Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

NURSE ROSTERING OPTIMIZATION USING REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC ALGORITHM ON BENCHMARK DATASET NORWEGIAN HOSPITAL

DIMAS RIZAL KUSUMA SHINDU
NRP 05211640000094

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

Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

DIMAS RIZAL KUSUMA SHINDU
NRP 05211640000094

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

Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS 184853

NURSE ROSTERING OPTIMIZATION USING REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC ALGORITHM ON BENCHMARK DATASET NORWEGIAN HOSPITAL

DIMAS RIZAL KUSUMA SHINDU
NRP 05211640000025

SUPERVISOR :
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

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

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING – SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

TUGAS AKHIR

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

Oleh:

DIMAS RIZAL KUSUMA SHINDU
NRP. 05211640000094

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 PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING – SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

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 :

DIMAS RIZAL KUSUMA SHINDU
NRP. 05211640000094

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., MT. **(Penguji I)**

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

Halaman ini sengaja dikosongkan

OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

Nama Mahasiswa	**: Dimas Rizal Kusuma Shindu**
NRP	**: 05211640000094**
Jurusan	**: Sistem Informasi FTEIC-ITS**
Pembimbing 1	**: Ahmad Muklason, S.Kom., M.Sc., Ph.D.**

ABSTRAK

Nurse Rostering adalah proses penyusunan jadwal kerja perawat rumah sakit dalam jangka waktu tertentu. Penyusunan jadwal kerja tersebut merupakan sebuah pekerjaan kompleks dan menarik untuk diamati. Terdapat lebih dari satu sudut pandang yang menjadi tujuan dari penyusunan jadwal kerja perawat, yaitu sudut pandang rumah sakit, sudut pandang perawat dan sudut pandang pasien. Selain itu penyusunan jadwal perawat harus memperhatikan batasan-batasan yang ada agar mendapatkan jadwal kerja yang mendekati optimal. Batasan-batasan yang sering muncul pada penjadwalan perawat yaitu jumlah perawat yang tersedia, kemampuan dari perawat, peraturan tenaga kerja dan kebijakan rumah sakit yang berlaku. Banyaknya batasan pada penjadwalan perawat akan memunculkan sebuah permasalahan yang sering disebut sebagai Nurse Rostering Problem (NRP). Hingga saat ini, NRP termasuk ke dalam permasalahan yang sulit terpecahkan secara konvensional atau permasalahan NP-Hard. Banyak penelitian yang mencoba untuk menyelesaikan NRP secara komputasi. Penyelesaian NRP secara komputasi akan mengurangi waktu yang dibutuhkan untuk menyusun jadwal kerja perawat rumah sakit. Semakin sedikit waktu yang dibutuhkan untuk menyusun jadwal kerja perawat, maka akan ada banyak pengalokasian waktu untuk meningkatkan pelayanan medis. Hal

itu akan berdampak pada kualitas kesehatan pada daerah sekitar rumah sakit tersebut. Beberapa penelitian telah mencoba menyelesaikan NRP dengan menggunakan beberapa pendekatan. Salah satu pendekatan yang digunakan yaitu hyper-heuristic. Hyper-heuristic adalah metode pencarian yang digunakan untuk menyelesaikan permasalahan optimasi. Penelitian tugas akhir ini akan menggunakan pendekatan hyper-heuristic dengan algoritma Reinforcement Learning - Simulated Annealing with Reheating untuk menyelesaikan permasalahan penjadwalan perawat pada rumah sakit di Norwegia. Hasil penelitian tugas akhir ini menunjukan bahwa algoritma Reinforcement Learning – Simulated Annealing with Reheating mampu menghasilkan jadwal kerja dengan peningkatan solusi hingga 82%. Algoritma Reinforcement Learning – Simulated Annealing with Reheating juga mampu menghasilkan peningkatan solusi yang lebih baik dari algoritma pembandingnya, yaitu 3% hingga 47% lebih baik dari algoritma Simple Random – Hill Climbing, 2% hingga 12% lebih baik dari algoritma Reinforcement Learning – Hill Climbing, dan 0.7% hingga 7% lebih baik dari algoritma Reinforcement Learning – Simulated Annealing.

Kata kunci: Nurse Rostering Problem, Hyper-heursitic, Reinforcement Learning, Simulated Annealing with Reheating

NURSE ROSTERING OPTIMIZATION USING REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC ALGORITHM ON BENCHMARK DATASET NORWEGIAN HOSPITAL

Name	**: Dimas Rizal Kusuma Shindu**
NRP	**: 05211640000094**
Department	**: Sistem Informasi FTEIC-ITS**
Supervisor	**: Ahmad Muklason, S.Kom., M.Sc., Ph.D.**

ABSTRACT

Nurse rostering is the process of creating a work schedule for nurses over a given planning horizon. The creation of work schedule is a complex work and interesting to observe. There is more than one perspective that is the goal of creating a work schedule for nurses, that is the hospital's perspective, nurse's perspective and patient's perspective. In addition, the process of creating work schedule must see to existing constraints in order to get a work schedule that is close to optimal. The constraints that often arise in nurse scheduling are the number of nurse, the competence of nurse, labor regulations and hopital policies. The many constraints that must be met will raise the problem that is often called as Nurse Rostering Problem (NRP). Until now, NRP is a problem that are difficult to be solved conventionally or NP-Hard problem. Many studies have tried to solve NRP computationally. Computational problem solving will reduce the time of creating a work schedule for nurses. The less time that spent for creating a work schedule, the more time will be allocated to improve medical sevices. Improved medical services will have an impact on improving the health quality around the hospital's area. Several studies have tried to resolve NRP by using several approaches. One approach used is hyper-heuristic. Hyper-heuristic is a search method used to solve optimization problem. In

this final project will use a Hyper-heuristic approach with the Reinforcement Learning – Simulated Annealing with Reheating algorithm to solve nurse scheduling problem at hospital in Norway. The result of this final project show that the Reinforcement Learning – Simulated Annealing with Reheating algorithm is able to create work schedule with an increase of solution up to 82%. Reinforcement Learning – Simulated Annealing with Reheating algorithm is also able to produce improved solutions that are better than the comparison algorithm, which is 3% to 47% better than Simple Random – Hill Climbing algorithm, 2% to 12% better than Reinforcement Learning – Hill Climbing algorithm, and 0.7% to 7% better than Reinforcement Learning – Simulated Annealing algorithm.

Keywords: Nurse Rostering Problem, Hyper-heursitic, Reinforcement Learning, Simulated Annealing with Reheating

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Dimas Rizal Kusuma Shindu
NRP	: 05211640000094
Tempat/Tanggal lahir	: Jombang, 6 September 1998
Fakultas/Departemen	: FTEIC / Departemen Sistem Informasi
Nomor Telp/Hp/email	: dimas.kusuma16@mhs.is.its.ac.id

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

OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL

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, 6 Agustus 2020

DIMAS RIZAL KUSUMA S.
NRP. 05211640000094

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan karunia dan hidayah-Nya sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir dengan judul, **"OPTIMASI PENJADWALAN PERAWAT RUMAH SAKIT MENGGUNAKAN ALGORITMA REINFORCEMENT LEARNING - SIMULATED ANNEALING WITH REHEATING HYPER-HEURISTIC PADA BENCHMARK DATASET NORWEGIAN HOSPITAL"** yang menjadi salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas akhir ini tidak akan pernah terwujud tanpa bantuan, bimbingan, dukungan, maupun doa dari berbagai pihak yang sudah meliburkan waktu, tenaga dan pikirannya. Oleh karena itu, penulis akan menyampaikan ucapan terima kasih yang sebanyak-banyaknya kepada :

1. Kedua orang tua penulis, Bapak Didiek Suhardi dan Ibu Lilik Sugiarti dan saudara penulis, Feronika Pratama Kusuma Shindu dan Aulia Rahma Kusuma Shindu yang senantiasa memberikan doa, motivasi dan dukungan yang terbaik untuk penulis.
2. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing yang senantiasa meluangkan waktu, memberikan ilmu dan petunjuk, serta memotivasi untuk kelancaran pengerjaan tugas akhir.
3. Bapak Edwin Riksakomara, S.Kom., MT. dan Bapak Faizal Mahananto S.Kom, M.Eng., Ph.D. selaku dosen penguji yang telah memberikan kritik dan saran untuk perbaikan tugas akhir.
4. Seluruh dosen departemen Sistem Informasi, FTEIC, ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.

5. Rizal, Sisca, dan Shafira sebagai tim *NRP* yang telah membantu, berdiskusi dan saling berdiskusi selama pengerjaan tugas akhir.
6. Teman-teman kontrakan HH, Adam, Ubai, Inud, Ferd, Rully, Fauzi, dan Fahmi yang selalu memberi semangat dan hiburan selama proses pengerjaan tugas akhir.
7. Teman-teman Artemis, khususnya sobat RDIB yang telah menemani dan menjadi tempat diskusi penulis dalam pengerjaan tugas akhir.
8. Teman-teman ‘till Jannah, Ari, Amanda, Mbak Dini, dan Mbak Fidah yang selalu memberi dukungan dari awal perkuliahan hingga pengerjaan tugas akhir ini.
9. Berbagai pihak yang tidak bisa disebutkan satu persatu yang telah ikut serta menyukseskan penulis dalam menyelesaikan tugas akhir.

Semoga semua dukungan maupun doa yang telah diberikan menjadi catatan amal baik di hadapan Tuhan Yang Maha Esa. Penulis juga menyadari bahwa penyusunan tugas akhir ini masih jauh dari kata sempurna sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan datang. Semoga penelitian tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, Juni 2020

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Bab ini membahas latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi tugas akhir yang dikerjakan. Uraian pada bab ini memberikan gambaran umum mengenai permasalahan dan pemecahan masalah pada tugas akhir.

1. 1. Latar Belakang

Nurse Rosterering adalah proses penyusunan jadwal kerja perawat rumah sakit dalam jangka waktu tertentu. Proses penyusunan jadwal tersebut harus memperhatikan beberapa aspek seperti kemampuan perawat, keadilan penjadwalan, hukum dan regulasi yang berlaku [1]. Penyusunan jadwal kerja perawat rumah sakit merupakan sebuah pekerjaan yang kompleks dan menarik untuk diamati. Terdapat lebih dari satu sudut pandang yang menjadi tujuan dari penyusunan jadwal kerja perawat. Pada sudut pandang rumah sakit, penjadwalan harus mempertimbangkan efisiensi sumber daya perawat untuk meminimalkan biaya yang keluar. Pada sudut pandang perawat, penyusunan harus mempertimbangkan keseimbangan antara jadwal kerja dan libur perawat. Pada sudut pandang pasien, penjadwalan harus mempertimbangkan kompetensi dari perawat agar mengurangi waktu tunggu pasien dan meningkatkan kualitas pelayanan medis. Selain itu penyusunan jadwal perawat harus memperhatikan batasan-batasan yang ada agar mendapatkan jadwal kerja yang mendekati optimal. Batasan-batasan yang sering muncul pada penjadwalan perawat yaitu jumlah perawat yang tersedia, kemampuan dari perawat, peraturan tentang tenaga kerja, dan kebijakan dari rumah sakit [2]. Banyaknya batasan yang harus terpenuhi akan memunculkan sebuah permasalahan yang sering disebut sebagai *Nurse Rostering Problem* (*NRP*).

Hingga saat ini, *Nurse Rostering Problem* (*NRP*) termasuk ke dalam permasalahan yang belum dapat terpecahkan secara eksak atau sering disebut permasalahan *NP-hard*. Banyak

penelitian yang mencoba untuk menyelesaikan *Nurse Rostering Problem* secara komputasi menggunakan berbagai algoritma dengan pendekatan *heuristic*, *metaheuristic*, maupun *hyper-heuristic*. Penyelesaian *Nurse Rostering Problem* secara komputasi akan mengurangi waktu yang dibutuhkan untuk menyusun jadwal kerja perawat rumah sakit. Semakin sedikit waktu yang dibutuhkan untuk menyusun jadwal kerja perawat, maka akan ada banyak pengalokasian waktu untuk meningkatkan pelayanan medis [2]. Hal itu akan berdampak pada kualitas kesehatan pada daerah sekitar rumah sakit tersebut.

Beberapa penelitian telah mencoba menyelesaikan *Nurse Rostering Problem* dengan menggunakan beberapa pendekatan. Salah satu penelitian yang membahas tentang *Nurse Rostering Problem* adalah penelitian oleh Martin Stølevik et al. dengan menggunakan penggabungan 2 algoritma [2]. Penelitian ini berfokus untuk memecahkan *Nurse Rostering Problem* pada rumah sakit di Norwegia. Penelitian tersebut menggunakan algoritma *Constraint Programming* (*CP*) dan *Variable Neighborhood Descent* (*VND*). Hasil dari penelitian tersebut menunjukan bahwa penggabungan 2 algoritma dapat memecahkan permasalahan yang besar dan realistis dengan waktu komputasi yang tidak terlalu lama.

Penelitian selanjutnya yang membahas tentang *Nurse Rostering Problem* adalah penelitian oleh Zhenyuan Liu et al. dengan menggunakan algoritma *Simulated Annealing* (*SA*) [3]. Penelitian ini berfokus pada penjadwalan perawat rumah sakit dengan memperhatikan tiga peran dari perawat. Tujuan dari penelitian ini adalah untuk menyusun jadwal kerja perawat yang dapat memenuhi permintaan sesuai dengan peran dari masing-masing perawat. Hasil dari penelitian ini menunjukan bahwa algoritma *Simulated Annealing* (*SA*) memberikan hasil yang lebih baik daripada algoritma pembandingnya yaitu *Hybrid Artificial Bee Colony* (*HABC*).

Penelitian selanjutnya yang membahas tentang modifikasi algoritma *Simulated Annealing* adalah penelitian oleh Say Leng

Goh et al. dengan menggunakan algoritma *Simulated Annealing with Reheating* (*SAR*) [4]. Penelitian ini berfokus pada pemecahan masalah penjadwalan mata kuliah dengan menggunakan algoritma *Tabu Search with Sampling and Perturbation* (*TSSP*) untuk mencari solusi awal dan algoritma *Simulated Annealing with Reheating* (*SAR*) untuk meningkatkan kualitas dari solusi awal. Hasil dari penelitian tersebut menunjukan bahwa algoritma yang digunakan dapat memberikan solusi yang baik. Algoritma *SAR* juga dapat menunjukan performa skalabilitasnya terhadap permasalahan penjadwalan mata kuliah.

Penelitian lain yang membahas permasalahan penjadwalan adalah penelitian yang dilakukan oleh Yunior César F. R. et al. dengan menggunakan algoritma *Reinforcement Learning* (*RL*) [5]. Penelitian ini berfokus untuk memecahkan masalah *Job Shop Scheduling Problem* (*JSSP*) dan *Flow Shop Scheduling Problem* (*FSSP*). Penelitian tersebut mengimplementasikan algoritma *Reinforcement Learning* untuk menyelesaikan kedua masalah tersebut. Hasil pada penelitian tersebut menunjukan bahwa algoritma *Reinforcement Learning* menghasilkan urutan kerja yang sangat baik dan memenuhi batasan pada permasalahan *JSSP*.

Berdasarkan penelitian sebelumnya, algoritma *Simulated Annealing with Reheating* dapat digabungkan dengan metode lain. Algoritma *Simulated Annealing with Reheating* digunakan untuk mencari solusi yang lebih baik. Konsep *reheating* akan bekerja ketika algoritma tersebut tidak mendapatkan solusi yang lebih baik pada beberapa iterasi. Penelitian lain juga menyebutkan bahwa algoritma *Reinforcement Learning* dapat digabungkan dengan metode lain. Penelitian tersebut menggabungkan metode *Artificial Neural Network* dengan konsep pembelajaran *Reinforcement Learning* [6]. Oleh karena itu, pada penelitian tugas akhir ini akan menggunakan penggabungan dari algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating* berbasis *hyper-heuristic* untuk memberikan solusi terhadap permasalahan penjadwalan

perawat dengan menggunakan data perawat rumah sakit di Norwegia.

Luaran pada pengerjaan tugas akhir ini adalah sebuah algoritma yang menghasilkan solusi layak dan mendekati optimal. Algoritma tersebut dapat membantu rumah sakit lainnya untuk menyusun jadwal kerja perawat secara efektif dan efisien.

1. 2. Rumusan Masalah

Berdasarkan latar belakang di atas, maka penelitian ini memiliki rumusan masalah sebagai berikut :

1. Bagaimana menerapkan algoritma *Reinforcement Learning - Simulated Annealing with Reheating Hyper-Heuristic* untuk menyelesaikan masalah penjadwalan perawat rumah sakit dari *benchmark dataset norwegian hospital*?
2. Bagaimana performa algoritma *Reinforcement Learning - Simulated Annealing with Reheating Hyper-Heuristic* dalam menyelesaikan masalah penjadwalan perawat rumah sakit dari *benchmark dataset norwegian hospital*?

1. 3. Batasan Permasalahan

Batasan masalah dalam mengerjakan tugas akhir ini adalah sebagai berikut:

1. Data yang digunakan pada pengerajaan tugas akhir ini merupakan *dataset norwegian hospital*.
2. Aplikasi yang digunakan untuk melakukan optimasi dibangun menggunakan bahasa pemrograman java.

1. 4. Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan untuk mencapai hal-hal sebagai berikut:

1. Mengoptimalkan penjadwalan perawat rumah sakit dari *benchmark dataset norwegian hospital* menggunakan metode *Reinforcement Learning - Simulated Annealing*

with Reheating Hyper-Heuristic untuk mendapatkan hasil yang lebih baik dari metode lainnya.
2. Melakukan analisa terhadap performa algoritma *Reinforcement Learning - Simulated Annealing with Reheating Hyper-Heuristic* dalam menyelesaikan masalah penjadwalan perawat rumah sakit dari *benchmark dataset norwegian hospital*.

1. 5. Manfaat Penelitian

Penelitian pada tugas akhir ini memberikan manfaat sebagai berikut:

1. Menambah wawasan pada bidang ilmu sistem informasi khususnya penerapan metode *Reinforcement Learning - Simulated Annealing with Reheating Hyper-Heuristic* pada penjadwalan perawat rumah sakit.
2. Menghasilkan suatu algoritma yang dapat melakukan optimasi penjadwalan perawat rumah sakit sehingga dapat membantu rumah sakit di Indonesia untuk melakukan penjadwalan kerja perawat.

1. 6. Relevansi

Topik pada penelitian tugas akhir ini adalah optimasi penjadwalan perawat rumah sakit. Topik optimasi merupakan bagian dari pokok penelitian Analisis Bisnis. Hal tersebut mengindikasikan bahwa penelitian pada tugas akhir ini sesuai dengan bidang keilmuan Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB) yang dapat dilihat pada Gambar 1.1. Pengerjaan tugas akhir ini menggunakan metode *Reinforcement Learning - Simulated Annealing with Reheating*. Mata kuliah yang mempelajari metode tersebut adalah mata kuliah Optimasi Kombinatorik dan Heuristik (OKH) yang merupakan mata kuliah pilihan laboratorium RDIB.

Sistem Pendukung Keputusan	• Pemodelan Sistem dan Analisis • Peramalan

Manajemen Data	• Sistem Manajemen Basis Data • Data warehouse
Analisis Bisnis	• Optimasi • *Data Mining* • *Web Analytic*
Manajemen Pengetahuan	• Sistem Manajemen Pengetahuan • Ontology
Sistem Cerdas	• Algoritma Genetika • Jaringan Syaraf Tiruan • Logika Fuzzy • *Expert System* • *Intelegent Agent*

Gambar 1.1 Bidang Keilmuan Laboratorium Rekayasa Data dan Intelegensi Bisnis

1. 7. Ringkasan Pendahuluan

Penelitian tugas akhir ini akan melakukan optimasi terhadap penjadwalan perawat rumah sakit menggunakan data rumah sakit di Norwegia. Luaran dari penelitian tugas akhir ini berupa algoritma yang dapat menghasilkan solusi yang layak dan mendekati optimal. Penelitian tugas akhir ini memiliki relevansi dengan bidang keilmuan Laboratorium Rekayasa Data dan Intelegensi Bisnis pada bagian optimasi. Penelitian tugas akhir ini juga memiliki relevansi dengan beberapa mata kuliah yang diajarkan pada Departemen Sistem Informasi, salah satunya yaitu mata kuliah Optimasi Kombinatorik dan Heuristik (OKH). Selanjutnya, untuk mendukung proses pengerjaan tugas akhir ini akan dilakukan studi literatur yang akan dijelaskan pada bab 2. Metodologi penelitian akan dijelaskan pada bab 3. Perancangan pengerjaan tugas akhir akan dijelaskan pada bab 4. Implementasi dari perancangan tugas akhir akan dijelaskan pada bab 5. Hasil dan pembahasan akan dijelaskan pada bab 6. Kemudian kesimpulan dan saran akan dijelaskan pada bab 7.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan tinjauan pustaka untuk mendukung latar belakang penelitian tugas akhir yang telah dijelaskan pada bab 1. Tinjauan pustakan pada bab ini terdiri dari peneltian sebelumnya dan dasar teori. Penelitian sebelumnya menjelaskan tentang beberapa penelitian tentang penjadwalan dengan menggunakan berbagai macam algoritma. Dasar teori menjelaskan tentang teori-teori yang digunakan dalam penelitian tugas akhir ini.

2. 1. Penelitian Sebelumnya

Pada sub bab ini menjelaskan penelitian sebelumnya yang menjadi dasar pengerjaan tugas akhir.

Tabel 2.1 Ringkasan Jurnal Penelitian Sebelumnya

No	Penelitian Sebelumnya	
1	Judul Penelitian	*A Hybrid Approach for Solving Real-World Nurse Rostering Problems* [2]
	Nama Peneliti	Martin Stølevik, Tomas Eric Nordlander, Atle Riise, Helle Frøyseth
	Tahun Penelitian	2011
	Deskripsi	Penelitian tersebut membahas pemecahan masalah *Nurse Rostering Problem* pada rumah sakit di Norwegia dengan pendekatan *hybrid*. Penelitian tersebut menggunakan algoritma *Constraint Programming* dan *Variable Neighborhood Descent*. Hasil dari penelitian tersebut menunjukan bahwa pendekatan *hybrid* dapat memecahkan permasalahan yang besar dan realistis dengan waktu komputasi yang tidak terlalu lama.

No	Penelitian Sebelumnya	
	Keterkaitan dengan Tugas Akhir	Penelitian tersebut akan menjadi referensi dalam memahami permasalahan yang menjadi topik pada tugas akhir ini.
	Perbedaan	Penelitian tersebut menggunakan pendekatan *hybrid* dengan algoritma *Constraint Programing* dan *Variable Neighborhood Descent* untuk memecahkan permasalahan *Nurse Rostering Problem*. Tugas akhir ini akan menggunakan pendekatan *hyper-heuristic* dengan algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating* untuk mencari solusi yang lebih baik daripada solusi yang dihasilkan pada penelitian tersebut.
2	Judul Penelitian	*Simulated Annealing for a Multi-Level Nurse Rostering Problem in Hemodialysis Service* [3]
	Nama Peneliti	Zhenyuan Liu, Zaisheng Liu, Zhipeng Zu, Yidong Shen, Junwu Dong
	Tahun Penelitian	2017
	Deskripsi	Penelitian tersebut berfokus pada penjadwalan perawat rumah sakit di Cina dengan memperhatikan 3 peran dari perawat. Tujuan dari penelitian ini adalah untuk menyusun jadwal kerja perawat yang dapat memenuhi permintaan sesuai dengan peran dari masing-masing perawat. Penelitian ini menggunakan algoritma *Simulated Annnealing* untuk menyusun jadwal

No	Penelitian Sebelumnya	
		kerja perawat. Hasil dari penelitian ini menunjukan bahwa algoritma *Simulated Annealing* memberikan hasil yang lebih baik daripada algoritma pembandingnya yaitu *Hybrid Artificial Bee Colony*.
	Keterkaitan dengan Tugas Akhir	Penelitian tersebut akan menjadi referensi dalam menentukan metode untuk menyelesaikan permasalahan pada tugas akhir dengan sedikit menambah variasi dari algoritma tersebut.
	Perbedaan	Penelitian tersebut berfokus untuk memberikan solusi terhadap *Nurse Rostering Problem* di Cina dengan menggunakan algoritma *Simulated Annealing*. Tugas akhir ini akan berfokus untuk memberikan solusi terhadap *Nurse Rostering Problem* di Norwegia menggunakan pendekatan *hyper-heuristic* dengan algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating*.
3	Judul Penelitian	*Improved Local Search Approaches to Solve The Post Enrolment Course Timetabling Problem* [4]
	Nama Peneliti	Say Leng Goh, Graham Kendall, Nasser R. Sabar
	Tahun Penelitian	2017
	Deskripsi	Penelitian tersebut berfokus untuk memberikan solusi terhadap masalah penjadwalan mata kuliah dengan menggunakan algoritma *Tabu Search*

No	Penelitian Sebelumnya	
		with Sampling and Perturbation (TSSP) untuk mencari solusi awal dan algoritma *Simulated Annealing with Reheating (SAR)* untuk meningkatkan kualitas dari solusi awal. Hasil dari penelitian tersebut menunjukan bahwa algoritma yang *TSSP-SAR* dapat memberikan solusi yang baik. Algoritma *SAR* juga dapat menunjukan performa skalabilitasnya terhadap permasalahan yang diangkat.
	Keterkaitan dengan Tugas Akhir	Penelitian tersebut akan menjadi referensi dalam menentukan metode untuk menyelesaikan permasalahan pada tugas akhir.
	Perbedaan	Penelitian tersebut berfokus untuk memberikan solusi terhadap masalah penjadwalan mata kuliah dengan mengunakan algoritma *Tabu Search with Sampling and Perturbation* dan *Simulated Annealing with Reheating*. Tugas akhir ini akan berfokus untuk memberikan solusi terhadap *Nurse Rostering Problem* di Norwegia menggunakan pendekatan *hyper-heuristic* dengan algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating*.
4	Judul Penelitian	*A Reinforcement Learning Approach for Scheduling Problems* [5]
	Nama Peneliti	Yunior César Fonseca Reyna, Yailen Martínez Jiménez, Juan Manuel Bermúdez Cabrera, Beatriz M. Méndez Hernández

No	Penelitian Sebelumnya	
	Tahun Penelitian	2015
	Deskripsi	Penelitian ini berfokus untuk memberikan solusi terhadap masalah penjadwalan produksi pada manufaktur dengan menggunakan pendekatan metode *Reinforcement Learning*. Masalah penjadwalan produksi terbagi menjadi dua yaitu *Job Shop Scheduling Problem* (*JSSP*) dan *Flow Shop Scheduling Problem* (*FSSP*). Penelitian tersebut mengimplementasikan algoritma *Reinforcement Learning* untuk menyelesaikan kedua masalah tersebut. Hasil pada penelitian tersebut menunjukan bahwa algoritma *Reinforcement Learning* menghasilkan urutan kerja yang sangat baik dan memenuhi batasan pada permasalahan *JSSP*.
	Keterkaitan dengan Tugas Akhir	Penelitian ini akan menjadi referensi dalam menentukan metode untuk menyelesaikan permasalahan pada tugas akhir.

No	Penelitian Sebelumnya	
	Perbedaan	Penelitian tersebut berfokus untuk memberikan solusi terhadap masalah penjadwalan produksi pada manufaktur dengan menggunakan pendekatan metode *Reinforcement Learning*. Tugas akhir ini akan berfokus untuk memberikan solusi terhadap *Nurse Rostering Problem* di Norwegia menggunakan pendekatan *hyper-heuristic* dengan algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating*.

2. 2. Landasan Teori

Pada sub bab ini menjelaskan teori-teori yang akan digunakan dalam pengerjaan tugas akhir.

2. 2. 1. Penjadwalan

Penjadwalan merupakan sebuah proses yang memiliki peran penting pada perusahaan manufaktur maupun jasa. Penjadwalan berhubungan dengan pengalokasian sumber daya untuk melakukan suatu pekerjaan dengan batas waktu tertentu. Tujuan dari penjadwalan adalah untuk mengoptimasi satu atau beberapa sasaran [7].

Penjadwalan pada perusahaan jasa memiliki sedikit perbedaan dengan penjadwalan pada perusahaan manufaktur. Penjadwalan pada perusahaan jasa memiliki tingkat kesulitan yang lebih tinggi daripada perusahaan manufaktur. Pada perusahaan jasa terdapat banyak permasalahan serta batasan yang terdapat dalam penjadwalan. Penjadwalan pada perusahaan jasa juga harus melakukan koordinasi dengan beberapa pendukung keputusan untuk menghasilkan solusi yang optimal.

2. 2. 2. *Nurse Rostering Problem (NRP)*

Perawat merupakan salah satu komponen yang mendukung berjalannya proses bisnis pada sebuah rumah sakit. Perawat memiliki tugas untuk merawat dan melayani pasien rumah sakit. Secara tidak langsung, perawat menjadi bagian penting pada rumah sakit sehingga memerlukan keberadaan perawat untuk berjaga selama 24 jam. Kondisi tersebut dapat tercapai jika penjadwalan perawat dilakukan secara tepat. Penjadwalan perawat menjadi semakin kompleks karena harus memenuhi kebutuhan perawat sekaligus pasien. Kompleksitas dari penjadwalan perawat tersebut menjadi sebuah permasalahan yang sering disebut sebagai *Nurse Rostering Problem*.

Nurse Rostering Problem adalah sebuah permasalahan pada mengalokasikan perawat ke dalam beberapa alokasi waktu kerja yang tersedia. Pengalokasian tersebut tentunya harus mematuhi beberapa batasan-batasan yang ada [8]. Beberapa batasan yang harus dipatuhi adalah tidak melakukan penjadwalan perawat pada hari libur dan penjadwalan satu waktu untuk perawat yang sama. Terdapat dua batasan pada *Nurse Rostering Problem* yaitu *hard constraint* dan *soft constraint*. *Hard constraint* adalah semua batasan yang wajib terpenuhi sedangkan *soft constraint* adalah batasan yang tidak wajib terpenuhi dan dapat dilanggar untuk menghasilkan solusi yang dapat diterapkan [9].

2. 2. 3. Dataset *Norwegian Hospital*

Dataset norwegian hospital merupakan dataset yang berasal dari salah satu rumah sakit di Norwegia. Dataset tersebut terbagi atas 7 unit rumah sakit yang dikumpulkan dan dikelola oleh sebuah organisasi peneliti dari Eropa yaitu SINTEF [10]. Dataset ini terdiri dari beberapa bagian yaitu bagian pekerja, sif, kebutuhan perawat, batasan dan pola sif yang diinginkan. Pada bagian pekerja terdapat data mengenai identitas perawat, jam kerja perawat setiap minggunya, pada minggu berapa perawat tersebut bekerja dan kompetensi dari setiap perawat. Pada bagian sif terdapat data mengenai identitas sif, panjangnya sif, kategori sif, nama sif, waktu mulainya sif, waktu berakhirnya sif dan

kompetensi yang dibutuhkan. Pada bagian kebutuhan perawat terdapat data mengenai nama sif, jumlah sif setiap harinya dan identitas sif. Pada bagian pola sif yang diinginkan terdapat identitas, hari mulai berkerja, dan pola yang diinginkan.

Permasalahan penjadwalan perawat pada *norwegian hospital* bertujuan untuk mengoptimalkan penjadwalan perawat pada *norwegian hospital*. Kondisi tersebut dapat dipenuhi dengan memperhatikan beberapa batasan yang ada, baik *hard constraint* maupun *soft constraint*. Berikut merupakan *hard constraint* pada permasalahan penjadwalan perawat *norwegian hospital* [2]:

a. Satu perawat setiap harinya dialokasikan maksimal satu sif kerja.
b. Kebutuhan sif harus tepat terpenuhi setiap harinya.
c. Total jam kerja untuk setiap perawat tidak boleh menyimpang terlalu jauh dari kontrak kerja.
d. Perawat hanya dapat bekerja pada sif yang sesuai dengan kompetensi yang dimiliki.
e. Waktu jeda antar dua sif yang berurutan harus melebihi batas minimal yang telah ditentukan.
f. Pada setiap minggu harus ada minimal periode bebas kerja.
g. Tidak boleh melanggar batas maksimal jam kerja satu minggu.

Soft constarint pada permasalahan penjadwalan perawat *norwegian hospital* adalah sebagai berikut [2]:

a. Tidak boleh terlalu banyak hari kerja berurutan dengan kategori sif yang sama.
b. Tidak boleh terlalu banyak hari kerja berurutan.
c. Tidak boleh terlalu sedikit hari kerja berurutan dengan kategori sif yang sama.
d. Tidak boleh terlalu sedikit hari kerja berututan.
e. Tidak boleh menyimpang terlalu jauh dari batas minimum dan maksimum jumlah sif pada setiap kategorinya.
f. Tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat.
g. Mengelompokan waktu libur setiap perawat.

h. Memenuhi pola sif yang diinginkan.
i. Tidak boleh terlalu banyak pola sif yang tidak diinginkan.

2. 2. 4. Model Matematis Dataset *Norwegian Hospital*

Model matematis *dataset norwegian hospital* disusun berdasarkan batasan-batasan yang telah dijelaskan pada bagian sebelumnya. Langkah awal dalam menyusun model matematis adalah menentukan notasi yang digunakan pada model matematis *dataset norwegian hospital* yang mengacu pada [11] dengan beberapa penyesuaian sebagai berikut:

S	:	Himpunan dari jenis sif. Sebuah sif $s \in S$ adalah anggota dari satu dan hanya satu kategori sif $c \in C$.
s	:	Anggota dari himpunan sif S.
C	:	impunan dari kategori sif.
c	:	Anggota dari himpunan kategori sif C. Terdapat tiga kategori sif yaitu sif pagi, sif siang, dan sif malam.
D	:	Himpunan dari hari.
d	:	Anggota dari himpunan hari D dimulai dari hari pertama yaitu hari Senin.
E	:	Himpunan dari perawat.
e	:	Anggota dari himpunan perawat E.
I_d	:	Himpunan dari pasangan sif hari d dan d+1 yang tidak sesuai karena waktu jeda antara sif yang terlalu pendek.
U	:	Himpunan dari pola sif yang tidak diinginkan.
u	:	Anggota dari himpunan pola sif yang tidak diinginkan.
V	:	Himpunan dari pola sif yang diinginkan.
v	:	Anggota dari himpunan pola sif yang diinginkan.
W	:	Himpunan dari minggu.
w	:	Anggota dari himpunan minggu dimulai dari minggu ke-0 dimana hari pertama pada minggu tersebut merupakan hari Senin dan hari ketujuh merupakan hari Minggu.
x	:	Sebuah solusi yang berupa jadwal kerja perawat.

