Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA *LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC* MENGGUNAKAN *BENCHMARK DATASET* DARI RUMAH SAKIT DI NORWEGIA

NURSE ROSTERING OPTIMIZATION WITH LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC ALGORITHM USING NORWEGIAN HOSPITALS BENCHMARK DATASET

RIZAL RISNANDA HUTAMA
NRP. 05211640000024

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IS184853

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA *LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC* MENGGUNAKAN *BENCHMARK DATASET* DARI RUMAH SAKIT DI NORWEGIA

RIZAL RISNANDA HUTAMA
NRP. 05211640000024

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS – IS184853

NURSE ROSTERING OPTIMIZATION WITH LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC ALGORITHM USING NORWEGIAN HOSPITALS BENCHMARK DATASET

RIZAL RISNANDA HUTAMA
NRP. 05211640000024

Supervisor
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTMENT OF INFORMATION SYSTEM
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC MENGGUNAKAN BENCHMARK DATASET DARI RUMAH SAKIT DI NORWEGIA

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

Rizal Risnanda Hutama
05211640000024

Surabaya, 14 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Mudjahidin, ST., MT.
NIP. 197010102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA *LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC* MENGGUNAKAN *BENCHMARK DATASET* DARI RUMAH SAKIT DI NORWEGIA

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :

RIZAL RISNANDA HUTAMA
NRP. 05211640000024

Disetujui Tim Penguji : Tanggal Ujian : 26 Juni 2020
Periode Wisuda : September 2020

Ahmad Muklason, S.Kom., M.Sc., Ph.D. **(Pembimbing I)**

Edwin Riksakomara, S.Kom., M.T. **(Penguji I)**

Faizal Mahananto S.Kom., M.Eng., Ph.D. **(Penguji II)**

Halaman ini sengaja dikosongkan

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA *LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC* MENGGUNAKAN *BENCHMARK DATASET* DARI RUMAH SAKIT DI NORWEGIA

Nama : Rizal Risnanda Hutama
NRP : 05211640000024
Departemen : Sistem Informasi FTEIC ITS
Pembimbing I : Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRAK

Penjadwalan perawat yang terdapat dalam sebuah rumah sakit merupakan sebuah permasalahan yang hingga saat ini masih sering diselesaikan dengan cara manual. Penjadwalan perawat dengan cara manual akan memakan waktu yang cukup lama selaras dengan jumlah perawat yang dibutuhkan. Selain itu, untuk melakukan penjadwalan perawat harus mempertimbangkan banyak kepentingan yaitu misalnya kepentingan perawat, kepentingan pemilik rumah sakit, dan kepentingan pasien. Penjadwalan perawat dengan cara manual akan menimbulkan permasalahan. Dari segi perawat, penjadwalan manual seringkali mengakibatkan pembagian beban kerja yang tidak merata, sehingga perawat yang memiliki beban kerja lebih banyak daripada perawat yang lain akan merasa kecewa dengan jadwal tersebut. Dari segi pemilik rumah sakit, penjadwalan manual tidak mempertimbangkan efektifitas perawat. Hal ini menyebabkan perawat yang dibutuhkan semakin banyak dan akan mengakibatkan pada meningkatnya biaya yang dialokasikan untuk upah perawat. Dari segi pasien, penjadwalan manual terkadang mengalami kesusahan untuk mengalokasikan perawat yang memiliki kompetensi tertentu yang dibutuhkan oleh pasien. Oleh karena itu, diperlukan optimasi penjadwalan perawat untuk

menghasilkan jadwal yang optimal dalam memenuhi kebutuhan semua kepentingan yang terlibat dan mempersingkat waktu yang dialokasikan untuk melakukan penjadwalan perawat. Dalam dunia komputasi, permasalahan penjadwalan perawat merupakan masalah yang termasuk dalam golongan NP-Hard Problem. Artinya, hingga saat ini belum ada algoritma eksak yang dapat menyelesaikannya karena banyaknya kemungkinan yang terjadi dan adanya batasan yang harus dipenuhi. Oleh karena itu, algoritma heuristic dibutuhkan untuk menyelesaikan permasalahan penjadwalan perawat. Begitu pula dengan penjadwalan perawat yang terdapat pada rumah sakit di Norwegia. Penjadwalan perawat pada rumah sakit di Norwegia juga memiliki batasan-batasan yang tidak boleh dilanggar sehingga sulit untuk diselesaikan dengan cara manual. Batasan yang terdapat pada rumah sakit di Norwegia ada dua yaitu, hard constraint dan soft constraint. Tujuan dari optimasi penjadwalan perawat pada rumah sakit di Norwegia ini yaitu untuk meminimalkan pelanggaran yang terjadi pada soft constraint. Dalam tugas akhir ini, optimasi benchmark dataset rumah sakit di Norwegia diselesaikan dengan menggunakan algoritma Late Acceptance Hill Climbing dengan dibandingkan dengan Hill Climbing. Hasil dari tugas akhir ini yaitu sebanyak 3 dari 7 instance berhasil feasible. Pada seluruh instance yang feasible, optimasi menggunakan algoritma Late Acceptance Hill Climbing mampu menghasilkan solusi yang lebih optimal daripada Hill Climbing dengan selisih penurunan penalti antara 2% hingga 7%. Hill Climbing hanya mampu menurunkan penalti sebesar 78% pada OpTur4, 74% pada OpTur5, dan 5% pada OpTur7. Sedangkan Late Acceptance Hill Climbing mampu menurunkan penalti sebesar 80% pada OpTur4, 81% pada OpTur5 dan 7% pada OpTur7 jika dibandingkan dengan solusi awal.

Kata Kunci : Optimasi, Permasalahan Penjadawalan Perawat, Late Acceptance Hill Climbing, Hyper-Heuristic

NURSE ROSTERING OPTIMIZATION WITH LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC ALGORITHM USING NORWEGIAN HOSPITALS BENCHMARK DATASET

Name	: Rizal Risnanda Hutama
NRP	: 05211640000024
Department	: Information System FTEIC ITS
Supervisor I	: Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRACT

Nurse Rostering in a hospital is a problem that still often solved manually. Manually rostering takes a lot time that goes in line with the number of nurses needed. In addition, it must consider a lot of interests i.e. the interests of the nurse, the interests of the owner of the hospital, and the patient's need. Manually rostering will cause many problem. From the nurse's side, manual rostering does not consider the nurse effectiveness. This cause more nurses needed and will result in increased costs allocated for nurse's wages. From the patient's side, manual rostering sometimes have trouble to allocate the nurse who have the specific competencies needed by the patient. Therefore, optimization of nurse rostering is needed in order to create an optimal roster that meet the needs of all the interests involved, and to shorten the time allocated to rostering. In computing, nurse rostering problem is an issue that belongs to the NP-Hard Prolem. It means that until now there is not exact algorithm that can solve it, because of many possibilities and the constraints that must be met. Therefore, heuristic algorithms are needed to solve this problem. Nurse rostering at hospitals in Norway also has constraints that should not be violated, so it is difficult to resolve manually. The constraints

are hard and soft constraint. The goal of optimizing of nurse rostering at hospitals in Norway is to minimize violations that may occur in soft constraint. In this final project, optimization of Norwegian hospital benchmark dataset is completed using the Late Acceptance Hill Climbing algorithm compared to Hill Climbing. The result of this final project are 3 out of 7 instace are feasible. For the all instances that are feasible, optimization using the Late Acceptance Hill Climbing algorithm is able to obtain more optimal solution than Hill Climbing with the difference in penalty reduction between 2% to 7%. Hill Climbing is only able to reduce penalties by 78% on OpTur4, 74% on OpTur5, and 5% on OpTur7. Whereas Late Acceptance Hill Climbing is able to reduce penalties by 80% in OpTur4, 81% in OpTur5 and 7% in OpTur7 when compared to the initial solution.

Keywords : Optimization, Nurse Rostering Problem, Late Acceptance Hill Climbing, Hyper-Heurustic

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Rizal Risnanda Hutama
NRP	: 05211640000024
Tempat/Tanggal lahir	: Surabaya, 13 April 1998
Fakultas/Departemen	: FTEIC / Departemen Sistem Informasi
Nomor Telp/Hp/email	: rizal16@mhs.is.its.ac.id

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul

OPTIMASI PENJADWALAN PERAWAT DENGAN ALGORITMA *LATE ACCEPTANCE HILL CLIMBING HYPER-HEURISTIC* MENGGUNAKAN *BENCHMARK DATASET* DARI RUMAH SAKIT DI NORWEGIA

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 16 Juni 2020

RIZAL RISNANDA H.
NRP. 05211640000024

Halaman ini sengaja dikosongkan.

KATA PENGANTAR

Alhamdulillahirobbil'alamin, segala puja dan puji syukur kehadirat Allah SWT yang telah melimpahkan segala rahmat dan berkah sehingga penulis dapat menyelesaikan tugas akhir dengan judul Optimasi Penjadwalan Perawat Dengan Algoritma *Late Acceptance Hill Climbing Hyper-Heuristic* Menggunakan *Benchmark Dataset* Dari Rumah Sakit Di Norwegia sebagai salah satu syarat kelulusan Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Ucapan terimakasih sebesar-besarnya tak lupa penulis sampaikan kepada seluruh pihak yang turut mendukung, memberikan semangat dan doa dalam pengerjaan tugas akhir ini:

1. Kedua orangtua dan kedua saudara penulis yang senantiasa memotivasi dan memberikan semangat sampai kapan pun.
2. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. sebagai dosen pembimbing penulis dalam menyelesaikan tugas akhir ini yang telah sabar dan meluangkan waktu dalam mengarahkan penulis.
3. Bapak Edwin Riksakomara, S.Kom., M.T. dan Bapak Faizal Mahananto, S.Kom., M.Eng., Ph.D. sebagai dosen penguji I dan penguji II yang telah memberikan kritik dan saran yang membangun agar tugas akhir ini menjadi lebih baik.
4. Dimas Rizal Kusuma Sindhu, Shafira Widya Hapsari dan Sisca Threecya Agatha selaku teman seperjuangan dalam topik NRP yang selalu menyemangati penulis.
5. Teman-teman HH yang telah membantu penulis dalam penyediaan tempat pengerjaan tugas akhir.
6. Teman-teman laboraturium RDIB yang senantiasa memberikan hiburan penulis.

7. Teman-teman SI angkatan 2016 yang selalu mendukung penulis dalam mengerjakan tugas akhir ini.
8. Teman-teman lain yang tidak kalah penting dalam mendukung, memberi semangat, hiburan, dan doa kepada penulis.
9. Dan seluruh pihak lain yang terlibat.

Penyusunan tugas akhir ini tidak luput dari kekurangan dan kesalahan. Oleh karena itu, penulis menerima saran dan kritik yang membangun dari semua pihak sebagai perbaikan untuk kedepannya.

Surabaya, 16 Juni 2020

Penulis

Rizal Risnanda Hutama

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Bab ini menjelaskan latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir ini dilakukan, dan juga relevansi terhadap pengerjaan tugas akhir ini. Dari penjelasan bab ini diharapkan dapat memberikan gambaran secara umum mengenai permasalahan dan penyelesaian masalah yang dilakukan dalam tugas akhir ini.

1.1. Latar Belakang

Dalam dunia kesehatan, khususnya pada sebuah rumah sakit, penjadwalan perawat merupakan suatu permasalahan yang sangat kompleks untuk diselesaikan [1]. Hal tersebut dikarenakan penjadwalan perawat yang ada pada sebuah rumah sakit banyak hal yang harus diperhatikan. Jumlah perawat yang dibutuhkan pada setiap harinya, maksimal beban kerja untuk setiap perawat, serta kualifikasi seorang perawat yang dibutuhkan untuk setiap harinya merupakan beberapa contoh hal yang harus diperhatikan ketika melakukan penjadwalan perawat di dalam sebuah rumah sakit. Selain itu, permasalahan hari masuk dan hari libur untuk tiap perawat juga harus dipertimbangkan. Perawat terkadang menginginkan hari masuk dan hari libur yang merata dan adil jika dibandingkan dengan perawat-perawat yang lain.

Hingga saat ini, kebanyakan penjadwalan perawat masih dilakukan secara manual yang dilakukan oleh seorang staf khusus yang memiliki kemampuan untuk menjadwalkan perawat [2]. Melakukan penjadwalan secara manual membutuhkan waktu yang cukup banyak. Oleh karena itu, untuk mempersingkat waktu dalam penjadwalan, maka dibutuhkan sebuah mekanisme penjadwalan secara otomatis. Selain untuk mempersingkat waktu, penjadwalan otomatis juga akan mengurangi beban kerja personel yang membuat penjadwalan. Dengan berkurangnya beban kerja personel

tersebut, maka beban kerja personel tersebut dapat dialokasikan untuk pekerjaan yang lain.

Dalam dunia komputasi, penjadwalan perawat secara otomatis hingga saat ini masih tergolong kedalam *NP-hard Problem* atau diartikan hingga saat ini masih belum ada algoritma eksak untuk menyelesaikan permasalahan tersebut [3]. Hanya pendekatan *heuristic* atau aproksimasi yang digunakan untuk menyelesaikan *NP-hard Problem*. Pendekatan *heuristic* tersebut tidak dapat menjamin bahwa solusi yang dihasilkan merupakan solusi yang optimal. Akan tetapi, solusi yang dihasilkan cukup baik dan dapat dikatakan mendekati optimal. Optimal atau tidaknya sebuah solusi akan diukur dari fungsi tujuannya (*objective function*). Dalam membangun sebuah solusi untuk mencapai *objective function* akan dihadapkan dengan batasan-batasan yang ada. Batasan tersebut dibagi menjadi dua yaitu, *hard constraints* dan *soft constraints*. *Hard constraints* merupakan batasan yang tidak boleh dilanggar, sedangkan *soft constraints* boleh dilanggar akan tetapi memiliki bobot tersendiri. Sebuah *objective function* biasanya mengukur banyaknya *soft constraint* yang dilanggar. Sebuah solusi akan semakin baik atau optimal apabila pelanggaran *soft constraints* semakin sedikit. Dengan kata lain, *objective function* sebuah permasalahan penjadwalan biasanya untuk meminimalkan pelanggaran *soft constraints*. Untuk mengoptimalkan sebuah solusi maka diperlukan algoritma untuk optimasi.

Sebagai salah satu contoh, rumah sakit di Norwegia memiliki sejumlah perawat dengan masing-masing spesialisasinya. Antara satu hari dengan hari yang lain, kebutuhan akan spesialisasi dan jumlah perawat berbeda-beda. Dalam satu hari terdapat tiga *shift*, dimana pada setiap *shift* juga membutuhkan jumlah dan spesialisasi perawat yang berbeda. Pada rumah sakit di Norwegia juga memiliki batasan-batasan *hard constraints* maupun *soft constraints* layaknya permasalahan penjadwalan yang lain. *Hard constraint* yang dimiliki yaitu sebanyak tujuh *constraints* dan *soft constraints* yang dimiliki sebanyak

sembilan *constraints*. Contoh dari *hard constraints* yang dimiliki yaitu terkait dengan maksimal *shift* pada tiap hari untuk satu perawat, maksimal jam kerja perawat dalam satu minggu, dan lain-lain. Contoh dari *soft constraints* yang dimiliki yaitu terkait dengan maksimal hari kerja berturut-turut untuk satu perawat. Dalam kasus rumah sakit di Norwegia ini memiliki *objective function* yang sama yaitu untuk meminimalkan pelanggaran *soft constraint* yang terjadi. Jadi, metode untuk menyelesaikan permasalahan penjadwalan pada rumah sakit di Norwegia ini dapat dikatakan lebih baik dari metode yang lain apabila, pelanggaran *soft constraint* pada jadwal yang dihasilkan lebih rendah daripada metode yang lain. Salah satu metode yang pernah digunakan untuk menyelesaikan kasus ini yaitu dengan mengkombinasi *Constraint Programming* (CP) dengan *Iterated Local Search* (ILS) dan *Variable Neighborhood Descent* (VND) [2].

Dalam tugas akhir ini, penulis menggunakan metode *Late Acceptance Hill Climbing* (LAHC) *hyper-heuristic* untuk menyelesaikan permasalahan penjadwalan perawat yang ada pada rumah sakit di Norwegia. *Hyper-heuristic* dipilih oleh penulis karena dalam *Hyper-Heuristic* akan mencoba untuk memilih *heuristic* yang tepat untuk menyelesaikan permasalahan [4]. LAHC merupakan algoritma yang digunakan untuk optimasi. LAHC merupakan sebuah metode pencarian *local search*. Metode LAHC merupakan pengembangan dari metode *hill climbing* yang sudah ada sebelumnya. Perbedaan antara LAHC dan *hill climbing* biasa yaitu pada titik penerimaan kandidat solusinya. Jika pada *hill climbing* biasa, kandidat solusi akan diterima sebagai solusi baru apabila kandidat solusi tersebut lebih baik dari solusi sebelumnya, sedangkan jika pada LAHC, kandidat solusi akan diterima sebagai solusi baru apabila solusi tersebut lebih baik dari beberapa iterasi sebelumnya. Oleh karena itu, LAHC dapat mencakup kandidat solusi yang lebih luas daripada *hill climbing* biasa. Metode pencarian yang juga sering digunakan selain LAHC yaitu *Simulated Annealing* (SA), *Threshold Accepting*

(TA) dan *the Great Deluge Algorithm* (GDA). Metode LAHC dipilih oleh penulis karena metode ini cukup mudah untuk diimplementasikan, *simple* dan cukup efektif untuk menyelesaikan masalah pencarian penjadwalan. Jika dibandingkan dengan metode pencarian tersebut, LAHC juga memiliki keunggulan tersendiri yaitu tidak adanya *cooling schedule* untuk penerapannya [5].

