TUGAS AKHIR – IS184853

OPTIMASI *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

NURSE ROSTERING PROBLEM (NRP) OPTIMIZATION USING TABU-SIMULATED ANNEALING BASED HYPER HEURISTICS ALGORITHM WITH BENCHMARK DATASET NORWEGIAN HOSPITALS

SISCA THREECYA AGATHA
NRP 05211640000037

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 *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

SISCA THREECYA AGATHA
NRP 05211640000037

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

FINAL PROJECT – IS184853

NURSE ROSTERING PROBLEM (NRP) OPTIMIZATION USING TABU-SIMULATED ANNEALING BASED HYPER HEURISTICS ALGORITHM WITH BENCHMARK DATASET NORWEGIAN HOSPITALS

SISCA THREECYA AGATHA
NRP 05211640000037

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

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

OPTIMASI *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

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:

SISCA THREECYA AGATHA
NRP. 05211640000037

Surabaya, 14 Agustus 2020

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Mudjahidin, S.T., M.T.
NIP. 19701010 200312 1 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

OPTIMASI *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

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:

SISCA THREECYA AGATHA
NRP. 05211640000037

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 *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

Nama Mahasiswa	**:**	**Sisca Threecya Agatha**
NRP	**:**	**05211640000037**
Departemen	**:**	**Sistem Informasi**
Dosen Pembimbing	**:**	**Ahmad Muklason, S.Kom., M.Sc., Ph.D**

ABSTRAK

Nurse Rostering atau penjadwalan perawat merupakan permasalahan penjadwalan yang kompleks, karena biasanya rumah sakit memiliki permintaan terkait personil yang bervariasi dari waktu ke waktu. Proses penjadwalan perawat membutuhkan ketelitian tinggi karena harus memperhatikan berbagai batasan, baik hard constraint maupun soft constraint. Berdasarkan optimasi kombinatorik permasalahan penjadwalan perawat merupakan NP-Hard yang artinya belum ada algoritma secara eksak yang mampu menyelesaikan permasalahan karena banyaknya kombinasi kemungkinan yang terbentuk, sehingga muncul algoritma approximate untuk mengatasi permasalahan. Tugas akhir ini membahas penjadwalan perawat secara otomatis untuk mengurangi nilai penalti akibat adanya pelanggaran soft constraint. Optimasi penjadwalan perawat menggunakan benchmark dataset Norwegian Hospitals. Algoritma yang akan digunakan yaitu Tabu Search dan Simulated Annealing berbasis Hyper-Heuristics. Tabu Search akan melakukan seleksi low-level

heuristics. Low level heuristics yang tidak menghasilkan solusi lebih baik akan disimpan di Tabu List agar tidak digunakan pada iterasi selanjutnya. Sedangkan Simulated Annealing akan menentukan acceptance criteria dari solusi serta melakukan proses evaluasi terhadap solusi tersebut. Pencarian solusi baru akan diterima sebagai solusi sementara jika solusi tersebut memiliki hasil yang lebih optimal dibandingkan solusi awal. Kedua algoritma yang digabungkan mampu melakukan diversifikasi sehingga menghindari adanya local optima. Hasil dari tugas akhir ini yaitu algoritma Tabu-Simulated Annealing berhasil membentuk solusi feasible untuk 3 instance dan menghasilkan luaran berupa jadwal kerja perawat dari dataset Norwegian Hospitals. Algoritma Tabu-Simulated Annealing juga mampu menurunkan nilai penalti hingga 80% dari penalti awal dan memberikan performa yang lebih baik jika dibandingkan dengan algoritma algoritma Hill Climbing, algoritma Tabu Search, dan algoritma Simulated Annealing.

Kata kunci: Nurse Rostering, Optimasi, Tabu Search, Simulated Annealing, Hyper-Heuristic

NURSE ROSTERING PROBLEM (NRP) OPTIMIZATION USING TABU-SIMULATED ANNEALING BASED HYPER HEURISTICS ALGORITHM WITH BENCHMARK DATASET NORWEGIAN HOSPITALS

Name	**: Sisca Threecya Agatha**
NRP	**: 05211640000037**
Department	**: Sistem Informasi**
Supervisor	**: Ahmad Muklason, S.Kom., M.Sc., Ph.D**

ABSTRACT

Nurse Rostering is a complex scheduling problem because hospitals usually have personnel-related requests that vary from time to time. Nurse scheduling process requires high accuracy because they have to pay attention to various constraints, both hard and soft constraints. Based on combinatoric optimization the nurse scheduling problem is NP-Hard which means there is no exact algorithm that is able to solve the problem because of the many possible combinations that are formed, so the approximate algorithm appears to overcome the problem. This final project discusses the scheduling of nurses automatically to reduce the penalty value due to soft constraint violations. Nurse scheduling optimization uses the Norwegian Hospitals benchmark dataset. The algorithm that will be used is Taboo and Simulated Annealing based on Hyper-Heuristics. Taboo Search will select the low-level heuristics. Low-level heuristics that do not produce better solutions will be stored in the Taboo List so that they are not used in the next iteration. While Simulated Annealing will

determine the acceptance criteria of the solution and evaluate the solution. The search for new solutions will be accepted as a temporary solution if the solution has more optimal results than the initial solution. The two algorithms combined are able to diversify so as to avoid local optima. The result of this final project is the Taboo Simulated Annealing algorithm that successfully formed a feasible solution for 3 instances and produced an output in the form of a nurse work schedule from the Norwegian Hospitals dataset. The Taboo-Simulated Annealing algorithm is also able to reduce the penalty value by up to 80% from the initial penalty and provide better performance when compared to the Hill Climbing algorithm, the Taboo Search algorithm, and the Simulated Annealing algorithm.

Keywords: Nurse Rostering, Optimization, Taboo Search, Simulated Annealing, Hyper-Heuristics

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Sisca Threecya Agatha
NRP	: 05211640000037
Tempat/Tanggal lahir	: Mojokerto, 5 Agustus 1998
Fakultas/Departemen	: FTEIC/Departemen Sistem Informasi
Nomor Telp/Hp/email	: sisca356@gmail.com

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

OPTIMASI *NURSE ROSTERING PROBLEM* (NRP) MENGGUNAKAN ALGORITMA *TABU-SIMULATED ANNEALING BASED HYPER-HEURISTICS* DENGAN *BENCHMARK DATASET NORWEGIAN HOSPITALS*

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, Juni 2020

Sisca Threecya Agatha

NRP. 05211640000037

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan petunjuk-Nya, akhirnya penulis dapat menyelesaikan laporan penelitian Tugas Akhir yang berjudul **Optimasi *Nurse Rostering Problem* (NRP) Menggunakan Algoritma *Tabu-Simulated Annealing Based Hyper-Heuristics* Dengan *Benchmark Dataset Norwegian Hospitals*** yang menjadi salah satu syarat kelulusan pada Departemen Sistem Infomasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Penyelesaian Tugas Akhir ini, diiringi dengan berbagai bantuan dari berbagai pihak yang selelu memberi dukungan, saran, dan doa sehingga penelitian berjalan dengan lancar. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua penulis, yang tidak pernah lelah memberikan dukungan, motivasi, doa serta nasihat yang tulus ikhlas.
2. Kakak-kakak dari penulis yang selalu memberikan doa, motivasi, memberi arahan, serta dukungan baik secara moril maupun materi.
3. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing yang telah meluangkan banyak waktu untuk sabar membimbing serta memberikan arahan penulis dalam pengerjaan tugas akhir ini.
4. Bapak Edwin Riksakomara, S.Kom., M.T dan Bapak Faizal Mahananto, S.Kom., M.Eng., Ph.D. selaku dosen penguji yang telah memberikan kritik serta saran yang membangun dalam proses pengerjaan tugas akhir
5. Teman grup *nurse rostering* Shafira, Dimas, Rizal yang menjadi teman berjuang, berkeluh kesah, dan tempat berdiskusi penulis.

6. Teman-teman Artemis yang memberikan semangat serta menemani penulis dari awal perkuliahan hingga menyelesaikan tugas akhir ini.
7. Teman-teman di Laboraotium SE, RDIB, MSI, dan ADDI yang telah menemani dan menjadi tempat diskusi penulis dalam pengerjaan tugas akhir saat di Lab.
8. Be yang selalu menemani dari awal hingga akhir, memberikan semangat, menjadi rekan 24/7 yang selalu ada, menghibur penulis, dan mendukung penulis dalam pengerjaan Tugas Akhir serta menjadi tempat penulis dalam berkeluh kesah.
9. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Terima kasih atas segala bentuk dukungan, bantuan, saran, dan doa yang sudah diberikan kepada penulis. Penulis menyadari bahawa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis menerima segala masukan, pertanyaan, dan kritikan yang membangun untuk menjadi saran pada penelitian selanjutnya. Semoga penelitian tugas akhir dapat memberikan manfaat bagi para pembaca.

Surabaya, Juni 2020

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN vii
LEMBAR PERSETUJUAN ix
ABSTRAK xi
ABSTRACT xiii
SURAT PERNYATAAN BEBAS PLAGIARISME xv
KATA PENGANTAR xvii
DAFTAR ISI xix
DAFTAR GAMBAR xxiv
DAFTAR TABEL xxvii
DAFTAR KODE xxix
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 3
1.3. Batasan Masalah 4
1.4. Tujuan Penelitian 4
1.5. Manfaat Penelitian 4
1.6. Relevansi 5
BAB II TINJAUAN PUSTAKA 7
2.1. Studi Literatur 7
2.2. Dasar Teori 12
2.2.1. Optimasi Kombinatorik 12
2.2.2. *Nurse Rostering Problem* 13

2.2.3. *Dataset Norwegian Hospitals*.....14
2.2.4. Hyper-Heuristics 20
2.2.5. Algoritma *Tabu Search* 21
2.2.6. Algoritma *Simulated Annealing* 22
BAB III METODOLOGI 25
3.1. Metodologi Tugas Akhir 25
3.1.1. Identifikasi Permasalahan.....26
3.1.2. Studi Literatur.....26
3.1.3. Pemahaman *Benchmark Dataset* 26
3.1.4. Implementasi Algoritma *Tabu-Simulated Annealing*.....26
3.1.5. Uji Coba Implementasi.....27
3.1.6. Analisis Performa Algoritma.....27
3.1.7. Analisis Hasil dan Kesimpulan.....27
3.1.8. Penyusunan Laporan Tugas Akhir.....27
3.2. Kesimpulan Metodologi 28
BAB IV PERANCANGAN 29
4.1. Pemahaman *Benchmark Dataset* 29
4.2. Pemahaman Model Matematis *Dataset* 30
4.2.1. Variabel Keputusan 31
4.2.2. Batasan 31
4.2.3. Fungsi Tujuan.....39
4.3. Pemodelan Algoritma *Tabu-Simulated Annealing* .39
4.4. Perencanaan Skenario Parameter.....42

4.5. Kesimpulan Perancangan 42
BAB V IMPLEMENTASI 43
5.1. Pembuatan Solusi Awal 43
5.1.1. Pembacaan File Input 43
5.1.2. Pembuatan Matriks *Employee – Day* 43
5.1.3. Pembuatan *Hard Constraint* 44
5.1.4. Perhitungan Nilai Penalti 47
5.1.5. Penyimpanan Solusi Awal 50
5.2. Optimasi Solusi 50
5.2.1. Implementasi *Low-Level Heuristics* 51
5.2.2. Implementasi Algoritma *Tabu-Simulated Annealing* 52
5.2.3. Penyimpanan Solusi Optimum 54
5.3. Kesimpulan Implementasi 55
BAB VI HASIL DAN PEMBAHASAN 57
6.1. Data Uji Coba 57
6.2. Lingkungan Uji Coba 57
6.3. Hasil Solusi Awal 58
6.4. Hasil Eksperimen Algoritma *Tabu-Simulated Annealing* 60
6.4.1. Hasil Eksperimen Skenario A 62
6.4.2. Hasil Eksperimen Skenario B 63
6.4.3. Hasil Eksperimen Skenario C 64
6.4.4. Hasil Eksperimen Skenario D 65
6.4.5. Hasil Eksperimen Skenario E 66

6.4.6. Hasil Eksperimen Skenario F 67
6.4.7. Hasil Eksperimen Skenario G.....68
6.4.8. Hasil Eksperimen Skenario H.....69
6.4.9. Hasil Eksperimen Skenario I 70
6.4.10. Hasil Eksperimen Skenario J 71
6.4.11. Hasil Eksperimen Skenario K.....72
6.4.12. Hasil Eksperimen Skenario L 73
6.4.13. Hasil Eksperimen Skenario M 74
6.5. Pemilihan Skenario Terbaik 75
6.6. Perbandingan Performa Algoritma 81
6.6.1. Perbandingan dengan Algoritma *Hill Climbing* 82
6.6.2. Perbandingan dengan Algoritma *Tabu Search* ... .86
6.6.3. Perbandingan dengan Algoritma *Simulated Annealing*.....91
6.6.4. Perbandingan Keseluruhan Algoritma.....95
6.6.5. Perbandingan dengan Hasil *Benchmark Dataset* 96
6.7. Hasil Solusi Optimum 98
BAB VII KESIMPULAN DAN SARAN 101
7.1. Kesimpulan.....101
7.2. Saran.....102
DAFTAR PUSTAKA.....105
Lampiran A Hasil Solusi Awal.....109

Lampiran B Hasil Eksperimen Skenario 112
Lampiran C Hasil Optimasi Solusi 114
Lampiran D Hasil Jadwal Kerja Perawat 117
Lampiran E Keseluruhan Hasil Penjadwalan 120
BIODATA PENULIS 121

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 1.1 Bidang Keilmuan Laboratorium RDIB 5
Gambar 2.1 Kerangka Hyper Heuristics 21
Gambar 3.1 Metodologi Tugas Akhir 25
Gambar 5.1 Ilustrasi low-level heuristics.....51
Gambar 6.1 Nilai Penalti Solusi Awal 59
Gambar 6.2 Hasil Eksperimen Skenario Optur 4.....76
Gambar 6.3 Persebaran Nilai Penalti Skenario Optur 4.....77
Gambar 6.4 Hasil Eksperimen Skenario Optur 5.....78
Gambar 6.5 Persebaran Nilai Penalti Skenario Optur 5.....79
Gambar 6.6 Hasil Eksperimen Skenario Optur 7.....80
Gambar 6.7 Persebaran Nilai Penalti Skenario Optur 7.....81
Gambar 6.8 Perbandingan Nilai Penalti Algoritma Tabu-Simulated Annealing dan Hill Climbing.....82
Gambar 6.9 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Hill Climbing pada Optur 4 83
Gambar 6.10 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Hill Climbing pada Optur 5 85
Gambar 6.11 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Hill Climbing pada Optur 7 86
Gambar 6.12 Perbandingan Nilai Penalti Algoritma Tabu-Simulated Annealing dan Tabu Search 87
Gambar 6.13 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Tabu Search pada Optur 4 88

Gambar 6.14 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Tabu Search pada Optur 5.....90
Gambar 6.15 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Tabu Search pada Optur 7.....91
Gambar 6.16 Perbandingan Nilai Penalti Algoritma Tabu-Simulated Annealing dan Simulated Annealing.....92
Gambar 6.17 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Simulated Annealing pada Optur 4 93
Gambar 6.18 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Simulated Annealing pada Optur 5 94
Gambar 6.19 Perbandingan Performa Algoritma Tabu-Simulated Annealing dan Simulated Annealing pada Optur 7 95
Gambar 6.20 Perbandingan Penalti Solusi Awal dan Penalti Optimasi Solusi 99

DAFTAR TABEL

Tabel 2.1 Penelitian Sebelumnya 7
Tabel 4.1 Informasi Umum Dataset 29
Tabel 4.2 Daftar Skenario Parameter 42
Tabel 6.1 Spesifikasi Perangkat Keras 57
Tabel 6.2 Spesifikasi Perangkat Lunak 57
Tabel 6.3 Hasil Solusi Awal 58
Tabel 6.4 Daftar Pelanggaran Soft Constraint Solusi Awal 60
Tabel 6.5 Daftar Parameter Skenario 61
Tabel 6.6 Parameter Skenario A 62
Tabel 6.7 Hasil Eksperimen Skenario A 62
Tabel 6.8 Parameter Skenario B 63
Tabel 6.9 Hasil Eksperimen Skenario B 64
Tabel 6.10 Parameter Skenario C 64
Tabel 6.11 Hasil Eksperimen Skenario C 65
Tabel 6.12 Parameter Skenario D 65
Tabel 6.13 Hasil Eksperimen Skenario D 66
Tabel 6.14 Parameter Skenario E 66
Tabel 6.15 Hasil Eksperimen Skenario E 67
Tabel 6.16 Parameter Skenario F 67
Tabel 6.17 Hasil Eksperimen Skenario F 68
Tabel 6.18 Parameter Skenario G 68
Tabel 6.19 Hasil Eksperimen Skenario G 69

Tabel 6.20 Parameter Skenario H.....69
Tabel 6.21 Hasil Eksperimen Skenario H 70
Tabel 6.22 Parameter Skenario I 70
Tabel 6.23 Hasil Eksperimen Skenario I.....71
Tabel 6.24 Parameter Skenario J 71
Tabel 6.25 Hasil Eksperimen Skenario J.....72
Tabel 6.26 Parameter Skenario K.....72
Tabel 6.27 Hasil Eksperimen Skenario K 73
Tabel 6.28 Parameter Skenario L 73
Tabel 6.29 Hasil Eksperimen Skenario L.....74
Tabel 6.30 Parameter Skenario M 74
Tabel 6.31 Hasil Eksperimen Skenario M.....75
Tabel 6.32 Perbandingan Keseluruhan Algoritma.....96
Tabel 6.33 Perbandingan dengan Hasil Benchmark Dataset..97
Tabel 6.34 Daftar Pelanggaran Soft Constraint Solusi Optimum.....99

DAFTAR KODE

Kode 4.1 Pseudocode algoritma Tabu Search.....40
Kode 4.2 Pseudocode algoritma Simulated Annealing.....41
Kode 5.1 Pseudocode algoritma Tabu-Simulated Annealing.53

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan gambaran umum tugas akhir yang meliputi latar belakang, perumusan masalah, batasan masalah, tujuan, manfaat dan relevansi tugas akhir dengan laboratorium penelitian penulis.

1.1. Latar Belakang

Rumah sakit adalah sebuah instansi yang bergerak di bidang pelayanan kesehatan. Tujuan utama rumah sakit yaitu memberikan pelayanan kepada masyarakat sesuai dengan standar medis dan fokus pada keselamatan pasien[1]. Salah satu bagian penting dalam layanan kesehatan adalah perawat. Perawat harus mengawasi pasien 24 jam penuh dan melakukan aktivitas lain seperti pelayanan administrasi yang cukup memeras tenaga fisik perawat. Berdasarkan studi yang dilakukan oleh Lockey et al mengatakan bahwa perawat yang bekerja 24 jam berturut-turut memiliki risiko tinggi dalam kesalahan pengambilan tindakan medis[2].

Menurut *International Council of Nursing*, pelayanan kesehatan 24 jam per hari membutuhkan pembagian *shift* untuk meringankan beban kerja perawat[3]. Namun terkadang *shift* kerja tetap memengaruhi kinerja dari perawat karena adanya ketidakseimbangan dalam pembagian *shift*. Lamanya waktu kerja menyebabkan perawat lelah lalu stres sehingga menurunkan performa mereka dalam bekerja. Hasil studi yang dilakukan oleh Callaghan menyatakan bahwa perawat Cina sebanyak 21% memiliki tingkat stres tinggi, 58% tingkat stres sedang, dan 11% tingkat stres rendah[4]. Penelitian tersebut menunjukkan bahwa terdapat hubungan antara *shift* kerja, kinerja serta kesehatan mental perawat. Peran perawat sebagai tim medis yang kontak langsung dengan pasien mengharuskan perawat mengutamakan kesehatan mereka. Solusi permasalahan tersebut adalah memberikan batasan jam kerja dan jadwal tugas yang sesuai preferensi perawat.

Nurse Rostering merupakan proses dari pembuatan jadwal kerja untuk perawat rumah sakit dengan cara memadukan *shift* karyawan dan mempertimbangkan ketrampilan, kompetensi, keadilan, dan peraturan dari rumah sakit[5]. Luaran dari penelitian *nurse rostering* adalah daftar jam kerja untuk perawat. Penjadwalan perawat adalah salah satu permasalahan yang kompleks karena biasanya rumah sakit memiliki permintaan terkait personil yang bervariasi dari waktu ke waktu. Ada batasan *hard constraint* dan *soft constraint* yang harus diperhatikan. Permasalahan penjadwalan perawat tidak hanya itu saja, namun juga melibatkan beberapa *stakeholder* lain seperti pemilik rumah sakit dan pasien yang harus dipertimbangkan pula. Fokus utama dari pemilik rumah sakit yaitu efisiensi pemanfaatan sumber daya dengan biaya minimum, sementara pasien pasti menginginkan waktu pelayanan yang singkat dan ditangani oleh perawat yang berkompeten[5].

Saat ini, penetapan jadwal perawat masih menggunakan cara manual, biasanya dilakukan oleh personil keperawatan yang berkualifikasi tinggi atau Kepala Bidang Keperawatan. Penjadwalan secara manual membutuhkan waktu yang lama karena harus memperhatikan jumlah *shift*, jumlah perawat dan lain-lain[5]. Hasilnya pun kurang efektif dan berakibat pada penurunan kualitas layanan. Adanya penjadwalan perawat secara otomatis membawa banyak keuntungan, salah satunya menambah performa penjadwalan[6]. Tidak hanya itu, dengan adanya penjadwalan perawat otomatis dapat mengurangi waktu penjadwalan manual secara drastis. Sisa waktu dari penjadwalan manual dapat digunakan untuk tugas klinis dan perawatan pasien.

Berdasarkan teori optimasi kombinatorik, penjadwalan merupakan permasalahan *NP-hard* yang artinya belum ada algoritma konvensional eksak yang dapat menyelesaikan permasalahan[7]. Dalam tugas akhir ini, penulis menggunakan pendekatan solusi heuristik dengan tujuan memenuhi persyaratan kerja perawat dari beberapa Rumah Sakit

Norwegia. Algoritma yang digunakan adalah *Tabu-Simulated Annealing Hyper-Heuristic*s.

Algoritma *Simulated Annealing* dipilih karena mampu menerima solusi yang tidak lebih baik (diversifikasi). Algortima ini menghasilkan solusi awal secara acak dan pada setiap iterasi akan ada sebuah solusi baru. Solusi tersebut bisa diterima apabila dapat meningkatkan fungsi tujuan. Jika dianalogikan algoritma ini seperti proses anil dalam metalurgi dimana suatu zat akan dipanaskan dengan temperatur yang sangat tinggi lalu didinginkan perlahan untuk menciptakan kristal superior[8]. Sementara algoritma *Tabu Search* dipilih karena mampu menghindar dari solusi *local optima* atau solusi akhir yang menjebak sehingga tidak dapat menerima solusi lain. Algoritma ini memiliki memori yang dinamakan *tabu list.* Setiap iterasi *Tabu Search,* maka *tabu list* akan menyimpan solusi yang dihasilkan namun *tabu list* akan menolak solusi pada daerah solusi yang sama[7]. Dengan menggunakan dua algoritma tersebut diharapkan mampu menghasilkan solusi yang lebih optimal untuk permasalahan penjadwalan perawat secara otomatis.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan pada sub bab sebelumnya, rumusan masalah yang diangkat dalam tugas akhir ini yaitu:

a. Bagaimana penerapan algoritma *Tabu-Simulated Annealing* berbasis *Hyper-Heuristic* yang diusulkan dapat menjadwalkan perawat pada *dataset Norwegian Hospitals*?

b. Bagaimana performa algoritma dalam menyelesaikan permasalahan penjadwalan perawat pada studi kasus *Norwegian Hospitals*?

1.3. Batasan Masalah

Batasan permasalahan dalam tugas akhir, antara lain:

a. Data yang digunakan dalam tugas akhir penjadwalan perawat adalah *Norwegian Hospitals Dataset*[1].

b. Algoritma penjadwalan perawat yang dibangun dan dikembangkan dengan menggunakan Bahasa Java.

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah dan batasan tugas akhir, maka tujuan yang ingin dicapai yaitu

1. Menerapkan dan melakukan eksperimen terhadap algoritma *Tabu-Simulated Annealing Hyper-Heuristics* untuk menyelesaikan permasalahan penjadwalan perawat pada *dataset Norwegian Hospitals*

2. Mendapatkan hasil penjadwalan perawat otomatis yang lebih optimal sesuai dengan batasan model yang dirumuskan serta membutuhkan waktu yang relatif cepat jika dibandingkan dengan penjadwalan secara manual.

1.5. Manfaat Penelitian

Manfaat yang diharapkan bisa tercapai pada tugas akhir ini antara lain:

a. Bagi Penulis
   Menambah ilmu dalam bidang optimasi khususnya mengenai kasus penjadwalan perawat menggunakan algoritma *Tabu-Simulated Annealing Hyper-Heuristics.*

b. Bagi Penelitian Selanjutnya
   Menjadi bahan referensi dan dapat dikembangkan menjadi penelitian lebih lanjut terhadap studi permasalahan penjadwalan perawat.

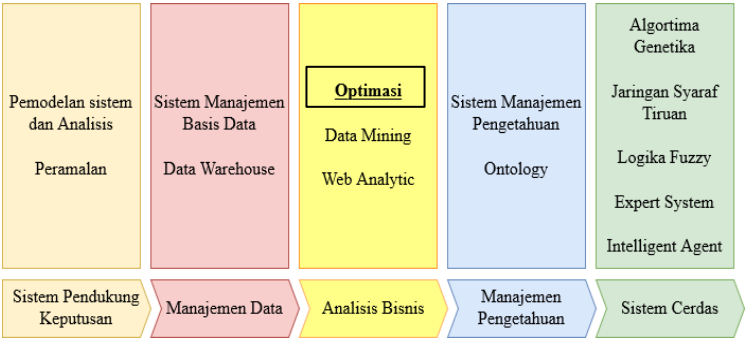
---

[1] https://www.sintef.no/en/digital/applied-mathematics/optimization/health-care-optimization/#*Benchmark*instances

c. Bagi Rumah Sakit di Indonesia
Hasil tugas akhir ini dapat dijadikan referensi untuk menyelesaikan studi kasus terkait permasalahan penjadwalan perawat di rumah sakit Indonesia dan disesuaikan dengan batasan pada masing-masing kasus.

1.6. Relevansi

Tugas akhir ini merupakan topik yang terkait dengan mata kuliah Optimasi Kombinatorial dan Heuristik (OKH) yang terdapat pada Laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB). Berikut merupakan relevansi dari topik yang diangkat oleh penulis dengan bidang keilmuan Laboratorium RDIB tertera pada Gambar 1.1:



Gambar 1.1 Bidang Keilmuan Laboratorium RDIB

Proses pengerjaan tugas akhir akan dijelaskan pada bab-bab selanjutnya. Bab 2 berisi studi literatur yang berhubungan dengan tugas akhir. Bab 3 berisi metodologi tugas akhir. Bab 4 akan dijelaskan perancangan tugas akhir lalu implementasi dari perancangan tersebut akan dijelaskan pada Bab 5. Hasil tugas akhir yang diperoleh akan dibahas pada Bab 6. Kemudian kesimpulan dari keseluruhan proses pengerjaan tugas akhir ditulis pada Bab 7 termasuk saran yang bisa diberikan penulis untuk penelitian selanjutnya.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Dalam bab ini dijelaskan penelitian terdahulu dan dasar teori yang digunakan sebagai acuan dalam mengerjakan tugas akhir serta mendukung latar belakang masalah yang ada pada Bab 1.