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

Adapun parameter yang digunakan dalam model matematis mengacu pada [11] dengan beberapa penyesuaian sebagai berikut:

N_c^{min} : Jumlah minimal sif yang berurutan dari kategori *c* untuk seluruh perawat.

N_c^{max} : Jumlah maksimal sif yang berurutan dari kategori *c* untuk seluruh perawat.

N^{min} : Jumlah minimal sif yang berurutan dari semua kategori untuk seluruh perawat.

N^{max} : Jumlah maksimal sif yang berurutan dari semua kategori untuk seluruh perawat.

N_{ec}^{max} : Jumlah maksimal sif kategori *c* untuk perawat *e*.

N_{ec}^{min} : Jumlah minimal sif kategori *c* untuk perawat *e*.

R_{ds} : Kebutuhan sif setiap harinya sesuai dengan kebutuhan perawat.

T_e : Jumlah maksimal jam kerja perawat *e* setiap minggunya.

T_e^h : Jam kerja perawat yang sesuai dengan kontrak kerja selama periode penjadwalan tertentu.

T^f : Jumlah minimal waktu libur perawat *e* setiap minggunya.

T_s : Durasi dari sif *s*.

A_{sc} : Bernilai 1 jika sif *s* merupakan sif dengan kategori *c*, bernilai 0 jika sebaliknya.

B_{es} : Bernilai 1 jika perawat *e* mempunyai keahlian untuk bekerja pada sif *s*, bernilai 0 jika sebaliknya.

$f_{ew}(x)$: Panjang dari waktu libur terpanjang untuk perawat e dalam minggu w pada solusi x.

$u_{ei}(x)$: Jumlah pola sif yang tidak diinginkan oleh perawat e untuk tipe sif *i* pada solusi x.

$v_{ei}(x)$: Jumlah pola sif yang diinginkan oleh perawat e untuk tipe sif *i* pada solusi x

Kemudian variabel keputusan yang digunakan pada model matematis mengacu pada [11] dengan beberapa penyesuaian sebagai berikut:

q_{eds} : Bernilai 1 jika perawat *e* dijadwalkan pada hari *d* dengan kategori sif *s*, bernilai 0 jika sebaliknya.

Langkah selanjutnya yaitu merumuskan fungsi tujuan dari optimasi yang dilakukan. Fungsi tujuan dari dataset *norwegian hopitals* adalah sebagai berikut [11]:

$$Min\ f = \sum_{m=1}^{9} K_m p_m$$

Dimana K_m merupakan nilai tetap yang mencerminkan bobot dari penalti dan p_m merupakan nilai penalti dari setiap pelanggaran terhadap *soft constraint* seperti yang telah dijelaskan pada bagian notasi model matematis.

Batasan-batasan yang telah dijelaskan pada bagaian sebelumnya kemudian diubah ke dalam model matematis. Berikut adalah model matematis dari *hard constraint norwegian hospital* [11]:

Hard constraint 1. Satu perawat setiap harinya dialokasikan maksimal satu sif kerja.

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

Hard constraint 2. Kebutuhan sif harus tepat terpenuhi setiap harinya.

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

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

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

Hard constraint 4. Perawat hanya dapat bekerja pada sif yang sesuai dengan kompetensi yang dimiliki.

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

Hard constraint 5. Waktu jeda antar dua sif yang berurutan harus melebihi batas minimal yang telah ditentukan.

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

Hard constraint 6. Pada setiap minggu harus ada minimal periode bebas kerja.

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

Hard connstraint 7. Tidak boleh melanggar batas maksimal jam kerja satu minggu.

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

Berikut merupakan model matematis dari *soft constraint norwegian hospital* [11]:

Soft constraint 1. Tidak boleh terlalu banyak hari kerja berurutan dengan kategori sif yang sama.

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

Soft constraint 2. Tidak boleh terlalu banyak hari kerja berurutan.

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

Soft constraint 3. Tidak boleh terlalu sedikit hari kerja berurutan dengan kategori sif yang sama.

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

Soft constraint 4. Tidak boleh terlalu sedikit hari kerja berututan.

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

Soft constraint 5. Tidak boleh menyimpang terlalu jauh dari batas minimum dan maksimum jumlah sif pada setiap kategorinya.

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

Soft constraint 6. Tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat.

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

Soft constraint 7. Mengelompokan waktu libur setiap perawat.

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

Soft constraint 8. Memenuhi pola sif yang diinginkan.

$$p_8(x) = |E||D| - \sum_{e \in E, d \in D} v_{ei}\,(x)$$

Soft constraint 9. Tidak boleh terlalu banyak pola sif yang tidak diinginkan.

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

Dengan penggunaan parameter *soft constraint* sebagai berikut:

$$y_{ecd} = \sum_{s \in S} \mathrm{A_{sc}} q_{\mathrm{eds}}$$

Dimana y_{ecd} akan bernilai 1 jika perawat *e* berkerja pada sif kategori *c* pada hari *d*, bernilai 0 jika sebaliknya.

$$z_{ed} = \sum_{s \in S} q_{\text{eds}}$$

Dimana z_{ed} akan bernilai 1 jika perawat *e* berkerja pada hari *d*, bernilai 0 jika sebaliknya.

2. 2. 5. *Simulated Annealing*

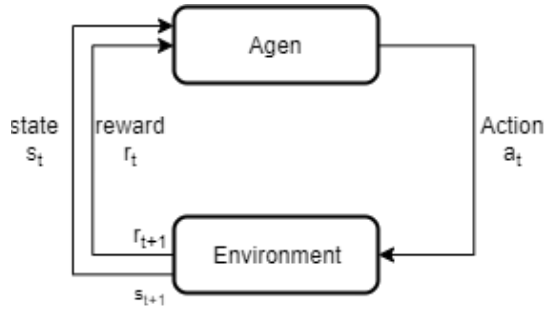
Simulated Annealing (*SA*) adalah sebuah teknik yang digunakan dalam permasalahan optimasi kombinatorial. *Simulated Annealing* merupakan teknik yang telah lama dikembangkan dengan menggunakan analogi dari pembuatan sebuah benda logam [12]. Pada pembuatan benda logam, sebuah logam padat dipanaskan pada suhu tertentu untuk memudahkan perubahan bentuknya. Logam yang telah dipanaskan kemudian dibentuk dan dibiarkan agar membeku secara perlahan hingga mencapai bentuk yang diinginkan [13]. Kekuatan dari struktur logam yang terbentuk bergantung pada proses pendinginan yang terjadi. Logam yang kuat dihasilkan dari proses yang dilakukan secara hati-hati dan proses pendinginan logam yang lambat. *Algoritma Simulated Annealing* mengadopsi proses pendinginan logam hingga mendapatkan solusi yang optimal [14].

2. 2. 6. *Simulated Annealing with Reheating*

Simulated Annealing with Reheating (*SAR*) adalah sebuah pengembangan dari algoritma *Simulated Annealing*. Setiap solusi yang lebih baik akan diterima, namun untuk solusi yang lebih buruk akan diterima dengan probabilitas tertentu. Probabilitas tersebut dipengaruhi oleh suhu awal yang telah ditentukan. Setiap iterasi, probabilitas diterimanya solusi yang lebih buruk akan semakin kecil. Proses pemanasan ulang suhu dilakukan ketika sudah tidak ada solusi yang lebih baik pada beberapa iterasi. Pemanasan ulang tesebut dilakukan agar algoritma dapat melakukan eksplorasi ketika sudah tidak ada solusi lebih baik yang dihasilkan [4].

2. 2. 7. *Reinforcement Learning*

Reinforcement Learning adalah bentuk dari teknik *machine learning* dimana terdapat sebuah agen (*heuristic*) yang membentuk sebuah strategi optimal berdasarkan pengalaman interaksi dengan lingkungan eksternal. Strategi tersebut menentukan aksi mana yang akan dipilih untuk keadaan (state) dan waktu tertentu. Saat agen berinteraksi dengan sebuah *environment* dan melakukan sebuah aksi, maka agen akan memasuki keadaan baru dan menerima *reward* dari *environment* tersebut [15]. *Reward* yang didapat akan digunakan oleh agen untuk meningkatkan *behavior* dari agen [6]. Konsep kerja dari algoritma *reinforcement learning* dapat dilihat pada Gambar 2.1 berikut ini.



Gambar 2.1. Konsep Kerja Algoritma *Reinforcement Learning*

Elemen utama yang menyusun formulasi dari algoritma reinforcement learning adalah sebagai berikut [6]:

a. Kumpulan dari keadaan, yaitu kumpulan dari semua keadaan yang mendeskripsikan *environment*.
b. Kumpulan dari aksi, yaitu kumpulan dari aksi yang dapat dilakukan.
c. *Environment* yaitu lingkungan dari sistem yang bersifat dinamis dan paling tidak telah terlihat sebagian.
d. Kontrol atau penentu kebijakan yaitu perilaku dari agen untuk mencapai tujuan pada jangka waktu tertentu.

e. *Reinforcement* atau *reward* yaitu umpan balik dari *environment* yang menunjukan hubungan antar perilaku dari agen.
f. Fungsi *reinforcement* yaitu fungsi dari *reinforcement learning* yang didefinisikan sesuai dengan permasalahan yang ada.
g. Nilai fungsi yaitu nilai yang didapat dari *reward* saat ini dan *reward* masa depan. Terdapat dua nilai fungsi yaitu nilai fungsi keadaan dan nilai fungsi aksi. Nilai fungsi keadaan hanya mempertimbangkan nilai dari keadaan, sedangkan nilai fungsi aksi mempertimbangkan nilai dari keadaan dan aksi.

2. 2. 8. *Heuristic*

Heuristic adalah sebuah metode yang mencari solusi terbaik pada waktu komputasi yang layak. Solusi yang dihasilkan pada metode *heuristic* tidak menjamin bahwa solusi tersebut merupakan solusi yang optimal. Hal tersebut menyebabkan metode *heuristic* tidak dapat memperkirakan tingkat optimal dari solusi yang dihasilkan. Terdapat banyak algoritma yang dapat digunakan untuk menerapkan metode *heuristic,* diantaranya yaitu *Simulated Annealing*, *Genetic Algorithm*, *Genetic Programming,* dan *Tabu Search* [16].

2. 2. 9. *Hyper-heuristic*

Hyper-Heuristic adalah sebuah metode pencarian atau mekanisme pembelajaran untuk memilih atau menghasilkan *heuristic* yang dapat memecahkan permasalahan optimasi kombinatorial [17]. *Heuristic* yang dipilih atau dihasilkan disebut sebagai *low-level heuristic*. *Low-level heuristic* dikelompokan menjadi 2 yaitu *low-level constructive heuristic* dan *low-level perturbative heuristic*. *Low-level constructive heuristic* digunakan untuk membuat solusi awal dari permasalahan yang ingin dicari solusi optimalnya. Terdapat beberapa *low-level constructive heuristic* yang dapat digunakan sesuai dengan jenis permasalahan yang ingin dicari solusi

optimalnya. Contoh dari *low-level constructive heuristic* untuk domain permasalahan *examination timetabling* adalah *largest degree, largest weighted degree, largest colour degree, largest enrolment* dan *saturation degree*. *Low-level perturbative heuristics* digunakan untuk memperbaiki solusi awal yang sudah ada secara acak atau menggunakan *constructive heuristic*. *Low-level pertubative heuristic* akan membuat perubahan dari solusi awal untuk menghasilkan solusi yang paling optimal. Terdapat beberapa *low-level perturbative heuristic* yang dapat digunakan sesuai dengan jenis permasalahan yang ingin dicari solusi optimalnya. Contoh dari *low-level perturbative heuristic* untuk domain permasalahan *Nurse Rostering Problem* adalah *change shift, swap shift, move shift*, dan *ruin and recreate*. *Hyper-heuristic* terdiri dari dua komponen yaitu pemilihan *heuristic* untuk memilih *low-level heuristic* dan *move acceptance* untuk menentukan perubahan yang dilakukan pada *low-level heuristic* dapat diterima atau tidak [18].

2. 3. Ringkasan Tinjauan Pustaka

Banyak penelitian yang telah mencoba untuk menyelesaikan masalah penjadwalan dengan menggunakan berbagai algoritma. Penyelesaian masalah penjadwalan secara komputasi terbukti lebih efektif karena tidak membutuhkan waktu yang terlalu lama. Berdasarkan konsep pembelajaran pada *Reinforcement Learning* dan konsep pemanasan ulang pada algoritma *Simulated Annealing with Reheating* akan menghasilkan algoritma yang dapat memberikan solusi yang mendekati optimal. Sehingga pada penelitian tugas akhir ini akan menggunakan penggabungan dari algoritma *Reinforcement Learning* dan *Simulated Annealing with Reheating* untuk menyelesaikan masalah penjadwalan perawat dengan menggunakan data rumah sakit di Norwegia. Selanjutnya pada bab 3 akan menjelaskan tentang metodologi penelitian yang digunakan pada penelitian tugas akhir ini.

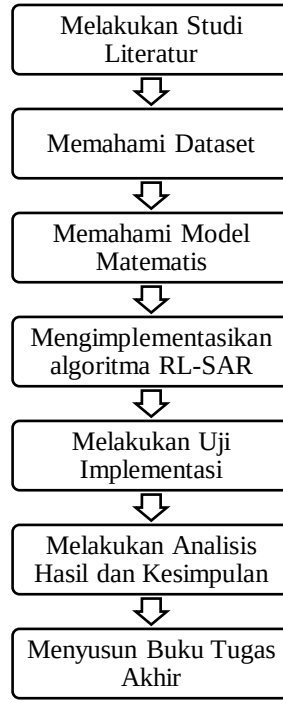
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan metodologi yang akan digunakan dalam mengerjakan penelitian tugas akhir ini. Metodologi yang digunakan mengacu pada penelitian sebelumnya dan dasar teori yang telah dijelaskan pada Bab 2.

3. 1. Diagram Metogologi

Berikut merupakan alur metodologi pengerjaan tugas akhir yang dapat dilihat pada Gambar 3.1



Gambar 3.1 Alur Metode Pengerjaan Tugas Akhir

3. 2. Uraian Metodologi

Berikut ini merupakan penjelasan dari masing-masing proses pengerjaan tugas akhir berdasarkan pada diagram alur metodologi yang dapat dilihat pada Gambar 3.1 di atas.

3. 2. 1. Melakukan Studi Literatur

Tahap ini merupakan tahap pendalaman pengetahuan terkait permasalahan yang akan diteliti dengan mengacu kepada penelitian yang telah dilakukan. Pada tahap ini akan memilih metode dan algoritma yang digunakan untuk menyelesaikan permasalahan penjadwalan perawat rumah sakit. Proses pemilihan metode dan algoritma dilakukan dengan cara membandingkan beberapa algoritma yang telah digunakan pada penelitian sebelumnya. Setelah proses pembandingan metode dan algoritma selesai, maka akan dipilih metode dan algoritma yang akan digunakan untuk menyelesaikan permasalahan yang diangkat pada tugas akhir ini.

3. 2. 2. Memahami Dataset

Tahap ini merupakan proses memahami *dataset norwegian hospital*. Dataset tersebut adalah dataset proyek penelitian oleh Gatsoft AS yang merupakan pengembang perangkat lunak manajemen perawat di Norwegia. *Dataset norwegian hospital* tersedia secara online dan dapat diakses melalui situs yang dikelola oleh SINTEF.

3. 2. 3. Memahami Model Matematis

Tahap ini merupakan proses memahami model matematis yang ada pada literatur. Luaran dari tahap ini yaitu pemahaman terhadap variabel keputusan, fungsi persamaan batasan masalah, dan fungsi tujuan. Fungsi persamaan batasan masalah pada permasalahan penjadwalan perawat rumah sakit terdiri dari *hard constraint* dan *soft constraint*. *Hard constraint* adalah semua batasan yang wajib terpenuhi, sedangkan *soft constraint* adalah batasan yang tidak wajib terpenuhi dan dapat dilanggar untuk menghasilkan solusi yang dapat diterapkan.

3. 2. 4. Mengimplementasi algoritma RL-SAR

Tahap ini merupakan proses implementasi terhadap algoritma pada *dataset norwegian hospital*. Terdapat dua algoritma yang akan diimplementasikan pada dataset tersebut. Algoritma *Reinforcement Learning* akan digunakan untuk memilih *low-level heuristic*. *Low-level heuristic* yang akan digunakan yaitu *2-exchange*, *3-exchange*, dan *double 2-exchange*. Algoritma *Simulated Annealing with Reheating* akan digunakan untuk melakukan *move acceptance* terhadap solusi yang dihasilkan oleh *low-level heuristic*. Hasil yang diperoleh akan dianalisa pada tahap selanjutnya.

3. 2. 5. Melakukan Uji Implementasi

Tahap ini merupakan proses validasi terhadap hasil yang didapat selama proses implementasi dengan algoritma *Reinforcement Learning – Simulated Annealing with Reheating*. Pengujian dilakukan terhadap setiap permasalahan penjadwalan perawat rumah sakit yang terdapat pada *dataset norwegian hospital*. Variasi permasalahan penjadwalan perawat pada masing-masing rumah sakit akan memberikan hasil yang berbeda. Jika hasil yang diperoleh dari implementasi algoritma sesuai dengan hasil yang diharapkan, maka hasil tersebut dapat diterima. Namun, jika hasil yang diperoleh tidak sesuai dengan hasil yang diharapkan, maka akan dilakukan perbaikan terhadap algoritma yang digunakan.

3. 2. 6. Melakukan Analisis Hasil dan Kesimpulan

Pada tahap ini bertujuan untuk memberikan analisa terkait hasil yang didapat selama proses implemantasi algoritma *Reinforcement Learning – Simulated Annealing with Reheating* dalam menyelesaikan permasalahan penjadwalan perawat rumah sakit. Analisis dilakukan dengan mempertimbangkan hasil penjadwalan yang didapat pada setiap iterasi. Hasil yang diperoleh akan dibandingkan dengan hasil yang yang didapat pada penelitian yang menggunakan metode lain.

3. 2. 7. Menyusun Buku Tugas Akhir

Tahap ini bertujuan untuk mendokumentasikan seluruh proses pengerjaan tugas akhir. Dokumentasi yang dilakukan meliputi seluruh rangkaian proses implementasi algoritma *Reinforcement Learning – Simulated Annealing with Reheating* untuk menyelesaikan permasalahan yang diangkat. Proses dokumentasi berawal dari formulasi fungsi tujuan hingga menemukan solusi akhir yang diharapkan. Luaran dari tahap ini adalah laporan tugas akhir atau sering disebut sebagai buku tugas akhir.

3. 3. Ringkasan Metodologi Penelitian

Metodologi penelitian yang telah dijelaskan akan menjadi alur pengerjaan penelitian tugas akhir ini. Pengerjaan tugas akhir diawali dengan melakukan studi literatur, kemudian memahami dataset dan model matematis, mengimplementasikan algoritma *RL-SAR*, melakukan uji implementasi, dan melakukan analisa hasil dan kesimpulan. Seluruh proses pengerjaan penelitian tugas akhir akan didokumentasikan pada buku tugas akhir. Selanjutnya, pada bab 4 akan menjelaskan tentang tahap perancangan yang terdiri dari pemahaman dataset, pemahaman model matematis, pemodelan algoritma, skenario penyusunan solusi awal, dan skenario uji coba parameter algoritma.

BAB IV
PERANCANGAN

Pada bab ini akan menguraikan rancangan penelitian tugas akhir yang merupakan tahap awal metodologi penelitian yang telah dijelaskan pada Bab 3. Rancangan tersebut meliputi pemahaman *dataset*, pemahaman model matematis, pemodelan algoritma yang akan digunakan, skenario penyusunan solusi awal dan skenario uji coba parameter algoritma.

4. 1. Pemahaman *Dataset*

Tugas akhir ini menggunakan *dataset* yang berasal dari salah satu rumah sakit di Norwegia. *Dataset* tersebut terdiri atas 7 unit rumah sakit (Optur) yang dikumpulkan dan dikelola oleh organisasi peneliti dari Eropa. Setiap unit rumah sakit menyimpan beberapa data yang tersimpan dalam berkas *excel*. Setiap berkas *excel* tersebut menyimpan data tentang perawat, sif, kebutuhan perawat, batasan, dan pola sif yang diinginkan atau tidak diinginkan. Pada bagian perawat menyimpan data tentang id perawat, kontrak kerja perawat, jadwal kerja perawat pada akhir pekan, dan kompetensi dari perawat. Pada bagian sif menyimpan data tentang id sif, durasi sif, kategori sif, nama sif, jam masuk sif, jam pulang sif, dan kebutuhan kompetensi dari sif. Pada bagian kebutuhan perawat menyimpan data tentang nama sif, kebutuhan sif setiap harinya, dan id sif. Pada bagian batasan menyimpan data tentang *hard constraint* dan *soft constraint* yang digunakan. Pada bagian pola sif menyimpan data tentang pola sif yang diinginkan dan pola sif yang tidak diinginkan.

Setiap unit rumah sakit memiliki tingkat kompleksitas permasalahan yang berbeda-beda. Hal tersebut dikarenakan setiap unit memiliki jumlah perawat, jumlah sif, jumlah kontrak jam kerja dan periode penjadwalan yang berbeda-beda. Selain itu pada unit rumah sakit 1 (Optur 1) dan unit rumah sakit 7 (Optur 7) membutuhkan perawat yang memiliki keahlian khusus untuk bekerja pada sif tertentu. Berikut merupakan

ringkasan dari kompleksitas permasalahan pada masing-masing unit rumah sakit yang dapat dilihat pada Tabel 4.1.

Tabel 4.1 Kompleksitas Permasalahan pada *Dataset Norwegian Hospital*

Unit Rumah Sakit	Jumlah Perawat	Jumlah Sif	Jumlah Kontrak Jam Kerja	Periode penjadwalan (hari)
Optur1	51	9	19170	84
Optur2	82	9	12819	42
Optur3	29	8	4159,5	42
Optur4	30	8	17352	168
Optur5	20	9	2280	28
Optur6	54	5	18978	84
OpTut7	15	6	2209,5	42

4. 2. Pemahaman Model Matematis

Model metematis digunakan untuk memodelkan permasalahan *Nurse Rostering Problem (NRP)* pada rumah sakit di Norwegia. Pemahaman model matematis bertujuan agar permasalahan tersebut dapat diselesaikan secara tepat. Secara singkat, model matematis telah dipaparkan pada BAB II yang terdiri atas variabel keputusan, fungsi batasan, dan fungsi tujuan. Berikut merupakan penjabaran dari model matematis yang telah dibuat.

4. 2. 1. Notasi, Parameter dan Variabel Keputusan

Penelitian tugas akhir ini menggunakan notasi, parameter, dan variabel keputusan yang telah dipaparkan secara singkat pada BAB II subbab model matematis *dataset Norwegian hospital*. Berikut penjelasan dari setiap notasi yang telah dibuat.

- S : merupakan himpunan dari jenis sif. Sebuah sif $s \in S$ adalah anggota dari satu dan hanya satu kategori sif $c \in C$.
- s : merupakan anggota dari himpunan jenis sif S. Setiap unit rumah sakit (Optur) memiliki jumlah sif

yang berbeda-beda. Sehingga, anggota dari himpunan jenis sif akan berbeda-beda sesuai dengan unit rumah sakit yang dimaksud.

C : merupakan himpunan dari kategori sif.

c : merupakan anggota dari himpunan kategori sif C. Terdapat 3 kategori sif, yaitu sif pagi, sif siang, dan sif malam.

D : merupakan himpunan dari hari.

d : merupakan anggota himpunan hari D. Setiap unit rumah sakit (Optur) memiliki panjang periode penjadwalan yang berbeda-beda. Sehingga, anggota dari himpunan hari akan berbeda-beda sesuai dengan unit rumah sakit yang dimaksud. Himpunan hari diawali dari hari Senin.

E : merupakan himpunan dari perawat.

e : merupakan anggota dari himpunan perawat Setiap unit rumah sakit (Optur) memiliki jumlah perawat yang berbeda-beda. Sehingga, anggota dari himpunan perawat akan berbeda-beda sesuai dengan unit rumah sakit yang dimaksud.

I_d : merupakan himpunan dari pasangan sif hari d dan d+1 yang tidak sesuai karena waktu jeda antara sif yang terlalu pendek. Setiap sif d akan dipasangkan dengan sif d + 1, kemudian dilakukan perhitungan antara selisih jam pulang sif d dengan jam masuk sif d + 1 untuk mendapatkan waktu jeda antar sif. Jika pasangan sif tersebut memiliki waktu jeda kurang dari batasan yang telah ditentukan, maka pasangan sif tersebut akan menjadi anggota himpunan I_d.

U : merupakan himpunan dari pola sif yang tidak diinginkan.

u : merupakan anggota dari himpunan pola sif yang tidak diinginkan. Masing-masing unit rumah sakit (Optur) memiliki pola sif yang tidak inginkan yang berbeda-beda. Sehingga, anggota dari himpunan

pola sif yang tidak diinginkan akan berbeda-beda sesuai dengan unit rumah sakit yang dimaksud.

V : merupakan himpunan dari pola sif yang diinginkan.

v : merupakan anggota dari himpunan pola sif yang diinginkan. Setiap unit rumah sakit (Optur) memiliki pola sif yang diinginkan masing-masing. Sehingga, anggota dari himpunan sif yang diinginkan akan berbeda-beda sesuai dengan unit rumah sakit.yang dimaksud.

W : merupakan himpunan dari minggu.

w : merupakan anggota dari himpunan minggu. Setiap minggu terdiri dari tujuh hari, dimana untuk minggu ke-0 diawali dengan hari pertama yaitu hari Senin dan diakhiri dengan hari ketujuh yaitu hari Minggu. Kemudian untuk minggu ke-1, diawali dengan hari kedelapan yaitu hari Senin dan diakhiri dengan hari keempat belas yaitu hari Minggu. Begitu pula untuk minggu-minggu selanjutnya sesuai dengan jumlah minggu penjadwalan setiap unit rumah sakit.

x : merupakan sebuah solusi yang berupa jadwal kerja perawat.

p_m : merupakan penalti dari pelanggaran *soft constraint* *m*, dimana *m* bernilai 1 sampai 9.

Kemudian model matematis yang telah dibuat memiliki beberapa parameter sebagai berikut.

N_c^{min} : Jumlah minimal sif yang berurutan dari kategori *c* untuk seluruh perawat. Parameter ini digunakan untuk mengitung nilai pelanggaran dari *soft constraint* 3. Setiap unit rumah sakit (Optur) memiliki nilai N_c^{min} yang berbeda.

N_c^{max} : Jumlah maksimal sif yang berurutan dari kategori *c* untuk seluruh perawat. Parameter ini digunakan untuk menghitung nilai pelanggaran

dari *soft constraint* 1. Setiap unit rumah sakit (Optur) memiliki nilai N_c^{max} yang berbeda-beda.

N^{min} : Jumlah minimal sif yang berurutan dari semua kategori untuk seluruh perawat. Parameter ini digunakan untuk menghitung nilai pelanggaran dari *soft constraint* 4. Setiap unit rumah sakit (Optur) memiliki nilai N^{min} yang sama yaitu N/A.

N^{max} : Jumlah maksimal sif yang berurutan dari semua kategori untuk seluruh perawat. Parameter ini digunakan untuk menghitung nilai pelanggaran dari *soft constraint* 2. Setiap unit rumah sakit (Optur) memiliki nilai N^{max} yang berbeda-beda.

N_{ec}^{max} : Jumlah maksimal sif kategori *c* untuk perawat *e*. Parameter ini digunakan untuk mengitung nilai pelanggaran dari *soft constraint* 5. Setiap unit rumah sakit (Optur) memiliki nilai N_{ec}^{max} yang berbeda-beda.

N_{ec}^{min} : Jumlah minimal sif kategori *c* untuk perawat *e*. Parameter ini digunakan untuk mengitung nilai pelanggaran dari *soft constraint* 5. Setiap unit rumah sakit (Optur) memiliki nilai N_{ec}^{min} yang berbeda-beda.

R_{ds} : Kebutuhan sif setiap harinya sesuai dengan kebutuhan perawat. Parameter ini digunakan untuk memastikan bahwa *hard constraint* 2 terpenuhi. Setiap unit rumah sakit (Optur) memiliki nilai R_{ds} yang berbeda-beda.

T_e : Jumlah maksimal jam kerja perawat *e* setiap minggunya. Parameter ini digunakan untuk memastikan bahwa *hard constraint* 7 terpenuhi. Setiap unit rumah sakit memiliki jumlah maksimal jam kerja perawat setiap minggu yang berbeda-beda.

T_e^h : Jam kerja perawat yang sesuai dengan kontrak kerja selama periode penjadwalan tertentu. Parameter ini digunakan untuk memastikan

bahwa *hard constraint* 3 terpenuhi dan untuk menghitung nilai pelanggaran *soft constraint* 6. Setiap perawat memiliki nilai T_e^h yang berbeda-beda.

T^f : Jumlah minimal waktu libur perawat *e* setiap minggunya. Parameter ini digunakan untuk memastikan bahwa *hard constraint* 6 terpenuhi. Setiap unit rumah sakit (Optur) memiliki jumlah minimal waktu libur yang sama yaitu 32 jam setiap minggunya.

T_s : Durasi dari sif *s*. Parameter ini digunakan untuk menghitung jam kerja perawat setiap minggunya. Setiap sif memiliki durasi yang berbeda-beda.

A_{sc} : Bernilai 1 jika sif *s* merupakan sif dengan kategori *c*, bernilai 0 jika sebaliknya.

B_{es} : Bernilai 1 jika perawat *e* mempunyai keahlian untuk bekerja pada sif *s*, bernilai 0 jika sebaliknya. Parameter ini digunakan untuk memastikan bahwa sif yang membutuhkan perawat dengan keahlian tertentu terpenuhi secara tepat. Parameter ini digunakan pada *hard constraint* 4.

Selain parameter di atas, terdapat beberapa parameter yang dibuat untuk mengurangi kompleksitas dari model matematis yang telah dibuat. Parameter tambahan tersebut adalah sebagai berikut.

$f_{ew}(x)$: Durasi dari waktu libur terpanjang untuk perawat *e* dalam minggu *w* pada solusi *x*. Parameter ini digunakan untuk mengitung waktu libur perawat sehingga tidak melanggar *hard constraint* 6. Setiap perawat memiliki nilai $f_{ew}(x)$ yang berbeda-beda sesuai dengan jadwal kerja yang terbentuk.

$u_{ei}(x)$: Jumlah pola sif yang tidak diinginkan oleh perawat *e* untuk tipe sif *i* pada solusi *x*.

Parameter ini digunakan untuk menghitung nilai pelanggaran *soft constraint* 9.

$v_{ei}(x)$: Jumlah pola sif yang diinginkan oleh perawat *e* untuk tipe sif *i* pada solusi *x*. Parameter ini digunakan untuk menghitung nilai pelanggaran *soft constraint* 8.

Selanjutnya model matematis yang telah dibuat memiliki variabel keputusan sebagai berikut.

q_{eds} : Bernilai 1 jika perawat *e* dijadwalkan pada hari *d* dengan kategori sif *s*, bernilai 0 jika sebaliknya.

4. 2. 2. *Hard Constraint*

Penelitian tugas akhir ini menggunakan *hard constraint* yang telah dipaparkan secara singkat pada BAB II subbab model matematis *dataset Norwegian hospital*. Terdapat 7 *hard constraint* untuk membuat solusi yang dapat diterima (*feasible*). Berikut penjelasan dari setiap *hard constraint* yang telah dibuat.

Hard constraint 1. Satu perawat setiap harinya dialokasikan maksimal satu sif kerja.

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

Dimana q_{eds} merupakan variabel keputusan yang bernilai 1 jika perawat *e* dijadwalkan pada hari *d* dengan sif *s*. Variabel keputusan tersebut diakumulasi sesuai dengan banyaknya sif pada unit rumah sakit. Jika akumulasi tersebut bernilai kurang dari 1, maka *hard constraint* 1 telah terpenuhi. Hal tersebut juga menandakan bahwa setiap perawat mendapatkan paling banyak satu sif setiap harinya.

Hard constraint 2. Kebutuhan sif harus tepat terpenuhi setiap harinya.

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

Dimana q_{eds} merupakan variabel keputusan yang bernilai 1 jika perawat *e* dijadwalkan pada hari *d* dengan sif *s*. Variabel keputusan tersebut diakumulasi sesuai dengan banyaknya perawat pada unit rumah sakit. Kemudian hasil akumulasi tersebut akan dibandingkan dengan kebutuhan perawat (R_{ds}). Jika hasil akumulasi tersebut sama dengan kebutuhan perawat, maka *hard constraint* 2 telah terpenuhi.

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

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

Model matematis tersebut akan mengakumulasikan jam kerja perawat selama periode penjadwalan tertentu. Akumulasi jam kerja perawat tersebut tidak boleh melebihi batas minimal $(1-0.02)T_e^h$ dan maksimal $(1-0.02)T_e^h$ yang telah ditentukan. Penentuan batas tersebut mengacu pada nilai T_e^h yang merupakan jam kerja perawat selama periode penjadwalan tertentu.

Hard constraint 4. Perawat hanya dapat bekerja pada sif yang sesuai dengan kompetensi yang dimiliki.

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

Dimana B_{es} bernilai 1 jika perawat *e* memiliki suatu keahlian untuk bekerja di sif *s*. Model matematis tersebut akan mengakumulasikan nilai $B_{es}q_{eds}$ untuk semua sif yang tersedia. Hasil dari akumulasi pada model matematis tersebut harus bernilai 1. Nilai 1 menunjukan bahwa karyawan *e* dapat dijadwalkan pada sif *s*.

