* **194**(1), 341–363.

URL: <http://link.springer.com/10.1007/s10479-011-0867-6>

Pandey, J. and Sharma, A. K. (2016), Survey on university timetabling problem, in ‘2016 3rd International Conference on Computing for Sustainable Global Development (INDIACoM)’, pp. 160–164.

- Phillips, A. E., O'Sullivan, M. and Walker, C. (2021), An adaptive large neighbourhood search matheuristic for the itc2021 sports timetabling competition, *in* ‘Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT’, Vol. 2.
- Pillay, N. (2012), ‘Evolving hyper-heuristics for the uncapacitated examination timetabling problem’, *Journal of the Operational Research Society* **63**(1), 47–58.
URL: <https://www.tandfonline.com/doi/full/10.1057/jors.2011.12>
- Pillay, N. (2016a), ‘Incorporating chaos into the developmental approach for solving the examination timetabling problem’, *International Journal of Bio-Inspired Computation* **8**(6), 355.
URL: <http://www.inderscience.com/link.php?id=81327>
- Pillay, N. (2016b), ‘A review of hyper-heuristics for educational timetabling’, *Annals of Operations Research* **239**(1), 3–38.
- Pillay, N. and Özcan, E. (2019), ‘Automated generation of constructive ordering heuristics for educational timetabling’, *Annals of Operations Research* **275**(1), 181–208.
URL: <http://link.springer.com/10.1007/s10479-017-2625-x>
- Premananda, I. G. A. and Muklason, A. (2021), ‘Complex university timetabling using iterative forward search algorithm and great deluge algorithm’, *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika* **7**.
- Premananda, I. G. A., Muklason, A. and Hutama, R. R. (2022), ‘A solving route optimization of airplane travel problem use artificial bee colony algorithm’, *International Journal on Advanced Science, Engineering and Information Technology* **12**(6), 2363–2369.
- Premananda, I. G. A., Tjahyanto, A. and Muklason, A. (2022), Hybrid whale optimization algorithm for solving timetabling problems of itc 2019, *in* ‘2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)’, pp. 317–322.
- Premananda, I. G. A., Tjahyanto, A. and Muklason, A. (2024), ‘Timetabling problems and the effort towards generic algorithms: A comprehensive survey’, *IEEE Access* pp. 1–1.

Qu, R., Pham, N., Bai, R. and Kendall, G. (2015), ‘Hybridising heuristics within an estimation distribution algorithm for examination timetabling’, *Applied Intelligence* **42**(4), 679–693.

URL: <http://link.springer.com/10.1007/s10489-014-0615-0>

Rappos, E., Thiémard, E., Robert, S. and Hêche, J. F. (2022), ‘A mixed-integer programming approach for solving university course timetabling problems’, *Journal of Scheduling* **25**.

Reeves, C. R. (1999), ‘Landscapes, operators and heuristic search’, *Annals of Operations Research* **86**(0), 473–490.

Rezaeipanah, A., Matoori, S. S. and Ahmadi, G. (2021), ‘A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search’, *Applied Intelligence* **51**(1), 467–492.

URL: <https://link.springer.com/10.1007/s10489-020-01833-x>

Rosati, R. M., Petris, M., Di Gaspero, L. and Schaerf, A. (2022), ‘Multi-neighborhood simulated annealing for the sports timetabling competition itc2021’, *Journal of Scheduling* **25**(3), 301–319.

Sabar, N. R., Ayob, M., Kendall, G. and Qu, R. (2012), ‘A honey-bee mating optimization algorithm for educational timetabling problems’, *European Journal of Operational Research* **216**(3), 533–543.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221711007181>

Sabar, N. R., Ayob, M., Kendall, G. and Qu, R. (2013), ‘Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems’, *IEEE Transactions on Evolutionary Computation* **17**(6), 840–861.

URL: <http://ieeexplore.ieee.org/document/6595625/>

Sabar, N. R., Ayob, M., Kendall, G. and Rong Qu (2015), ‘A Dynamic Multiarmed Bandit-Gene Expression Programming Hyper-Heuristic for Combinatorial Optimization Problems’, *IEEE Transactions on Cybernetics* **45**(2), 217–228.

URL: <http://ieeexplore.ieee.org/document/6824192/>

Sabar, N. R., Ayob, M., Qu, R. and Kendall, G. (2012), ‘A graph coloring constructive

hyper-heuristic for examination timetabling problems', *Applied Intelligence* **37**(1), 1–11.

URL: <http://link.springer.com/10.1007/s10489-011-0309-9>

Sabar, N. R. and Kendall, G. (2015), 'Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems', *Information Sciences* **314**, 225–239.

Schaerf, A. (1999), 'A survey of automated timetabling', *Artificial intelligence review* **13**(2), 87–127.

Sipser, M. (1996), 'Introduction to the theory of computation', *ACM Sigact News* **27**(1), 27–29.

Sloan, D., Manns, H., Mellor, A. and Jeffries, M. (2020), 'Factors influencing student non-attendance at formal teaching sessions', *Studies in Higher Education* **45**(11), 2203–2216.

Socha, K., Sampels, M. and Manfrin, M. (2003), 'Ant algorithms for the university course timetabling problem with regard to the state-of-the-art', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2611**.

Soghier, A. and Qu, R. (2013), 'Adaptive selection of heuristics for assigning time slots and rooms in exam timetables', *Applied Intelligence* **39**(2), 438–450.

URL: <http://link.springer.com/10.1007/s10489-013-0422-z>

Song, T., Chen, M., Xu, Y., Wang, D., Song, X. and Tang, X. (2021), 'Competition-guided multi-neighborhood local search algorithm for the university course timetabling problem', *Applied Soft Computing* **110**, 107624.

Song, T., Liu, S., Tang, X., Peng, X. and Chen, M. (2018), 'An iterated local search algorithm for the university course timetabling problem', *Applied Soft Computing* **68**, 597–608.

Soria-Alcaraz, J. A., Ochoa, G., Sotelo-Figeroa, M. A. and Burke, E. K. (2017), 'A methodology for determining an effective subset of heuristics in selection hyper-heuristics', *European Journal of Operational Research* **260**(3), 972–983.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221717300772>

Soria-Alcaraz, J. A., Ochoa, G., Swan, J., Carpio, M., Puga, H. and Burke, E. K. (2014), ‘Effective learning hyper-heuristics for the course timetabling problem’, *European Journal of Operational Research* **238**(1), 77–86.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377221714002859>

Soria-Alcaraz, J. A., Özcan, E., Swan, J., Kendall, G. and Carpio, M. (2016), ‘Iterated local search using an add and delete hyper-heuristic for university course timetabling’, *Applied Soft Computing* **40**, 581–593.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S1568494615007760>

Subba, E. and Stordal, O. J. L. (2021), Scheduling sports tournaments by mixed-integer linear programming and a cluster pattern approach: computational implementation using data from the international timetabling competition 2021, Master’s thesis.

Sumin, D. and Rodin, I. (2021), Milp based approaches for scheduling double round-robin tournaments, in ‘Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT’, Vol. 2.

Sylejmani, K., Gashi, E. and Ymeri, A. (2022), ‘Simulated annealing with penalization for university course timetabling’, *Journal of Scheduling* .

Sánchez, M., Cruz-Duarte, J. M., Ortíz-Bayliss, J. c., Ceballos, H., Terashima-Marin, H. and Amaya, I. (2020), ‘A systematic review of hyper-heuristics on combinatorial optimization problems’, *IEEE Access* **8**, 128068–128095.

Tan, J. S., Goh, S. L., Kendall, G. and Sabar, N. R. (2021), ‘A survey of the state-of-the-art of optimisation methodologies in school timetabling problems’, *Expert Systems with Applications* **165**, 113943.

URL: <https://www.sciencedirect.com/science/article/pii/S0957417420307314>

Thepphakorn, T. and Pongcharoen, P. (2020), ‘Performance improvement strategies on cuckoo search algorithms for solving the university course timetabling problem’, *Expert Systems with Applications* **161**, 113732.

Van Bulck, D. and Goossens, D. (2023), ‘The international timetabling competition on sports timetabling (itc2021)’, *European Journal of Operational Research* **308**(3), 1249–1267.