2.1. Studi Literatur

Pada sub bab ini dilakukan pencarian penelitian yang memiliki relevansi dan bisa dijadikan sebagai referensi pengerjaan tugas akhir. Studi literatur mengenai penelitian sebelumnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Sebelumnya

Penelitian 1	
Judul Penelitian	*A hybrid approach for solving real-world nurse rostering problems*[5]
Penulis Penelitian	Martin Stølevik, Tomas Eric Nordlander, Atle Riise, Helle Frøyseth
Tahun Penelitian	2011
Deskripsi Umum	Penelitian ini membahas tentang permasalahan penjadwalan perawat di beberapa rumah sakit di Norwegia. Studi kasus ini dikembangkan oleh perusahaan perangkat lunak Gatsoft AS. Penelitian ini menggunakan pendekatan metode *hybrid* dengan *framework Iterated Local Search*. Metode yang digabungkan adalah *Constraint Programming* untuk inisial solusi dan diversifikasi, kemudian *Variabel Neighborhood Descent* untuk pengembangan iterasi. *Output* dari penjadwalan ini adalah jam kerja perawat yang juga menyediakan gambaran pemanfaatan staf dan pengalokasian biaya.

	Tujuan utama penelitian dari sisi rumah sakit yaitu efisiensi pemanfaatan sumber daya dan meminimumkan biaya, sedangkan dari sisi karyawan mendapatkan jam kerja yang adil. Kemudian untuk pasien mendapatkan waktu tunggu yang singkat dan perawatan yang terbaik.
Hasil Penelitian	Hasil dari penelitian ini menunujukkan bahwa dengan menggunakan metode *hybrid* ditemukan solusi yang cukup optimal dan dapat memecahkan permasalahan dunia nyata dengan komputasi waktu yang wajar.
Keterkaitan Penelitian	Keterkaitan antara penelitian ini dengan tugas akhir adalah studi kasus yang digunakan sebagai subjek penelitian sama yaitu rumah sakit di Norwegia. Namun, metode yang digunakan pada tugas akhir adalah *Tabu-Simulated Annealing based Hyper Heuristics.*
Penelitian 2	
Judul Penelitian	*A tabu-search hyperheuristic for timetabling and rostering*[9]
Penulis Penelitian	E.K. Burke, G. Kendall, E.Soubeiga
Tahun Penelitian	2003
Deskripsi Umum	Penelitian ini membahas tentang pendekatan *Hyper Heuristics* yang diterapkan pada dua permasalahan penjadwalan yang berbeda. Pada penelitian *Hyper Heuristics* dilengkapi dengan *tabu list.* Penerapan *Tabu Search hyper heuristics* berhasil menghasilkan solusi yang layak untuk rumah sakit jika dibandingkan dengan algoritma genetika yang dirancang untuk masalah penjadwalan perawat. Namun, dalam hal biaya algoritma genetika mengungguli *hyper heuristic*.

	Tujuan dari penelitian adalah mengembangkan pendekatan yang kompetitif dan dapat digunakan pada penjadwalan yang berbeda masalah.
Hasil Penelitian	Hasil dari penelitian ini menunjukkan bahwa *tabu-search hyper heuristic* akan mencari permasalahan di ruang *low-level heuristic*. Metode *tabu-search hyper heuristic* akan memutuskan heuristic mana yang akan digunakan dengan menentukan parameter pada masalah yang diteliti. Perbandingan kasus dengan metode yang berbeda ternyata *tabu-search hyper heuristic* menghasilkan solusi yang lebih baik.
Keterkaitan Penelitian	Keterkaitan antara penelitian ini dengan tugas akhir adalah penggunaan metode yang sama yaitu *Tabu-Search Hyper Heuristic*. Metode tersebut digunakan untuk menyelesaikan permasalahan penjadwalan perawat.
Penelitian 3	
Judul Penelitian	*Medical Staff Scheduling Using Simulated Annealing*[8]
Penulis Penelitian	Ladislav Rosocha, Silvia Vernerová, Robert Verner
Tahun Penelitian	2014
Deskripsi Umum	Penelitian ini membahas tentang optimalisasi penjadwalan staf medis menggunakan Simulated Annealing. Tujuan dari penelitian ini adalah efisiensi staf medis untuk meningkatkan keseimbangan jam kerja dokter dan perawat serta memberikan perawatan kepada pasien dengan lebih baik. Penjadwalan staf medis yang dilakukan memperhatikan berbagai batasan seperti, *hard constraint*, peraturan hukum,

	meminimalkan pelanggaran *soft constraint*, kualitas pekerjaan hingga keseimbangan jam kerja staf. Sampel penjadwalan dilakukan pada 60 dokter dan perawat di Departemen Ginekologi.
Hasil Penelitian	Hasil dari penelitian menunjukkan bahwa algoritma *Simulated Annealing* yang dirancang mampu menjadwalkan staf medis sampai memenuhi *hard constraint* dan *soft constraint.* Selain itu, dengan menggunakan algoritma ini penjadwalan jauh lebih cepat daripada pembuatan jadwal standar.
Keterkaitan Penelitian	Keterkaitan antara penelitian ini dengan tugas akhir adalah penggunaan algoritma yang sama yaitu *Simulated Annealing* dan ruang lingkup yang hampir serupa yaitu optimasi penjadwalan staf medis di rumah sakit.
Penelitian 4	
Judul Penelitian	Optimasi Penjadwalan Mata Kuliah Otomatis Menggunakan Algoritma *Tabu-Simulated Annealing Hyper-Heuristics*[10]
Penulis Penelitian	Ahsanul Marom
Tahun Penelitian	2019
Deskripsi Umum	Penelitian ini membahas permasalahan penjadwalan mata kuliah yang sering kali terjadi di perguruan tinggi. Penjadwalan mata kuliah harus dilakukan setiap satu semester dan menyita banyak waktu untuk mendapatkan jadwal yang optimal sesuai batasan yang telah ditentukan. Selain itu, penjadwalan mata kuliah juga harus menyesuaikan sumber daya yang ada pada suatu semester. Algoritma yang digunakan pada penelitian ini adalah *Tabu-Simulated*

	Annealing Hyper-Heuristics dengan permasalahan *Socha Dataset.* Tujuan dari penelitian ini adalah melakukan penjadwalan mata kuliah secara otomatis dan mendapatkan hasil yang optimal dengan kelas dan data yang besar.
Hasil Penelitian	Hasil dari penelitian menyatakan bahwa dengan algoritma *Tabu-Simulated Annealing Hyper-Heuristics* dapat menyelesaikan permasalahan pada *Socha Dataset* yaitu mengurangi sekitar 40-80% skor penalti dari solusi awal.
Keterkaitan Penelitian	Keterkaitan antara penelitian ini dengan tugas akhir adalah penggunaan algoritma yang sama yaitu *Tabu-Simulated Annealing Hyper-Heuristic*. Namun, pada tugas akhir permasalahan yang akan diselesaikan yaitu optimasi penjadwalan perawat.
Penelitian 5	
Judul Penelitian	Solver Penjadwal Ujian Otomatis Dengan Algoritma *Maximal Clique* dan *Hyper-heuristics*[11]
Penulis Penelitian	Ahmad Muklason
Tahun Penelitian	2017
Deskripsi Umum	Penelitian ini membahas permasalahan penjadwalan ujian. Tujuan dari penjadwalan ujian adalah memastikan tidak ada satupun siswa atau mahasiswa yang harus menempuh dua ujian dengan waktu yang sama. Selain itu, penjadwalan ujian ini juga menentukan ruang ujian dan penjadwalan pengawas ujian. Masalah penjadwalan yang kompleks tentu akan menyita waktu apabila dikerjakan secara manual. Maka penelitian ini mengusulkan solver ujian otomatis

	dengan menggunakan konsep *maximal clique* pada teori graf digabung dengan metode *hyper heuristic*. Secara umum algoritma yang diusulkan terdiri dari dua fase. Fase pertama mengonstruksi solusi awal yang *feasible* dan fase kedua mengoptimalkan solusi awal dengan meminimumkan jumlah penalti akibat pelanggaran *soft constraint*.
Hasil Penelitian	Hasil dari penelitian ini adalah komputasi algoritma yang digunakan menunjukkan bahwa metode ini bisa menyelesaikan permasalahan penjadwalan dengan efektif jika dibandingkan dengan penelitian sebelumnya.
Keterkaitan Penelitian	Keterkaitan antara penelitian ini dengan tugas akhir adalah penggunaan metode yang sama yaitu *hyper heuristic* dengan studi kasus yang berbeda yaitu untuk optimasi penjadwalan perawat.

2.2. Dasar Teori

Pada sub bab ini dijabarkan mengenai dasar teori yang akan digunakan untuk mendukung penulisan tugas akhir.

2.2.1. Optimasi Kombinatorik

Optimasi merupakan salah satu teknik riset operasi atau manajemen sains yang mendukung pengambilan keputusan dengan cara menggambarkan sistem dengan pendekatan model matematis[12]. Optimasi memiliki fungsi tujuan yang akan dicapai baik dimaksimalkan atau diminimalkan berdasarkan batasan yang ada. Berdasarkan hasil akhirnya optimasi memiliki dua solusi, yaitu *Local Optimum Solution* dan *Global Optimum Solution*. Suatu permasalahan optimasi yang memiliki satu titik optimum di ruang pencarian dengan nilai paling baik dan menganggap titik lain di ruang pencarian lebih buruk maka

solusi tersebut dinamakan *Global Optimum Solution.* Apabila suatu permasalahan optimasi memiliki beberapa titik dengan nilai paling baik maka disebut *Local Optimum Solution*[7].

Algoritma *approximate* muncul untuk mengatasi permasalahan optimasi yang tidak bisa diselesaikan secara eksak. Algoritma ini mampu memberikan solusi yang relatif cepat dan layak diterima walaupun belum tentu optimal. Algoritma *approximate* dapat berupa *heuristic, meta heuristic,* maupun *hyper heuristic*[13].

2.2.2. *Nurse Rostering Problem*

Perawat merupakan tenaga kesehatan rumah sakit yang bertugas memberi pelayanan terhadap pasien. Peranan perawat sangat penting di sebuah rumah sakit karena perawat dibutuhkan selama 24 jam setiap hari[14]. Oleh karena itu, rumah sakit harus melakukan penjadwalan perawat dan pengaturan *shift* yang sesuai demi memenuhi permintaan layanan kesehatan. Baik buruknya penjadwalan perawat akan menentukan kualitas pelayanan dan kinerja rumah sakit[15].

Nurse Rostering Problem (NRP) merupakan permasalahan kombinatorial tentang penjadwalan perawat yang harus dilakukan oleh rumah sakit[16]. Penjadwalan perawat bukan persoalan yang mudah karena harus mempertimbangkan ketrampilan, kompetensi, keadilan, dan aturan. Selain itu dalam proses penjadwalan perawat memiliki batasan atau *constraint* yang harus diperhatikan. Batasan tersebut dibagi menjadi dua yaitu, *hard constraint* dan *soft constraint*. *Hard constraint* adalah kondisi dalam permasalahan yang harus dipenuhi sedangkan *soft constraint* adalah kondisi yang ingin dipenuhi tetapi tidak mutlak penting[7]. Beberapa contoh *constraint* yang umum pada kasus penelitian penjadwalan perawat, seperti:

- Lama bekerja dalam satu hari atau *shift*
- Jumlah minimum perawat dalam satu hari
- Ketrampilan dan kemampuan bekerja perawat

- Waktu istirahat dan hari libur
- Catatan kerja perawat
- Tidak boleh mendapat jadwal *shift* berurutan

2.2.3. *Dataset Norwegian Hospitals*

Dataset Norwegian Hospital merupakan *dataset* mengenai beberapa rumah sakit yang berada di Norwegia. *Dataset* tersebut berisi tentang data karyawan, data *shift*, dan data jumlah jam kerja karyawan serta data jumlah hari kerja karyawan. *Dataset* ini terdiri dari 7 kasus yang berbeda baik dari jumlah karyawan hingga jumlah *shift* yang disediakan. *Dataset Norwegian Hospitals* memiliki batasan baik *hard constraint* maupun *soft constraint*. Batasan – batasan tersebut akan diubah menjadi model matematis. Beberapa simbol dan terminologi yang terdapat dalam *dataset* yaitu[17]:

S	:	Himpunan dari jenis *shift*. Sebuah *shift* $s \in S$ adalah anggota dari satu dan hanya satu kategori *shift* $c \in C$.
s	:	Anggota dari himpunan *shift* S
C	:	Himpunan dari kategori *shift.*
c	:	Anggota dari himpunan kategori shift C. Tiga kategori *shift* yaitu "*Day*", "*Evening*" dan "*Night*".
D	:	Himpunan dari hari.
d	:	Anggota dari himpunan hari D dimulai dari hari ke-1 yaitu hari Senin.
E	:	Himpunan dari karyawan atau perawat.
e	:	Anggota dari himpunan karyawan E
I_d	:	Himpunan pasangan *shift* yang tidak kompatibel pada hari d dan d + 1 karena waktu antara *shift* terlalu pendek. I_d dihasilkan setelah menemukan semua

pasangan (s, s') dimana s merupakan *shift* pada hari d dan s' merupakan *shift* pada hari d+1.

U : Himpunan pola yang tidak diinginkan

u : Anggota dari himpunan pola yang tidak diinginkan

V : Himpunan pola yang diinginkan

v : Anggota dari himpunan pola yang diinginkan

W : Himpunan minggu

w : Anggota dari himpunan minggu dimulai dari minggu ke-0 dimana hari ke-1 pada minggu tersebut merupakan hari Senin dan hari ke-7 merupakan hari Minggu

x : Solusi yang berupa jadwal kerja perawat

p_m : Penalti dari pelanggaran soft constraint m, dimana m bernilai 1 sampai 9.

Lalu, parameter yang digunakan dalam model matematis antara lain:

N_c^{min} : Jumlah minimum pergantian berturut – turut dari kategori C untuk setiap perawat.

N_c^{max} : Jumlah maksimum *shift* berturut-turut dari kategori c untuk setiap perawat.

N^{min} : Jumlah minimum *shift* berturut-turut dari kategori apapun untuk perawat.

N^{max} : Jumlah maksimum *shift* berturut-turut dari kategori apapun untuk perawat.

N_{ec}^{max} : Jumlah maksimum *shift* yang harus dikerjakan perawat e pada kategori *shift c*.

N_{ec}^{min}	:	Jumlah minimum *shift* yang harus dikerjakan perawat e pada kategori *shift c*.
R_{ds}	:	Kebutuhan *shift* setiap harinya sesuai dengan rencana tenaga kerja
T_e	:	Jam kerja maksimum perawat per minggu
T_e^h	:	Jam kerja karyawan sesuai dengan kontrak kerja selama periode penjadwalan tertentu
T^f	:	Jam minimum waktu luang terus menerus setiap minggu
T_s	:	Durasi *shift* s
A_{sc}	:	1 jika *shift* S dalam kategori *shift* C, 0 sebaliknya.
B_{es}	:	1, jika karyawan e memiliki kompetensi untuk bekerja pada *shift* s, 0 sebaliknya
$f_{ew}(x)$	:	Panjang periode bebas untuk karyawan e dalam minggu w dalam solusi x.
$u_{ei}(x)$	:	Jumlah pola *shift* yang tidak diinginkan oleh karyawan e dari tipe i dalam solusi x.
$v_{ei}(x)$	:	Jumlah pola *shift* yang diinginkan oleh karyawan e dari tipe i dalam solusi x.

Hard constraint dan model matematis dari *dataset* meliputi:

1. Tiap perawat setiap harinya dialokasikan maksimal satu *shift*.

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

2. Kebutuhan *shift* harus terpenuhi pada setiap harinya

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

3. Total jam kerja setiap perawat tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat.

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

4. Perawat hanya dapat bekerja pada *shift* yang sesuai dengan kompetensi yang dimiliki

$$\sum_{s \in S} B_{es} q_{eds} = 1, \quad \forall e \in E, d \in D \quad (2.4)$$

5. Tidak boleh ada *shift* berurutan

$$q_{eds} + q_{e(d+1)s'} \leq 1, \forall e \in E, d \in D, (s, s') \in Id \quad (2.5)$$

6. Setiap minggu harus ada periode bebas kerja

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

7. Batas jam kerja setiap minggu tidak boleh dilanggar

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

Sedangkan *soft constraint* dan model matematis dari *dataset* antara lain:

1. Tidak boleh terlalu banyak hari kerja berurutan pada kategori *shift* yang sama

$$\mathrm{p1(x)} = \sum_{c \in C} \sum_{e \in E} \sum_{\mathrm{d}=1}^{|\mathrm{D}|-\mathrm{N_c^{max}}} \left(\prod_{\mathrm{d}'=\mathrm{d}}^{\mathrm{d}+\mathrm{N_c^{max}}} \mathrm{Yecd}' \right) \quad (2.8)$$

2. Tidak boleh terlalu banyak hari kerja berurutan

$$p2(x) = \sum_{e \in E} \sum_{d=1}^{|D|-N^{max}} \left(\prod_{d'=d}^{d+N^{max}} Zed' \right) \tag{2.9}$$

3. Tidak boleh terlalu sedikit hari kerja berurutan pada kategori *shift* yang sama

$$p3(X) = \sum_{c \in C} \sum_{e \in E} \sum_{d=1}^{|D|-N_c^{min}} \max\left(0, Y_{ec(d+1)} - Y_{ecd}\right) \left(N_c^{min} - \sum_{d'=d+1}^{d+N_c^{min}} \left(\prod_{d''=d+1}^{d+N_c^{min}} Yecd'' \right) \right) \tag{2.10}$$

4. Tidak boleh terlalu sedikit hari kerja berurutan

$$p4(x) = \sum_{e \in E} \sum_{d=1}^{|D|-N^{min}} \max\left(0, Z_{e(d+1)} - Z_{ed}\right) \left(N^{min} - \sum_{d'=d+1}^{d+N^{min}} \left(\prod_{d''=d+1}^{d+N^{min}} Zed'' \right) \right) \tag{2.11}$$

5. Tidak boleh terlalu banyak menyimpang dari jumlah minimum dan maksimum *shift* di setiap kategori *shift*

$$p5(x) = \sqrt{\sum_{c \in C} \sum_{e \in E} \left(max\left(0, N_{ec}^{min} - \sum_{d \in D} Yecd, \sum_{d \in D} Yecd - N_{ec}^{max}\right) \right)^2} \tag{2.12}$$

6. Tidak boleh menyimpang dari kontrak jam kerja perawat

$$p6(x) = \sqrt{\sum_{e \in E} \left(T_e^h - \sum_{d \in D} \sum_{s \in S} T_s X_{eds} \right)^2} \tag{2.13}$$

7. Pengelompokan waktu libur perawat

$$p7(x) = \sqrt{\sum_{e \in E} \left(\sum_{d=1}^{|D|-1} max\,(0, Z_{ed} - Z_{e(d+1)} \right)^2} \quad (2.14)$$

8. Tidak boleh terlalu sedikit pola *shift* yang diinginkan perawat

$$p8(x) = |E||D| - \sum_{e \in E, d \in D} v_{ei}(x) \quad (2.15)$$

9. Tidak boleh terlalu banyak pola *shift* yang tidak diinginkan perawat

$$p9(x) = \sum_{e \in E, i \in U} u_{ei}(x) \quad (2.16)$$

Dimana pada beberapa *soft constraint*,

- $Y_{ecd} = \sum_{s \in S} A_{sc} q_{eds}$ (2.17)

  Benilai 1 jika perawat e bekerja pada kategori *shift* c pada hari d, 0 sebaliknya.

- $Z_{ed} = \sum_{s \in S} q_{eds}$ (2.18)

  Bernilai 1 jika perawat e bekerja pada hari d, 0 sebaliknya.

Variabel keputusan dalam *dataset* adalah sebagai berikut:

q_{eds} : 1, jika karyawan e ditugaskan untuk pada hari d shift s, 0 sebaliknya.

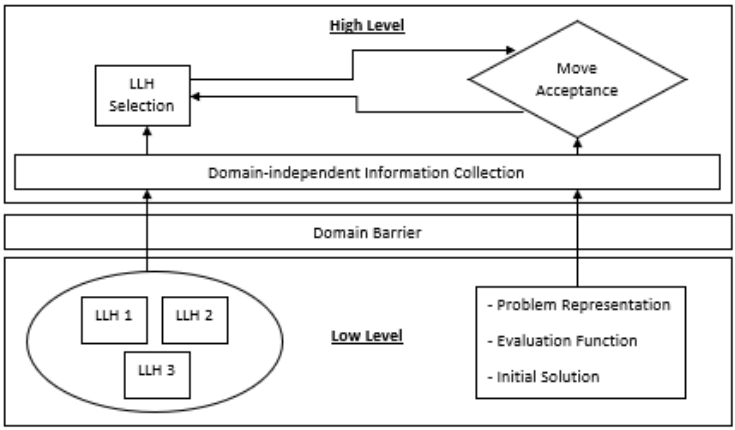
Fungsi tujuan yang ingin dicapai adalah meminimalkan total penalti. Fungsi tujuan permasalahan tugas akhir ditunjukkan pada persamaan 2.19. Dimana K_m adalah konstanta tetap yang mencerminkan bobot penalti dan P_m adalah nilai penalti yang diperoleh dari pelanggaran *soft constraint*.

$$minimize : f = \sum_{m=1}^{9} K_m p_m \tag{2.19}$$

2.2.4. Hyper-Heuristics

Heuristic merupakan sebuah metode pendekatan untuk mencari solusi yang baik atau mendekati optimal[7]. *Heuristic* berasal dari Bahasa Yunani (*heuriskein*) yang artinya sebuah seni untuk menemukan strategi dalam menyelesaikan permasalahan. Metode *hyper-heuristics* adalah pendekatan tingkat tinggi dalam menghasilkan heuristic untuk memecahan permasalahan kombinatorik yang rumit. Metode *hyper-heuristics* dikatakan sebagai metode yang *problem independent* artinya tidak bergantung pada permasalahan tertentu, jadi bisa diterapkan pada berbagai jenis permasalahanan.

Gambar 2.1 merupakan kerangka dari *Hyper-Heuristics*. Pada gambar, terdapat dua tingkatan proses pencarian pada *hyper-heuristics*, pertama metode *hyper-heuristics* melakukan pencarian solusi optimal diatas ruang pencarian berupa *low-level heuristics*[11]. *Low-level heuristics* (LLH) merupakan sebuah metode atau aturan sederhana untuk memilih komponen solusi. Kedua, metode *hyper-heuristics* melakukan pencarian berupa *high-level heuristics* (HLH). Pada HLH terdapat dua mekanisme, yaitu seleksi *low-level heuristics* dan penerimaan solusi atau *move acceptance.*



Gambar 2.1 Kerangka *Hyper Heuristics*

Metode *hyper-heuristics* membangun solusi secara bertahap dimulai dari inisial solusi, pemilihan low-level heuristics, hingga membentuk solusi secara lengkap[18]. Pencapaian *hyper-heuristics* ditentukan oleh seleksi low-level heuristics oleh HLH. Seleksi low-level heuristics dilakukan agar mendapatkan solusi yang sesuai, kemudian penerimaan solusi akan menentukan solusi hasil dari low-level heuristics tersebut diterima atau ditolak.

2.2.5. Algoritma *Tabu Search*

Tabu Search dimunculkan pertama pada tahun 1986 oleh Fred Glover dengan tujuan mengatasi masalah dari *hill climbing.* Glover mengatakan bahwa *Tabu Search* merupakan salah satu prosedur *heuristic* untuk penyelesaian masalah optimasi kombinatorial[19]. Kemampuan *Tabu Search* untuk menghasilkan solusi mendekati optimal dimanfaatkan untuk penyelesaian berbagai masalah dari berbagai bidang. Konsep dari *Tabu Search* yaitu algoritma yang memiliki tahapan agar menghasilkan fungsi tujuan yang paling optimum tanpa terjebak dalam solusi awal yang ditemukan pada saat pencarian[7].

Kata *tabu* atau *taboo* berasal dari Bahasa Tongan yang artinya hal yang tidak boleh disentuh karena kesakralannya. Jika dihubungkan dengan teori *Tabu Search* bahaya yang harus dihindari dari algoritma ini adalah rute perjalanan yang tidak seharusnya atau terjebak tanpa ada jalan keluar[19]. Algoritma *Tabu Search* mencegah proses pencarian untuk menerima ulang pada solusi yang sudah pernah dikunjungi. TS atau *Tabu Search* akan bergerak dari satu solusi ke solusi yang lain. Berbeda dari *hill climbing*, *Tabu Search* memperbaiki pencarian informasi akan solusi dengan memanfaatkan penggunaan struktur memori yaitu *adaptive memory*[12]. Struktur memori dari *tabu* disebut *tabu list*. *Tabu list* menyimpan solusi yang telah ditemukan pada iterasi sebelumnya. *Tabu list* akan menolak solusi pada daerah solusi yang sama.

Keunikan algoritma *Tabu Search* adalah adanya struktur memori yaitu *tabu list* yang membedakan algoritma ini dengan algoritma *Simulated Annealing*. Algoritma *Simulated Annealing* tidak memiliki struktur memori[19].

2.2.6. Algoritma *Simulated Annealing*

Simulated Annealing adalah varian dari teknik *heuristic search hill climbing*[20]. *Simulated Annealing* dianalogikan seperti proses *annealing* baja. Semakin rendah suhu baja maka atom dalam baja akan lebih terstruktur[21]. Secara umum algoritma ini dirancang untuk meminimalkan fungsi tujuan. Misalnya fungsi tujuan pada suatu permasalahan adalah biaya, maka semakin kecil biaya semakin baik kualitas dari algoritma[20]. Metode ini memiliki solusi awal yang dihasilkan secara acak kemudian dilakukan proses evaluasi. Apabila solusi baru yang ditemukan lebih optimal dibandingkan solusi awal, maka solusi tersebut akan diterima sebagai solusi terbaik sementara.

Metode *Simulated Annealing* memiliki ketergantungan pada parameter tertentu untuk setiap iterasinya, antara lain[10]:

1. Suhu awal dan akhir, semakin lebar selisih suhu awal dan akhir maka semakin luas area solusi.
2. Jumlah iterasi setiap penurunan suhu diturunkan secara berkala setelah mencapai kondisi pada suhu tertentu.
3. *Cooling rate*, yaitu nilai koefisien *alpha* yang berfungsi untuk menurunkan suhu

Pada *Simulated Annealing* terdapat pengembangan algoritma berupa *reheating* dimana suhu akan dipanaskan pada saat mencapai iterasi tertentu. Tujuannya adalah untuk menambah peluang pencarian pada area solusi yang semakin sempit karena penurunan suhu. Melakukan *reheating* sama dengan menambah luas area solusi karena suhu yang dinaikkan. Besar nilai koefisien beta suhu *reheating* ditentukan sebelum proses penerapan *Simulated Annealing*.