Hard constraint 5. Waktu jeda antar dua sif yang berurutan harus melebihi batas minimal yang telah ditentukan.

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

Model matematis tersebut digunakan untuk memastikan bahwa waktu jeda antar dua sif yang berurutan melebihi batas minimal yang telah ditentukan. Pada model matematis, nilai s dan s' merupakan sebuah pasangan sif, dimana s merupakan sif pada hari d dan s' merupakan sif pada hari d + 1. Penentuan pasangan sif tersebut dilihat dari waktu jeda antara kedua sif. Terdapat beberapa pasangan sif yang harus memiliki waktu jeda melebihi batas yang telah ditentukan, yaitu pasangan sif dengan kategori sif malam dan sif siang, sif siang dan sif pagi, sif malam dan sif pagi. Jika pasangan sif tersebut memiliki waktu jeda kurang dari batas yang telah ditentukan, maka pasangan sif tersebut masuk ke dalam himpunan pasangan sif yang tidak sesuai (I_d). Model matematis tersebut akan menghitung penjumlahan nilai q_{eds} dan $q_{e(d+1)s'}$. Jika hasil penjumlahan lebih dari 1, maka jadwal kerja yang terbentuk masih memiliki pasangan sif yang memiliki waktu jeda kurang dari batas minimal yang telah ditentukan.

Hard constraint 6. Pada setiap minggu harus ada minimal periode bebas kerja.

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

Dimana $f_{ew}(x)$ merupakan durasi dari waktu libur terpanjang untuk perawat *e* dalam minggu *w* pada solusi *x*. Durasi tersebut harus lebih besar dari nilai batas yang telah ditentukan. Hal ini bertujuan untuk memastikan bahwa setiap perawat memiliki waktu libur yang cukup pada setiap minggunya.

Hard connstraint 7. Tidak boleh melanggar batas maksimal jam kerja satu minggu.

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

Model matematis tersebut akan mengakumulasikan jam kerja perawat setiap minggunya. Jam kerja perawat tesebut tidak boleh melebihi nilai batas yang telah ditentukan (T_e). Hal

tersebut bertujuan agar perawat tidak memiliki jam kerja terlalu banyak.

4. 2. 3. *Soft Constraint*

Penelitian tugas akhir ini menggunakan *soft constraint* yang telah dipaparkan secara singkat pada BAB II subbab model matematis *dataset Norwegian hospital*. Terdapat 9 *soft constraint* yang menjadi perhitungan nilai penalti pada solusi yang berhasil terbentuk. Setiap *soft constraint* memiliki nilai batas yang berbeda-beda. Nilai batas tersebut telah ditentukan oleh setiap unit rumah sakit. Secara umum, nilai batas tersebut digunakan sebagai parameter pada model matematis *soft constraint.* Berikut penjelasan dari setiap *soft constraint* yang telah dibuat.

Soft constraint 1. Tidak boleh terlalu banyak hari kerja berurutan dengan kategori sif yang sama.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 1. Nilai penalti *soft constraint* 1 muncul akibat terlalu banyaknya hari kerja berurutan dengan kategori sif yang sama. Batas maksimal hari kerja berurutan dengan kategori sif yang sama telah ditentukan oleh unit rumah sakit yang bersangkutan (N_c^{max}). Batas maksimal tersebut terbagi atas 3 kategori sif, yaitu sif pagi, sif siang, dan sif malam. Setiap kategori sif memiliki batas maksimalnya masing-masing. Secara umum, batas maksimal yang ditetapkan yaitu 6, namun terdapat beberapa unit rumah sakit yang memiliki batas maksimal kurang dari 6. Variabel y_{ecd} akan bernilai 1 jika perawat *e* bekerja pada kategori sif *c* di hari *d*, dan akan bernilai 0 jika sebaliknya. Variabel tersebut akan menghitung hari kerja berurutan berdasarkan kategori sif yang sama untuk semua perawat. Jika pada jadwal kerja terdapat hari kerja berurutan dengan kategori sif sama yang melebihi batas, maka jadwal kerja tersebut akan mendapatkan nilai penalti

sebesar 1. Nilai penalti tersebut kemudian diakumulasikan untuk semua hari, perawat dan kategori sif.

Soft constraint 2. Tidak boleh terlalu banyak hari kerja berurutan.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 2. Nilai penalti *soft constraint* 2 muncul akibat terlalu banyaknya hari kerja berurutan. Batas maksimal hari kerja berurutan telah ditentukan oleh unit rumah sakit yang bersangkutan (N^{max}). Variabel z_{ed} akan bernilai 1 jika perawat *e* bekerja pada hari *d*, dan akan bernilai 0 jika sebaliknya. Variabel tersebut akan menghitung hari kerja berurutan untuk semua perawat. Jika pada jadwal kerja terdapat hari kerja berurutan yang melebihi batas, maka jadwal kerja tersebut akan mendapatkan nilai penalti sebesar 1. Nilai penalti tersebut kemudian diakumulasikan untuk semua hari dan perawat.

Soft constraint 3. Tidak boleh terlalu sedikit hari kerja berurutan dengan kategori sif yang sama.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 3. Nilai penalti *soft constraint* 3 muncul akibat terlalu sedikitnya hari kerja berurutan dengan kategori sif yang sama. Batas minimal hari kerja berurutan dengan kategori sif yang sama telah ditentukan oleh unit rumah sakit bersangkutan (N_c^{min}). Batas minimal tersebut terbagi atas 3 kategori sif, yaitu sif pagi, sif siang, dan sif malam. Secara umum, batas minimal yang ditentukan yaitu 2. Variabel y_{ecd} akan bernilai 1 jika perawat *e* bekerja pada kategori sif *c* di hari *d*, dan akan bernilai 0 jika sebaliknya. Variabel tersebut akan menghitung hari kerja berurutan dengan

kategori sif yang sama. Jika pada jadwal kerja terdapat hari kerja dengan kategori sif sama yang tidak memenuhi batas minimal, maka jadwal kerja tersebut akan mendapatkan nilai penalti sebesar nilai N_c^{min}. Nilai penalti tersebut kemudian diakumulasikan untuk semua hari, perawat dan kategori sif.

Soft constraint 4. Tidak boleh terlalu sedikit hari kerja berututan.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 4. Nilai penalti *soft constraint* 4 muncul akibar terlalu sedikitnya hari kerja berurutan. Batas minimal hari kerja berurutan telah ditentukan oleh unit rumah sakit yang bersangkutan (N^{min}). Variabel z_{ed} akan bernilai 1 jika perawat *e* bekerja pada hari *d*, dan akan bernilai 0 jika sebaliknya. Variabel tersebut akan menghitung hari kerja berurutan untuk semua perawat. Jika pada jadwal kerja terdapat hari kerja berurutan yang tidak memenuhi batas minimal, maka jadwal kerja tersebut akan mendapatkan nilai penalti sebesar nilai N^{min}. Nilai penalti tersebut kemudian diakumulasikan untuk semua hari dan perawat.

Soft constraint 5. Tidak boleh menyimpang terlalu jauh dari batas minimal dan maksimal jumlah sif pada setiap kategorinya.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 5. Nilai penalti *soft constraint* 5 muncul akibat adanya penyimpangan terlalu jauh dari jumlah minimal dan maksimal jumlah sif pada setiap kategorinya. Batas minimal (N_{ec}^{min}) dan maksimal (N_{ec}^{max}) masing-masing kategori sif telah ditentukan oleh unit rumah sakit yang bersangkutan. Variabel y_{ecd} akan bernilai 1 jika perawat *e* bekerja pada kategori sif *c* di hari *d*, dan akan bernilai

0 jika sebaliknya. Variabel tersebut akan menghitung hari kerja berurutan dengan kategori sif yang sama. Jika pada jadwal kerja terdapat hari kerja yang melebihi batas minimal dan maksimal, maka jadwal kerja tersebut akan mendapatkan nilai penalti sebesar nilai N_{ec}^{min} atau N_{ec}^{max}. Nilai tersebut kemudian dikuadratkan dan diakumulasikan untuk semua perawat dan kategori sif. Setelah itu, nilai akumulasi tersebut akan diakar kuadrat untuk mendapatkan nilai penalti pelanggaran *soft constraint* 5.

Soft constraint 6. Tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 6. Nilai penalti *soft constraint* 6 muncul akibat adanya ketidaksesuaian antara jam kerja perawat dengan kontrak kerja perawat. Pada model matematis, variabel T_e^h merupakan kontrak kerja selama periode penjadwalan tertentu. Jika pada jadwal kerja yang terbentuk terdapat jam kerja perawat yang tidak sesuai kontrak kerja, maka jadwal kerja tersebut akan mendapatkan nilai penalti sebesar selisih antara kontrak kerja perawat dengan jam kerja perawat. Selisih tersebut kemudian dikuadratkan dan diakumulasikan untuk semua perawat. Setelah itu, nilai akumulasi tersebut akan diakar kuadrat untuk mendapatkan nilai penalti pelanggaran *soft constraint* 6.

Soft constraint 7. Mengelompokan waktu libur setiap perawat.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 7. Nilai penalti *soft constraint* 7 muncul akibat tidak adanya pengelompokan waktu

libur untuk perawat. Sebagai contoh, pengelompokan waktu libur yang dimaksud yaitu libur pada hari Selasa dan Rabu lebih baik daripada hari Selasa dan Kamis. Pada model matematis, variabel z_{ed} akan bernilai 1 jika perawat *e* bekerja pada hari *d*, dan akan bernilai 0 jika sebaliknya. Perhitungan nilai penalti *soft constraint* 7 dilakukan dengan melihat nilai maksimal antara nilai 0 dan nilai $z_{ed} - z_{e(d+1)}$. Nilai tersebut kemudian diakumulasikan dan dikuadratkan untuk semua hari pada jadwal kerja. Setelah itu, nilai akumulasi tersebut akan diakar kuadrat untuk mendapatkan nilai penalti pelanggaran *soft constaint* 7.

Soft constraint 8. Memenuhi pola sif yang diinginkan.

$$p_8(x) = |E||D| - \sum_{e \in E, d \in D} v_{ei}\,(x)$$

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 8. Nilai penalti *soft constraint* 8 muncul akibat tidak memenuhi pola sif yang diinginkan. Variabel $|E|$ merupakan jumlah perawat dan variabel $|D|$ merupakan jumlah hari pada jadwal kerja perawat. Perkalian antara nilai dua variabel tersebut akan menghasilkan nilai yang sangat besar. Hal tersebut menyebabkan penalti *soft constraint* 8 tidak akan bernilai negatif. Nilai penalti *soft constraint* 8 akan turun seiring dengan semakin banyaknya pola sif yang diinginkan pada jadwal kerja perawat.

Soft constraint 9. Tidak boleh terlalu banyak pola sif yang tidak diinginkan.

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

Model matematis tersebut digunakan untuk menghitung nilai penalti pelanggaran *soft constraint* 9. Nilai penalti *soft constraint* 9 muncul akibat terlalu banyaknya pola sif yang tidak diinginkan. Variabel u_{ei} merupakan jumlah pola sif yang tidak diinginkan pada jadwal kerja perawat. Nilai penalti *soft constraint* 9 akan bertambah 1 jika pada jadwal kerja perawat terdapat 1 pola sif yang tidak diingiinkan. Nilai tersebut akan

bertambah seiring dengan semakin banyaknya pola sif yang tidak diinginkan pada jadwal kerja perawat.

Kemudian terdapat parameter tambahan yang digunakan untuk menghitung nilai pelanggaran *soft constraint* yaitu:

$$y_{ecd} = \sum_{s \in S} \mathrm{A_{sc}} q_{\mathrm{eds}}$$

Dimana y_{ecd} akan bernilai 1 jika perawat *e* berkerja pada sif kategori *c* pada hari *d*, bernilai 0 jika sebaliknya.

$$z_{ed} = \sum_{s \in S} q_{\mathrm{eds}}$$

Dimana z_{ed} akan bernilai 1 jika perawat *e* berkerja pada hari *d*, bernilai 0 jika sebaliknya.

4. 2. 4. Fungsi Tujuan

Penelitian tugas akhir ini menggunakan fungsi tujuan yang telah dipaparkan secara singkat pada BAB II subbab model matematis *dataset Norwegian hospital*. Secara umum, fungsi tujuan dari penjadwalan perawat rumah sakit yaitu meminimalkan nilai penalti dari jadwal kerja yang terbentuk. Nilai penalti tersebut didapatkan dari akumulasi nilai penalti pelanggaran *soft constraint* 1 hingga *soft constraint* 9. Berikut merupakan model matematis yang digunakan untuk mengitung fungsi tujuan dari penjadwalan perawat rumah sakit.

$$Min\ f = \sum_{m=1}^{9} K_m p_m$$

Variabel K_m merupakan nilai tetap yang telah ditentukan oleh beberapa pemangku kepentingan, sedangkan variabel p_m merupakan nilai penalti dari masing-masing *soft constraint*. Pada penelitian sebelumnya, terdapat tiga nilai K_m yang ditentukan oleh tiga pemangku kepentingan yang berbeda.

Namun pada penelitian tugas akhir ini hanya menggunakan satu nilai K_m karena kurangnya informasi yang didapat dari penelitian sebelumnya. Nilai K_m yang digunakan pada penelitian tugas akhir ini yaitu 1. Perkalian nilai p_m dengan angka 1 akan menghasilkan nilai p_m itu sendiri. Sehingga, fungsi tujuan dari penelitian tugas akhir ini yaitu penjumlahan dari nilai penalti masing-masing *soft constraint*.

4. 3. Pemodelan Algoritma

Pemodelan algoritma merupakan proses untuk memodelkan algoritma yang digunakan dalam optimasi permasalahan *Nurse Rostering Problem*. Pada tugas akhir ini menggunakan 1 algoritma utama dan 3 algoritma pembanding. Algoritma utama yang digunakan yaitu *Reinforcement Learning Simulated Annealing with Reheating*, sedangkan algoritma pembanding yang digunakan yaitu *Simple Random Hill Climbing, Reinforcement Learning Hill Climbing,* dan *Reinforcement Learning Simulated Annealing*. Algoritma tersebut digunakan setelah mendapatkan solusi awal yang dapat diterima (*feasible*). Secara singkat, algoritma yang digunakan dapat dilihat pada Tabel 4.2 berikut.

Tabel 4.2 Daftar Algoritma yang Digunakan dalam Optimasi *Nurse Rostering Problem*

Nama Algoritma	**Pemilihan *Low Level Heuristic***	***Move Acceptance***
SR-HC	*Simple Random*	*Hill Climbing*
RL-HC	*Reinforcement Learning*	*Hill Climbing*
RL-SA	*Reinforcement Learning*	*Simulated Annealing*
RL-SAR	*Reinforcement Learning*	*Simulated Annealing with Reheating*

4. 3. 1. Pemodelan Algoritma *Simple Random Hill Climbing*

Algoritma *Simple Random Hill Climbing* atau sering disebut algoritma *Hill Climbing* merupakan algoritma metaheuristika yang paling sederhana. Konsep dari algoritma ini dapat dianalogikan seperti pemanjat tebing. Algoritma ini akan mencari solusi yang lebih baik dan menolak solusi yang lebih buruk. Algoritma *Hill Climbing* akan mencari solusi selama iterasi yang telah ditentukan. *Simple Random* digunakan untuk memilih *Low Level Heuristic (LLH)* secara acak. *LLH* yang digunakan yaitu *2-Exchange*, *3-Exchange*, dan *double 2-Exchange*. *Pseudocode* dari algoritma *Hill Climbing* dapat dilihat pada Gambar 4.1 berikut.

```
1: procedure SR-HC (problem P) returns a state that is a
   local maximum
2:     Set stoping condition St
3:     Set M ← P.getNumberofHeuristic()
4:     P.initiateSolution(0)
5:     Sbest ← P.getPenalti(0)
6:     Scurr ← P.getPenalti(0)
7:     while St is not met do
8:            l ← selectRandomLLH(M)
9:            Scurr ← P.getPenalti(l)
10:           if Scurr < Sbest
11:                  Sbest ← Scurr
12:           end if
13:           else
14:                  Scurr ← Sbest
15:    end while
16:    return Sbest
17: end procedure
```

Gambar 4.1 Pseudocode Algoritma Simple Random Hill Climbing

4. 3. 2. Pemaodelan Algoritma *Reinforcement Learning Hill Climbing*

Algoritma *Reinforcement Learning Hill Climbing* merupakan modifikasi dari algoritma *Hill Climbing*. Secara konsep, kedua algoritma tersebut hampir sama. Perbedaannya terletak pada saat memilih *Low Level Heuristic* (*LLH*). Algoritma *Hill*

Climbing akan memilih *LLH* secara acak, sedangkan algoritma *Reinforcement Learning Hill Climbing* akan menggunakan konsep *Reinforcement Learning* untuk memilih *LLH*. Konsep dari *Reinforcement Learning* telah dijelaskan pada BAB II. Sederhananya, *Reinforcement Learning* akan memberikan penambahan nilai pada *LLH* yang memperoleh solusi yang lebih baik dan memberikan pengurangan nilai pada *LLH* yang memperoleh solusi yang lebih buruk. *LLH* yang digunakan yaitu *2-Exchange*, *3-Exchange*, dan *double 2-Exchange*. *Pseudocode* dari algoritma *Reinforcement Learning Hill Climbing* dapat dilihat pada Gambar 4.2 berikut.

```
1: procedure RL-HC (ProblemDomain P)
2:    Set stopping condition St
3:    Set M ← P.getNumberofHeuristic()
4:    Set R ← new int[M]
5:    initiateReward(R)
6:    P.initiateSolution
7:    Sbest ← P.getFunctionValue(0)
8:    Scurr ← P.getFunctionValue(0)
9:    while St is not met do
10:      l ← selectLLH(R)
11:      newFunctionVal ← P.applyLLH(1,0,1)
12:      if(newFunctionVal) ≤ Scurr then
13:          //the new solution is accepted
14:          P.copySolution(1,0)
15:          Scurr ← newFunctionVal
16:          upgradeReward(R[l])
17:          if newFunctionVal < Sbest then
18:              Set Sbest ← newFunctionVal
19:          newFunctionVal
20:          end if
21:     else
22:          P.copySolution(0,1)
23:          downgradeReward(R[l])
24:     end if
25:   end while
26:   return Sbest
27: end procedure
```

Gambar 4.2 Pseudocode Algoritma Reinforcement Learning Hill Climbing

4. 3. 3. Pemodelan Algoritma *Reinforcement Learning Simulated Annealing*

Algoritma *Reinforcement Learning Simulated Annealing* merupakan modifikasi dari algoritma *Simulated Annealing*. Secara konsep, kedua algoritma tersebut hampir sama. Perbedaannya terletak pada proses pemilihan *Low Level Heuristic*. Algoritma *Simulated Annealing* akan memilih *Low Level Heuristic* secara acak, sedangkan algoritma *Reinforcement Learning Simulated Annealing* akan menggunakan konsep *Reinforcemnet Learning* untuk memilih *Low Level Heuristic*. Konsep dari *Reinforcement Learning* telah dijelaskan pada BAB II. Sederhananya, *Reinforcement Learning* akan memberikan penambahan nilai pada *Low Level Heuristic* yang memperoleh solusi yang lebih baik dan memberikan pengurangan nilai pada *Low Level Heuristic* yang memperoleh solusi yang lebih buruk. *Low Level Heuristic (LLH)* yang digunakan yaitu *2-Exchange, 3-Exchange*, dan *double 2-Exchange*. Kemudian algoritma *Simulated Annealing* akan bertindak sebagai *move acceptance* terhadap kandidat solusi yang terbentuk. Kandidat solusi yang memiliki nilai penalti lebih baik akan diterima dan kandidat solusi yang memiliki nilai penalti lebih buruk akan diterima dengan probabilitas tertentu. *Pseudocode* dari algoritma *Reinforcement Learning Simulated Annealing* dapat dilihat pada Gambar 4.3 berikut.

```
1: procedure RL-SA (ProblemDomain P)
2:    Set stopping condition St
3:    Set M ← P.getNumberofHeuristic()
4:    //R is array to store reward given to each low-
      level heuristic
5:    Set R ← new int[M]
6:    initiateReward(R)
7:    P.initiateSolution
8:    Sbest ← P.getFunctionValue(0)
9:    Scurr ← P.getFunctionValue(0)
10:   Set Prob ← 0
11:   Set temp ← Scurr x C
12:   while St is not met do
```

```
13:      l ← selectLLH(R)
14:      newFunctionVal ← P.applyLLH(1,0,1)
15:      if(newFunctionVal) ≤ Scurr then
16:          //the new solution is accepted
17:          P.copySolution(1,0)
18:          Scurr ← newFunctionVal
19:          upgradeReward(R[l])
20:          if newFunctionVal < Sbest then
21:              Set Sbest ← newFunctionVal
22:          newFunctionVal
23:          end if
24:     else
25:          Set Prob ← exp(-(newFunctionVal−Scurr)/temp)
26:            if Prob ≥ RANDOM(0,1)
27:               Scurr ← newFunctionVal
28:            end if
29:          downgradeReward(R[l])
30:      end if
31:     temp ← temp x β
32:   end while
33:   return Sbest
34: end procedure
```

Gambar 4.3 Pseudocode Algoritma Reinforcement Learning Simulated Annealing

4. 3. 4. Pemodelan Algoritma *Reinforcement Learning Simulated Annealing with Reheating*

Algoritma *Reinforcement Learning Simulated Annealing with Reheating* merupakan modifikasi dari algoritma *Simulated Annealing with Reheating*. Secara konsep, kedua algoritma tersebut hampir sama. Perbedaannya terletak pada proses pemilihan *Low Level Heuristic*. Algoritma *Simulated Annealing with Reheating* akan memilih *Low Level Heuristic* secara acak, sedangkan algoritma *Reinforcement Learning Simulated Annealing with Reheating* akan menggunakan konsep *Reinforcemnet Learning* untuk memilih *Low Level Heuristic*. Konsep dari *Reinforcement Learning* telah dijelaskan pada BAB II. Sederhananya, *Reinforcement Learning* akan memberikan penambahan nilai pada *Low Level Heuristic* yang memperoleh solusi yang lebih baik dan memberikan pengurangan nilai pada

Low Level Heuristic yang memperoleh solusi yang lebih buruk. *Low Level Heuristic (LLH)* yang digunakan yaitu *2-Exchange, 3-Exchange,* dan *double 2-Exchange*. Kemudian algoritma *Simulated Annealing with Reheating* akan bertindak sebagai *move acceptance* terhadap kandidat solusi yang terbentuk. Kandidat solusi yang memiliki nilai penalti lebih baik akan diterima dan kandidat solusi yang memiliki nilai penalti lebih buruk akan diterima dengan probabilitas tertentu. Ketika *Low Level Heuristic* cenderung menghasilkan kandidat solusi dengan nilai penalti yang tidak lebih baik secara terus menerus, maka akan dilakukan pemanasan ulang suhu. *Pseudocode* dari algoritma *Reinforcement Learning Simulated Annealing with Reheating* dapat dilihat pada Gambar 4.4 berikut.

```
1: procedure RL-SAR (ProblemDomain P)
2:    Set stopping condition St
3:    Set M ← P.getNumberofHeuristic()
4:    //R is array to store reward given to each low-
      level heuristic
5:    Set R ← new int[M]
6:    initiateReward(R)
7:    P.initiateSolution
8:    Sbest ← P.getFunctionValue(0)
9:    Scurr ← P.getFunctionValue(0)
10:   Set Prob ← 0
11:   Set temp ← Scurr x C
12:   heat ← 0
13:   previousCost ← Scurr
14:   currentStagnantCount ← 0
15:   stuckBestCost ← Scurr
16:   stuckCurrentCost ← Scurr
17:   while St is not met do
18:      l ← selectLLH(R)
19:      newFunctionVal ← P.applyLLH(1,0,1)
20:      if(newFunctionVal) ≤ Scurr then
21:         //the new solution is accepted
22:         P.copySolution(1,0)
23:         Scurr ← newFunctionVal
24:         upgradeReward(R[l])
25:         if newFunctionVal < Sbest then
26:            Set Sbest ← newFunctionVal
27:         newFunctionVal
```

```
28:          end if
29:      else
30:          Set Prob ← exp(-(newFunctionVal−Scurr)/temp)
31:            if Prob ≥ RANDOM(0,1)
32:               Scurr ← newFunctionVal
33:            end if
34:          downgradeReward(R[1])
35:      end if
36:      if f(current) – previousCost < 1% then
37:              currentStagnantCount =
                 currentStagnantCount + 1
38:              if currentStagnantCount > 50000 then
39:                 if Sbest = stuckBestCost then
40:                    if Scurr – stuckCurrentCost < 2%
                       then
41:                       heat = heat + 1
42:                    else
43:                       heat ← 0
44:                    end if
45:                 else
46:                    heat ← 0
47:                 end if
48:                 temp <- [heat x 0.2 x Scurr + Scurr] x C
49:                 stuckBestCost ← Sbest
50:                 stuckCurrentCost ← Scurr
51:              end if
52:          else
53:              currentStagnantCount ← 0
54:          end if
55:        else
56:            temp ← temp x β
57:        end if
58:   end while
59:   return Sbest
60: end procedure
```

Gambar 4.4 Pseudocode Algoritma Reinforcement Learning Simulated Annealing with Reheating

4. 4. Skenario Penyusunan Solusi Awal

Skenario penyusunan solusi awal bertujuan untuk mengetahui metode penyusunan solusi awal yang tepat. Skenario yang menghasilkan paling banyak solusi awal yang *feasible* akan

dipilih dan digunakan dalam penyusunan solusi awal. Terdapat 2 skenario penyusunan solusi awal yang digunakan pada peneltian tugas akhir ini. Tabel 4.3 menampilkan skenario penyusunan solusi awal pada penelitian tugas akhir ini.

Tabel 4.3 Daftar Skenario Penyusunan Solusi Awal

No	Skenario
1	Skenario pengecekan *hard constraint* 3
2	Skenario pengecekan *hard constraint* 3 dan 6

Skenario penyusunan solusi awal dijalankan selama 50000 iterasi dengan menukar sif yang telah dijadwalkan hingga mendapatkan solusi awal yang *feasible*. Skenario pengecekan *hard constraint* 3 dilakukan dengan menghitung selisih jam kerja perawat dengan kontrak kerja yang telah ditentukan. Jika selisih jam kerja setiap iterasi menurun, maka solusi tersebut menjadi kandidat solusi awal yang *feasible*. Skenario pengecekan *hard constraint* 3 dan 6 dilakukan dengan menghitung selisih jam kerja perawat dengan kontrak kerja yang telah ditentukan dan mengitung banyaknya pelanggaran yang disebabkan oleh *hard constraint* 6. Jika selisih jam kerja dan pelanggaran *hard constraint* 6 setiap iterasi menurun, maka solusi tersebut menjadi kandidat solusi awal yang *feasible*. Proses tersebut dilakukan hingga akhir iterasi dan mendapatkan solusi awal yang *feasible*.

4. 5. Skenario Uji Coba Parameter Algoritma *RL-SAR*

Uji coba parameter bertujuan untuk mengetahui parameter terbaik dari algoritma RL-SAR. Uji coba parameter dilakukan dengan cara membuat beberapa skenario dengan kombinasi parameter yang bervariasi. Terdapat 4 parameter yang digunakan untuk melakukan uji coba. Setiap parameter memiliki 3 skenario pengujian. Pada Tabel 4.4 menampilkan daftar skenario uji coba yang dilakukan pada penelitian tugas akhir ini.

Tabel 4.4 Daftar Skenario Uji Coba Parameter

NO	PARAMETER			
	HEURISTIC SELECTION	***UTILIITY UPDATE***	***COOLING RATE***	***REHEATING FREQUENCY***
	UJI COBA *HEURISTIC SELECTION*			
1	Greedy	Sum	0.99995	50000
2	Epsilon Greedy	Sum	0.99995	50000
3	Epsilon Decay Greedy	Sum	0.99995	50000
	UJI COBA *UTILITY UPDATE*			
4	Epsilon Decay Greedy	Sum	0.99995	50000
5	Epsilon Decay Greedy	Average	0.99995	50000
6	Epsilon Decay Greedy	Discount Factor	0.99995	50000
	UJI COBA *COOLING RATE*			
7	Epsilon Decay Greedy	Sum	0.5	50000
8	Epsilon Decay Greedy	Sum	0.85	50000
9	Epsilon Decay Greedy	Sum	0.99995	50000
	UJI COBA *REHEATING FREQUENCY*			
10	Epsilon Decay Greedy	Sum	0.99995	10000
11	Epsilon Decay Greedy	Sum	0.99995	50000
12	Epsilon Decay Greedy	Sum	0.99995	100000

Skenario uji coba *heuristic selection* bertujuan untuk menentukan metode pemilihan *Low Level Heuristic.* Skenario uji coba *utility update* bertujuan untuk menentukan mekanisme *reward/punishment* pada *Low Level Heuristic*. Skenario uji coba *cooling rate* bertujuan untuk mengurangi suhu pada algoritma

Simulated Annealing with Reheating secara bertahap. Skenario uji coba *reheating frequency* bertujuan untuk menentukan frekuensi peningkatan suhu pada saat algoritma tidak menemukan solusi yang lebih baik. Skenario tersebut kemudian akan diimplementasikan pada beberapa unit rumah sakit untuk mendapatkan jadwal kerja yang memiliki nilai penalti paling kecil. Skenario yang menghasilkan jadwal kerja dengan nilai penalti paling kecil akan digunakan untuk melihat performa dari algortima *Reinforcement Learning – Simulated Annealing with Reheating*.

4. 6. Ringkasan Perancangan

Tahap perancangan merupakan tahap awal pengerjaan penelitian tugas akhir sebelum melakukan implementasi pada permasalahan yang ada. Tahap perancangan terdiri dari pemahaman dataset, pemahaman model matematis, skenario penyusunan solusi awal, dan skenario uji coba parameter. Uraian pemahaman dataset, model matematis, dan *pseudocode* akan menjadi dasar penyelesaian masalah pada proses implementasi. Selanjutnya, proses implementasi akan dijelaskan pada bab 5.

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi dari perancangan yang telah dilakukan sesuai dengan metodologi penelitian pada Bab 3 dan perancangan pada Bab 4. Bagian implementasi akan menjelaskan mengenai pembacaan data masukan, pembuatan solusi awal, serta optimasi solusi. Seluruh kode program proses implementasi dapat dilihat pada Lampiran A-1.

5. 1. Pembacaan dan Penyimpanan Data Masukan

Penelitian tugas akhir ini menggunakan *dataset* salah satu rumah sakit di Norwegia. *Dataset* tersebut terdiri atas data 7 unit rumah sakit yang tersimpan pada berkas excel dengan ekstensi .xls. Setiap unit rumah sakit menyimpan data tentang perawat, sif, kebutuhan perawat, batasan dan pola sif yang diinginkan atau tidak diinginkan. Masing-masing data tersimpan pada lembar kerja yang berbeda-beda, sehingga pembacaan data masukan dilakukan sebanyak 5 kali. Setelah terbaca, masing-masing data disimpan ke dalam *array* dengan tipe data *String*. Kemudian melakukan tahap pengubahan tipe data sesuai dengan kebutuhan. Setelah mendapatkan data dengan tipe data yang sesuai, maka tahap selanjutnya yaitu membuat solusi awal.

5. 2. Peembuatan Solusi Awal

Setelah membaca dan menyimpan data masukan, tahap selanjutnya yaitu membuat solusi awal dari penjadwalan perawat rumah sakit di Norwegia. Solusi awal yang terbentuk harus memenuhi seluruh batasan-batasan yang ada. Terdapat 4 tahap yang dilakukan, yaitu pembuatan matriks solusi awal, pengecekan pelanggaran *hard constraint*, penyusunan solusi awal, dan perhitungan nilai penalti solusi awal.

5. 2. 1. Pembuatan Matriks Solusi Awal

Tahap pertama dalam membuat solusi awal yaitu membuat matriks dalam bentuk *array* 2 dimensi. Matriks tersebut terdiri

dari beberapa baris dan kolom. Baris pada matriks menunjukan jumlah perawat pada unit rumah sakit. Kolom pada matriks menunjukan panjang penjadwalan yang dilakukan pada unit rumah sakit. Matriks yang terbentuk memiliki nilai awal 0.

5. 2. 2. Pengecekan Pelanggaran *Hard Constraint*

Setelah matriks terbentuk, tahap selanjutnya yaitu membuat *method* yang digunakan untuk pengecekan terhadap pelanggaran masing-masing *hard constraint*. Hal ini bertujuan agar solusi awal yang terbentuk merupakan solusi yang *feasible*. Studi kasus pada penelitian tugas akhir ini memiliki 7 *hard constraint* yang harus terpenuhi. Masing-masing *hard constraint* memiliki 1 *method* pengecekan, kecuali *hard constraint* 1.

5. 2. 2. 1. Pengecekan Pelanggaran *Hard Constraint* 2

Hard constraint 2 bertujuan untuk memastikan bahwa jadwal kerja perawat yang terbentuk sesuai dengan kebutuhan perawat setiap harinya. Tahap pertama pengecekan *hard constraint* 2 yaitu membuat *method needs* untuk menghitung jumlah sif yang telah dijadwalkan. Kemudian dalam penyusunan jadwal kerja perawat, terdapat *method checkHC2* untuk memastikan bahwa sif yang akan dijadwalkan tidak melebihi jumlah sif dari kebutuhan perawat. *Method checkHC2* memilki nilai pengembalian berupa *boolean* yang akan bernilai *true* jika sif yang akan dijadwalkan tidak melebihi jumlah sif dari kebutuhan perawat dan akan bernilai *false* jika sebaliknya. Selanjutnya untuk memastikan bahwa jadwal kerja perawat yang terbentuk sesuai dengan kebutuhan perawat, maka dibuat sebuah *method validHC2* dengan nilai pengembalian berupa *boolean*. *Method validHC2* akan bernilai *true* jika sif pada jadwal kerja yang terbentuk sesuai dengan kebutuhan perawat dan akan bernilai *false* jika sebaliknya.