URL: <https://www.sciencedirect.com/science/article/pii/S0377221722009201>

- van Doornmalen, J., Hojny, C., Lambers, R. and Spieksma, F. (2021), A hybrid model to find schedules for double round robin tournaments with side constraints, in ‘Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT 2021’, pp. 412–419.
- Vassilev, V. K., Fogarty, T. C. and Miller, J. F. (2003), ‘Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application’, *Advances in evolutionary computing: theory and applications* pp. 3–44.
- Vazirani, V. V. (1997), ‘Approximation algorithms’, *Georgia Inst. Tech .*
- Vince, A. (2002), ‘A framework for the greedy algorithm’, *Discrete Applied Mathematics* **121**(1-3), 247–260.
- Williamson, D. P. and Shmoys, D. B. (2011), *The design of approximation algorithms*, Cambridge university press.
- Wu, Y., Tang, J., Yu, Y. and Pan, Z. (2015), ‘A stochastic optimization model for transit network timetable design to mitigate the randomness of traveling time by adding slack time’, *Transportation Research Part C: Emerging Technologies* **52**, 15–31.
- Zou, F., Chen, D., Liu, H., Cao, S., Ji, X. and Zhang, Y. (2022), ‘A survey of fitness landscape analysis for optimization’, *Neurocomputing* **503**, 129–139.
- URL:** <https://www.sciencedirect.com/science/article/pii/S0925231222008153>

LAMPIRAN A

Tabel A.1: Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 1

<i>Dataset</i>	Bellio et al. (2021)	Leite et al. (2018)	Burke and Bykov (2016)	Penelitian ini	Demeester et al. (2012)	Mandal et al. (2020)
<i>CAR92</i>	3,64	3,68	3,67	3,87	3,78	3,82
Peringkat	1	3	2	7	5	6
<i>CAR91</i>	4,24	4,31	4,32	4,48	4,52	4,58
Peringkat	1	2	3	5	6	7
<i>EAR83</i>	32,42	32,48	32,62	32,4	32,49	32,48
Peringkat	2	3,5	6	1	5	3,5
<i>HEC92</i>	10,03	10,03	10,06	10,05	10,03	10,32
Peringkat	2	2	5	4	2	8
<i>KFU93</i>	12,81	12,81	12,80	12,93	12,9	13,34
Peringkat	2,5	2,5	1	5	4	10
<i>LSE91</i>	9,78	9,78	9,78	9,85	10,04	10,24
Peringkat	2	2	2	4	6	9
<i>PUR93</i>	4,22	4,14	3,88	4,56	5,67	-
Peringkat	3	2	1	5	11	-
<i>RYE92</i>	7,84	7,89	7,91	8,19	8,05	9,79
Peringkat	1	2	3	5	4	19
<i>STA83</i>	157,03	157,03	157,03	157,03	157,03	157,03
Peringkat	6	6	6	6	6	6
<i>TRE92</i>	7,59	7,66	7,64	7,69	7,69	7,72
Peringkat	1	3	2	4,5	4,5	6

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 1

<i>Dataset</i>	Bellio et al. (2021)	Leite et al. (2018)	Burke and Bykov (2016)	Penelitian ini	Demeester et al. (2012)	Mandal et al. (2020)
<i>UTA92</i>	2,95	3,01	2,98	3,17	3,13	3,13
Peringkat	1	3	2	10	8	8
<i>UTE92</i>	24,76	24,80	24,78	24,77	24,77	25,28
Peringkat	1	5	4	2,5	2,5	15
<i>YOR83</i>	34,4	34,45	34,71	34,59	34,64	35,46
Peringkat	1	2	5	3	4	7
Rata-Rata Peringkat	1,88	2,92	3,23	4,77	5,23	8,71

Tabel A.2: Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 2

<i>Dataset</i>	Alzaqebah and Abdullah (2015)	Fong et al. (2014a)	Muklason (2017)	Pillay (2016a)	Alzaqebah and Abdullah (2014)	Sabar, Ayob, Kendall and Qu (2012)
<i>CAR92</i>	3,88	3,77	4,23	3,96	4,00	3,9
Peringkat	8	4	17	11	12,5	9
<i>CAR91</i>	4,38	4,71	5,24	–	4,62	4,79
Peringkat	4	11	23	0	8,5	12
<i>EAR83</i>	33,44	33,15	33,14	33,91	33,14	34,69
Peringkat	10	9	7,5	14	7,5	17
<i>HEC92</i>	10,39	10,38	10,08	10,43	10,43	10,66
Peringkat	10	9	6	12,5	12,5	18
<i>KFU93</i>	13,23	13,69	13,21	13,54	13,59	13,00
Peringkat	9	17	8	14	15	6,5
<i>LSE91</i>	10,52	10,25	10,56	10,23	10,75	10,00
Peringkat	13	10	14	8	16	5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 2

<i>Dataset</i>	Alzaqebah and Abdullah (2015)	Fong et al. (2014a)	Muklason (2017)	Pillay (2016a)	Alzaqebah and Abdullah (2014)	Sabar, Ayob, Kendall and Qu (2012)
<i>PUR93</i>	—	—	4,96	—	—	4,76
Peringkat	—	—	9	—	—	7
<i>RYE92</i>	8,92	—	8,51	8,88	9,17	10,97
Peringkat	10	—	6	9	14	24
<i>STA83</i>	157,06	157,03	157,03	156,55	157,06	157,04
Peringkat	14	6	6	1	14	11,5
<i>TRE92</i>	7,89	7,84	7,85	8,06	8,00	7,87
Peringkat	10	7	8	14	13	9
<i>UTA92</i>	3,13	3,10	3,57	—	3,27	3,10
Peringkat	8	4,5	21	—	11,5	4,50
<i>UTE92</i>	25,12	25,32	24,86	25,27	25,16	25,94
Peringkat	9	16	7	14	10	20
<i>YOR83</i>	35,49	36,06	35,03	36,57	35,58	36,15
Peringkat	8	12	6	16	9	13
Rata-Rata	9,42	9,59	10,65	11,35	11,96	12,04
Peringkat						

Tabel A.3: Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 3

<i>Dataset</i>	Al-Betar et al. (2014)	Abdullah and Alzaqebah (2013)	Sabar et al. (2013)	Aldeeb et al. (2021)	Qu et al. (2015)	Al-Betar (2021)
<i>CAR92</i>	4,29	3,94	4,00	4,59	4,09	4,22
Peringkat	19	10	12,5	25	14	15,5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 3

<i>Dataset</i>	Al-Betar et al. (2014)	Abdullah and Alzaqebah (2013)	Sabar et al. (2013)	Aldeeb et al. (2021)	Qu et al. (2015)	Al-Betar (2021)
<i>CAR91</i>	4,99	4,67	4,62	5,48	4,95	5,14
Peringkat	15	10	8,5	26	13,5	18,5
<i>EAR83</i>	34,42	33,61	34,71	33,64	34,97	34,38
Peringkat	16	12	18	13	19	15
<i>HEC92</i>	10,40	10,56	10,68	10,15	11,11	10,49
Peringkat	11	17	19	7	22	15
<i>KFU93</i>	13,5	13,44	13,00	13,49	14,09	13,64
Peringkat	13	11	6,5	12	19	16
<i>LSE91</i>	10,48	10,87	10,11	10,44	10,71	10,92
Peringkat	12	20	7	11	15	21,5
<i>PUR93</i>	-	-	4,80	-	4,73	-
Peringkat	-	-	8	-	6	-
<i>RYE92</i>	8,79	8,81	10,79	8,93	9,20	9,08
Peringkat	7	8	23	11	15	12
<i>STA83</i>	157,04	157,09	158,02	157,03	157,64	157,07
Peringkat	11,5	17	26	6	22,5	16
<i>TRE92</i>	8,16	7,94	7,90	8,34	8,27	8,52
Peringkat	15	12	11	18	16	25,5
<i>UTA92</i>	3,43	3,27	3,12	3,72	3,33	3,47
Peringkat	17	11,5	6	23	13,5	19
<i>UTE92</i>	25,09	25,36	26,00	24,81	26,18	25,23
Peringkat	8	17	21	6	22	13
<i>YOR83</i>	35,86	35,74	36,20	36,65	37,88	37,95
Peringkat	11	10	14,5	17	19	20
Rata-Rata	12,96	12,96	13,92	14,58	16,65	17,25
Peringkat						

Tabel A.4: Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 4

<i>Dataset</i>	Pais and Amaral (2012)	Pillay (2012)	Lei et al. (2017)	Li et al. (2015)	Alinia Ahandani et al. (2012)	Pillay and Özcan (2019)
<i>CAR92</i>	4,57	4,22	4,45	4,91	4,67	4,32
Peringkat	24	15,5	22	28	26	21
<i>CAR91</i>	5,46	4,95	5,43	5,81	5,22	5,16
Peringkat	25	13,5	24	27	22	20
<i>EAR83</i>	33,50	35,95	35,57	35,08	35,74	36,52
Peringkat	11	25	22	20	23	26
<i>HEC92</i>	10,52	11,27	10,71	10,44	10,74	11,87
Peringkat	16	24	20	14	21	27
<i>KFU93</i>	14,05	14,12	14,31	15,08	14,47	14,67
Peringkat	18	20	21	27	22	24
<i>LSE91</i>	-	10,76	11,26	11,28	10,76	10,81
Peringkat	-	17,5	25	26	17,5	19
<i>PUR93</i>	-	-	-	-	-	4,46
Peringkat	-	-	-	-	-	4
<i>RYE92</i>	9,11	9,23	9,84	-	9,95	9,48
Peringkat	13	16	20	-	21	18
<i>STA83</i>	157,29	157,69	157,18	157,06	157,10	157,64
Peringkat	21	24	19,5	14	18	22,5
<i>TRE92</i>	8,71	8,43	8,47	8,39	8,47	8,48
Peringkat	28	20	21,5	19	21,5	23
<i>UTA92</i>	3,71	3,33	-	-	3,52	3,35
Peringkat	22	13,5	-	-	20	15
<i>UTE92</i>	25,18	26,95	25,92	25,17	25,86	27,16
Peringkat	12	25	19	11	18	26
<i>YOR83</i>	39,08	39,63	36,20	36,97	38,72	41,31
Peringkat	22	25	14,5	18	21	28