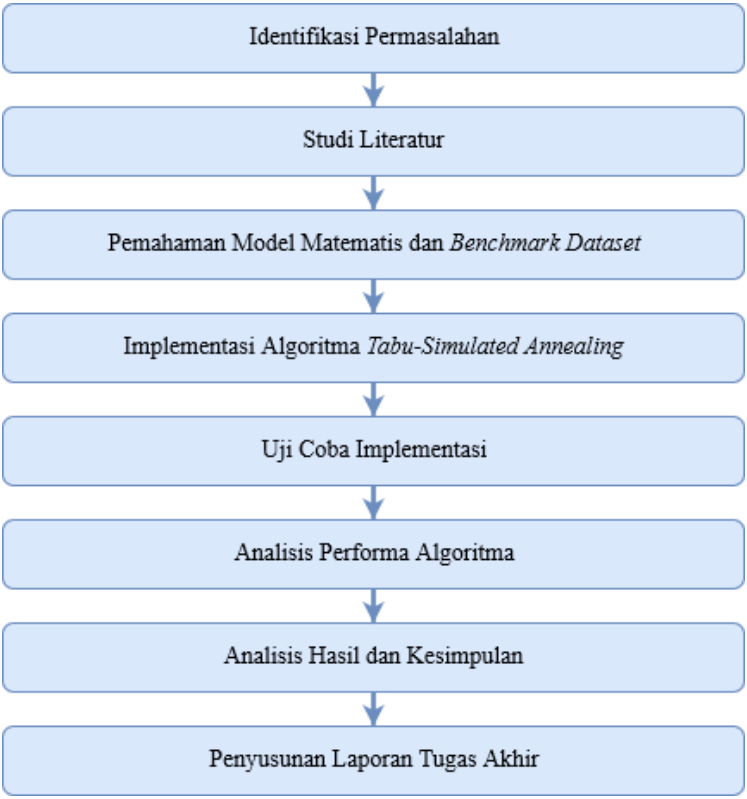
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Pada bab ini dijelaskan metodologi yang akan dilakukan dalam melaksanakan tugas akhir. Metodologi ini akan digunakan sebagai pedoman untuk mengerjakan tugas akhir agar terarah dan sistematis.

3.1. Metodologi Tugas Akhir

Diagram metodologi dari tugas akhir terdapat pada Gambar 3.1.



Gambar 3.1 Metodologi Tugas Akhir

Berdasarkan Gambar 3.1, langkah-langkah penelitian adalah sebagai berikut:

3.1.1. Identifikasi Permasalahan

Identifikasi permasalahan dilakukan dengan menganalisis studi kasus terkait topik yang akan dibahas. Hasil yang didapatkan dari tahap ini adalah permasalahan yang akan menjadi topik tugas akhir.

3.1.2. Studi Literatur

Studi literatur dilakukan dengan mempelajari penelitian-penelitian terkait yang telah dilakukan sebelumnya. Penelitian tersebut dapat ditemukan dari berbagai sumber seperti *paper*, jurnal, buku, dan lain-lain. Hasil yang didapatkan dari studi literatur adalah algoritma yang digunakan untuk memecahkan permasalahan yang ditentukan pada tahap sebelumya.

3.1.3. Pemahaman *Benchmark Dataset*

Data yang dipilih dan digunakan dalam penelitian berupa *benchmark data*, yaitu *Norwegian Hospitals dataset*. Setelah itu, dilakukan pemahaman terhadap *dataset*, termasuk model matematis dari *dataset*. Pemahaman dilakukan dengan studi literatur mengenai *Norwegian Hospitals dataset*.

3.1.4. Implementasi Algoritma *Tabu-Simulated Annealing*

Model matematis selanjutnya dikonversi ke dalam struktur data bahasa pemrograman. Implementasi algoritma *Tabu-Simulated Annealing* dilakukan dengan menggunakan *input* berupa *Norwegian Hospitals dataset*. Program akan membaca *file data* yang kemudian menentukan solusi awal. Penggunaan algoritma *Simulated Annealing* dan *Tabu Search* memungkinkan diversifikasi, yaitu dapat menerima solusi yang tidak lebih baik. Penggabungan kedua algoritma diharapkan dapat menjadikan solusi lebih optimal. Hasil yang didapatkan dari tahapan ini adalah struktur data bahasa pemrograman dari model matematis *Norwegian Hospitals Dataset.*

3.1.5. Uji Coba Implementasi

Uji coba dilakukan dengan memvalidasi algoritma *Tabu-Simulated Annealing* yang telah diimplementasikan untuk *dataset*. Apabila algoritma yang telah diiimplementasikan tersebut tidak sesuai dengan permasalahan, akan dilakukan evaluasi dan perbaikan terhadap algortima. Hasil dari uji coba yaitu struktur data yang lebih baik.

3.1.6. Analisis Performa Algoritma

Algoritma *Tabu Search* dan *Simulated Annealing* yang telah diuji coba akan dilakukan analisis terhadap algoritma dengan tujuan mendapatkan hasil yang paling optimum dari penyelesaian permasalahan. Apabila algoritma telah menunjukkan performa yang optimal dan hasil yang layak maka bisa beralih ke tahapan selanjutnya, namun apabila performa algoritma masih belum optimal dan hasil yang didapatkan tidak layak maka dilakukan pemeriksaan ulang terhadap model algoritma yang dibuat. Analisis performa algoritma juga dilakukan dengan memilih parameter yang bisa menghasilkan solusi optimal.

3.1.7. Analisis Hasil dan Kesimpulan

Analisis hasil dilakukan dengan menghitung hasil keluaran dari setiap iterasi maupun data masukan serta membandingkan hasil dengan penelitian yang menggunakan metode lainnya. Analisis juga memperhatikan *hard constraint* dan *soft constraint* apa saja yang telah dipenuhi lalu menarik kesimpulan atas hasil yang didapatkan. Tidak lupa pada setiap penelitian akan menghasilkan pembelajaran yang mungkin bisa digunakan untuk penelitian selanjutnya.

3.1.8. Penyusunan Laporan Tugas Akhir

Penyusunan laporan dilakukan dengan cara mengumpulkan semua dokumentasi masukan, proses, dan keluaran dari seluruh penelitian tugas akhir. Hasilnya berupa buku laporan tugas akhir.

3.2. Kesimpulan Metodologi

Tugas akhir dikerjakan sesuai dengan alur yang telah disusun pada metodologi tugas akhir, dimulai dari identifikasi permasalahan, studi literatur hingga penyusunan laporan tugas akhir. Selanjutnya, pada Bab 4 akan dilakukan perancangan tugas akhir yang sesuai dengan metodologi yang telah dibentuk dengan tujuan menciptakan kerangka sebelum implementasi dan penyelesaian masalah.

BAB IV
PERANCANGAN

Pada bab ini berisi tentang penjelasan persiapan perancangan terkait dengan *dataset* yang digunakan serta konversi dari model matematis ke struktur data untuk selanjutnya dilakukan implementasi algoritma.

4.1. Pemahaman *Benchmark Dataset*

Pada tugas akhir, *dataset* yang digunakan adalah *benchmark dataset Norwegian Hospitals* yang diperoleh dari situs resmi SINTEF. *Dataset Norwegian hospitals* merupakan *dataset* proyek *Research and Development* untuk Gatsoft AS yaitu sebuah lembaga pengembang perangkat lunak manajemen personalia yang saat ini melayani 80% rumah sakit di Norwegia[5].

Dataset yang dikumpulkan terdiri dari 7 kasus rumah sakit yang ada di Norwegia. Untuk setiap rumah sakit memiliki jumlah karyawan, jumlah jam kerja, jumlah hari kerja dan jumlah *shift* yang berbeda. Berikut merupakan informasi secara umum mengenai *dataset Norwegian Hospitals* yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Informasi Umum *Dataset*

Rumah Sakit	**Jumlah Perawat**	**Jumlah Jam Kerja**	**Jumlah Hari Kerja**	**Jumlah *Shift***
OpTur 1	51	19170	84	9
OpTur 2	83	12819	42	9
OpTur 3	29	4159.5	42	8
OpTur 4	30	17352	168	8
OpTur 5	20	2280	28	9
OpTur 6	54	18978	84	5
OpTur 7	15	2209.5	42	6

Masing-masing unit rumah sakit disimpan dalam *file* dengan ekstensi xls atau *excel* mulai dari OpTur1 hingga OpTur7. Pada

setiap *file* OpTur terdiri dari beberapa *sheet*, yaitu *employee*, *shift*, *manpower plan*, *constraint*, dan *pattern*. Penjelasan isi dari *dataset* tersebut meliputi:

1. *Sheet Employee* berisi tentang data perawat yang terdiri dari Id perawat, rata-rata jumlah jam kerja setiap perawat tiap minggu, akhir pekan yang diperbolehkan untuk bekerja, serta kompetensi yang dimiliki perawat.
2. *Sheet Shift* berisi tentang data *shift* yang terdiri dari jumlah *shift* yang disediakan oleh rumah sakit, id dan nama *shift*, kategori *shift* beserta durasi setiap harinya, serta waktu mulai dan berakhirnya *shift*.
3. *Sheet Manpower plan* berisi tentang rencana kerja rumah sakit pada periode tertentu yang terdiri dari panjang minggu yang akan dijadwalkan dan kebutuhan perawat tiap harinya berdasarkan *shift*.
4. *Sheet Constraint* berisi tentang data *hard constraint* dan *soft constraint* atau peraturan-peraturan yang diterapkan pada setiap rumah sakit.
5. *Sheet Pattern* berisi tentang pola *shift* baik yang diinginkan maupun tidak diinginkan oleh perawat. Pada sheet ini terdiri dari id perawat, pola *shift*.

Setiap rumah sakit memiliki detil permasalahan dan kompleksitas yang berbeda dipengaruhi dari jumlah perawat, lama periode penjadwalan, hingga *shift* dan jam kerja yang dibutuhkan.

4.2. Pemahaman Model Matematis *Dataset*

Permasalahan penjadwalan perawat pada tugas akhir dimodelkan dalam sebuah model matematis yang nantinya akan dikonversi ke dalam struktur data. Model matematis dari permasalahan terdiri dari 3 hal, yaitu variabel keputusan, batasan, dan fungsi tujuan. Sebelumnya, pada Subbab 2 telah ditulis tentang model matematis dari *dataset*. Subbab ini akan menjelaskan lebih detail tentang pemodelan tersebut.

4.2.1. Variabel Keputusan

Variabel keputusan merupakan variabel yang nilainya mempengaruhi fungsi tujuan yang ingin dicapai. Pada permasalahan *nurse rostering* tugas akhir ini, terdapat variabel keputusan yaitu:

$$q_{eds} = \begin{cases} 1, \text{jika perawat e bekerja pada hari d } shift \text{ s} \\ 0 \end{cases}$$

Dimana,
e = {e_1, e_2, e_3, …, $e_{|E|}$}, $e \in E$
d = {d_1, d_2, d_3, …, $d_{|D|}$}, $d \in D$
s = {s_1, s_2, s_2, …, $s_{|S|}$}, $s \in S$

q_{eds} merupakan variabel biner dimana bernilai 1 jika perawat e bekerja pada hari d dan pada *shift* s, 0 sebaliknya. Perawat e yang disebutkan pada variabel keputusan berasal dari himpunan perawat E yang berbeda tiap rumah sakit. Lalu, *Shift* s berasal dari himpunan jenis *shift* S dimana pada *dataset* tiap rumah sakit memiliki jumlah *shift* yang berbeda-beda. *Shift* pada *dataset* ada tiga kategori yaitu *day* untuk *shift* pagi, evening untuk *shift* siang dan night untuk *shift* malam. Dan hari d merupakan hari yang berasal dari himpunan hari D pada periode penjadwalan.

4.2.2. Batasan

Batasan berguna untuk membatasi variabel keputusan yang ada. Terdapat dua batasan dalam tugas akhir ini, *hard constraint* dan *soft constraint* seperti yang sudah dipaparkan pada subbab 2. Pada *dataset Norwegian Hospitals* ada 7 *hard constraint* dan 9 *hard constraint*. Masing-masing batasan ini diterapkan untuk setiap rumah sakit.

Berikut merupakan penjelasan dari masing masing batasan tersebut, yaitu:

1. *Hard Constraint* 1

   Satu perawat dialokasikan setiap hari maksimal satu *shift*. Batasan ini akan mengatur bahwa seorang perawat hanya memiliki dua pilihan, yaitu bekerja pada hari d *shift* s atau sebaliknya. Model matematis dari *hard constraint* 1 ditunjukkan pada persamaan 2.20.

$$\sum_{s \in S} q_{eds} \leq 1, \qquad \forall e \in E, d \in D \tag{2.20}$$

   q_{eds} merupakan variabel biner dimana bernilai 1 jika perawat e bekerja pada hari d dan pada *shift* s, 0 sebaliknya.

2. *Hard Constraint* 2

   Kebutuhan *shift* harus dapat terpenuhi setiap harinya. Batasan ini akan disesuaikan dengan kebutuhan perawat yang terdapat dalam sheet manpower plan atau rencana tenaga kerja tiap rumah sakit. Apabila rencana tenaga kerja tersebut telah sama dengan perawat yang dialokasikan setiap harinya maka batasan ini telah terpenuhi. Model matematis dari *hard constraint* 2 ditunjukkan pada persamaan 2.21.

$$\sum_{e \in E} q_{eds} = R_{ds}, \qquad \forall d \in D, s \in S \tag{2.21}$$

   P_{ds} merupakan rencana tenaga kerja yang mencakup jumlah *shift* s yang diperlukan setiap harinya.

3. *Hard Constraint* 3

   Total jam kerja setiap perawat tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat. Kontrak kerja perawat diatur pada masing-masing *dataset* yang dituliskan pada *sheet employee* kolom *workweek*. Setiap perawat telah memiliki rata-rata jam kerja yang dituliskan

dalam tiap *dataset*. Besar penyimpangan jam kerja yang diperbolehkan adalah ± 2% dari kontrak kerja. Model matematis dari *hard constraint* 3 ditunjukkan pada persamaan 2.22.

$$(1 - 0.02)\mathrm{T}_{\mathrm{e}}^{\mathrm{h}} \leq \sum_{\mathrm{d} \in D, s \in S} \mathrm{T}_{\mathrm{s}} q_{\mathrm{eds}} \leq (1 + 0.02)\mathrm{T}_{\mathrm{e}}^{\mathrm{h}}, \forall \mathrm{e} \in \mathrm{E} \quad (2.22)$$

T_e^h menyatakan kontrak kerja perawat dan T_s merupakan durasi *shift* perawat.

4. *Hard Constraint* 4

Perawat hanya dapat bekerja pada *shift* yang sesuai dengan kompetensi yang dimiliki. Untuk memenuhi batasan ini maka perawat yang dialokasikan hanyalah perawat yang memiliki kompetensi. Model matematis dari *hard constraint* 4 ditunjukkan pada persamaan 2.23.

$$\sum_{s \in S} \mathrm{B}_{\mathrm{es}} q_{\mathrm{eds}} = 1, \quad \forall \mathrm{e} \in \mathrm{E}, \mathrm{d} \in \mathrm{D} \quad (2.23)$$

Parameter B_{es} akan bernilai 1 jika perawat yang bekerja pada *shift* s memiliki kompetensi dan menunjukkan bahwa batasan ini dapat terpenuhi.

5. *Hard Constraint* 5

Tidak boleh ada *shift* berurutan untuk setiap perawat. Artinya batasan ini mengatur tentang pasangan *shift* yang tidak boleh diisi oleh perawat yang sama. Perawat boleh bekerja pada hari selanjutnya apabila telah melebihi batas waktu libur yang telah ditentukan. Model matematis dari *hard constraint* 5 ditunjukkan pada persamaan 2.24.

$$q_{\mathrm{eds}} + q_{\mathrm{e}(\mathrm{d}+1)\mathrm{s}'} \leq 1, \quad \forall \mathrm{e} \in \mathrm{E}, \mathrm{d} \in \mathrm{D}, (\mathrm{s}, \mathrm{s}') \in \mathrm{I}\mathrm{d} \quad (2.24)$$

Simbol I_d menunjukkan pasangan *shift* berpasangan yang dilarang. Pasangan *shift* yang dilarang antara lain:

- *Shift* night dan *shift evening*
- *Shift* evening dan *shift day*
- *Shift day* dan *shift night*

Apabila antara dua *shift* yang akan dijadwalkan merupakan pasangan *shift* seperti diatas dan memiliki waktu libur kurang dari batas yang ditentukan maka kedua *shift* tersebut akan masuk ke dalam himpunan I_d dan tidak boleh dijadwalkan secara berurutan untuk setiap perawat.

6. *Hard Constraint* 6

Setiap minggu harus ada periode bebas kerja. Pada setiap *dataset* akan diatur batas minimal periode libur perawat tiap minggunya. Setiap perawat harus memiliki total waktu libur yang melebihi batas tersebut. Model matematis dari *hard constraint* 6 ditunjukkan pada persamaan 2.25.

$$f_{ew}(\mathbf{x}) > T^f, \qquad \forall e \in E, w \in W \tag{2.25}$$

Parameter $f_{ew}(\boldsymbol{x})$ menunjukkan durasi terpanjang dari waktu libur perawat pada minggu w. Sedangkan T^f merupakan batas minimal waktu libur yang ditetapkan pada masing-masing *dataset*.

7. *Hard Constraint* 7

Batas jam kerja setiap minggu tidak boleh dilanggar. Batasan ini mengatur tentang total jam kerja perawat tiap minggunya dimana setiap perawat yang bekerja tidak boleh melebihi batas jam kerja yang ditentukan. Model matematis dari *hard constraint* 7 ditunjukkan pada persamaan 2.26.

$$\sum_{d=7w+1}^{7w+7} \sum_{s \in S} T_s q_{eds} \leq T_e \ , \forall e \in E, w \in W \tag{2.26}$$

T_e merupakan batas jam kerja maksimal setiap perawat per minggunya.

Selain *hard constraint*, terdapat pula *soft constraint* yang nantinya akan digunakan untuk menghitung nilai penalti. Berikut merupakan penjelasan tentang *soft constraint* yaitu:

1. *Soft Constraint* 1

   Tidak boleh terlalu banyak hari kerja berurutan pada kategori *shift* yang sama. Pada batasan ini akan diatur tentang maksimal kategori *shift* yang diperbolehkan untuk dialokasikan pada perawat secara berturut-turut. Setiap *dataset* rumah sakit sudah mengatur batas maksimal tersebut mulai dari kategori *shift day*, *evening*, maupun *night*. Model matematis dari *soft constraint* 1 ditunjukkan pada persamaan 2.27.

$$\mathrm{p1(x)} = \sum_{c \in C} \sum_{e \in E} \sum_{\mathrm{d}=1}^{|\mathrm{D}|-\mathrm{N_c^{max}}} \left(\prod_{\mathrm{d}'=\mathrm{d}}^{\mathrm{d}+\mathrm{N_c^{max}}} \mathrm{Yecd}' \right) \tag{2.27}$$

   Y_{ecd} benilai 1 jika perawat e bekerja pada kategori *shift* c pada hari d, 0 sebaliknya.

2. *Soft Constraint* 2

   Tidak boleh terlalu banyak hari kerja berurutan. Batasan ini akan mengatur tentang maksimal hari kerja yang boleh dialokasikan untuk setiap perawat. Tujuannya supaya setiap perawat tidak mendapatkan hari kerja yang terlalu banyak. Model matematis dari *soft constraint* 2 ditunjukkan pada persamaan 2.28.

$$\mathrm{p2(x)} = \sum_{e \in E} \sum_{\mathrm{d}=1}^{|\mathrm{D}|-\mathrm{N^{max}}} \left(\prod_{\mathrm{d}'=\mathrm{d}}^{\mathrm{d}+\mathrm{N^{max}}} \mathrm{Zed}' \right) \tag{2.28}$$

   Z_{ed} bernilai 1 jika perawat e bekerja pada hari d, 0 sebaliknya.

3. *Soft Constraint* 3

   Tidak boleh terlalu sedikit hari kerja berurutan pada kategori *shift* yang sama. *Soft constraint* ini mengatur batas minimal untuk setiap perawat pada bekerja pada kategori *shift* yang sama secara berturut-turut. Batasan ini digunakan untuk menghindari *shift* malam tunggal. Model matematis dari *soft constraint* 3 ditunjukkan pada persamaan 2.29.

$$p3(X) = \sum_{c \in C} \sum_{e \in E} \sum_{d=1}^{|D|-N_c^{min}} \max\left(0, Y_{ec(d+1)} - Y_{ecd}\right) \left(N_c^{min} - \sum_{d'=d+1}^{d+N_c^{min}} \left(\prod_{d''=d+1}^{d+N_c^{min}} Yecd'' \right) \right) \tag{2.29}$$

   N_c^{min} adalah batas minimum *shift* berturut-turut dari kategori c untuk setiap perawat.

4. *Soft Constraint* 4

   Tidak boleh terlalu sedikit hari kerja berurutan. Batasan ini kebalikan dari *soft constraint* 2 yaitu untuk mengatur minimal hari kerja yang dialokasikan untuk setiap perawat. Tujuannya untuk menghindari masa kerja hanya satu atau dua hari. Model matematis dari *soft constraint* 4 ditunjukkan pada persamaan 2.30.

$$p4(x) = \sum_{e \in E} \sum_{d=1}^{|D|-N^{min}} \max\left(0, Z_{e(d+1)} - Z_{ed}\right) \left(N^{min} - \sum_{d'=d+1}^{d+N^{min}} \left(\prod_{d''=d+1}^{d+N^{min}} Zed'' \right) \right) \tag{2.30}$$

   N^{min} adalah batas minimum *shift* berturut-turut dari kategori apapun untuk perawat.

5. *Soft Constraint* 5

   Tidak boleh terlalu banyak menyimpang dari jumlah minimum dan maksimum *shift* di setiap kategori *shift*. Setiap rumah sakit telah mengatur batas minimal dan

maksimal jumlah kategori *shift* sama yang harus dialokasikan kepada perawat. Tujuannya adalah untuk membuat jadwal yang lebih merata dan beban kerja dari perawat tidak terlalu banyak maupun tidak terlalu sedikit. Model matematis dari *soft constraint* 5 ditunjukkan pada persamaan 2.31.

$$p5(x) = \sqrt{\sum_{c \in C} \sum_{e \in E} \left(max\left(0, N_{ec}^{min} - \sum_{d \in D} Yecd, \sum_{d \in D} Yecd - N_{ec}^{max}\right)\right)^2} \tag{2.31}$$

N_{ec}^{max} adalah batas maksimum *shift* kategori c yang boleh dikerjakan oleh e perawat, sedangkan N_{ec}^{min} adalah batas minimum dhift kategori c yang harus dipenuhi oleh perawat e.

6. *Soft Constraint* 6

Tidak boleh menyimpang dari kontrak jam kerja perawat. Setiap perawat memiliki kontrak kerja masing-masing yang sudah dituliskan dalam *sheet employee* di kolom rata-rata jam kerja perawat. Model matematis dari *soft constraint* 6 ditunjukkan pada persamaan 2.32.

$$p6(x) = \sqrt{\sum_{e \in E} \left(T_e^w - \sum_{d \in D} \sum_{s \in S} T_s X_{eds} \right)^2} \tag{2.32}$$

T_e^w merupakan kontrak kerja perawat setiap minggu. Jadwal yang dibentuk akan mendapatkan pelanggaran apabila jam kerja perawat berselisih dengan kontrak kerja yang telah ditetapkan.

7. *Soft Constraint* 7

Pengelompokan waktu libur perawat. Batasan ini mengontrol waktu libur perawat dimana waktu libur yang

berurutan akan lebih baik dibandingkan dengan waktu libur yang terlalu acak. Misalnya seorang perawat e mendapatkan hari libur pada Senin dan Selasa akan lebih baik jika dibandingkan dengan hari libur Senin dan Kamis. Model matematis dari *soft constraint* 7 ditunjukkan pada persamaan 2.33.

$$p7(x) = \sqrt{\sum_{e \in E} \left(\sum_{d=1}^{|D|-1} max\ (0, Z_{ed} - Z_{e(d+1)} \right)^2} \qquad (2.33)$$

8. *Soft Constraint* 8

   Tidak boleh terlalu sedikit pola *shift* yang diinginkan perawat. Pola *shift* yang diinginkan oleh perawat terdapat pada *sheet pattern*. Semakin banyak pola *shift* yang diinginkan dapat dijadwalkan maka nilai pelanggaran akan semakin kecil. Model matematis dari *soft constraint* 8 ditunjukkan pada persamaan 2.34.

$$p8(x) = |E||D| - \sum_{e \in E, d \in D} v_{ei}(x) \qquad (2.34)$$

   $v_{ei}(x)$ bernilai 1 jika *shift* adalah pola yang diinginkan. Nilai kurang dari 1 apabila *shift* mendekati pola yang diinginkan tetapi tidak sama persis.

9. *Soft Constraint* 9

   Tidak boleh terlalu banyak pola *shift* yang tidak diinginkan perawat. Kebalikan dari *soft constraint* 8, batasan ini mengatur tentang pola yang tidak diinginkan oleh perawat. Semakin banyak pola penjadwalan yang tidak diinginkan maka semakin besar nilai penalti yang dihasilkan. Model matematis dari *soft constraint* 8 ditunjukkan pada persamaan 2.35.

$$p9(x) = \sum_{e \in E, i \in U} u_{ei}(x) \quad (2.35)$$

$u_{ei}(x)$ merupakan pola dari *shift* yang tidak diinginkan karyawan e dalam sebuah solusi penjadwalan.

4.2.3. Fungsi Tujuan

Fungsi tujuan dari permasalahan adalah meminimalkan nilai penalti yang didapatkan dari penyusunan jadwal perawat akibat pelanggaran *soft constraint*. Berikut merupakan model matematis dari fungsi tujuan permasalahan *nurse rostering* ditunjukkan pada persamaan 2.36.

$$minimize : f = \sum_{m=1}^{9} K_m p_m \quad (2.36)$$

K_m merupakan bobot yang diberikan untuk setiap pelanggaran *soft constraint*. Nilai K_m yang digunakan pada penelitian tugas akhir diasumsikan 1 karena keterbatasan informasi dari peneliti terdahulu. Kemudian p_m adalah nilai penalti yang dihasilkan pada setiap pelanggaran *soft constraint*. Hasil perkalian antara bobot dan nilai penalti akan dijumlahkan menjadi total skor akhir penalti untuk setiap solusi.

4.3. Pemodelan Algoritma *Tabu-Simulated Annealing*

Algoritma *Tabu-Simulataed Annealing* akan digunakan untuk menyelesaikan permasalahan penjadwalan. Sebelum melakukan implementasi algoritma *Tabu-Simulated Annealing* dilakukan penyusunan inisiasi solusi terlebih dahulu. Kemudian solusi awal tersebut akan dioptimasi menggunakan algoritma *Tabu-Simulated Annealing*. Implementasi algoritma nantinya akan menggunakan kerangka *hyper-heuristi*cs. Pada *hyper-heuristic* akan terdapat dua mekanisme yaitu pemilihan *low-level heuristics* dan *acceptance criteria*. Algoritma *Tabu Search*

dan *Simulated Annealing* akan digabungkan sehingga mendapatkan solusi yang lebih optimal.