5. 2. 2. 2. Pengecekan Pelanggaran *Hard Constraint* 3

Hard constraint 3 bertujuan untuk memastikan bahwa jam kerja perawat pada jadwal yang terbentuk tidak menyimpang terlalu jauh dari kontrak jam kerja perawat. Tahap pertama pengecekan *hard constraint* 3 yaitu membuat *method sumHours* untuk menghitung jam kerja perawat setiap minggunya dan disimpan pada sebuah *array* 1 dimensi. Setelah menghitung jam kerja masing-masing perawat, tahap selanjutnya yaitu menghitung rata-rata jam kerja masing-masing perawat dengan membuat *method avgHours*. *Method avgHours* akan mengambil nilai dari *array* 1 dimensi pada *method sumHours* dan dijumlahkan untuk masing-masing perawat. Setelah mendapatkan total jam kerja masing-masing perawat, maka tahap selanjutnya yaitu membagi total jam kerja dengan jumlah minggu penjadwalan untuk mendapatkan rata-rata jam kerja perawat. Setelah mendapatkan rata-rata jam kerja masing-masing perawat, maka tahap selanjutnya yaitu melakukan pengecekan terhadap jadwal yang terbentuk dengan membuat *method validHC3*. Jika jadwal yang terbentuk melebihi batas minimal dan maksimal dari kontrak kerja perawat, maka jadwal tersebut ditolak dan harus dilakukan perpindahan sif hingga jadwal tersebut tidak melanggar *hard constraint* 3.

5. 2. 2. 3. Pengecekan Pelanggaran *Hard Constraint* 4

Terdapat 2 tahap yang dilakukan untuk memenuhi *hard constraint* 4, yaitu pengecekan kompetensi perawat dan pengecekan jadwal kerja akhir pekan. Pengecekan kompetensi perawat bertujuan untuk memastikan bahwa sif yang membutuhkan keahlian tertentu hanya dapat dijadwalkan pada perawat dengan keahlian tertentu pula. Tedapat *method checkHC4Competence* dengan nilai pengembalian berupa *boolean* yang akan bernilai *true* jika sif yang dijadwalkan sesuai dengan keahlian perawat dan akan bernilai *false* jika sebaliknya. Tahap kedua yaitu pengecekan jadwal kerja akhir pekan yang bertujuan untuk memastikan bahwa pada akhir pekan perawat hanya bisa bekerja sesuai dengan kontrak kerja. Terdapat

method checkHC4Week dengan nilai pengembalian berupa *boolean* yang akan bernilai *true* jika perawat dapat dijadwalkan pada akhir pekan dan akan bernilai *false* jika sebaliknya. Selanjutnya untuk memastikan bahwa pada jadwal kerja yang terbentuk tidak melanggar *hard constraint* 4, maka dibuat *method validHC4Competence* dan *validHC4Week*. Kedua *method* tersebut memiliki nilai pengembalian berupa *boolean* yang akan bernilai *true* jika pada jadwal kerja yang terbentuk tidak melanggar *hard constraint 4* dan akan bernilai *false* jika sebaliknya.

5. 2. 2. 4. Pengecekan Pelanggaran *Hard Constraint* 5

Hard constraint 5 bertujuan untuk memastikan bahwa jadwal kerja perawat yang terbentuk memiliki waktu jeda antar dua sif yang berurutan melebihi batas minimal yang telah ditentukan. Tahap pertama pengecekan *hard constraint* 5 yaitu membuat pasangan sif yang tidak sesuai dan disimpan pada sebuah *array*. Penentuan pasangan sif yang dilarang dilihat dari waktu jeda antar dua sif yang berurutan. Jika waktu jeda tersebut kurang dari batasan yang telah ditentukan, maka pasangan sif tersebut akan masuk ke dalam pasangan sif yang tidak sesuai. Setelah pasangan sif yang tidak sesuai terbentuk, tahap selanjutnya yaitu memastikan bahwa pada saat penyusunan jadwal kerja tidak terdapat pasangan sif yang tidak sesuai. Hal tersebut dapat dilakukan dengan membuat *method checkHC5* yang memiliki nilai pengembalian berupa *boolean*. *Method* tersebut akan bernilai *true* jika sif yang akan dijadwalkan tidak termasuk ke dalam pasangan sif yang tidak sesuai dan akan bernilai *false* jika sebaliknya. Selanjutnya untuk memastikan bahwa jadwal kerja yang terbentuk tidak memiliki pasangan sif yang tidak sesuai, maka dibuat *method validHC5* yang memiliki nilai pengembalian berupa *boolean*. *Method* tersebut akan bernilai *true* jika pada jadwal kerja yang terbentuk tidak melanggar *hard constraint* 5 dan akan bernilai *false* jika sebaliknya.

5. 2. 2. 5. Pengecekan Pelanggaran *Hard Constraint* 6

Hard constraint 6 bertujuan untuk memastikan bahwa setiap perawat memiliki minimal waktu libur setiap minggunya. Tahap pertama pengecekan *hard constraint* 6 yaitu membuat sebuah *method checkFreeWeekly* untuk memastikan bahwa setiap perawat memiliki minimal waktu libur setiap minggunya. *Method* tersebut memiliki nilai pengembalian berupa *boolean* yang akan bernilai *true* jika perawat tersebut memiliki minimal waktu libur setiap minggunya dan akan bernilai *false* jika sebaliknya. Pada *method checkFreeWeekly* terdapat *array* 1 dimensi yang digunakan untuk menyimpan nilai pengembalian dari *method* tersebut. Tahap selanjutnya yaitu melakukan pengecekan terhadap jadwal kerja yang terbentuk dengan membuat *method validHC6*. *Method* tersebut memiliki nilai pengembalian berupa *boolean* yang akan bernilai *true* jika pada jadwal kerja yang terbentuk tidak melanggar *hard constraint* 6 dan akan bernilai *false* jika sebaliknya.

5. 2. 2. 6. Pengecekan Pelanggaran *Hard Constraint* 7

Hard constraint 7 bertujuan untuk memastikan bahwa setiap perawat memiliki maksimal jam kerja setiap minggunya. Tahap pertama pengecekan *hard constraint* 7 yaitu membuat *method workingHour* yang akan menghitung jam kerja perawat setiap minggunya. Setelah menghitung jam kerja perawat setiap minggunya, tahap selanjutnya yaitu memastikan bahwa pada saat penyusunan jadwal kerja tidak melanggar *hard constraint* 7. Hal tersebut dapat dilakukan dengan membuat *method checkHC7* yang memiliki nilai pengembalian berupa *boolean*. *Method* tersebut akan bernilai *true* jika sif yang akan dijadwalkan tidak membuat jam kerja perawat melebihi batas yang ditentukan dan akan bernilai *false* jika sebaliknya. Tahap selanjutnya yaitu melakukan pengecekan terhadap jadwal kerja yang terbentuk dengan membuat *method validHC7*. *Method* tersebut memiliki nilai pengembalian berupa *boolean* yang akan bernilai *true* jika pada jadwal kerja yang terbentuk tidak

melanggar *hard constraint* 7 dan akan bernilai *false* jika sebaliknya.

5. 2. 3. Penyusunan Solusi Awal

Setelah menyusun *method* pengecekan *hard constraint*, maka tahap selanjutnya yaitu menyusun solusi awal. Solusi awal yang terbentuk berupa jadwal kerja perawat sepanjang periode penjadwalan yang telah ditentukan. Tahap pertama dalam penyusunan jadwal yaitu menyusun jadwal kerja untuk hari sabtu dan minggu terlebih dahulu. Hal ini dilakukan karena jadwal kerja sabtu dan minggu hanya bisa dialokasikan untuk beberapa perawat sesuai dengan kontrak kerja. Penyusunan jadwal kerja untuk hari sabtu dan minggu dilakukan secara acak pada beberapa perawat. Setelah menyusun jadwal kerja untuk hari sabtu dan minggu, maka tahap selanjutnya yaitu menyusun jadwal kerja untuk hari senin hingga jumat. Penyusunan jadwal kerja hari senin hingga jumat juga dilakukan secara acak untuk semua perawat. Jadwal kerja yang terbentuk merupakan jadwal kerja sementara yang sudah memenuhi *hard constraint* 2, 4, 5, dan 7. Pemenuhan terhadap *hard constraint* 3 dan 6 dilakukan dengan cara menukar beberapa sif dengan memperhatikan selisih antara jam kerja dengan kontrak kerja dan banyaknya pelanggaran terhadap *hard constraint* 6. Tahap penukaran sif tersebut dilakukan secara terus menerus hingga jadwal kerja yang *feasible* terbentuk.

5. 2. 4. Perhitungan Nilai Penalti Solusi Awal

Setelah solusi awal terbentuk, tahap selanjutnya yaitu membuat *method* yang digunakan untuk mengitung nilai penalti pelanggaran *soft constraint*. Studi kasus pada penelitian tugas akhir ini memiliki 9 *soft constraint*. Masing-masing *soft constraint* memiliki 1 *method* yang memiliki nilai pengembalian berupa perhitungan nilai penalti.

5. 2. 4. 1. Perhitungan Nilai Penalti *Soft Constraint* 1

Soft constraint 1 bertujuan untuk membatasi terlalu banyaknya hari kerja berurutan dengan kategori sif yang sama. Pelanggaran terhadap *soft constraint* 1 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 1 dilakukan dengan membuat sebuah *method penalti1*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal dalam perhitungan nilai penalti *soft constraint* 1 yaitu mengidentifikasi batas maksimal hari kerja berurutan dari masing-masing kategori sif, yaitu sif pagi, sif siang, dan sif malam. Masing-masing kategori sif memiliki batas maksimal yang berbeda-beda. Jika batas maksimal tidak sama dengan N/A maka perhitungan nilai penalti *soft* constraint 1 dilakukan, begitu pula sebaliknya. Pada *method penalti1* terdapat variabel *penalti1* yang digunakan untuk menyimpan nilai penalti *soft constraint* 1.

5. 2. 4. 2. Perhitungan Nilai Penalti *Soft Constraint* 2

Soft constraint 2 bertujuan untuk membatasi terlalu banyaknya hari kerja berurutan. Pelanggaran terhadap *soft constraunt* 2 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yant telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 2 dilakukan dengan membuat sebuah *method penalti2*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal dalam perhitungan nilai penalti *soft constraint* 2 yaitu mengidentifikasi batas maksimal hari kerja. Setiap unit rumah sakit memiliki batas maksimal hari kerja yang berbeda-beda. Jika batas maksimal hari kerja tidak sama dengan N/A maka perhitungan nilai penalti *soft constraint* 2 dilakukan, begitu pula sebaliknya. Pada *method penalti2* terdapat variabel *penalti2* yang digunakan untuk menyimpan nilai penalti *soft constraint* 2.

5. 2. 4. 3. Perhitungan Nilai Penalti *Soft Constraint* 3

Soft constraint 3 bertujuan untuk membatasi terlalu sedikitnya hari kerja berurutan dengan kategori sif yang sama. Pelanggaran terhadap *soft constraint* 3 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 3 dilakukan dengan membuat sebuah *method penalti3*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai penalti *soft constraint* 3 yaitu mengidentifikasi batas minimal hari kerja berurutan pada kategori sif yang sama, yaitu sif pagi, sif siang, dan sif malam. Setiap kategori sif memiliki batas minimal yang berbeda-beda. Jika batas minimal tidak sama dengan N/A maka perhitungan nilai penalti *soft constraint* 3 dilakukan, begitu pula sebaliknya. Pada *method penalti3* terdapat variabel *penalti3* yang digunakan untuk menyimpan nilai penalti *soft constraint* 3.

5. 2. 4. 4. Perhitungan Nilai Penalti *Soft Constraint* 4

Soft constraint 4 bertujuan untuk membatasi terlalu sedikitnya hari kerja berurutan. Pelanggaran terhadap *soft constraint* 4 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 4 dilakukan dengan membuat sebuah *method penalti4*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai penalti *soft constraint* 4 yaitu mengidentifikasi batas minimal hari kerja dari setiap unit rumah sakit. Setiap unit rumah sakit memiliki batas minimal hari kerja yang berbeda-beda. Jika batas minimal hari kerja tidak sama dengan N/A maka perhitungan nilai penalti *soft constraint* 4 dilakukan, begitu pula sebaliknya. Pada *method penalti4* terdapat variabel *penalti4* yang digunakan untuk menyimpan nilai penalti *soft constraint* 4.

5. 2. 4. 5. Perhitungan Nilai Penalti *Soft Constraint* 5

Soft constraint 5 bertujuan untuk membatasi penyimpangan dari batas minimal dan maksimal jumlah sif pada setiap kategorinya. Pelanggaran terhadap *soft constraint* 5 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 5 dilakukan dengan membuat sebuah *method penalti5*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal dari perhitungan nilai penalti *soft constraint* 5 yaitu mengidentifikasi batas minimal dan maksimal dari hari kerja dengan kategori sif yang sama. Setiap unit rumah sakit memiliki batas minimal dan maksimal yang berbeda-beda. Jika batas minimal dan maksimal tidak sama dengan N/A maka perhitungan nilai penalti *soft constraint* 5 dilakukan, begitu pula sebaliknya. Pada *method penalti5* terdapat variabel *min* yang digunakan untuk menyimpan batas minimal dan variabel *max* yang digunakan untuk menyimpan batas maksimal. Kemudian terdapat variabel *count* yang digunakan untuk menghitung jumlah hari kerja berurutan dengan kategori sif yang sama. Ketiga varibel tersebut digunakan untuk menghitung nilai penalti *soft constraint* 5 yang disimpan pada variabel *penalti5*.

5. 2. 4. 6. Perhitungan Nilai Penalti *Soft Constraint* 6

Soft constraint 6 bertujuan untuk membatasi penyimpangan jam kerja perawat. Pelanggaran terhadap *soft constraint* 6 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 6 dilakulan dengan membuat sebuah *method penalti6*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai *soft constraint* 6 yaitu mengidentifikasi nilai batasan dari *soft constraint* 6. Jika nilai batasan tersebut tidak sama dengan N/A maka perhitungan nilai *soft constraint* 6 dilakukan, begitu pula sebaliknya. Pada perhitungan *soft constraint* 6 terdapat variabel

hour yang digunakan untuk menentukan jumlah jam kerja perawat selama penjadwalan. Kemudian terdapat variabel *workingHour* yang digunakan menghitung jam kerja perawat setiap minggunya. Kedua variabel tersebut dihitung selisihnya untuk mendapatkan nilai penalti *soft constraint* 6.

5. 2. 4. 7. Perhitungan Nilai Penalti *Soft Constraint* 7

Soft constraint 7 bertujuan untuk mengelompokan waktu libur perawat. Pelanggaran terhadap *soft constraint* 7 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 7 dilakukan dengan membuat sebuah *method penalti7*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai penalti *soft constraint* 7 yaitu mengidentifikasi nilai batasan dari *soft constraint* 7. Jika nilai batasan *soft constraint* 7 tidak sama dengan N/A maka perhitungan nilai penalti *soft constraint* 7 dilakukan, begitu pula sebaliknya. Pada perhitungan *soft constraint* 7 terdapat variabel *penalti7* yang digunakan untuk menyimpan nilai penalti *soft constraint* 7. Kemudian terdapat variabel *count* yang digunakan untuk menghitung waktu libur yang tidak berkelompok. Nilai *count* tersebut kemudian dikuadratkan dan dijumlahkan untuk mendapatkan nilai *penalti7*. Nilai *penalti7* kemudian diakar kuadrat untuk mendapatkan nilai akhir dari penalti *soft constraint* 7.

5. 2. 4. 8. Perhitungan Nilai Penalti *Soft Constraint* 8

Soft constraint 8 bertujuan untuk memenuhi pola sif yang diinginkan. Pelanggaran terhadap *soft constraint* 8 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft conatraint* 8 dilakukan dengan membuat sebuah *method penalti8*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai

penalti *soft constraint* 8 yaitu mengidentifikasi nilai batasan dari *soft constraint* 8. Nilai batasan tersebut berupa jumlah pola sif yang diinginkan. Setiap unit rumah sakit memiliki nilai batasan *soft constraint* 8 yang berbeda-beda. Jika nilai batasan tersebut tidak sama dengan N/A maka perhitungan penalti *soft constraint* 8 dilakukan, begitu pula sebaliknya. Setelah mengidentifikasi nilai batasan, tahap selanjutnya yaitu menghitung suatu nilai perkalian dari jumlah perawat dengan jumlah hari penjadwalan. Nilai tersebut akan berkurang seiring dengan semakin banyaknya pola sif yang diinginkan. Nilai tersebut kemudian disimpan pada variabel *penalti8* yang merupakan nilai penalti *soft constraint* 8.

5. 2. 4. 9. Perhitungan Nilai Penalti *Soft Constraint* 9

Soft constraint 9 bertujuan untuk membatasi adanya pola sif yang tidak diinginkan. Pelanggaran terhadap *soft conatraint* 9 akan menimbulkan nilai penalti yang dihitung berdasarkan model matematis yang telah dijelaskan pada BAB 2 dan 4. Dalam tahap ini, perhitungan nilai penalti pelanggaran *soft constraint* 9 dilakukan dengan membuat sebuah *method penalti9*. *Method* tersebut merupakan sebuah fungsi yang memiliki nilai pengembalian berupa nilai *double*. Tahap awal perhitungan nilai penalti *soft constraint* 9 yaitu mengidentifikasi nilai batasan dari *soft constraint* 9. Nilai batasan tersebut berupa jumlah pola sif yang tidak diinginkan. Setiap unit rumah sakit memiliki nilai batasan *soft constraint* 9 yang berbeda-beda. Jika nilai batasan *soft constraint* 9 tidak sama dengan N/A maka perhitungan penalti *soft constraint* 9 dilakukan, begitu pula sebaliknya. Setelah mengidentifikasi nilai batasan, tahap selanjutnya yaitu mengitung banyaknya pola sif yang tidak diinginkan pada jadwal kerja perawat yang terbentuk. Perhitungan tersebut akan menghasilkan suatu nilai yang disimpan pada variabel *penalti9*. Nilai pada variabel *penalti9* merupakan nilai penalti *soft constraint* 9.

5. 3. Optimasi Solusi Awal

Tahap akhir dari implementasi penelitian tugas akhir ini yaitu melakukan optimasi solusi awal. Optimasi dilakukan untuk mendapatkan nilai penalti yang terbaik, semakin kecil nilai penalti maka semakin baik solusi yang terbentuk. Solusi pada penelitian tugas akhir ini yaitu berupa jadwal kerja perawat. Optimasi solusi awal dilakukan dengan cara menukar antara satu sif dengan sif lainnya. Penukaran sif tersebut menggunakan beberapa metode penukaran atau sering disebut sebagai *low level heuristic*. Terdapat 3 *low level heuristic* yang digunakan pada penelitian tugas akhir ini, yaitu 2-*exchange*, 3-*exchange*, dan *double* 2-*exchange*. Penukaran sif dilakukan terus menerus selama iterasi dengan menggunakan beberapa algoritma yang telah dijelaskan pada BAB 4. Pada akhir iterasi akan mendapatkan suatu nilai yang merupakan nilai penalti paling optimal.

5. 3. 1. Pembuatan *Low Level Heuristic*

Low level heuristic merupakan sebuah *heuristic* yang digunakan untuk mengubah sebuah solusi menjadi kandidat solusi baru. Solusi yang dimaksud yaitu jadwal kerja perawat. Implementasi *low levek heuristic* dilakukan dengan membuat *method* untuk setiap *low level heuristic*. Terdapat 3 *low level heuristic* yang digunakan, yaitu 2-*exchange*, 3-*exchange*, *double* 2-*exchange*. Sistem kerja 2-*exchange* yaitu memilih 2 perawat secara acak dimana salah satu perawat memiliki jadwal kerja dan perawat lainnya tidak memiliki jadwal kerja. Kemudian sif antara 2 perawat tersebut ditukar pada hari yang sama. Sistem kerja 3-*exchange* yaitu memilih 3 perawat secara acak dimana ketiga perawat tersebut memiliki jadwal kerja. Kemudian sif antara 3 perawat tersebut ditukar pada hari yang sama. Sementara itu sistem kerja *double* 2-*exchange* yaitu memilih 2 perawat dan 2 hari secara acak dimana pada masing-masing perawat memiliki 1 waktu libur. Kemudian sif antara 2 perawat tersebut ditukar. Penukaran sif tersebut akan menghasilkan sebuah kandidat solusi baru. Jika kandidat solusi baru tersebut merupakan solusi

feasible maka solusi tersebut diterima, begitu pula sebaliknya. Penukaran sif akan berlangsung terus menerus sesuai dengan jumlah iterasi yang telah ditentukan.

5. 3. 2. Implementasi Algoritma *RL-SAR*

Tahap awal implementasi algoritma *RL-SAR* yaitu menetapkan beberapa parameter yang digunakan. Parameter-parameter tersebut diantaranya yaitu metode pemilihan *heuristic*, *utility update*, suhu awal, *cooling rate*, *reheating frequency*, dan *reheating limit*. Metode pemilihan *heuristic* digunakan untuk memilih *low level heuristic* mana yang akan digunakan untuk menghasilkan kandidat solusi baru. *Utility update* digunakan untuk menentukan mekanisme pemberian *reward/punishment* pada algortima *reinforcement learning*. Suhu awal merupakan sebuah nilai yang digunakan untuk menentukan probabilitas diterimanya solusi yang lebih buruk pada algoritma *simulated annealing with reheating*. Suhu tersebut akan berkurang seiring dengan bertambahnya iterasi. *Cooling rate* merupakan sebuah nilai yang digunakan untuk mengurangi suhu awal secara bertahap selama iterasi. *Reheating frequency* merupakan sebuah nilai yang menentukan frekuensi peningkatan suhu pada saat algoritma tidak menemukan solusi yang lebih baik. *Reheating limit* merupakan sebuah nilai yang digunakan untuk membatasi jumlah pemanasan ulang yang dapat dilakukan.

Pada algoritma *RL-SAR*, *low level heuristic* dipilih berdasarkan konsep *reinforcement learning*. *Low level heuristic* yang menghasilkan kandidat solusi dengan nilai penalti yang lebih baik akan mendapatkan *reward*, begitu pula sebaliknya. Nilai *reward* atau *punishment* yang didapat akan diakumulasi dan disimpan pada sebuah *array* 1 dimensi. Kandidat solusi dengan nilai penalti yang lebih kecil akan diterima dan kandidat solusi dengan nilai penalti yang lebih besar akan diterima dengan probabilitas tertentu. Nilai probabilitas tersebut dipengarungi oleh nilai suhu pada saat itu. Nilai suhu akan berkurang seiring dengan bertambahnya iterasi. Ketika kandidat solusi cenderung menghasilkan solusi yang lebih buruk secara terus menerus,

maka akan dilakukan pemanasan ulang suhu. Hal ini dilakukan agar algoritma dapat menemukan kandidat solusi yang menghasilkan solusi yang lebih baik.

5. 4. Ringkasan Implementasi

Tahap implementasi dilakukan secara komputasi. Tahap ini terdiri dari pembacaan dan penyimpanan data masukan, pembuatan solusi awal, dan optimasi solusi awal. Pada tahap pembuatan solusi awal terdiri dari proses pembuatan matriks solusi awal, pengecekan *hard constraint*, penyusunan solusi awal, dan perhitungan nilai penalti. Pada tahap optimasi solusi awal terdiri dari proses pembuatan *low-level heuristic* dan proses implementasi algoritma *RL-SAR*. Selanjutnya, hasil dari proses implementasi akan dijelaskan pada bab 6.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil dan analisis terhadap proses implementasi yang telah dilakukan pada bab 5. Pada bab ini terdapat beberapa penjelasan yang meliputi lingkungan uji coba, hasil solusi awal, hasil uji coba parameter algoritma dan performa dari algoritma.

6. 1. Lingkungan Uji Coba

Pada penelitian tugas akhir ini, uji coba otomasi dan optimasi dilakukan dengan menggunakan bantuan Intelij dan bahasa pemrograman Java. Perangkat keras yang digunakan berupa sebuah laptop AMD FX-7500 Radeon R7 dengan kapasitas RAM sebesar 8 Gb dan kapasitas SSD sebesar 240 Gb. Perangkat lunak yang digunakan berupa aplikasi komputasi Intellij IDEA dan media penyimpanan data Microsoft Excel yang beroperasi pada sistem operasi Windows 10. Pada Tabel 6.1 menampilkan perangkat keras dan perangkat lunak yang digunakan dalam penelitian tugas akhir ini.

Tabel 6.1 Perangkat Keras dan Perangkat Lunak yang Digunakan

Perangkat keras	
Perangkat	Laptop
Processor	AMD FX-7500 Radeon R7
RAM	8 Gb
Hard disk	SSD 240 Gb
Perangkat lunak	
Sistem operasi	Windows 10
Aplikasi pengembangan	Intellij IDEA
Media penyimpanan data	Microsoft Excel 365

6. 2. Hasil Solusi Awal

Penyusunan solusi awal dilakukan dengan cara menyusun jadwal untuk hari sabtu dan minggu terlebih dahulu kemudian

menyusun jadwal untuk hari senin sampai jumat. Penyusunan solusi awal dilakukan dengan menerapkan skenario yang telah dirancang pada BAB 4. Terdapat 2 skenario yang diterapkan, yaitu skenario pengecakan *hard constraint* 3 dan skenario pengecekan *hard constraint* 3 dan 6. Pada skenario pengecekan *hard constraint* 3 menghasilkan solusi awal yang *feasible* untuk 2 unit rumah sakit. Sementara itu, pada skenario pengecekan *hard constraint* 3 dan 6 menghasilkan solusi awal yang *feasible* untuk 3 unit rumah sakit. Berdasarkan hasil tersebut, maka skenario pengecekan *hard constraint* 3 dan 6 digunakan untuk penyusunan solusi awal. Ringkasan penyusunan solusi awal untuk seluruh unit rumah sakit dapat dilihat pada Tabel 6.2 berikut.

Tabel 6.2 Ringkasan Penyusunan Solusi Awal

Unit Rumah Sakit	Status	Keterangan
Optur 1	Tidak *feasible*	Melanggar *hard constraint* 3 dan 6
Optur 2	Tidak *feasible*	Melanggar *hard constraint* 3 dan 6
Optur 3	Tidak *feasible*	Melanggar *hard constraint* 3 dan 6
Optur 4	*Feasible*	Memenuhi seluruh *hard constraint*
Optur 5	*Feasible*	Memenuhi seluruh *hard constraint*
Optur 6	Tidak *feasible*	Melanggar *hard constraint* 3 dan 6
Optur 7	*Feasible*	Memenuhi seluruh *hard constraint*

Dari 7 unit rumah sakit yang ada, penjadwalan otomatis berhasil dilakukan pada 3 unit rumah sakit yaitu Optur 4, Optur 5 dan Optur 7. Keberhasilan penjadwalan otomatis yang dimaksud yaitu jadwal kerja yang terbentuk memenuhi seluruh *hard constraint* yang ada. Ketiga jadwal kerja tersebut kemudian akan dilakukan optimasi untuk mendapatkan nilai penalti yang paling kecil. Salah satu jadwal kerja yang berhasil terbentuk dapat dilihat pada Lampiran A-2.

Terdapat 4 unit rumah sakit yang tidak berhasil dilakukan penjadwalan otomatis karena masih melanggar beberapa *hard constraint*. Mayoritas *hard constraint* yang masih dilanggar

yaitu *hard constraint* 3 dan *hard constraint* 6. *Hard constraint* 3 bertujuan untuk memastikan bahwa jam kerja perawat pada jadwal yang terbentuk tidak menyimpang terlalu jauh dari kontrak jam kerja perawat. Pada proses implementasi, keempat jadwal kerja unit rumah sakit yang tidak berhasil dilakukan penjadwalan otomatis masih memiliki penyimpangan terlalu jauh dari kontrak jam kerja perawat. Sementara itu *hard constraint* 6 bertujuan untuk memastikan bahwa setiap perawat memiliki minimal waktu libur setiap minggunya. Pada proses implementasi, keempat unit rumah sakit masih terdapat perawat yang memiliki waktu libur kurang dari batas yang telah ditentukan. *Hard constraint* 3 dan 6 merupakan *hard constraint* yang saling berhubungan. Tidak terpenuhinya *hard constraint* 3 dan 6 dapat disebabkan karena jam kerja yang harus dijadwalkan terlalu besar pada keempat unit rumah sakit.

6. 3. Hasil Uji Coba Parameter Algoritma *RL-SAR*

Uji coba parameter algoritma *RL-SAR* dilakukan pada ketiga jadwal kerja unit rumah sakit yang memenuhi seluruh *hard constraint*. Jadwal kerja tersebut yaitu jadwal kerja unit rumah sakit pada Optur 4, Optur 5 dan Optur 7. Masing-masing skenario akan diuji coba pada ketiga jadwal kerja dengan percobaan sebanyak 10 kali untuk setiap unit rumah sakit. Hasil yang ditampilkan merupakan rata-rata nilai penalti pada setiap skenario. Seluruh hasil uji coba parameter dapat dilihat pada Lampiran B-1 hingga Lampiran B-4.

6. 3. 1. Hasil Uji Coba Parameter *Heuristic Selection*

Uji coba parameter *heuristic selection* dilakukan dengan membandingkan metode pemilihan *Low Level Heurictic*. Terdapat 3 metode *heuristic selection* yang digunakan, yaitu *greedy*, *epsilon greedy*, dan *epsilon decay greedy*. Metode *greedy* merupakan sebuah metode yang akan memilih *Low Level Heuristic* dengan nilai skor paling tinggi. Sementara itu metode *epsilon greedy* merupakan sebuah metode yang memilih *Low Level Heuristic* dengan nilai probabilitas tertentu. Jika nilai

probabilitas tersebut lebih kecil dari nilai *epsilon*, maka pemilihan *Low Level Heuristic* dilakukan secara acak. Namun, jika nilai probabilitas tersebut lebih besar dari nilai *epsilon*, maka pemilihan *Low Level Heuristic* dilakukan dengan memilih *Low Level Heuristic* dengan nilai skor paling tinggi. Metode *epsilon decay greedy* memiliki konsep yang sama dengan metode *epsilon greedy*. Perbedaan antara metode *epsilon decay greedy* dengan *epsilon greedy* terletak pada nilai *epsilon* yang digunakan. Nilai *epsilon* pada metode *epsilon greedy* bersifat tetap untuk setiap iterasi, sedangkan nilai *epsilon* pada metode *epsilon decay greedy* akan berkurang seiring dengan bertambahnya iterasi. Metode *heuristic selection* yang menghasilkan nilai penalti paling kecil akan dipilih dan digunakan sebagai parameter pada algoritma *RL-SAR*. Hasil uji coba parameter *heuristic selection* dapat dilihat pada Tabel 6.3 berikut.

Tabel 6.3 Hasil Uji Coba Parameter *Heuristic Selection*

	Optur 4	**Optur 5**	**Optur 7**
Penalti awal	628.420	241.984	720.167
Greedy			
Penalti akhir	155.196	49.677	668.000
Perbaikan (%)	75.304%	79.471%	7.244%
Runing Time (*s*)	3056	351	233
Epsilon Greedy			
Penalti akhir	151.804	50.138	670.734
Perbaikan (%)	75.843%	79.281%	6.864%
Runing Time (*s*)	3265	340	231
Epsilon Decay Greedy			
Penalti akhir	150.851	47.639	667.658
Perbaikan (%)	75.995%	80.313%	7.291%
Runing Time (*s*)	3078	359	238

Hasil uji coba parameter *heuristic selection* menunjukan bahwa nilai parameter *epsilon decay greedy* menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 4, unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Berdasarkan hasil tersebut, maka nilai parameter *epsilon decay greedy* akan digunakan sebagai parameter dalam proses pencarian solusi akhir menggunakan algoritma *RL-SAR*.

6. 3. 2. Hasil Uji Coba Parameter *Utility Update*

Uji coba parameter *utility update* dilakukan dengan membandingkan metode pemberian *reward/punishment* pada setiap *Low Level Heuristic*. Terdapat 3 metode *utility update* yang digunakan, yaitu *sum, average* dan *discount factor*. Metode *sum* merupakan sebuah metode pemberian *reward/punishment* dengan cara menjumlahkan nilai *reward/punishment* yang didapat. Sementara itu metode *average* merupakan sebuah metode pemberian *reward/punishment* dengan cara menghitung rata-rata dari nilai *reward/punishment* yang didapat. Metode *discount factor* memiliki konsep yang hampir sama dengan metode *sum*. Perbedaan antara metode *sum* dengan metode *discount factor* terletak pada nilai *reward/punishment* yang diberikan. Nilai *reward/punishment* pada metode *sum* bersifat tetap untuk setiap iterasi, sedangkan nilai *reward/punishment* pada metode *discount factor* bersifat dinamis yang dipengaruhi oleh suatu nilai γ (*discount factor*). Metode *utility update* yang menghasilkan nilai penalti paling kecil akan dipilih dan digunakan sebagai parameter pada algoritma *RL-SAR*. Hasil uji coba parameter *utility update* dapat dilihat pada Tabel 6.4 berikut.

Tabel 6.4 Hasil Uji Coba Parameter *Utility Update*

	Optur 4	Optur 5	Optur 7
Penalti awal	628.420	241.984	720.167
Sum			
Penalti akhir	150.851	47.639	667.658
Perbaikan (%)	75.995%	80.313%	7.291%
Runing Time (*s*)	3078	359	238
Average			
Penalti akhir	147.381	49.646	668.996
Perbaikan (%)	76.547%	79.484%	7.105%
Runing Time (*s*)	2981	358	237
Discount Factor			
Penalti akhir	162.438	88.805	685.104
Perbaikan (%)	74.151%	63.301%	4.869%
Runing Time (*s*)	2982	354	206

Hasil uji coba parameter *utility update* menunjukan bahwa nilai parameter *average* menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 4. Sementara itu nilai parameter *sum* menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Berdasarkan hasil tersebut, nilai parameter *sum* berhasil unggul pada 2 unit rumah sakit, sehingga nilai parameter *sum* akan digunakan sebagai parameter dalam proses pencarian solusi akhir menggunakan algoritma *RL-SAR.*

6. 3. 3. Hasil Uji Coba Parameter *Cooling Rate*

Uji coba parameter *cooling rate* dilakukan dengan membandingkan nilai *cooling rate* pada algoritma *Simulated Annealing with Reheating*. *Cooling rate* tersebut digunakan untuk menurunkan suhu pada setiap iterasi. Semakin kecil suhu maka algoritma akan berpindah dari proses eksplorasi menuju proses eksploitasi. Terdapat 3 nilai *cooling rate* yang digunakan, yaitu 0.5, 0.85 dan 0.99995. Nilai *cooling rate* yang

menghasilkan nilai penalti paling kecil akan dipilih dan digunakan sebagai parameter pada algoritma *RL-SAR*. Hasil uji coba parameter *cooling rate* dapat dilihat pada Tabel 6.5 berikut.