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 4

<i>Dataset</i>	Pais and Amaral (2012)	Pillay (2012)	Lei et al. (2017)	Li et al. (2015)	Alinia Ahandani et al. (2012)	Pillay and Özcan (2019)
Rata-Rata Peringkat	19,27	19,92	20,35	20,40	20,92	21,04

Tabel A.5: Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 5

<i>Dataset</i>	Burke et al. (2014)	Hao et al. (2021)	Sabar, Ayob, Qu and Kendall (2012)	Lei et al. (2015)
<i>CAR92</i>	4,31	4,27	4,70	4,53
Peringkat	20	18	27	23
<i>CAR91</i>	5,19	5,11	5,14	5,03
Peringkat	21	17	18,5	16
<i>EAR83</i>	35,79	35,31	37,86	37,42
Peringkat	24	21	28	27
<i>HEC92</i>	11,19	11,45	11,90	11,53
Peringkat	23	25	28	26
<i>KFU93</i>	14,51	14,79	15,30	14,93
Peringkat	23	25	28	26
<i>LSE91</i>	10,92	11,07	12,33	11,23
Peringkat	21,5	23	27	24
<i>PUR93</i>	-	-	5,37	-
Peringkat	-	-	10	-
<i>RYE92</i>	-	9,30	10,71	-
Peringkat	-	17	22	-
<i>STA83</i>	157,18	158,50	160,12	157,91
Peringkat	19,5	27	28	25

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* Toronto Bagian 5

<i>Dataset</i>	Burke et al. (2014)	Hao et al. (2021)	Sabar, Ayob, Qu and Kendall (2012)	Lei et al. (2015)
<i>TRE92</i>	8,49	8,55	8,32	8,52
Peringkat	24	27	17	25,5
<i>UTA92</i>	3,44	3,38	3,88	3,75
Peringkat	18	16	25	24
<i>UTE92</i>	26,70	27,20	32,67	26,30
Peringkat	24	27	28	23
<i>YOR83</i>	39,47	39,59	40,53	41,01
Peringkat	23	24	26	27
Rata-Rata	21,91	22,25	24,04	24,23
Peringkat				

LAMPIRAN B

Tabel B.1: Hasil Pemeringkatan Penelitian pada *Benchmark ITC 2019* Kategori Early Bagian 1

Dataset	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Penelitian ini	Premananda, Tjahyanto and Muklason (2022)
<i>agh-fis-spr17</i>	3094	4557	6799	7780	9345
Peringkat	1	2	3	4	5
<i>agh-ggis-spr17</i>	35147	36616	77932	89081	92556
Peringkat	1	2	3	4	5
<i>bet-fal17</i>	290127	295427	299205	309109	320360
Peringkat	1	2	4	5	6
<i>iku-fal17</i>	18989	26840	50613	58373	86908
Peringkat	1	2	4	5	6
<i>mary-spr17</i>	14922	15021	15894	17880	27525
Peringkat	1	2	3	5	6
<i>muni-fi-spr16</i>	3764	3844	5006	6859	7698
Peringkat	1	2	3	4	5
<i>muni-fsp-spr17</i>	868	883	1938	8949	16161
Peringkat	1,5	3	4	5	6
<i>muni-pdf-spr16c</i>	35731	37487	58206	87813	140026
Peringkat	1	2	3	4	5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Early Bagian 1

<i>Dataset</i>	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Penelitian ini	Premananda, Tjahyanto and Muklason (2022)
<i>pu-llr-spr17</i>	10038	13385	16874	27729	35767
Peringkat	1	3	4	5	6
<i>tg-fal17</i>	4215	4215	8044	7818	11285
Peringkat	2	2	7	6	8
Rata-Rata	1,15	2,2	3,8	4,7	5,8
Peringkat					

Tabel B.2: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Early Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
<i>agh-fis-spr17</i>	35139	10858	Tidak <i>feasible</i>	35139
Peringkat	7,5	6	9	7,5
<i>agh-ggis-spr17</i>	161118	107558	Tidak <i>feasible</i>	194138
Peringkat	7	6	9	8
<i>bet-fal17</i>	296015	385290	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	3	7	8,5	8,5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Early Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
<i>iku-fal17</i>	29929	112187	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	3	7	8,5	8,5
<i>mary-spr17</i>	51147	29161	15932	51147
Peringkat	8,5	7	4	8,5
<i>muni-fi-spr16</i>	19314	11222	Tidak <i>feasible</i>	19314
Peringkat	7,5	6	9	7,5
<i>muni-fsp-spr17</i>	211142	150459	868	211142
Peringkat	8,5	7	1,5	8,5
<i>muni-pdf-spr16c</i>	567900	517913	Tidak <i>feasible</i>	567900
Peringkat	7,5	6	9	7,5
<i>pu-llr-spr17</i>	68003	55275	10710	68003
Peringkat	8,5	7	2	8,5
<i>tg-fal17</i>	6774	14548	4215	6774
Peringkat	4,5	9	2	4,5
Rata-Rata	6,55	6,8	6,25	7,75
Peringkat				

Tabel B.3: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Middle Bagian 1

<i>Dataset</i>	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Penelitian ini	Premananda, Tjahyanto and Muklason (2022)
<i>agh-ggos-spr17</i>	3237	6320	9328	15905	18985
Peringkat	1	2	3	4	5
<i>agh-h-spr17</i>	21559	26159	25081	29315	30893
Peringkat	1	3	2	4	5
<i>lums-spr18</i>	95	114	107	128	147
Peringkat	1,5	4	3	6	7
<i>muni-fi-spr17</i>	3796	4289	4692	5381	7127
Peringkat	1	2	3	4	5
<i>muni-fsps-spr17c</i>	2780	3303	9222	13614	31534
Peringkat	1	2	3	4	5
<i>muni-pdf-spr16</i>	19320	24318	40074	52565	77907
Peringkat	1	2	3	4	5
<i>nbi-spr18</i>	18014	19055	26517	32782	53254
Peringkat	1	3	4	5	8
<i>pu-d5-spr17</i>	15842	18813	19440	21250	24815
Peringkat	1	3	4	5	6
<i>pu-proj-fal19</i>	178135	561194	237909	237544	466488
Peringkat	2	6	4	3	5
<i>yach-fal17</i>	1410	1844	1727	1949	4206
Peringkat	1	3	2	4	5
Rata-Rata	1,15	3	3,1	4,3	5,6
Peringkat					

Tabel B.4: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Middle Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
<i>agh-ggos-spr17</i>	79745	22730	Tidak <i>feasible</i>	79745
Peringkat	7,5	6	9	7,5
<i>agh-h-spr17</i>	55887	32717	Tidak <i>feasible</i>	55887
Peringkat	7,5	6	9	7,5
<i>lums-spr18</i>	119	361	95	594
Peringkat	5	8	1,5	9
<i>muni-fi-spr17</i>	18080	10158	24572	18080
Peringkat	7,5	6	9	7,5
<i>muni-fsp-spr17c</i>	618217	488529	Tidak <i>feasible</i>	618217
Peringkat	7,5	6	9	7,5
<i>muni-pdf-spr16</i>	310994	212005	Tidak <i>feasible</i>	310994
Peringkat	7,5	6	9	7,5
<i>nbi-spr18</i>	49924	58516	18212	49924
Peringkat	6,5	9	2	6,5
<i>pu-d5-spr17</i>	17513	30762	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	2	7	8,5	8,5
<i>pu-proj-fal19</i>	126568	831950	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	1	7	8,5	8,5
<i>yach-fal17</i>	32198	8382	19046	32198
Peringkat	8,5	6	7	8,5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Middle Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
Rata-Rata Peringkat	6,05	6,7	7,25	7,85

Tabel B.5: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Late Bagian 1

<i>Dataset</i>	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Penelitian ini	Premananda, Tjahyanto and Muklason (2022)
<i>agh-fal17</i>	140194	Tidak <i>feasible</i>	184030	183741	215516
Peringkat	1	8	4	3	5
<i>bet-spr18</i>	350410	360057	360437	365852	376676
Peringkat	1	3	4	5	6
<i>iku-spr18</i>	25863	36711	85969	86121	111952
Peringkat	1	2	4	5	6
<i>lums-fal17</i>	349	386	486	531	717
Peringkat	1	2	4	5	6
<i>mary-fal18</i>	4331	5637	7199	7215	11194
Peringkat	1	2	3	4	5
<i>muni-fi-fal17</i>	3129	3794	4712	6100	7776
Peringkat	1	2	3	4	5
<i>muni-fpsx-fal17</i>	12390	33001	41933	65020	129419
Peringkat	1	2	3	4	5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Late Bagian 1

<i>Dataset</i>	Mikkelsen and Holm (2022)	Rappos et al. (2022)	Sylejmani et al. (2022)	Penelitian ini	Premananda, Tjahyanto and Muklason (2022)
<i>muni-pdfx-fal17</i>	84703	151464	159203	190830	281751
Peringkat	1	2	3	4	6
<i>pu-d9-fal19</i>	39251	134009	82757	101744	201122
Peringkat	1	5	3	4	6
<i>tg-spr18</i>	12704	12856	15992	24498	31710
Peringkat	1,5	3	4	5	6
Rata-Rata	1,05	3,1	3,5	4,3	5,6
Peringkat					