Algoritma *Tabu Search* memiliki *tabu list* yang dapat digunakan untuk seleksi *low-level heuristics*. Solusi yang tidak menghasilkan nilai penalti lebih baik maka *low-level heuristics* yang digunakan akan terseleksi dan masuk ke dalam *tabu list*. Apabila tabu list yang digunakan pada algoritma *Tabu Search* sudah penuh, maka *low-level heuristics* yang pertama kali masuk ke *tabu list* akan dikeluarkan. Lalu *low-level heuristics* baru akan dimasukkan. Keadaan akan terus berulang hingga pada batas yang ditentukan. Batas yang dimaksud bisa berupa iterasi atau batas waktu eksekusi algoritma. Kode 4.1 merupakan *pseudocode* dari algoritma *Tabu Search*.

```
procedure Tabu Search
    best penalty = current penalty
    current penalty = candidate penalty
    set tabu list T
    set initial length L
    repeat
        for i = 1 to L do
        random LLH
        if (candidate penalty <= current penalty) then
            current penalty = candidate penalty
            if (current penalty <= best penalty) then
                best penalty = current penalty
            else
                if (T size = T Length) then
                    delete the oldest entry of llh
                    insert llh to tabu T
                else
                    insert llh to tabu T
                end if
            end if
        end if
        end for
    until stop condition
end procedure
```

Kode 4.1 *Pseudocode* algoritma *Tabu Search*

Sedangkan algoritma *Simulated Annealing* bertindak sebagai *acceptance* criteria untuk mengambil keputusan solusi mana yang akan diterima maupun ditolak. Solusi yang tidak lebih baik akan di evaluasi pada proses *annealing*. Jika perhitungan melebihi nilai *random* maka solusi bisa diterima. Berikut merupakan *pseudocode* dari *Simulated Annealing* yang ditunjukkan pada Kode 4.2:

```
procedure Simulated Annealing
    best penalty = current penalty
    current penalty = candidate penalty
    set initial temperature T
    set initial length L
    repeat
        for i = 1 to L do
        random LLH
            prob = exp (current penalty - candidate penalty/T)
            if (candidate penalty <= current penalty) then
            current penalty = candidate penalty
                if (current penalty <= best penalty) then
                best penalty = current penalty
                else
                    if prob > random then
                    current penalty = candidate penalty
                    end if
                end if
            end if
        end for
    until stop condition
end procedure
```

Kode 4.2 *Pseudocode* algoritma *Simulated Annealing*

Kedua algoritma akan digabungkan untuk penyusunan jadwal perawat. Penggabungan kedua algoritma memungkinkan adanya diversifikasi yaitu menerima solusi yang tidak lebih baik sehingga tidak terjebak dalam *local optima*. Setelah mendapatkan solusi optimum nantinya performa dari algoritma Tabu-*Simulated Annealing* akan dibandingkan dengan algoritma *Hill Climbing*, *Simulated Annealing*, dan *Tabu Search* untuk melihat dari keempat algoritma tersebut manakah yang memiliki performa paling unggul.

4.4. Perencanaan Skenario Parameter

Algoritma Tabu-*Simulated Annealing* memiliki beberapa parameter yang bisa digunakan untuk bereksperimen dengan tujuan mencari hasil optimasi yang terbaik. Parameter yang digunakan berasal dari parameter algoritma *Tabu Search* dan algoritma *Simulated Annealing*. Berikut merupakan beberapa parameter yang akan digunakan pada penelitian yang ditunjukkan pada Tabel 4.2:

Tabel 4.2 Daftar Skenario Parameter

Parameter	Keterangan
T_0	Suhu awal algoritma *Simulated Annealing*
a	*Cooling rate* atau koefisien penurunan suhu pada algoritma *Simulated Annealing*
b	*Reheating rate* atau koefisien kenaikan suhu pada proses *reheating* algoritma *Simulated Annealing*
Nb	Jumlah iterasi tiap kenaikan suhu (*reheating*)
TL	Panjang *tabu list*

4.5. Kesimpulan Perancangan

Pada perancangan terdapat proses pemahaman dataset dan penyesuaian model matematis dengan tujuan memudahkan pada saat konversi model matematis ke dalam struktur data. Tugas akhir akan menggunakan dua algoritma yaitu *Tabu Search* dan *Simulated Annealing*. Keduanya akan digabungkan menggunakan kerangka *Hyper-Heuristics*. Pada bab ini juga merencanakan tentang parameter-parameter yang mungkin untuk diuji coba pada saat implementasi dengan tujuan mendapatkan hasil paling optimal. Selanjutnya, hasil dari perancangan yang telah disusun akan diimplementasikan dan dibahas pada Bab 5.

BAB V
IMPLEMENTASI

Bab implementasi akan menjelaskan proses implementasi algoritma untuk menyelesaikan permasalahan pada tugas akhir dimulai dari pembuatan solusi awal hingga akhir proses optimasi solusi. Implementasi dilakukan sesuai dengan rancangan yang telah disusun pada Bab 4. Implementasi dari algoritma *Tabu-Simulated Annealing* menggunakan Bahasa pemorgraman Java.

5.1. Pembuatan Solusi Awal

Proses awal dari implementasi adalah pembentukan solusi awal atau *generate initial solution*. Solusi awal akan terbentuk apabila seluruh *hard constraint* telah terpenuhi. Proses pertama dari pembentukan solusi awal diawali dengan membaca *file* masukan atau *input*, membentuk matriks, membangun penyelesaian *hard constraint*, menghitung nilai penalti, dan diakhiri dengan penyimpanan solusi awal.

5.1.1. Pembacaan File Input

Dataset Norwegian Hospitals memiliki *file input* dengan ekstensi xls atau *excel*. Untuk membaca *file input* dibuat beberapa *class* yang merepresentasikan tiap *sheet* dari *file*. Setelah itu dibuat objek-objek sesuai dengan data yang akan dimasukkan. Objek yang dibuat memiliki bermacam-macam atribut beserta tipe data. Tipe data akan digeneralisasi menjadi bertipe *String* lalu disimpan untuk digunakan pada tahap selanjutnya.

5.1.2. Pembuatan Matriks *Employee – Day*

Matriks dibuat berdasarkan hasil pembacaan *file input* fungsinya untuk penyimpanan pembacaan *file input*. Struktur penyimpanan matriks berbentuk *array* 2 dimensi dengan ukuran

penyimpanan sebesar banyaknya karyawan dan panjang hari yang akan dijadwalkan. Baris pada matriks akan merepresentasikan karyawan sedangkan kolom pada matriks merepresentasikan panjang hari.

5.1.3. Pembuatan *Hard Constraint*

Aktivitas utama dari pembuatan solusi awal adalah menyusun jadwal perawat dengan memenuhi seluruh *hard constraint* yang ditetapkan. Pada *dataset* memiliki 7 *hard constraint* yang harus dipenuhi agar solusi menjadi *feasible.*

5.1.3.1. Implementasi *Hard Constraint* Secara Berurutan

Pembuatan *Hard Constraint* secara berurutan diimplementasikan untuk beberapa *hard constraint* yaitu *hard constraint* 2, 4, 5, dan 7. Maksud dari berurutan disini adalah penempatan *shift* pada *employee* diurut dari hari pertama untuk perawat pertama hingga seterusnya.

Pada *hard constraint* 2 penempatan *shift* secara berurutan dimulai dari perawat pertama dialokasikan *shift* pertama untuk hari pertama hingga seterusnya. Pada penempatan *shift* juga diperhatikan bahwa seorang perawat hanya bisa bekerja pada *shift* di akhir pekan sesuai dengan *sheet employee* pada kolom *working weekend.* Maka dari itu penempatan *shift* untuk akhir pekan mendapatkan prioritas baru secara urut diisikan ke hari lainnya. Terpenuhinya *hard constraint* 2 apabila *shift* yang dialokasikan telah sesuai dengan kebutuhan rencana tenaga kerja. Namun, karena penyelesaian *hard constraint* 2 ini merupakan tahap paling awal dari pembuatan solusi awal, jadi hasil yang didapatkan adalah tidak seluruh perawat mendapatkan *shift* melainkan mayoritas perawat di urutan teratas.

Hard constraint 4 diimplementasikan setelah *hard constraint* 2. Masih dengan penempatan *shift* yang berurutan namun pada

hard constraint ini akan di cek apakah *shift* yang dialokasikan memiliki kesesuaian antara kompetensi yang dimiliki oleh perawat atau tidak. Apabila kompetensi yang dimiliki karyawan tidak sesuai dengan *shift* yang akan dialokasikan maka *shift* akan ditempatkan pada perawat selanjutnya yang memiliki kompetensi sesuai.

Selanjutnya adalah implementasi dari *hard constraint* 5, batasan ini dibuat dengan cara menghitung waktu jarak *shift* dari setiap perawat. *Shift* yang dihitung merupakan *shift* berpasangan seperti yang sudah dijelaskan pada bab sebelumnya. Cara perhitungan *shift* adalah dimulai dari akhir waktu berakhirnya *shift* hingga waktu dimulainya *shift* berikutnya. Apabila jarak waktu antar *shift* kurang dari batas yang ditentukan maka *shift* akan dialokasikan untuk perawat lainnya.

Terakhir adalah implementasi *hard constraint* 7 yang menggunakan metode urutan. Pada saat implementasi *hard constraint* 2 masih banyak perawat yang belum mendapatkan *shift* kerja sehingga ada ketimpangan jam kerja perawat tiap minggu. Perawat urutan atas akan terlalu banyak mendapatkan jam kerja sedangkan perawat urutan bawah mendapatkan sedikit jam kerja bahkan tidak sama sekali. Pada *hard constraint* 7 akan di cek apabila perawat memiliki jam kerja yang melewati batas maksimal, maka *shift* yang dialokasikan untuk perawat tersebut akan dipindahkan kepada perawat lain yang memiliki kekurangan jam kerja jauh di bawah batas maksimal.

Seluruh *hard constraint* yang diimplementasikan akan membentuk jadwal perawat yang lebih merata untuk setiap perawat. Namun, untuk *hard constraint* 3 dan 6 akan diterapkan menggunakan metode penyelesaian lain karena jika memakai metode berurutan kedua *hard constraint* ini masih belum menciptakan solusi yang *feasible.*

5.1.3.2. Implementasi *Hard Constraint* Secara Acak

Hard Constraint yang diselesaikan secara acak ada 2 yaitu *hard constraint* 3 dan *hard constraint* 6. Implementasi kedua *hard constraint* ini menggunakan metode acak dengan menerapkan *low-level heuristic*s yang disebutkan pada artikel rujukan. *Low-level heuristics* ada 3 meliputi:

1. 2-*Exchange*
   Metode *random shift* dengan cara menukar dua *shift* pada hari yang sama. Penukaran dilakukan pada perawat yang berbeda.
2. 3-*Exchange*
   Metode *random shift* dengan cara menukar tiga *shift* pada hari yang sama dan dilakukan pada perawat yang berbeda.
3. *Double* 2-*Exhange*
   Metode *random shift* dengan cara menukar *shift* diantara dua perawat pada dua hari. Dua hari yang ditukar tidak harus hari yang berurutan tetapi dua *shift* yang akan ditukar harus berasal dari kategori *shift* yang sama.

Implementasi *Hard Constraint* 3 dilakukan dengan cara menghitung selisih rata-rata jam kerja kontrak perawat dengan batas atas dan batas bawah sebesar 2%. Setiap minggu seorang perawat akan memiliki total jam kerja, lalu total jam kerja tersebut dibagi dengan jumlah minggu penjadwalan untuk mendapatkan rata-ratanya. Apabila rata-rata tiap minggunya kurang atau melebihi batas pada kontrak kerja maka dilakukan *random shift* menggunakan *low-level heuristics*. Solusi akan diterima apabila setelah dilakukan random *shift* selisih jam berubah mendekati 0 atau berkurang dari jam sebelumnya hingga sesuai dengan batas yang ditentukan.

Terakhir adalah implementasi dari *Hard Constraint* 6 yang mangatur tentang durasi dari waktu libur seorang perawat. Durasi waktu libur perawat yang dipilih pada *hard constraint*

ini adalah waktu libur yang memiliki durasi paling panjang. Perhitungan waktu libur dimulai dari berakhirnya jam kerja hingga dimulainya kembali jam kerja. Apabila ada karyawan yang memiliki waktu libur dengan durasi melebihi batasan maka akan dilakukan random *shift* menggunakan *low-level heuristics* hingga akhirnya ditemukan solusi yang *feasible.*

5.1.4. Perhitungan Nilai Penalti

Nilai penalti akan diberikan apabila terjadi pelanggaran terhadap *soft constraint*. Perhitungan penalti dilakukan ketika solusi yang terbentuk sudah *feasible* atau memenuhi keseluruhan *hard constraint*. Berikut merupakan penjelasan dari masing-masing perhitungan nilai penalti akibat pelanggaran *soft constraint*.

5.1.4.1. Perhitungan Nilai Penalti *Soft Constraint* 1

Soft Constraint 1 mengatur tentang batas maksimal kerja secara berturut-turut pada kategori yang sama. Perhitungan penalti disesuaikan dengan model matematis yang telah ditentukan. Cara menghitung batasan ini yaitu solusi yang terbentuk akan diidentifikasi berapa banyak seorang perawat akan bekerja berturut-turut pada kategori *shift* yang sama, baik *shift day*, *evening*, maupun *night*. Apabila seorang perawat bekerja melebihi dari batas maksimal *dataset* maka akan diberikan penalti pada hari diluar batas tersebut. Penalti akan diakumulasikan sebanyak hari, perawat, dan kategori *shift* yang dilanggar.

5.1.4.2. Perhitungan Nilai Penalti *Soft Constraint* 2

Tidak berbeda jauh dengan batasan sebelumnya, *soft constraint* 2 mengatur tentang batas maksimal bekerja pada hari yang berurutan. Jika pada sebelumnya yang diidentifikasi adalah kategori *shift* yang sama pada hari berurutan, batasan ini akan mengidentifikasi berapa banyak seorang perawat akan bekerja pada hari berurutan walaupun pada kategori *shift* apapun.

Apabila seorang perawat bekerja melebihi dari batas maksimal *dataset* maka akan diberikan penalti pada hari diluar batas tersebut. Penalti akan diakumulasikan sebanyak perawat dan hari yang dilanggar.

5.1.4.3. Perhitungan Nilai Penalti *Soft Constraint* 3

Soft Constraint 3 adalah kebalikan dari *soft constraint* 1. Pada batasan ini mengatur tentang batas minimal seorang perawat bekerja pada kategori *shift* yang sama paha hari berurutan. Perhitungan penalti dilakukan dengan cara mengidentifikasi pada kategori apa saja seorang perawat bekerja. Perhitungan penalti dilakukan dengan mempertimbangkan hari besok untuk melihat kategori *shift* apa yang diterima oleh perawat. Apabila seorang perawat bekerja kurang dari batas minimal *dataset* maka akan diberikan penalti pada hari pertama perhitungan penalti kategori *shift*. Penalti akan diakumulasikan sebanyak hari, perawat, dan kategori *shift* yang dilanggar.

5.1.4.4. Perhitungan Nilai Penalti *Soft Constraint* 4

Perhitungan penalti *soft constraint* 4 sama halnya dengan *soft constraint* 3 namun yang diatur pada *soft constraint* ini adalah banyaknya hari dimana seorang perawat bekerja secara berturut-turut. Pelanggaran akan terjadi ketika seorang perawat kurang dari batas minimal bekerja pada hari berurutan. Perhitungan penalti dilakukan dengan mempertimbangkan hari besok untuk melihat apakah seorang perawat bekerja atau tidak, dari kategori *shift* apapun. Apabila seorang perawat bekerja kurang dari batas minimal *dataset* maka akan diberikan penalti pada hari pertama perhitungan penalti hari kerja. Penalti akan diakumulasikan sebanyak perawat dan hari yang dilanggar.

5.1.4.5. Perhitungan Nilai Penalti *Soft Constraint* 5

Soft constraint 5 dihitung dengan cara mempertimbangkan batas minimal dan batas maksimal dari jumlah *shift* yang

dialokasikan untuk setiap perawat pada setiap kategori *shift*. Nilai penalti akan diberikan ketika jadwal yang diterima perawat tidak dalam range yang diatur batasan ini. Penalti akan diakumulasikan sebanyak kategori *shift*, perawat, dan hari yang dilanggar. Lalu dikuadratkan dan diakumulasi kembali setelah itu diakar untuk mendapatkan total penalti pelanggaran.

5.1.4.6. Perhitungan Nilai Penalti *Soft Constraint* 6

Penalti *soft constraint* 6 dihitung dengan cara mencari selisih antara jam kontrak kerja perawat pada *dataset* dengan durasi shif yang diterima karyawan dalam 1 minggu. Kontrak jam kerja *dataset* telah dipaparkan dalam kolom *workweek* pada sheet employee *dataset*. Lalu untuk mendapatkan durasi *shift* dilakukan penjumlahan dari setiap jadwal *shift* yang diterima oleh karyawan. Selisih kedua jam kontrak kerja akan dikuadratkan dan diakumulasikan untuk tiap karyawan lalu skor penaltinya akan diakar. Sehingga didapatkan skor penalti akhir untuk pelanggaran *soft constraint* 6.

5.1.4.7. Perhitungan Nilai Penalti *Soft Constraint* 7

Skor penalti *soft constraint* 7 dihitung dengan cara mengidentifikasi hari libur perawat untuk hari ini dan hari besok. Jika hari libur berurutan maka skor pelanggaran akan berkurang. Penalti akan diakumulasikan sebanyak hari dan jumlah karyawan. Lalu dikuadratkan dan diakumulasi kembali setelah itu diakar untuk mendapatkan total penalti pelanggaran.

5.1.4.8. Perhitungan Nilai Penalti *Soft Constraint* 8

Soft Constraint 8 mengatur tentang pola-pola *shift* yang diinginkan oleh perawat. Total pelanggaran penalti didapatkan dari jumlah perawat yang mencantumkan pola *shift* yang diinginkan dan dikalikan dengan jumlah hari yang akan dijadwalkan. Pelanggaran akan berkurang seiring dengan

semakin banyak alokasi penjadwalan dari pola *shift* yang diinginkan untuk seorang perawat.

5.1.4.9. Perhitungan Nilai Penalti *Soft Constraint* 9

Berkebalikan dengan *soft constraint* 8, *soft constraint* 9 akan menghitung pelanggaran-pelanggaran yang muncul akibat dari alokasi pola *shift* yang tidak diinginkan. Pada setiap minggu yang berhasil dijadwalkan akan diidentifikasi apakah terdapat pola *shift* yang tidak diinginkan dari setiap perawat. Total skor pelanggaran akan semakin besar jika pola *shift* yang tidak diinginkan semakin banyak dialokasikan. Pelanggaran akan dihitung untuk tiap perawat.

5.1.5. Penyimpanan Solusi Awal

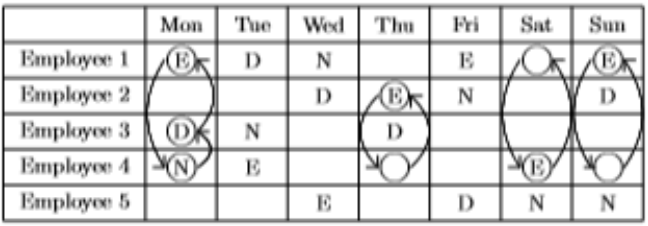
Solusi awal yang sudah terbentuk akan disimpan dalam bentuk *file* dengan ekstensi .sol. *File* ini nantinya akan dijadikan masukan untuk optimasi solusi. Solusi yang terbentuk berupa matriks *employee – day* dimana baris adalah urutan perawat dan kolom adalah urutan hari. Solusi yang disimpan berbentuk integer yang merepresentasikan *shift*. Angka 0 adalah *shift Free*, 1 dan seterusnya merupakan *shift* yang sesuai dengan urutan pada *file excel*. Pada inisiasi solusi juga dilakukan perhitungan penalti untuk melihat angka pelanggaran yang terdapat pada solusi awal. Proses optimasi nantinya akan menurunkan penalti tersebut dengan cara mengurangi pelanggaran *soft constraint*.

5.2. Optimasi Solusi

Bagian ini akan menjelaskan tentang cara optimasi dari solusi awal hingga menjadi solusi optimum. Optimasi akan menggunakan algortima Tabu-*Simulated Annealing* dengan kerangka kerja *Hyper Heuristics*. Pada optimasi solusi kali ini, memiliki tahapan utama yaitu proses dari implementasi algoritma utama.

5.2.1. Implementasi *Low-Level Heuristics*

Low-level heuristic atau biasa disingkat LLH merupakan metode untuk melakukan penukaran atau pemindahan solusi hingga menghasilkan solusi baru yang memiliki penalti lebih baik. Pada permasalahan tugas akhir ini, solusi yang akan ditukar berupa *shift*. Seperti pada proses pembuatan inisiasi solusi, low-level heuristics pada tugas akhir sudah ditentukan pada artikel yang menjadi rujukan. Terdapat 3 *low-level heuristics* yaitu 2-*Exchange*, 3-*Exchange*, dan *Double* 2-*Exchange*. Ilustrasi dari penukaran *shift*[5] ditunjukkan seperti gambar Gambar 5.1.



Gambar 5.1 Ilustrasi *low-level heuristics*

Pada hari senin merupakan ilustrasi dari 3-*Exchange* dimana terdapat 3 *shift* pada karyawan yang berbeda akan ditukar pada hari yang sama, lalu pada hari kamis merupakan ilustrasi dari 2-*Exchange* yaitu 2 *shift* dari perawat yang berbeda akan ditukar pada hari yang sama, lalu terakhir pada hari sabtu dan minggu adalah ilustrasi dari *Double* 2-*Exchange* yaitu penukaran *shift* yang terjadi pada 2 perawat dan 2 hari yang berbeda, namun harus kategori *shift* yang sama. Sel kosong pada gambar artinya adalah *shift* kosong.

Low-level heuristics akan digunakan secara bersamaan dalam tugas akhir, jadi akan dilakukan pemilihan secara acak dari ketiga *low-level heuristics* tersebut. Pemilihan *low-level heuristics* akan menggunakan algoritma *Tabu Search* yang memanfaatkan parameter *tabu list*. Setiap *low-level heuristics*

yang terpilih akan menghasilkan solusi, namun ketika solusi sementara yang didapatkan tidak lebih baik dari sebelumnya maka *low-level heuristics* terpilih tadi akan dimasukkan *tabu list.*

5.2.2. Implementasi Algoritma *Tabu-Simulated Annealing*

Implementasi algoritma utama yaitu *Tabu-Simulated Annealing* dilakukan untuk menemukan solusi yang paling optimum. Algoritma *Tabu Search* akan diimplementasikan sebagai penyeleksi *low-level heuristics* yaitu apabila solusi sementara yang dihasilkan tidak lebih baik dari solusi sebelumnya maka *low-level heuristics* akan dimasukkan dalam daftar pengecualian (*Tabu List*). Sehingga tidak akan digunakan kembali pada iterasi selanjutnya. *low-level heuristics* akan bisa digunakan kembali setelah keluar dari *tabu list.* Metode pengeluaran *low-level heuristics* dari *tabu list* adalah layaknya sistem *first in first out* yaitu yang pertama masuk maka akan dikeluarkan pertama.

Algoritma *Simulated Annealing* diimplementasikan sebagai *acceptance criteria*. Apabila solusi sementara yang dihasilkan mendapatkan nilai penalti lebih kecil akan langsung diterima sebagai solusi terbaik. Jika nilai penalti lebih tinggi akan dilakukan proses perhitungan atau dikenal proses *annealing.* Proses *annealing* menggunakan persamaan Boltzman yaitu $P = e^{-\frac{c}{t}}$, dimana P adalah Probabilitas Boltzman, e adalah bilangan eksponensial, c adalah perbedaan nilai fungsi tujuan antara solusi dengan kandidat solusi, dan t adalah parameter suhu. Jika nilai *random* tidak memenuhi persamaan Bolztmann, maka solusi sementara tidak akan diterima sebagai solusi baru

Berdasarkan penjelasan tersebut, kedua algoritma akan bekerja sama untuk menghasilkan solusi optimum. Sehingga alur dari implementasi algoritma Tabu-*Simulated Annealing* dapat dituliskan melalui *pseudocode* seperti Kode 5.1 berikut:

```
procedure Tabu Simulated Annealing
    best penalty = current penalty
    current penalty = candidate penalty
    set initial temperature A
    set reheat temperature B
    set initial length L
    set tabu list T
    set reheat range R
    repeat
        for i = 1 to L do
        random LLH
        if common solution is feasible then
            prob = exp (current penalty - candidate penalty/T)
            if (candidate penalty <= current penalty) then
                current penalty = candidate penalty
                copy solution to new solution
                if (current penalty <= best penalty) then
                    best penalty = current penalty
                    copy solution to best solution
                else
                    if (prob > random) then
                        if (T size = T Length) then
                            delete the oldest entry of llh
                            insert llh to tabu T
                        else
                            insert llh to tabu T
                    copy solution to new solution
                    end if
                else
                    copy new solution to solution
                end if
            else
                if (T size = T Length) then
                    delete the oldest entry of llh
                    insert llh to tabu T
                else
                    insert llh to tabu T
                end if
            end if
        else
            if (T size = T Length) then
                delete the oldest entry of llh
                insert llh to tabu T
            else
                insert llh to tabu T
            end if
        end if
        update temperature A
        if (reach reheat range R) then
            reheat temperature by B
        end if
        end for
    until stop condition
end procedure
```

Kode 5.1 *Pseudocode* algoritma *Tabu-Simulated Annealing*

Pada implementasi dijalankan dengan menggunakan 1000.000 iterasi. Terdapat beberapa kali pengecekan pada proses optimasi untuk pemenuhan kondisi penerimaan solusi. Solusi baru dapat diterima menjadi solusi terbaik:

(1) jika pada saat penukaran *shift* menggunakan *low-level heuristics* semua *hard constraint* dapat terpenuhi maka akan dilanjutkan proses optimasi menggunakan algoritma *Tabu-Simulated Annealing*. Apabila masih ada *hard constraint* yang tidak terpenuhi maka *low-level heuristics* dimasukkan *tabu list.*

(2) jika nilai penalti solusi sementara lebih rendah daripada nilai penalti solusi terbaik maka solusi sementara akan diterima menggantikan solusi terbaik.

(3) jika solusi sementara tidak menghasilkan nilai penalti yang lebih baik, maka dilakukan perhitungan proses *annealing*. Jika nilai *random* tidak lebih besar daripada nilai perhitungan proses *annealing* maka solusi sementara bisa diterima namun *low-level heuristics* akan dimasukkan ke dalam *tabu list.*

(4) jika berada pada kondisi dimana harus memasukkan *low-level heuristics* pada *tabu list* maka akan dicek apakah kapasitas *low-level heuristics* penuh atau tidak. Apabila penuh maka *low-level heuristics* lama akan dikeluarkan dan *low-level heuristics* baru dimasukkan.

Setelah proses optimasi diimplementasikan, selanjutnya dilakukan eksperimen parameter untuk menemukan solusi terbaik dari penerapan algoritma. Eksperimen yang dilakukan akan mengganti konstanta parameter.

5.2.3. Penyimpanan Solusi Optimum

Solusi optimum yang didapatkan dari hasil optimasi disimpan dalam *file* dengan ekstensi txt. Tiap solusi dari optimasi disimpan dalam 1 folder berdasarkan skenario. Pada tiap

skenario terdapat 10 *file* percobaan. Penyimpanan solusi akhir dalam bentuk matriks *employee–day*, sama dengan penyimpanan solusi awal. Solusi akhir yang disimpan sudah dalam bentuk nomor *shift*. Baris pada *file* merepresentasikan urutan dari perawat. Sedangkan kolom merupakan panjang hari yang dijadwalkan. Solusi akhir yang disimpan juga termasuk dengan skor penalti akhir hasil dari akumulasi pelanggaran *soft constraint* serta nilai penalti yang dicetak tiap kelipatan 5000 iterasi.

5.3. Kesimpulan Implementasi

Implementasi berhasil dilakukan pada *dataset Norwegian Hospitals*. Alur dari implementasi dimulai dari proses pembuatan solusi awal hingga optimasi solusi. Optimasi solusi tidak cukup pada 1 kali percobaan namun dilakukan eksperimen skenario pula untuk mencari parameter terbaik dari algoritma utama. Setiap percobaan implementasi dilakukan sebanyak 10 kali percobaan dengan 1000.000 iterasi. Seluruh proses implementasi dari algoritma *Tabu-Simulated Annealing* terdapat pada Lampiran E Keseluruhan Hasil Penjadwalan. Hasil dan pembahasan dari proses implementasi mulai dari pembuatan solusi awal hingga optimasi solusi dijelaskan pada Bab 6.

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan membahas tentang hasil dari uji coba algoritma *Tabu-Simulated Annealing* berdasarkan implementasi pada bab 5, hasil uji coba parameter, pemilihan skenario terbaik, serta perbandingan dengan algoritma lainnya.