1.2. Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, maka rumusan masalah yang menjadi fokusan pada tugas akhir ini yaitu;

1. Bagaimana penerapan algoritma *Late Acceptance Hill Climbing Hyper-Heuristic* untuk menyelesaikan penjadwalan perawat dengan *benchmark dataset* rumah sakit di Norwegia?
2. Bagaimana performa algoritma *Late Acceptance Hill Climbing Hyper-Heuristic* dibandingkan dengan algoritma lainnya untuk menyelesaikan penjadwalan perawat dengan *benchmark dataset* rumah sakit di Norwegia?

1.3. Batasan Masalah

Berdasarkan permasalahan yang telah dijelaskan sebelumnya, tugas akhir ini memiliki batasan-batasan sebagai berikut;

1. Tugas akhir ini hanya berfokus pada *benchmark dataset* rumah sakit di Norwegia yang didapat dari web Sintef .
2. Tugas akhir ini mencakup 7 *instance* rumah sakit yang disediakan.
3. Penerapan algoritma dibangun menggunakan bahasa pemrograman Java.

1.4. Tujuan Tugas Akhir

Tugas akhir ini dilakukan untuk mencapai beberapa tujuan sebagai berikut;

1. Menghasilkan jadwal yang optimal untuk penjadwalan perawat dengan *benchmark dataset* rumah sakit di

Norwegia dengan menggunakan algoritma *Late Acceptance Hill Climbing Hyper-Heuristic*.

2. Mengetahui perbandingan performa algoritma *Late Acceptance Hill Climbing* dengan algoritma lainnnya.

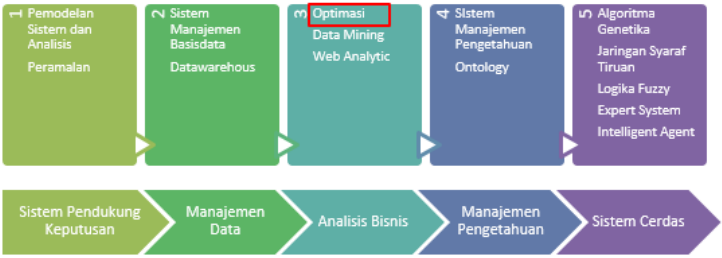
1.5. Manfaat Tugas Akhir

Tugas akhir ini diharapkan dapat memberikan manfaat diantaranya:

1. Bagi Akademis
   Manfaat tugas akhir ini bagi akademis yaitu dapat menambah wawasan terkait penerapan algoritma *Late Acceptance Hill Climbing Hyper-Heuristic* terhadap optimasi permasalahan penjadwalan perawat.
2. Bagi Instansi Kesehatan
   Manfaat tugas akhir ini bagi instansi kesehatan yaitu dapat menambah wawasan terkait dengan melakukan penjadwalan staf otomatis yang menghasilkan jadwal yang optimal dengan berbagai batasan yang ada.
3. Bagi Penelitian Selanjutnya
   Manfaat tugas akhir ini bagi penelitian selanjutnya yaitu dapat menjadi referensi/rujukan dalam mengembangkan penelitian terhadap optimasi penjadwalan perawat.

1.6. Relevansi

Topik tugas akhir ini yaitu optimasi penjadwalan perawat rumah sakit. Bidang optimasi merupakan bidang yang terkait dengan mata kuliah Optimasi Kombinatorik Heuristik (OKH) yang berada dibawah naungan laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB) Departemen Sistem Informasi ITS seperti yang tercantum pada gambar 1.1. sebagai berikut :



Gambar 1.1 Road Map Penelitian Laboratorium RDIB

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penelitian yang sudah dilakukan sebelumnya serta dasar teori yang memiliki keterikatan dengan pengerjaan tugas akhir ini.

2.1. Studi Sebelumnya

Dalam pengerjaan tugas akhir ini, penulis tak lupa menjadikan beberapa penelitian sebelumnya yang terkait sebagai acuan untuk menambah wawasan teori penulis. Beberapa penelitian sebelumnya yang dijadikan acuan pengerjaan tugas akhir ini akan disajikan dalam Tabel 2.1 sebagai berikut;

Tabel 2.1 Penelitian Sebelumnya

Penelitian Pertama	
Judul Penelitian	*A hybrid approach for solving real-world nurse rostering problems* [2].
Penulis; Tahun	Martin Stølevik, Tomas Eric Nordlander, Atle Riise, Helle Frøyseth; 2011
Deskripsi Umum	Penelitian ini berisi tentang optimasi penjadwalan perawat otomatis pada rumah sakit di Norwegia dengan menggunakan penggabungan pendekatan *Constraint Programming* (CP) dan *Variable Neighborhood Descent* (VND) dalam sebuah kerangka kerja *Iterated Local Search* (ILS). Hasil yang didapatkan yaitu, *objective function* yang diperoleh jauh lebih baik jika dibandingkan dengan solusi awal.
Keterkaitan Penelitian	Penelitian ini akan dijadikan referensi untuk mengetahui bagaimana menyelesaikan penjadwalan perawat dan menjelaskan dataset yang digunakan dalam tugas akhir ini.
Penelitian Kedua	
Judul Penelitian	*The late acceptance Hill-Climbing heuristic* [5].
Penulis; Tahun	Edmund K. Burke, Yuri Bykov; 2017

Deskripsi Umum	Penelitian ini berisi tentang pengenalan terhadap metodologi *Late Acceptance Hill Climbing* (LAHC). Selain itu, dalam penelitian ini juga membandingkan LAHC dengan metodologi lain yaitu, *Simulated Annealing* (SA), *Threshold Accepting* (TA) dan the *Great Deluge Algorithm* (GDA). Hasil dari penelitian ini yaitu, metode LAHC ini merupakan metode yang sederhana, mudah untuk diimplementasikan, dan cukup efektif. Bahkan, jika dibandingkan dengan metode pesaingnya, LAHC memiliki rata-rata kinerja yang lebih baik khususnya pada *instance* yang memiliki ukuran cukup besar
Keterkaitan Penelitian	Penelitian ini akan dijadikan sebagai referensi untuk mengimplementasikan algoritma *Late Acceptance Hill Climbing* dalam menyelesaikan permasalahan penjadwalan.
Penelitian Ketiga	
Judul Penelitian	*Automated Examination Timetabling Optimization Using Greedy-Late Acceptance-Hyperheuristic Algorithm* [6].
Penulis; Tahun	Ahmad Muklason, Putri C. Bwananesia, Sasmi Hidayatul Y. T., Nisa D. Angresti, Vicha Azthanty Supoyo; 2018
Deskripsi Umum	Penelitian ini berisi tentang optimasi penjadwalan ujian secara otomatis menggunakan algoritma *hybrid* yaitu *Greedy-Late Acceptance* dalam kerangka kerja *hyper-heuristic*. Hasilnya yaitu dengan algoritma tersebut mampu menghasilkan solusi jadwal yang *feasible*. Jika dilihat dari nilai *proximity cost*, algoritma yang digunakan dapat menghasilkan nilai yang lebih baik. Dengan menggunakan algoritma tersebut juga hanya diperlukan waktu lima menit untuk running. Waktu tersebut sangatlah singkat jika dibandingkan dengan waktu penjadwalan manual yang membutuhkan waktu selama satu hingga dua minggu.
Keterkaitan Penelitian	Penelitian ini akan dijadikan sebagai referensi untuk menggunakan pendekatan *Hyper Heuristic*

	untuk menyelesaikan masalah optimasi penjadwalan.
Penelitian Keempat	
Judul Penelitian	Optimasi penjadwalan staf rumah sakit dengan menggunakan metode *Late Acceptance Hill-Climbing Hyper-Heuristic* (studi kasus: RSIA Kendangsari Surabaya) [7].
Penulis; Tahun	Rizka Pordella; 2018
Deskripsi Umum	Penelitian ini berisi tentang optimasi penjadwalan staf Rumah Sakit Ibu dan Anak di Kendangsari Surabaya dengan menggunakan metode *Late Acceptance Hill-Climbing* (LAHC). Di dalam penelitian ini juga melakukan penjadwalan dengan melakukan generate otomatis sesuai dengan pola bulan sebelumnya. Dari kedua metodologi tersebut akan dihitung keadilan jumlah libur untuk masing-masing staf dengan menggunakan *Jain Fairness Index* (JFI). Hasil dari penelitian LAHC dominan memiliki JFI yang lebih tinggi atau dapat dikatakan lebih bagus dari generate otomatis maupun cara manual.
Keterkaitan Penelitian	Penelitian ini akan dijadikan sebagai referensi untuk mengoptimasi masalah penjadwalan perawat dengan metode LAHC.

2.2. Dasar Teori

Bagian ini berisi teori-teori yang berkaitan dan mendukung dalam pengerjaan tugas akhir.

2.2.1. Optimasi

Optimasi merupakan pendekatan umum untuk melakukan komputasi atau perhitungan dalam memaksimalkan atau meminimalkan sebuah poin tujuan [8]. Optimasi dapat diartikan sebagai cara untuk mendapatkan suatu nilai yang terbaik sesuai dengan poin tujuannya. Poin tujuan atau yang lebih kerap

disebut dengan fungsi tujuan dapat berupa maksimasi ataupun minimasi yang memiliki cara berbeda untuk menyelesaikannya.

Dahulu, optimasi bukan menjadi cabang penting dari matematika terapan, sehingga para ahli matematika tidak memberikan perhatian lebih terhadap permasalahan otomasi. Oleh karena itu, pada tahun-tahun yang lalu, bidang optimasi minim akan publikasi dan optimasi meninggalkan misteri dalam sejarahnya [8]. Akan tetapi, pada saat ini permasalahan optimasi berkembang dengan cepat seiring dengan semakin kompleksnya permasalahan. Optimasi saat ini dapat dilakukan dengan menggunakan bantuan *software* komputer dengan berbagai algoritma yang bermacam-macam. Hal ini disebabkan oleh ketidakmungkinan untuk diselesaikan dengan cara manual.

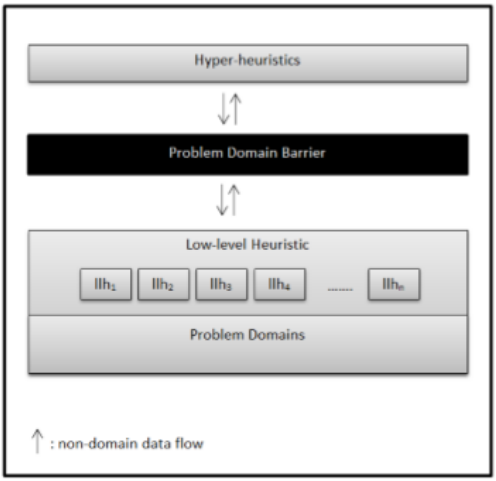
2.2.2. Permasalahan Penjadwalan Perawat

Permasalahan penjadwalan merupakan sebuah permasalahan penjadwalan yang kompleks. Hal ini disebabkan karena penjadwalan perawat digunakan untuk menghasilkan sebuah solusi yang dapat mengefisienkan waktu dan beban kerja untuk setiap perawat agar setiap perawat merasa puas dengan pembagian jadwal tersebut. Selain itu, batasan yang ada dalam penjadwalan perawat juga sangatlah bermacam-macam. Dalam pembuatan solusi penjadwalan perawat ini memerlukan waktu yang sangat banyak karena tingkat kompleksitas permasalahannya yang tinggi.

Penjadwalan perawat pada intinya yaitu bertujuan untuk memutuskan perawat mana yang harus bertugas agar sesuai dengan kualifikasi perawat seperti apa yang dibutuhkan oleh rumah sakit pada saat itu. Permasalahan ini tergolong dalam jenis permasalahan *NP-hard* [3]. Hal ini dapat diartikan, hingga pada saat ini belum ada algoritma eksak yang dapat menyelesaikan penjadwalan perawat secara optimal, melainkan hanya dengan menggunakan pendekatan *heuristic*.

2.2.3. *Hyper-heuristic*

Hyper-Heuristic merupakan beberapa pendekatan yang dikumpulkan dengan tujuan untuk memilih dan melakukan kombinasi terhadap heuristic yang lebih sederhana (*Heuristic Selection*) atau membuat *heuristic* baru menggunakan komponen yang sudah ada sebelumnya (*Heuristic Generation*) yang nantinya digunakan untuk menyelesaikan permasalahan komputasi yang susah jika diselesaikan dengan cara manual [9].



Gambar 2.1 Framework Hyper-Heuristic
(Sumber: Muklason, 2017)

Pada gambar 2.1. dapat dikatakan bahwa *Hyper-Heuristic* tidak secara langsung bersinggungan dengan problem domain akan tetapi *Hyper-Heurisitc* akan bersinggungan dengan *Low-Level Heuristic* dan mengetahui hasil dari *objective function* yang dihasilkan. Artinya *Hyper-Heuristic* akan memilih atau membuat *Heuristic* yang lebih rendah dan cocok untuk memecahkan permasalahan yang ada [10].

2.2.4. *Dataset* Rumah Sakit di Norwegia

Dataset Rumah Sakit di Norwegia diunduh dari web Sintef. Unduhan dari web Sintef berisikan 7 *instance* yang berupa *excel*

yang diberi nama “OpTur1” hingga “OpTur7”. *Instance* satu dengan *instance* yang lainnya memiliki jumlah perawat yang berbeda. Jumlah batasan (*hard* dan *soft constraint*) yang dimiliki tiap *instance* sama, akan tetapi terdapat beberapa nilai batasan yang berbeda tiap *instance*. Tabel 2.2 menampilkan *hard constraints* dan *soft constraints* pada *dataset* rumah sakit di Norwegia.

Tabel 2.2 Hard Constraints dan Soft Constraints

Hard Constraints
1. Maksimal 1 *shift* dalam 1 hari untuk tiap perawat
2. Jumlah kebutuhan perawat tiap hari harus terpenuhi
3. Jumlah jam kerja tiap perawat tidak melebihi batas toleransi kontrak kerja
4. Perawat hanya dapat bekerja sesuai dengan kompetensi yang dimiliki
5. Jam jeda antar *shift* pada hari yang berurutan untuk seorang perawat harus melebihi batas minimal
6. Dalam setiap minggu harus memiliki jam libur kerja secara kontinu melebihi batas minimal
7. Jam kerja dalam seminggu tidak boleh melebihi batas minimal
Soft Constraints
1. Tidak melanggar batas maksimal hari kerja secara berurutan dengan kategori *shift* yang sama
2. Tidak melanggar batas maksimal hari kerja secara berurutan
3. Tidak melanggar batas minimal hari kerja secara berurutan dengan kategori *shift* yang sama
4. Tidak melanggar batas minimal hari kerja secara berurutan
5. Tidak melanggar batas maksimal dan minimal jumlah hari kerja pada tiap kategori *shift*
6. Peyimpangan terhadap kontrak kerja untuk tiap perawat
7. Terpisahnya (tidak berurutan) hari libur perawat
8. Jumlah pola *shift* yang diinginkan pada solusi yang dihasilkan harus ditingkatkan
9. Jumlah pola *shift* yang tidak diinginkan pada solusi yang dihasilkan harus dikurangi

2.2.5. Model Matematika Penjadwalan Perawat

Model matematika untuk penjadwalan perawat memiliki dua batasan yaitu *hard* dan *soft constraint*. *Hard constraint* merupakan batasan yang tidak boleh dilanggar. Sedangkan *soft constraint* merupakan batasan yang boleh dilanggar, tetapi harus diminimalkan. Sebuah solusi dapat dikatakan *feasible* apabila memenuhi semua persyaratan *hard constraint*. Kualitas dari solusi yang dihasilkan dapat diukur dari *objective function* yang didapat dari perhitungan pelanggaran *soft constraint*. Semakin sedikit pelanggaran *soft constraint*, maka semakin baik pula solusi tersebut.