Tabel 6.5 Hasil Uji Coba Parameter *Cooling Rate*

	Optur 4	**Optur 5**	**Optur 7**
Penalti awal	628.420	241.984	720.167
***Cooling Rate* = 0.5**			
Penalti akhir	135.199	74.691	681.748
Perbaikan (%)	78.486%	69.134%	5.335%
Runing Time (*s*)	2804	394	245
***Cooling Rate* = 0.85**			
Penalti akhir	137.409	72.132	681.125
Perbaikan (%)	78.134%	70.192%	5.421%
Runing Time (*s*)	2850	387	239
***Cooling Rate* = 0.99995**			
Penalti akhir	150.851	47.639	667.658
Perbaikan (%)	75.995%	80.313%	7.291%
Runing Time (*s*)	3078	359	238

Hasil uji coba parameter *cooling rate* menunjukan bahwa nilai parameter *cooling rate* 0.5 menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 4. Sementara itu nilai parameter *cooling rate* 0.99995 menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Berdasarkan hasil tersebut, nilai parameter *cooling rate* 0.99995 berhasil unggul pada 2 unit rumah sakit, sehingga nilai parameter *cooling rate* 0.99995 akan digunakan sebagai parameter dalam proses pencarian solusi akhir menggunakan algoritma *RL-SAR*.

6. 3. 4. Hasil Uji Coba Parameter *Reheating Frequency*

Uji coba parameter *reheating frequency* dilakukan dengan membandingkan nilai *reheating frequency* pada algoritma *Simulated Annealing with Reheating*. Nilai *reheating frequency* digunakan untuk menentukan kapan algoritma harus melakukan pemanasan ulang suhu. Hal tersebut dilakukan agar algoritma tidak terjebak pada *local optima*. Terdapat 3 nilai *reheating frequency* yang digunakan, yaitu 10000, 50000 dan 100000. Nilai *reheating frequency* yang menghasilkan nilai penalti paling kecil akan dipilih dan digunakan sebagai parameter pada algoritma *RL-SAR.* Hasil uji coba parameter *reheating frequency* dapat dilihat pada Tabel 6.6 berikut.

Tabel 6.6 Hasil Uji Coba Parameter *Reheating Frequency*

	Optur 4	**Optur 5**	**Optur 7**
Penalti awal	628.420	241.984	720.167
***Reheating Frequency* = 10000**			
Penalti akhir	139.051	51.031	669.360
Perbaikan (%)	77.873%	78.912%	7.055%
Runing Time (*s*)	2842	359	235
***Reheating Frequency* = 50000**			
Penalti akhir	150.851	47.639	667.658
Perbaikan (%)	75.995%	80.313%	7.291%
Runing Time (*s*)	3078	359	238
***Reheating Frequency* = 100000**			
Penalti akhir	157.280	50.233	669.687
Perbaikan (%)	74.972%	79.241%	7.010%
Runing Time (*s*)	3006	348	232

Hasil uji coba parameter *reheating frequecy* menunjukan bahwa nilai parameter *reheating frequency* 10000 menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 4. Sementara itu nilai parameter *reheating frequency* 50000 menghasilkan nilai penalti paling kecil pada unit rumah sakit Optur 5 dan unit

rumah sakit Optur 7. Berdasarkan hasil tersebut, nilai parameter *reheating frequency* 50000 berhasil unggul pada 2 unit rumah sakit, sehingga nilai parameter *reheating frequency* 50000 akan digunakan sebagai parameter dalam proses pencarian solusi akhir menggunakan algoritma *RL-SAR*.

6. 3. 5. Hasil Solusi Akhir

Setelah melakukan seluruh skenario uji coba parameter dan menemukan nilai parameter yang tepat, maka tahap selanjutnya yaitu menerapkan nilai parameter tersebut untuk melihat performa dari algoritma *RL-SAR*. Nilai parameter yang digunakan pada algoritma *RL-SAR* dapat dilihat pada Tabel 6.7 berikut.

Tabel 6.7 Nilai Parameter Algoritma *RL-SAR*

HEURISTIC SELECTION	*UTILIITY UPDATE*	*COOLING RATE*	*REHEATING FREQUENCY*
Epsilon Decay Greedy	Sum	0.99995	50000

Nilai parameter tersebut digunakan pada algoritma *RL*-SAR dengan percobaan sebanyak 10 kali terhadap jadwal kerja unit rumah sakit Optur 4, Optur 5 dan Optur 7. Salah satu hasil solusi akhir dapat dilihat pada Lampiran C-1. Hasil nilai penalti dari percobaan yang telah dilakukan dapat dilihat pada Tabel 6.8 berikut.

Tabel 6.8 Hasil Solusi Akhir Algoritma *RL-SAR*

	Optur 4	**Optur 5**	**Optur 7**
Penalti awal	628.420	241.984	720.167
Penalti akhir	146.835	44.715	667.291
Perbaikan (%)	76.634%	81.522%	7.342%
***Runing Time* (*s*)**	3076	352	231

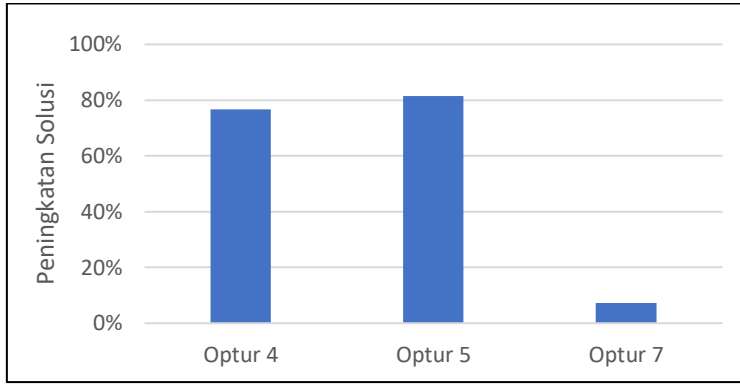
Nilai penalti akhir pada Tabel 6.8 akan dibandingkan dengan nilai penalti awal, nilai penalti algoritma *SR-HC*, *RL-HC*, dan *RL-SA* untuk mengetahui performa dari algortima *RL-SAR*.

6. 4. Performa Algortima *RL-SAR*

Setelah melakukan uji coba parameter, maka tahap selanjutnya yaitu melakukan analisa terhadap performa algoritma *RL-SAR*. Performa algoritma *RL-SAR* dapat dilihat dengan membandingkan antara nilai penalti solusi awal dengan nilai penalti algoritma *RL-SAR*. Hal tersebut dilakukan untuk mengetahui seberapa besar peningkatan yang dihasilkan oleh algoritma *RL-SAR*. Selain itu, performa algoritma *RL-SAR* juga dapat dilihat dengan membandingkannya dengan algoritma lain. Pada tugas akhir ini, algoritma *RL-SAR* akan dibandingkan dengan algoritma *Simple Random – Hill Climbing* (*SR-HC*), algoritma *Reinforcement Learning – Hill Climbing* (*RL-HC*) dan algoritma *Reinforcement Learning – Simulated Annealing* (*RL-SA*). Masing-masing algoritma diuji sebanyak 10 kali untuk setiap unit rumah sakit. Seluruh hasil percobaan performa algoritma dapat dilihat pada Lampiran C-2 hingga Lampiran C-6.

6. 4. 1. Perbandingan dengan Solusi Awal

Perbandingan antara nilai penalti solusi awal dengan nilai penalti algoritma *RL-SAR* dilakukan untuk mengetahui peningkatan yang dihasilkan oleh algoritma *RL-SAR*. Semakin besar peningkatan yang dihasilkan, maka semakin baik performa dari algoritma *RL-SAR*. Dalam permasalahan ini, peningkatan yang dimaksud yaitu nilai penalti yang semakin kecil. Peningkatan yang dihasilkan oleh algoritma *RL-SAR* dapat dilihat pada Gambar 6.1 berikut.



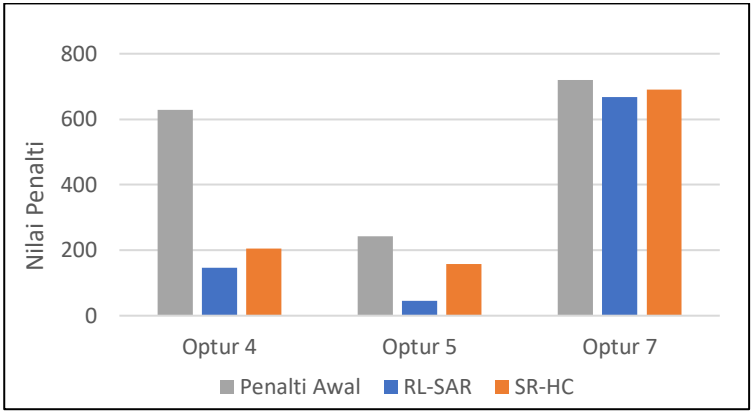
Gambar 6.1 Peningkatan Solusi oleh Algoritma *RL-SAR*

Pada Gambar 6.1 dapat dilihat bahwa algoritma *RL-SAR* menghasilkan peningkatan solusi yang cukup signifikan. Hal tersebut terlihat dari peningkatan pada unit rumah sakit Optur 4 dan unit rumah sakit Optur 5. Pada unit rumah sakit Optur 4 terjadi peningkatan solusi hingga di atas 70%. Kemudian pada unit rumah sakit Optur 5 terjadi peningkatan solusi hingga di atas 80%. Sementara itu pada unit rumah sakit Optur 7 hanya terjadi peningkatan solusi di bawah 10%. Meskipun peningkatan solusi pada unit rumah sakit Optur 7 tidak cukup signifikan, namun algoritma *RL-SAR* mampu memberikan peningkatan yang signifikan untuk unit rumah sakit Optur 4 dan unit rumah sakit Optur 5. Hal tersebut menunjukan bahwa algoritma *RL-SAR* memiliki performa yang cukup baik dalam menyelesaikan masalah penjadwalan perawat rumah sakit di Norwegia.

6. 4. 2. Perbandingan dengan Algoritma *Simple Random - Hill Climbing*

Perbandingan performa algoritma *RL-SAR* dengan *SR-HC* dilakukan dengan melihat nilai penalti yang dihasilkan oleh kedua algoritma tersebut. Performa algoritma yang lebih baik ditunjukan dengan semakin kecil nilai penalti yang dihasilkan. Parameter yang digunakan pada algoritma *SR-HC* telah

disesuaikan sedemikian rupa agar algoritma tersebut mampu mencari solusi dengan jumlah iterasi yang sama. Performa dari algoritma *RL-SAR* dan *SR-HC* dapat dilihat pada Gambar 6.2 berikut.

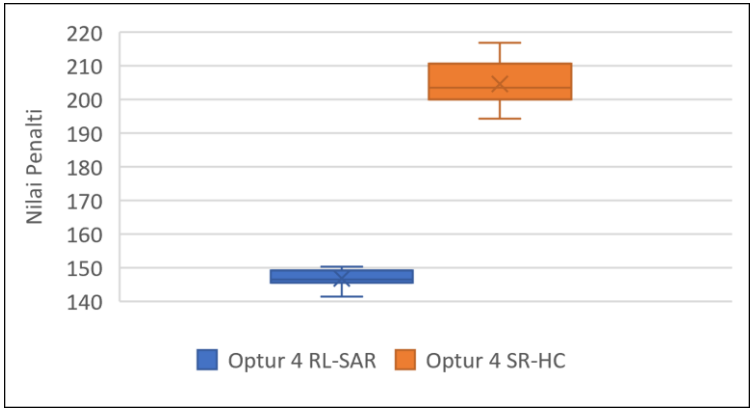


Gambar 6.2 Perbandingan Peningkatan Solusi Algoritma *RL-SAR* dan *SR-HC*

Pada Gambar 6.2 menampilkan perbandingan rata-rata nilai penalti dari algoritma *RL-SAR* dan algoritma *SR-HC* pada unit rumah sakit Optur 4, unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Pada unit rumah sakit Optur 4, masing-masing algoritma *RL-SAR* dan *SR-HC* menghasilkan nilai penalti sebesar 146.835 dan 204.681 dari nilai penalti awal sebesar 628.420. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 77%, sedangkan algoritma *SR-HC* menghasilkan peningkatan solusi sebesar 67%. Pada unit rumah sakit Optur 5, masing-masing algoritma *RL-SAR* dan *SR-HC* menghasilkan nilai penalti sebesar 44.715 dan 157.284 dari nilai penalti awal sebesar 241.985. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 82%, sedangkan algoritma *SR-HC* hanya menghasilkan peningkatan solusi sebesar 35%. Pada unit rumah sakit Optur 7, masing-masing algoritma *RL-SAR* dan *SR-HC* menghasilkan nilai penalti sebesar 667.291 dan 690.575 dari nilai penalti awal sebesar

720.167. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan penigkatan solusi sebesar 7.3%, sedangkan algoritma *SR-HC* menghasilkan peningkatan solusi sebesar 4%.

Selanjutnya untuk melihat performa algoritma secara lebih rinci akan dilakukan analisa terhadap setiap unit rumah sakit. Terdapat 2 jenis analisa yang dilakukan, yaitu analisa terhadap persebaran nilai penalti dan analisa terhadap kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *SR-HC* pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.3 berikut.

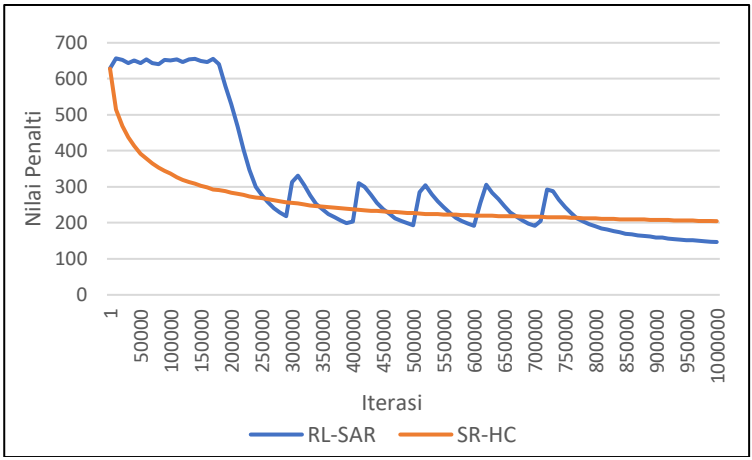


Gambar 6.3 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *SR-HC* pada Optur 4

Pada Gambar 6.3 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal tersebut dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *SR-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *SR-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti yang lebih baik

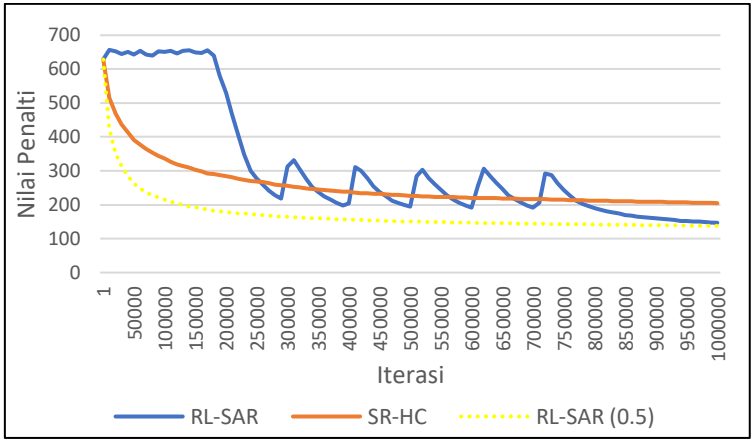
pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.4 berikut.



Gambar 6.4 Perbandingan Kemampuan Algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.4 dapat dilihat bahwa garis algoritma *SR-HC* memiliki kecenderungan menurun. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Karakteristik itu menyebabkan algoritma *SR-HC* hanya melakukan ekploitasi dan terjebak pada *local optima*. Kondisi tersebut dapat dilihat dari penurunan nilai penalti yang tidak signifikan pada bagian tengah hingga akhir iterasi. Kemudian algoritma *RL-SAR* memiliki garis yang bersifat fluktuatif. Hal tersebut dikarenakan algoritma *RL-SAR* memiliki karakteristik untuk menerima nilai penalti yang lebih buruk. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*.

Pada tahap uji coba parameter menunjukan bahwa terdapat parameter yang dapat menghasilkan nilai penalti yang lebih kecil, yaitu *cooling rate* 0.5. Sehingga dilakukan percobaan dengan mengubah parameter *cooling rate* untuk mengetahui kemampuan algoritma *RL-SAR* dalam menemukan nilai penalti yang lebih baik. Hasil percobaan tersebut kemudian dibandingkan dengan percobaan yang telah dilakukan sebelumnya. Perbandingan kemampuan algoritma *RL-SAR*, S*R-HC* dan *RL-SAR* dengan *cooling rate* 0.5 dapat dilihat pada Gambar 6.5 berikut.

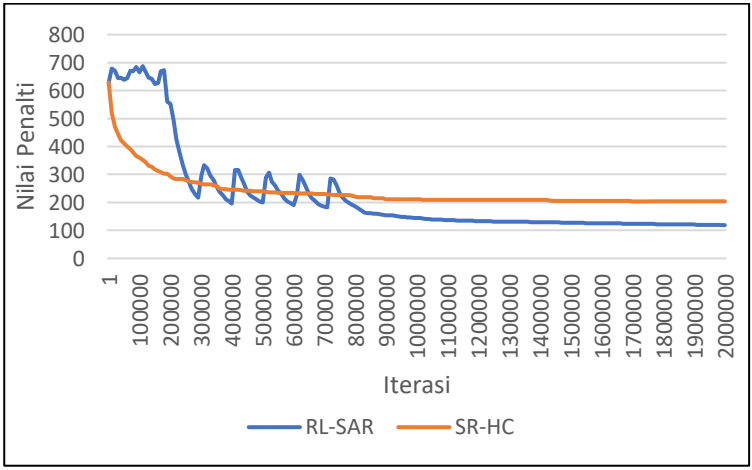


Gambar 6.5 Perbandingan Kemampuan Algoritma *RL-SAR*, S*R-HC* dan *RL-SAR (cooling rate 0.5)* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.5 dapat dilihat bahwa garis putus-putus kuning merupakan garis algoritma *RL-SAR* dengan *cooling rate 0.5*. Garis tersebut menunjukan bahwa algoritma *RL-SAR (cooling rate* 0.5*)* mampu menemukan nilai penalti yang lebih baik daripada algortima S*R-HC*. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih baik, namun parameter tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi dan terjebak pada *local optima*. Hal tersebut dapat dilihat dari garis yang terbentuk

bersifat menurun pada awal iterasi kemudian melandai hingga akhir iterasi.

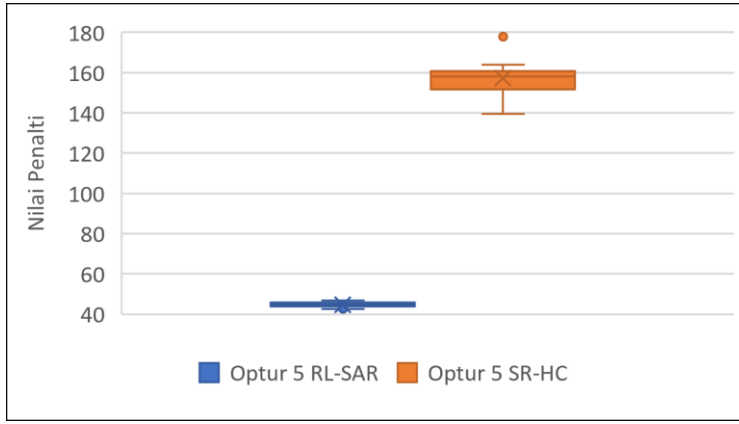
Selanjutnya dilakukan percobaan dengan menambah iterasi menjadi 2000000 iterasi. Percobaan ini hanya dilakukan pada unit rumah sakit Optur 4 untuk melihat kemampuan algoritma *RL-SAR* secara lebih jauh. Hasil percobaan penambahan iterasi dapat dilihat pada Gambar 6.6 berikut.



Gambar 6.6 Perbandingan Kemampuan Algoritma *RL-SAR* dan *SR-HC* dalam Menemukan Nilai Penalti Selama 2000000 Iterasi pada Unit Rumah Sakit Optur 4

Berdasarkan Gambar 6.6 algoritma *RL-SAR* dapat menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi. Hal tersebut menunjukan bahwa pada percobaan 1000000 iterasi algoritma mampu melakukan eksplorasi dan eksploitasi dari awal hingga akhir iterasi. Kemudian jika iterasi ditambah menjadi 2000000 iterasi, maka penambahan iterasi tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi hingga akhir iterasi.

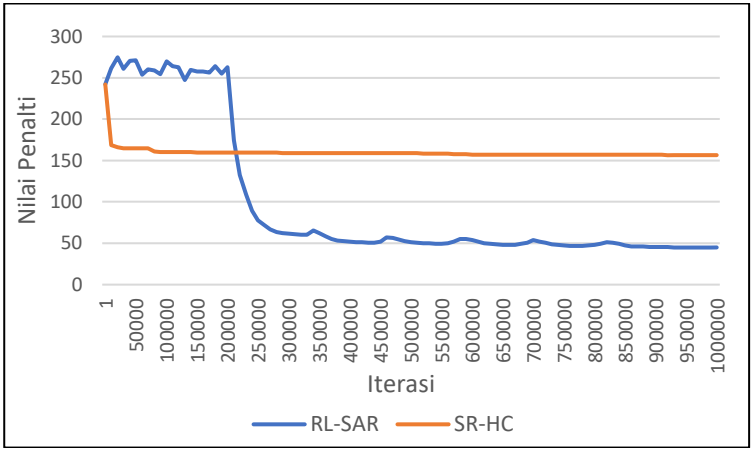
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *SR-HC* pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.7 berikut.



Gambar 6.7 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *SR-HC* pada Optur 5

Pada Gambar 6.7 dapat dilihat bahwa algoritma *SR-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *SR-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan. Algoritma *SR-HC* juga memiliki nilai penalti yang jauh dari nilai maksimal penalti.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.8 berikut.

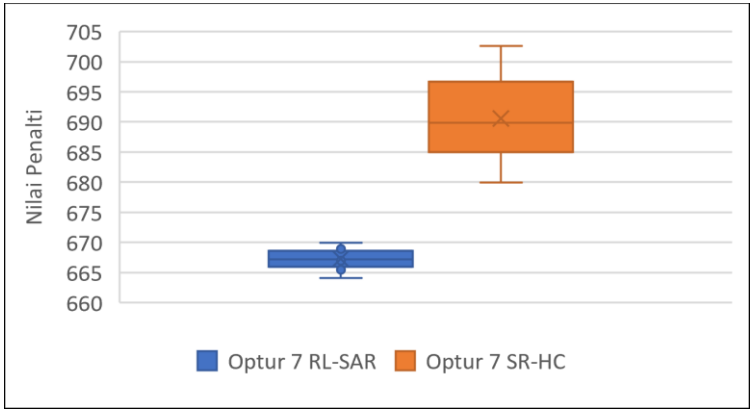


Gambar 6.8 Perbandingan Kemampuan Algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti pada Unit Rumah Sakit Optur 5

Pada Gambar 6.8 dapat dilihat bahwa garis algoritma *SR-HC* mengalami penurunan secara signifikan pada awal iterasi kemudian cenderung melandai hingga akhir iterasi. Pada awal iterasi, algoritma *SR-HC* mampu menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis menurun yang signifikan. Namun, setelah itu algoritma *SR-HC* tidak lagi menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis yang melandai. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Karakteristik itu menyebabkan algoritma *SR-HC* hanya melakukan eksploitasi dan terjebak pada *local optima*. Kemudian algortima *RL-SAR* memiliki garis yang bersifat fluktuatif pada awal iterasi dan cenderung menurun hingga akhir iterasi. Pada awal iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus.

Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma RL-SAR mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma RL-SAR memiliki kemungkinan untuk tidak terjebak pada *local optima*.

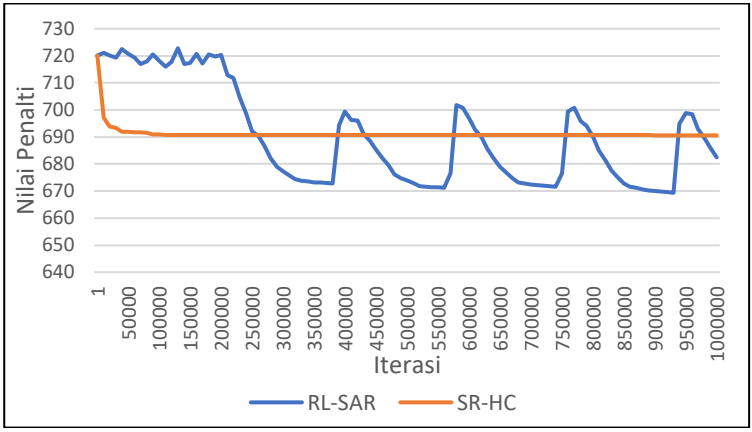
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *SR-HC* pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.9 berikut.



Gambar 6.9 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *SR-HC* pada Optur 7

Pada Gambar 6.9 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal ini dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *SR-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *SR-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.10 berikut.



Gambar 6.10 Perbandingan Kemampuan Algoritma *RL-SAR* dan *SR-HC* dalam menemukan nilai penalti pada Unit Rumah Sakit Optur 7

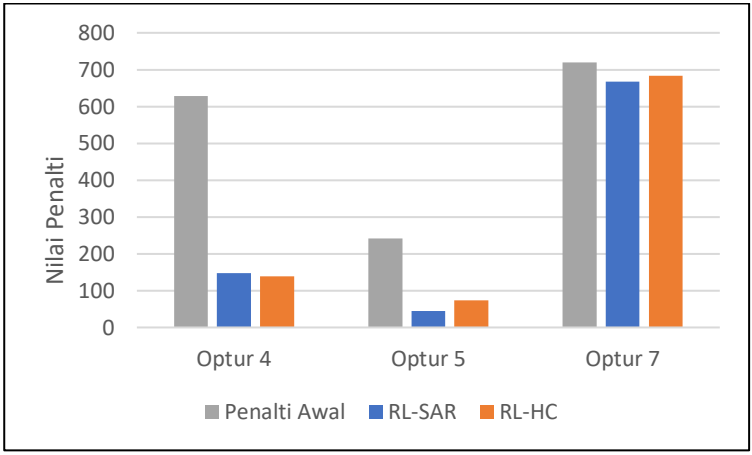
Pada Gambar 6.10 dapat dilihat bahwa garis algoritma *SR-HC* mengalami penurunan yang signifikan pada awal iterasi kemudian cenderung melandai hingga akhir iterasi. Pada awal iterasi, algoritma *SR-HC* mampu menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis menurun yang signifikan. Namun, setelah itu algoritma *SR-HC* tidak lagi menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis yang melandai. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Karakteristik itu menyebabkan algoritma *SR-HC* hanya melakukan eksploitasi dan terjebak pada *local optima*. Kemudian algoritma *RL-SAR* memiliki garis yang bersifat fluktiatif hingga akhir iterasi. Hal tersebut menandakan bahwa algortima mampu menemukan nilai penalti yang bervariasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif

menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma RL-SAR memiliki kemungkinan untuk tidak terjebak pada local optima. Bahkan hingga akhir iterasi algoritma *RL*-SAR masih melakukan eksplorasi yang memungkinkan algoritma untuk menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi.

Berdasarkan analisa yang telah dilakukan, algoritma *RL-SAR* memiliki pesebaran nilai penalti yang lebih konstan daripada algoritma *SR-HC*. Selain itu algoritma *RL-SAR* juga memiliki kemampuan yang lebih baik daripada algoritma *SR-HC* dalam hal menemukan nilai penalti yang lebih baik. Hal tersebut dapat dilihat dari penurunan nilai penalti yang dihasilkan oleh kedua algoritma. Algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih kecil daripada algoritma *SR-HC* pada ketiga unit rumah sakit yang diuji coba. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* dapat menemukan nilai penalti yang lebih baik, namun dengan *cooling rate* 0.5 membuat algoritma *RL-SAR* cenderung melakukan eksploitasi daripada eksplorasi.

6. 4. 3. Perbandingan dengan Algoritma *Reinforcement Learning – Hill Climbing*

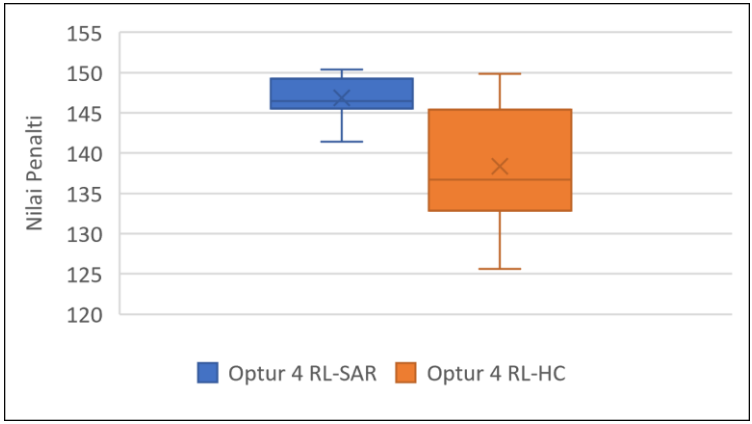
Perbandingan performa algoritma *RL-SAR* dengan *RL-HC* dilakukan dengan melihat nilai penalti yang dihasilkan oleh kedua algoritma tersebut. Performa algoritma yang lebih baik ditunjukan dengan semakin kecil nilai penalti yang dihasilkan. Parameter yang digunakan pada algoritma *RL-HC* telah disesuaikan sedemikian rupa agar algoritma tersebut mampu mencari solusi dengan jumlah iterasi yang sama. Performa dari algoritma *RL-SAR* dan *RL-HC* dapat dilihat pada Gambar 6.11 berikut.



Gambar 6.11 Perbandingan Peningkatan Solusi Algoritma *RL-SAR* dan *RL-HC*

Pada Gambar 6.11 menampilkan perbandingan rata-rata nilai penalti dari algoritma *RL-SAR* dan algoritma *RL-HC* pada unit rumah sakit Optur 4, unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Pada unit rumah sakit Optur 4, masing-masing algoritma *RL-SAR* dan algoritma *RL-HC* menghasilkan nilai penalti sebesar 146.835 dan 138.331 dari nilai penalti awal sebesar 628.420. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 77%, sedangkan algoritma *RL-HC* menghasilkan peningkatan solusi sebesar 78%. Pada unit rumah sakit Optur 5, masing-masing algoritma *RL-SAR* dan *RL-HC* menghasilkan nilai penalti sebesar 44.715 dan 76.674 dari nilai penalti awal sebesar 241.985. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 82%, sedangkan algoritma *RL-HC* menghasilkan peningkatan solusi sebesar 70%. Pada unit rumah sakit Optur 7, masing-masing algoritma *RL-SAR* dan *RL-HC* menghasilkan nilai penalti sebesar 667.291 dan 683.669 dari nilai penalti awal sebesar 720.167. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 7.3%, sedangkan algoritma *RL-HC* menghasilkan peningkatan solusi sebesar 5%.

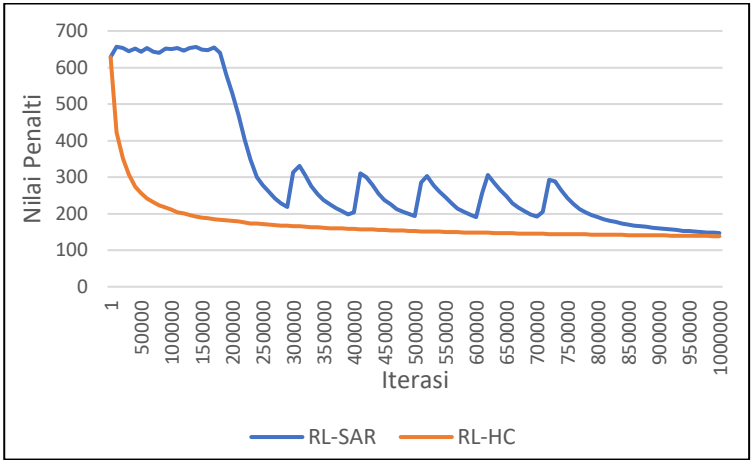
Selanjutnya untuk melihat performa algoritma secara lebih rinci akan dilakukan analisa terhadap setiap unit rumah sakit. Terdapat 2 jenis analisa yang dilakukan, yaitu analisa terhadap persebaran nilai penalti dan analisa kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-HC* pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.12 berikut.



Gambar 6.12 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-HC* pada Optur 4

Pada Gambar 6.12 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal tersebut dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-HC* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.13 berikut.