6.1. Data Uji Coba

Data yang digunakan dalam penyelesaian masalah penjadwalan perawat adalah *Dataset Norwegian Hospitals.*

6.2. Lingkungan Uji Coba

Lingkungan pengujian yang digunakan untuk implementasi dari penelitian tugas akhir menjelaskan tentang spesifikasi dari perangkat keras dan perangkat lunak yang digunakan selama proses penelitian. Spesifikasi lingkungan uji coba berupa perangkat keras ditunjukkan pada Tabel 6.1.

Tabel 6.1 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi
Jenis	Laptop Lenovo ideapad 320
Processor	Intel® Core™ i5-7200U CPU @ 2.50GHz 2.71 Ghz
RAM	4,00 GB
Hard Disk Drive	SSD 240 GB

Jenis perangkat lunak yang digunakan sebagai lingkungan uji coba pada tugas akhir ditunjukkan pada Tabel 6.2.

Tabel 6.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	Jenis
Windows 10	Sistem Operasi
Intellij IDEA	Aplikasi Pengembangan
Microsoft Excel	Pengolahan Hasil Eksperimen
Microsoft Word	Penulisan Laporan Tugas Akhir
Notepad	Penyimpanan Solusi

6.3. Hasil Solusi Awal

Pada bagian ini akan menjelaskan solusi awal yang terbentuk setelah proses implementasi yang dijelaskan pada bab sebelumnya. Pembentukan solusi awal harus memenuhi *hard constraint* untuk dikatakan sebagai solusi awal yang *feasible.*

Dari keseluruhan *dataset* beberapa belum memenuhi *hard constraint* sehingga belum dikatakan sebagai solusi *feasible.* Hasil solusi awal menunjukkan bahwa 3 dari 7 *instance dataset* dapat memenuhi seluruh *hard constraint* sehingga dikatakan *feasible.* Solusi awal yang *feasible* adalah OpTur 4, OpTur 5, dan OpTur 7. Contoh keluaran hasil solusi awal dari *dataset* tersebut dapat dilihat pada Lampiran A Hasil Solusi Awal. Solusi tersebut nantinya akan dijadikan sebagai input atau masukan untuk proses optimasi. Berikut merupakan hasil penyusunan solusi awal dari keseluruhan *dataset* yang dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Solusi Awal

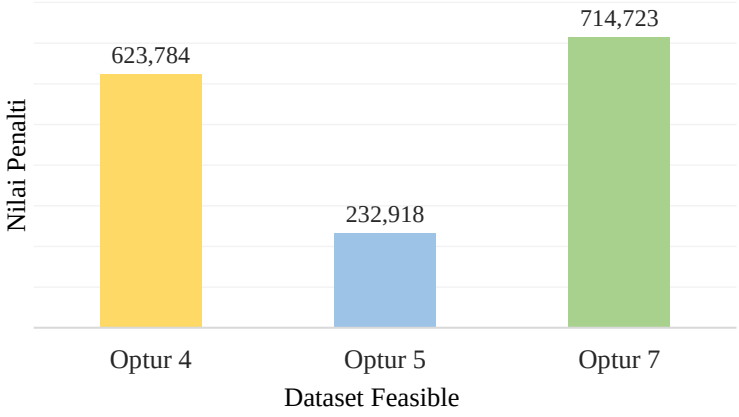
Dataset	**Keterangan**
OpTur1	Melanggar *hard constraint* 3
OpTur2	Melanggar *hard constraint* 3
OpTur3	Melanggar *hard constraint* 3
OpTur4	Solusi *feasible*
OpTur5	Solusi *feasible*
OpTur6	Melanggar *hard constraint* 3
OpTur7	Solusi *feasible*

Dataset sisanya yaitu OpTur 1, 2, 3, dan 6 melanggar *hard constraint* 3 sehingga solusi yang dihasilkan tidak *feasible.* Pelanggaran *hard constraint* yang terjadi pada keempat *dataset* adalah penyimpangan dari *hard constraint* 3 yang tidak sesuai dengan batas yang ditentukan. Pada *hard constraint* 3 mengatur tentang total jam kerja setiap perawat tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat. Penyimpangan yang diperbolehkan *dataset* adalah 2% sedangkan pada keempat

dataset memiliki penyimpangan yang lebih dari 2% sehingga solusinya tidak bisa diterima sebagai solusi yang feasible.

Proses penyusunan solusi awal khususnya pada implementasi *hard constraint* 3 tidak terlepas dari faktor acak atau *random*, sehingga diprediksi bahwa dari hasil penelitian ada faktor *random* pada saat melakukan penyusunan *shift* yang kurang tepat sehingga menyebabkan solusi tidak *feasible*. Langkah yang bisa diambil untuk mengatasi permasalahan penyusunan solusi awal ini adalah dengan memperbaiki algoritma sehingga mampu menghasilkan solusi awal yang *feasible* untuk keseluruhan *dataset*.

Setelah solusi awal disimpan, dilakukan perhitungan penalti berdasarkan pelanggaran *soft constraint*. Gambar 6.1 menunjukkan hasil dari penalti solusi awal. Sumbu X merupakan OpTur dari dataset yang *feasible.* Sedangkan sumbu Y merupakan nilai penalti. OpTur 5 mendapatkan total skor penalti terendah sebesar 232,917 diikuti dengan OpTur 4 dengan total penalti sebesar 623,78 dan terakhir OpTur 7 dengan total penalti sebesar 714,722.



Gambar 6.1 Nilai Penalti Solusi Awal

Rincian dari rata-rata nilai pelanggaran penalti solusi awal pada *dataset* ditunjukkan pada Tabel 6.4. Berdasarkan Tabel 6.4, diketahui bahwa pada solusi awal beberapa pelanggaran terjadi pada *soft constraint*. Nilai penalti 0 artinya tidak ada pelanggaran yang terjadi seperti pada *soft constraint* 4 dan 9 di semua OpTur *feasible*. Pada OpTur 4 pelanggaran terjadi pada *soft constraint* 4, 5, 8, dan 9. Pada OpTur 5 pelanggaran terjadi pada *soft constraint* 4, 8, dan 9. Lalu pada OpTur 7 pelangaran terjadi pada *soft constraint* 1, 2, 4, 5, dan 9. Penjelasan mengenai perhitungan nilai penalti dapat dilihat pada subbab 5.1.4.1.

Tabel 6.4 Daftar Pelanggaran Soft Constraint Solusi Awal

Soft Constraint	OpTur 4	OpTur 5	OpTur 7
SC 1	54	19	0
SC 2	31	20	0
SC 3	338	138	36
SC 4	0	0	0
SC 5	0	30.512	0
SC 6	2.1789	0.8903	9.851
SC 7	198.605	24.515	38.871
SC 8	0	0	630
SC 9	0	0	0

6.4. Hasil Eksperimen Algoritma *Tabu-Simulated Annealing*

Eksperimen algoritma dilakukan pada OpTur 4, OpTur 5, dan OpTur 7 karena ketiga *dataset* tersebut yang *feasible* pada saat penyusunan solusi awal. Parameter yang akan dipilih sebagai uji coba eksperimen telah ditentukan pada subbab perancangan.

Ada 13 skenario yang akan di uji pada implementasi algoritma *Tabu-Simulated Annealing*. Masing-masing skenario akan di uji coba sebanyak 10 kali dengan iterasi 1000.000. Nilai parameter didasarkan pada percobaan penulis secara acak. Perbedaan skenario terletak pada parameter yang dimiliki oleh algoritma

Tabu Search dan algoritma *Simulated Annealing*. Pada proses eksperimen ini juga dilakukan pengembangan algoritma dengan menambahkan skenario *reheating* pada proses implementasi algoritma.

Berikut merupakan skenario dari beberapa percobaan dengan mengganti konstanta parameter untuk mencari solusi terbaik ditunjukkan pada Tabel 6.5.

Tabel 6.5 Daftar Parameter Skenario

Skenario	Iterasi (Juta)	T_0	a	b	Nb	TL
A	1	1000000	0.99998	-	-	1
B	1	1000000	0.88889	-	-	1
C	1	1000000	0.99998	-	-	2
D	1	500000	0.99998	-	-	2
E	1	1000000	0.99998	0.1	500000	1
F	1	1000000	0.88889	0.1	500000	1
G	1	1000000	0.88889	0.5	500000	1
H	1	1000000	0.88889	0.5	200000	1
I	1	1000000	0.99998	0.1	500000	2
J	1	500000	0.99998	0.1	500000	2
K	1	1000000	0.99998	0.2	500000	2
L	1	1000000	0.99998	0.2	200000	2
M	1	1000000	0.99998	0.2	200000	1

Hasil yang akan dicatat untuk setiap skenario adalah rata-rata penalti 10 kali percobaan, penalti terbaik yang didapatkan tiap skenario, penalti terburuk dari setiap skenario, dan persentase penurunan penalti yang dihitung dari penalti solusi awal dikurangi dengan penalti hasil optimasi lalu dibagi penalti solusi awal. Terakhir tidak lupa dicatat durasi dari masing-masing eksperimen skenario yang dilakukan. Pencatatan durasi menggunakan millisecond atau ms. Pada subbab selanjutnya akan dibahas secara detil hasil eksperimen pada masing-masing

skenario. Contoh rekapitulasi hasil eksperimen skenario terlampir pada Lampiran B Hasil Eksperimen Skenario.

6.4.1. Hasil Eksperimen Skenario A

Pada skenario A, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* namun tanpa *reheating*. *Low-level heuristics* yang digunakan berjumlah 3 seperti yang ditetapkan oleh artikel rujukan dan menggunakan parameter berupa suhu awal, *cooling rate*, dan *tabu list*. Rincian dari koefisien parameter A terdapat pada Tabel 6.6.

Tabel 6.6 Parameter Skenario A

Skenario	Iterasi (Juta)	T_0	a	b	Nb	TL
A	1	1000000	0.99998	-	-	1

Hasil dari eksperimen skenario A menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Penurunan paling signifikan terjadi pada OpTur 5 sebanyak 77.42%. Berikut merupakan hasil dari eksperimen implementasi skenario A pada *dataset feasible* ditunjukkan pada pada Tabel 6.7.

Tabel 6.7 Hasil Eksperimen Skenario A

Dataset	Penalti Solusi Awal	Rata-Rata Penalti	Penalti Terbaik	Penalti Terburuk	Penurunan Penalti
OpTur 4	623.78	139.45	131.57	156.40	77.22%
OpTur 5	232.92	52.60	47.67	58.07	77.42%
OpTur 7	714.72	664.52	660.48	668.58	7.02%

Pada 1000.000 iterasi dibutuhkan durasi sekitar 981.228,3 *millisecond* atau kurang lebih 16 menit untuk OpTur 4, lalu dibutuhkan rata-rata durasi sekitar 65 detik untuk OpTur 5, dan durasi sepanjang 31 detik untuk OpTur 7. Lama dari durasi eksekusi skenario A dipengaruhi oleh ukuran dari *dataset*. OpTur 4 memiliki ukuran paling besar, sehingga durasi

eksekusi paling lama. Hasil rekapitulasi menunjukkan bahwa dari 10 kali percobaan yang dilakukan untuk skenario A dipaparkan rata-rata penalti yang didapatkan, penalti terbaik, penalti terburuk serta persentase penurunan penalti.

6.4.2. Hasil Eksperimen Skenario B

Pada skenario B, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* masih tanpa *reheating*. Perbedaan dengan skenario A adalah terletak pada nilai koefisien penurunan suhu. Skenario B menggunakan koefisien suhu yang lebih rendah yaitu 0.88889. Parameter yang digunakan adalah suhu awal, penurunan suhu, dan *tabu list* dengan koefisien parameter seperti yang tertera pada Tabel 6.8.

Tabel 6.8 Parameter Skenario B

Skenario	Iterasi (Juta)	T0	a	b	Nb	TL
B	1	1000000	0.88889	-	-	1

Hasil dari eksperimen skenario B menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. OpTur 4 mengalami kenaikan persentase penurunan penalti jika dibandingkan skenario A yaitu dari 77% menjadi 80%. Pada OpTur4 pula, *dataset* mengalami penurunan paling signifikan. Sebaliknya, pada OpTur 5 dan OpTur 7 justru nilai penalti mengalami kenaikan dibandingkan dengan skenario A. Pembahasan lebih lanjut mengenai analisis parameter akan dijelaskan pada saat pemilihan skenario terbaik. Berikut merupakan hasil dari eksperimen implementasi skenario B pada *dataset* feasible.

Eksekusi algoritma dengan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 16 menit untuk OpTur 4. Durasi sepanjang 61 detik untuk implementasi OpTur 5 dan 30 detik untuk OpTur 7. Lama dari durasi eksekusi skenario B dipengaruhi oleh ukuran dari *dataset*. OpTur 7 memiliki ukuran paling kecil,

sehingga durasi eksekusi paling cepat. Hasil dari eksperimen skenario B ditunjukkan pada Tabel 6.9

Tabel 6.9 Hasil Eksperimen Skenario B

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	124.33	119.32	129.55	80%
OpTur 5	232.92	76.52	68.58	90.28	67%
OpTur 7	714.72	678.15	673.56	682.65	5%

6.4.3. Hasil Eksperimen Skenario C

Pada skenario C, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* tanpa *reheating*. Pada skenario ini, dibedakan parameter panjang *tabu list* supaya penulis dapat melihat perbedaan dengan skenario sebelumnya. Parameter yang digunakan berjumlah 3 terdiri dari suhu awal, koefisien penurunan suhu, dan panjang *tabu list* dengan koefisien parameter seperti yang tertera pada Tabel 6.10.

Tabel 6.10 Parameter Skenario C

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
C	1	1000000	0.99998	-	-	2

Hasil dari eksperimen skenario C menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Namun, hasil dari eksperimen ini menunjukkan bahwa dengan mengubah parameter panjang *tabu list* yang awalnya 1 menjadi 2, hasilnya terjadi peningkatan persentase penurunan penalti untuk OpTur 5 dan OpTur 7. Lain halnya dengan OpTur 4 yang justru mengalami penurunan rata-rata penalti jika dibandingkan skenario B.

Eksekusi algoritma dengan 1000.000 iterasi dibutuhkan durasi rata-rata 16 menit untuk OpTur 4, 66 detik untuk OpTur 5, dan 32 detik untuk OpTur 7. Lama dari durasi eksekusi tergantung

pada ukuran *dataset*. Berikut merupakan hasil dari eksperimen implementasi skenario C pada *dataset* feasible ditunjukkan pada Tabel 6.11.

Tabel 6.11 Hasil Eksperimen Skenario C

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	137.16	129.52	152.46	78%
OpTur 5	232.92	52.43	46.04	61.49	77%
OpTur 7	714.72	661.84	658.78	665.35	7%

6.4.4. Hasil Eksperimen Skenario D

Pada skenario D, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* tanpa *reheating*. Skenario D berbeda dengan skenario sebelumnya. Letak perbedaannya terdapat pada inisiasi suhu awal yang sebelumnya 1.000.000 menjadi 500.000. Low level heuristic yang digunakan berjumlah 3 seperti yang ditentukan oleh artikel rujukan dan parameter yang digunakan berjumlah 3 terdiri dari suhu awal, koefisien penurunan suhu, dan panjang *tabu list* dengan koefisien parameter seperti yang tertera pada Tabel 6.12.

Tabel 6.12 Parameter Skenario D

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
D	1	500000	0.99998	-	-	2

Hasil dari eksperimen skenario D menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Namun tidak ada perubahan yang signifikan untuk nilai penalti yang didapatkan jika dibandingkan dengan skenario yang sebelumnya. OpTur 4 dan OpTur 5 mendapatkan rata-rata penurunan nilai penalti yang hamper sama yaitu sebesar 77%. Percobaan dilakukan sebanyak 10 kali dengan jumlah iterasi sebesar 1000.000.

Berikut merupakan hasil eksperimen yang didapatkan dari percobaan skenario D ditunjukkan pada pada Tabel 6.13.

Tabel 6.13 Hasil Eksperimen Skenario D

Dataset	Penalti Solusi Awal	Rata-Rata Penalti	Penalti Terbaik	Penalti Terburuk	Penurunan Penalti
OpTur 4	623.78	138.15	130.75	149.72	77%
OpTur 5	232.92	53.23	47.16	57.57	77%
OpTur 7	714.72	664.34	660.76	669.58	7%

Durasi rata-rata yang dibutuhkan untuk menyelesaikan optimasi menggunakan skenario D adalah 17 menit untuk OpTur 4, 67 detik untuk OpTur 5, dan 31.579 mili detik atau 31,5 detik untuk OpTur 7. Lama dari durasi eksekusi tergantung pada ukuran *dataset*. Semakin banyak jumlah perawat dan jumlah hari yang akan dijadwalkan maka akan semmakin lama durasi yang dibutuhkan untuk 1 kali percoban tiap skenario.

6.4.5. Hasil Eksperimen Skenario E

Pada skenario E, dilakukan pengembangan algoritma *Tabu-Simulated Annealing* dimana skenario ini akan mencoba menambahkan proses *reheating* untuk setiap proses eksperimen. Pada Tabel 6.14 di skenario E ada tambahan parameter berupa koefisian *reheating* dan jumlah iterasi untuk setiap kenaikan suhu.

Tabel 6.14 Parameter Skenario E

Skenario	Iterasi (Juta)	T0	a	b	Nb	TL
E	1	1000000	0.99998	0.1	500000	1

Hasil dari eksperimen skenario E menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Pada skenario E hasilnya hampir sama dengan skenario A, namun ditambah dengan proses *reheating* sebesar 10% pada tiap kelipatan iterasi

500.000. Menambah proses *reheating* belum memberikan dampak yang signifikan pada keseluruhan hasil penalti skenario E. Hal itu bisa jadi timbul karena kenaikan suhu *reheating* yang tidak terlalu besar, sehingga perubahan yang dihasilkan tampak tidak signifikan. Ditambah lagi bahwa kenaikan suhu hanya terjadi 1 kali yaitu pada iterasi ke 500.000. Perubahan penalti yang lebih baik hanya terjadi pada OpTur 5, selebihnya hampir sama dengan skenario sebelumnya. Pada Tabel 6.15 merupakan hasil dari eksperimen yang dilakukan pada *dataset feasible* menggunakan skenario E

Tabel 6.15 Hasil Eksperimen Skenario E

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	139.61	132.11	149.49	77%
OpTur 5	232.92	52.32	46.07	59.31	78%
OpTur 7	714.72	664.71	662.41	668.23	7%

Pada scenario E, ntuk menjalankan 1000.000 iterasi dibutuhkan durasi rata-rata 16 menit untuk OpTur 4, 65 detik untuk OpTur 5, dan 32 detik untuk OpTur 7. Lama dari durasi eksekusi tergantung pada ukuran *dataset*.

6.4.6. Hasil Eksperimen Skenario F

Eksperimen selanjutnya masih menggunakan proses *reheating* dalam algoritma *Tabu-Simulated Annealing*. Namun skenario F akan mengubah koefisien suhu menjadi lebih rendah. Berikut merupakan detil dari parameter pada skenario F ditunjukkan pada Tabel 6.16.

Tabel 6.16 Parameter Skenario F

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
F	1	1000000	0.88889	0.1	500000	1

Hasil dari eksperimen skenario F menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. OpTur 4 mengalami kenaikan persentase penurunan penalti jika dibandingkan skenario E yaitu dari 77% menjadi 79%. Namun, sebaliknya terjadi pada OpTur 5 dan OpTur 7 dimana nilai penalti mengalami kenaikan dibandingkan dengan skenario E. Hasil dari eksperimen menggunakan kelima parameter pada skenario F mendapatkan hasil seperti pada Tabel 6.17 dibawah ini.

Tabel 6.17 Hasil Eksperimen Skenario F

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	125.54	119.11	137.21	79%
OpTur 5	232.92	72.75	57.18	89.58	69%
OpTur 7	714.72	677.61	672.91	689.25	5%

Pada scenario F, untuk menjalankan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 947394.8 milidetik atau sekitar 15,7 menit untuk OpTur 4. Durasi sepanjang 63 detik untuk implementasi OpTur 5 dan 32 detik untuk OpTur 7. Lama dari durasi eksekusi skenario B dipengaruhi oleh ukuran dari *dataset*.

6.4.7. Hasil Eksperimen Skenario G

Percobaan eksperimen skenario G dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* dengan *reheating*. Parameter koefisien kenaikan suhu yang awalnya 0.1 menjadi 0.5 untuk melihat apakah hasil akan lebih baik. Detil dari parameter pada skenario G ditunjukkan pada Tabel 6.18.

Tabel 6.18 Parameter Skenario G

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
G	1	1000000	0.88889	0.5	500000	1

Pada skenario G proses *reheating* suhu sebesar 50% pada tiap kelipatan iterasi 500.000. Menambah proses *reheating* belum memberikan dampak yang signifikan pada OpTur 5 dan OpTur 7. Hal itu bisa jadi timbul karena kenaikan suhu hanya terjadi 1 kali yaitu pada iterasi ke 500.000. Perubahan penalti yang lebih yang cukup signifikan terjadi pada OpTur 4 karena sepanjang eksperimen yang sudah dilakukan, dengan menggunakan skenario G inilah OpTur 4 mencapai rata-rata penalti paling baik yaitu 121.99. Berikut hasil dari eksperimen pada *dataset feasible* ditunjukkan pada Tabel 6.19.

Tabel 6.19 Hasil Eksperimen Skenario G

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	121.99	117.35	128.11	80%
OpTur 5	232.92	73.84	63.86	82.36	68%
OpTur 7	714.72	677.56	669.90	685.13	5%

Hasil dari eksperimen skenario G menunjukkan bahwa untuk menjalankan 1000.000 iterasi dibutuhkan durasi rata-rata 17 menit untuk OpTur 4, 67 detik untuk OpTur 5, dan 31 detik untuk OpTur 7. Lama dari durasi eksekusi tergantung pada ukuran *dataset*.

6.4.8. Hasil Eksperimen Skenario H

Eksperimen algoritma *Tabu-Simulated Annealing* selanjutnya menggunakan *reheating* dan *low-level heuristics* yang digunakan berjumlah 3. Perbedaan skenario terletak pada jumlah iterasi untuk tiap kenaikan suhu. Berikut merupakan detil dari skenario parameter yang ditunjukkan pada Tabel 6.20.

Tabel 6.20 Parameter Skenario H

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
H	1	1000000	0.88889	0.5	200000	1

Hasil dari eksperimen skenario H menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Tetapi, tidak jauh berbeda dengan skenario G. OpTur 5 mengalami penurunan rata-rata nilai penalti dari 73.84 menjadi 68.89.

Pada skenario kali ini digunakan iterasi sejumlah 200.000 untuk tiap kenaikan suhu atau *reheating*. Dengan menggunakan *reheating* tersebut artinya suhu akan dinaikkan sebesar 50% pada setiap iterasi kelipatan 200.000. Namun, ternyata *reheating* tidak memberikan dampak signifikan. Hasil eksperimen dirangkum pada Tabel 6.21.

Tabel 6.21 Hasil Eksperimen Skenario H

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	124.78	122.88	126.59	80%
OpTur 5	232.92	68.89	60.17	81.41	70%
OpTur 7	714.72	679.29	676.08	683.90	5%

Pada skenario H untuk menjalankan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 19 menit untuk mengoptimasi OpTur 4, 61 detik untuk OpTur 5, dan 31 detik untuk OpTur 7.

6.4.9. Hasil Eksperimen Skenario I

Pada skenario I, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* dengan *reheating*. Parameter hampir sama dengan skenario E namun panjang *tabu list* dibedakan dari 1 menjadi 2. Tabel 6.22 merupakan rincian dari parameter skenario I.

Tabel 6.22 Parameter Skenario I

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
I	1	1000000	0.99998	0.1	500000	2

Hasil eksperimen skenario I menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Penambahan panjang *tabu list* untuk skenario ini tidak memberikan hasil yang lebih baik dari skenario E. Hal ini mungkin bisa terjadi karena jumlah *low-level heuristics* yang sedikit namun ukuran *tabu list* yang panjang sehingga menyebabkan eksplorasi solusi yang dilakukan tidak maksimal. Berikut merupakan hasil dari eksperimen pada *dataset feasible* ditunjukkan pada Tabel 6.23.

Tabel 6.23 Hasil Eksperimen Skenario I

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	140.28	131.50	146.81	77%
OpTur 5	232.92	53.14	49.22	56.64	77%
OpTur 7	714.72	665.14	658.95	668.25	7%

Pada skenario I, untuk menjalankan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 16 menit untuk mengoptimasi OpTur 4, 65 detik untuk OpTur 5, dan 32 detik untuk OpTur 7. Durasi optimasi tiap *dataset* berbanding lurus dengan ukuran *dataset*, artinya semakin besar *dataset* maka akan semakin lama waktu yang dibutuhkan untuk optimasi.

6.4.10. Hasil Eksperimen Skenario J

Percobaan selanjutnya mengimplementasikan algoritma *Tabu-Simulated Annealing* menggunakan skenario J. Inisiasi suhu awal akan diubah menjadi lebih kecil dari skenario sebelumnya. Skenario J akan dibandingkan dengan skenario I, apakah memberikan perbedaan hasil lebih baik. Detil parameter yang digunakan pada skenario J ditunjukkan pada Tabel 6.24.

Tabel 6.24 Parameter Skenario J

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
J	1	500000	0.99998	0.1	500000	2

Hasil dari eksperimen skenario J menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Penurunan penalti paling banyak terjadi pada OpTur 5 yaitu sebanyak 77.41% dari penalti solusi awal. Penurunan penalti tidak jauh berbeda dari skenario I. *Reheating* dilakukan tiap kelipatan iterasi 500.000 sebesar 10% dari suhu sementara pada saat penurunan suhu. Hasil dari eksperimen skenario J ditunjukkan pada Tabel 6.25.

Tabel 6.25 Hasil Eksperimen Skenario J

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	139.45	127.94	147.73	77.22%
OpTur 5	232.92	52.61	44.83	55.88	77.41%
OpTur 7	714.72	663.50	659.66	666.53	7.17%

Pada 10 kali percobaan, untuk menjalankan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 16 menit untuk mengoptimasi OpTur 4, 65 detik untuk OpTur 5, dan 32 detik untuk OpTur 7. Semakin besar *dataset* maka akan semakin lama waktu yang dibutuhkan untuk melakukan optimasi.

6.4.11. Hasil Eksperimen Skenario K

Eksperimen selanjutnya masih menggunakan proses *reheating* dalam algoritma *Tabu-Simulated Annealing*. Namun skenario K akan mengubah koefisien kenaikan suhu menjadi lebih tinggi. Berikut merupakan detil dari parameter yang digunakan pada skenario K ditunjukkan pada Tabel 6.26.

Tabel 6.26 Parameter Skenario K

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
K	1	1000000	0.99998	0.2	500000	2

Hasil dari eksperimen skenario K dirangkum menjadi sebuah tabel yang berisi penalti solusi awal, rata-rata penalti optimasi,

penalti terbaik, penalti terburuk, dan persentase penurunan penalti. Hasil eksperimen menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Perbedaan skenario pada eksperimen ini adalah koefisien kenaikan suhu. Sebelumnya, pada skenario J koefisien kenaikan suhu yang digunakan adalah 0.1, pada skenario kali ini digunakan koefisien sebesar 0.2 untuk kenaikan suhu atau *reheating*. Dengan menggunakan *reheating* tersebut artinya suhu akan dinaikkan sebesar 20% pada setiap iterasi kelipatan 500.000. Hasil penalti yang diperoleh tidak memberikan perbedaan yang signifikan. Berikut hasil eksperimen skenario K ditunjukkan pada Tabel 6.27

Tabel 6.27 Hasil Eksperimen Skenario K

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	140.41	134.59	145.83	77%
OpTur 5	232.92	52.94	45.65	59.62	77%
OpTur 7	714.72	663.30	658.66	666.55	7%

Percobaan dengan 1000.000 iterasi dibutuhkan rata-rata durasi sekitar 18 menit untuk mengoptimasi OpTur 4, 77 detik untuk OpTur 5, dan 36 detik untuk OpTur 7. Durasi eksekusi berbanding lurus dengan besarnya *dataset*.