Tabel B.6: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2019 Kategori Late Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
<i>agh-fal17</i>	142687	493834	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	2	6	8	8
<i>bet-spr18</i>	353920	443992	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	2	7	8,5	8,5
<i>iku-spr18</i>	45537	149340	Tidak <i>feasible</i>	Tidak <i>feasible</i>
Peringkat	3	7	8,5	8,5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark ITC* 2019 Kategori Late Bagian 2

<i>Dataset</i>	Lemos et al. (2021)	Premananda and Muklason (2021)	Holm et al. (2022)	Lemos et al. (2020)
<i>lums-fal17</i> Peringkat	813 7	1467 9	405 3	1151 8
<i>mary-fal18</i> Peringkat	44097 7,5	23326 6	Tidak <i>feasible</i> 9	44097 7,5
<i>muni-fi-fal17</i> Peringkat	19683 7,5	12044 6	Tidak <i>feasible</i> 9	19683 7,5
<i>muni-fspsx-fal17</i> Peringkat	401155 6	1002804 7	Tidak <i>feasible</i> 8,5	Tidak <i>feasible</i> 8,5
<i>muni-pdfx-fal17</i> Peringkat	228560 5	871244 7	Tidak <i>feasible</i> 8,5	Tidak <i>feasible</i> 8,5
<i>pu-d9-fal19</i> Peringkat	71903 2	480907 7	Tidak <i>feasible</i> 8,5	Tidak <i>feasible</i> 8,5
<i>tg-spr18</i> Peringkat	31900 7,5	36242 9	12704 1,5	31900 7,5
Rata-Rata Peringkat	4,95	7,1	7,3	8,1

LAMPIRAN C

Tabel C.1: Hasil Pemeringkatan Penelitian pada *Benchmark ITC 2021* Kategori Early Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
1 Peringkat	362 1	423 4	421 3	388 2	804 7
2 Peringkat	222 1	318 4	309 3	354 5	402 7
3 Peringkat	1052 1	1068 2	1146 3	1161 4	1246 6
4 Peringkat	536 1	556 2	Tidak <i>feasible</i> 7,5	1359 4	764 3
5 Peringkat	3127 1	4117 3	Tidak <i>feasible</i> 6,5	4008 2	Tidak <i>feasible</i> 6,5
6 Peringkat	3714 1	3927 2	4088 4	4276 5	5506 7
7 Peringkat	4763 1	5205 2	6434 4	5826 3	6881 5
8 Peringkat	1114 3	1051 1	1064 2	1612 6	1409 5

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Early Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
9 Peringkat	108 1	132 3	538 6	848 7	122 2
10 Peringkat	3400 1	4986 3	Tidak <i>feasible</i> 6,5	4296 2	Tidak <i>feasible</i> 6,5
11 Peringkat	4436 1	4526 2	5127 4	4920 3	6843 6
12 Peringkat	510 1	1010 4	890 2	1300 7	1025 5,5
13 Peringkat	121 1	173 2	331 3	659 8	360 4
14 Peringkat	47 2	63 3,5	84 5	837 7	25 1
15 Peringkat	3368 1	3556 2	4196 3	5335 7	4616 5
Rata-Rata Peringkat	1,2	2,63	4,16	4,8	5,1

Tabel C.2: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Early Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
1 Peringkat	666 6	512 5	1209 8	Tidak <i>feasible</i> 9
2 Peringkat	379 6	266 2	461 8	Tidak <i>feasible</i> 9
3 Peringkat	1171 5	1354 7	2245 8	2686 9
4 Peringkat	Tidak <i>feasible</i> 7,5	Tidak <i>feasible</i> 7,5	2360 5	Tidak <i>feasible</i> 7,5
5 Peringkat	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5
6 Peringkat	4821 6	3957 3	5660 8	Tidak <i>feasible</i> 9
7 Peringkat	7208 6	9644 8	7879 7	Tidak <i>feasible</i> 9
8 Peringkat	1191 4	1614 7	5639 9	3212 8
9 Peringkat	447 4	448 5	5843 9	1828 8
10 Peringkat	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5
11 Peringkat	6713 5	8189 8	7149 7	Tidak <i>feasible</i> 9
12 Peringkat	925 3	1025 5,5	1955 9	1535 8
13 Peringkat	382 6	380 5	414 7	Tidak <i>feasible</i> 9
14	106	63	4884	1986

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Early Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
Peringkat	6	3,5	9	8
15	4667	4470	6297	6466
Peringkat	6	4	8	9
Rata-Rata	5,56	5,56	7,66	8,3
Peringkat				

Tabel C.3: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Middle Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
1	5177	5657	Tidak <i>feasible</i>	6240	Tidak <i>feasible</i>
Peringkat	1	2	6,5	3	6,5
2	7381	Tidak <i>feasible</i>	Tidak <i>feasible</i>	7115	Tidak <i>feasible</i>
Peringkat	2	6	6	1	6
3	9800	9542	9837	8648	9700
Peringkat	4	2	5	1	3
4	7	16	7	21	7
Peringkat	3	6	3	7	3
5	494	510	543	629	413
Peringkat	2	3	4	5	1
6	1275	1701	1630	2130	2270
Peringkat	1	3	2	6	7
7	2049	2203	2394	2237	2991
Peringkat	1	2	4	3	6

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Middle Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
8 Peringkat	129 1	136 2	200 4	302 7	174 3
9 Peringkat	450 1	640 2	1050 5	1195 6	810 3
10 Peringkat	1250 1	1357 2	1537 3	1869 5	1813 4
11 Peringkat	2608 1	2696 2	2798 3	3237 6	3367 7
12 Peringkat	923 2	950 3	1007 4	1354 5	1538 6
13 Peringkat	282 1	362 2	430 3	521 4	1051 7
14 Peringkat	1323 2	1172 1	1682 5	2216 8	1679 4
15 Rank	965 1	985 2	1089 3	1099 4	1614 6
Rata-Rata Peringkat	1,6	2,666666667	4,03333333	4,73333333	4,83333333

Tabel C.4: Hasil Pemeringkatan Penelitian pada *Benchmark ITC* 2021 Kategori Middle Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
1 Peringkat	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5
2 Peringkat	Tidak <i>feasible</i> 6	Tidak <i>feasible</i> 6	Tidak <i>feasible</i> 6	Tidak <i>feasible</i> 6
3 Peringkat	11235 6	12170 7	12504 8	Tidak <i>feasible</i> 9
4 Peringkat	7 3	7 3	82 8	96 9
5 Peringkat	681 6	732 7	3888 9	2016 8
6 Peringkat	2026 5	1900 4	3050 8	3135 9
7 Peringkat	3317 7	2792 5	5010 8	6701 9
8 Peringkat	277 5	301 6	371 8	997 9
9 Peringkat	1315 7	1015 4	2315 8,5	2315 8,5
10 Peringkat	2370 7	Tidak <i>feasible</i> 8,5	2024 6	Tidak <i>feasible</i> 8,5
11 Peringkat	3143 5	2956 4	3868 8	4223 9
12 Peringkat	911 1	1596 7	4142 9	3592 8
13 Peringkat	1044 6	780 5	3001 8	5758 9
14	1704	1619	1712	Tidak <i>feasible</i>

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Middle Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
Peringkat	6	3	7	9
15	1401	1833	4363	6947
Rank	5	7	8	9
Rata-Rata Peringkat	5,43333333	5,53333333	7,73333333	8,43333333

Tabel C.5: Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Late Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
1	1969	2021	2073	2804	2068
Peringkat	1	2	4	7	3
2	5400	5715	Tidak feasible	6270	5525
Peringkat	1	4	8	6	2
3	2369	2457	2474	2564	3069
Peringkat	1	2	3	4	7
4	0	0	0	0	0
Peringkat	5	5	5	5	5
5	2218	2341	Tidak feasible	2503	Tidak feasible
Peringkat	1	2	6,5	3	6,5
6	923	930	1082	1130	1212
Peringkat	1	2	3	4	5
7	1652	1765	2333	2255	2525
Peringkat	1	2	4	3	5
8	934	997	1165	1138	1454

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Late Bagian 1

<i>Dataset</i>	Lamas-Fernandez et al. (2021)	Rosati et al. (2022)	Fonseca and Toffolo (2022)	Penelitian ini	Berthold et al. (2021)
Peringkat	1	2	4	3	7
9	563	715	1219	1574	790
Peringkat	1	2	5	7	3
10	2031	2571	Tidak feasible	2964	2544
Peringkat	1	3	7,5	4	2
11	226	207	361	321	305
Peringkat	2	1	6	5	3
12	3912	3944	4786	4951	5669
Peringkat	1	2	3	4	6
13	2110	1868	1820	1960	3877
Peringkat	4	2	1	3	7
14	1363	1202	1562	2160	1433
Peringkat	2	1	4	7	3
15	40	60	160	870	40
Peringkat	1,5	3	6	7	1,5
Rata-Rata	1,63333333	2,333333333	4,66666667	4,8	4,4
Peringkat					