Model matematika penjadwalan perawat untuk *benchmark dataset* rumah sakit di Norwegia adalah sebagai berikut [11];

a. Variabel :
   - S = Himpunan *shift.*
   - s = Sebuah *shift* anggota himpunan *shift.*
   - C = Himpunan kategori *shift*.
   - c = Sebuah kategori *shift* anggota himpunan kategori *shift*.
   - D = Himpunan hari.
   - d = Sebuah hari anggota himpunan hari dan dimulai pada hari ke 1.
   - E = Himpunan perawat.
   - e = Seorang perawat anggota himpunan perawat.
   - I_d = Himpunan pasangan *shift* yang tidak cocok pada hari d dan $d + 1$ karena jarak waktu antar *shift* terlalu pendek.
   - U = Himpunan pola *shift* yang tidak dikehendaki.
   - u = Sebuah pola *shift* yang tidak dikenedaki anggota himpunan pola *shift* yang tidak dikehendaki.
   - V = Himpunan pola *shift* yang dikehendaki.
   - v = Sebuah pola *shift* yang dikenedaki anggota himpunan pola *shift* yang dikehendaki.
   - W = Himpunan mingguan.
   - w = Sebuah minggu anggota himpunan mingguan. Dalam tiap minggu dimulai hari ke 1 (senin) dan diakhiri hari ke 7 minggu).

P_m = Penalti pelanggaran *soft constraint* m. Dimana m bernilai 1 hingga 9.
x = Sebuah solusi jadwal kerja perawat.

b. Parameter
N_c^{min} = Batas nilai minimal untuk kategori *shift* c yang berurutan.
N_c^{max} = Batas nilai maksimal untuk kategori *shift* c yang berurutan.
N^{min} = Jumlah minimal *shift* yang berurutan untuk semua kategori *shift* dan semua perawat.
N^{max} = Jumlah maksimal *shift* yang berurutan untuk semua kategori *shift* dan semua perawat.
N_{ec}^{max} = Jumlah maksimal kategori *shift* c yang harus dijalankan oleh perawat e.
N_{ec}^{min} = Jumlah minimal kategori *shift* c yang harus dijalankan oleh perawat e.
R_{ds} = Jumlah tipe *shift* s yang dibutuhkan dalam hari d.
T_e = Waktu maksimal perawat e bekerja dalam seminggu.
T_e^h = Waktu kerja yang seharusnya dikerjakan perawat e sesuai dengan kontrak kerja sepanjang periode penjadwalan.
T^f = Waktu minimal dalam tiap minggu untuk tidak bekerja (*off* atau libur).
T_s = Durasi dari *shift* s.
A_{sc} = Bernilai 1 jika *shift* s masuk dalam kategori *shift* c dan bernilai 0 jika sebaliknya.
B_{es} = Bernilai 1 jika perawat e memiliki kompetensi sesuai dengan *shift* s, dan bernilai 0 jika sebaliknya.

c. *Decision Variable* :
q_{eds} = Bernilai 1 jika perawat e dijadwalkan bekerja pada *shift* s pada hari d dan bernilai 0 jika sebaliknya.

d. *Derived Information* :

$f_{ew}(x)$ = Dalam solusi x, periode tidak bekerja terpanjang untuk perawat e pada minggu w.

$u_{ei}(x)$ = Dalam solusi x, jumlah pola *shift* tipe i yang tidak dikehendaki untuk perawat e.

$v_{ei}(x)$ = Dalam solusi x, jumlah pola *shift* tipe i yang dikehendaki untuk perawat e.

e. *Hard Constraint* :

Persamaan 2.1 menjelaskan bahwa tiap perawat hanya dapat bekerja 1 *shift* untuk tiap harinya.

$$\sum_{s \epsilon S} q_{eds} \leq 1 , \qquad \forall\, e \epsilon E, d \epsilon D \tag{2.1}$$

Persamaan 2.2 menjelaskan bahwa kebutuhan jumlah tiap tipe *shift* harus terpenuhi untuk tiap harinya.

$$\sum_{e \epsilon E} q_{eds} = R_{ds} , \qquad \forall\, d \epsilon D, s \epsilon S \tag{2.2}$$

Persamaan 2.3 menjelaskan bahwa jumlah jam kerja tiap perawat hanya boleh menyimpang dari kontrak kerja sebesar $\pm$ 2%.

$$(1 - 0.02)T_e^h \leq \sum_{d \epsilon D, s \epsilon S} T_s q_{eds} \leq (1 + 0.02)T_e^h , \qquad \forall\, e \epsilon E \tag{2.3}$$

Persamaan 2.4 memastikan jika perawat hanya bekerja pada *shift* dengan kebutuhan kompetensi yang dimiliki perawat tersebut.

$$\sum_{s \epsilon S} B_{es} q_{eds} = 1, \qquad \forall\, e \epsilon E, d \epsilon D \tag{2.4}$$

Persamaan 2.5 menjelaskan tidak boleh adanya *shift* tertentu yang berurutan untuk tiap perawat.

$$q_{eds} + q_{e(d+1)s'} \leq 1, \qquad \forall\, e \epsilon E, d \epsilon D, (s, s') \epsilon I_d \tag{2.5}$$

Persamaan 2.6 digunakan untuk memastikan dalam tiap minggu memiliki periode bebas kerja secara kontinu untuk tiap perawat yang nilainya berbeda-beda tiap *instance*.

$$f_{ew}(x) > T^f, \ \forall e \in E, w \in W \qquad (2.6)$$

Persamaan 2.7 digunakan untuk memastikan maksimal jam kerja dalam seminggu tidak dilanggar untuk tiap perawat yang nilainya berbeda-beda tiap *instance*.

$$\sum_{d=7w+1}^{7w+7} \sum_{s \in S} T_x q_{eds} \leq T_e, \qquad \forall e \in E, w \in W \qquad (2.7)$$

f. *Soft Constraint* :

Persamaan 2.8 digunakan untuk menghitung penalti 1 yang disebabkan oleh terlalu banyak hari kerja yang berurutan dengan kategori *shift* yang sama. Jumlah maksimalnya berbeda-beda tiap *instance*.

$$p_1(x) = \sum_{c \in C} \sum_{e \in E} \sum_{d=1}^{|D|-N_c^{max}} \left(\prod_{d'=d}^{d+N_c^{max}} y_{ecd'} \right) \qquad (2.8)$$

Persamaan 2.9 digunakan untuk menghitung penalti 2 yang disebabkan oleh terlalu banyak hari kerja yang berurutan. Jumlah maksimalnya berbeda-beda tiap *instance*.

$$p_2(x) = \sum_{e \in E} \sum_{d=1}^{|D|-N^{max}} \left(\prod_{d'=d}^{d+N^{max}} z_{ed'} \right) \qquad (2.9)$$

Persamaan 2.10 digunakan untuk menghitung penalti 3 yang disebabkan oleh terlalu sedikit hari kerja yang berurutan dengan kategori *shift* yang sama. Jumlah minimalnya berbeda-beda tiap *instance*.

$$p_3(x) = \sum_{c \epsilon C} \sum_{e \epsilon E} \sum_{d=1}^{|D|-N_c^{min}} max(0, y_{ec(d+1)} - y_{ecd}) \left(N_c^{min} - \sum_{d'=d+1}^{d+N_c^{min}} \left(\prod_{d'=d+1}^{d+N_c^{min}} y_{ecd''} \right) \right) \quad (2.10)$$

Persamaan 2.11 digunakan untuk menghitung penalti 4 yang disebabkan oleh terlalu sedikit hari kerja yang berurutan.

$$p_4(x) = \sum_{e \epsilon E} \sum_{d=1}^{|D|-N^{min}} max\left(0, z_{e(d+1)} - z_{ed}\right) \left(N^{min} - \sum_{d'=d+1}^{d+N^{min}} \left(\prod_{d'=d+1}^{d+N^{min}} z_{ed''} \right) \right) \quad (2.11)$$

Persamaan 2.12 digunakan untuk menghitung penalti 5 yang disebabkan oleh penyimpangan yang terlalu banyak dari jumlah minimal dan maksimal untuk tiap kategori *shift*.

$$p_5(x) = \sqrt{\sum_{c \epsilon C} \sum_{e \epsilon E} \left(\max \left(0, N_{ec}^{min} - \sum_{d \epsilon D} y_{ecd}, \sum_{d \epsilon D} y_{ecd} - N_{ec}^{max} \right) \right)^2} \quad (2.12)$$

Persamaan 2.13 digunakan untuk menghitung penalti 6 yang disebabkan oleh penyimpangan jumlah jam kerja perawat jika dibanding dengan kontrak kerja perawat.

$$p_6(x) = \sqrt{\sum_{e \epsilon E} \left(T_e^h - \sum_{d \epsilon D} \sum_{s \epsilon S} T_s q_{eds} \right)^2} \quad (2.13)$$

Persamaan 2.14 digunakan untuk menghitung penalti 7 yang disebabkan oleh terpisahnya (tidak berurutan) hari libur perawat.

$$p_7(x) = \sqrt{\sum_{e \epsilon E} \left(\sum_{d=1}^{|D|-1} \max\left(0, z_{ed} - z_{e(d+1)}\right) \right)^2} \tag{2.14}$$

Persamaan 2.15 digunakan untuk menghitung penalti 8 yaitu memaksimalkan jumlah pola *shift* yang diinginkan.

$$p_8(x) = |E||D| - \sum_{e \epsilon E, i \epsilon V} v_{ei}(x) \tag{2.15}$$

Persamaan 2.16 digunakan untuk menghitung penalti 9 yaitu meminimalkan jumlah pola *shift* yang tidak diinginkan.

$$p_9(x) = \sum_{e \epsilon E, i \epsilon U} u_{ei}(x) \tag{2.16}$$

Dimana,

$y_{ecd} = \sum_{s \epsilon S} A_{sc} q_{eds}$ adalah bernilai 1 ketika perawat e bekerja dalam kategori *shift* c dalam hari d dan bernilai 0 jika sebaliknya.

$z_{ed} = \sum_{s \epsilon S} q_{eds}$ adalah bernilai 1 ketika perawat e bekerja pada hari d dan benilai 0 jika sebaliknya.

g. *Objective Function* :

$$Minimize: f = \sum_{m=1}^{9} K_m p_m \tag{2.17}$$

Persamaan 2.17 menjelaskan mengenai perhitungan *objective function*. K_m merupakan konstanta tetap sebagai bobot penalti. Dan P_m merupakan nilai penalti yang didapat dari perhitungan pelanggaran *soft constraint*.

2.2.6. *Hill Climbing*

Hill Climbing merupakan salah satu contoh dari *local search*. Cara kerja *hill climbing* yaitu mencari solusi yang lebih baik pada setiap solusi *neighbor*. Pencarian solusi dengan menggunakan *hill climbing* ini akan berhenti ketika tidak ditemukannya kandidat solusi pada *neighbor* yang lebih baik [12]. Kelemahan dari *Hill Climbing* yaitu seringkali terjebak dalam *local optima*.

2.2.7. *Late Acceptance Hill Climbing* (LAHC)

*Late Acceptance Hill Climbi*ng (LAHC) merupakan pengembangan dari *hill climbing* yang dibedakan oleh penerimaan solusinya. Pada LAHC, kandidat solusi akan dibandingkan dengan solusi beberapa iterasi sebelumnya [5]. Penyimpanan solusi akan dilakukan pada *history length* yang panjangnya dapat diatur sesuai keinginan.

LAHC merupakan salah satu *locah search*. Hal yang membedakan LAHC dengan yang lain yaitu tidak ada mekanisme khusus seperti *cooling time* pada SA dan *threshold* pada TA. Tidak adanya mekanisme khusus ini, LAHC dapat dikatakan menjadi algoritma yang efektif dan dapat diandalkan jika dibanding dengan kedua *local search* tersebut [5].

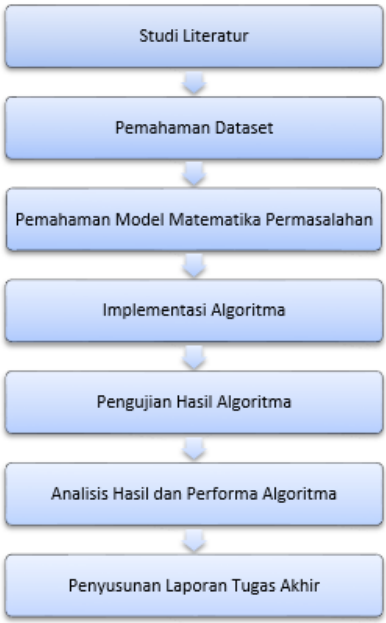
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Bab metodologi ini menjelaskan seluruh tahapan yang dilakukan oleh penulis dalam mengerjakan tugas akhir ini yang disertai dengan deskripsi untuk setiap langkahnya.

3.1. Tahapan Pelaksanaan Tugas Akhir

Bagian ini menjelaskan alur dari pengerjaan tugas akhir yang dilakukan oleh penulis. Alur pengerjaannya dapat dilihat pada gambar 3.1 sebagai berikut :



Gambar 3.1 Alur Pengerjaan Tugas Akhir

3.2. Uraian Metodologi

Bagian ini menjelaskan secara lebih rinci mengenai alur pelaksanaan tugas akhir untuk setiap proses yang sesuai dengan Gambar 3.1.

3.2.1. Studi Literatur

Pengumpulan berbagai macam jurnal atau penelitian sebelumnya yang memiliki topik pembahasan terkait akan dilakukan dalam bagian ini. Berbagai macam penelitian tersebut akan digunakan sebagai rujukan oleh penulis untuk mengerjakan tugas akhir ini. Tujuan dilakukannya tahap studi literatur ini yaitu menambah wawasan terkait dasar teori yang akan digunakan oleh penulis untuk mengerjakan tugas akhir.

3.2.2. Pemahaman *Dataset*

Pemahaman *dataset* bertujuan untuk mengetahui isi dari *dataset* tersebut. Pemahaman mengenai *constraint* yang ada merupakan hal yang utama dalam pemahaman dataset ini. Ketika tahapan ini sudah dilakukan, maka diharapkan agar bisa lebih mudah dalam mengimplementasikan algoritma untuk dataset ini.

3.2.3. Pemahaman Model Matematika Permasalahan

Model matematika untuk penjadwalan perawat pada *benchmark dataset* rumah sakit di Norwegia sudah tersedia pada penelitian yang ada sebelumnya. Oleh karena itu, pemahaman terkait model matematika permasalahan perlu dilakukan. Pemahaman model matematika yang dilakukan meliputi *decision variable*, *constraints* (*hard* dan *soft constraint*) dan *objective function*. Tujuan dari pemahaman model matematika adalah untuk memudahkan pengimplementasian algoritma yang digunakan untuk menyelesaikan permasalahan.

3.2.4. Implementasi Algoritma

Implementasi algoritma akan dibagi menjadi dua bagian yaitu pembuatan solusi awal dan optimasi solusi awal.

3.2.4.1. Pembuatan Solusi Awal

Pembuatan solusi awal dilakukan dengan cara menentukan *shift* secara urut maupun acak. Solusi awal yang dihasilkan harus *feasible* atau dapat diartikan tidak boleh melanggar *hard constraint* yang ada. Pembuatan solusi awal ini tidak mempertimbangkan *soft constraint* yang ada.

3.2.4.2. Optimasi Solusi Awal

Setelah terbentuk solusi awal, maka akan dilakukan optimasi terhadap solusi tersebut untuk menghasilkan *objective function* seminimal mungkin. Optimasi dilakukan dengan menggunakan algoritma LAHC. Algoritma LAHC akan mencari solusi hingga mencapai kondisi tertentu. Kondisi untuk memberhentikan proses LAHC dinamakan *stopping criteria*. Implementasi LAHC dalam tugas akhir ini memiliki dua *stopping criteria* yaitu ketika sudah mencapai jumlah iterasi tertentu dan ketika tidak ada solusi baru dalam jumlah iterasi tertentu.

3.2.5. Pengujian Hasil Algoritma

Solusi jadwal yang sudah dihasilkan dan dioptimasi akan diuji. Solusi jadwal tersebut akan dicek apakah sudah memenuhi *hard constraint* yang ditentukan. Jika sudah memenuhi *hard constraint*, maka solusi jadwal tersebut dapat diterapkan. Akan tetapi, jika solusi belum memenuhi *hard constraint*, maka solusi tersebut perlu dilakukan peninjauan kembali. Peninjauan akan dilakukan pada algoritma yang digunakan agar hasil yang dihasilkan dapat memenuhi persyaratan *hard constraint*.

3.2.6. Analisis Hasil dan Performa Algoritma

Analisis hasil dilakukan terhadap solusi optimal yang dihasilkan dan sudah memenuhi persyaratan *hard constraint*. Analisis dilakukan dengan menghitung pelanggaran *soft constraint* yang terjadi sesuai dengan *objective function* yang sudah ditentukan. Nilai hasil analisis terhadap solusi tersebut merupakan nilai performa algoritma yang digunakan. Nilai performa algoritma yang didapat akan dibandingkan dengan nilai performa algoritma *benchmark*. Dari hasil perbandingan tersebut dapat diketahui nilai performa algoritma yang digunakan lebih baik atau lebih buruk dari algoritma *benchmark*.