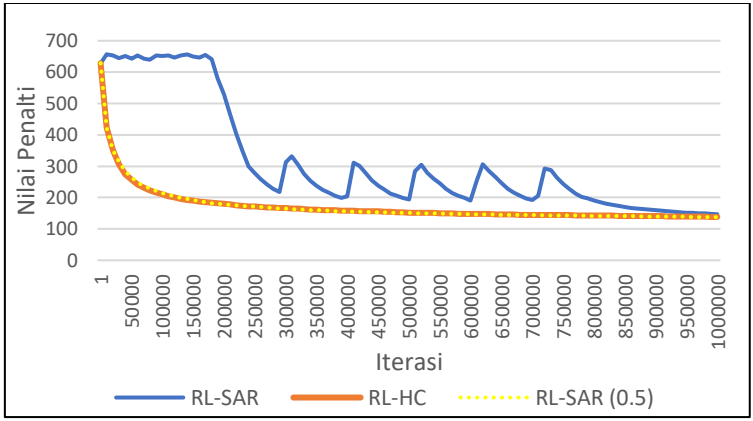


Gambar 6.13 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-HC* dalam menemukan nilai penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.13 dapat dilihat bahwa garis algoritma *RL-HC* menurun secara signifikan pada awal iterasi kemudian melandai hingga akhir iterasi. Pada awal iterasi, algoritma *RL-HC* mampu menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis menurun yang signifikan. Namun, setelah itu algoritma *RL-HC* tidak lagi menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis yang melandai. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Meskipun algoritma *RL-HC* menghasilkan peningkatan solusi yang lebih baik daripada algoritma *RL-SAR*, namun dengan karakteristik algoritma *RL-HC* menyebabkan algoritma hanya melakukan eksploitasi dan terjebak pada *local optima*. Kemudian algortima *RL-SAR* memiliki garis yang bersifat fluktuatif dari awal iterasi hingga akhir iterasi. Pada awal hingga akhir iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai

penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*.

Pada tahap uji coba parameter menunjukan bahwa terdapat parameter yang dapat menghasilkan nilai penalti yang lebih kecil, yaitu *cooling rate* 0.5. Sehingga dilakukan percobaan dengan mengubah parameter *cooling rate* untuk mengetahui kemampuan algoritma *RL-SAR* dalam menemukan nilai penalti yang lebih baik. Hasil percobaan tersebut kemudian dibandingkan dengan percobaan yang telah dilakukan sebelumnya. Perbandingan kemampuan algoritma *RL-SAR*, *RL-HC* dan *RL-SAR* dengan *cooling rate* 0.5 dapat dilihat pada Gambar 6.14 berikut.

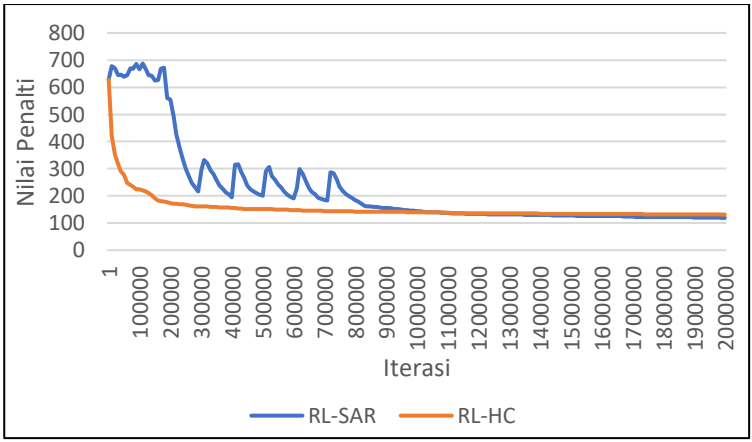


Gambar 6.14 Perbandingan Kemampuan Algoritma *RL-SAR, RL-HC* dan *RL-SAR (cooling rate 0.5)* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.14 dapat dilihat bahwa garis putus-putus kuning merupakan garis algoritma *RL-SAR* dengan *cooling rate 0.5*. Garis tersebut menunjukan bahwa algoritma *RL-SAR (cooling*

rate 0.5*)* mampu menemukan nilai penalti yang lebih baik daripada algortima *RL-HC*. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih baik, namun parameter tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi dan terjebak pada *local optima*. Hal tersebut dapat dilihat dari garis yang terbentuk bersifat menurun pada awal iterasi kemudian melandai hingga akhir iterasi.

Selanjutnya dilakukan percobaan dengan menambah iterasi menjadi 2000000 iterasi. Percobaan ini hanya dilakukan pada unit rumah sakit Optur 4 untuk melihat kemampuan algoritma *RL-SAR* secara lebih jauh. Hasil percobaan penambahan iterasi dapat dilihat pada Gambar 6.15 berikut.

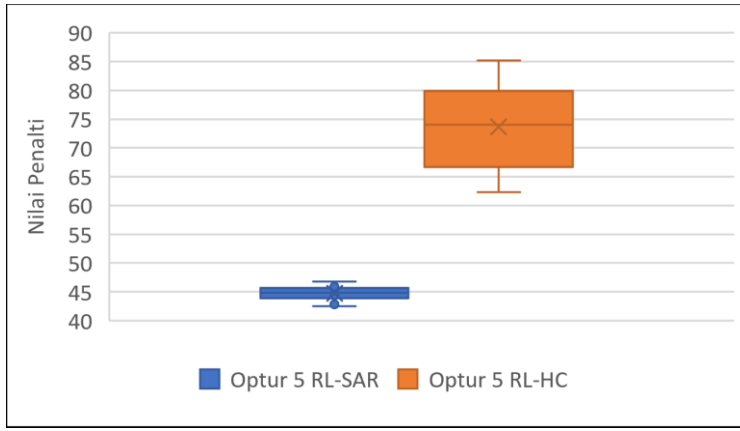


Gambar 6.15 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-HC* dalam Menemukan Nilai Penalti Selama 2000000 Iterasi pada Unit Rumah Sakit Optur 4

Berdasarkan Gambar 6.15 algoritma *RL-SAR* dapat menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi. Dengan penambahan iterasi, algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih kecil daripada algoritma *RL-HC*. Hal tersebut menunjukan bahwa pada percobaan 1000000 iterasi algoritma mampu melakukan eksplorasi dan eksploitasi dari awal hingga akhir iterasi. Kemudian jika iterasi

ditambah menjadi 2000000 iterasi, maka penambahan iterasi tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi hingga akhir iterasi.

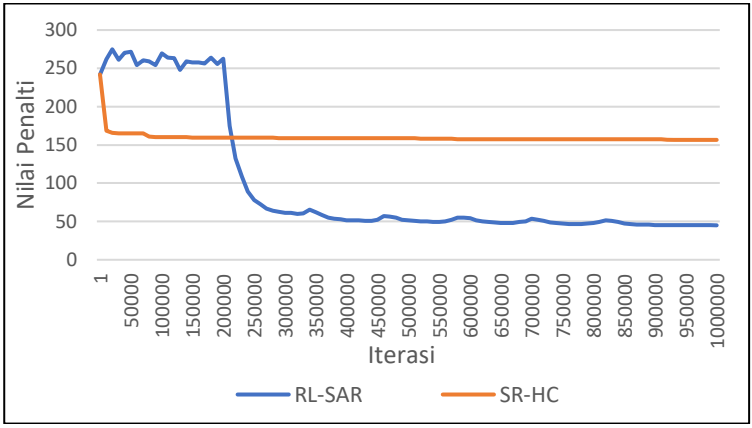
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-HC* pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.16 berikut.



Gambar 6.16 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-HC* pada Optur 5

Pada Gambar 6.16 dapat dilihat bahwa algoritma *RL-HC* memiliki persebaran nilai penalti yang tidak simetris. Hal ini dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-HC* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.17 berikut.

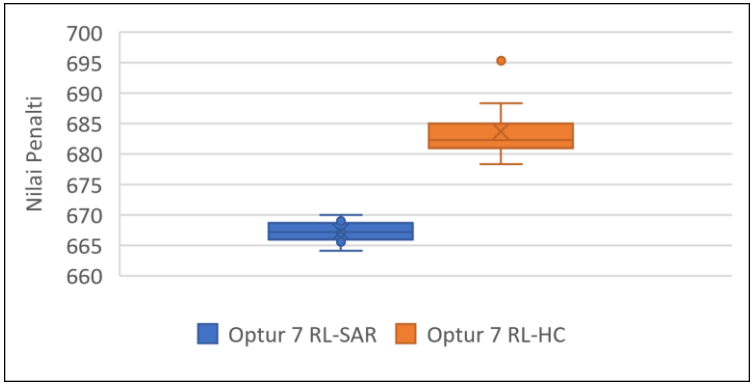


Gambar 6.17 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-HC* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 5

Pada Gambar 6.17 dapat dilihat bahwa garis algoritma *RL-HC* menurun secara signifikan pada awal iterasi kemudian melandai hingga akhir iterasi. Pada awal iterasi, algoritma *RL-HC* mampu menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis menurun yang signifikan. Namun, setelah itu algoritma *RL-HC* tidak lagi menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis yang melandai. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Karakteristik itu menyebabkan algoritma *RL-HC* hanya melakukan eksploitasi dan terjebak pada *local optima*. Kemudian algortima *RL-SAR* memiliki garis yang bersifat fluktuatif pada awal iterasi dan cenderung menurun hingga akhir iterasi. Pada awal iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima

nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*.

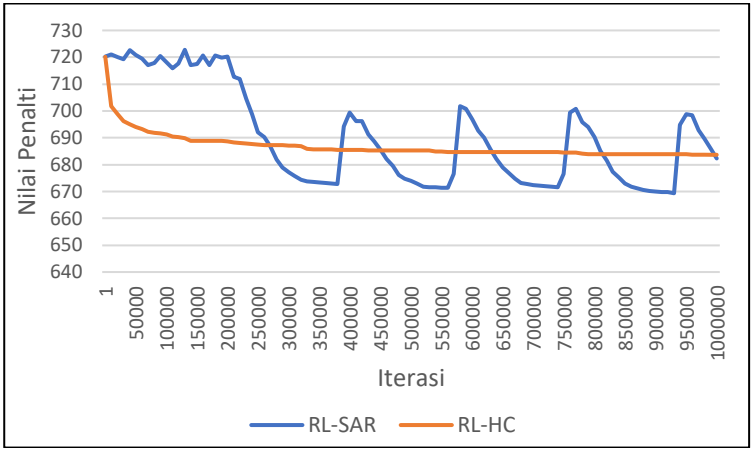
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-HC* pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.18 berikut.



Gambar 6.18 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-HC* pada Optur 7

Pada Gambar 6.18 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal tersebut dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-HC* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-HC* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan. Selain itu algoritma *RL-HC* memiliki nilai penalti yang jauh dari nilai maksimal penalti.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-HC* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.19 berikut.



Gambar 6.19 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-HC* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 7

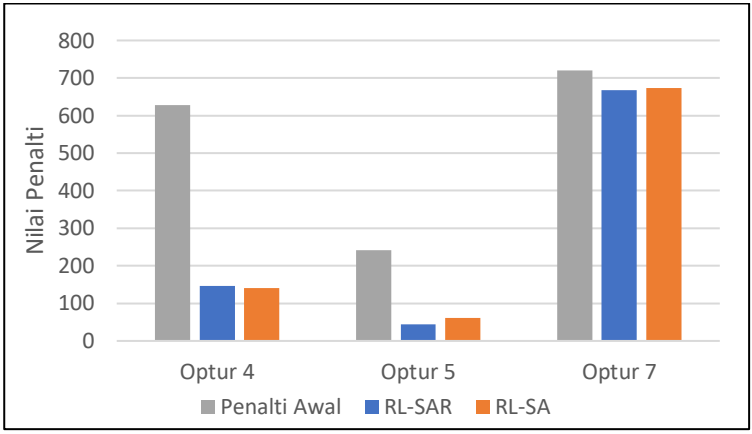
Pada Gambar 6.19 dapat dilihat bahwa garis algoritma *RL-HC* mengalami penurunan yang signifikan pada awal iterasi kemudian cenderung melandai hingga akhir iterasi. Pada awal iterasi, algoritma *RL-HC* mampu menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis menurun yang signifikan. Namun, setelah itu algoritma *RL-HC* tidak lagi menemukan nilai penalti yang lebih baik secara berurutan sehingga terbentuk garis yang melandai. Hal tersebut terjadi karena karakteristik algoritma yang selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih buruk. Karakteristik itu menyebabkan algoritma *RL-HC* hanya melakukan eksploitasi dan terjebak pada *local optima*. Kemudian algoritma *RL-SAR* memiliki garis yang bersifat fluktiatif hingga akhir iterasi. Hal tersebut menandakan bahwa algortima mampu menemukan nilai penalti yang bervariasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima

nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*. Bahkan hingga akhir iterasi algoritma *RL-SAR* masih melakukan eksplorasi yang memungkinkan algoritma untuk menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi.

Berdasarkan analisa yang telah dilakukan, algoritma *RL-SAR* memiliki persebaran nilai penalti yang lebih konstan daripada algoritma *RL-HC*. Selain itu algoritma *RL-SAR* juga memiliki kemampuan yang lebih baik daripada algoritma *RL-HC* dalam hal menemukan nilai penalti yang lebih baik. Hal tersebut dapat dilihat dari penurunan nilai penalti yang dihasilkan oleh kedua algoritma. Algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih kecil pada 2 unit rumah sakit, sedangkan algorima *RL-HC* hanya menemukan nilai penalti yang lebih kecil pada 1 unit rumah sakit. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* dapat menemukan nilai penalti yang lebih baik, namun dengan *cooling rate* 0.5 membuat algoritma *RL-SAR* cenderung melakukan eksploitasi daripada eksplorasi.

6. 4. 4. Perbandingan dengan Algoritma *Reinforcement Learning – Simulated Annealing*

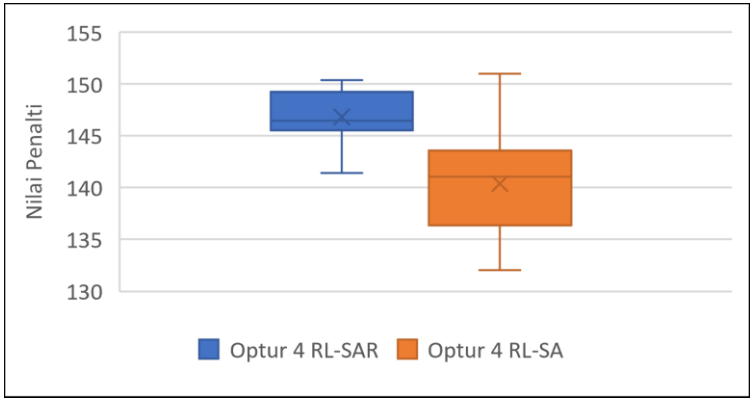
Perbandingan performa algoritma *RL-SAR* dengan *RL-SA* dilakukan dengan melihat nilai penalti yang dihasilkan oleh kedua algoritma tersebut. Performa algoritma yang lebih baik ditunjukan dengan semakin kecil nilai penalti yang dihasilkan. Parameter yang digunakan pada algoritma *RL-SA* telah disesuaikan sedemikian rupa agar algoritma tersebut mampu mencari solusi dengan jumlah iterasi yang sama. Performa dari algoritma *RL-SAR* dan *RL-SA* dapat dilihat pada Gambar 6.20 berikut.



Gambar 6.20 Perbandingan Peningkatan Solusi Algoritma *RL-SAR* dan *RL-SA*

Pada Gamber 6.20 menampilkan perbandingan rata-rata nilai penalti dari algoritma *RL-SAR* dan algoritma *RL-SA* pada unit rumah sakit Optur 4, unit rumah sakit Optur 5 dan unit rumah sakit Optur 7. Pada unit rumah sakit Optur 4, masing-masing algoritma *RL-SAR* dan *RL-SA* menghasilkan nilai penalti sebesar 146.835 dan 140.357 dari nilai penalti awal sebesar 628.420. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 77%, sedangkan algoritma *RL-SA* menghasilkan peningkatan solusi sebesar 78%. Pada unit rumah sakit Optur 5, masing-masing algoritma *RL-SAR* dan *RL-SA* menghasilkan nilai penalti sebesar 44.715 dan 60.802 dari nilai penalti sebesar 241.985. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 82%, sedangkan algoritma *RL-SA* menghasilkan peningkatan solusi sebesar 75%. Pada unit rumah sakit Optur 7, masing-masing algoritma *RL-SAR* dan *RL-SA* menghasilkan nilai penalti sebesar 667.291 dan 672.874 dari nilai penalti awal sebesar 720.167. Berdasarkan nilai penalti tersebut algoritma *RL-SAR* menghasilkan peningkatan solusi sebesar 7.3%, sedangkan algoritma *RL-SA* menghasilkan peningkatan solusi sebesar 6.6%.

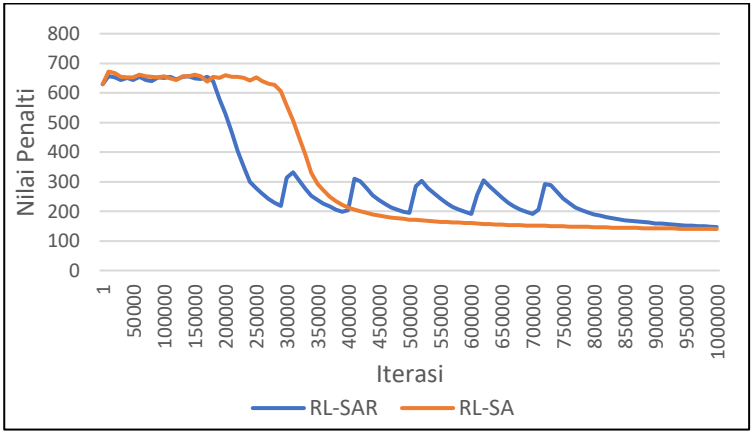
Selanjutnya untuk melihat performa algoritma secara lebih rinci akan dilakukan analisa terhadap setiap unit rumah sakit. Terdapat 2 jenis analisa yang dilakukan, yaitu analisa terhadap persebaran nilai penalti dan analisa terhadap kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-SA* pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.21 berikut.



Gambar 6.21 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-SA* pada Optur 4

Pada Gambar 6.21 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal tersebut dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-SA* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-SA* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-SA* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.22 berikut.

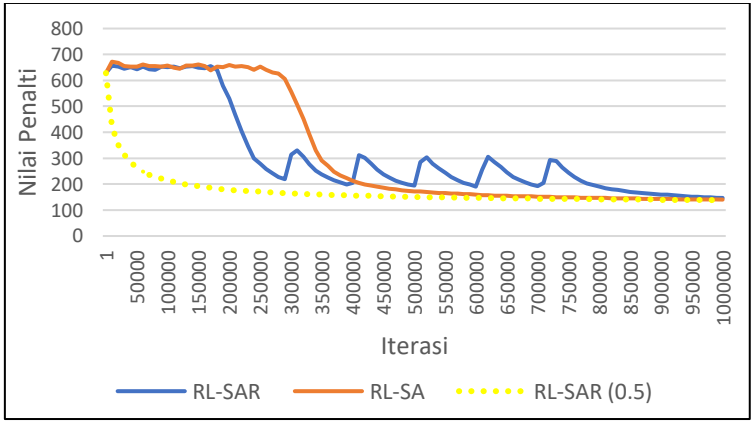


Gambar 6.22 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-SA* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.22 dapat dilihat bahwa garis algoritma *RL-SA* bersifat fluktuatif pada awal iterasi kemudian menurun hingga akhir iterasi. Pada awal iterasi algoritma *RL-SA* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SA* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SA* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SA* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SA* mampu melakukan eksploitasi sekaligus eksplorasi. Meskipun algoritma *RL-SA* mampu melakukan eksploitasi dan eksplorasi sekaligus, namun algoritma *RL-SA* masih memiliki kemungkinan terjebak pada *local optima*. Hal tersebut dapat dilihat pada akhir iterasi algoritma *RL-SA* cenderung memiliki garis yang melandai. Kemudian algortima *RL-SAR* memiliki garis yang bersifat fluktuatif dari awal iterasi hingga akhir iterasi. Pada awal hingga akhir iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai

penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*.

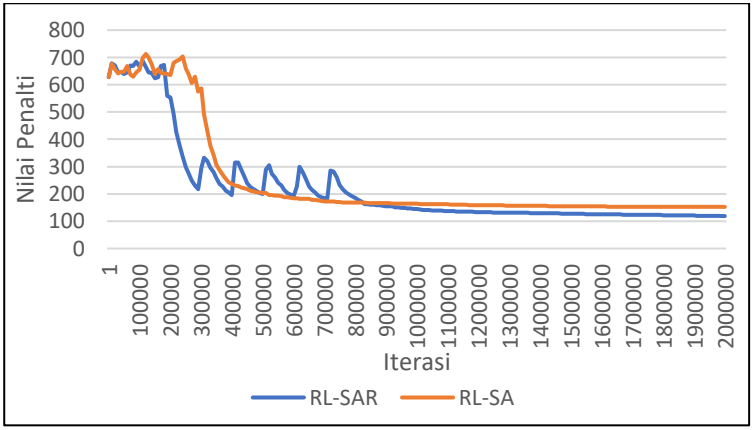
Pada tahap uji coba parameter menunjukan bahwa terdapat parameter yang dapat menghasilkan nilai penalti yang lebih kecil, yaitu *cooling rate* 0.5. Sehingga dilakukan percobaan dengan mengubah parameter *cooling rate* untuk mengetahui kemampuan algoritma *RL-SAR* dalam menemukan nilai penalti yang lebih baik. Hasil percobaan tersebut kemudian dibandingkan dengan percobaan yang telah dilakukan sebelumnya. Perbandingan kemampuan algoritma *RL-SAR*, *RL-SA* dan *RL-SAR* dengan *cooling rate* 0.5 dapat dilihat pada Gambar 6.23 berikut.



Gambar 6.23 Perbandingan Kemampuan Algoritma *RL-SAR*, *RL-SA* dan *RL-SAR (cooling rate 0.5)* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 4

Pada Gambar 6.23 dapat dilihat bahwa garis putus-putus kuning merupakan garis algoritma *RL-SAR* dengan *cooling rate 0.5*. Garis tersebut menunjukan bahwa algoritma *RL-SAR (cooling rate* 0.5*)* mampu menemukan nilai penalti yang lebih baik daripada algortima *RL-SA*. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih baik, namun parameter tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi dan terjebak pada *local optima*. Hal tersebut dapat dilihat dari garis yang terbentuk bersifat menurun pada awal iterasi kemudian melandai hingga akhir iterasi.

Selanjutnya dilakukan percobaan dengan menambah iterasi menjadi 2000000 iterasi. Percobaan ini hanya dilakukan pada unit rumah sakit Optur 4 untuk melihat kemampuan algoritma *RL-SAR* secara lebih jauh. Hasil percobaan penambahan iterasi dapat dilihat pada Gambar 6.24 berikut.

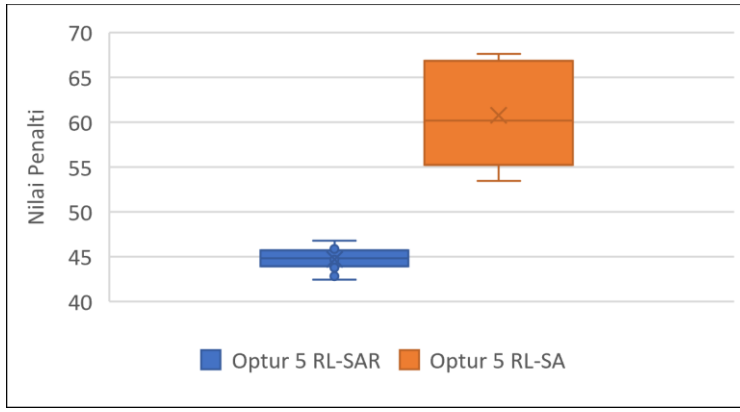


Gambar 6.24 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-SA* dalam Menemukan Nilai Penalti Selama 2000000 Iterasi pada Unit Rumah Sakit Optur 4

Berdasarkan Gambar 6.24 algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi. Dengan penambahan iterasi, algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih kecil daripada

algoritma *RL-SA*. Hal tersebut menunjukan bahwa pada percobaan 1000000 iterasi algoritma mampu melakukan eksplorasi dan eksploitasi dari awal hingga akhir iterasi. Kemudian jika iterasi ditambah menjadi 2000000 iterasi, maka penambahan iterasi tersebut membuat algoritma *RL-SAR* cenderung melakukan eksploitasi hingga akhir iterasi.

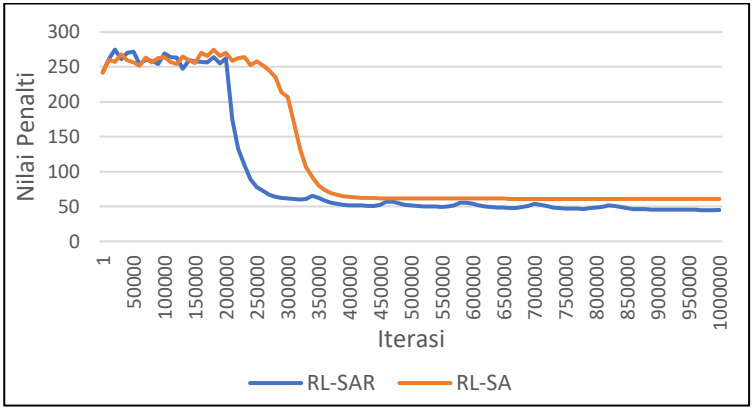
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-SA* pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.25 berikut.



Gambar 6.25 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-SA* pada Optur 5

Pada Gambar 6.25 dapat dilihat bahwa algoritma *RL-SA* memiliki persebaran nilai penalti yang tidak simetris. Hal ini dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-SA* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-SA* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-SA* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.26 berikut.

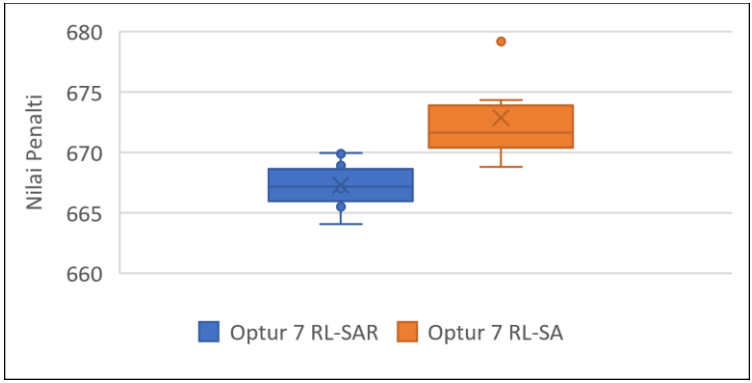


Gambar 6.26 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-SA* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 5

Pada Gambar 6.26 dapat dilihat bahwa garis algoritma *RL-SA* bersifat fluktuatif pada awal iterasi kemudian menurun hingga akhir iterasi. Pada awal iterasi algoritma *RL-SA* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SA* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SA* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SA* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SA* mampu melakukan eksploitasi sekaligus eksplorasi. Meskipun algoritma *RL-SA* mampu melakukan eksploitasi dan eksplorasi sekaligus, namun algoritma *RL-SA* masih memiliki kemungkinan terjebak pada *local optima*. Hal tersebut dapat dilihat pada akhir iterasi algoritma *RL-SA* cenderung memiliki garis yang melandai. Kemudian algortima *RL-SAR* memiliki garis yang bersifat fluktuatif dari awal iterasi hingga akhir iterasi. Pada awal hingga akhir iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada

algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*.

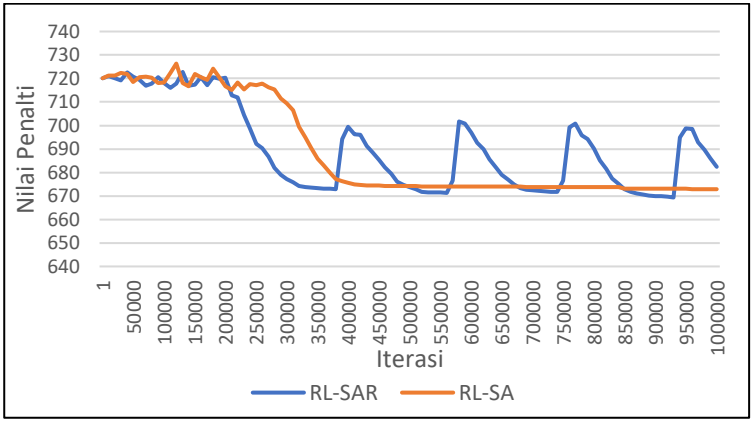
Kemudian perbandingan persebaran nilai penalti algoritma *RL-SAR* dan *RL-SA* pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.27 berikut.



Gambar 6.27 Persebaran Nilai Penalti Algoritma *RL-SAR* dan *RL-SA* pada Optur 7

Pada Gambar 6.27 dapat dilihat bahwa kedua algoritma memiliki persebaran nilai penalti yang tidak simetris. Hal tersebut dapat dilihat dari jarak antara tepi bawah dan tepi atas ke median data yang tidak sama. Jika dibandingkan, algoritma *RL-SA* memiliki persebaran nilai penalti yang lebih luas daripada algoritma *RL-SAR*. Hal ini menunjukan bahwa algoritma *RL-SA* memiliki nilai penalti yang lebih beragam dan cenderung tidak konstan. Selain itu algoritma *RL-SA* memiliki nilai penalti yang jauh dari nilai maksimal penalti.

Kemudian perbandingan kemampuan algoritma *RL-SAR* dan *RL-SA* dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.28 berikut.



Gambar 6.28 Perbandingan Kemampuan Algoritma *RL-SAR* dan *RL-SA* dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 7

Pada Gambar 6.28 dapat dilihat bahwa garis algoritma *RL-SA* bersifat fluktuatif pada awal iterasi kemudian menurun hingga akhir iterasi. Pada awal iterasi algoritma *RL-SA* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SA* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SA* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SA* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SA* mampu melakukan eksploitasi sekaligus eksplorasi. Meskipun algoritma *RL-SA* mampu melakukan eksploitasi dan eksplorasi sekaligus, namun algoritma *RL-SA* masih memiliki kemungkinan terjebak pada *local optima*. Hal tersebut dapat dilihat pada akhir iterasi algoritma *RL-SA* cenderung memiliki garis yang melandai. Kemudian algortima *RL-SAR* memiliki garis yang bersifat

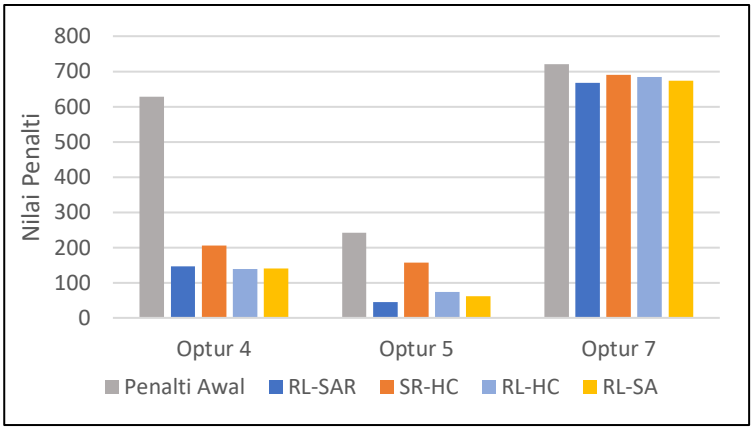
fluktuatif dari awal iterasi hingga akhir iterasi. Pada awal hingga akhir iterasi algoritma *RL-SAR* mampu menemukan nilai penalti yang bervarasi, baik nilai penalti yang lebih baik maupun nilai penalti yang lebih buruk. Hal tersebut menyebabkan garis pada algoritma *RL-SAR* bersifat fluktuatif. Garis fluktuatif pada algoritma *RL-SAR* disebabkan karena karakteristik algoritma yang mampu menerima nilai penalti yang lebih baik dan nilai penalti yang lebih buruk sekaligus. Karakteristik tersebut memungkinkan algoritma *RL-SAR* menemukan nilai penalti yang lebih baik setelah menerima nilai penalti yang lebih buruk. Selain itu, garis fluktuatif menunjukan bahwa algoritma *RL-SAR* mampu melakukan ekspoitasi sekaligus eksplorasi, sehingga algoritma *RL-SAR* memiliki kemungkinan untuk tidak terjebak pada *local optima*. Bahkan hingga akhir iterasi algoritma *RL-SAR* masih melakukan eksplorasi yang memungkinkan algoritma untuk menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi.

Berdasarkan analisa yang telah dilakukan, algoritma *RL-SAR* memiliki persebaran nilai penalti yang lebih konstan daripada algoritma *RL-SA*. Selain itu algoritma *RL-SAR* juga memiliki kemampuan yang lebih baik daripada algoritma *RL-SA* dalam hal menemukan nilai penalti yang lebih baik. Hal tersebut dapat dilihat dari penurunan nilai penalti yang dihasilkan oleh kedua algoritma. Algoritma *RL-SAR* mampu menemukan nilai penalti yang lebih kecil pada 2 unit rumah sakit, sedangkan algorima *RL-SA* hanya menemukan nilai penalti yang lebih kecil pada 1 unit rumah sakit. Meskipun dengan *cooling rate* 0.5 algoritma *RL-SAR* dapat menemukan nilai penalti yang lebih baik, namun dengan *cooling rate* 0.5 membuat algoritma *RL-SAR* cenderung melakukan eksploitasi daripada eksplorasi.

6. 4. 5. Perbandingan Antar Algoritma

Perbandingan antar algoritma bertujuan untuk melihat performa masing-masing algoritma secara keseluruhan. Performa algoritma dilihat berdasarkan nilai penalti yang dihasilkan oleh masing-masing algoritma. Semakin kecil nilai penalti, maka

performa algoritma semakin baik. Performa dari masing-masing algoritma dapat dilihat pada Gambar 6.29 berikut.