Tabel C.6: Hasil Pemeringkatan Penelitian pada *Benchmark ITC* 2021 Kategori Late Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
1 Peringkat	2406 6	2234 5	2987 8	3166 9
2 Peringkat	Tidak <i>feasible</i> 8	5680 3	6035 5	Tidak <i>feasible</i> 8
3 Peringkat	2900 5	3004 6	3739 8	7228 9
4 Peringkat	0 5	0 5	0 5	0 5
5 Peringkat	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5	Tidak <i>feasible</i> 6,5
6 Peringkat	1310 6	1440 7	2436 8	2910 9
7 Peringkat	2805 7	3009 8	2638 6	Tidak <i>feasible</i> 9
8 Peringkat	1252 5	1375 6	2843 9	2424 8
9 Peringkat	1343 6	1108 4	2220 8	2691 9
10 Peringkat	Tidak <i>feasible</i> 7,5	Tidak <i>feasible</i> 7,5	3256 5	Tidak <i>feasible</i> 7,5
11 Peringkat	376 7	511 8	311 4	Tidak <i>feasible</i> 9
12 Peringkat	5542 5	7218 8	6916 7	Tidak <i>feasible</i> 9
13 Peringkat	3099 5	3576 6	5641 8	7317 9
14	1714	1650	2400	3543

(Lanjutan) Hasil Pemeringkatan Penelitian pada *Benchmark* ITC 2021 Kategori Late Bagian 2

<i>Dataset</i>	Phillips et al. (2021)	Dimitsas et al. (2022)	Subba and Stordal (2021)	Lester (2022)
Peringkat	6	5	8	9
15	80	80	6925	2030
Peringkat	4,5	4,5	9	8
Rata-Rata	5,96666667	5,96666667	6,96666667	8,26666667
Peringkat				

LAMPIRAN D

Bagian ini disertakan untuk menunjukkan efektivitas algoritma dengan menerapkannya pada permasalahan penjadwalan mata kuliah di Departemen Sistem Informasi, yang merupakan permasalahan yang familiar untuk dosen di Departemen Sistem Informasi. Pada penerapan ini, preferensi dosen menjadi satu-satunya faktor yang digunakan sebagai *soft constraint*. Gambar D.1 menampilkan preferensi dosen terhadap 20 slot waktu, yang dikategorikan sebagai berikut:

- -1: Dosen lebih menyukai untuk dijadwalkan pada slot waktu tersebut.
- 0: Dosen tidak memiliki keberatan khusus terhadap slot waktu tersebut.
- 1: Dosen masih dapat mengajar pada slot waktu tersebut, tetapi sebaiknya dihindari jika memungkinkan.
- 2: Dosen tidak dapat mengajar pada slot waktu tersebut (hard constraint).

Setelah algoritma diimplementasikan, pendekatan AT-ILS terbukti menghasilkan jadwal yang lebih baik dibandingkan penjadwalan manual. Tabel D.1 menyajikan perbandingan pemenuhan preferensi dosen antara solusi yang dioptimasi dan solusi manual. Dari tabel tersebut, terlihat bahwa jadwal hasil PL-ILS hanya menempatkan dosen pada dua slot waktu yang tidak disukai, sekaligus berhasil mengakomodasi 61 slot waktu yang memang disukai oleh dosen. Sebaliknya, penjadwalan manual memiliki 7 slot waktu yang tidak disukai dosen, serta hanya mengakomodasi 49 slot waktu yang disukai.

Meskipun demikian, solusi AT-ILS menempatkan satu dosen pada dua slot waktu yang seharusnya tidak memungkinkan karena dosen tersebut (berkode MH) sebenarnya tidak memiliki ketersediaan waktu. Hal ini terjadi karena dosen bersangkutan masih tercantum dalam dataset penjadwalan yang diuji.

Untuk lebih jelas, hasil penjadwalan menggunakan algoritma PL-ILS dapat dilihat pada

ID	IN	Nama	Hari	Senin				Selasa				Rabu				Kamis				Jumat			
				1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	AM	Ahmad Muklason, S.Kom., M.Sc., Ph.D.	Senin	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
2	AU	Amalia Utamina, S.Kom., M.B.A., Ph.D.	Senin	2	-1	0	2	0	-1	0	2	0	-1	0	2	1	-1	0	2	0	1	0	2
3	AP	Andre Pervian Aristio, S.Kom., M.Sc.	Senin	0	-1	0	1	-1	-1	-1	1	0	-1	-1	1	0	0	0	1	0	2	2	2
4	AN	Anisah Herdiyanti, S.Kom., M.Sc.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	AW	Arif Wibisono, S.Kom., M.Sc., Ph.D.	Senin	-1	-1	-1	-1	-1	-1	-1	1	1	-1	2	2	2	2	2	2	2	2	2	2
6	BS	Bambang Setiawan, S.Si., M.Kom.	Senin	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
7	AO	Dr. Aisyah Hidayah, S.T., M.T.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
8	BS	Dr. Bambang Setiawan, S.Kom., M.T.	Senin	2	1	0	1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	AT	Dr. Ir. Aris Tjahranto, M.Kom.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
10	MJ	Dr. Mudzihidin, S.T., M.T.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
11	RM	Dr. Rarasayama Indrawati, S.Kom.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
12	VR	Retro Aulia Vinarti, S.Kom., M.Kom., Ph.D.	Senin	2	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2
13	FS	Dr.Eng Febriyani Sampono, S.Kom., M.Kom.	Senin	2	2	2	2	-1	-1	-1	1	2	-1	2	2	2	2	2	2	0	0	2	2
14	ED	Edwin Rikasikomara, S.Kom., M.T.	Senin	-1	2	2	1	1	-1	2	1	1	-1	2	1	1	-1	2	1	2	2	2	
15	TE	Eko Wahyu Tyas D, S.Kom., M.B.A.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
16	FJ	Faizal Johan Atletiko, S.Kom., M.T.	Senin	2	-1	1	-1	2	0	0	0	1	0	0	0	0	0	0	0	1	2	2	2
17	FM	Faizal Mahananta, S.Kom., M.Eng., Ph.D.	Senin	2	-1	0	1	2	-1	0	1	2	-1	0	1	2	-1	0	1	2	-1	0	1
18	FA	Dr. Feby Arhawdinia Mufidroh, S.Kom., M.T.	Senin	2	2	1	2	2	1	0	-1	0	2	-1	-1	1	2	1	2	1	2	2	2
19	HM	Harini Maria Astuti, S.Kom., M.Sc.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
20	IK	Ika Nurkhasanah, S.Kom., M.Sc.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
21	AH	Ir. Ahmed Holl Noor Ali, M.Kom.	Senin	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
22	IH	Imasari Hafidz, S.Kom., M.Sc.	Senin	-1	-1	1	2	-1	-1	1	1	-1	2	-1	2	0	0	2	2	1	1	1	2
23	IZ	Izzat Aulia Akbar, S.Kom., M.Eng., Ph.D.	Senin	2	0	0	1	2	0	0	1	2	0	0	1	2	0	0	1	2	1	1	2
24	MW	Prof. Mahendrawathi ER, S.T., M.Sc., Ph.D.	Senin	0	-1	0	2	0	0	-1	0	2	0	0	-1	0	0	2	1	1	1	1	2
25	NF	Nisfu Asril Sani, S.Kom., M.Sc.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
26	NA	Prof. Nur Aini Rahmatia, S.Kom., M.Sc.Eng., Ph.D.	Senin	2	2	2	2	-1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
27	PT	Purnomo Triyono, S.T., M.T., Ph.D.	Senin	0	0	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
28	AD	Prof. Ir. Arief Djunaidi, M.Sc., Ph.D.	Senin	0	0	-1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
29	RP	Radityo Prasetyanto, W.S.Kom., M.Kom.	Senin	0	0	0	1	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	1
30	RT	Rara Tyasnurita, S.Kom., M.B.A., Ph.D.	Senin	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
31	RK	Renny Pradiani K, S.T., M.T.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
32	RN	Renny Nadifatin, S.Kom., MBA, Ph.D.	Senin	2	0	0	1	2	0	0	0	2	0	0	0	2	1	0	0	0	0	0	1
33	RH	Rully Agus Hendrawan, S.Kom., M.Eng.	Senin	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
34	SH	Dr. Sholih, S.T., M.T.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
35	TD	Tony Dwi Susanto, S.T., M.T., Ph.D.	Senin	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
36	WA	Dr. Wink Anggraeni, S.Si., M.Kom.	Senin	-1	-1	1	2	-1	-1	-1	1	2	-1	-1	1	2	0	0	0	2	2	2	2
37	RZ	Rizal Risanda, S.Kom., M.Kom.	Senin	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1
38	YG	Yogantara Setya Dhamawan, S.Kom., MBusProcessMgt.	Senin	0	-1	-1	1	0	-1	-1	1	0	-1	-1	1	1	1	2	2	1	2	2	2

Gambar D.1: Preferensi Slot Waktu

Gambar D.2, sedangkan hasil penjadwalan manual disajikan pada Gambar D.3.