3.2.7. Penyusunan Laporan Tugas Akhir

Pada tahap ini, dilakukan penyusunan dokumentasi tugas akhir ini yang dituangkan kedalam bentuk sebuah laporan. Laporan

yang dihasilkan nantinya dapat digunakan oleh orang lain sebagai acuan penelitian dengan topik terkait. Laporan disusun sesuai dengan ketentuan Departemen Sistem Informasi ITS dan Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB). Konten dari laporan tugas akhir ini terdiri dari;

1. BAB I PENDAHULUAN
   Bagian ini berisi faktor yang mendorong dan melatarbelakangi tugas akhir ini perlu dilakukan. Selain itu, bagian ini juga berisi manfaat dilakukannya tugas akhir ini.
2. BAB II TINJAUAN PUSTAKA
   Bagian ini berisi acuan-acuan yang digunakan untuk mendukung tugas akhir ini.
3. BAB III METODOLOGI
   Bagian ini berisi tahapan-tahapan yang dilakukan dalam mengerjakan tugas akhir ini..
4. BAB IV PERANCANGAN
   Bagian ini berisi rancangan metode yang akan diterapkan dalam pengerjaan tugas akhir ini.
5. BAB V IMPLEMENTASI
   Bagian ini berisi tahapan pengimplementasian metode terhadap data yang digunakan.
6. BAB VI HASIL DAN PEMBAHASAN
   Bagian ini berisi hasil dari implementasi metode terhadap data digunakan dan analisis terhadap hasil tersebut.
7. BAB VII KESIMPULAN DAN SARAN
   Bagian ini berisi simpulan dari analisis hasil untuk menjawab permasalahan yang ada serta saran untuk penelitian selanjutnya agar lebih baik dan lebih berkembang.

3.3. Kesimpulan Metodologi

Pengerjaan tugas akhir ini dilakukan menggunakan alur yang telah dijelaskan pada bab ini, dimulai dari studi literatur hingga penyusunan laporan tugas akhir. Sebelum tugas akhir ini diimplementasikan, akan disusun rancangan yang dijelaskan pada bab 4 perancangan.

BAB IV
PERANCANGAN

Bab ini akan menjelaskan mengenai rancangan dalam melakukan optimasi pada penjadwalan perawat yang mengacu pada bab 3 metodologi.

4.1. Pemahaman *Dataset*

Dataset yang digunakan dalam tugas akhir ini diperoleh dari web Sintef dan berjumlah tujuh *instance* yang diberi nama OpTur (OpTur1 – OpTur7). Setiap *instance* memiliki perbedaan pada jumlah perawat, jumlah *shift*, panjang penjadwalan serta nilai pada *hard* dan *soft constraint*. Dengan perbedaan tersebut dapat dikatakan ukuran tiap *instance* berbeda. Setiap *instance* berisi lima *sheet* yaitu, *Employees* yang berisi daftar perawat dalam *instance* tersebut dan beserta atribut yang diperlukan, *Shifts* yang berisi daftar *shift* dalam *instance* tersebut dan beserta atribut yang diperlukan, *Manpower plan* yang berisi jumlah *shift* yang dibutuhkan dalam setiap harinya, *Constraints* yang berisi batasan pada *instance* tersebut berserta nilainya, dan *Patterns* yang berisi pola *shift* diinginkan dan tidak diinginkan yang nantinya digunakan dalam perhitungan penalti. Informasi umum tiap *instance* disajikan pada tabel 4.1 berikut;

Tabel 4.1 Informasi Umum Dataset

Instance	**Jumlah Perawat**	**Jumlah Jam**	**Jumlah Hari**	**Jumlah *Shift***
OpTur1	51	19170	84	9
OpTur2	83	12819	42	9
OpTur3	29	4159.5	42	8
OpTur4	30	17352	168	8
OpTur5	20	2280	28	9
OpTur6	54	18978	84	5
OpTur7	15	2209.5	42	6

Dari tabel 4.1 yang telah diatas, dapat dihitung ukuran *instance* berdasarkan jumlah kemungkinan penentuan *shift* untuk setiap

perawat pada setiap harinya (termasuk libur). Pada tabel 4.2 berikut dapat disimpulkan bahwa ukuran *instance* terkecil yaitu pada OpTur7 dan yang terbesar yaitu pada OpTur4.

Tabel 4.2 Ukuran Instance Berdasarkan Kemungkinan

Instance	**Kemungkinan**
OpTur1	42840
OpTur2	34860
OpTur3	10962
OpTur4	45360
OpTur5	5600
OpTur6	27216
OpTur7	4410

4.2. Pembacaan *Dataset*

Pembacaan seluruh *instance* dilakukan untuk tiap *sheet* dalam satu *file*. Sebuah *instance* memiliki lima *sheet* sehingga pembacaan *instance* dilakukan sebanyak lima kali sesuai dengan kebutuhan untuk tiap *sheet*. Hasil dari pembacaan akan disimpan kedalam *class* dan dibuat menjadi *object* sebanyak isi dari sebuah *sheet*. Pada *sheet Employees* akan disimpan kedalam *class* perawat beserta atribut yang ada. Pada *sheet Shifts* dan *Manpower plan* akan disimpan menjadi satu *class* yaitu *class shift* dengan beserta atribut yang ada. *Sheet Constraints* dan *Patterns* disimpan menjadi *class* yang berbeda sesuai dengan kebutuhan.

4.3. Penyesuaian Model Matematika

Model matematika yang ada perlu dilakukan penyesuaian menggunakan asumsi karena keterbatasan informasi yang diperoleh. Penyesuaian model matematika dilakukan pada *objective function*.

$$Minimize: f = \sum_{m=1}^{9} K_m p_m \qquad (4.1)$$

Model matematika *objective function* pada persamaan 4.1 terdapat bobot (K_m) untuk tiap penalti, nilai bobot tersebut diasumsikan menjadi 1 (satu), sehingga dapat diartikan untuk semua penalti memiliki bobot yang sama. Sehingga, perhitungan *objective function* hanya menjumlahkan semua penalti yang ada tanpa menggunakan bobot.

4.4. Pembuatan Solusi Awal

Solusi awal yang dihasilkan berupa sebuah tabel atau matriks dua dimesi dengan kolom sebanyak jumlah hari yang akan dijadwalkan dan baris sebanyak jumlah perawat yang akan dijadwalan. Isi dari tabel atau matriks tersebut adalah sebuah nomor *shift* yang akan dijadwalkan untuk seorang perawat dalam tiap harinya, atau nomor nol jika tidak mendapat jadwal *shift*.

Penentukan nomor *shift* dilakukan dengan menggunakan dengan 2 skenario yaitu secara urut dan secara acak. Penentuan nomor *shift* secara urut dilakukan dari nomor *shift* yang pertama hingga nomor yang terakhir yang diberikan kepada perawat secara urut dari perawat pertama hingga terakhir. Penentuan nomor *shift* secara acak dilakukan dari nomor *shift* yang pertama hingga terakhir dan diberikan kepada perawat secara acak hingga kebutuhan *shift* dalam hari tersebut terpenuhi. Kedua scenario penentuan *shift* tersebut dilakukan dengan tahapan hari diakhir pekan terlebih dahulu dan dilanjutkan pada hari biasa biasa. Seiring dilakukannya percobaan tersebut, pengecekan terhadap *hard constraints* juga dilakukan. Pada pengecekan tersebut, tidak dilakukan pada semua *hard constraints*, tetapi hanya beberapa *hard constraints* yang dapat dicek pada saat penentuan nomor *shift*. Hal tersebut dikarenakan terdapat 2 macam *hard constraints* dari segi pengecekannya yaitu ketika pembuatan solusi dan solusi telah terbentuk. Pada tabel 4.3 menampilkan tipe pengecekan *hard constraint*.

Hard constraint yang belum terpenuhi pada saat penentuan nomor *shift*, selanjutnya dilakukan penyelesaian seacara satu persatu. Penyelesaian *hard constraint* yang belum terpenuhi dilakukan dengan menggunakan optimasi. Optimasi tersebut dijalankan dengan tujuan menimalkan pelanggaran terhadap *hard constraint* yang terjadi hingga tidak terjadi pelanggaran.

Tabel 4.3 Tipe Pengecekan Hard Constraints

No	Keterangan *Hard Constraint*	Tipe Pengecekan
1	Maksimal 1 *shift* dalam 1 hari untuk tiap perawat	Pembuatan solusi
2	Kebutuhan perawat tiap hari harus terpenuhi	Pembuatan solusi
3	Jumlah jam kerja tiap perawat tidak melebihi batas toleransi kontrak kerja	Solusi terbentuk
4	Perawat hanya dapat bekerja sesuai dengan kompetensi yang dimiliki	Pembuatan solusi
5	Jam jeda antar *shift* pada hari yang berurutan untuk seorang perawat harus melebihi batas minimal	Pembuatan solusi
6	Dalam setiap minggu harus memiliki jam libur kerja secara kontinu melebihi batas minimal	Solusi terbentuk
7	Jam kerja dalam seminggu tidak boleh melebihi batas maksimal	Pembuatan solusi

4.5. Penerapan *Low Level Heuristics*

Low level heuristics atau yang disebut LLH merupakan suatu cara yang digunakan untuk menghasilkan suatu kandidat solusi baru dan menggantikan solusi yang sudah ada. LLH yang diterapkan pada tahap ini ada 3 sesuai pada rujukan yang digunakan yaitu, *2-exchange*, *3-exchange*, dan *double 2-exchange*. Penerapan LLH digunakan dalam pembuatan solusi awal dan optimasi solusi awal.

Penerapan LLH *2-exchange* yaitu melakukan pemilihan acak pada salah satu hari. Kemudian memilih 2 perawat secara acak yang 1 perawat harus memiliki *shift* atau jam kerja dan 1

perawat yang lain harus tidak memiliki *shift* atau jam kerja pada hari tersebut. Setelah 1 hari dan 2 perawat ditemukan maka dilakukan penukaran shift kepada 2 perawat tersebut.

Penerapan LLH *3-exchange* yaitu dengan memilih acak pada salah satu hari. Kemudian memilih 3 perawat secara acak yang pada hari tersebut memiliki *shift* atau jam kerja. Ketika hari dan perawat sudah terpilih maka dilakukan penukaran *shift*. *Shift* pada perawat 1 dikerjakan oleh perawat 3, *shift* pada perawat 2 dikerjakan oleh perawat 1, dan *shift* pada perawat 3 dikerjakan oleh perawat 2.

Penerapan LLH *double 2-exchange* yaitu dengan memilih 2 hari secara acak. Kemudian, memilih 2 perawat secara diurut dari perawat pertama hingga menemukan 2 perawat dengan ketentuan berikut;

a. Perawat 1 pada hari ke 1 memiliki jam kerja atau *shift* dengan kategori *shift* yang sama dengan perawat 2 pada hari ke 2.
b. Perawat 1 pada hari ke 2 tidak memiliki jam kerja atau *shift.*
c. Perawat 2 pada hari ke 1 tidak memiliki jam kerja atau *shift*.
d. Perawat 2 pada hari ke 2 memiliki jam kerja atau *shift* dengan kategori *shift* yang sama dengan perawat 1 pada hari ke 1.

Setelah menemukan 2 perawat dengan ketentuan tersebut, maka *shift* hari ke 1 untuk perawat 1 ditukar dengan *shift* hari ke 1 untuk perawat 2 dan berlaku juga untuk hari ke 2.

4.6. Penerapan Algoritma *Late Acceptance Hill Climbing*

Penerapan optimasi menggunakan algoritma LAHC dilakukan pada solusi awal yang telah *feasible*. Algoritma LAHC digunakan *move acceptance* atau penerimaan solusi pada tiap iterasi. Dalam setiap iterasi terbentuk sebuah kandidat solusi baru yang diperoleh dari perubahan oleh LLH terhadap solusi

yang sudah ada. Kemungkinan yang terjadi dalam setiap iterasi yaitu kandidat solusi memiliki penalti lebih baik (atau sama), lebih buruk, dan melanggar *hard constraint*. Penerapan pemberhentian iterasi atau *stopping criteria* yang dugunakan sesuai pada *pseudocode* LAHC pada gambar 4.1 yaitu karena jumlah batas iterasi dan batas *idle* yang sudah tercapai.

```
Produce an initial solution s
Calculate initial cost function C(s)
Specify L_h
For all k∈{0...L_h-1} f_k:=C(s)
First iteration I:=0; I_idle:=0
Do until (I>100000) and (I_idle>I*0.02)
   Construct a candidate solution s*
   Calculate a candidate cost function C(s*)
   If C(s*)≥C(s)
      Then increment the idle iterations number I_idle:=I_idle+1
      Else reset the idle iterations number I_idle:=0
   Calculate the virtual beginning v:=I mod L_h
   If C(s*)<f_v or C(s*)≤C(s)
      Then accept the candidate s:=s*
      Else reject the candidate s:=s
   If C(s)<f_v
      Then update the fitness array f_v:=C(s)
   Increment the iteration number I:=I+1
```

Gambar 4.1 Pseudocode Late Acceptance Hill Climbing
(Sumber : Burke, 2017)

4.7. Pembuatan Skenario Parameter

Pembuatan skenario parameter merupakan sebuah percobaan yang bertujuan untuk menemukan hasil optimasi yang terbaik melalui pencarian *history length* dari LAHC yang optimal untuk digunakan. *History length* merupakan *array* pembanding yang berisi *objective function* pada iterasi sebelumnya yang akan dibandingkan dengan kandidiat solusi pada iterasi selanjutnya. Nilai *history length* yang diuji coba menggunakan rujukan *paper* [13] dan disesuaikan dengan permasalahan yang ada yaitu 100, 300, 500, 700 dan 900 dengan menggunakan iterasi 1.000.000.

4.8. Perbandingan Algoritma

Algoritma yang digunakan akan dibandingkan dengan algoritma lain untuk mengetahui apakah performanya lebih bagus atau tidak. Setiap algoritma dan setiap parameternya akan dilakukan uji coba sebanyak 10 kali perulangan [14]. Perbandingan algoritma yang akan digunakan pada tugas akhir ini dicantumkan pada tabel 4.4.

Tabel 4.4 Perbandingan Algoritma

No.	Algoritma	Jumlah Iterasi
1.	Hill Climbing	1.000.000
2.	LAHC tanpa penyesuaian parameter	1.000.000
3.	LAHC dengan penyesuaian parameter	1.000.000

4.9. Kesimpulan Perancangan

Rancangan yang disusun pada intinya yaitu dalam tugas akhir ini akan melakukan perbandingan performa 2 algoritma yaitu *Hill Climbing* dan LAHC dengan 1.000.000 iterasi dan 10 kali perulangan. Performa algoritma diukur menggunakan penalti yang didapat dari perhitungan *objective function* yang telah disesuaikan. Cara yang dilakukan untuk menemukan *objective function* yang paling optimal yaitu menggunakan 3 LLH yang telah dijelaskan. Rancangan yang telah disusun akan diimplementasikan dan dijelaskan pada bab 5 implementasi.

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Bab ini menjelaskan proses implementasi penyelesaian permasalahan penjadwalan perawat serta optimasi dengan menggunakan algoritma LAHC yang dilakukan sesuai pada bab 4 perancangan. Seluruh kode program yang digunakan pada tugas akhir ini tercantum pada LAMPIRAN A.

5.1. Pembacaan *Dataset*

Dataset yang digunakan dalam tugas akhir ini berupa 7 *instance* (*file excel*). Setiap *file* memiliki 5 *sheet*. Pembacaan pada tiap *instance* dilakukan untuk tiap *sheet* dan bersifat *generic* atau dapat digunakan pada *file* yang lain. Oleh karena itu, untuk membaca *file* yang lain tidak diperlukan pengubahan terhadap kode program yang dibuat.

5.1.1. Pembacaan *Sheet Employees*

Pembacaan pada *sheet emplyees* dimulai pada data yang berupa tabel (baris keenam), atau dengan kata lain mengabaikan adanya data pada baris kedua dan data nama kolom tabel. Hal tersebut disebabkan karena isi dari baris kedua dapat juga diperoleh dari perhitungan jumlah *employee* pada tabel. Pembacaan dilakukan dengan membatasi hanya pada baris dan kolom yang memiliki isi. Pengecekan pada *cell* yang memiliki isi dilakukan secara otomatis, sehingga pembacaan akan terhenti otomatis ketika menemukan *cell* yang tidak memiliki isi. Data yang terbaca akan disimpan kedalam struktur data *array* 2 dimensi yang bertipe *String*. Ukuran *array* untuk menyimpan data yang sudah dibaca disesuaikan dengan data yang ada.

Setelah data tersimpan kedalam *array*, data per baris akan diubah bentuknya menjadi sebuah objek pada *class Perawat*. Data pada tiap kolom akan disimpan menjadi atribut dengan tipe data tertentu dari objek terebut. Penyimpanan atribut akan ditunjukkan pada tabel 5.1.

Tabel 5.1 Penyimpanan Data Atribut Sheet Employees

Nomor Kolom	Nama Kolom	Tipe Data	Nama Atribut
1	ID	*integer*	*id*
2	Work Week	*double*	*jam*
3	Working Weekends	*array integer*	*minggu*
4	Competence	*String*	*ahli*

5.1.2. Pembacaan *Sheet Shifts*

Seperti halnya pembacaan *sheet employees*, pembacaan pada *sheet shifts* juga hanya dilakukan pada data dalam isi tabel yang dimulai pada baris kelima. Pembacaan *sheet shifts* ini juga dilakukan pembatasan otomatis agar pembacaan bias berhenti ketika menemui *cell* yang kosong. Data yang sudah terbaca juga akan disimpan kedalam *array* sebagai bentuk pengembalian dari *method*.