Gambar 6.29 Perbandingan Nilai Penalti antar Algoritma dengan Penalti Awal

Pada Gambar 6.29 menampilkan perbandingan rata-rata nilai penalti yang dihasilkan oleh masing-masing algoritma. Algoritma yang digunakan pada penelitian tugas akhir ini yaitu *RL-SAR*, *SR-HC*, *RL-HC*, dan *RL-SA*. Berdasarkan Gambar 6.26 masing-masing algoritma mampu menghasilkan nilai penalti yang lebih kecil dari nilai penalti awal. Pada unit rumah sakit Optur 4, masing-masing algoritma *RL-SAR*, *SR-HC*, *RL-HC*, dan *RL-SA* menghasilkan nilai penalti secara berturut-turut sebesar 146.835, 204.681, 138.331, dan 140.357 dari nilai penalti sebesar 628.420. Berdasarkan nilai penalti tersebut, maka masing-masing algoritma menghasilkan peningkatan solusi secara berturut-turut sebesar 77%, 67%, 78%, dan 78%. Pada unit rumah sakit Optur 5, masing-masing algoritma *RL-SAR*, *SR-HC*, *RL-HC*, dan *RL-SA* menghasilkan nilai penalti secara berturut-turut sebesar 44.715, 157.284, 73.674, dan 60.802 dari nilai penalti sebesar 241.985. Berdasarkan nilai penalti tersebut, maka masing-masing algoritma menghasilkan peningkatan solusi secara berturut-turut sebesar 82%, 35%, 70%, dan 75%. Pada unit rumah sakit Optur 7, masing-masing algoritma *RL-SAR*, *SR-HC*, *RL-HC*, dan *RL-SA* menghasilkan

nilai penalti secara berturut-turut sebesar 667.291, 690.575, 683.669, dan 672.874 dari nilai penalti sebesar 720.167. Berdasarkan nilai penalti tersebut, maka masing-masing algoritma menghasilkan peningkatan solusi secara berturut-turut sebesar 7.3%, 4%, 5%, dan 6.6%.

Nilai penalti yang dihasilkan oleh solusi awal dan solusi hasil optimasi algoritma *RL-SAR* berasal dari pelanggaran *soft constraint* yang ada. Perhitungan nilai penalti tersebut berdasarkan pada model matematis yang telah dijelaskan pada BAB 4. Terdapat 9 *soft constraint* yang digunakan untuk menghitung nilai penalti seperti pada Tabel 6.9 berikut.

Tabel 6.9 Daftar *Soft Constraint*

Soft Constraint	Deskripsi
SC – 1	Tidak boleh terlalu banyak hari kerja berurutan dengan kategori sif yang sama
SC – 2	Tidak boleh terlalu banyak hari kerja berurutan
SC – 3	Tidak boleh terlalu sedikit hari kerja berurutan dengan kategori sif yang sama
SC – 4	Tidak boleh terlalu sedikit hari kerja berututan
SC – 5	Tidak boleh menyimpang terlalu jauh dari jumlah minimum dan maksimum jumlah sif pada setiap kategorinya
SC – 6	Tidak boleh menyimpang terlalu jauh dari kontrak kerja perawat
SC – 7	Mengelompokan waktu libur setiap perawat
SC – 8	Memenuhi pola sif yang diinginkan
SC – 9	Tidak boleh terlalu banyak pola sif yang tidak diinginkan

Masing-masing *soft constraint* menghasilkan sebuah nilai penalti yang mencerminkan pelanggaran dari *soft constraint* tersebut. Nilai penalti pada Gambar 6.29 merupakan akumulasi nilai penalti dari pelanggaran seluruh *soft constraint* pada masing-masing unit rumah sakit. Selanjutnya dilakukan penjabaran nilai penalti untuk melihat *soft constaraint* mana yang menyebabkan besarnya nilai penalti yang didapat. Penjabaran nilai penalti hanya dilakukan pada nilai penalti solusi awal dan nilai penalti hasil optimasi algoritma *RL-SAR* yang dapat dilihat pada Tabel 6.10 berikut.

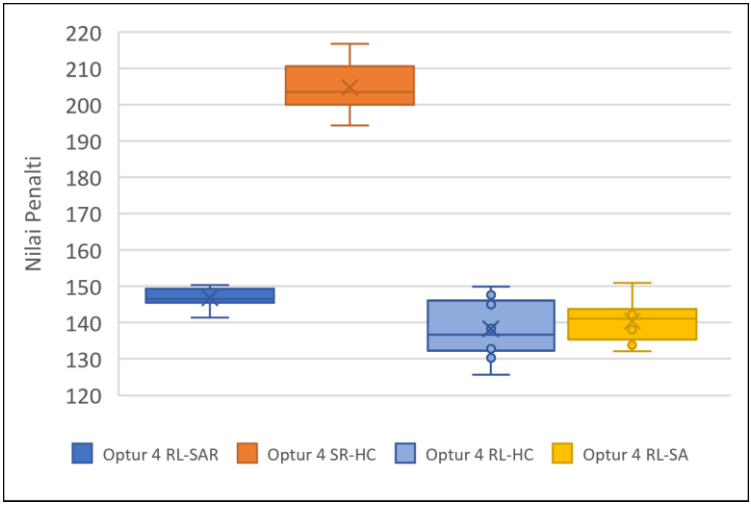
Tabel 6.10 Penjabaran Nilai Penalti Pelanggaran *Soft Constraint*

Soft Constraint	Optur 4		Optur 5		Optur 7	
	p(a)	*p(s)*	*p(a)*	*p(s)*	*p(a)*	*p(s)*
SC – 1	49	2	12	2	0	0
SC – 2	33	0	19	0	0	0
SC – 3	344	14	144	4	44	4
SC – 4	0	0	0	0	0	0
SC – 5	0	0	39.50	16.76	0	0
SC – 6	2.04	1.50	0.94	1.44	9.88	9.63
SC – 7	200.3	113.1	26.55	20.37	36.29	23.13
SC – 8	0	0	0	0	630	630
SC – 9	0	0	0	0	0	0
Total Penalti	628.4	130.6	241.9	44.58	720.2	666.8

Keterangan : *p(a)* = penalti awal; *p(s)* = penalti optimasi.

Berdasarkan Tabel 6.10 dapat dilihat bahwa terdapat dua *soft constraint* yang tidak menghasilkan nilai penalti yaitu *soft constraint* 4 dan *soft constraint* 9. Hal tersebut dikarenakan pada *soft constraint* 4 dan *soft constraint* 9 tidak terdapat sebuah nilai parameter yang membatasi *soft constraint* tersebut. *Soft constraint* 4 membatasi terlalu sedikitnya hari kerja berurutan untuk setiap perawat, sedangkan *soft constraint* 9 membatasi adanya pola sif yang tidak diinginkan pada setiap perawat. Kemudian *soft constraint* yang banyak menghasilkan nilai penalti yaitu *soft constraint* 3, *soft constraint* 7, dan *soft constraint* 8. *Soft constraint* 3 dan *soft constraint* 7 memiliki nilai penalti yang besar pada solusi awal, namun nilai penalti tersebut masih dapat diminimalisir setelah dilakukan optimasi solusi. *Soft constraint* 3 membatasi terlalu sedikitnya hari kerja berurutan dengan kategori sif tertentu, sedangkan *soft constraint* 7 membahas tentang pengelompokan waktu libur perawat. Kemudian *soft constraint* 8 yang membahas pola sif yang diinginkan perawat hanya menghasilkan nilai penalti yang cukup besar pada unit rumah sakit Optur 7. Hal tersebut disebabkan karena pada unit rumah sakit Optur 7 terdapat pola sif yang diinginkan perawat, namun bila memenuhi pola sif tersebut akan melanggar *hard constraint* 5. Sehingga nilai penalti *soft constraint* 8 pada unit rumah sakit Optur 7 tidak dapat diminimalisir meskipun telah dilakukan optimasi.

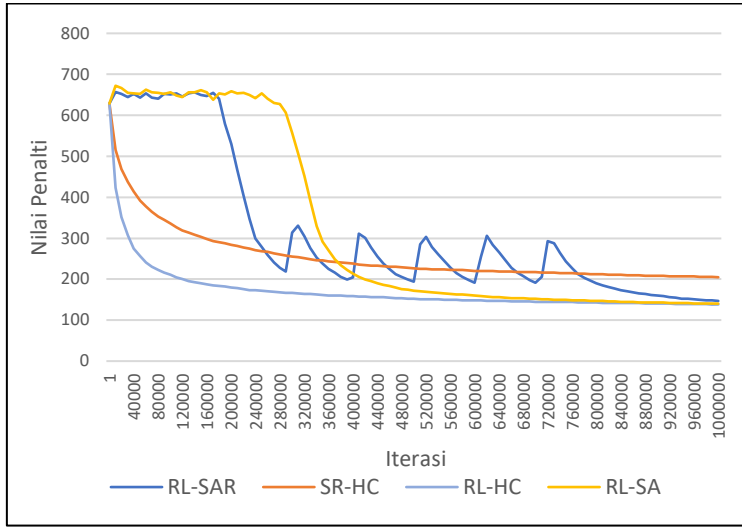
Selanjutnya untuk melihat performa algoritma secara lebih rinci akan dilakukan analisa terhadap setiap unit rumah sakit. Terdapat 2 jenis analisa yang dilakukan, yaitu analisa terhadap persebaran nilai penalti dan analisa terhadap kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Perbandingan persebaran nilai penalti masing-masing algoritma pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.30 berikut.



Gambar 6.30 Perbandingan Persebaran Nilai Penalti Masing-masing Algoritma pada Unit Rumah Sakit Optur 4

Berdasarkan Gambar 6.30 masing-masing algoritma menghasilkan persebaran nilai penalti yang berbeda-beda. Dari keempat algoritma yang diuji, algoritma *RL-SAR* cenderung menghasilkan persebaran nilai penalti yang lebih konstan daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Hal tersebut dapat dilihat dari diagram boxplot yang terbentuk. Semakin kecil diagram boxplot yang terbentuk, maka nilai penalti yang dihasilkan cenderung konstan dan seragam.

Kemudian perbandingan kemampuan masing-masing algoritma dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 4 dapat dilihat pada Gambar 6.31 berikut.

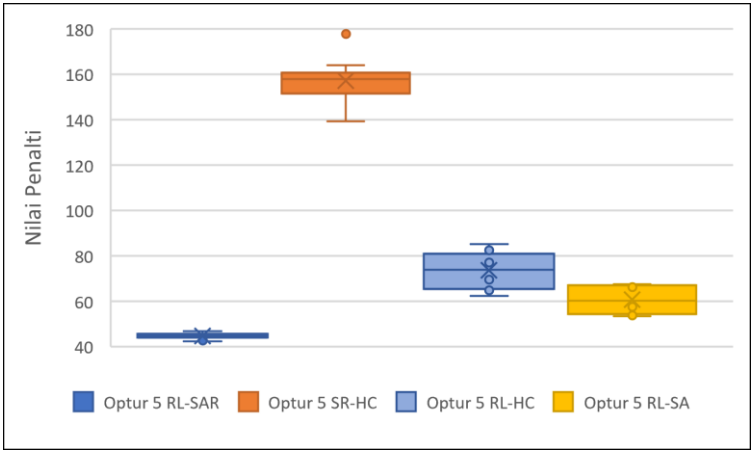


Gambar 6.31 Perbandingan Kemampuan Masing-masing Algoritma dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 4

Berdasarkan Gambar 6.31 masing-masing algoritma memiliki karakteristik yang berbeda-beda dalam menemukan nilai penalti yang lebih baik. Algoritma *SR-HC* dan *RL-HC* memiliki garis menurun pada awal iterasi kemudian melandai hingga akhir iterasi. Garis tersebut menunjukan bahwa algoritma *SR-HC* dan *RL-HC* memiliki karakteristik selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih jelek. Dengan karakteristik tersebut membuat algoritma *SR-HC* dan *RL-HC* hanya melakukan eksploitasi dan dapat terjebak pada *local optima*. Berbeda dengan algoritma *RL-SA* dan *RL-SAR* yang memiliki garis fluktuatif selama proses pencarian nilai penalti. Garis fluktuatif tersebut menunjukan bahwa algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik tidak hanya menerima nilai penalti yang lebih baik, namun nilai penalti yang lebih jelek juga diterima dengan probabilitas tertentu. Dengan karakteristik tersebut membuat algoritma *RL-SA* dan *RL-SAR* dapat melakukan eksploitasi sekaligus eksplorasi yang dapat membantu algoritma agar tidak terjebak pada *local optima*. Meskipun algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik

yang sama, namun algoritma *RL-SA* hanya melakukan eksplorasi pada awal iterasi dan cenderung melakukan eksploitasi hingga akhir iterasi. Berbeda dengan algoritma *RL-SAR* yang dapat melakukan eksplorasi dan eksploitasi hingga akhir iteasi. Meskipun nilai penalti yang dihasilkan oleh algoritma *RL-SAR* lebih besar daripada algoritma *RL-HC* dan *RL-SA*, namun dengan karakteristik algoritma *RL-SAR* memungkinkan algoritma menemukan nilai penalti yang lebih baik jika dilakukan penambahan iterasi.

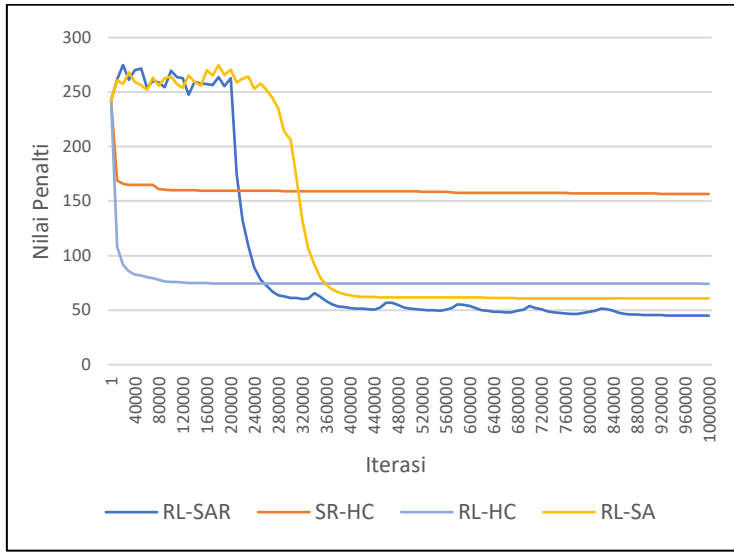
Kemudian perbandingan persebaran nilai penalti antar algoritma pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.32 berikut.



Gambar 6.32 Perbandingan Persebaran Nilai Penalti Masing-masing Algoritma pada Unit Rumah Sakit Optur 5

Berdasarkan Gambar 6.32 masing-masing algoritma menghasilkan persebaran nilai penalti yang berbeda-beda. Dari keempat algoritma yang diuji, algoritma *RL-SAR* cenderung menghasilkan persebaran nilai penalti yang lebih konstan daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Hal tersebut dapat dilihat dari diagram boxplot yang terbentuk. Semakin kecil diagram boxplot yang terbentuk, maka nilai penalti yang dihasilkan cenderung konstan dan seragam.

Kemudian perbandingan kemampuan masing-masing algoritma dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 5 dapat dilihat pada Gambar 6.33 berikut.

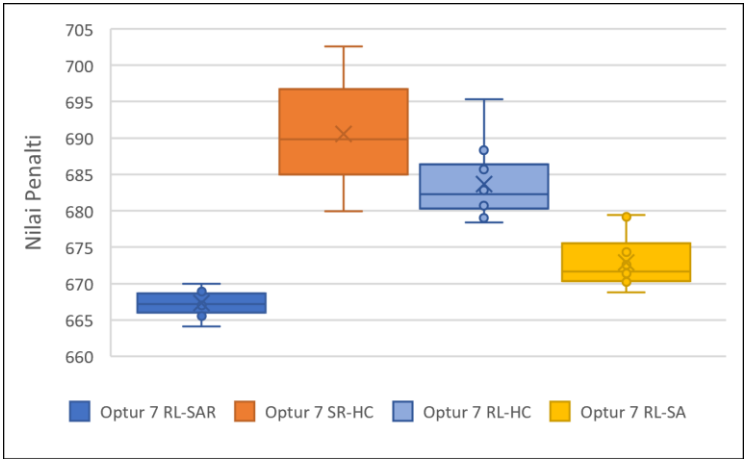


Gambar 6.33 Perbandingan Kemampuan Masing-masing Algoritma dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 5

Berdasarkan Gambar 6.33 masing-masing algoritma memiliki karakteristik yang berbeda-beda dalam menemukan nilai penalti yang lebih baik. Algoritma *SR-HC* dan *RL-HC* memiliki garis menurun pada awal iterasi kemudian melandai hingga akhir iterasi. Garis tersebut menunjukan bahwa algoritma *SR-HC* dan *RL-HC* memiliki karakteristik selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih jelek. Dengan karakteristik tersebut membuat algoritma *SR-HC* dan *RL-HC* hanya melakukan eksploitasi dan dapat terjebak pada *local optima*. Berbeda dengan algoritma *RL-SA* dan *RL-SAR* yang memiliki garis fluktuatif selama proses pencarian nilai penalti. Garis fluktuatif tersebut menunjukan bahwa algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik tidak hanya menerima nilai penalti yang lebih baik, namun nilai penalti yang lebih jelek juga diterima dengan probabilitas tertentu. Dengan

karakteristik tersebut membuat algoritma *RL-SA* dan *RL-SAR* dapat melakukan eksploitasi sekaligus eksplorasi yang dapat membantu algoritma agar tidak terjebak pada *local optima*. Meskipun algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik yang sama, namun algoritma *RL-SA* hanya melakukan eksplorasi pada awal iterasi dan cenderung melakukan eksploitasi hingga akhir iterasi. Berbeda dengan algoritma *RL-SAR* yang dapat melakukan eksplorasi dan eksploitasi hingga akhir iteasi.

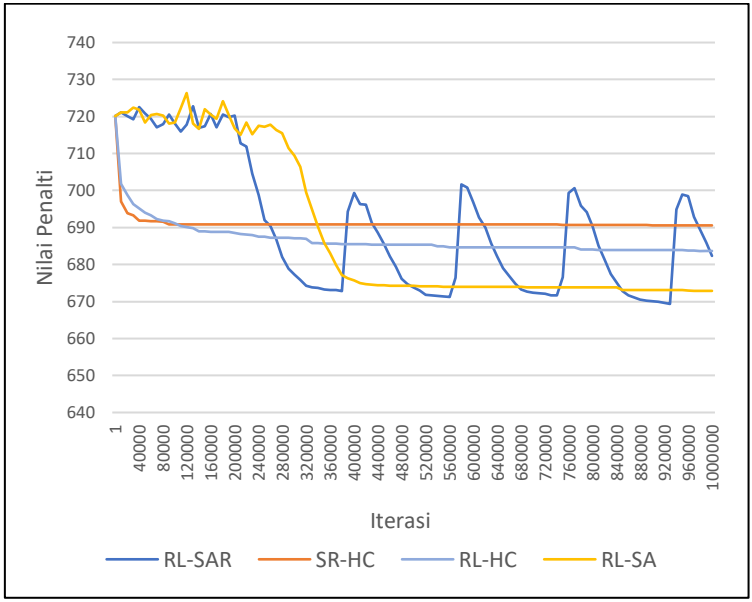
Kemudian perbandingan persebaran nilai penalti antar algoritma pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.34 berikut.



Gambar 6.34 Perbandingan Persebaran Nilai Penalti Masing-masing Algoritma pada Unit Rumah Sakit Optur 7

Berdasarkan Gambar 6.34 masing-masing algoritma menghasilkan persebaran nilai penalti yang berbeda-beda. Dari keempat algoritma yang diuji, algoritma *RL-SAR* cenderung menghasilkan persebaran nilai penalti yang lebih konstan daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Hal tersebut dapat dilihat dari diagram boxplot yang terbentuk. Semakin kecil diagram boxplot yang terbentuk, maka nilai penalti yang dihasilkan cenderung konstan dan seragam.

Kemudian perbandingan kemampuan masing-masing algoritma dalam menemukan nilai penalti yang lebih baik pada unit rumah sakit Optur 7 dapat dilihat pada Gambar 6.35 berikut.



Gambar 6.35 Perbandingan Kemampuan Masing-masing Algoritma dalam Menemukan Nilai Penalti pada Unit Rumah Sakit Optur 7

Berdasarkan Gambar 6.35 masing-masing algoritma memiliki karakteristik yang berbeda-beda dalam menemukan nilai penalti yang lebih baik. Algoritma *SR-HC* dan *RL-HC* memiliki garis menurun pada awal iterasi kemudian melandai hingga akhir iterasi. Garis tersebut menunjukan bahwa algoritma *SR-HC* dan *RL-HC* memiliki karakteristik selalu menerima nilai penalti yang lebih baik dan menolak nilai penalti yang lebih jelek. Dengan karakteristik tersebut membuat algoritma *SR-HC* dan *RL-HC* hanya melakukan eksploitasi dan dapat terjebak pada *local optima*. Berbeda dengan algoritma *RL-SA* dan *RL-SAR* yang memiliki garis fluktuatif selama proses pencarian nilai penalti. Garis fluktuatif tersebut menunjukan bahwa algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik tidak hanya

menerima nilai penalti yang lebih baik, namun nilai penalti yang lebih jelek juga diterima dengan probabilitas tertentu. Dengan karakteristik tersebut membuat algoritma *RL-SA* dan *RL-SAR* dapat melakukan eksploitasi sekaligus eksplorasi yang dapat membantu algoritma agar tidak terjebak pada *local optima*. Meskipun algoritma *RL-SA* dan *RL-SAR* memiliki karakteristik yang sama, namun algoritma *RL-SA* hanya melakukan eksplorasi pada awal iterasi dan cenderung melakukan eksploitasi hingga akhir iterasi. Berbeda dengan algoritma *RL-SAR* yang dapat melakukan eksplorasi dan eksploitasi hingga akhir iteasi.

Berdasarkan analisa yang telah dilakukan, algoritma *RL-SAR* memiliki performa yang lebih baik daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Hal tersebut dapat dilihat dari nilai penalti yang dihasilkan, persebaran nilai penalti, dan kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Berdasarkan nilai penalti yang dihasilkan, algoritma *RL-SAR* mampu menghasilkan nilai penalti dengan peningkatan sebesar 3% - 47% dari algoritma *SR-HC*, 2% - 12% dari algoritma *RL-HC*, dan 0.7% - 7% dari algoritma *RL-SA*. Berdasarkan persebaran nilai penalti yang dihasilkan, algortima *RL-SAR* memiliki persebaran nilai penalti yang cenderung konstan daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Berdasarkan kemampuan algoritma dalam menemukan nilai penalti yang lebih baik, algoritma *RL-SAR* memiliki kemampuan yang lebih baik daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Hal tersebut dapat dilihat dari karakteristik algoritma *RL-SAR* yang dapat melakukan eksplorasi sekaligus eksploitasi. Berbeda dengan algoritma *SR-HC* dan *RL-HC* yang cenderung melakukan eksploitasi selama proses pencarian nilai penalti. Meskipun algoritma *RL-SA* juga memiliki karakteristik yang sama dengan algoritma *RL-SAR*, namun algoritma *RL-SA* hanya mampu melakukan eksplorasi pada awal iterasi dan cenderung melakukan eksploitasi hingga akhir iterasi. Berbeda dengan karakteristik algoritma *RL-SAR* yang mampu melakukan eksplorasi dan eksploitasi dari awal hingga akhir iterasi.

6. 4. 6. Perbandingan dengan Penelitian Sebelumnya

Perbandingan dengan penelitian sebelumnya bertujuan untuk melihat performa dari algoritma yang digunakan pada penelitian tugas akhir ini. Selain itu, perbadingan dengan penelitian sebelumnya dilakukan untuk mengetahui kesesuaian perhitungan nilai penalti pada penelitian tugas akhir ini. Pada penelitian sebelumnya menggunakan algoritma *Variabel Neighbourhood Descent* (VND) sedangkan pada penelitian tugas akhir ini menggunakan algoritma *Reinforcement Learning – Simulated Annealing with Reheating* (*RL-SAR*). Perbandingan dilakukan pada unit rumah sakit yang memiliki jadwal kerja yang *feasible* yaitu unit rumah sakit Optur 4, unit rumah sakit Optur 5, dan unit rumah sakit Optur 7. Perbandingan hasil optimasi dengan penelitian sebelumnya dapat dilihat pada Tabel 6.11 berikut.

Tabel 6.11 Perbandingan Hasil Optimasi dengan Penelitian Sebelumnya

Unit Rumah Sakit	Indikator	*RL-SAR*	*VND*
Optur 4	Penalti awal	628.419	203
	Penalti akhir	146.835	2.48
	Peningkatan (%)	76.634	98.778
	Running Time (s)	3076	1200
Optur 5	Penalti awal	241.984	253
	Penalti akhir	44.715	8.34
	Peningkatan (%)	81.522	96.703
	Running Time (s)	352	1200
Optur 7	Penalti awal	720.167	378
	Penalti akhir	667.291	156
	Peningkatan (%)	7.342	58.730
	Running Time (s)	231	1200

Berdasarkan Tabel 6.11 dapat dilihat bahwa hasil optimasi algoritma *RL-SAR* memiliki nilai penalti yang lebih besar daripada hasil optimasi penelitian sebelumnya dengan menggunakan algoritma *VND*. Hal tersebut dikarenakan pada

penelitian sebelumnya terdapat pembobotan nilai penalti (nilai k_m), sedangkan pada penelitian tugas akhir ini tidak melakukan pembobotan nilai penalti karena kurangnya informasi terkait bobot nilai penalti yang digunakan pada penelitian sebelumnya.

Selain itu, pada penelitian sebelumnya terdapat penyesuaian terhadap nilai penalti pelanggaran *soft constraint* 8 tentang pola sif yang diinginkan perawat. Penyesuaian yang dilakukan yaitu menghitung nilai penalti pelanggaran *soft constraint* 8 berdasarkan kemiripan pola sif pada jadwal kerja yang terbentuk dengan pola sif yang diinginkan perawat. Sebagai contoh, jika pola sif yang diinginkan perawat yaitu sif siang, sif pagi, sif pagi. Kemudian pada jadwal kerja yang terbentuk terdapat pola sif pagi, sif pagi, sif pagi, maka pola sif tersebut akan dihitung untuk meminimalisir nilai penalti pelanggaran *soft constraint* 8 dengan perhitungan tertentu. Sementara pada penelitian tugas akhir ini, untuk meminimalisir nilai penalti pelanggaran *soft constraint* 8 maka pola sif pada jadwal kerja yang terbentuk harus sama dengan pola sif yang diinginkan perawat. Pemenuhan terhadap pola sif tersebut sering kali melanggar *hard constraint* 5 tentang pasangan sif yang dilarang, sehingga nilai penalti pada penelitian tugas akhir ini cukup besar karena pelanggaran *soft constraint* 8.

Kemudian jika dilihat berdasarkan *running time* optimasi, pada penelitian sebelumnya menggunakan *running time* sebagai parameter batas optimasi. Hal tersebut dapat dilihat dari *running time* optimasi yang sama pada ketiga unit rumah sakit. Sementara itu, pada penelitian tugas akhir ini menggunakan jumlah iterasi sebagai parameter batas optimasi. Pada unit rumah sakit Optur 5 dapat dilihat bahwa dengan *running time* yang cukup jauh berbeda, algoritma *RL-SAR* mampu menghasilkan peningkatan solusi mendekati algoritma *VND*. Hal tersebut memungkinkan algoritma *RL-SAR* menghasilkan peningkatan solusi yang hampir sama dengan algoritma *VND* jika dijalankan dengan parameter batas optimasi yang sama.

6. 5. Ringkasan Hasil dan Pembahasan

Berdasarkan hasil percobaan yang telah dilakukan, parameter yang digunakan pada algoritma *RL-SAR* yaitu *epsilon decay greedy*, *sum*, *cooling rate* 0.99995, dan *reheating frequency* 50000. Algoritma *RL-SAR* juga memiliki performa yang cukup baik dalam menyelesaikan permasalahan penjadwalan perawat pada rumah sakit di Norwegia. Performa algoritma *RL-SAR* dapat dilihat dari nilai penalti yang dihasilkan, persebaran nilai penalti, dan kemampuan algoritma dalam menemukan nilai penalti yang lebih baik. Berdasarkan nilai penalti yang dihasilkan, algoritma *RL-SAR* mampu menghasilkan nilai penalti dengan peningkatan sebesar 3% - 47% dari algoritma *SR-HC*, 2% - 12% dari algoritma *RL-HC*, dan 0.7% - 7% dari algoritma *RL-SA*. Berdasarkan persebaran nilai penalti yang dihasilkan, algortima *RL-SAR* memiliki persebaran nilai penalti yang cenderung konstan daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Berdasarkan kemampuan algoritma dalam menemukan nilai penalti yang lebih baik, algoritma *RL-SAR* memiliki kemampuan yang lebih baik daripada algoritma *SR-HC*, *RL-HC*, dan *RL-SA*. Selanjutnya, kesimpulan pada peneltian tugas akhir dan saran untuk penelitian berikutnya akan dijelaskan pada bab 7.

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan dari hasil dan pembahasan pada bab 6. Pada bab ini juga akan membahas tentang saran yang dapat diberikan untuk pengembangan yang lebih baik.

7.1. Kesimpulan

Berdasarkan percobaan yang telah dilakukan, maka kesimpulan yang didapat pada tugas akhir ini adalah sebagai berikut:

1. Algoritma yang digunakan untuk menyusun solusi awal kurang generik sehingga hanya mampu menyusun jadwal kerja yang *feasible* untuk 3 unit rumah sakit.
2. Algoritma *RL-SAR* mampu menyelesaikan *Nurse Rostering Problem* pada rumah sakit di Norwegia dengan baik.
3. Algoritma *RL-SAR* dapat memberikan peningkatan solusi hingga 80% dari solusi awal.
4. Algoritma *RL-SAR* memiliki performa yang lebih baik daripada algoritma pembandingnya yaitu algoritma *SR-HC*, *RL-HC* dan *RL-SA*. Algoritma *RL-SAR* mampu menghasilkan solusi dengan peningkatan 3% hingga 47% lebih baik dari algoritma *SR-HC*, 2% hingga 12% lebih baik dari algoritma *RL-HC*, dan 0.7% hingga 7% lebih baik dari algoritma *RL-SA*. Selain itu performa algoritma juga dapat dilihat dari persebaran nilai penalti dan kemampuan algoritma dalam menemukan nilai penalti yang lebih baik.
5. Parameter yang digunakan pada tahap uji coba sangat mempengaruhi kemampuan algoritma dalam menemukan nilai penalti yang lebih baik.

7.2. Saran

Berdasarkan hasil dan kesimpulan, maka saran yang dapat diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Penelitian ini hanya dapat membentuk solusi awal yang *feasible* untuk 3 unit rumah sakit. Hal tersebut dikarenakan algoritma yang digunakan untuk menyusun solusi awal kurang generik. Pada penelitian selanjutnya dapat menggunakan algoritma yang lebih generik sehingga dapat membentuk solusi yang *feasible* untuk seluruh unit rumah sakit.
2. Penelitian ini menggunakan 3 skenario untuk masing-masing parameter algoritma *RL-SAR* dengan total sebanyak 12 skenario uji coba parameter. Pada penelitian selanjutnya, skenario uji coba parameter dapat dikembangkan dengan membuat skenario yang lebih bervariasi sehingga dapat menemukan parameter yang lebih tepat dan mendapatkan hasil yang lebih optimal.
3. Penelitian ini hanya menggunakan paramater *heuristic selection*, *utility update*, *cooling rate*, dan *reheating frequency* pada algoritma *RL-SAR*. Pada penelitian selanjutnya dapat menggunakan parameter yang lebih bervariasi seperti jumlah iterasi, suhu awal, *reheating limit* untuk mendapatkan kemungkinan solusi yang lebih baik lagi.

DAFTAR PUSTAKA

[1] "Optimisation in Health care - SINTEF." [Online]. Available: https://www.sintef.no/en/digital/applied-mathematics/optimization/health-care-optimization/#Personnelscheduling. [Accessed: 29-Oct-2019].

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

[3] Z. Liu, Z. Liu, Z. Zhu, Y. Shen, and J. Dong, "Simulated annealing for a multi-level nurse rostering problem in hemodialysis service," *Appl. Soft Comput. J.*, vol. 64, pp. 148–160, 2018.

[4] S. L. Goh, G. Kendall, and N. R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *Eur. J. Oper. Res.*, vol. 261, no. 1, pp. 17–29, 2017.

[5] Y. C. Fonseca Reyna, Y. Martínez Jiménez, J. M. Bermúdez Cabrera, and B. M. Méndez Hernández, "A reinforcement learning approach for scheduling problems," *Investig. Operacional*, vol. 36, no. 3, pp. 225–231, 2015.

[6] M. A. Lopes Silva, S. R. de Souza, M. J. Freitas Souza, and A. L. C. Bazzan, "A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems," *Expert Syst. Appl.*, vol. 131, pp. 148–171, 2019.

[7] M. L. Pinedo, *Scheduling Theory, Algorithms, and Systems (5th ed.)*. New York: Springer Science+Business Media, 2016.

[8] S. Petrovic and G. Vanden Berghe, "Comparison of algorithms for nurse rostering problems," *7th Int. Conf.*

Pract. Theory Autom. Timetabling, PATAT 2008, vol. 44, no. 0, pp. 1–18, 2008.

[9] E. Burke, P. Causmaecker, G. Vanden Berghe, and H. Van Landeghem, "The state of the art of nurse scheduling," *J. Sched.*, vol. 7, no. 6, pp. 441–499, 2004.

[10] "Applied Research,Technology and Innovation - SINTEF." [Online]. Available: https://www.sintef.no/en/this-is-sintef/. [Accessed: 29-Oct-2019].

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

[12] C. T. Greenwood, "Simulated Annealing Algorithms: An Overview," no. x, pp. 102–104, 1986.

[13] D. Henderson, S. H. Jacobson, and A. W. Johnson, *The Theory and Practice of Simulated Annealing*, no. April. 2006.

[14] E. K. Burke and G. Kendall, *Search Methodologies*. New York: Springer Science+Business Media, 2014.

[15] M. Melnik and D. Nasonov, "Workflow scheduling using Neural Networks and Reinforcement Learning," *Procedia Comput. Sci.*, vol. 156, pp. 29–36, 2019.

[16] E.-G. Talbi, *Metaheuristics from Design to Implementation*. New Jersey: John Wiley & Sons, Inc., 2009.

[17] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: Revisited," *Int. Ser. Oper. Res. Manag. Sci.*, vol. 272, pp. 453–477, 2019.

[18] N. Pillay and R. Qu, *Hyper- Heuristics : Theory and Applications*. Switzerland: Springer Nature Switzerland, 2018.

