TUGAS AKHIR – IS184853

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING REINFORCEMENT LEARNING-LATE ACCEPTANCE ALGORITHM

ANANG FIRDAUS
NRP 05211540000109

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

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IS184853

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

ANANG FIRDAUS
NRP 05211540000109

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

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

FINAL PROJECT – IS184853

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING REINFORCEMENT LEARNING-LATE ACCEPTANCE ALGORITHM

ANANG FIRDAUS
NRP 05211540000109

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

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information and Communication Technology
Sepuluh Nopember Institut of Technology
Surabaya 2019

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

ANANG FIRDAUS
NRP. 05211540000109

Surabaya, Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi ER, S.T., M.Sc., Ph.D
NIP. 19761011 200604 2 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperolch Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Oleh :

ANANG FIRDAUS
NRP. 05211540000109

Disetujui Tim Penguji : Tanggal Ujian : 9 Juli 2019
Periode Wisuda : September 2019

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

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

Raras Tyasnurita S.Kom., MBA., Ph.D. **(Penguji II)**

Halaman ini sengaja dikosongkan

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

Nama Mahasiswa : Anang Firdaus
NRP : 05211540000109
Departemen : Sistem Informasi
Dosen Pembimbing : Ahmad Muklason, S.Kom., M.Sc., Ph.D

ABSTRAK

Sebuah organisasi terkadang membutuhkan solusi untuk permasalahan optimasi lintas domain. Permasalahan optimasi lintas domain merupakan permasalahan yang memiliki karakteristik berbeda, misalnya antar domain optimasi penjadwalan, rute kendaraan, bin packing, dan SAT. Optimasi tersebut digunakan untuk mendukung pengambilan keputusan sebuah organisasi. Dalam menyelesaikan permasalahan optimasi tersebut, dibutuhkan metode pencarian komputasi. Di literatur, hampir semua permasalahan optimasi dalam kelas NP-hard diselesaikan dengan pendekatan meta-heuristics. Akan tetapi meta-heuristic ini memiliki kekurangan, yaitu diperlukan parameter tunning untuk setiap problem domain yang berbeda. Sehingga pendekatan ini dirasa kurang efektif. Oleh karena itu diperlukan pendekatan baru, yaitu pendekatan hyper-heuristics. Metode hyper-heuristic merupakan metode pencarian komputasi approximate yang dapat menyelesaikan permasalahan optimasi lintas domain dengan waktu lebih cepat. Pada tugas akhir ini lintas domain permasalahan yang akan diselesaikan ada enam, yaitu satisfiability (SAT), one

dimensional bin packing, permutation flow shop, personnel scheduling, travelling salesman problem (TSP), dan vehicle routing problem (VRP). Dalam menigkatkan kinerja, tugas akhir ini menguji pengaruh dari adaptasi algoritma Reinforcement Learning (RL) sebagai seleksi LLH dikombinasikan dengan algoritma Late Acceptance sebagai move acceptance. Hasil dari penelitian ini menunjukan algoritma Reinforcement Learning – Late Acceptance mendapatkan skor persentase kinerja sebesar 80% yang diuji coba pada 30 variasi instance data dalam enam domain permasalahan dan RL-LA unggul 8% dibandingkan dengan SR-LA.

Kata Kunci: **Optimasi Lintas Domain,** ***Hyper-heuristic, High level heuristic, Reinforcement Learning, Late Acceptance***

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING REINFORCEMENT LEARNING-LATE ACCEPTANCE ALGORITHM

Name	**:**	**Anang Firdaus**
NRP	**:**	**05211540000109**
Department	**:**	**Information Systems**
Supervisior	**:**	**Ahmad Muklason, S.Kom., M.Sc., Ph.D**

ABSTRACT

An organization sometimes needs solutions to cross-domain optimization problems. Cross domain optimization problem is a problem that has different characteristics, for example between domains of optimization scheduling, vehicle routing problem, bin packing, and SAT. This optimization is used to support the decision making of an organization. In solving the optimization problem, a computational search method is needed. In the literature, almost all optimization problems in NP-hard classes are solved by the meta-heuristics approach. However, this meta-heuristic has its disadvantages, that is, it requires parameters tunning for each different domain problem. So this approach is considered less effective. Therefore a new approach is needed, namely the hyper-heuristics approach. The hyper-heuristic method is an approximate computational search method that can solve cross-domain optimization problems with faster time. In this final project, there are six problem domains to be solved, namely

satisfiability (SAT), one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman problem (TSP), and vehicle routing problem (VRP). In improving performance, this final project examines the effect of adaptation of the Reinforcement Learning (RL) algorithm as LLH selection combined with the Late Acceptance algorithm as a acceptance acceptance. The results of this study show that the Reinforcement Learning - Late Acceptance algorithm gets a performance percentage score of 80% which is tested on 30 variations of data instances in six problem domains and RL-LA is superior to 8% compared to SR-LA.

Keywords: Cross Domain Optimization, Hyper-heuristic, High level heuristic, Reinforcement Learning, Late Acceptance

KATA PENGANTAR

Alhamdulillahi robbil 'alamin, segala puji bagi Allah SWT atas limpahan nikmat, rahmat, petunjuk, dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan penelitian tugas akhir dengan judul, "**PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE***" yang menjadi salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih atas pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik materi maupun spiritual demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Orangtua dan saudara yang selalu memberikan segala bentuk dukungan serta doa sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Dosen Pembimbing, Ahmad Muklason, S.Kom., M.Sc., Ph.D terimakasih atas segala bimbingan, arahan, dukungan, ilmu, serta motivasi yang sangat bermanfaat bagi penulis.
3. Bapak Edwin Riksakomara, S.Kom., M.T dan Ibu Raras Tyasnurita S.Kom., MBA., Ph.D sebagai dosen penguji, terima kasih atas kritikan dan masukan yang bersifat membangun untuk peningkatan kualitas tugas akhir ini.