6.4.12. Hasil Eksperimen Skenario L

Pada skenario L, percobaan dilakukan dengan menggunakan algoritma *Tabu-Simulated Annealing* dengan proses *reheating*. Pada skenario L diubah parameter iterasi tiap kenaikan suhu (Nb). Tabel 6.28 merupakan rincian dari parameter skenario L.

Tabel 6.28 Parameter Skenario L

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
L	1	1000000	0.99998	0.2	200000	2

Hasil dari eksperimen menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Jika dibandingkan dengan skenario K pengurangan jumlah iterasi untuk *reheating* tidak memberikan pengaruh pada rata-rata hasil penalti. Namun pada penalti terbaik mengalami perbaikan dimana OpTur 4 pada skenario K memiliki penalti terbaik 134.59 pada skenario L menjadi 132.71. Begitu juga dengan OpTur 7 pada skenario K memiliki penalti 666.55 sedangkan pada skenario L menjadi 660.02. Berikut hasil dari eksperimen skenario menggunakan *dataset* yang *feasible* ditunjukkan pada Tabel 6.29.

Tabel 6.29 Hasil Eksperimen Skenario L

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	143.15	132.71	151.70	77%
OpTur 5	232.92	54.34	49.16	60.09	77%
OpTur 7	714.72	664.33	660.02	667.60	7%

Durasi yang dibutuhkan untuk menjalankan 1000.000 iterasi pada 10 kali percobaan yaitu sekitar 17,5 menit untuk mengoptimasi OpTur 4, 76 detik untuk OpTur 5, dan 35 detik untuk OpTur 7. Lama durasi dipengaruhi oleh ukuran *dataset*.

6.4.13. Hasil Eksperimen Skenario M

Percobaan terakhir yaitu skenario M menggunakan algoritma *Tabu-Simulated Annealing* dengan *reheating*. Perbedaan parameter terdapat pada panjang *tabu list*. Jika pada skenario L panjang *tabu list* 2, skenario M menggunakan panjang *tabu list* 1. Detil parameter ditunjukkan pada Tabel 6.30.

Tabel 6.30 Parameter Skenario M

Skenario	**Iterasi (Juta)**	**T0**	**a**	**b**	**Nb**	**TL**
M	1	1000000	0.99998	0.2	200000	1

Berdasarkan skenario M menunjukkan bahwa terjadi penurunan pada masing masing *dataset*. Pada skenario M pengurangan panjang *tabu list* ternyata memberikan hasil yang lebih baik jika dibandingkan dengan skenario yang memiliki ukuran *tabu list* lebih panjang seperti pada skenario L. Dari rata-rata penalti yang dicatat terlihat bahwa ketiga *dataset* mengalami perbaikan. Mulai dari OpTur 4 pada skenario L memiliki rata-rata penalti 143.15 menjadi 140.52 pada skenario M. OpTur 5 pada skenario L memiliki rata-rata penalri 54.34 menjadi 53.34 pada skenario M dan terakhir pada OpTur 7 rata-rata penaltinya dari 664.33 menjadi 663.07. Berikut merupakan hasil eksperimen dari skenario M ditunjukkan pada Tabel 6.31.

Tabel 6.31 Hasil Eksperimen Skenario M

Dataset	**Penalti Solusi Awal**	**Rata-Rata Penalti**	**Penalti Terbaik**	**Penalti Terburuk**	**Penurunan Penalti**
OpTur 4	623.78	140.52	133.50	148.49	77%
OpTur 5	232.92	53.34	47.41	58.74	77%
OpTur 7	714.72	663.07	659.47	666.39	7%

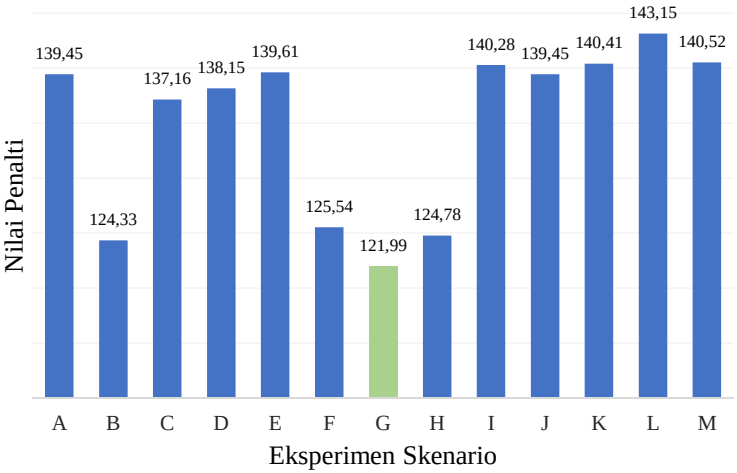
Rata-rata durasi yang dibutuhkan untuk menjalankan 1000.000 iterasi pada 10 kali percobaan yaitu rata-rata durasi 17 menit untuk mengoptimasi OpTur 4, 74 detik untuk OpTur 5, dan 35 detik untuk OpTur 7. Lama durasi dipengaruhi oleh ukuran *dataset*.

6.5. Pemilihan Skenario Terbaik

Berdasarkan hasil eksperimen skenario A hingga M, dilakukan perbandingan untuk menentukan skenario dengan parameter yang menghasilkan solusi terbaik. Hasil dari perbandingan skenario menunjukkan bahwa tiap *dataset* memiliki perbedaan pada skenario yang diimplementasikan.

OpTur 4 menunjukkan bahwa dengan menggunakan skenario G solusi dapat mencapai titik paling optimum dengan rata-rata nilai penalti 121.99 seperti yang ditunjukkan pada Gambar 6.2.

Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sedangkan sumbuh Y menunjukkan nilai penalti. Label rata-rata nilai penalti dari 10 kali percobaan eksperimen terletak diatas grafik tiap skenario.

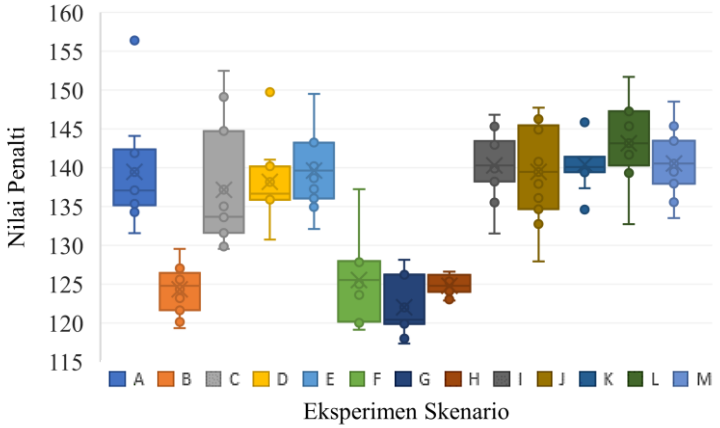


Gambar 6.2 Hasil Eksperimen Skenario Optur 4

OpTur 4 merupakan *dataset* dengan ukuran paling besar sehingga durasi yang dibutuhkan untuk menjalankan 1 kali percobaan paling lama diantara *dataset* lainnya. Pada OpTur 4 ketika koefisien penurunan suhu semakin rendah maka penalti yang dihasilkan akan semakin bagus. Hal itu bisa disebabkan oleh semakin kecil koefisien penurunan suhu akan mempercepat penurunan suhu sehingga algoritma *Tabu-Simulated Annealing* akan lebih lama dalam tahap eksploitasi solusi. Ukuran dari *dataset* yang besar juga bisa menambah peluang untuk mencari kombinasi solusi paling optimal, karena *search space* dari *dataset* luas.

Grafik *box plot* pada Gambar 6.3 menunjukkan persebaran dari penalti tiap skenario pada OpTur 4. Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sedangkan sumbu Y menunjukkan nilai penalti. Persebaran

nilai penalti pada grafik merupakan hasil dari 10 kali percobaan untuk setiap skenario. Bagian utama *boxplot* adalah kotak berbentuk persegi yang berisi nilai-nilai penalti. Semakin panjang *boxplot* maka menunjukkan data semakin menyebar atau luas. Misalnya, seperti pada *boxplot* skenario C.

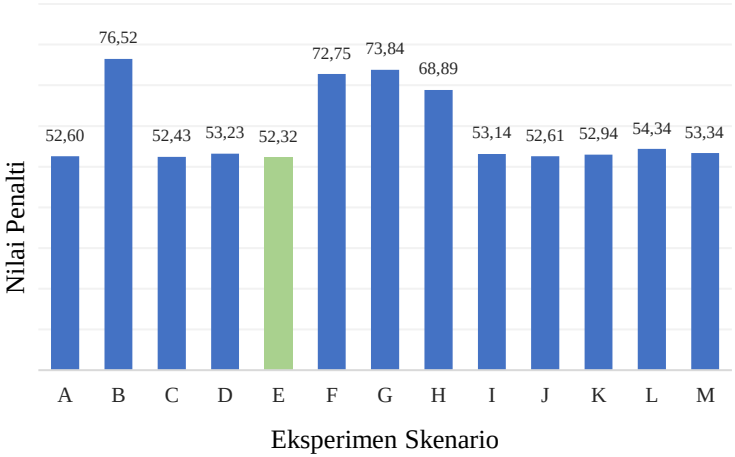


Gambar 6.3 Persebaran Nilai Penalti Skenario Optur 4

Lalu pada Gambar 6.4 menunjukkan bahwa dengan menggunakan skenario E di OpTur 5 dapat mencapai titik paling optimum dengan rata-rata nilai penalti 52.33. Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sumbu Y menunjukkan nilai penalti. Label rata-rata nilai penalti dari 10 kali percobaan eksperimen terletak diatas grafik tiap skenario.

Berbeda dengan OpTur 4, pada OpTur 5 ketika koefisien penurunan suhu semakin rendah maka penalti yang dihasilkan semakin jelek. Pada grafik dapat dilihat kecenderungan selain skenario B, F, G, dan H mayoritas skenario menggunakan koefisien penurunan suhu sebesar 0.9998 (koefisien penurunan suhu tinggi), dan solusi yang dihasilkan lebih bagus. Hal ini bisa disebabkan karena ketika menggunakan koefisien suhu yang

tinggi, penurunan suhu akan semakin lambat sehingga waktu eksplorasi dari *dataset* menjadi semakin lama.



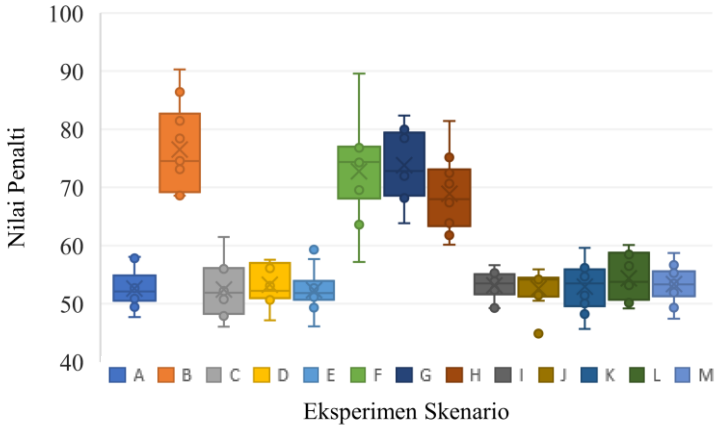
Gambar 6.4 Hasil Eksperimen Skenario Optur 5

Pada setiap eksperimen, parameter *tabu list* tidak memberikan pengaruh yang signifikan karena pada optimasi kali ini yang disimpan di dalam *tabu list* merupakan *low-level heuristics*. Sedangkan kondisinya, jumlah *low-level heuristics* yang terlalu sedikit membuat metode untuk *random* solusi semakin sempit dan sebagian kandidat metode yang tidak menghasilkan solusi lebih baik, maka *low-level heuristics* akan dimasukkan *tabu list.* Jumlah *low-level heuristics* yang semakin sedikit menyebabkan antara pemakaian *tabu list* dengan ukuran yang besar maupun kecil tidak memberikan perbedaan signifikan.

Jika *tabu list* yang diimplementasikan digunakan untuk menyimpan solusi maka akan ada terlalu banyak kemungkinan untuk memasukkan solusi ke dalam *tabu list* karena luasnya *search area* dan terlalu banyak variasi dari solusi yang dihasilkan dari hasil *random* solusi. Sehingga kemungkinan untuk menemukan pola solusi yang sama sangat sedikit sekali.

Grafik *boxplot* pada Gambar 6.5 menunjukkan persebaran dari penalti tiap skenario pada OpTur 5. Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sedangkan sumbu Y menunjukkan nilai penalti.

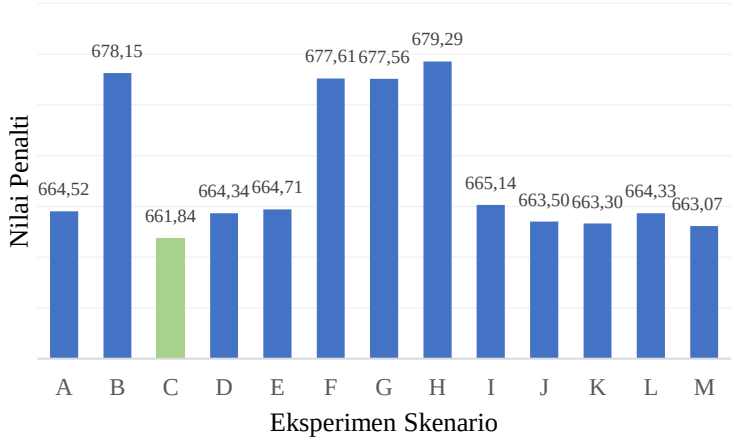
Skenario B memiliki persebaran nilai penalti yang luas dibandingkan dengann lainnya. Garis perpanjangan dari *boxplot* baik ke arah atas maupun bawah menunjukkan terdapat nilai penalti yang lebih tinggi dari kumpulan nilai penalti yang terdapat di dalam *boxplot*, contohnya skenario F. Skenario F terlihat memiliki garis perpanjangan ke atas yang artinya skenario tersebut memiliki nilai penalti yang lebih tinggi dari kumpulan nilai penalti yang terdapat pada *boxplot* skenario F.



Gambar 6.5 Persebaran Nilai Penalti Skenario Optur 5

Kemudian pada Gambar 6.6 menunjukkan bahwa dengan menggunakan skenario C dapat mencapai titik paling optimum dengan rata-rata nilai penalti 661,84 pada OpTur 7. Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sedangkan sumbu Y menunjukkan nilai penalti. Label rata-rata nilai penalti dari 10 kali percobaan eksperimen terletak diatas grafik tiap skenario.

Sama halnya dengan OpTur 5 ketika koefisien penurunan suhu semakin rendah maka penalti yang dihasilkan semakin jelek. Proses *reheating* pada eksperimen skenario memberikan hasil yang tidak terlalu berbeda dari eksperimen skenario tanpa *reheating*, dikarenakan kenaikan suhu dilakukan pada iterasi 200.000 dan 500.000 sehingga *reheating* dalam 1 kali percobaan eksekusi hanya dilakukan maksimal 5 kali. Hal itu membuat koefisien kenaikan suhu yang dipilih pada skenario masih belum ditemukan yang memberikan hasil terbaik. Serta inisiasi suhu yang digunakan pada awal iterasi sangat tinggi, lalu pada saat dilakukan *reheating* perubahan suhu tidak terlalu signifikan.

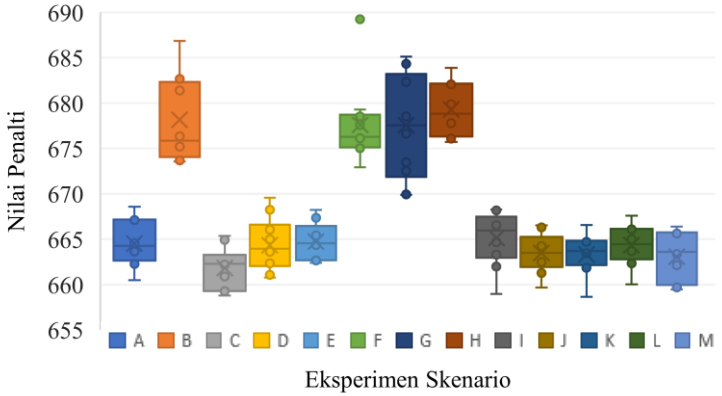


Gambar 6.6 Hasil Eksperimen Skenario Optur 7

Hasil eksperimen juga tidak terlepas dari *random* yang artinya setiap percobaan mungkin akan menghasilkan penalti lebih baik atau lebih buruk tergantung pada solusi yang di *random* pada saat itu.

Grafik *boxplot* pada Gambar 6.7 menunjukkan persebaran dari penalti tiap skenario pada OpTur 5. Sumbu X pada grafik menunjukkan eksperimen skenario dari scenario A hingga M, sedangkan sumbu Y menunjukkan nilai penalti. Skenario G

memiliki persebaran nilai penalti yang luas dibandingkan dengann lainnya. Skenario B memiliki nilai penalti yang lebih tinggi dibandingkan kumpulan nilai penalti dalam *boxplot* ditandai dengan adanya garis perpanjangan ke arah atas.



Gambar 6.7 Persebaran Nilai Penalti Skenario Optur 7

Pada diagram *boxplot* juga terdapat titik *outlier* atau pencilan yang menandakan bahwa di dalam kumpulan nilai penalti tersebut ada angka yang memiliki nilai yang sangat jauh berbeda dari angka lainnya, ditunjukkan pada skenario F.

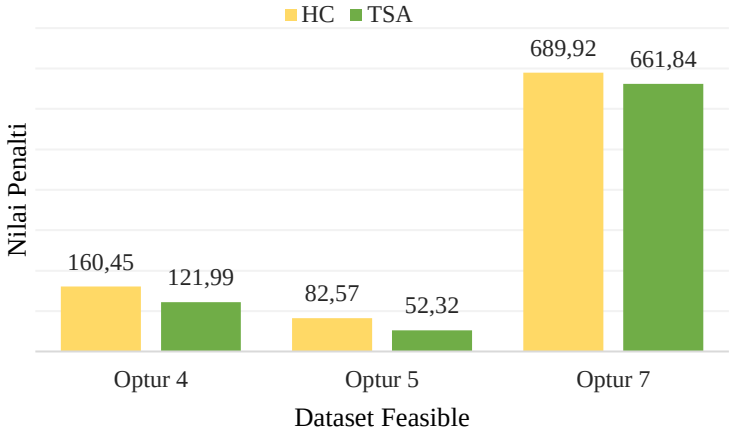
6.6. Perbandingan Performa Algoritma

Hasil dari eksperimen skenario yang memiliki rata-rata penalti terbaik kemudian dibandingkan dengan metode lain untuk melihat performa dari algoritma yang dibangun. Algoritma utama *Tabu-Simulated Annealing* akan dibandingkan dengan algoritma *Hill Climbing*, *Tabu Search*, dan *Simulated Annealing* dengan menggunakan kerangka *Hyper Heuristics.* Parameter yang digunakan adalah berdasarkan skenario dengan hasil eskperimen yang memiliki rata-rata nilai penalti terbaik.

6.6.1. Perbandingan dengan Algoritma *Hill Climbing*

Perbandingan performa dilakukan antara algoritma utama *Tabu-Simulated Annealing* dengan algoritma *Hill Climbing*. Perbandingan tersebut dilakukan untuk melihat performa yang lebih baik diantara keduanya. Parameter yang digunakan algoritma *Tabu-Simulated Annealing* diambil dari eksperimen skenario yang menghasilkan penalti terbaik. Parameter tersebut juga diterapkan pada algoritma *Hill Climbing*.

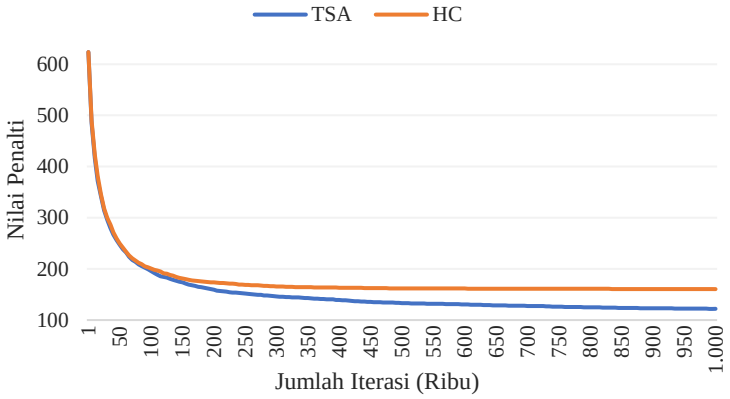
Berikut merupakan perbandingan nilai penalti algoritma *Tabu-Simulated Annealing* untuk 3 *dataset* feasible yang dibandingkan dengan algoritma *Hill Climbing*. Hasil perbandingan pada Gambar 6.8 menunjukkan bahwa rata-rata nilai penalti dari algoritma *Tabu-Simulated Annealing* masih lebih unggul jika dibandingkan dengan algoritma *Hill Climbing* secara keseluruhan. Sumbu X merupakan OpTur dari dataset yang *feasible*, sumbu Y merupakan nilai penalty. Kemudian label nilai penalti terletak diatas grafik masing-masing OpTur.



Gambar 6.8 Perbandingan Nilai Penalti Algoritma *Tabu-Simulated Annealing* dan *Hill Climbing*

Pada OpTur 4 algoritma *Tabu-Simulated Annealing* mampu menghasilkan nilai penalti 121.99 yang berselisih sekitar 40 angka dibawah nilai penalti algoritma *Hill Climbing*. Pada OpTur 5 nilai penalti dari algoritma Tabu-*Simulated Annealing* lebih unggul sekitar 30 angka dari nilai penalti algoritma *Hill Climbing*. Dan pada OpTur 7 algoritma *Tabu-Simulated Annealing* mampu menghasilkan nilai penalti 661.84 yang berada jauh dibawah nilai penalti algoritma *Hill Climbing*. Selanjutnya, performa dari algoritma masing-masing OpTur akan dijelaskan lebih detil pada Gambar 6.9 sampai Gambar 6.11.

Grafik pada Gambar 6.9 merupakan grafik *trajectory* yang gunanya untuk melihat *track* dari nilai penalti yang dihasilkan pada 1000.000 iterasi dan menunjukkan apakah hasil dari *running* algoritma *stuck* pada titik tertentu atau tidak. Garis berwarna orange merepresentasikan algoritma *Hill Climbing* sedangkan garis biru merepresentasikan algoritma *Tabu-Simulated Annealing*. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.



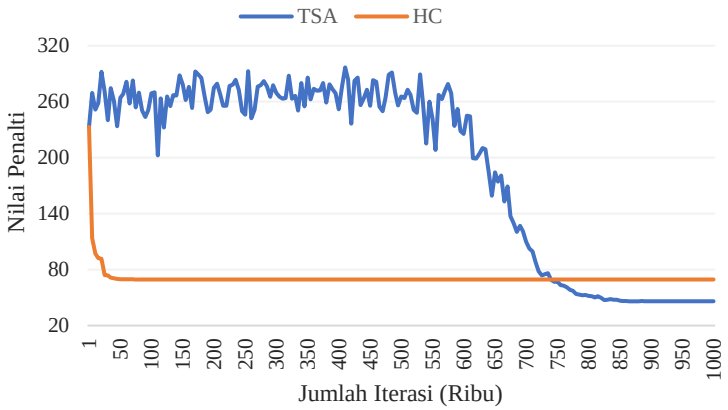
Gambar 6.9 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Hill Climbing* pada Optur 4

Pada OpTur 4, algoritma *Hill Climbing* cenderung turun lalu *stuck* pada iterasi ke 100.000. Hasil analisis penulis bahwa algoritma *Hill Climbing* cenderung turun dan *stuck* di titik tertentu karena pada konsepnya algoritma ini tidak melakukan diversifikasi sehingga mudah terjebak dalam *local optima*.

Sedangkan nilai penalti algoritma *Tabu-Simulated Annealing* turun perlahan sampai batas iterasi walaupun penurunan tidak terlalu signifikan. Parameter *cooling rate* yang rendah berpengaruh dalam proses eksekusi dimana parameter tersebut membuat algoritma Tabu-*Simulated Annealing* cenderung lebih lama dalam proses eksploitasi solusi, sehingga *track* dari penalti cenderung turun. Hal tersebut membuat pencarian solusi menggunakan algoritma *Tabu-Simulated Annealing* mungkin akan terjebak di dalam *local optima*, walaupun hasil dari nilai penalti yang didapatkan lebih baik.

Penggunaan koefisien penurunan suhu yang lebih besar mungkin bisa menghasilkan solusi yang lebih baik pada OpTur 4 karena akan melakukan eksploitasi dan eksplorasi sekaligus, namun pada percobaan eksperimen hasil yang didapatkan lebih buruk karena terbatas oleh iterasi.

Selain pada OpTur 4, perbandingan algoritma *Tabu-Simulated Annealing* dan *Hill Climbing* juga diimplementasikan pada OpTur 5 Grafik pada Gambar 6.10 menunjukkan nilai penalti OpTur 5 yang dihasilkan kedua algoritma. Garis berwarna orange merepresentasikan algoritma *Hill Climbing* sedangkan garis biru merepresentasikan algoritma *Tabu-Simulated Annealing*. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.



Gambar 6.10 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Hill Climbing* pada Optur 5

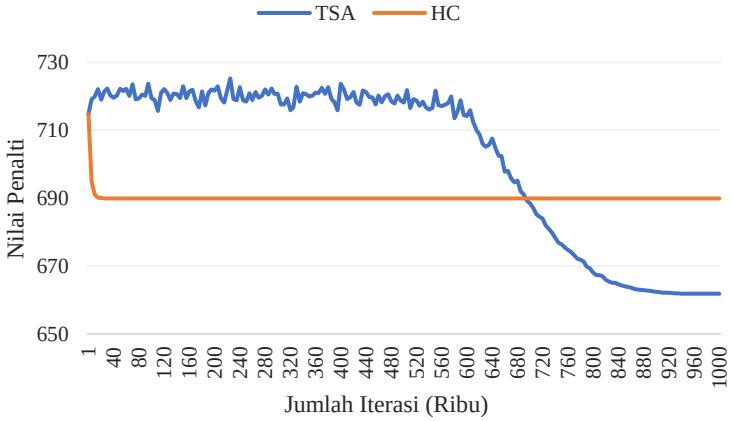
Algoritma *Hill Climbing* cenderung turun lalu stuck pada iterasi ke 50.000. Setelah itu, algoritma tidak mengalami penurunan hingga pada batas iterasi. Hal tersebut disebabkan karena algoritma *Hill Climbing* tidak menemukan nilai penalti yang lebih baik dari solusi sebelumnya, sehingga iterasi yang dilakukan hanya menyimpan nilai penalti yang sama. Sedangkan nilai penalti algoritma *Tabu-Simulated Annealing* melakukan eksplorasi solusi sampai pada iterasi 700.000 lalu turun perlahan hingga mencapai batas iterasi.

Pada OpTur 5 ini, algoritma *Tabu-Simulated Annealing* menggunakan parameter dengan koefisien penurunan suhu yang tinggi sehingga mampu melakukan eksplorasi lebih panjang lalu melakukan eksploitasi pada nilai penalti sampai mendapatkan nilai penalti yang paling optimal. Algoritma *Tabu-Simulated Annealing* juga mampu melakukan diversifikasi atau menerima solusi yang lebih buruk, sehingga mencegah peluang algoritma terjebak dalam *local optima*. Proses diversifikasi pada algoritma Tabu-*Simulated Annealing* terlihat pada awal hingga pertengahan iterasi yang tergambar pada grafik. Garis biru menunjukkan adanya kenaikan dan penurunan lalu turun secara signifikan pada iterasi ke 600.000.