Ketika data telah tersimpan dalam *array*, data per baris akan diubah menjadi bentuk objek pada *class Shift*. Data pada tiap kolom akan disimpan menjadi atribut dari objek tersebut. Penyimpnan atribut akan ditunjukkan pada tabel 5.2.

5.1.3. Pembacaan *Sheet Manpower Plan*

Pada pembacaan *sheet Manpower Plan* terdapat 2 data diluar tabel yang tidak dilakukan pembacaan yaitu jumlah *shift* dan panjang penjadwalan (*length of planning horizon*). Data jumlah *shift* sudah tersimpan ketika melakukan pembacaan *sheet Shifts*, sedangkan panjang penjadwalan dilakukan penyimpanan secara manual. Hal ini dilakukan untuk mempermudah proses pembacaan data. Pembacaan *sheet Manpower Plan* sama dengan *sheet* sebelumnya yaitu, menggunakan pengecekan otomatis pembacaan berhenti ketika bertemu dengan *cell* yang kosong. Ketika data terbaca, data akan disimpan dalam *array String* 2 dimensi.

Tabel 5.2 Penyimpanan Data Atribut Sheet Shifts

Nomor Kolom	Nama Kolom	Tipe Data	Nama Atribut
1	ID	*integer*	*no*
2	Monday	*array double*	*durasi*
3	Tuesday		
4	Wednesday		
5	Thursday		
6	Friday		
7	Saturday		
8	Sunday		
9	Shift Category	*integer*	*kat*
10	Name	*String*	*nama*
11	Start Time (Hours)	*Localtime*	*mulai*
12	Start Time (Minutes)		
13	End Time (Hours)	*Localtime*	*akhir*
14	End Time (Minutes)		
15	Competence Needed	*String*	*kompe*

Setelah data tersimpan dalam *array*, data tidak disimpan kedalam *class* baru. Akan tetapi, data akan disimpan kedalam *class Shift* yang sudah terbentuk. Penyimpanan hanya dilakukan pada kolom 2-8, karena kolom 1 dan 9 data sudah sama dengan pembacaan *sheet Shift*. Data pada kolom 2-8 disimpan pada atribut baru dengan tipe data *array integer* dengan nama kebutuhan.

5.1.4. Pembacaan *Sheet Constraints*

Pembacaan *sheet Constraints* dilakukan hanya pada kolom *value* (kolom D) pada baris ke-5 hingga ke-33. Data yang terbaca akan disimpan kedalam *array String* 1 dimensi dengan mengabaikan *cell* yang tidak memiliki isi.

Setelah data tersimpan kedalam *array*, data akan disimpan menjadi objek dalam *class Batasan*. Setiap baris data akan menjadi atribut sesuai dengan ketentuan *hard* dan *soft constraint* yang ada. Tabel 5.3 menampilkan penyimpanan atribut pada *sheet Constraints*.

Tabel 5.3 Penyimpanan Data Atribut Sheet Constraints

Nomor Baris	Tipe Data	Nama Atribut	Keterangan
5	*boolean*	*hc1*	Hard Constraint 1
6	*boolean*	*hc2*	Hard Constraint 2
7	*double*	*hc3*	Hard Constraint 3
8	*boolean*	*hc4*	Hard Constraint 4
9	*Localtime*	*hc51a*	Hard Constraint 5
10	*Localtime*	*hc51b*	
11	*Localtime*	*hc52a*	
12	*Localtime*	*hc52b*	
13	*Localtime*	*hc53a*	
14	*Localtime*	*hc53b*	
15	*integer*	*hc6*	Hard Constraint 6
16	*double*	*hc7*	Hard Constraint 7
19	*integer*	*sc1a*	Soft Constraint 1
20	*integer*	*sc1b*	
21	*integer*	*sc1c*	
22	*integer*	*sc2*	Soft Constraint 2
23	*integer*	*sc3a*	Soft Constraint 3
24	*integer*	*sc3b*	
25	*integer*	*sc3c*	
26	*integer*	*sc4*	Soft Constraint 4
27	*integer*	*sc5aMax*	Soft Constraint 5
	integer	*sc5aMin*	
28	*integer*	*sc5bMax*	
	integer	*sc5aMin*	
29	*integer*	*sc5bMax*	
	integer	*sc5aMin*	
30	*boolean*	*sc6*	Soft Constraint 6
31	*boolean*	*sc7*	Soft Constraint 7
32	*integer*	*sc8*	Soft Constraint 8
33	*integer*	*sc9*	Soft Constraint 9

5.1.5. Pembacaan *Sheet Patterns*

Pembacaan *sheet Patterns* dilakukan 2 kali yaitu, untuk *wanted patterns* dan *unwanted patterns*. Untuk *wanted patterns* pembacaan dilakukan pada baris ke-5. Pengecekan otomatis terkait *cell* yang kosong juga dilakukan pada tahap ini. Data yang terbaca akan disimpan dalam *array String* 2 dimensi.

Serupa dengan *wanted patterns*, pada pembacaan *unwanted* patterns juga menggunakan pengecekan otomatis ketika menemui *cell* kosong. Untuk pembacaan *unwanted* patterns dimulai pada baris ke-19. Setelah pembacaan data disimpan dalam *array String* 2 dimensi.

Setelah *wanted* dan *unwanted patterns* sudah tersimpan dalam array, masing-masing akan dijadikan sebuah objek dalam *class Pattern*. Setiap kolomnya akan dijadikan atribut. Atribut yang dibuat ditampilkan pada tabel 5.4.

Tabel 5.4 Penyimpanan Data Atribut Sheet Patterns

Nomor Kolom	Nama Kolom	Tipe Data	Nama Atribut
1	Starting Day	*integer*	*start*
2	Pattern	*array String*	*pat*

5.2. Pembuatan Solusi Awal

Pembuatan solusi awal bertujuan untuk menghasilkan sebuah solusi yang tidak melanggar *hard constraint* yang ada (*feasible*). Cara pembuatan solusi awal dilakukan dengan menentukan nomor *shift* secara berurutan dan secara acak, pengecekan terhadap *hard constraint*, penyimpanan solusi, dan perhitungan penalti.

5.2.1. Penentuan *Shift*

Tahap awal dalam pembuatan solusi awal yaitu dengan mencoba penentuan *shift* secara urut maupun acak. Dalam penentuan *shift* secara urut kepada perawat dilakukan dengan menggunakan 3 kali *looping for* yang ditujukan untuk hari,

perawat dan *shift*. *Looping* tersebut dapat diartikan mencoba kemungkinkan sebuah *shift* yang ditujukan untuk seorang perawat pada suatu hari. Dalam penentuan *shift* secara acak kepada perawat dilakukan dengan kombinasi *looping for* dan *while*. *Looping for* digunakan untuk perulangan hari, sedangkan *looping while* digunakan untuk perulangan *shift* hingga kebutuhan dalam hari suatu hari terpenuhi. Untuk penentuan perawat digunakan angka acak dengan fungsi *Math.random*. Dalam percobaan penentuan *shift*, dilakukan pengecekan pada *hard constraint* 2, 4, 5, dan 7. Pengecekan tersebut dilakukan dengan *method Boolean*. Hasil dari tahap ini yaitu sebuah solusi yang *feasible* terhadap *hard constraint* 1, 2, 4, 5, dan 7.

5.2.2. Penerapan *Low Level Heuristics*

Low level heuristics atau yang disebut LLH pada tahap ini digunakan untuk menyelesaikan *hard constraint* yang belum terpenuhi pada solusi yang dihasilkan pada penentuan *shift* secara berurutan. Implementasi kode program yang digunakan untuk penerapan LLH ini yaitu berupa *method void* untuk tiap LLH. Nama *method* yang digunakan yaitu *exhcange2* untuk LLH *2-exchange*, *exchange3* untuk LLH *3-exchange*, dan *doubleExchange2* untuk LLH *double 2-exchange*. Pada *method* tersebut terdapat parameter berupa *array integer* 2 dimensi yang merupakan solusi awal atau solusi yang akan dilakukan pemrosesan dengan LLH.

5.2.3. Penyelesaian *Hard Constraint* 3

Penyelesaian *hard constraint* 3 dilakukan agar solusi yang sudah didapat *feasible* terhadap *hard constraint* 3. Cara pertama yang dilakukan yaitu dengan menghitung selisih pelanggaran jam pada *hard constraint* 3 yang disebabkan oleh solusi yang ada. Penghitungan selisih pelanggaran jam tersebut dilakukan menggunakan *method* yang bernama *diffHour* dengan nilai pengembalian berupa *double*. *Method* tersebut memiliki parameter *array integer* 2 dimensi atau yang merupakan solusi yang akan dihitung selisih pelanggaran jam terjadi.

Dalam menyelesaikan *hard constraint* 3, dilakukan sebuah optimasi untuk meminimalkan selisih jam pelanggaran terjadi. Optimasi dilakukan dilakukan dengan semua LLH yang telah dibuat. Algoritma yang digunakan dengan *hill climbing* sederhana yaitu hanya menerima solusi dengan selisih jam pelanggaran yang sama atau yang lebih sedikit pada tiap iterasinya. Selain dari selisih jam, solusi yang akan diterima harus *feasible* terhadap *hard constraint* 2, 4, 5, dan 7. Oleh karena itu, setiap iterasi pada optimasi untuk menyelesaikan *hard constraint* 3 diperlukan pengecekan terhadap *hard constraint* 2, 4, 5, dan 7 pada kandidat solusi yang terbentuk. Iterasi yang dilakukan sebanyak 50000. Jika hingga akhir iterasi solusi tidak *feasible* terhadap *hard constraint* 3 maka dinyatakan solusi tersebut tidak bisa *feasible* terhadap *hard constraint* 3.

5.2.4. Penyelesaian *Hard Constraint* 6

Penyelesaian *hard constraint* 6 dilakukan hampir sama dengan penyelesaian pada *hard constraint* 3. Penyelesaian *hard constraint* 6 dilakukan dengan menggunakan metode optimasi. Optimasi dilakukan untuk meminimalkan pelanggaran yang terjadi oleh *hard constraint* 6. Pelanggaran yang terjadi dihitung dengan menggunakan *method* yang bernama *hitungHc6*. *Method hitungHc6* memiliki parameter solusi berupa *array integer* 2 dimensi dan memiliki luaran berupa *integer*. Sehingga, dapat diartikan *method hitungHc6* akan menghitung pelanggaran *hard constraint* 6 pada solusi dan memberikan hasil berupa jumlah pelanggarannya.

Sama halnya dengan penyelesaian *hard constraint* 3, optimasi untuk penyelesaian *hard constraint* 6 juga dilakukan menggunakan algoritma *hill climbing* sederhana. Yang membedakan yaitu, pada penyelesaian *hard constraint* 6 ditiap iterasinya dilalukan pengecekan terhadap *hard constraint* 2, 3, 4 dan 5. Oleh karena itu, hasil dari optimasi tersebut solusinya tidak *feasible* terhadap *hard constraint* 7 dan perlu dilakukan penyelesaian. Iterasi pada optimasi dilakukan hingga tidak

terjadi pelanggaran *hard constraint* 6 atau jumlah pelanggarannya 0.

5.2.5. Penyelesaian *Hard Constraint* 7

Penyelesaian *hard constraint* 7 dilakukan serupa dengan *hard constraint* 6 yaitu dengan optimasi. Yang membedakan yaitu, pada penyelesaian *hard constraint* 7, optimasi dilakukan untuk meminimalkan pelanggaran *hard constraint* 7 yang terjadi. Pelanggaran yang terjadi dihitung dengan *method hitungHc7*. Pada optimasi yang dilakukan, setiap kandidat solusi diterima apabila jumlah pelanggaran terhadap *hard constraint* 7 sama atau lebih sedikit dan tidak melanggar *hard constraint* 2, 3, 4, 5, dan 6. Sehingga dalam optimasinya juga diperlukan *method* pengecekan pelanggaran *hard constraint* 2, 3, 4, 5, dan 6. Iterasi pada optimasi dilakukan hingga tidak terjadi pelanggaran *hard constraint* 7 atau jumlah pelanggarannya 0. Jika solusi telah *feasible* pada tahap ini, maka solusi tersebut dapat dinyatakan solusi awal yang *feasible*.

5.2.6. Penghitungan Penalti Solusi

Penghitungan penalti solusi dilakukan ketika solusi tersebut sudah *feasible*. Penghitungan penalti dilakukan dengan mengitung seluruh pelanggaran terhadap *soft constraint* yang terjadi. Solusi yang *feasible* akan disimpan menjadi objek dalam *class Solusi*. Dalam *class Solusi* tersebut terdapat *method* untuk menghitung penalti yang bernama *countPenalty*. *Method countPenaty* memiliki parameter yaitu solusi dan luaran berupa nilai *double*. Sehingga, dapat diartikan *method countPenalty* akan menghitung penalti dari sebuah solusi dan memberikan nilai penalti tersebut berupa angka desimal.

5.2.7. Penyimpanan Solusi Awal

Solusi awal yang sudah diketahui penaltinya akan disimpan untuk dilanjutkan kedalam tahap optimasi. Penyimpanan yang dilakukan terbagi menjadi 3 bagian yaitu, penyimpanan solusi berupa nomor *shift*, penyimpanan solusi berupa nama *shift*, dan penyimpanan *history* solusi.

Penyimpanan solusi berupa nomor *shift* yaitu, solusi akan disimpan kedalam *file .txt*. Nomor *shift* pada solusi antar hari akan diberi jarak menggunakan spasi (secara horizontal) dan antar perawat ditulis berurutan dari atas kebawah (secara vertikal). Penyimpanan solusi berupa nama *shift* juga dilakukan seperti nomor *shift* akan tetapi yang ditullis berupa nama *shift*. Penyimpanan *history* dilakukan untuk menyimpan detail dari penyimpanan solusi. Tabel 5.5 menampilan data dan tipe data dari solusi awal yang disimpan dalam *history*.

Tabel 5.5 Data dan Tipe Data Penyimpanan Solusi Awal

No.	Data	Tipe Data
1.	Penalti	String
2.	Durasi pembuatan solusi awal	String
3.	Waktu pembuatan solusi awal	String

5.3. Optimasi Solusi

Optimasi dilakukan bertujuan untuk meminimalkan penalti atau jumlah pelanggaran terhadap SC. Implementasi optimasi yang digunakan yaitu dengan cara membuat *method void* sesuai dengan algoritma yang digunakan pada *class Solusi* yang telah dibuat sebelumnya

5.3.1. Pembacaan Solusi Awal

Tahap pertama dalam melakukan optimasi yaitu membaca solusi awal yang berupa nomor *shift* yang terlah disimpan pada tahap pembuatan solusi awal dan data mentah. Pembacaan solusi awal dilakukan dengan menggunakan *BufferedReader*. Hasil pembacaan solusi awal akan disimpan menjadi sebuah objek pada *class Solusi* dengan nama objek yaitu *sol*. Objek *sol* tersebut yang akan dilakukan optimasi dengan menggunakan *method* atau fungsi yang ada pada *class Solusi*.

5.3.2. Implementasi Algoritma Late Acceptance Hill Climbing

Implementasi algoritma LAHC dilakukan dengan menggunakan *method void* pada *class Solusi*. Penggunaan *method* tersebut terdapat beberapa parameter yang dapat diubah-ubah yaitu, iterasi, *idle*, dan *history length*. Penggunaan parameter ditentukan sesuai dengan rujukan yang digunakan. Dalam tugas akhir ini tidak semua parameter dilakukan kustomisasi.

Iterasi merupakan jumlah perulangan yang dilakukan dalam optimasi. Iterasi disimpan dalam tipe data *integer* dengan nama *iterasi*. Jumlah iterasi yang dilakukan yaitu sebanyak 1.000.000 iterasi. *Idle* merupakan sebuah parameter yang digunakan untuk menandai iterasi ketika tidak menemukan solusi yang lebih baik. *Idle* akan kembali menjadi nol (0) ketika dalam optimasi menemukan solusi yang lebih baik. Jumlah *idle* yang digunakan sesuai dengan rujukan yaitu 2% dari jumlah iterasi atau dalam hal ini sebanyak 20.000. *Idle* disimpan dengan tipe data *integer* dengan nama *idle*. *History length* merupakan sebuah *array double* 1 dimensi dengan nama *penn* yang memuat *history* penalti. Panjang *history length* dalam tugas ini dilakukan kustomisasi. Cara kerja dari *history length* yaitu pada setiap iterasi, penalti dari kandidat solusi akan dibandingkan dengan *array penn* pada indeks yang sama dengan sisa hasil bagi iterasi sekarang dengan panjang *history length*. Jika kandidat solusi memiliki penalti lebih baik maka solusi yang sudah ada akan diganti dan *history length* pada indeks yang telah ditentukan juga akan diganti dengan penalti dari kandidat solusi. Dalam *method* implementasi LAHC juga terdapat pengecekan pada iterasi kelipatan 20.000. Pengecekan ini bertujuan untuk menampilkan progress optimasi dan pencatatan *trajectory* ke dalam *array double* 2 dimensi.