LAMPIRAN A. HASIL SOLUSI AWAL

A-1. KODE PROGRAM PROSES IMPLEMENTASI

Seluruh kode program proses implementasi dapat dilihat pada https://github.com/dimaskusuma99/tugasAkhir

A-2. HASIL SOLUSI AWAL PADA UNIT RUMAH SAKIT OPTUR 5

ID Perawat	Hari				
	1	2	3	4	5
483100659	<Free>	E2	N	<Free>	<Free>
483102571	D2	E	E2	D	N
483116704	D	E	<Free>	D1	<Free>
483132223	E2	<Free>	D9	E2	E
483145632	N	<Free>	D8	D	<Free>
483147489	D1	D1	<Free>	D3	D
483159693	E	D8	D1	<Free>	D3
483160705	<Free>	N	<Free>	N	<Free>
483171951	E	<Free>	E	<Free>	E2
483174037	<Free>	D	D	D	E
483178925	<Free>	E2	<Free>	D8	D1
483179088	D3	<Free>	E	E2	D8
483181232	<Free>	D3	D	E	D2
483182045	<Free>	N	<Free>	N	<Free>
483185702	N	<Free>	N	<Free>	D
483187661	D8	<Free>	D3	D9	<Free>
483193673	D	D9	E2	D2	<Free>
483195628	<Free>	D	<Free>	E	E2
483196167	E2	D2	D2	<Free>	D
483197481	D	D	D	<Free>	N

ID Perawat	Hari				
	6	7	8	9	10
483100659	<Free>	<Free>	E	N	<Free>
483102571	<Free>	<Free>	<Free>	E	D
483116704	<Free>	<Free>	<Free>	D3	N
483132223	D	D2	N	N	N
483145632	<Free>	<Free>	<Free>	<Free>	<Free>
483147489	N	N	<Free>	D	D
483159693	<Free>	<Free>	D2	D2	E2
483160705	D	D	D	D	E
483171951	<Free>	<Free>	D	D1	D3
483174037	<Free>	<Free>	D	D	E2
483178925	<Free>	<Free>	D8	D9	D2
483179088	D2	E	E2	E2	<Free>
483181232	<Free>	<Free>	D3	<Free>	<Free>
483182045	<Free>	<Free>	<Free>	<Free>	D9
483185702	<Free>	<Free>	<Free>	<Free>	<Free>
483187661	<Free>	<Free>	N	<Free>	D1
483193673	<Free>	<Free>	<Free>	D8	D8
483195628	E	D	E2	E2	D
483196167	<Free>	<Free>	D1	E	<Free>
483197481	<Free>	<Free>	E	<Free>	E

ID Perawat	Hari				
	11	12	13	14	15
483100659	D9	<Free>	<Free>	<Free>	<Free>
483102571	D3	D2	D	D	E
483116704	N	<Free>	N	N	<Free>
483132223	<Free>	D3	<Free>	<Free>	E2
483145632	E	D8	<Free>	<Free>	D8

ID Perawat	Hari				
	11	12	13	14	15
483147489	E	E2	<Free>	<Free>	D
483159693	<Free>	E	D2	D2	N
483160705	N	N	<Free>	<Free>	E2
483171951	E2	<Free>	<Free>	<Free>	<Free>
483174037	<Free>	D	E	D	N
483178925	<Free>	<Free>	<Free>	<Free>	<Free>
483179088	D	N	<Free>	<Free>	E
483181232	D2	D	<Free>	<Free>	<Free>
483182045	D1	<Free>	<Free>	<Free>	<Free>
483185702	<Free>	D1	<Free>	<Free>	<Free>
483187661	<Free>	<Free>	<Free>	<Free>	D1
483193673	E2	E	<Free>	<Free>	D3
483195628	D	E2	<Free>	<Free>	D
483196167	D8	D	<Free>	<Free>	D2
483197481	D	<Free>	D	E	D

ID Perawat	Hari				
	16	17	18	19	20
483100659	D3	E	D2	<Free>	N
483102571	D	N	N	<Free>	<Free>
483116704	E	E	<Free>	D2	<Free>
483132223	D	D1	D9	D8	<Free>
483145632	E2	D2	E	<Free>	D
483147489	N	N	<Free>	D	<Free>
483159693	N	<Free>	D1	N	<Free>
483160705	<Free>	D	D	E2	<Free>
483171951	E2	E2	<Free>	<Free>	D
483174037	<Free>	D	<Free>	E	<Free>

ID Perawat	Hari				
	16	17	18	19	20
483178925	D8	<Free>	<Free>	E2	<Free>
483179088	D1	D	E2	E	<Free>
483181232	<Free>	<Free>	D	D	E
483182045	D9	D9	N	<Free>	<Free>
483185702	<Free>	<Free>	<Free>	D3	<Free>
483187661	<Free>	D8	D3	D1	<Free>
483193673	<Free>	<Free>	E2	<Free>	<Free>
483195628	D	E2	E	N	<Free>
483196167	D2	D3	D8	<Free>	D2
483197481	E	<Free>	D	D	<Free>

ID Perawat	Hari				
	21	22	23	24	25
483100659	N	<Free>	E2	D	D2
483102571	<Free>	N	N	<Free>	D
483116704	<Free>	D	<Free>	D2	D3
483132223	<Free>	N	N	<Free>	E2
483145632	E	<Free>	D	D8	E
483147489	<Free>	D2	D1	D3	D
483159693	<Free>	E	<Free>	<Free>	E2
483160705	<Free>	D3	E2	E	E
483171951	D	E2	E	E	<Free>
483174037	<Free>	D	E	D	N
483178925	<Free>	<Free>	<Free>	D1	<Free>
483179088	<Free>	D	D8	E2	D
483181232	D	E2	D9	N	<Free>
483182045	<Free>	<Free>	D2	<Free>	<Free>
483185702	<Free>	<Free>	<Free>	<Free>	D8

ID Perawat	Hari				
	21	22	23	24	25
483187661	<Free>	<Free>	<Free>	<Free>	<Free>
483193673	<Free>	<Free>	D3	E2	D1
483195628	<Free>	D8	D	N	<Free>
483196167	D2	D1	<Free>	D9	D9
483197481	<Free>	E	D	D	N

ID Perawat	Hari		
	26	27	28
483100659	E	<Free>	<Free>
483102571	<Free>	<Free>	<Free>
483116704	<Free>	<Free>	<Free>
483132223	D	<Free>	<Free>
483145632	E2	<Free>	<Free>
483147489	D	<Free>	<Free>
483159693	<Free>	<Free>	<Free>
483160705	D	<Free>	<Free>
483171951	D2	<Free>	<Free>
483174037	N	<Free>	<Free>
483178925	<Free>	D	E
483179088	E2	<Free>	<Free>
483181232	E	<Free>	<Free>
483182045	<Free>	E	D2
483185702	D8	N	N
483187661	<Free>	D2	D
483193673	D1	D	D
483195628	D3	<Free>	<Free>
483196167	<Free>	<Free>	<Free>
483197481	N	<Free>	<Free>

Halaman ini sengaja dikosongkan

LAMPIRAN B. HASIL UJI COBA PARAMETER

B-1. HASIL UJI COBA PARAMETER *HEURISTIC SELECTION*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
GREEDY				
Optur 4	1	158.118	74.839%	2946
	2	145.238	76.888%	2879
	3	148.848	76.314%	2843
	4	151.861	75.834%	2873
	5	148.125	76.429%	3049
	6	166.605	73.488%	3242
	7	160.402	74.475%	3379
	8	152.253	75.772%	3141
	9	159.944	74.548%	3095
	10	160.561	74.450%	3112
Optur 5	1	48.514	79.952%	350
	2	51.662	78.651%	352
	3	53.576	77.860%	352
	4	51.594	78.679%	354
	5	48.030	80.152%	357
	6	48.621	79.907%	353
	7	47.111	80.531%	357
	8	45.389	81.243%	351
	9	49.241	79.651%	341
	10	53.036	78.083%	346
Optur 7	1	663.945	7.807%	228
	2	666.897	7.397%	230
	3	670.538	6.891%	234
	4	669.087	7.093%	238
	5	664.481	7.732%	227
	6	667.624	7.296%	238
	7	668.859	7.125%	227

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	8	670.152	6.945%	235
	9	671.126	6.810%	235
	10	667.294	7.342%	235
EPSILON GREEDY				
Optur 4	1	153.584	75.560%	2827
	2	146.022	76.764%	2852
	3	152.654	75.708%	2879
	4	162.416	74.155%	2847
	5	147.272	76.565%	2931
	6	160.894	74.397%	2888
	7	149.629	76.190%	2831
	8	145.643	76.824%	3689
	9	148.661	76.344%	4627
	10	151.265	75.929%	4276
Optur 5	1	47.312	80.448%	342
	2	55.296	77.149%	329
	3	53.071	78.068%	340
	4	48.271	80.052%	350
	5	58.652	75.762%	344
	6	49.536	79.529%	342
	7	50.374	79.183%	342
	8	45.663	81.130%	332
	9	44.390	81.656%	342
	10	48.809	79.829%	338
Optur 7	1	671.897	6.703%	231
	2	668.469	7.179%	227
	3	671.584	6.746%	225
	4	671.105	6.813%	239
	5	668.602	7.160%	225
	6	671.949	6.695%	226
	7	673.048	6.543%	258
	8	667.733	7.281%	228

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	9	671.674	6.734%	225
	10	671.275	6.789%	230
EPSILON DECAY GREEDY				
Optur 4	1	153.709	75.540%	3091
	2	151.530	75.887%	2953
	3	151.061	75.962%	3162
	4	150.528	76.047%	3030
	5	152.831	75.680%	2901
	6	152.395	75.749%	2828
	7	151.606	75.875%	2897
	8	149.171	76.263%	3249
	9	147.413	76.542%	2847
	10	148.270	76.406%	3825
Optur 5	1	47.571	80.341%	368
	2	52.064	78.485%	351
	3	48.954	79.770%	356
	4	47.250	80.474%	354
	5	46.740	80.685%	365
	6	46.073	80.961%	353
	7	45.995	80.993%	367
	8	51.048	78.904%	359
	9	45.766	81.087%	357
	10	44.927	81.434%	358
Optur 7	1	665.986	7.523%	261
	2	670.043	6.960%	228
	3	669.212	7.076%	234
	4	666.929	7.392%	238
	5	666.555	7.444%	237
	6	667.410	7.326%	233
	7	667.166	7.360%	236
	8	665.330	7.615%	236
	9	669.759	7.000%	237

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	10	668.186	7.218%	236

B-2. HASIL UJI COBA PARAMETER *UTILITY UPDATE*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
SUM				
Optur 4	1	153.709	75.540%	3091
	2	151.530	75.887%	2953
	3	151.061	75.962%	3162
	4	150.528	76.047%	3030
	5	152.831	75.680%	2901
	6	152.395	75.749%	2828
	7	151.606	75.875%	2897
	8	149.171	76.263%	3249
	9	147.413	76.542%	2847
	10	148.270	76.406%	3825
Optur 5	1	47.571	80.341%	368
	2	52.064	78.485%	351
	3	48.954	79.770%	356
	4	47.250	80.474%	354
	5	46.740	80.685%	365
	6	46.073	80.961%	353
	7	45.995	80.993%	367
	8	51.048	78.904%	359
	9	45.766	81.087%	357
	10	44.927	81.434%	358
Optur 7	1	665.986	7.523%	261
	2	670.043	6.960%	228
	3	669.212	7.076%	234
	4	666.929	7.392%	238
	5	666.555	7.444%	237
	6	667.410	7.326%	233

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	7	667.166	7.360%	236
	8	665.330	7.615%	236
	9	669.759	7.000%	237
	10	668.186	7.218%	236
AVERAGE				
Optur 4	1	152.562	75.723%	2740
	2	141.728	77.447%	2875
	3	146.625	76.668%	2869
	4	147.588	76.514%	2862
	5	143.324	77.193%	3211
	6	153.471	75.578%	2981
	7	155.151	75.311%	2886
	8	144.901	76.942%	2904
	9	142.369	77.345%	3411
	10	146.086	76.753%	3070
Optur 5	1	47.360	80.429%	363
	2	47.903	80.204%	356
	3	56.832	76.514%	358
	4	49.391	79.589%	442
	5	45.126	81.352%	367
	6	48.657	79.892%	343
	7	54.309	77.557%	354
	8	51.537	78.702%	343
	9	47.647	80.310%	325
	10	47.694	80.290%	331
Optur 7	1	669.209	7.076%	237
	2	665.536	7.586%	231
	3	667.546	7.307%	243
	4	673.160	6.527%	241
	5	663.788	7.829%	245
	6	671.925	6.699%	240
	7	672.617	6.603%	238

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	8	668.946	7.112%	232
	9	669.967	6.971%	232
	10	667.265	7.346%	231
DISCOUNT FACTOR				
Optur 4	1	263.113	58.131%	2562
	2	162.738	74.104%	3014
	3	149.579	76.198%	3005
	4	146.029	76.762%	3015
	5	156.504	75.096%	2907
	6	145.573	76.835%	2952
	7	152.175	75.784%	2940
	8	145.762	76.805%	3288
	9	152.842	75.678%	3147
	10	150.067	76.120%	2987
Optur 5	1	111.286	54.011%	413
	2	78.012	67.761%	331
	3	71.242	70.559%	327
	4	72.767	69.929%	325
	5	94.509	60.944%	418
	6	96.474	60.132%	347
	7	97.998	59.502%	358
	8	73.123	69.782%	318
	9	96.998	59.916%	355
	10	95.646	60.474%	344
Optur 7	1	665.600	7.577%	251
	2	671.225	6.796%	216
	3	697.721	3.117%	185
	4	695.180	3.470%	182
	5	691.546	3.974%	212
	6	667.641	7.294%	219
	7	669.016	7.103%	214
	8	715.585	0.636%	183
	9	666.735	7.419%	216

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	10	710.791	1.302%	179

B-3. HASIL UJI COBA PARAMETER *COOLING RATE*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
COOLING RATE = 0.5				
Optur 4	1	131.551	79.066%	2806
	2	139.274	77.837%	2908
	3	140.325	77.670%	2844
	4	136.591	78.264%	2923
	5	137.714	78.086%	2778
	6	138.083	78.027%	2737
	7	126.986	79.793%	2744
	8	126.883	79.809%	2773
	9	129.346	79.417%	2765
	10	145.237	76.888%	2766
Optur 5	1	66.729	72.424%	348
	2	86.519	64.246%	437
	3	70.988	70.664%	409
	4	75.958	68.610%	418
	5	68.290	71.779%	430
	6	79.503	67.146%	432
	7	76.116	68.545%	420
	8	75.358	68.858%	351
	9	69.856	71.132%	336
	10	77.589	67.936%	355
Optur 7	1	676.389	6.079%	252
	2	683.994	5.023%	256
	3	679.546	5.640%	248
	4	688.613	4.381%	233
	5	681.746	5.335%	261

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	6	687.222	4.575%	264
	7	682.173	5.276%	255
	8	673.222	6.519%	229
	9	681.859	5.319%	227
	10	682.718	5.200%	226
COOLING RATE = 0.85				
Optur 4	1	128.124	79.612%	2808
	2	140.482	77.645%	2775
	3	140.492	77.644%	2731
	4	134.731	78.560%	2737
	5	134.834	78.544%	2905
	6	141.745	77.444%	2873
	7	134.455	78.604%	2900
	8	140.982	77.566%	2950
	9	141.082	77.550%	2992
	10	137.165	78.173%	2828
Optur 5	1	67.112	72.266%	389
	2	67.273	72.199%	383
	3	72.512	70.034%	382
	4	64.583	73.311%	363
	5	76.043	68.575%	370
	6	69.567	71.252%	393
	7	82.480	65.915%	430
	8	72.273	70.133%	388
	9	69.555	71.256%	384
	10	79.918	66.974%	389
Optur 7	1	685.470	4.818%	308
	2	670.937	6.836%	241
	3	680.106	5.563%	232
	4	678.557	5.778%	231
	5	675.258	6.236%	231
	6	676.679	6.039%	229
	7	682.701	5.203%	226

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	8	689.031	4.323%	229
	9	687.407	4.549%	228
	10	685.101	4.869%	231
COOLING RATE = 0.99995				
Optur 4	1	153.709	75.540%	3091
	2	151.530	75.887%	2953
	3	151.061	75.962%	3162
	4	150.528	76.047%	3030
	5	152.831	75.680%	2901
	6	152.395	75.749%	2828
	7	151.606	75.875%	2897
	8	149.171	76.263%	3249
	9	147.413	76.542%	2847
	10	148.270	76.406%	3825
Optur 5	1	47.571	80.341%	368
	2	52.064	78.485%	351
	3	48.954	79.770%	356
	4	47.250	80.474%	354
	5	46.740	80.685%	365
	6	46.073	80.961%	353
	7	45.995	80.993%	367
	8	51.048	78.904%	359
	9	45.766	81.087%	357
	10	44.927	81.434%	358
Optur 7	1	665.986	7.523%	261
	2	670.043	6.960%	228
	3	669.212	7.076%	234
	4	666.929	7.392%	238
	5	666.555	7.444%	237
	6	667.410	7.326%	233
	7	667.166	7.360%	236

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	8	665.330	7.615%	236
	9	669.759	7.000%	237
	10	668.186	7.218%	236

B-4. HASIL UJI COBA PARAMETER *REHEATING FREQUENCY*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
REHEATING FREQUENCY = 10000				
Optur 4	1	140.103	77.706%	2896
	2	137.966	78.046%	2868
	3	141.199	77.531%	2843
	4	126.768	79.828%	2857
	5	155.339	75.281%	2787
	6	150.708	76.018%	2779
	7	127.799	79.663%	2793
	8	134.945	78.526%	2822
	9	136.874	78.219%	2825
	10	138.809	77.911%	2947
Optur 5	1	49.274	79.638%	344
	2	51.743	78.617%	349
	3	52.125	78.459%	339
	4	49.420	79.577%	340
	5	51.803	78.593%	338
	6	52.502	78.304%	343
	7	46.663	80.717%	340
	8	47.081	80.544%	337
	9	52.895	78.141%	350
	10	56.804	76.526%	339
Optur 7	1	670.272	6.928%	263
	2	666.386	7.468%	252
	3	667.351	7.334%	230

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	4	672.506	6.618%	232
	5	667.970	7.248%	237
	6	669.745	7.001%	227
	7	665.664	7.568%	227
	8	672.836	6.572%	227
	9	669.551	7.028%	226
	10	671.322	6.783%	230
REHEATING FREQUENCY = 50000				
Optur 4	1	153.709	75.540%	3091
	2	151.530	75.887%	2953
	3	151.061	75.962%	3162
	4	150.528	76.047%	3030
	5	152.831	75.680%	2901
	6	152.395	75.749%	2828
	7	151.606	75.875%	2897
	8	149.171	76.263%	3249
	9	147.413	76.542%	2847
	10	148.270	76.406%	3825
Optur 5	1	47.571	80.341%	368
	2	52.064	78.485%	351
	3	48.954	79.770%	356
	4	47.250	80.474%	354
	5	46.740	80.685%	365
	6	46.073	80.961%	353
	7	45.995	80.993%	367
	8	51.048	78.904%	359
	9	45.766	81.087%	357
	10	44.927	81.434%	358
Optur 7	1	665.986	7.523%	261
	2	670.043	6.960%	228
	3	669.212	7.076%	234
	4	666.929	7.392%	238

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	5	666.555	7.444%	237
	6	667.410	7.326%	233
	7	667.166	7.360%	236
	8	665.330	7.615%	236
	9	669.759	7.000%	237
	10	668.186	7.218%	236
REHEATING FREQUENCY = 100000				
Optur 4	1	157.091	75.002%	2923
	2	156.036	75.170%	2973
	3	155.408	75.270%	2907
	4	154.240	75.456%	3090
	5	164.159	73.877%	3087
	6	155.804	75.207%	3559
	7	153.396	75.590%	2951
	8	160.276	74.495%	2933
	9	162.710	74.108%	2823
	10	153.680	75.545%	2810
Optur 5	1	54.154	77.621%	360
	2	49.770	79.433%	353
	3	46.240	80.891%	344
	4	51.642	78.659%	350
	5	52.042	78.494%	352
	6	45.365	81.253%	336
	7	55.886	76.905%	347
	8	49.607	79.500%	348
	9	46.245	80.889%	337
	10	51.376	78.769%	348
Optur 7	1	666.585	7.440%	264
	2	666.442	7.460%	229
	3	669.622	7.019%	228
	4	670.185	6.940%	227
	5	670.228	6.934%	230

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	6	672.020	6.686%	231
	7	674.664	6.318%	227
	8	665.648	7.570%	229
	9	671.371	6.776%	230
	10	670.103	6.952%	229

Halaman ini sengaja dikosongkan

LAMPIRAN C. HASIL OPTIMASI

C-1. HASIL SOLUSI SETELAH OPTIMASI PADA UNIT RUMAH SAKIT OPTUR 5

ID Perawat	Hari				
	1	2	3	4	5
483100659	<Free>	<Free>	E2	D3	D2
483102571	E	D	D	D9	E2
483116704	N	N	<Free>	E2	<Free>
483132223	<Free>	D	D	<Free>	D
483145632	D1	D3	D2	E2	E
483147489	D3	<Free>	E	D	D8
483159693	<Free>	<Free>	<Free>	E	E
483160705	E2	E	<Free>	D	D
483171951	D8	<Free>	<Free>	E	<Free>
483174037	N	N	<Free>	D	D
483178925	<Free>	D8	D1	D8	<Free>
483179088	D	D	E	<Free>	E2
483181232	<Free>	<Free>	N	N	N
483182045	<Free>	E2	D	D2	<Free>
483185702	<Free>	<Free>	<Free>	<Free>	<Free>
483187661	E2	D9	D3	<Free>	<Free>
483193673	D	D1	E2	<Free>	<Free>
483195628	D	D2	D8	<Free>	D3
483196167	D2	E2	D9	D1	D1
483197481	E	E	N	N	N

ID Perawat	Hari				
	6	7	8	9	10
483100659	<Free>	<Free>	<Free>	<Free>	D2

ID Perawat	Hari				
	6	7	8	9	10
483102571	<Free>	<Free>	E	D	D
483116704	<Free>	<Free>	D8	D8	<Free>
483132223	D	D	N	N	<Free>
483145632	<Free>	<Free>	<Free>	E	D
483147489	N	N	<Free>	E2	E
483159693	<Free>	<Free>	E2	D1	D1
483160705	E	D2	D	<Free>	<Free>
483171951	<Free>	<Free>	<Free>	D	D
483174037	<Free>	<Free>	<Free>	E	N
483178925	<Free>	<Free>	D1	D2	<Free>
483179088	D	D	D	<Free>	E2
483181232	<Free>	<Free>	<Free>	E2	D9
483182045	<Free>	<Free>	<Free>	<Free>	<Free>
483185702	<Free>	<Free>	D	D3	E
483187661	<Free>	<Free>	D2	D	D3
483193673	<Free>	<Free>	E2	<Free>	E2
483195628	D2	E	N	N	<Free>
483196167	<Free>	<Free>	D3	D9	D8
483197481	<Free>	<Free>	E	<Free>	N

ID Perawat	Hari				
	11	12	13	14	15
483100659	D9	E2	<Free>	<Free>	<Free>
483102571	E2	<Free>	D	D	D
483116704	D2	D8	N	N	N
483132223	D	D	<Free>	<Free>	E2
483145632	D1	D	<Free>	<Free>	<Free>
483147489	D	D3	<Free>	<Free>	D8

ID Perawat	Hari				
	11	12	13	14	15
483159693	E	<Free>	D2	D2	N
483160705	<Free>	E	<Free>	<Free>	D
483171951	E	E2	<Free>	<Free>	<Free>
483174037	<Free>	D	D	D	<Free>
483178925	<Free>	<Free>	<Free>	<Free>	<Free>
483179088	D	D2	<Free>	<Free>	D1
483181232	D8	E	<Free>	<Free>	<Free>
483182045	<Free>	<Free>	<Free>	<Free>	E
483185702	E2	N	<Free>	<Free>	E2
483187661	<Free>	<Free>	<Free>	<Free>	<Free>
483193673	D3	D1	<Free>	<Free>	D2
483195628	N	N	<Free>	<Free>	E
483196167	<Free>	<Free>	<Free>	<Free>	D3
483197481	N	<Free>	E	E	D

ID Perawat	Hari				
	16	17	18	19	20
483100659	<Free>	<Free>	N	N	N
483102571	N	N	<Free>	<Free>	<Free>
483116704	<Free>	E	D8	D8	<Free>
483132223	E2	D	D9	D	<Free>
483145632	<Free>	<Free>	<Free>	<Free>	E
483147489	D8	D2	E	E	<Free>
483159693	N	<Free>	D3	D	<Free>
483160705	D9	N	N	N	<Free>
483171951	D	D8	<Free>	E	D
483174037	E	E	D	D	<Free>
483178925	<Free>	E2	<Free>	E2	<Free>

ID Perawat	Hari				
	16	17	18	19	20
483179088	D	D	E2	E2	<Free>
483181232	<Free>	E2	D2	D1	D2
483182045	D2	D1	D	<Free>	<Free>
483185702	E2	<Free>	<Free>	<Free>	<Free>
483187661	<Free>	<Free>	<Free>	<Free>	<Free>
483193673	D3	D	E2	<Free>	<Free>
483195628	E	D9	D1	D3	<Free>
483196167	D1	D3	E	<Free>	D
483197481	D	<Free>	D	D2	<Free>

ID Perawat	Hari				
	21	22	23	24	25
483100659	N	N	<Free>	D9	D
483102571	<Free>	E	N	N	<Free>
483116704	<Free>	<Free>	E	<Free>	<Free>
483132223	<Free>	D	D1	E	D
483145632	D	D8	E	N	<Free>
483147489	<Free>	<Free>	D	D	N
483159693	<Free>	D1	D	D1	E
483160705	<Free>	D	D	E	N
483171951	D	D2	<Free>	E2	D3
483174037	<Free>	E2	E2	D	D
483178925	<Free>	<Free>	<Free>	<Free>	D2
483179088	<Free>	D3	D3	D	E2
483181232	E	<Free>	<Free>	E2	D8
483182045	<Free>	<Free>	<Free>	<Free>	<Free>
483185702	<Free>	<Free>	<Free>	<Free>	<Free>
483187661	<Free>	E2	E2	<Free>	E2

ID Perawat	Hari				
	21	22	23	24	25
483193673	<Free>	<Free>	D2	D3	D9
483195628	<Free>	N	N	<Free>	D1
483196167	D2	E	D9	D2	<Free>
483197481	<Free>	D	D8	D8	E

ID Perawat	Hari		
	26	27	28
483100659	E	<Free>	<Free>
483102571	E	<Free>	<Free>
483116704	<Free>	<Free>	<Free>
483132223	D2	<Free>	<Free>
483145632	E2	<Free>	<Free>
483147489	N	<Free>	<Free>
483159693	<Free>	<Free>	<Free>
483160705	N	<Free>	<Free>
483171951	D	<Free>	<Free>
483174037	D	<Free>	<Free>
483178925	D3	E	E
483179088	E2	<Free>	<Free>
483181232	D1	<Free>	<Free>
483182045	D	D	D
483185702	<Free>	N	N
483187661	<Free>	D2	D2
483193673	<Free>	D	D
483195628	D8	<Free>	<Free>
483196167	<Free>	<Free>	<Free>
483197481	<Free>	<Free>	<Free>

C-2. HASIL OPTIMASI MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING – SIMULATED ANNEALING WITH REHEATING*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
Optur 4	1	141.390	77.501%	2884
	2	149.985	76.133%	3373
	3	144.453	77.013%	3054
	4	148.120	76.430%	3286
	5	149.634	76.189%	3171
	6	146.106	76.750%	2831
	7	145.326	76.874%	3053
	8	146.505	76.687%	3037
	9	146.459	76.694%	2863
	10	150.371	76.072%	3208
Optur 5	1	46.794	80.663%	423
	2	46.156	80.926%	351
	3	44.916	81.438%	340
	4	42.846	82.294%	344
	5	45.140	81.346%	346
	6	44.704	81.526%	346
	7	44.421	81.643%	342
	8	45.898	81.033%	341
	9	42.483	82.444%	340
	10	43.790	81.904%	344
Optur 7	1	666.993	7.384%	234
	2	667.654	7.292%	238
	3	667.253	7.348%	242
	4	665.628	7.573%	231
	5	669.874	6.984%	228
	6	667.040	7.377%	228
	7	669.964	6.971%	228
	8	668.936	7.114%	227
	9	665.495	7.592%	228

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	10	664.070	7.790%	229

C-3. HASIL OPTIMASI MENGGUNAKAN ALGORITMA *SIMLE RANDOM – HILL CLIMBING*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
Optur 4	1	213.869	65.967%	2100
	2	201.219	67.980%	1885
	3	202.748	67.737%	1918
	4	211.759	66.303%	1847
	5	216.842	65.494%	2026
	6	204.347	67.482%	2002
	7	194.641	69.027%	1973
	8	207.512	66.979%	2123
	9	199.568	68.243%	2041
	10	194.302	69.081%	2024
Optur 5	1	164.003	32.226%	226
	2	158.648	34.439%	213
	3	161.075	33.436%	214
	4	159.729	33.992%	216
	5	151.134	37.544%	223
	6	139.429	42.381%	227
	7	157.377	34.964%	228
	8	153.273	36.660%	227
	9	150.483	37.813%	220
	10	177.686	26.571%	228
Optur 7	1	686.751	4.640%	169
	2	688.392	4.412%	160
	3	697.644	3.127%	154
	4	702.592	2.440%	159
	5	693.817	3.659%	146

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	6	681.171	5.415%	154
	7	699.725	2.839%	153
	8	684.440	4.961%	153
	9	679.926	5.588%	164
	10	691.290	4.010%	167

C-4. HASIL OPTIMASI MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING – HILL CLIMBING*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
Optur 4	1	132.859	78.858%	2911
	2	145.031	76.921%	2570
	3	138.420	77.973%	2563
	4	132.947	78.844%	2803
	5	145.521	76.843%	2906
	6	130.353	79.257%	2649
	7	125.654	80.005%	2722
	8	147.660	76.503%	2833
	9	135.004	78.517%	2697
	10	149.861	76.153%	2788
Optur 5	1	69.602	71.237%	392
	2	85.166	64.805%	378
	3	76.979	68.188%	392
	4	64.848	73.202%	400
	5	62.340	74.238%	398
	6	71.062	70.633%	400
	7	78.202	67.683%	391
	8	80.447	66.755%	405
	9	82.398	65.949%	393
	10	65.692	72.853%	395
Optur 7	1	681.701	5.341%	253
	2	681.769	5.332%	265

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	3	678.981	5.719%	275
	4	685.701	4.786%	263
	5	682.944	5.169%	268
	6	695.333	3.448%	211
	7	680.722	5.477%	213
	8	688.328	4.421%	210
	9	682.852	5.181%	215
	10	678.362	5.805%	215

C-5. HASIL OPTIMASI MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING – SIMULATED ANNEALING*

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
Optur 4	1	143.694	77.134%	2995
	2	142.198	77.372%	2815
	3	138.053	78.032%	2784
	4	139.889	77.740%	2794
	5	150.986	75.974%	2920
	6	133.844	78.701%	2918
	7	143.188	77.215%	2914
	8	132.058	78.986%	2825
	9	143.828	77.113%	2815
	10	135.827	78.386%	3029
Optur 5	1	59.994	75.208%	397
	2	57.544	76.220%	406
	3	67.614	72.058%	355
	4	53.458	77.909%	354
	5	67.020	72.304%	352
	6	60.410	75.035%	352
	7	66.271	72.613%	350

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
	8	54.453	77.497%	351
	9	67.434	72.133%	352
	10	53.818	77.760%	349
Optur 7	1	679.412	5.659%	251
	2	668.801	7.133%	251
	3	670.240	6.933%	248
	4	671.856	6.708%	248
	5	672.540	6.613%	251
	6	674.345	6.363%	230
	7	670.334	6.920%	227
	8	670.630	6.879%	231
	9	671.396	6.772%	231
	10	679.187	5.690%	266

C-6. HASIL OPTIMASI MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING – SIMULATED ANNEALING WITH REHATING* (*COOLING RATE* 0.5)

Unit Rumah Sakit	Percobaan	Penalti Akhir	Perbaikan	*Running Time* (s)
Optur 4	1	133.005	78.835%	3139
	2	135.367	78.459%	3105
	3	130.386	79.252%	3102
	4	140.489	77.644%	4121
	5	132.819	78.865%	4025
	6	137.094	78.184%	4586
	7	136.696	78.248%	3967
	8	135.865	78.380%	3701
	9	136.893	78.216%	3757
	10	137.034	78.194%	4292

BIODATA PENULIS

Penulis lahir di Jombang pada tanggal 6 September 1998 dengan nama Dimas Rizal Kusuma Shindu. Penulis merupakan anak kedua dari 3 bersaudara. Penulis telah menyelesaikan pendidikan dasar di SD Negeri Pulo Lor 1 Jombang pada tahun 2011, kemudian penulis menyelesaikan pendidikan menengah pertama di SMP Negeri 1 Jombang pada tahun 2014, selanjutnya penulis menyelesaikan pendidikan menengah atas di SMA Negeri 3 Jombang pada tahun 2016. Setelah menyelesaikan pendidikan menengah atas, penulis melanjutkan pendidikan di Depatemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas – Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan mulai dari kepanitiaan hingga organisasi. Penulis pernah menjabat sebagai *Head of Information Media Department* pada organisasi Himpunan Mahasiswa Sistem Informasi. Selain itu penulis juga pernah bergabung pada BEM FTIK ITS sebagai staf *Student Resource Development*. Pada bidang akademik, penulis pernah menjadi asisten dosen pada mata kuliah Sistem Basis Data dan Sistem Enterprise. Selain itu, pada tahun 2019 penulis menjadi finalis pada perlombaan *Data Analyst Competition* (DAC) yang diselenggarakan oleh Universitas Padjajaran Bandung.

Pada tahun keempat, penulis memiliki ketertarikan pada bidang *data science* dan mengambil bidang minat Rekayasa Data dan Intelegensi Bisnis di Departemen Sistem Informasi. Penulis dapat dihubungi melalui email dimaskusuma99@gmail.com

Halaman ini sengaja dikosongkan