Tabel D.1: Perbandingan Solusi dengan Algoritma PL-ILS dan Solusi Manual

Preferensi	PL-ILS	Manual
Suka (-1)	61	49
Biasa (0)	114	115
Tidak Suka (1)	2	7
Tidak Bisa (2)	2	8

Day	Timeslot	1101-1102	-	2104	2208	2209	2102/SCR	2102/DR	2103	4101	4102	4201	4202	STU 1 (APTER) 1305	STU 2 (PG) 1302	1103
Senin / Monday	1 07.30 - 10.00	PTEIC A SEM 1 F1	PTEIC IUP SEM 1 XX	PSTI B SEM 5 SH	OFB D SEM 1 AW	TBD ID SEM 3 RP	DPIIUP SEM 4 IH	ADD D SEM 3 RT	T-SDABB S2 SEM 3 ES	SE IUP SEM 3 YG	MLTIA SEM 5 BC	ASD A SEM 3 AM				
	2 10.15 - 12.45	SE B SEM 3 AP	PSTI D SEM 5 BS	OFB IUP SEM 1 AW	PAP1 IUP SEM 4 WA	SEA SEM 3 AD	FILSAFATA SEM 4 AU	IPO S2 SEM 2 VR	SE C SEM 2 MW	MRTIA SEM 3 RN	ASD SEM 3 AM	MLTIB SEM 5 TD	ADD IUP SEM 3 RK			
	3 13.30 - 16.00	ASD B SEM 3 RK	SKBMA SEM 6 ES				MDK S2 SEM 1 AO	DTIUP SEM 1 YG	MRTIC SEM 4 RN	LSD C SEM 1 AU	PD ID SEM 1 RZ	DPP B SEM 4 FJ	PN ID SEM 3 SH	METD.SANS A SEM 2 AW		
	4 16.00 - 18.30				OFB C SEM 1 AW				T-PS S2 SEM 2 ES							
Selasa / Tuesday	5 07.30 - 10.00	T-KXIS2 SEM 3 BS	GRAFA SEM 6 NA	EB ID SEM 1 AO	PAP1 A SEM 4 WA	RO C SEM 3 ED	ITI2 S2 SEM 1 FS	T-SC S2 SEM 3 AU	LSD A SEM 1 RM	ABD SEM 3 RZ	DRPD SEM 4 IH	MPTA SEM 3 AP		DTB SEM 1 AH		
	6 10.15 - 12.45	SE C SEM 3 AP	PAP2 A SEM 7 AM	SED SEM 3 MW	OFB A SEM 1 AW	PAP1 C SEM 4 WA	METPEN S2 SEM 2 AD	PBA A SEM 7 AT	ASD A SEM 3 RK	DLA SEM 4 FM	TEKBER C SEM 5 YG	OFB B SEM 1 MI	TIKTA SEM 7 BS	SC IUP SEM 7 AU	T-DDC S2 SEM 3 FS	
	7 13.30 - 16.00	MPTC SEM 3 FA	PPS2 SEM 1 AT				DTA SEM 1 AH		PSTI D SEM 1 SH	PSTIUP SEM 5 BS	DTD SEM 3 MW	KSID SEM 1 YG				
	8 16.00 - 18.30															
Rabu / Wednesday	9 07.30 - 10.00	MLTID SEM 5 BC	MPTD SEM 3 FA	PEB IUP SEM 1 MU	PSTI C SEM 5 SH	PAP1 B SEM 4 WA	MRTID SEM 4 HM	RO A SEM 3 ED	PSO A SEM 3 NF	KTD ID SEM 6 AH	ITI1 C SEM 1 FS	ADD B SEM 3 RT	LSD B SEM 1 RM	TEORI SI S3 SEM 1 TD		
	10 10.15 - 12.45	DTID SEM 1 YG	T-KG S2 SEM 3 NA	ITI1 B SEM 1 FS	TEKBER B SEM 5 IZ	MPT B SEM 3 AP	T-MPB S2 SEM 3 MW	ASD D SEM 3 AM	ABD C SEM 3 RZ	MRTIIUP SEM 4 HM	PAP1 A SEM 4 FM	ABD IUP SEM 3 IH	DPP C SEM 4 VR	COGNITIF A SEM 6 AO	METKIALUKANT S3 SEM 1 AO	
	11 13.30 - 16.00		CAPS A SEM 7 AP	ETIKA C SEM 4 NA	ASD IUP SEM 3 RK		SEA SEM 3 MW	ITI1 A SEM 1 NF	TEKBER A SEM 5 YG	PDD ID SEM 3 AH	DLIUP SEM 4 IZ	MRTI B SEM 4 RP	EB ID SEM 4 BC	SEI SEM 1 AO		
	12 16.00 - 18.30															
Kamis / Thursday	13 07.30 - 10.00	MPTIUP SEM 3 FA	PSTI A SEM 5 BS	RO B SEM 3 ED	MLTIIUP SEM 5 TD	PEB D SEM 1 MJ	ASD C SEM 1 AM	PWEB A SEM 4 FJ	PSO A SEM 6 NF	ETIKA B SEM 6 NA	PRO ID SEM 3 AH					
	14 10.15 - 12.45	SEIUP SEM 3 MW	PAP1IUP SEM 4 WA	PSTI B SEM 5 BS	TEKBER IUP SEM 5 AD	LSD IUP SEM 1 IZ	T-COCN S2 SEM 1 AU	LSD D SEM 1 VR	PEB B SEM 1 RM	ASD B SEM 3 AM						
	15 13.30 - 16.00	T-EGOV SEM 3 TD	PEMDASA SEM 2 AT	PAP1 C SEM 4 WA	T-PREDIC S2 SEM 2 AD	DLB SEM 4 RP	FORENSIC A SEM 6 BC	ABD A SEM 3 RZ	T-MIP SEM 3 AO	SE D SEM 3 MW	RO IUP SEM 3 RM		TEKBER D SEM 5 IZ			
	16 16.00 - 18.30															
Jumat / Friday	17 07.00 - 09.30	PTEIC B SEM 1 F2	KDT ID SEM 3 AH	PTEIC C SEM 1 F3	ITI1IUP SEM 1 NF	PSTI C SEM 5 SH	ITI1 D SEM 1 FS	MPO A SEM 7 AO	MHPP SEM 6 MJ	ASD C SEM 3 AM		SDI A SEM 6 RP	ABD B SEM 3 RZ			
	18 09.30 - 11.30	APID SEM 3 RN	DLC SEM 4 RP	DLD SEM 4 RP	ETIKA A SEM 4 NA	PSTIUP SEM 5 SH			PEB C SEM 1 MJ	ADD C SEM 3 RT			PRATAA SEM 7 VR			
	19 13.30 - 16.00	PRATAB SEM 7 VR	MOSI S2 SEM 1 TD	PEB A SEM 1 RN	MB ID SEM 3 AO	T-PD S2 SEM 3 AT	DTC SEM 1 RM	ASD IUP SEM 3 AH	MLTIC SEM 5 RK	PSTI A SEM 5 BC	ADD A SEM 3 SH		ADD A SEM 3 RT			
	20 16.00 - 18.30															

D.2: Hasil Jadwal dengan Algoritma AT-ILS

D.3: Hasil Jadwal dengan Pembentukan Manual

LAMPIRAN E

Peer Review Comments and Responses (Annals of Operations Research)

Comment 1:

My major concern about the paper are the many grammatical errors that exist across the paper. Some examples follow:

Page 1, line 36: has been successfully → has successfully

Page 1, line 40: Our contribution include the → Our contribution includes the

Page2, line 2: Timetabling and scheduling problem → Timetabling and scheduling problems

Page2, line 3: that attract interest many researchers → have attracted the interest of many researchers

...and the list goes on and on.

Response to Comment 1:

We have thoroughly reviewed and corrected all grammatical errors throughout the paper. Some examples include:

- **Abstract:**

- Page 1 line 20: has been successfully → has successfully
- Page 1 line 22: to solve ITC 2021 → to solve the ITC 2021
- Page 1 line 25: Our contribution include → Our contribution

includes

- **Section 1:**

- Page 2 line 2: Timetabling and scheduling problem → Timetabling and scheduling problems
- Page 2 line 2–3: optimisation problems that attract interest many researchers → optimization problems that have attracted the interest of many researchers
- Page 2 line 6: sport timetabling problem. → the sports timetabling problem.
- Page 2 line 16: sport timetabling → sports timetabling
- Page 2 line 27–28: stand out as the sole researchers → stands out as the sole researcher
- Page 2 line 29 and 32: solution → solutions
- Page 2 line 39: pertubation → perturbation
- Page 2 line 40: to search feasible solutions → to search for feasible solutions
- Page 2 line 41–42: the probability to find feasible solutions. → the probability of finding feasible solutions.

- **Section 2:**

- Page 3 line 4: sport timetabling problem → sport timetabling problems.
- Page 3 line 12: Bench marking → benchmarking
- Page 3 line 20: as follow → as follows

- **Section 3:**

- Page 4 line 18: Generating feasible solution → Generating feasible solutions

- Page 4 line 19–20: there is only one prior work reported → there has been only one prior work reported
- Page 4 line 22: optimise → optimize
- Page 5 line 6: as worse solution → as a worse solution
- Page 5 line 8: in which it start with empty solution → which begins with an empty solution
- Page 7 line 2: conflicts should be considered → conflicts that should be considered.
- Page 7 line 6: related to time slots constraint → related to the time slot constraints
- Page 7 line 16: causes hard constraint → causes a hard constraint
- Page 7 line 20–21: limits only one team can play away → limits only one team to play away
- Page 7 line 23: the before and after condition → the before and after conditions
- Page 7 line 28: The steps within perturbation → The steps within the perturbation
- Page 7 line 34: the time slot chosen randomly → the time slot is chosen randomly
- Page 8 line 1 and 3: randomisation → randomization
- Page 8 line 10: in local search phase is shuffling process. → in the local search phase is the shuffling process.
- Page 8 line 11–12: while maintaining hard constraints satisfaction. → while maintaining satisfaction with the hard constraints.
- Page 9 line 5: neighbourhoods → neighborhoods

- Page 10 line 13: how Partial Round Swap operator works. → how the Partial Round Swap operator works.