Perbandingan dari kedua algoritma pada OpTur 5 menunjukkan bahwa algoritma *Tabu-Simulated Annealing* bisa menghasilkan penalti lebih rendah, yang artinya performa algoritma *Tabu-Simulated Annealing* dinilai lebih bagus daripada algoritma *Hill Climbing*.

Sama halnya dengan OpTur 5, pada OpTur 7 juga menunjukkan hasil bahwa algoritma *Tabu-Simulated Annealing* memberikan performa yang lebih baik. Terlihat dari Gambar 6.11, perbedaan dari kedua algoritma cukup signifikan. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.

Algoritma *Tabu-Simulated Annealing* masih bisa memberikan penurunan nilai penalti hingga batas iterasi. Sedangkan algortima *Hill Climbing* melakukan sedikit eksplorasi dan stuck pada awal iterasi, serta tidak memberikan perubahan pada nilai penalti yang didapatkan.



Gambar 6.11 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Hill Climbing* pada Optur 7

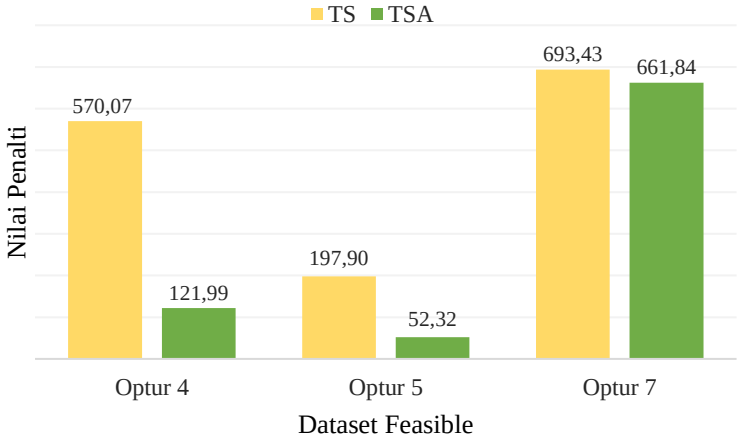
6.6.2. Perbandingan dengan Algoritma *Tabu Search*

Perbandingan performa dilakukan antara algoritma utama *Tabu-Simulated Annealing* dengan algoritma *Tabu Search*.

Tujuannya untuk melihat performa yang lebih baik diantara keduanya. Parameter yang digunakan pada kedua algoritma diambil dari eksperimen skenario yang menghasilkan penalti terbaik. Berikut merupakan perbandingan nilai penalti untuk 3 *dataset feasible* yang dibandingkan dengan algoritma *Tabu Search*.

Hasil perbandingan pada Gambar 6.12 menunjukkan bahwa rata-rata nilai penalti dari algoritma *Tabu-Simulated Annealing* masih lebih unggul jika dibandingkan dengan algoritma *Tabu Search* secara keseluruhan. Sumbu X merupakan OpTur dari dataset yang *feasible*, sumbu Y merupakan nilai penalti. Kemudian label nilai penalti terletak diatas grafik masing-masing OpTur.

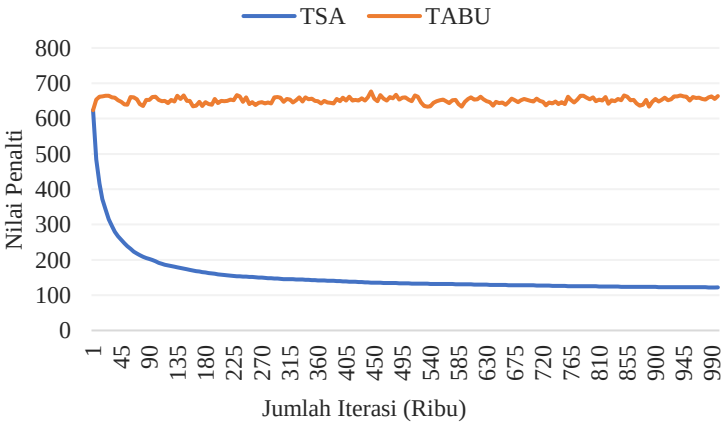
Pada 3 instance *dataset* algoritma Tabu-*Simulated Annealing* mampu menghasilkan nilai penalti yang berada jauh dibawah nilai penalti algoritma *Tabu Search*. Bahkan pada OpTur 4 dan 5 algoritma Tabu-*Simulated Annealing* penaltinya bisa turun lebih dari 50% penalti *Tabu Search*.



Gambar 6.12 Perbandingan Nilai Penalti Algoritma *Tabu-Simulated Annealing* dan *Tabu Search*

Selanjutnya, performa dari algoritma masing-masing OpTur akan dijelaskan lebih detil pada Gambar 6.13 sampai Gambar 6.15.

Grafik *trajectory* pada Gambar 6.13 berguna untuk melihat track dari nilai penalti yang dihasilkan pada 1000.000 iterasi dan menunjukkan apakah hasil dari running algoritma *stuck* pada titik tertentu atau tidak. Grafik tersebut merupakan perbandingan performa dari algoritma Tabu-*Simulated Annealing* dan algoritma *Tabu Search* pada OpTur 4. Garis berwarna *orange* merepresentasikan algoritma *Tabu Search* sedangkan garis biru merepresentasikan algoritma *Tabu-Simulated Annealing*. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.



Gambar 6.13 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Tabu Search* pada Optur 4

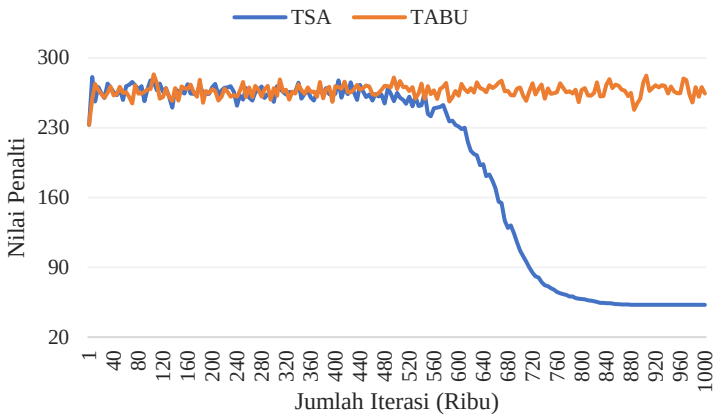
Algoritma *Tabu Search* pada Gambar 6.13 cenderung naik turun tetapi tidak menunjukkan adanya penurunan penalti yang signifikan. Sedangkan algoritma Tabu-*Simulated Annealing* turun tajam sebelum iterasi 50.000. Hasil analisis dari penulis adalah pada algoritma *Tabu Search* sangat dipengaruhi oleh

parameter *tabu list* dan jumlah *low-level heuristics* yang digunakan.

Pada saat proses penyusunan jadwal menggunakan algoritma *Tabu Search*, ketika solusi yang diciptakan tidak lebih baik dari solusi sebelumnya maka *low-level heuristics* akan dimasukkan ke dalam *tabu list*. Ketika ada *low-level heuristics* yang dimasukkan ke dalam *tabu list* maka jumlah *low-level heuristics* yang digunakan untuk random solusi akan semakin berkurang, padahal 1 *low-level heuristics* bisa membuat banyak kombinasi solusi.

Hasil menunjukkan bahwa algoritma *Tabu Search* hanya melakukan eksplorasi terus menerus tanpa sempat melakukan eksploitasi nilai penalti sebuah solusi. Sedangkan untuk algoritma *Tabu-Simulated Annealing* khusus pada OpTur 4 masih sama dengan performa pada perbandingan *Hill Climbing* dimana algoritma ini sangat dipengaruhi oleh koefisien penurunan suhu atau *cooling rate*. Algoritma *Tabu-Simulated Annealing* pada OpTur 4 terlihat bahwa grafiknya cenderung turun dan dianalisis bahwa algoritma ini tidak menerima solusi yang lebih jelek dan mungkin akan terjebak pada *local optima*.

Perbandingan algoritma *Tabu-Simulated Annealing* dan *Tabu Search* juga di uji coba pada OpTur 5. Hasilnya pada Gambar 6.14, menunjukkan bahwa kedua algoritma memberikan perbedaan yang signifikan pada nilai penalti akhir yang didapatkan. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti. Algoritma *Tabu-Simulated Annealing* mampu mendapatkan nilai penalti yang lebih baik. Walaupun keduanya mampu melakukan diversifikasi atau penerimaan solusi yang lebih buruk, tetapi pada algoritma *Tabu Search* terkendala oleh jumlah *low-level heuristics* yang digunakan.

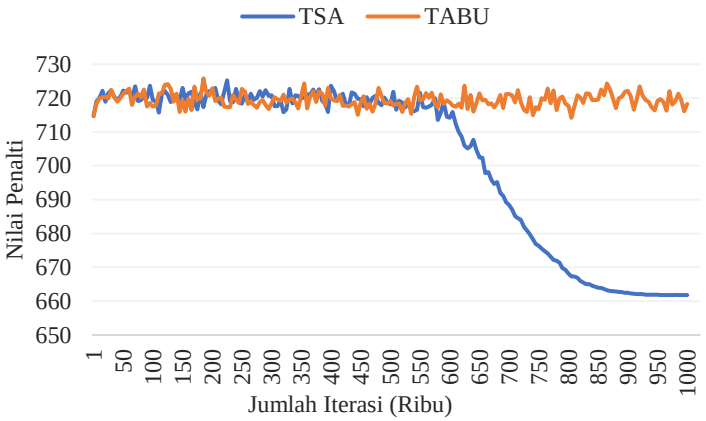


Gambar 6.14 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Tabu Search* pada Optur 5

Seperti pada penjelasan percobaan eksperimen skenario, jumlah *low-level heuristics* yang sedikit membuat kombinasi solusi yang diciptakan juga semakin sempit. Algoritma *Tabu Search* akan lama dalam proses eksplorasi solusi sampai tidak melakukan eksploitasi nilai penalti hingga terbatas pada iterasi. Pada algoritma *Tabu-Simulated Annealing*, mungkin juga terkendala pada saat proses seleksi low-level heuristics yang dilakukan oleh algoritma *Tabu Search* namun algoritma ini dibantu oleh algoritma *Simulated Annealing* sehingga mampu melakukan proses eksplorasi dan ekploitasi nilai penalti secara bersamaan.

Begitu juga pada OpTur 7, percobaan yang dilakukan mendapatkan hasil yang sama dengan OpTur 5 terlihat pada Gambar 6.15. Keduanya memberikan perbedaan nilai penalti yang sangat jauh. Perbedaan perbandingan algoritma pada OpTur 4, OpTur 5, dan OpTur 7 kali ini terletak pada penggunaan koefisien penurunan suhu. Semakin besar *cooling rate* yang digunakan akan membuat algoritma semakin banyak melakukan eksplorasi solusi dan melakukan diversifikasi. Namun, pada OpTur 4 sedikit berbeda karena terbatas oleh

iterasi. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.

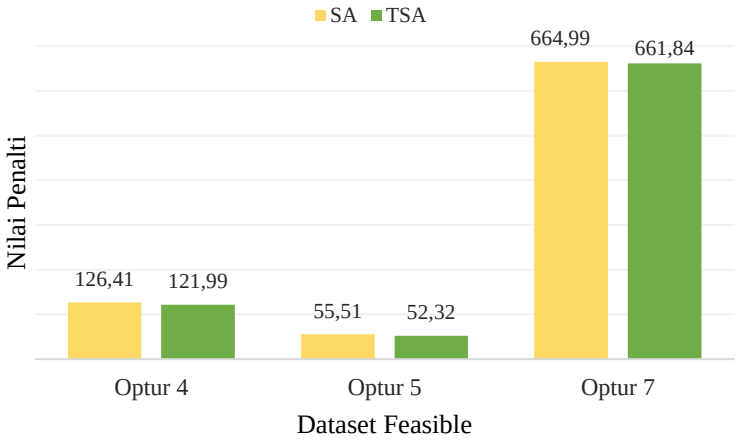


Gambar 6.15 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Tabu Search* pada Optur 7

6.6.3. Perbandingan dengan Algoritma *Simulated Annealing*

Perbandingan performa selanjutnya dilakukan antara algoritma utama *Tabu-Simulated Annealing* dengan algoritma *Simulated Annealing*. Perbedaan keduanya terletak dari parameter Tabu List. Berikut merupakan perbandingan nilai penalti untuk 3 *dataset* feasible yang dibandingkan dengan algoritma *Simulated Annealing* yang ditunjukkan pada Gambar 6.16. Sumbu X merupakan OpTur dari dataset yang *feasible*, sumbu Y merupakan nilai penalti. Kemudian label nilai penalti terletak diatas grafik masing-masing OpTur.

Hasil perbandingan menunjukkan bahwa rata-rata nilai penalti dari algoritma *Tabu-Simulated Annealing* masih lebih unggul jika dibandingkan dengan algoritma *Simulated Annealing* secara keseluruhan. Namun, perbedaan nilai penalti kedua algoritma sangat tipis sekali.

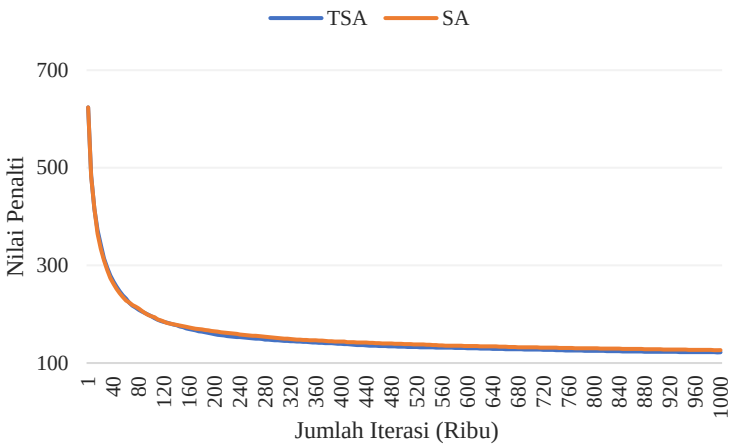


Gambar 6.16 Perbandingan Nilai Penalti Algoritma *Tabu-Simulated Annealing* dan *Simulated Annealing*

Algoritma *Tabu-Simulated Annealing* hanya berbeda 2 hingga 6 angka dari algoritma *Simulated Annealing*. Performa dari algoritma masing-masing OpTur akan dijelaskan lebih detil pada Gambar 6.17 sampai Gambar 6.19.

Grafik *trajectory* pada Gambar 6.17 menunjukkan hasil dari running algoritma untuk 1000.000 iterasi. Hasil penalti dicetak setiap 5000 iterasi. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.

Terlihat bahwa performa antara algoritma Tabu-*Simulated Annealing* dan *Simulated Annealing* menghasilkan rata-rata nilai penalti yang tidak berbeda jauh bahkan sangat tipis. Berdasarkan grafik tersebut, analisis yang dilakukan penulis adalah mencari penyebab diantara kedua algoritma tidak memberikan hasil yang berbeda. Hasil yang ditemukan adalah terletak pada parameter *tabu list* yang digunakan oleh algoritma *Tabu-Simulated Annealing*.



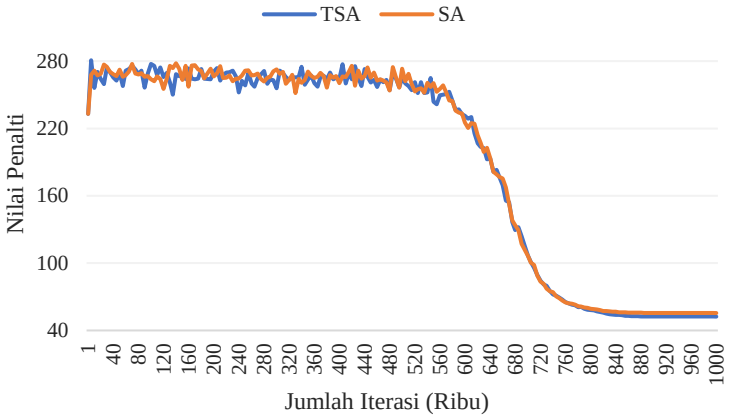
Gambar 6.17 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Simulated Annealing* pada Optur 4

Jumlah *low-level heuristics* yang sedikit membuat penggunaan parameter ini tidak memberikan dampak signifikan seperti yang sudah dijelaskan pada subbab sebelumnya. Maka dari itu hasil dari kedua algoritma memiliki selisih yang sangat sedikit. Namun, berkat adanya algoritma *Tabu Search* yang digabungkan dengan *Simulated Annealing* membuat algoritma utama ini menjadi lebih unggul walaupun sedikit.

Hasil analisis kedua terletak pada *cooling rate* atau koefisien penurunan suhu pada kedua algoritma. Karakteristik *dataset* OpTur 4 membuat *cooling rate* yang rendah menghasilkan penalti lebih baik karena terus melakukan eksploitasi nilai penalti pada solusi. Namun, ada kemungkinan solusinya terjebak dalam *local optima* karena tidak ada proses penerimaan solusi yang lebih buruk sama sekali. Pada OpTur 4 masih mungkin untuk mendapatkan solusi yang lebih baik lagi dengan menemukan *cooling rate* yang sesuai dan juga tidak terbatas oleh iterasi.

Hasil dari perbandingan algoritma *Tabu-Simulated Annealing* dan algoritma *Simulated Annealing* OpTur 5 pada Gambar 6.18

menunjukkan sedikit perbedaan. Grafik *trajectory* dibawah merupakan hasil eksekusi algoritma yang nilai penaltinya dicatat setiap 5000 iterasi dengan batas 1000.000 iterasi. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.



Gambar 6.18 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Simulated Annealing* pada Optur 5

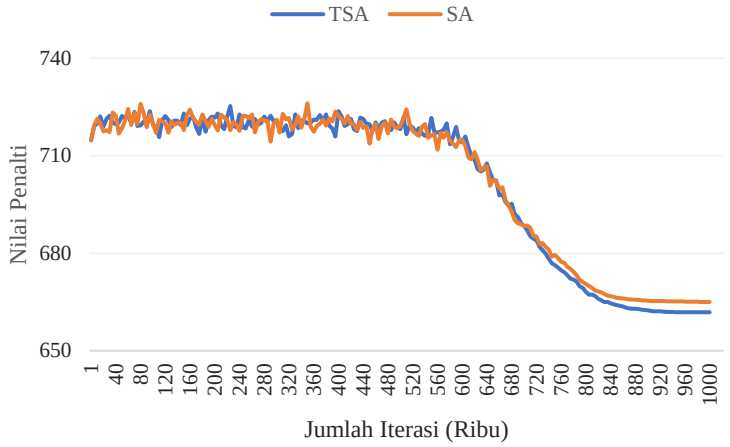
Parameter yang digunakan sama dengan pada percobaan skenario yang memberikan hasil penalti terbaik. Namun, di grafik tersebut terlihat bahwa pada kedua algoritma perbedaan nilai penalti akhir sangat tipis sekali. Hal itu disebabkan oleh parameter yang digunakan algoritma *Tabu-Simulated Annealing* tidak memberikan dampak besar pada penalti yang dihasilkan. Parameter tersebut adalah parameter *tabu list.*

Seperti yang sudah dijelaskan pada subbab sebelumnya, rendahnya jumlah *low-level heuristics* yang digunakan pada penilitan tidak memberikan dampak signifikan. Peluang *low-level heuristics* untuk masuk ke dalam Tabu List dan terpilih pada saat proses *random* akan semakin besar. Sedangkan esensi dari penggunaan *tabu list* adalah untuk menahan supaya *low-level heuristics* yang menghasilkan solusi kurang bagus tidak

terpakai pada iterasi selanjutnya. Terlebih jika skenario panjang *tabu list* yang digunakan memiliki ukuran besar. Maka kombinasi solusi yang diciptakan juga akan semakin sempit.

Pada Gambar 6.19 menunjukkan tidak ada perbedaan jauh antara algoritma *Tabu-Simulated Annealing* dan algoritma *Simulated Annealing* pada OpTur 7. Sumbu X pada grafik menunjukkan jumlah iterasi percobaan, sedangkan sumbu Y menunjukkan nilai penalti.

Kelebihan dari parameter *tabu list* yang dimiliki oleh algoritma *Tabu-Simulated Annealing* memberikan dampak yang tidak signifikan. Walaupun begitu, hasil akhir tetap menunjukkan algoritma *Tabu-Simulated Annealing* memiliki performa yang lebih baik.



Gambar 6.19 Perbandingan Performa Algoritma *Tabu-Simulated Annealing* dan *Simulated Annealing* pada Optur 7

6.6.4. Perbandingan Keseluruhan Algoritma

Perbandingan keseluruhan algoritma dilakukan untuk menemukan algoritma yang memiliki performa terbaik dalam penyelesaian penjadwalan perawat. Performa dinilai baik apabila dapat menghasilkan solusi dengan nilai penalti paling

rendah. Algoritma yang dibandingkan pada subbab sebelumnya kemudian dirangkum menjadi 1 tabel untuk melihat perbandingannya secara keseluruhan.

Tabel 6.32 merupakan hasil perbandingan penalti dari keempat algortima. Dari Tabel 6.32 diketahui bahwa Algoritma *Tabu-Simulated Annealing* memiliki rata-rata nilai penalti terendah yaitu pada OpTur 4 nilai penalti yang dihasilkan sebesar 121.99. Lalu pada OpTur 5 nilai penalti yang dihasilkan mencapai angka 52.32 dan terakhir pada OpTur 7 nilai penalti mencapai 661.84. Hasil perbandingan keseluruhan algoritma terlampir pada Lampiran E Keseluruhan Hasil Penjadwalan

Tabel 6.32 Perbandingan Keseluruhan Algoritma

Dataset	**HC**	**TS**	**SA**	**TSA**
OpTur 4	160.4505	570.0738	126.4086	121.99
OpTur 5	82.5711	197.8952	55.50724	52.32
OpTur 7	689.9153	693.4251	664.9901	661.84

6.6.5. Perbandingan dengan Hasil *Benchmark Dataset*

Algoritma yang memiliki performa paling optimum dari hasil eksperimen dan uji coba kemudian dibandingkan dengan hasil *benchmark dataset.* Hasil *benchmark dataset* merujuk pada artikel utama yang dijadikan acuan dalam pengerjaan penelitian tugas akhir. Artikel tersebut berjudul *A Hybrid Approach for Solving Real-World Nurse Rostering Problems* yang ditulis pada tahun 2011[5].

Pada Tabel 6.33 menunjukkan perbandingan nilai penalti dari penelitian tugas akhir dan penelitian sebelumnya pada *benchmark dataset*. Penelitian tugas akhir menggunakan *hybrid* algoritma *Tabu-Simulated Annealing* sedangkan pada *benchmark dataset* menggunakan *hybrid* algoritma *Iterated Local Search* dan *Variable Neighborhood Descent*.

Dari perbandingan nilai penalti pada Tabel 6.33 dapat dilakukan analisis bahwa algoritma *Tabu-Simulated Annealing* yang diimplementasikan mampu menyelesaikan 3 dari 7 *instance* pada *dataset Norwegian Hospitals*. Sedangkan algoritma *Variable Neighborhood Descent* mampu menyelesaikan keseluruhan *instance dataset.* Artinya, performa algoritma *Tabu-Simulated Annealing* yang digunakan pada penelitian tugas akhir masih belum mampu menyelesaikan permasalahan pada *dataset* secara keseluruhan.

Tabel 6.33 Perbandingan dengan Hasil *Benchmark Dataset*

Dataset	**TSA**	**VND**
OpTur 1	-	17
OpTur 2	-	3.09
OpTur 3	-	81.7
OpTur 4	121.99	2.48
OpTur 5	52.32	8.34
OpTur 6	-	1.83
OpTur 7	661.84	156

Pada penelitian sebelumnya, dilakukan percobaan sebanyak 66 kali tiap eksperimen. Sedangkan pada penelitian tugas akhir, hanya dilakukan percobaan sebanyak 10 kali tiap eksperimen. Hal tersebut bisa menjadi salah satu penyebab dari perbedaan performa yang didapatkan dari kedua implementasi algoritma baik dari penelitian tugas akhir maupun dari penelitian sebelumnya.

Algoritma *Variable Neighborhood Descent* yang digunakan pada penelitian *benchmark dataset* lebih unggul dibandingkan algoritma *Tabu-Simulated Annealing* yang diimplementasikan pada penelitian tugas akhir. Hal tersebut diketahui dari nilai penalti yang lebih rendah dari penelitian tugas akhir yang dilakukan penulis. Implementasi algoritma *Tabu-Simulated*

Annealing menggunakan bobot penalti yang diasumsikan bernilai 1 dikarenakan keterbatasan informasi dari penelitian *benchmark dataset* sebelumnya. Hal tersebut menyebabkan perbedaan dari hasil nilai penalti yang didapatkan pada beberapa *insctance dataset* yang *feasible* yaitu pada OpTur 4, OpTur 5 dan OpTur 7.

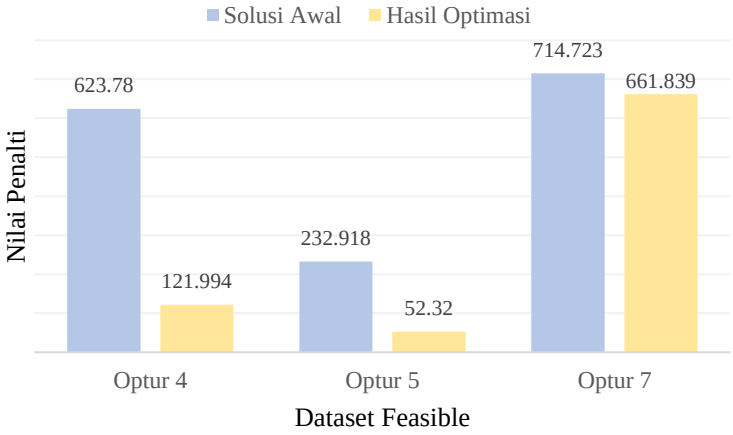
6.7. Hasil Solusi Optimum

Hasil solusi optimum didapatkan setelah melalui berbagai eksperimen, dimulai dari eksperimen percobaan skenario untuk menemukan skenario yang sesuai. Eksperimen skenario dengan parameter yang menghasilkan penalti paling rendah akan di uji coba dengan dibandingkan algoritma yang lain. Contoh keluaran hasil optimasi solusi terlampir pada Lampiran C Hasil Optimasi Solusi. Hasil tersebut masih berupa nomor *shift*, nomor *shift* akan dikonversi menjadi jadwal kerja seperti pada Lampiran D Hasil Jadwal Kerja Perawat.

Secara umum, algoritma *Tabu-Simulated Annealing* lebih unggul dibandingkan algoritma lain pada 3 *dataset* yang *feasible* yaitu OpTur 4, OpTur 5, dan OpTur 7. Penurunan paling banyak terdapat pada OpTur 4 yang mampu mereduksi nilai penalti sebanyak 80% yang awalnya 623,78 menjadi 121.994 Diikuti oleh OpTur 5 yang mampu mereduksi nilai penalti sebanyak 78% dari 232.918 menjadi 52.32 dan terakhir penurunan terendah dari implementasi algoritma terjadi pada OpTur 7 yang mampu mereduksi nilai penalti sebanyak 7% yang awalnya 714.723 menjadi 661.839.

Keseluruhan hasil optimasi baik dari solusi awal hingga optimasi solusi terlampir pada Lampiran E Keseluruhan Hasil Penjadwalan. Viasualisasi dari hasil solusi optimum yang dibandingkan dengan penalti awal penelitian ditunjukkan pada Gambar 6.20. Sumbu X merupakan OpTur dari dataset yang

feasible, sumbu Y merupakan nilai penalti. Kemudian label nilai penalti terletak diatas grafik masing-masing OpTur.