5.3.3. Penyimpanan Hasil Optimasi

Penyimpanan hasil optimasi dilakukan dengan menggunakan *method simpanSolusi* yang berada pada *class Solusi*. Format

nama penyimpanan *file* berbeda pada tiap macam *file*. Pada setiap *file* memiliki nomor percobaan yang menandakan hasil optimasi tersebut merupakan percobaan berapa yang dilakukan. Pemberian nomor percobaan dilakukan dengan pengecekan pada *folder* penyimpanan, apabila dalam *folder* masih kosong maka nomor percobaan dimulai dari nomor 1. Jika dalam *folder* sudah berisi sebuah hasil optimasi, maka penyimapan solusi selanjutnya diberi nomor setelah hasil optimasi yang sudah ada sebelumnya dengan menggunakan fungsi *while file exist*. Penyimpanan hasil optimasi dilakukan menjadi 4 bagian yaitu, penyimpanan solusi berupa nomor *shift*, nama *shift*, pencatatan *history*, dan pencatatan *trajectory*.

Penyimpanan solusi berupa nomor *shift*, nama *shift*, dan pencatatan *history* dilakukan sama dengan penyimpanan solusi yang ada pada pembuatan solusi awal. Yang membedakan yaitu, hanya pada format penamaan *file*. Dalam menyimpan hasil optimasi, format penamaan *file* diberi metode atau algoritma dan parameter yang digunakan. Pencatatan *trajectory*, merupakan pecatatan yang digunakan untuk keperluan analisis algoritma. Pencatatan *trajectory*, berisi *history* penalti ketika dioptimasi setiap 20.000 iterasi mulai dari iterasi ke 0 hingga iterasi ke 1.000.000. Semua bentuk penyimpanan solusi dilakukan dengan format *file .txt*.

5.4. Kesimpulan Implementasi

Implementasi yang dilakukan pada tugas akhir ini terbagi menjadi 3 bagian yaitu pembacaan *dataset* dari data mentah yang didapat, pembuatan solusi awal yang *feasible* terhadap semua *hard constraint* dan optimasi solusi untuk meminimalkan penalti. Hasil dari implementasi tugas akhir ini akan dibahas pada bab 6 hasil dan pembahasan.

Halaman ini sengaja dikosongkan.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dan analisis dari proses uji coba pembuatan solusi awal hingga optimasi permasalahan penjadwalan perawat menggunakan algoritma LAHC berdasarkan implementasi pada bab 5.

6.1. Lingkungan Uji Coba

Bagian ini membahas lingkungan uji coba yang dilakukan dalam implementasi tugas akhir. Lingkungan uji coba terkait dengan perangkat keras dan perangkat lunak yang digunakan. Tabel 6.1 menampilkan lingkungan uji coba yang digunakan dalam tugas akhir ini.

Tabel 6.1 Lingkungan Uji Coba

Perangkat Keras	**Spesifikasi**
Laptop	Lenovo Ideapad 300 14isk
Tipe Processor	Intel Core i5-6200 CPU @ 2.30GHz
Max. Memory	8 GB
Hard Drive Type	HDD 500GB
Perangkat Lunak	**Fungsi**
Windows 10 Edu	Sistem operasi
IntelliJ IDEA	IDE pemrgoraman Java
Microsoft Excel	Analisis hasil uji coba
Notepad	Penyimpanan hasil uji coba
Microsoft Word	Pembuatan laporan tugas akhir

6.2. Hasil Pembuatan Solusi Awal

Solusi awal merupakan sebuah solusi yang *feasible* terhadap seluruh *hard constraint* yang ada. Solusi tersebut masih terdapat pelanggaran pada *soft constraint* (SC). Pembuatan solusi awal dilakukan dengan menggunakan IntelliJ pada *method InitialSolution*.

6.2.1. Hasil Solusi Awal Instance OpTur1

Hasil pembuatan solusi awal untuk instance OpTur1 dapat dilihat pada tabel 6.2. Dari tabel tersebut proses pembuatan solusi awal terhenti karena tidak berhasilnya penyelesaian solusi yang feasible terhadap *hard constraint* 3. Sehingga, dengan kedua uji coba skenario penentuan *shift*, OpTur 1 dinyatakan tidak *feasible*.

Tabel 6.2 Hasil Pembuatan Solusi Awal OpTur1

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	Tidak *feasible*	Tidak *feasible*	Hingga akhir iterasi, solusi tidak dapat *feasible* karena masih ada perawat yang bekerja melebihi /kurang dari batas kontrak.
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	Tidak *feasible*	Tidak *feasible*	Proses tidak dilakukan karena terhenti pada penyelesaian *hard constraint* 3.
7	*Feasible*	*Feasible*	-

6.2.2. Hasil Solusi Awal Instance OpTur2

Hasil pembuatan solusi awal untuk OpTur2 tidak *feasible* dengan cara penentuan *shift* urut maupun acak. Hasil tersebut ditampilkan pada tabel 6.3. Pembuatan solusi awal OpTur2 disebabkan oleh tidak berhasilnya penyelesaian terhadap pelanggaran *hard constraint* 3.

Tabel 6.3 Hasil Pembuatan Solusi Awal OpTur2

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	Tidak *feasible*	Tidak *feasible*	Hingga akhir iterasi, solusi tidak dapat *feasible* karena masih ada perawat yang bekerja melebihi/kurang dari batas kontrak.
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	Tidak *feasible*	Tidak *feasible*	Proses tidak dilakukan karena terhenti pada penyelesaian *hard constraint* 3.
7	*Feasible*	*Feasible*	-

6.2.3. Hasil Solusi Awal Instance OpTur3

Solusi awal untuk *instance* OpTur3 tidak *feasible* dengan penentuan *shift* secara urut maupun acak. Hal ini disebabkan oleh pelanggaran *hard constraint* 3 yang tidak dapat diselesaikan, atau dengan kata lain masih terdapat perawat yang bekerja melebihi atau kurang dari batas kontrak jam kerja. Pada penyelesaian *hard constraint* 3, hingga akhir iterasi tidak mampu menghasilkan solusi yang *feasible. Hard constraint* 3 yang tidak *feasible* menyebabkan pembuatan solusi awal terhenti dan tidak melanjutkan pada penyelesaian *hard constraint* 6 secara otomatis. Tabel 6.4 menampilkan hasil detail pembuatan solusi awal OpTur 3.

Tabel 6.4 Hasil Pembuatan Solusi Awal OpTur3

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	Tidak *feasible*	Tidak *feasible*	Hingga akhir iterasi, solusi tidak dapat *feasible* karena masih ada perawat yang bekerja melebihi/kurang dari batas kontrak.
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	Tidak *feasible*	Tidak *feasible*	Proses tidak dilakukan karena terhenti pada penyelesaian *hard constraint* 3.
7	*Feasible*	*Feasible*	-

6.2.4. Hasil Solusi Awal Instance OpTur4

Pembuatan solusi awal OpTur4 dengan menggunakan skenario penentuan *shift* secara acak dapat memenuhi semua *hard constraint* yang ada. Sehingga, solusi awal OpTur4 dapat dikatakan *feasible* dan dapat dilanjutkan pada tahap optimasi untuk meminimalkan penalti. Penalti dari solusi awal OpTur4 yaitu sebesar 617.712 dengan durasi pembuatan selama 27.041 detik. Tabel 6.5 menampilkan hasil detail pembuatan solusi awal OpTur4.

Pembuatan solusi awal OpTur4 dengan menggunakan skenario penentuan *shift* secara urut menghasilkan solusi yang tidak *feasible*. Hal tersebut disebabkan oleh ketidakberhasilan penyelesaian terhadap pelanggaran *hard constraint* 3. Pada proses penyelesaian *hard constraint* 6 tidak dilakukan karena terhenti oleh ketidakberhasilan *hard constraint* 3.

Tabel 6.5 Hasil Pembuatan Solusi Awal OpTur4

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	Tidak *feasible*	*Feasible*	Pada penentuan *shift* secara urut, hingga akhir iterasi, solusi tidak dapat *feasible* karena masih ada perawat yang bekerja melebihi/kurang dari batas kontrak.
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	Tidak *feasible*	*Feasible*	Pada penentuan *shift* secara urut, proses tidak dilakukan karena terhenti pada penyelesaian *hard constraint* 3.
7	*Feasible*	*Feasible*	-

6.2.5. Hasil Solusi Awal Instance OpTur5

Hasil dari pembuatan solusi awal OpTur5 yaitu *feasible* dengan menggunakan skenario penentuan *shift* secara acak maupun dengan skenario penentuan *shift* secara urut. Semua *hard constraint* yang ada dapat terpenuhi sesuai dengan aturannya. Sehingga, solusi awal OpTur5 dapat dilanjutkan pada tahap optimasi untuk meminimalkan penalti. Hasil solusi awal yang terbentuk yaitu dengan penalti 274.193 dengan durasi 1.219 detik untuk skenario penentuan *shift* secara urut dan penalti 254.952 dengan durasi 0.812 untuk skenario penentuan *shift* secara acak. Tabel 6.6 menampilkan hasil pembuatan solusi awal OpTur5.

Tabel 6.6 Hasil Pembuatan Solusi Awal OpTur5

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	*Feasible*	*Feasible*	-
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	*Feasible*	*Feasible*	-
7	*Feasible*	*Feasible*	-

6.2.6. Hasil Solusi Awal Instance OpTur6

Hasil pembuatan solusi awal OpTur6 ditampilkan dalam tabel 6.7. Berdasarkan tabel tersebut, solusi awal OpTur6 tidak *feasible* menggunakan skenario penentuan *shift* secara urut maupun secara acak karena masih ada pelanggaran terhadap *hard constraint* 3 dan *hard constraint* 6. Pada penyelesaian *hard constraint* 3, hingga akhir iterasi tidak mampu menghasilkan solusi yang *feasible. Hard constraint* 3 yang tidak *feasible* menyebabkan pembuatan solusi awal terhenti dan tidak melanjutkan pada penyelesaian *hard constraint* 6 secara otomatis. Oleh karena itu, solusi awal OpTur6 tidak dapat dilanjutkan pada tahap optimasi.

Tabel 6.7 Hasil Pembuatan Solusi Awal OpTur6

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	Tidak *feasible*	Tidak *feasible*	Pada penentuan *shift* secara urut, hingga akhir iterasi, solusi tidak dapat *feasible* karena masih ada perawat yang bekerja melebihi/kurang dari batas kontrak.
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	Tidak *feasible*	Tidak *feasible*	Pada penentuan *shift* secara urut, proses tidak dilakukan karena terhenti pada penyelesaian *hard constraint* 3.
7	*Feasible*	*Feasible*	-

6.2.7. Hasil Solusi Awal Instance OpTur7

Pembuatan solusi awal OpTur7 sudah *feasible* dengan menggunakan skenario penentuan *shift* secara urut maupun secara acak. Hal tersebut disebabkan oleh tidak adanya pelanggaran terhadap *hard constraint*. Sehingga, solusi awal OpTur7 dapat dilanjutkan pada tahap optimasi. Hasil solusi awal yang terbentuk yaitu dengan penalti 710.645 dengan durasi 1.734 detik untuk skenario penentuan *shift* secara urut dan penalti 715.458 dengan durasi 0.562 detik untuk skenario penentuan *shift* secara acak. Tabel 6.8 menampilkan hasil detail pembuatan solusi awal OpTur7.

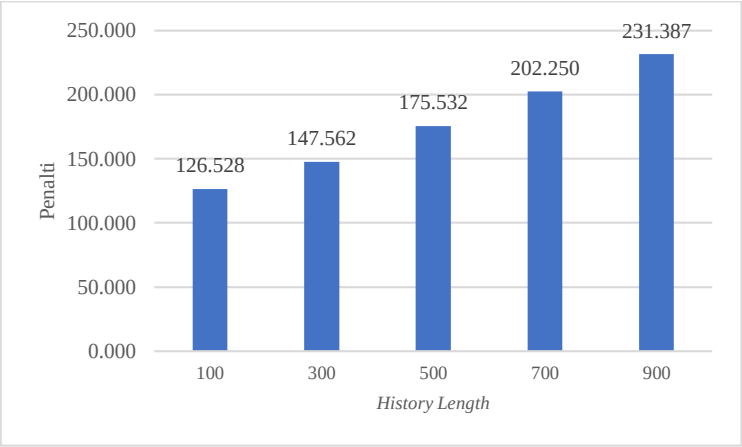
Tabel 6.8 Hasil Pembuatan Solusi Awal OpTur7

Hard Constraint	**Skenario Penentuan *Shift***		**Keterangan**
	Urut	**Acak**	
1	*Feasible*	*Feasible*	-
2	*Feasible*	*Feasible*	-
3	*Feasible*	*Feasible*	-
4	*Feasible*	*Feasible*	-
5	*Feasible*	*Feasible*	-
6	*Feasible*	*Feasible*	-
7	*Feasible*	*Feasible*	-

6.3. Penentuan Parameter Contoh *Instance*

Penentuan parameter salah satu contoh *instance* bertujuan untuk menenemukan sebuah parameter *history length* algoritma LAHC yang memiliki hasil paling optimal. Pencarian parameter hanya dilakukan pada salah satu contoh *instance*. *Instance* yang digunakan sebagai contoh yaitu OpTur4 karena merupakan *instance* yang berukuran paling besar. Parameter tersebut nantinya juga akan diterapkan pada *instance* yang lain untuk menjawab perbandingan algoritma LAHC tanpa penyesuaian parameter.

Hasil penentuan parameter untuk *instance* OpTur4, algoritma LAHC dengan *history length* 100 dapat menghasilkan solusi yang paling optimal dengan nilai penalti sebesar 126.528 seperti pada gambar 6.1.



Gambar 6.1 Hasil Penentuan Parameter Contoh Instance OpTur4

6.4. Hasil Optimasi OpTur4

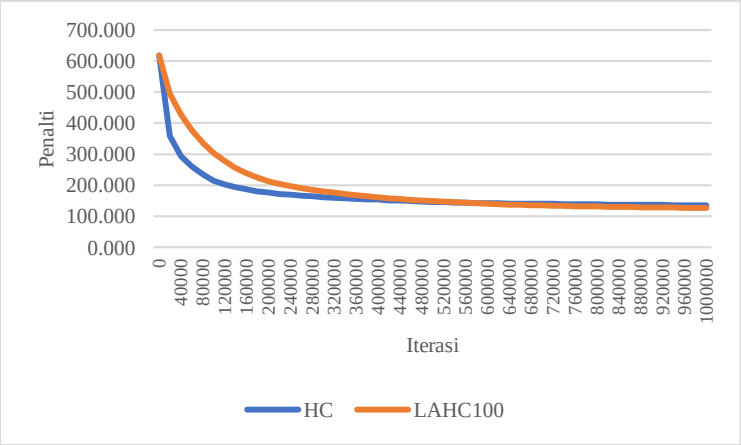
Perbandingan hasil optimasi yang dilakukan pada OpTur4 yaitu LAHC dengan penyesuaian parameter dan *Hill Climbing*. LAHC tanpa dilakukan penyesuai parameter tidak dilakukan karena OpTur4 sebagai contoh *instance* untuk yang lain sehingga harus dilakukan penyesuaian parameter.

Penyesuaian parameter untuk OpTur4 telah dibahas pada bagian penentuan parameter contoh *instance*. Hasil dari penyesuaian parameter yaitu menggunakan *history length* 100. Pada Tabel 6.9 dapat disimpulkan dari hasil rata-rata semua percobaan, algoritma LAHC dengan penyesuaian parameter dapat menghasilkan solusi yang lebih optimal daripada algortima *Hill Climbing* karena memiliki penurunan penalti yang lebih bessar yaitu 80% menjadi 126.528. Contoh solusi OpTur4 dari hasil optimasi yang paling optimal menggunakan LAHC dengan *history length* 100 dapat dilihat pada LAMPIRAN C.