- **Section 4:**

- Page 10 line 23: two folds → twofold
- Page 11 line 8: parameter tuning → a parameter tuning
- Page 11 line 9–10: five parameters need to be tuned → five parameters that need to be tuned
- Page 11 line 10: to get the best performance of the algorithm. → to achieve the best algorithm performance.
- Page 12 line 3: Due to resource limitation → Due to resource limitations
- Page 12 line 7: displays run time statistics → displays runtime statistics
- Page 12 line 10: generate feasible → generate a feasible
- Page 12 line 21: there is significant differences → there are significant differences
- Page 13 line 3: feasible solution for all 45 → feasible solutions for all 45
- Page 13 line 6: the different approach used → the different approaches used
- Page 14 line 3: without mathematical → without a mathematical

- **Section 5:**

- Page 16 line 2–3: to generate initial feasible solution → to generate feasible solutions
- Page 16 line 6: Through comprehensive → Through a

comprehensive

- Page 16 line 13: generating feasible solution → generating feasible solutions

Comment 2:

According to the authors each problem instance was run for 10 times. For middle 2 in only 1 out of the 10 runs the algorithm managed to produce a feasible solution under the time limit of 120 hours in 113 hours. I would suggest repeating the experiment to validate that the 10% success rate is repeated consistently.

Response to Comment 2:

We have rerun the experiment for the *middle 2* dataset an additional 10 times and obtained consistent results, with only one run producing a feasible solution within the 120-hour time limit. In this repetition, the feasible solution was found after 60 hours. This result has been added to the manuscript on page 12, lines 21 to 27, and Table 6 and Figure 4 have been updated accordingly. Additionally, we have included a note below Table 6 to clarify that the *middle 2* instance was run 10 additional times, unlike the other instances.

Comment 3:

Please improve the quality of the algorithm presentations. For example:

- Algorithms 4 & 5: `Suffle` → `Shuffle` (multiple occurrences here and elsewhere)
- Algorithm 5, line 3: `for i from 0` → `for i ← 0 to`

Response to Comment 3:

We have improved the algorithm presentations as suggested. The typo `suffle` has been corrected to `shuffle` throughout the manuscript.

Regarding the comment on “Algorithm 5, line 3: `for i from 0` → `for i ← 0 to`,” we believe there was a slight confusion, as our paper does not include an Algorithm 5. The correction you mentioned was found in *Algorithm 4*, and we have updated the line

accordingly from “`for i from 0 → limitShuffle { 1}`” to “`for i ← 0 to limitShuffle { 1}`.”

Comment 4:

Please, associate the swap moves used in sections 3.2.1 and 3.2.2 to moves identified in section 3.5 of Ribeiro, Celso C., Sebastián Urrutia, and Dominique de Werra. *Combinatorial Models for Scheduling Sports Tournaments*. Berlin: Springer, 2023. See moves “Home-Away Swap” (HAS) and “Partial Round Swap” (PRS).

Response to Comment 4:

The swap moves discussed in sections 3.2.1 and 3.2.2 are indeed conceptually similar to the *Home-Away Swap* (HAS) and *Partial Round Swap* (PRS) as described in *Combinatorial Models for Scheduling Sports Tournaments*. To avoid ambiguity, we have renamed the operators in sections 3.2.1 and 3.2.2 to align with the terminology used in the book. Additionally, we have cited the work by Ribeiro, Celso C., Sebastián Urrutia, and Dominique de Werra on page 9, lines 1–2, to properly reference these operators.

Comment 5:

Figure 6: The claim that there is a correlation between the “Ratio of hard Constraints / Game” and the “Average Running time” seems to be invalidated by problem instance “Late 1”. It seems that the hardness that each constraint contribute to the problem is not equal. Also consider that some constraints could have been written equivalently with multiple other constraints, which invalidates the assumption that the total number of constraints is crucial to the problem hardness.

Response to Comment 5:

Upon further review, we agree that the statement regarding the correlation between the “Ratio of hard Constraints / Game” and the “Average Running time” is invalidated by the “Late 1” instance, which has a similar ratio of hard constraints per game but shows significantly faster average times to find a feasible solution. We conducted additional analysis to investigate potential correlations involving the number and types of hard

constraints, the total number of constraints, and the number of games. However, we were unable to find any strong correlations to explain why instances such as “Middle 2”, “Middle 3”, and “Late 2” are harder and require longer times to find feasible solutions. As a result, we have removed Figure 6 and the corresponding paragraph that described it from the manuscript.

Comment 6:

Use consistent naming: e.g., slot vs. timeslot.

Response to Comment 6:

We have reviewed similar papers and found that “time slot” is the more suitable term for discussing timetabling problems. As a result, we have updated all occurrences of “slot” to “time slot” in our manuscript. Additionally, we have standardized the usage by replacing “timeslot” with “time slot” for consistency.

We also identified other inconsistencies in naming, such as “double round-robin” versus “2-round robin.” We have chosen to use “double round-robin” throughout the manuscript, as it is the most accurate term.

Finally, we addressed the inconsistency between “metaheuristic” and “meta-heuristic,” and have standardized the term to “metaheuristic.”

Comment 7:

The approach is a metaheuristic (ILS) and not a heuristic, please revise.

Response to Comment 7:

We have revised the terminology throughout the manuscript to correctly reflect that the approach is a metaheuristic rather than a heuristic. Specifically, changes have been made to the title, keywords, the abstract (page 1, lines 23–24), the introduction section (page 2, line 37), and the conclusion (page 16, line 14).

Comment 8:

Page 6: The initial value of `decreasingValue` is not specified.

Response to Comment 8:

The initial value of `decreasingValue` is determined through a parameter tuning process, which should have been discussed in Section 4.2, Table 4. However, instead of `decreasingValue`, we mistakenly listed `initialProbability` with a value of 0.7. The correct value for `decreasingValue` is 0.3, which should have been reflected in Table 4.

Additionally, `initialProbability` should not have appeared in the manuscript, as its value is directly derived from Equation 1: `probabilityRandomTimeSlot = 1 - decreasingValue`. In this case, `probabilityRandomTimeSlot` is calculated as 0.7 based on the correct `decreasingValue` of 0.3.

To address this issue, we have made the following updates to the manuscript:

- **Table 4:** We have replaced `initialProbability` (with a value of 0.7) by `decreasingValue` (with a value of 0.3) as determined by the parameter tuning process.
- **Algorithm 1:** We corrected the mistake by replacing`probabilityRandomTimeSlot ← initialProbability` with`probabilityRandomTimeSlot ← calculateProbability(decreasingValue, constantFactor)`.
- **Section 3, Page 6, Lines 6–17:** We have revised the paragraph to better explain the calculation process involving Equations 1 and 2.

Comment 9:

Page 6: At the text the variable `decreasingValue` is defined, but in Algorithm 1 the variable name used is `decreaseRate`.

Response to Comment 9:

We have identified the mistake where `decreaseRate` was incorrectly used instead of `constantFactor`. These variables refer to the same concept. We have updated the manuscript to use the consistent variable name `constantFactor` throughout the text and in Algorithm 1 to avoid any confusion.

Comment 10:

Page 7 line 13: “This condition causes hard constraint violation that permits only one team per slot.” → “This condition causes hard constraint violation that permits exactly one occurrence of each team per time slot.”

Response to Comment 10:

We have made the modification as suggested. The sentence on page 7, lines 16–17, now states:

“This condition causes hard constraint violation that permits exactly one occurrence of each team per slot.”

Comment 11:

Page 7 line 18: Team 5.. → Team 5.

Response to Comment 11:

We have corrected the typo by removing the extra dot at the end of the sentence at page 7 line 21.

Comment 12:

Figure 1: Add Slot 1 at the top of the figures such as: “Slot 1: Before”

Response to Comment 12:

We have updated Figure 1 by adding “Slot 1” at the top of the time slot table illustration, as suggested.

Comment 13:

Fig.1 (b) Remove Second Conflic → (b) Remove Second Conflict

Response to Comment 13:

We have corrected the typo in the caption of Figure 1(b) by changing “Second Conflic” to “Second Conflict.”

Comment 14:

Fig. 1 Tim → Team

Response to Comment 14:

We have corrected the typo in the caption of Figure 1 by changing “Tim” to “Team.”

Comment 15:

Page 8 line 37: Further explanation is needed for: “no games can be randomized successfully.”

Response to Comment 15:

This sentence refers to the exit condition in the local search phase. The phrase “no games can be randomized successfully” indicates the second exit condition, which occurs when the shuffling process fails to make any changes to the solution.

To clarify this, we have rewritten the explanation of the exit conditions in the local search phase on page 8, lines 4–9.

Comment 16:

Table 4: limitStuck vs. limitStuckLocalSearch in Algorithm 1

Response to Comment 16:

We identified this as a mistake, as both terms refer to the same variable. We have corrected this by consistently using the variable name `limitStuck` throughout the manuscript.

Comment 17:

Page 12 line 12: report the time needed for getting solutions for those 8 datasets.