Gambar 6.20 Perbandingan Penalti Solusi Awal dan Penalti Optimasi Solusi

Berdasarkan hasil optimasi yang didapatkan, penulis melakukan analisis terhadap nilai penalti hasil solusi optimum untuk melihat pelanggaran terhadap *soft constraint* dan jumlah penalti dari setiap *soft constraint* yang dilanggar. Tabel 6.34 menunjukkan daftar pelanggaran terhadap *soft constraint* dari solusi optimum. Secara umum, hasil optimasi *dataset feasible* mampu mengurangi pelanggaran *soft constraint* sehingga nilai penalti yang didapatkan lebih rendah.

Nilai penalti 0 menunjukkan bahwa tidak ada pelanggaran *soft constraint* seperti pada *soft constraint* 2, *soft constraint* 4, dan *soft constraint* 9. Pada OpTur 4 pelanggaran terjadi pada *soft constraint* 1, 3, 6, dan 7. Pada OpTur 5 pelanggaran terjadi pada *soft constraint* 1, 3, 5, 6, dan 7. Lalu pada Optur 7 pelanggaran *soft constraint* terdapat pada *soft constraint* 1, 3, 6, 7, dan 8.

Tabel 6.34 Daftar Pelanggaran Soft Constraint Solusi Optimum

Soft Constraint	OpTur 4	OpTur 5	OpTur 7
SC 1	2.9	0.6	0.4
SC 2	0	0	0
SC 3	9	6.6	1.8
SC 4	0	0	0
SC 5	0	23.54836	0
SC 6	1.692276	0.961871	9.648431
SC 7	108.4013	20.61243	19.99025
SC 8	0	0	630
SC 9	0	0	0

Pelanggaran terhadap *soft constraint* 1 yaitu terdapat terlalu banyak hari kerja berurutan pada kategori *shift* yang sama. Pelanggaran terhadap *soft constraint* 3 yaitu terlalu sedikit hari kerja berurutan pada kategori *shift* yang sama. Pelanggaran terhadap *soft constraint* 5 yaitu menyimpang dari jumlah minimum dan maksimum *shift* di setiap kategori *shift*.

Hampir keseluruhan *dataset feasible* melanggar *soft constraint* 6 dan 7. *Soft constraint* 6 mengatur tentang penyimpangan dari jam kerja perawat artinya pada setiap OpTur terdapat jam kerja perawat yang menyimpang dari kontrak kerja. Sedangkan *soft constraint* 7 mengatur tentang pengelompokan waktu libur perawat, artinya terdapat perawat yang waktu liburnya terpisah sehingga mendapatkan nilai penalti.

Pada Optur 7 terdapat pelanggaran *soft constraint* 8 yang nilainya sangat besar yaitu 630. *Soft constraint* 8 mengatur tentang pola yang diinginkan perawat. Munculnya nilai penalti yang sangat besar dikarenakan pola yang diinginkan perawat pada OpTur 7 bertentangan dengan *hard constraint* 5 yaitu adanya pasangan *shift* yang dilarang secara berurutan karena jarak antar *shift* yang terlalu pendek. Keseluruhan rekap hasil perhitungan pelanggaran *soft constraint* dapat dilihat pada Lampiran E Keseluruhan Hasil Penjadwalan.

BAB VII
KESIMPULAN DAN SARAN

Bab ini akan menjelaskan kesimpulan dari keseluruhan penelitian dan eksperimen yang dilakukan pada tugas akhir. Selain itu, akan ada saran untuk menunjang penelitian selanjutnya.

7.1. Kesimpulan

Berdasarkan penelitian dan eksperimen yang telah dilakukan pada bagian sebelumnya, dapat disimpulkan bahwa:

1. Algoritma Tabu-*Simulated Annealing* mampu menyelesaikan permasalahan penjadwalan perawat pada *Dataset* Norwegian Hospitals. Jumlah *dataset* yang mampu dijadwalkan adalah 3 dari 7 *dataset* dengan rincian OpTur 4, OpTur 5, dan OpTur 7.
2. Performa algoritma Tabu-*Simulated Annealing* lebih unggul jika dibandingkan dengan algoritma Hill Climbing, *Tabu Search* dan *Simulated Annealing*.
3. Algoritma Tabu-*Simulated Annealing* Hyper Heuristics mampu mereduksi penalti solusi awal, diantaranya:
    a. Rata-rata nilai penalti OpTur 4 berhasil diturunkan sebanyak 80% dari solusi awal yang awalnya 623.78 menjadi 121.994
    b. Rata-rata nilai penalti OpTur 5 berhasil diturunkan sebanyak 78% dari solusi awal yang awalnya 232.918 menjadi 52.32
    c. Rata-rata nilai penalti OpTur 7 berhasil diturunkan sebanyak 7% dari solusi awal yang awalnya 714.723 menjadi 661.839
4. Pada algoritma Tabu-*Simulated Annealing* berbasis Hyper Heuristics, terdapat beberapa parameter yang mempengaruhi hasil, diantaranya:

a. Koefisien penurunan suhu atau *Cooling Rate* memberikan pengaruh cukup besar pada hasil penalti yang diimplementasikan. Semakin rendah *cooling rate* maka algoritma akan cenderung melakukan eksploitasi. Sedangkan semakin tinggi *cooling rate* maka algoritma akan melakukan proses eksplorasi lebih lama sehingga hasil yang didapatkan lebih baik.
b. Parameter *Tabu List* tidak memberikan hasil yang signifikan dikarenakan jumlah dari *low-level heuristics* yang sedikit sehingga metode untuk pencarian solusi terbatas. Semakin sedikit jumlah *low-level heuristics*, maka akan semakin besar peluang *low-level heuristics* untuk terpilih maupun masuk *tabu list*, penahanan *low-level heuristics* pada *tabu list* menjadi sangat singkat dan tidak memberikan dampak besar. Jika *tabu list* yang diimplementasikan digunakan untuk menyimpan solusi maka akan ada terlalu banyak kemungkinan untuk memasukkan solusi ke dalam *tabu list* karena luasnya *search area* dan terlalu banyak variasi dari solusi yang dihasilkan dari hasil *random* solusi.

5. Durasi implementasi algoritma pada masing-masing *dataset* tergantung pada ukuran dari *dataset*, semakin besar *dataset* maka akan semakin lama waktu eksekusi yang dibutuhkan.

7.2. Saran

Berdasarkan hasil dan kesimpulan tersebut, saran yang dapat diberikan untuk penelitian selanjutnya diantaranya:

1. Penelitian belum mampu menghasilkan solusi yang *feasible* untuk 7 instance *dataset*, hal tersebut bisa disebabkan oleh performa algoritma yang belum mampu diterapkan untuk keseluruhan *dataset*. Perbaikan algoritma bisa dilakukan untuk menghasilkan solusi *feasible* atau

melakukan percobaan berkali-kali hingga *feasible* karena penelitian tidak terlepas dari faktor *random.*

2. Penelitian hanya menggunakan 13 kombinasi parameter yang digunakan dalam melakukan eksperimen. Kedepannya, penelitian dapat menggunakan lebih banyak variasi parameter lain dan bisa menemukan kombinasi yang paling sesuai.
3. Penelitian dengan menggunakan *Tabu Search* kurang cocok diimplementasikan pada permasalahan apabila jumlah low-level heuristics yang digunakan sedikit. Untuk mendapatkan hasil yang lebih baik, jumlah low-level heuristics dapat ditambahkan sehingga peluang menghasilkan solusi lebih optimum akan lebih besar.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] R. F. Nova, "Pengaruh Kualitas Layanan Terhadap Kepuasan Pasien Rawat Inap pada Rumah Sakit PKU Muhammadiyah Surakarta," no. 45, p. 39, 2010.

[2] G. L. Goldblatt, "Joint Commission Journal on Quality and Patient Safety: Introduction," *Jt. Comm. J. Qual. Patient Saf.*, vol. 33, no. 10, p. 591, 2007.

[3] K. Konoralma, L. Moningka, and S. Palamani, "Hubungan Shift Kerja Perawat Dengan Stres Kerja Di Ruang Irdm Blu Rsup Prof Dr. R. D. Kandou Manado," *J. Ilm. Perawat Manad.*, vol. 2, no. 1, pp. 16–24, 2013.

[4] O. Article, "Effect of Shift Work on the Frequency of Depression in Nursing Staff of Yazd University of Medical Sciences," *J. Community Heal. Res.*, vol. 1, no. 2, pp. 104–109, 2012.

[5] 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.

[6] I. Stoilkovska, "Solving the Nurse Rostering Problem," *Tech. Univ. Dresden*, pp. 1–11, 2013.

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

[8] L. Rosocha, S. Vernerová, and R. Verner, "Medical staff scheduling using simulated annealing," *Qual. Innov. Prosper.*, vol. 19, no. 1, pp. 1–11, 2015.

[9] E. K. Burke, G. Kendall, and E. Soubeiga, "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *J. Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[10] A. Marom, “Optimasi Kuliah Otomatis Menggunakan Algoritma Tabu- Simulated Annealing Hyper-Heuristics Automated Course Timetabling Tabu-Simulated Annealing Hyper-Heuristics Algorithm,” 2019.

[11] 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.

[12] Z. Maulidati, “Optimasi Penjadwalan Staf Rumah Sakit dengan Menggunakan Algoritma Tabu Search Based Hyper-Heuristics (Studi Kasus: Rumah Sakit Ibu dan Anak Kendangsari),” 2017.

[13] E.-G. Talbi, *Metaheuristics From Design To Implementation*, vol. 66. 2012.

[14] L. Lilham, A. Aman, and F. Hanum, “Model Penjadwalan Perawat Di Rumah Sakit,” *J. Math. Its Appl.*, vol. 8, no. 2, p. 11, 2009.

[15] A. Afandi, L. Setiawati, and D. Mufti, “Penjadwalan Shift Kerja Perawat Dengan Mempertimbangkan Tingkat Kemampuan Perawat dan Kebutuhan Hari Libur.”

[16] S. A. Simanjuntak, “Penyelesaian Nurse Rostering Problem dengan Modified Harmony Search Algorithm,” 2018.

[17] M. Stølevik, T. Eric, and A. Riise, “A mathematical model for the nurse rostering problem,” *SINTEF Tech. Rep. no. A19133*, pp. 1–9, 2011.

[18] N. D. Angresti, A. Djunaidy, and A. Mukhlason, “Penerapan Hiperheuristik Berbasis Metode Simulated Annealing untuk Penyelesaian Permasalahan Optimasi Lintas Domain,” *J. Nas. Teknol. dan Sist. Inf.*, vol. 5, no.

1, pp. 33–40, 2019.

[19] Miswanto, F. Pernando, and I. Aditya Firmansyah, "Implementasi Algoritma Tabu Search Untuk Mengoptimasi Penjadwalan Preventive Maintenancept Solusi Aplikasi Interaktif," vol. 2018, no. Sentika, pp. 23–24, 2018.

[20] F. Chahyadi, A. SN, and H. Kurniawan, "Hospital Nurse Scheduling Optimization Using Simulated Annealing and Probabilistic Cooling Scheme," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 12, no. 1, p. 21, 2018.

[21] H. A. S, C. Farela, and M. Tantrika, "Implementasi Algoritma Simulated Annealing pada Penjadwalan Produksi untuk Meminimasi Makespan (Studi Kasus di PT . Gatra Mapan , Karang Ploso , Malang)," *J. Rekayasa Dan Manaj. Sist. Ind.*, vol. 3, no. 1, pp. 43–52, 2015.

Halaman ini sengaja dikosongkan

Lampiran A Hasil Solusi Awal

- Hasil Solusi Awal Optur 5

E\D	1	2	3	4	5	6	7	8	9	10
1	8	3	6	0	0	0	0	8	6	5
2	7	5	4	7	2	0	0	4	4	4
3	0	7	8	2	0	0	0	0	0	4
4	8	2	2	0	0	2	2	0	3	6
5	0	0	0	8	6	0	0	0	4	4
6	0	6	8	4	3	7	4	3	1	1
7	2	0	0	0	4	0	0	1	1	0
8	1	2	0	8	8	1	7	4	4	8
9	6	9	3	3	7	0	0	8	7	9
10	5	4	4	1	1	0	0	4	0	1
11	0	0	0	2	0	0	0	2	0	0
12	4	0	1	1	4	4	4	2	2	2
13	2	0	9	5	8	0	0	6	0	8
14	1	4	4	4	4	0	0	0	0	0
15	0	8	0	0	0	0	0	0	9	0
16	0	0	0	9	0	0	0	0	5	7
17	0	1	5	0	5	0	0	5	8	0
18	4	1	2	0	0	4	1	1	8	2
19	3	8	7	6	2	0	0	7	0	3
20	4	4	1	4	1	0	0	0	2	0

E\D	11	12	13	14	15	16	17	18	19	20
1	6	4	0	0	5	9	5	6	0	2
2	4	0	4	1	0	2	0	1	2	0
3	1	4	7	7	0	1	4	2	0	0
4	2	2	0	0	4	3	4	4	1	0
5	8	8	0	0	7	7	8	0	0	4
6	3	4	0	0	8	4	1	0	4	0
7	1	1	1	4	0	8	7	4	8	0
8	4	7	0	0	4	0	1	8	3	0
9	9	0	0	0	6	1	0	7	0	1
10	4	1	4	4	1	4	8	2	0	0
11	8	6	0	0	0	0	0	9	8	0
12	2	0	0	0	2	0	4	1	4	0
13	0	5	0	0	3	0	2	0	5	7
14	0	0	0	0	0	4	0	0	0	0
15	0	0	0	0	0	0	3	4	2	0
16	0	0	0	0	8	0	2	0	0	0
17	7	0	0	0	0	5	9	8	1	0
18	0	2	0	0	1	8	6	5	4	0
19	5	3	0	0	4	6	0	0	7	4
20	0	8	2	2	2	2	0	3	6	0

E\D	21	22	23	24	25	26	27	28
1	2	0	0	0	7	3	0	0
2	0	1	0	4	2	2	0	0
3	0	1	2	0	1	0	0	0
4	0	0	4	4	3	1	0	0
5	4	2	0	2	2	0	0	0
6	0	4	4	1	1	4	0	0
7	0	2	0	0	4	6	0	0
8	0	4	3	5	8	4	0	0
9	4	5	5	3	0	0	0	0
10	0	4	7	4	4	0	0	0
11	0	0	9	0	9	0	1	1
12	0	0	1	1	0	2	0	0
13	7	7	8	9	8	0	0	0
14	0	0	2	0	0	0	4	4
15	0	0	6	6	0	8	4	4
16	0	6	8	7	0	5	7	7
17	0	8	0	2	0	7	2	2
18	0	3	4	8	5	4	0	0
19	1	0	0	0	6	8	0	0
20	0	8	1	8	4	1	0	0

Lampiran B Hasil Eksperimen Skenario

- Hasil Eksperimen Skenario Optur 5

P	INIT	A	Δ	B	Δ	C	Δ
P1	232.91	57.78	75%	81.43	65%	52.17	78%
P2		51.08	78%	69.37	70%	48.37	79%
P3		53.90	77%	74.47	68%	53.38	77%
P4		50.89	78%	73.12	69%	46.04	80%
P5		47.67	80%	90.28	61%	47.89	79%
P6		52.68	77%	86.39	63%	51.54	78%
P7		58.07	75%	78.41	66%	55.96	76%
P8		51.54	78%	74.57	68%	61.49	74%
P9		52.93	77%	68.63	71%	56.67	76%
P10		49.41	79%	68.58	71%	50.77	78%
AVG		52.60	77%	76.52	67%	52.43	77%

P	INIT	D	Δ	E	Δ	F	Δ
P1	232.91	51.39	78%	51.67	78%	69.56	70%
P2		53.06	77%	51.46	78%	89.58	62%
P3		56.95	76%	52.67	77%	74.50	68%
P4		57.57	75%	49.32	79%	63.58	73%
P5		47.16	80%	51.99	78%	77.58	67%
P6		51.07	78%	57.67	75%	57.18	75%
P7		56.08	76%	59.31	75%	69.92	70%
P8		57.16	75%	51.13	78%	76.80	67%
P9		51.28	78%	51.93	78%	74.28	68%
P10		50.62	78%	46.07	80%	74.56	68%
AVG		53.23	77%	52.32	78%	72.75	69%

P	INIT	G	Δ	H	Δ	I	Δ
P1	232.918	72.74	69%	72.45	69%	53.83	77%
P2		68.75	70%	67.74	71%	55.26	76%
P3		63.86	73%	63.90	73%	52.74	77%
P4		71.95	69%	61.79	73%	49.22	79%
P5		82.36	65%	67.40	71%	52.36	78%
P6		78.48	66%	60.17	74%	53.18	77%
P7		68.17	71%	75.15	68%	56.64	76%
P8		79.99	66%	68.24	71%	55.02	76%
P9		79.24	66%	81.41	65%	53.91	77%
P10		72.87	69%	70.66	70%	49.25	79%
AVG		73.84	68%	68.89	70%	53.14	77%

P	J	Δ	K	Δ	L	Δ	M	Δ
P1	54.16	77%	59.62	74%	53.17	77%	55.24	76%
P2	55.88	76%	55.79	76%	58.49	75%	54.46	77%
P3	51.49	78%	52.32	78%	59.57	74%	53.24	77%
P4	54.14	77%	45.65	80%	54.36	77%	49.34	79%
P5	54.18	77%	48.23	79%	50.87	78%	56.65	76%
P6	44.83	81%	51.41	78%	50.16	78%	53.43	77%
P7	54.31	77%	55.55	76%	49.16	79%	58.74	75%
P8	51.68	78%	54.66	77%	60.09	74%	47.41	80%
P9	50.48	78%	50.03	79%	51.09	78%	52.96	77%
P10	54.95	76%	56.18	76%	56.46	76%	51.93	78%
AVG	52.61	77%	52.94	77%	54.34	77%	53.34	77%

Lampiran C Hasil Optimasi Solusi

- Hasil Optimasi Solusi Optur 5

E\D	1	2	3	4	5	6	7	8	9	10
1	4	9	0	0	0	0	0	8	8	5
2	0	5	4	4	0	0	0	4	4	6
3	8	3	6	2	2	0	0	0	0	0
4	4	2	2	0	0	2	2	0	4	4
5	2	0	0	0	8	0	0	8	8	4
6	6	4	0	7	4	1	7	4	0	4
7	0	0	8	6	3	0	0	2	2	0
8	3	2	0	8	6	4	1	4	4	8
9	8	8	7	3	5	0	0	6	9	9
10	1	4	4	5	1	0	0	1	0	1
11	0	0	0	0	0	0	0	2	2	0
12	4	4	1	0	4	4	4	0	1	2
13	2	0	9	4	8	0	0	7	5	8
14	0	1	4	4	1	0	0	0	0	0
15	0	0	1	1	0	0	0	0	0	0
16	0	0	5	9	7	0	0	0	3	3
17	5	1	8	0	0	0	0	5	6	7
18	1	8	0	1	4	7	4	0	1	2
19	0	7	3	8	0	0	0	3	7	1
20	7	6	2	2	2	0	0	1	0	0

E\D	11	12	13	14	15	16	17	18	19	20
1	9	7	0	0	3	5	9	8	0	2
2	4	0	4	4	2	2	0	1	1	0
3	1	8	7	7	4	0	0	1	8	0
4	2	2	0	0	1	1	3	3	7	0
5	3	3	0	0	4	7	2	0	0	4
6	4	4	0	0	4	4	1	4	4	0
7	0	0	1	1	0	3	4	7	1	0
8	4	5	0	0	8	8	5	5	4	0
9	0	0	0	0	5	9	8	8	0	1
10	8	4	4	4	0	6	4	2	0	0
11	0	0	0	0	0	0	0	9	3	0
12	0	1	0	0	1	2	2	2	0	0
13	6	6	0	0	0	8	8	0	5	7
14	0	0	0	0	0	0	0	0	2	0
15	0	0	0	0	0	0	6	6	2	0
16	7	8	0	0	0	0	0	0	0	0
17	1	1	0	0	7	4	7	0	0	0
18	2	0	0	0	8	1	4	4	4	0
19	5	4	0	0	6	4	0	0	8	4
20	8	2	2	2	2	0	1	4	6	0

E\D	21	22	23	24	25	26	27	28
1	2	0	0	6	6	5	0	0
2	0	1	1	1	2	2	0	0
3	0	2	2	0	0	0	0	0
4	0	0	4	4	9	1	0	0
5	4	4	0	0	0	2	0	0
6	0	8	3	4	7	1	0	0
7	0	2	2	0	4	7	0	0
8	0	4	6	4	1	0	0	0
9	1	5	7	9	0	0	0	0
10	0	4	4	8	4	4	0	0
11	0	7	9	8	8	0	1	1
12	0	1	1	2	2	0	0	0
13	7	8	8	0	0	0	0	0
14	0	0	0	0	1	4	4	4
15	0	0	5	3	0	6	4	4
16	0	0	0	2	0	8	7	7
17	0	0	0	0	8	8	2	2
18	0	3	4	1	5	4	0	0
19	4	6	0	7	3	0	0	0
20	0	0	8	5	4	3	0	0

Lampiran D Hasil Jadwal Kerja Perawat

- Hasil Jadwal Kerja Perawat OpTur 5

E\D	1	2	3	4	5	6	7	8	9	10
1	D	D9	FE	FE	FE	FE	FE	E2	E2	D3
2	FE	D3	D	D	FE	FE	FE	D	D	D1
3	E2	D8	D1	N	N	FE	FE	FE	FE	FE
4	D	N	N	FE	FE	N	N	FE	D	D
5	N	FE	FE	FE	E2	FE	FE	E2	E2	D
6	D1	D	FE	D2	D	E	D2	D	FE	D
7	FE	FE	E2	D1	D8	FE	FE	N	N	FE
8	D8	N	FE	E2	D1	D	E	D	D	E2
9	E2	E2	D2	D8	D3	FE	FE	D1	D9	D9
10	E	D	D	D3	E	FE	FE	E	FE	E
11	FE	FE	FE	FE	FE	FE	FE	N	N	FE
12	D	D	E	FE	D	D	D	FE	E	N
13	N	FE	D9	D	E2	FE	FE	D2	D3	E2
14	FE	E	D	D	E	FE	FE	FE	FE	FE
15	FE	FE	E	E	FE	FE	FE	FE	FE	FE
16	FE	FE	D3	D9	D2	FE	FE	FE	D8	D8
17	D3	E	E2	FE	FE	FE	FE	D3	D1	D2
18	E	E2	FE	E	D	D2	D	FE	E	N
19	FE	D2	D8	E2	FE	FE	FE	D8	D2	E
20	D2	D1	N	N	N	FE	FE	E	FE	FE

E\D	11	12	13	14	15	16	17	18	19	20
1	D9	D2	FE	FE	D8	D3	D9	E2	FE	N
2	D	FE	D	D	N	N	FE	E	E	FE
3	E	E2	D2	D2	D	FE	FE	E	E2	FE
4	N	N	FE	FE	E	E	D8	D8	D2	FE
5	D8	D8	FE	FE	D	D2	N	FE	FE	D
6	D	D	FE	FE	D	D	E	D	D	FE
7	FE	FE	E	E	FE	D8	D	D2	E	FE
8	D	D3	FE	FE	E2	E2	D3	D3	D	FE
9	FE	FE	FE	FE	D3	D9	E2	E2	FE	E
10	E2	D	D	D	FE	D1	D	N	FE	FE
11	FE	FE	FE	FE	FE	FE	FE	D9	D8	FE
12	FE	E	FE	FE	E	N	N	N	FE	FE
13	D1	D1	FE	FE	FE	E2	E2	FE	D3	D2
14	FE	FE	FE	FE	FE	FE	FE	FE	N	FE
15	FE	FE	FE	FE	FE	FE	D1	D1	N	FE
16	D2	E2	FE	FE	FE	FE	FE	FE	FE	FE
17	E	E	FE	FE	D2	D	D2	FE	FE	FE
18	N	FE	FE	FE	E2	E	D	D	D	FE
19	D3	D	FE	FE	D1	D	FE	FE	E2	D
20	E2	N	N	N	N	FE	E	D	D1	FE

E\D	21	22	23	24	25	26	27	28
1	N	FE	FE	D1	D1	D3	FE	FE
2	FE	E	E	E	N	N	FE	FE
3	FE	N	N	FE	FE	FE	FE	FE
4	FE	FE	D	D	D9	E	FE	FE
5	D	D	FE	FE	FE	N	FE	FE
6	FE	E2	D8	D	D2	E	FE	FE
7	FE	N	N	FE	D	D2	FE	FE
8	FE	D	D1	D	E	FE	FE	FE
9	E	D3	D2	D9	FE	FE	FE	FE
10	FE	D	D	E2	D	D	FE	FE
11	FE	D2	D9	E2	E2	FE	E	E
12	FE	E	E	N	N	FE	FE	FE
13	D2	E2	E2	FE	FE	FE	FE	FE
14	FE	FE	FE	FE	E	D	D	D
15	FE	FE	D3	D8	FE	D1	D	D
16	FE	FE	FE	N	FE	E2	D2	D2
17	FE	FE	FE	FE	E2	E2	N	N
18	FE	D8	D	E	D3	D	FE	FE
19	D	D1	FE	D2	D8	FE	FE	FE
20	FE	FE	E2	D3	D	D8	FE	FE

Lampiran E Keseluruhan Hasil Penjadwalan

Seluruh kode program implementasi dapat diakses pada link: https://bit.ly/LampiranKodeE

Seluruh hasil penjadwalan dapat diakses pada link: https://bit.ly/LampiranHasilE

BIODATA PENULIS

Penulis lahir di Mojokerto pada tanggal 5 Agustus 1998 dengan nama Sisca Threecya Agatha. Penulis merupakan anak terakhir dari tiga bersaudara. Orang tua penulis bernama Siswanto dan Lilik Juwariyah. Penulis menempuh Pendidikan dasar di SDN 1 Cepokolimo II pada tahun 2004 dan lulus pada tahun 2010. Kemudian penulis melanjutkan pendidikan sekolah menengah pertama di SMPN 1 Pacet pada tahun 2010 hingga 2013. Pada tahun yang sama, penulis melanjutkan pendidikan formal sekolah menengah atas di SMAN 1 Sooko Mojokerto dan lulus pada tahun 2016. Selanjutnya, pada tahun 2016 penulis melanjutkan pendidikan tinggi di Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Penulis masuk pendidikan tinggi melalui jalur Seleksi Nasional Masuk Perguruan Tinggi Negeri.

Penulis mengambil bidang minat Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB) di Departemen Sistem Informasi ITS. Selama menjalani masa perkuliahan di ITS, penulis aktif mengikuti beberapa lomba, seperti pada tahun 2019 penulis menjadi finalis pada perlombaan Gemastik XII cabang Karya Tulis Ilmiah. Penulis juga aktif mengikuti berbagai pelatihan diantaranya SAP University Alliance Course yang diadakan di Departemen Sistem Informasi. Penulis aktif mengikuti kegiatan kampus seperti pada kepanitiaan Information Systems Expo (ISE! 2018) penulis pernah menjabat sebagai Bendahara.

Lalu organisasi lainnya yang sempat diikuti penulis seperti Himpunan Mahasiswa Sistem Informasi pada tahun 2017 dan 2018, dan masih banyak organisasi lainnya. Selain itu, penulis juga menambah ilmu dan pengalaman dengan mendaftar menjadi asisten dosen beberapa mata kuliah dan pernah menjalani kerja praktik.

Informasi lebih detil terkait penulis, dapat menghubungi melalui email sisca356@gmail.com