Tabel 6.9 Hasil Perbandingan Optimasi OpTur4

	Hill Climbing	**LAHC dengan Penyesuaian Parameter (100)**
Penalti Awal	617.712	617.712
Terburuk	140.591	130.894
Terbaik	127.971	122.798
Rata-rata	135.499	126.528
Penurunan	78%	80%

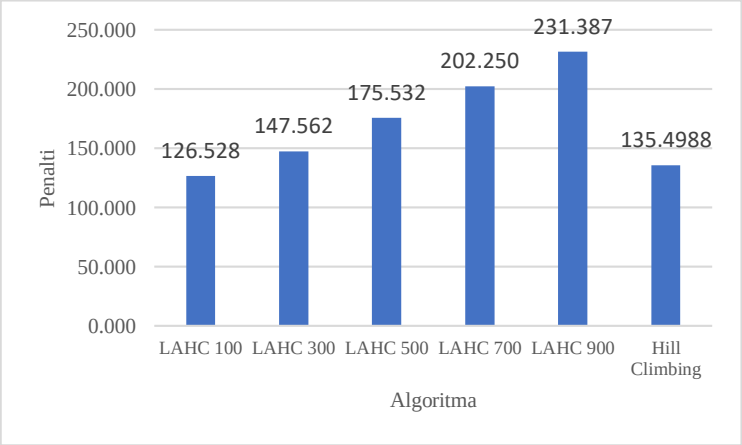
Pada gambar 6.2 menjelaskan penurunan penalti untuk 1.000.000 iterasi. Algoritma *Hill Climbing* memiliki grafik yang lebih curam diawal iterasi daripada LAHC, akan tetapi memiliki hasil akhir yang lebih buruk. Algoritma *Hill Climbing* mengalami penurunan penalti yang melandai sekitar pada iterasi 200.000 dan LAHC pada iterasi 400.000. Sehingga dapat disimpulkan jika pada OpTur4, perfomra algoritma LAHC lebih baik daripada *Hill Climbing*. Algoritma *Hill Climbing* memiliki performa lebih baik pada iterasi awal karena seiring bertambahnya iterasi akan terjebak dalam *local optima*.



Gambar 6.2 Grafik Trajectory Perbandingan Algoritma OpTur4

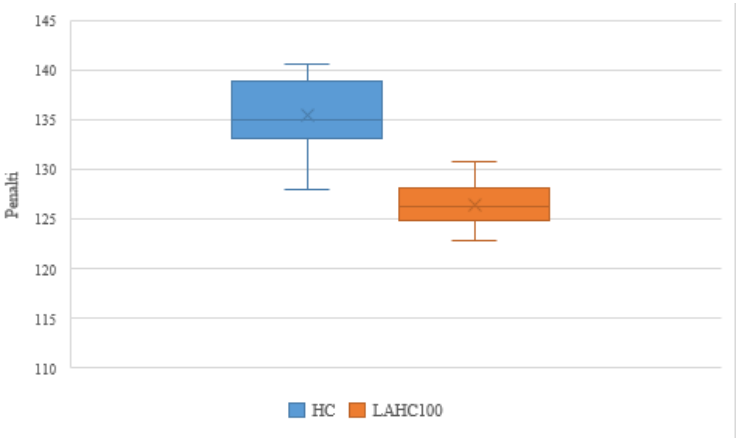
Apabila parameter algoritma LAHC yang digunakan tidak tepat, hasil yang dihasilkan bisa jadi lebih buruk daripada hasil optimasi dengan menggunakan algoritma *hill climbing* seperti

terlihat pada gambar 6.3. Gambar tersebut menampilkan perbandingan semua percobaan parameter LAHC yang digunakan dengan algoritma *hill climbing*. Hasil untuk parameter *history length* 300, 500, 700 dan 900 lebih buruk daripada *hill climbing*.



Gambar 6.3 Perbandingan Seluruh Hasil Percobaan

Persebaran data hasil penalti dengan 10 kali perulangan yang terangkum pada gambar 6.4 dapat disimpulkan, hasil optimasi dengan menggunakan LAHC cenderung menghasilkan nilai yang lebih konstan daripada menggunakan *hill climbing*. Selain itu, pada kedua algoritma yang digunakan tidak terdapat hasil nilai penalti yang *outlier*. Hal tersebut dapat diartikan tidak ada nilai penalti yang menyimpang dari nilai penalti lainnya.

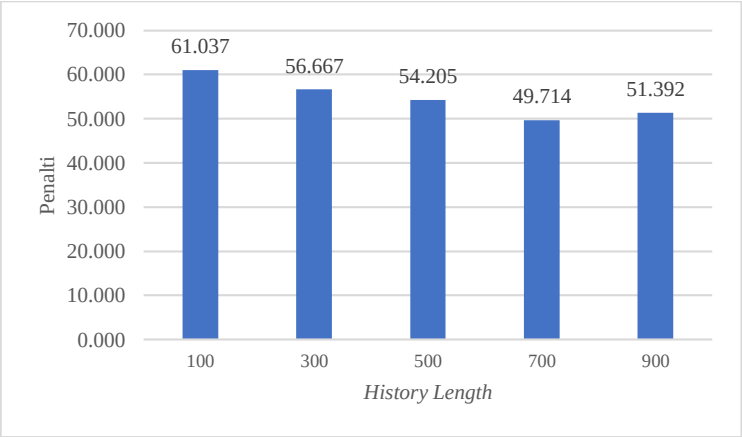


Gambar 6.4 Box Plot Perbandingan Algoritma OpTur4

6.5. Hasil Optimasi OpTur5

Dalam optimasi OpTur5 dilakukan perbandingan algoritma *Hill Climbing*, LAHC tanpa penyesuaian parameter dan LAHC dengan penyesuaian parameter. LAHC tanpa penyesuaian parameter menggunakan parameter yang sudah ditentukan pada contoh *instance* yaitu *history length* 100. LAHC dengan penyesuaian parameter akan mencari parameter yang sesuai untuk OpTur5 melalui uji coba.

Uji coba penyesuaian parameter OpTur5 dilakukan dengan 1.000.000 iterasi dengan 10 kali perulangan. Pada gambar 6.5 dapat disimpulkan bahwa *history length* 700 merupakan parameter terbaik karena dapat menghasilkan solusi paling optimal dengan penurunan penalti 81% menjadi 49.714. Dari hasil tersebut juga dapat mengindikasikan bahwa semakin panjang *history length* algoritma LAHC dengan jumlah iterasi yang sama hasil yang dihasiilkan tidak selalu lebih baik, akan tetapi memiliki suatu titik yang optimal sebagai *history length* yang paling tepat.

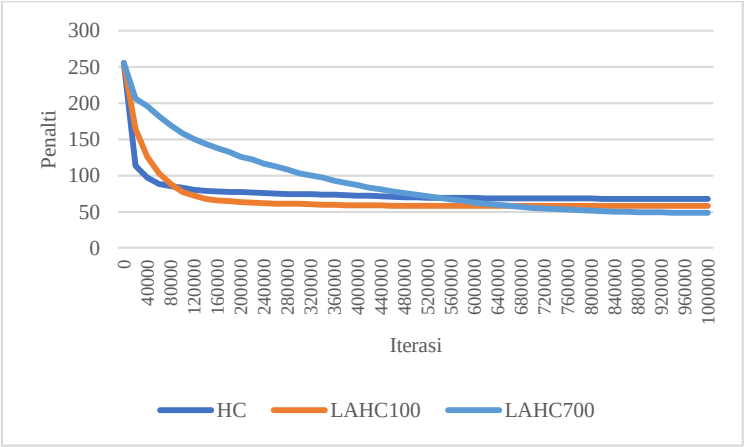


Gambar 6.5 Hasil Uji Coba Penyesuaian Parameter OpTur5

Hasil perbandingan algoritma untuk OpTur5 dapat dilihat pada tabel 6.10. Dari tabel tersebut hasil optimasi dengan menggunakan algoritma LAHC dengan penyesuian parameter (*history length* 700) mampu menghasilkan solusi yang paling optimal dengan nilai penalti 49.714 dibanding dengan algoritma *hill climbing* maupun LAHC tanpa penyesuaian parameter (*history length* 100). Contoh solusi OpTur5 dari hasil optimasi yang paling optimal menggunakan LAHC dengan *history length* 700 dapat dilihat pada LAMPIRAN D.

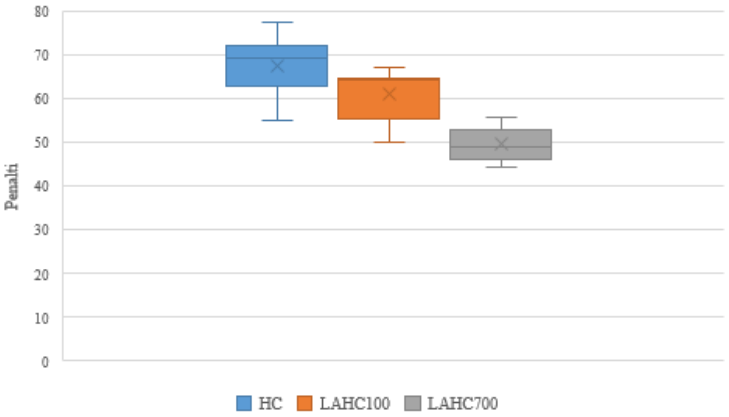
Tabel 6.10 Hasil Perbandingan Optimasi OpTur5

	Hill Climbing	LAHC tanpa Penyesuaian Parameter (100)	LAHC dengan Penyesuaian Parameter (700)
Penalti Awal	254.952	254.952	254.952
Terburuk	77.596	67.272	55.858
Terbaik	55.123	50.247	44.373
Rata-rata	67.474	61.037	49.714
Penurunan	74%	76%	81%



Gambar 6.6 Grafik Trejectory Perbandingan Algoritma OpTur5

Pada gambar 6.6 terlihat grafik perubahan penalti LAHC dengan *history length* 700 lebih landai. Algoritma *hill climbing* dan LAHC dengan *history length* 100 terlihat sangat curam diawal iterasi dan selanjutnya cenderung stagnan. Hal tersebut mengindikasikan algoritma *hill climbing* dan LAHC dengan *history length* 100 terjebak dalam kondisi *local optima*.



Gambar 6.7 Box Plot Perbandingan Algoritma OpTur5

Persebaran data hasil penalti 10 kali perulangan yang terlihat pada gambar 6.7 yaitu optimasi menggunakan algoritma LAHC dengan *history length* 700 memiliki hasil yang lebih konstan. Pada gambar tersebut juga tidak terdapat nilai hasil penalti yang *outlier* atau tidak menyimpang dari nilai penalti yang lain.

6.6. Hasil Optimasi OpTur7

Pada OpTur7 perbandingan algoritma yang dilakukan sama dengan OpTur5 yaitu *hill climbing*, LAHC tanpa penyesuaian parameter dan LAHC dengan penyesuaian parameter. LAHC tanpa penyesuaian parameter juga menggunakan *history length* 100 sesuai hasil dari penentuan parameter contoh *instance*.

Percobaan dalam penyesuaian parameter yang digunakan pada OpTur5 juga dilakukan 10 kali dengan 1.000.000 iterasi. Hasil yang didapat dari percobaan yaitu LAHC dengan *history length* 300 memiliki hasil yang paling optimal. Walaupun hasil *history length* 100, 300, 500, dan 700 memiliki penurunan penalti yang sama yaitu sebesar 7%, akan tetapi *history length* 300 memiliki nilai penalti yang paling rendah sebesar 665.00 seperti pada tabel 6.11.

Tabel 6.11 Hasil Uji Coba Penyesuaian Parameter OpTur7

	History Length				
	100	**300**	**500**	**700**	**900**
Terburuk	674.59	665.80	668.94	670.80	674.59
Terbaik	664.02	664.14	662.51	664.95	667.40
Rata-rata	668.15	665.00	665.78	668.39	669.88
Penurunan	7%	7%	7%	7%	6%

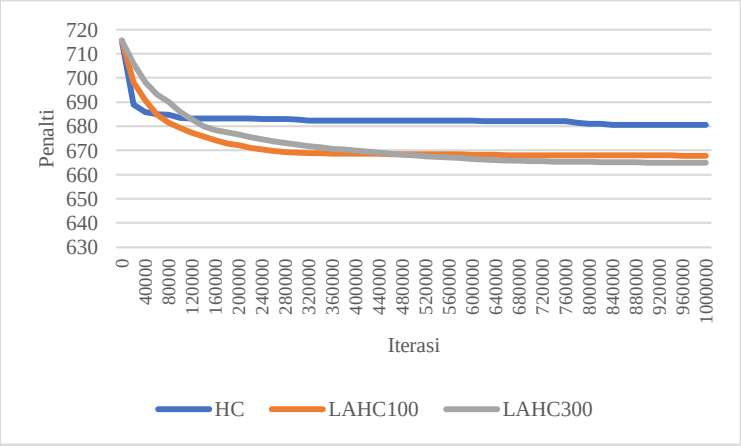
Perbandingan algoritma *Hill Climbing*, LAHC tanpa penyesuaian parameter dan LAHC dengan penyesuaian parameter memiliki hasil jika LAHC dengan penyesuaian parameter *history length* 300 merupakan solusi yang paling optimal dengan hasil 665.00 seperti yang terlihat pada tabel 6.12. Contoh solusi OpTur7 dari hasil optimasi yang paling

optimal menggunakan LAHC dengan *history length* 300 dapat dilihat pada LAMPIRAN E.

Tabel 6.12 Hasil Perbandingan Optimasi OpTur7

	Hill Climbing	LAHC tanpa Penyesuaian Parameter (100)	LAHC dengan Penyesuaian Parameter (300)
Penalti Awal	715.458	715.458	715.458
Terburuk	689.954	674.59	665.80
Terbaik	674.305	664.02	664.14
Rata-rata	680.466	668.15	665.00
Penurunan	5%	7%	7%

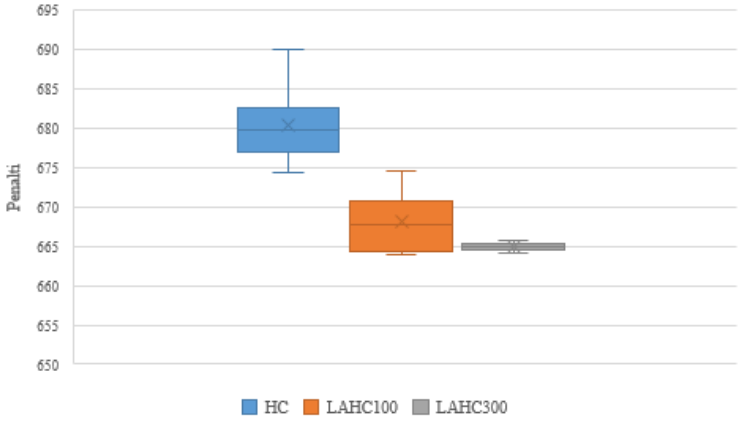
Pada gambar 6.8 terlihat jika pada algoritma *hill climbing* grafik perubahan penalti diawal iterasi cukup curam dan melandai di sektar iterasi 40.000. Hal tersebut menandakan jika algoritma *hill climbing* terjebak dalam *local optima*. Grafik LAHC dengan *history length* 300 melandai di sekitar iterasi 300.000 dan di iterasi tersebut LAHC dengan *history length* terjebak dalam *local optima*.



Gambar 6.8 Grafik Trajectory Perbandingan Algoritma OpTur7

Pada persebaran data perulangan optimasi sebanyak 10 kali yang tercantum pada gambar 6.9, dapat dilihat jika batas bawah

nilai penalti algoritma LAHC dengan *history length* 100 dan 300 memiliki persamaan pada kisaran 664. Akan tetapi, pada LAHC dengan *history length* 100 memiliki persebaran nilai penalti yang lebih luas sehingga menghasilkan nilai rata-rata yang lebih buruk dari LAHC dengan *history length* 300. Pada persebaran data tersebut juga tidak terdapat nilai *outlier* atau nilai yang menyimpang.



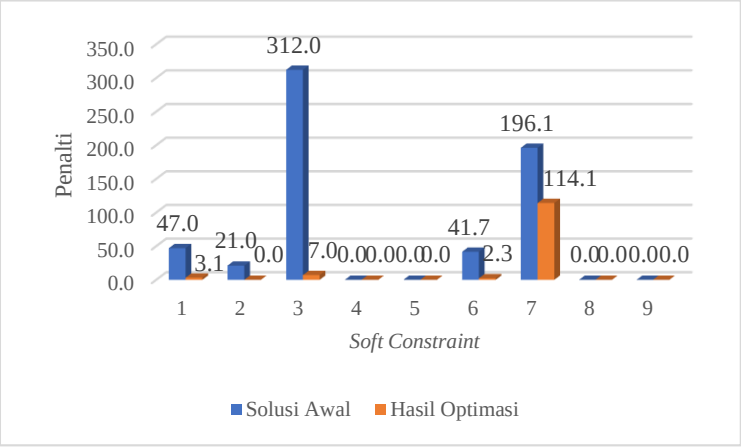
Gambar 6.9 Box Plot Perbandingan Algoritma OpTur7

6.7. Analisis Penalti Pelanggaran *Soft Constraint*

Solusi yang telah terbentuk dari solusi awal maupun yang telah dioptimasi terjadi pelanggaran terhadap *soft constraint* atau yang dinamakan sebagai penalti. Hal tersebut ditunjukkan oleh hasil yang telah dibahas pada beberapa sub bab sebelumnya jika penalti solusi yang dihasilkan lebih dari 0. Akan tetapi penalti dari solusi awal mengalami penurunan ketika solusi tersebut dioptimasi.