Response to Comment 17:

We have added the runtime required, which ranges between 86 and 1691 minutes, on page 12, lines 17–18.

Comment 18:

Check the references for formatting errors, e.g., references ending with . or ...

Response to Comment 18:

We have reviewed the references and corrected formatting issues, such as references ending with , , or Additionally, we referred to the Springer L^AT_EX template guidelines. According to these guidelines, articles with a DOI should not end with a period or comma, while articles without a DOI should end with a comma. Other references, such as books or websites, should end with a period. We have updated all references accordingly, including adding DOIs where applicable.

Peer Review Comments and Responses (IEEE Access)

Reviewer #1, Concern #1

Reviewer's Comment: "The authors have introduced a complete review of a complex Timetabling Optimization problem considering different evolutionary algorithms taxonomy. The only limitation is to respect the multiobjective Timetabling Problem. It sounds exciting to introduce the main constraints when the problem is defined as multiobjective. The authors could also provide the MOEAs applied to Multiobjective Timetabling."

Author Response: We agree that including multiobjective timetabling in our survey would enhance its comprehensiveness. However, we encountered limitations that make it impractical to incorporate the multiobjective Timetabling Problem into our paper. Our survey aims to identify approaches and algorithms with superior performance in timetabling problems, motivated by the varying claims of superiority in existing studies that lack consistent comparison standards. To address this gap, we focused on popular benchmarks used in timetabling and surveyed papers utilizing these benchmarks to create a global ranking and analyze the results.

In examining papers that investigate multiobjective timetabling problems, we found that those meeting our criteria (published between 2012 and 2022 in Q2 journals) used case studies or popular benchmarks with modified objective functions. For example:

- **Study by Stefan Binder et al.**, titled "A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem," utilized the Dutch railway network as a case study, which is not used in other studies.
- **Study by Yinghui Wu et al.**, titled "Multi-objective re-synchronizing of bus timetable: Model, complexity and solution," used a case study provided by a transit company in Shenyang, China, which is also unique to this study.
- **Study by Yu Lei**, titled "A memetic algorithm based on MOEA/D for the examination timetabling problem," used the popular Carter Benchmark, a single-objective timetabling problem. However, this study added another objective function to minimize the number of periods, making it incomparable with studies using the Carter Benchmark as a single-objective problem. Additionally, we did not

find other studies using the same benchmark and objective function, preventing us from performing a ranking analysis.

These conditions prevent us from including multiobjective timetabling problems in our survey paper, as the objective of our paper does not align with the studies on multiobjective timetabling problems.

Author Action: Due to the reasons described in the “author response,” we cannot make any modifications for this concern.

Reviewer #2, Concern #1

Reviewer’s Comment: “The introduction section should clearly highlight the gap in the current literature. While existing survey papers cover a wide range of algorithms and techniques, they often lack a comprehensive comparison and critical discussion of non-genetic algorithms. Additionally, there is a need for updated surveys that incorporate the latest advancements and applications in the field.”

Author Response: The gap we aim to fill with our survey paper has two main points. First, we seek to identify the most promising approach and algorithm in the field of timetabling by conducting a comprehensive ranking based on the results of the papers we surveyed. Second, we aim to evaluate the effectiveness of *generic algorithms*—defined as algorithms that can solve various timetabling problems without the need for modifications, which is different from “genetic algorithms.”

Author Action: To clearly highlight the gap in our survey paper, we have revised the introduction section. First, we added a discussion of existing survey papers in the second paragraph. Next, we rewrote the subsequent two paragraphs to explicitly address the gaps in the existing literature. The third paragraph discusses the first gap, emphasizing the lack of comprehensive rankings in existing papers to determine the most superior and promising algorithms. The fourth paragraph addresses the gap concerning generic algorithms, which have not been covered in previous survey papers.

Reviewer #2, Concern #2

Reviewer's Comment: “Clearly state the unique contributions of your paper in the introduction. This could include a detailed comparison of non-genetic algorithms, an in-depth critical analysis of these techniques, and the provision of updated and high-quality figures. Additionally, emphasize any novel insights or future research directions that your paper offers.”

Author Response: First, we acknowledge that our previous introduction did not clearly describe our contributions in this survey paper. Second, there seems to be a misunderstanding: our paper focuses on discussing *generic algorithms*, not genetic algorithms, as explained in our response to Concern #1. Third, novel insights or future research directions have been mentioned and discussed in Section VI and are briefly included in the abstract and conclusion. We have also modified the future research directions in Section VI to address Concern #6. However, we believe it is uncommon to include novel insights or future research directions in the introduction section.

Author Action: To clarify the unique contributions of our paper, we have rewritten the fifth paragraph of the introduction. We explicitly state that our contribution is to address the two gaps discussed in the third and fourth paragraphs: conducting a comprehensive ranking of the surveyed papers and investigating the actual performance of generic algorithms in the timetabling field.

Reviewer #2, Concern #3

Reviewer's Comment: “Apart from the genetic algorithm, other algorithms and techniques should be critically discussed and categorized. This includes, but is not limited to, simulated annealing, particle swarm optimization, ant colony optimization, and differential evolution. Each of these techniques has its strengths and weaknesses, and a detailed analysis will provide a more comprehensive understanding of their applicability in various scenarios.”

Author Response: Our survey paper does not focus exclusively on genetic algorithms; instead, we survey all papers that use heuristic methods to solve the timetabling problem from 2012 to 2022. As mentioned in previous responses, there seems to be a misunderstanding between the terms “generic algorithm” and “genetic algorithm.” Our

survey focuses on “generic algorithms.” Therefore, we have indeed compared algorithms beyond genetic algorithms, including simulated annealing, great deluge, artificial bee colony, and others. We have detailed some approaches that provide good results and others that yield poor results, with an analysis provided in Section 5.

Author Action: Since our paper already discusses algorithms besides genetic algorithms, no additional action is required for this concern.

Reviewer #2, Concern #4

Reviewer’s Comment: “The quality of the figures in the paper is poor and needs significant improvement. High-quality, clear, and well-labeled figures are essential for effective communication of complex ideas. Consider using high-resolution images and ensuring that all figures are readable and well-organized.”

Author Response: We agree that some figures, especially those depicting the Distribution of Average Ranking in each benchmark with scatter plots, are poor and difficult to read.

Author Action: We have recreated the scatter plot figures (Figures 4–9) to be clearer and more readable, ensuring high-quality images for better understanding and communication.

Reviewer #2, Concern #5

Reviewer’s Comment: “Section IV requires more critical discussion and summarization. This section should not only present the findings but also critically analyze them, providing insights into the implications and potential limitations. Summarizing key points and comparing them to existing literature will strengthen the section.”

Author Response: We agree that our previous version primarily presented descriptions of the findings and lacked deep and critical analysis of the results.

Author Action: We have updated and rewritten the subsections for the Carter, Socha, ITC 2007 (Track 1, 2, and 3), and ITC 2011 benchmarks in Section IV to include more structured and in-depth analysis. The updates follow a structured sequence for each benchmark:

1. Description of the Benchmark: We describe each benchmark along with its constraints.
2. Surveyed Papers: We present the papers surveyed for each benchmark.
3. Descriptive Findings: We explain the best, average, and worst performances.
4. Deep Analysis: We provide a deeper analysis of these findings and their correlation with results from other benchmarks, focusing on the approaches used.
5. Algorithm Analysis: We analyze the algorithms' performance.

Additionally, we have summarized all findings from Section IV in the “VI. DISCUSSION AND FURTHER RESEARCH” section, which has also been updated and rewritten to provide a comprehensive overview and highlight future research directions.

Reviewer #2, Concern #6

Reviewer’s Comment: “The future directions section should be extended to provide a more detailed and comprehensive outlook on potential research avenues. This could include exploring new algorithms, hybrid approaches, real-world applications, and addressing any gaps identified in the current study. Providing a clear roadmap for future research will significantly enhance the value of your paper.”

Author Response: We agree with the suggestion to extend the future directions in Section 6.C.

Author Action: We have updated the manuscript by rewriting and extending Section 6.C. In this section, we suggest future research directions based on our findings. These include exploring single-solution-based algorithms, developing more strategies using existing successful strategies for creating generic algorithms, and conducting further research on the latest benchmarks. This comprehensive outlook aims to provide a clear roadmap for future research.

BIOGRAFI PENULIS



Penulis menamatkan pendidikan SMA pada tahun 2015, kemudian melanjutkan studi Sarjana (S1) di Institut Teknologi Sepuluh Nopember (ITS) dengan jurusan Sistem Informasi. Pada tugas akhir S1, penulis melakukan penelitian tentang optimasi kombinatorika dengan topik Traveling Salesman Problem. Pada tahun 2019, penulis meneruskan pendidikan ke jenjang Magister (S2) di program studi yang sama. Fokus penelitian yang dilakukan adalah pengembangan algoritma dalam menyelesaikan permasalahan penjadwalan. Selanjutnya, pada tahun 2021, penulis memulai pendidikan Doktoral di jurusan dan universitas yang sama, dengan penelitian yang berfokus pada pengembangan algoritma generik dalam menyelesaikan permasalahan penjadwalan. Untuk pertanyaan, kritik, dan saran, penulis dapat dihubungi melalui email berikut: igustiagungpremananda@gmail.com.