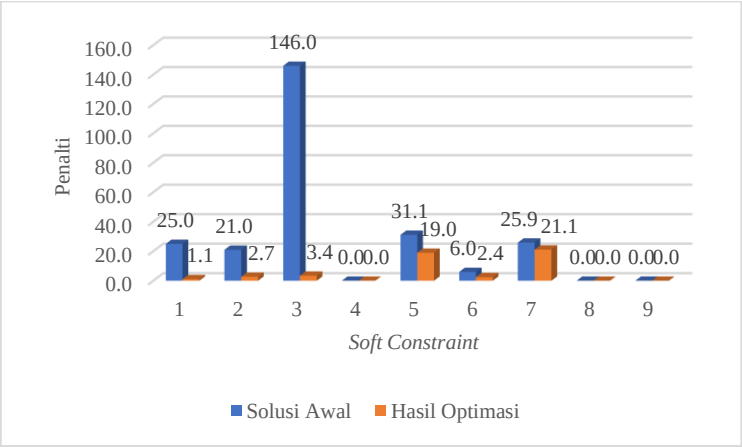
Pada solusi awal OpTur4, pelanggaran *soft constraint* didominasi oleh *soft constraint* 3 yaitu kurangnya hari kerja berurutan dengan kategori *shift* yang sama dari batas minimal yang ditentukan dan *soft constraint* 7 yaitu hari libur perawat perawat yang terpisah-pisah atau tidak berurutan. Ketika solusi

awal dioptimasi, seluruh penalti berhasil berkurang secara signifikan kecuali penalti pelanggaran *soft constraint* 7 yaitu sebesar 114.1. Hal tersebut dapat diartikan masih cukup banyak perawat yang hari liburnya terpisah-pisah oleh hari kerja. Hasil persebaran penalti OpTur4 ditampilkan pada gambar 6.10.



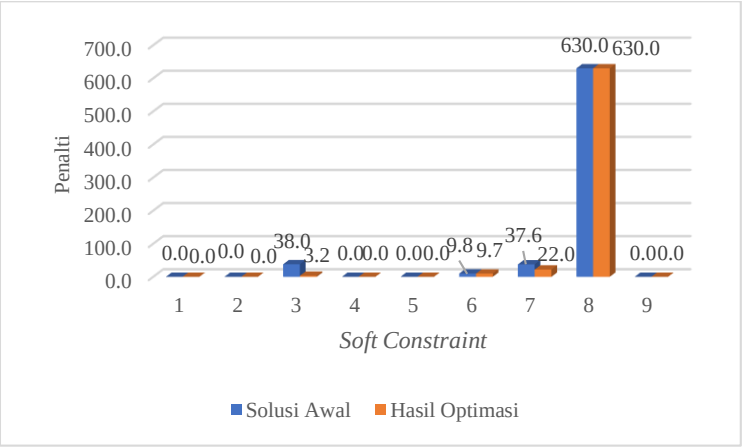
Gambar 6.10 Persebaran Penalti OpTur4

Pada solusi awal OpTur5 penalti yang terjadi didominasi oleh pelanggaran *soft constraint* 3 yaitu kurangnya hari kerja berurutan dengan kategori *shift* yang sama dari batas minimal yang ditentukan. Ketika solusi awal dioptimasi pelanggaran *soft constraint* 3 dapat berkurang secara siginifikan hingga mencapai 3.4. Akan tetapi pelanggaran terhadap *soft constraint* 5 dan 7 hanya sedikit mengalami penurunan jika dibandingkan dengan pelanggaran *soft constraint* yang lain. Hal tersebut dapat diartikan pada solusi yang telah dioptimasi, masih cukup banyak terdapat perawat yang memiliki jumlah hari kerja pada kategori *shift* tertentu melebihi atau kurang dari batas yang telah ditentukan serta hari libur perawat masih terpisah-pisah oleh hari kerja. Hasil persebaran penalti OpTur5 ditampilkan pada gambar 6.11.



Gambar 6.11 Persebaran Penalti OpTur5

Pada solusi awal dan hasil optimasi OpTur7 mengalami perbedaan persebaran penalti yang mencolok jika dibanding pada *instance* sebelumnya. Persebaran penalti pada solusi awal maupun hasil optimasi OpTur7 sangat didominasi oleh pelanggaran *soft constraint* 8 yaitu tidak adanya pola *shift* yang diinginkan dengan nilai penalti 630. Hal tersebut disebabkan oleh pola *shift* yang diinginkan berlawanan dengan *hard constraint* 5 yaitu adanya pola *shift* yang dilarang berurutan. Pada *paper* yang digunakan sebagai rujukan, perhitungan pada *soft constraint* 8 menggunakan penyesuaian dari model matematika yang ada, sedangkan pada tugas akhir ini perhitungan *soft constraint* 8 dilakukan sesuai model matematika tanpa adanya penyesuaian. Hasil persebaran penalti OpTur7 ditampilkan pada gambar 6.12.



Gambar 6.12 Persebaran Penalti OpTur7

6.8. Perbandingan Hasil dengan *Paper* Rujukan

Hasil optimasi dengan algoritma dan skenario terbaik yang telah dilakukan pada tugas akhir ini akan dibandingkan dengan *paper* rujukan dengan *dataset* yang sama yang berjudul *A hybrid approach for solving real-world nurse rostering problems*. Perbandingan tersebut bertujuan untuk mengetahui hasil dari tugas akhir ini lebih baik dari *paper* rujukan atau tidak.

Perbandingan hasil dengan *paper* rujukan ditampilkan pada tabel 6.13. Dari tabel tersebut dapat disimpulkan, walaupun pada tugas akhir ini telah berhasil menurunkan penalti yang terjadi, akan tetapi hasilnya tidak lebih baik daripada *paper* rujukan baik ditahap solusi awal maupun optimasi. Hal tersebut disebabkan oleh beberapa faktor. Faktor yang telah teridentifikasi pada tugas akhir ini yaitu perbedaan perhitungan bobot pada *objective function* yang telah dijelaskan pada sub bab penyesuaian model matematika dan perbedaan perhitungan pada penalti pelanggaran *soft constraint* 8 yang telah dijelaskan pada sub bab analisis penalti pelanggaran *soft constraint*.

Tabel 6.13 Perbandingan Hasil dengan Paper Rujukan

Instance	**Tahap**	**Penalti**	
		Tugas Akhir	***Paper* Rujukan**
OpTur4	Solusi Awal	617.712	203
	Optimasi	126.528	2.48
OpTur5	Solusi Awal	254.952	253
	Optimasi	49.714	8.34
OpTur7	Solusi Awal	715.458	378
	Optimasi	665	156

6.9. Kesimpulan Hasil Uji Coba

Hasil pembuatan solusi awal yang *feasible* terbanyak dilakukan dengan menggunakan skenario penentuan *shift* secara acak dengan jumlah 3 dari 7 *instance* seperti yang terlihat pada tabel 6.13. 3 *instance* tersebut akan dilanjutkan pada proses optimasi. Pada pembuatan solusi awal yang tidak berhasil *feasible*, mayoritas terhalang oleh *hard constraint* 3 dan 6 yang tidak terselesaikan. Dari *hard constraint* 3 yaitu masih adanya perawat yang bekerja melebihi atau kurang dari kontrak kerja. Sedangkan *hard constraint* 6 yaitu masih adanya perawat yang memiliki jam libur secara kontinu kurang dari batas minimal yang telah ditentukan dalam *hard constraint*.

Tabel 6.14 Hasil Umum Pembuatan Solusi Awal

No.	**Nama *Instance***	**Keterangan**
1.	OpTur1	Tidak *feasible*
2.	OpTur2	Tidak *feasible*
3.	OpTur3	Tidak *feasible*
4.	OpTur4	*Feasible*
5.	OpTur5	*Feasible*
6.	OpTur6	Tidak *feasible*
7.	OpTur7	*Feasible*

Dalam tahap optimasi, algoritma LAHC dengan penyesuaian parameter mampu menghasilkan solusi yang paling optimal dibandingkan algoritma *hill climbing* dan LAHC tanpa

penyesuaian parameter pada ketiga *instance* yang telah *feasible* seperti yang ditampilkan pada tabel 6.15. Akan tetapi tidak menutup kemungkinan hasil optimasi dari algoritma LAHC bisa lebih buruk dari *hill climbing*. Hal tersebut disebabkan oleh tidak tepatnya pemilihan parameter *history length* untuk suatu iterasi tertentu.

Tabel 6.15 Hasil Optimasi Terbaik

Instance	**Algoritma Terbaik**	**Penalti**	
		Nilai	Penurunan
OpTur4	LAHC (penyesuaian parameter)	126.5	80%
OpTur5	LAHC (penyesuaian parameter)	49.7	81%
OpTur6	LAHC (penyesuaian parameter)	665	7%

Jika dibandingkan dengan *paper* rujukan, hasil optimasi yang dilakukan pada tugas akhir ini tidak lebih baik untuk seluruh *instance* yang dilakukan optimasi. Hal tersebut disebabkan oleh beberapa faktor yang telah dijelaskan pada sub bab perbandingan hasil dengan *paper* rujukan. Untuk ringkasan hasil keseluruhan dari uji coba yang dilakukan pada tugas akhir ini akan dirangkum pada bab kesimpulan dan saran.

BAB VII
KESIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan dari hasil yang didapat pada tugas akhir ini dan saran untuk pengembangan penelitian selanjutnya agar dapat menjadi lebih baik.

7.1. Kesimpulan

Kesimpulan yang dapat diambil dari tugas akhir ini yaitu:

1. Solusi awal yang berhasil *feasible* sebanyak 3 dari 7 *instance* dengan menggunakan skenario penentuan *shift* secara acak.
2. Penerapan algoritma LAHC pada tiap *instance* untuk menghasilkan solusi yang paling optimal memiliki perbedaan penggunaan *history length*.
3. Dalam penggunaan algoritma LAHC dengan jumlah iterasi yang sama, *history length* lebih panjang tidak menjamin bahwa solusi yang dihasilkan akan lebih optimal.
4. Pada semua *instance* yang dioptimasi, algoritma LAHC dengan penyesuaian parameter untuk tiap *instance* memiliki solusi yang lebih optimal daripada algoritma *hill climbing* dengan selisih penurunan penalti 2% hingga 7%. Jika *history length* algoritma LAHC yang digunakan pada salah satu *instance* tidak tepat, maka solusi yang dihasilkan tidak menutup akan lebih buruk daripada algoritma *hill climbing*.
5. Pada *instance* OpTur4 algoritma LAHC mampu mereduksi penalti sebesar 80% sedangkan algoritma *hill climbing* hanya 78%.
6. Pada *instance* Optur5 algoritma LAHC mampu mereduksi penalti sebesar 81% sedangkan algoritma *hill climbing* hanya 74%.
7. Pada *instance* OpTur7 algoritma LAHC mampu mereduksi penalti sebesar 7% sedangkan algoritma *hill climbing* hanya 5%.

7.2. Saran

Berdasarkan proses penyelesaian tugas akhir ini, untuk penelitian kedepan penulis memberi saran yaitu:

1. Tugas akhir ini hanya menggunakan 1.000.000 iterasi dengan kombinasi *history length* 100, 300, 500, 700 dan 900. Untuk penelitian kedepan, iterasi yang digunakan dapat ditingkatkan dan diimbangi penyesuaian parameter dengan harapan untuk menghasilkan solusi yang lebih optimal.
2. Algorima pembanding yang digunakan dalam tugas akhir ini hanya menggunakan *hill climbing*. Untuk penelitian kedepan, dapat menggunakan algoritma yang lain seperti *Threshold Accepting*, *Tabu Search* dan lain-lain untuk membandingkan performa LAHC dengan algoritma lain khususnya pada permasalah *nurse rostering*.

DAFTAR PUSTAKA

[1] E. K. Burke, P. Causemaecker, G. Vanden Berghe, and H. Van Landeghem, "The State of the Art of Nurse Rostering," *J. Sched.*, vol. 7, no. November 2004, pp. 441–499, 2004.

[2] M. Stølevik, T. E. Nordlander, A. Riise, and H. Frøyseth, "A hybrid approach for solving real-world nurse rostering problems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6876 LNCS, no. 0314, pp. 85–99, 2011, doi: 10.1007/978-3-642-23786-7_9.

[3] C. A. Glass and R. A. Knight, "The nurse rostering problem: A critical appraisal of the problem structure," *Eur. J. Oper. Res.*, vol. 202, no. 2, pp. 379–389, 2010, doi: 10.1016/j.ejor.2009.05.046.

[4] E. K. Burke *et al.*, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013, doi: 10.1057/jors.2013.71.

[5] E. K. Burke and Y. Bykov, "The late acceptance Hill-Climbing heuristic," *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 70–78, 2017, doi: 10.1016/j.ejor.2016.07.012.

[6] A. Muklason, P. C. Bwananesia, Y. T. Sasmi Hidayatul, N. D. Angresti, and V. A. Supoyo, "Automated Examination Timetabling Optimization Using Greedy-Late Acceptance-Hyperheuristic Algorithm," *Proc. 2018 Int. Conf. Electr. Eng. Comput. Sci. ICECOS 2018*, pp. 201–206, 2019, doi: 10.1109/ICECOS.2018.8605194.

[7] R. Pordella, "Optimasi Penjadwalan Staf Rumah Sakit dengan Menggunakan Metode Late Acceptance Hill-Climbing Hyper-Heuristic (Studi Kasus: RSIA Kendangsari Surabaya)," Sepuluh Nopember Institute of Technology, 2018.

[8] D.-Z. Du, P. M. Pardalos, and W. Wu, "History of Optimization," *Encyclopedia of Optimization*. Springer,

pp. 1538–1542, 2009.

[9] A. Muklason, "Solver Penjadwal Ujian Otomatis Dengan Algoritma Maximal Clique dan Hyper-heuristics," *Semin. Nas. Teknol. Informasi, Komun. dan Ind. 9*, pp. 18–19, 2017.

[10] A. Muklason, "Hyper-heuristics and fairness in examination timetabling problems," 2017.

[11] M. Stølevik, T. Eric, and A. Riise, "A mathematical model for the nurse rostering problem," *SINTEF Tech. Rep. no. A19133*, no. August 2011, 2011.

[12] E. K. Burke and G. Kendall, *Search Methodologies*. 2014.

[13] E. Özcan, Y. Bykov, M. Birben, and E. K. Burke, "Examination timetabling using late acceptance hyper-heuristics," *2009 IEEE Congr. Evol. Comput. CEC 2009*, pp. 997–1004, 2009, doi: 10.1109/CEC.2009.4983054.

[14] B. Yuan, C. Zhang, and X. Shao, "A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints," *J. Intell. Manuf.*, vol. 26, no. 1, pp. 159–168, 2013, doi: 10.1007/s10845-013-0770-x.

BIODATA PENULIS

Penilis bernama Rizal Risnanda Hutama, lahir di Surabaya, 13 April 1998. Penulis telah menempuh pendidikan formal di SD Negeri Baratajaya Surabaya, SMP Negeri 6 Surabaya, SMA Negeri 5 Surabaya dan pendidikan sarjana di Departemen Sistem Informasi ITS dengan NRP 05211640000024 pada tahun 2016 dengan mengambil bidang minat Rekayasa Data dan Intelegensi Bisnis (RDIB). Selama berkuliah, penulis aktif dalam organisasi mahasiswa yaitu, Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi dan Komunikasi selama 2 periode 2017-2018 dan 2018-2019 menjadi staff Departemen External Affairs. Selain itu, penulis juga tergabung dalam kepantiaan yang dibawahi oleh himpunan yaitu Information System Expo! selama 2 periode yaitu pada 2017 dan 2018 sebagai staf kemanan dan perizinan. Selain aktif dalam organisasi dan kepanitian, penulis juga memiliki sertifikasi Key Application dari IC3 dan mengikuti pelatihan Oracle Java Fundamental dan SAP Practice. Penulis dapat dihubungi melalui email r130498@gmail.com.

Halaman ini sengaja dikosongkan.

LAMPIRAN A. KODE PROGRAM TUGAS AKHIR

Seluruh kode program yang digunakan dalam tugas akhir terlampir pada https://ciut.in/RizalLampiranA.

Halaman ini sengaja dikosongkan.

LAMPIRAN B. HASIL SOLUSI AWAL

Seluruh hasil pembuatan solusi awal terlampir pada *link* https://ciut.in/RizalLampiranB.

Halaman ini sengaja dikosongkan.

LAMPIRAN C. CONTOH HASIL OPTIMASI OPTUR4

Contoh hasil optimasi OpTur4 dengan nilai penalti terendah dapat dilihat pada *link* https://ciut.in/RizalLampiranC.

Halaman ini sengaja dikosongkan

LAMPIRAN D. CONTOH HASIL OPTIMASI OPTUR5

Contoh hasil optimasi OpTur5 dengan nilai penalti terendah dapat dilihat pada *link* https://ciut.in/RizalLampiranD.

Halaman ini sengaja dikosongkan

LAMPIRAN E. CONTOH HASIL OPTIMASI OPTUR7

Contoh hasil optimasi OpTur4 dengan nilai penalti terendah dapat dilihat pada *link* https://ciut.in/RizalLampiranD.

Halaman ini sengaja dikosongkan