4. Ibu Mahendrawathi ER, S.T., M.Sc., Ph.D, selaku Ketua Departemen Sistem Informasi ITS, yang telah menyediakan fasilitas terbaik untuk kebutuhan penelitian mahasiswa.
5. Keluarga besar Lannister yang telah memberi dukungan kepada penulis untuk menyelesaikan tugas akhir ini.
6. Serta semua pihak yang terlibat dan membantu dalam pengerjaan tugas akhir ini yang belum mampu penulis sebutkan di atas.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih jauh dari kata sempurna. Oleh karena itu, penulis meminta maaf atas kesalahan yang dibuat dalam penulisan tugas akhir ini. Penulis membuka pintu selebar-lebarnya bagi pihak yang ingin memberikan kritik dan saran, dan penelitian selanjutnya yang ingin menyempurnakan karya dari tugas akhir ini. Akhir kata, semoga buku tugas akhir ini bermanfaat bagi seluruh pembaca.

Surabaya, Juli 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR PERSAMAAN

Halaman ini sengaja dikosongkan

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai pendahuluan tugas akhir. Bab ini terdiri dari latar belakang permasalahan, perumusan masalah, tujuan tugas akhir, manfaat tugas akhir, dan relevansi dari tugas akhir yang dikerjakan.

1.1. Latar Belakang

Permasalahan optimasi saat ini berkembang menjadi masalah yang kompleks karena besarnya jumlah input dataset dan batasan-batasan yang harus Terpenuhi dalam menentukan optimalitas [1]. Saat ini terdapat berbagai macam domain permasalahan optimasi kombinatorial, seperti permasalahan optimasi *vehicle routing problem* (VRP), *flow shop, bin packing*, dan penjadwalan. Sebuah organisasi terkadang juga membutuhkan banyak optimasi untuk mendukung bisnisnya. Sebagai contoh perusahaan yang bergerak dalam bidang logistik tidak hanya membutuhkan optimasi untuk satu domain, tetapi membutuhkan optimasi untuk banyak domain, yaitu optimasi penjadwalan sopir, rute kendaraan, dan pengemasan barang (*bin packing*) sehingga permasalahan ini dapat disebut dengan permasalahan lintas domain.

Dalam menyelesaikan banyak domain permasalahan optimasi dibutuhkan berbagai jenis metode atau algoritma pencarian komputasi. Metode atau algoritma pencarian dalam menyelesaikan masalah optimasi dikelompokkan menjadi dua kelompok, yaitu algoritma *exact* dan algoritma *approximate* [2]. Algoritma *exact* digunakan dalam menyelesaikan permasalahan

optimasi sederhana. Dalam menyelesaikan masalah sederhana, algoritma *exact* dapat menemukan solusi yang paling optimal dalam waktu singkat. Namun, pada permasalahan optimasi yang kompleks seperti TSP, rute kendaraan, dan penjadwalan dengan input dataset yang besar, algoritma *exact* belum mampu mencari solusi yang paling optimal dalam waktu singkat. Oleh karena itu, algoritma *approximate* seperti heuristik, *meta-heuristic, dan hyper-heuristic* digunakan sebagai pilihan dalam menyelesaikan permasalahan optimasi yang kompleks.

Algoritma *approximate* lebih mengutamakan pencarian yang cepat sehingga efisien daripada pencarian yang lengkap (*linear*). Hal tersebut menyebabkan hasil solusi yang ditemukan dengan metode ini bukanlah hasil yang paling optimal, tetapi cukup baik dan dapat diselesaikan dalam waktu singkat (*polynomial*).

Pengembangan metode heuristik adalah *meta-heuristic* yang menyelesaikan permasalahan optimasi secara spesifik berdasarkan karakteristik satu permasalahan tertentu [3]. Dalam menyelesaikan banyak permasalahan optimasi, *meta-heuristic* harus mendeskripsikan dan memodelkan setiap domain permasalahan, serta menemukan metode pencarian yang tepat dalam menyelesaikan permasalahan setiap domain secara spesifik. Penyelesaian permasalahan optimasi secara spesifik membutuhkan biaya yang mahal dan waktu yang lama karena jika ada domain permasalahan baru, metode tersebut harus memodelkan masalah dan memodifikasi metode pencarian dari awal. Penerapan metode metode tersebut dapat memperlambat proses pengambilan keputusan untuk dalam menemukan solusi optimal untuk banyak permasalahan optimasi yang berbeda

(lintas domain). Oleh karena itu, perlu suatu metode yang dapat menyelesaikan berbagai permasalahan optimasi secara *general* sehingga dapat mempercepat penyelesaian masalah dalam mencari solusi [4].

Hyper-heuristic adalah metodologi otomatis untuk mencari atau membuat heuristik dalam meningkatkan generalitas domain permasalahan dan instance berbagai domain permasalahan optimasi yang kompleks [3]. Walaupun *hyper-heuristic* dapat meningkatkan generalitas pencarian komputasi pada banyak domain permasalahan optimasi, tetapi hasil pencarian *hyper-heuristic* tidak selalu optimal untuk semua domain permasalahan [4]. Hal tersebut disebabkan oleh perbedaan karakteristik setiap domain permasalahan.

Strategi *high level heuristic* yang diusulkan, diuji coba pada *kerangka kerja Hyper-heuristics Flexible (HyFlex)*. Terdapat enam domain permasalahan kombinatorial pada framework HyFlex, yaitu *satisfiability (SAT), one dimensional bin packing, permutation flow shop, personel scheduling, travelling salesman problem (TSP), dan vehicle routing problem (VRP)* [5]. Berdasarkan latar belakang tersebut, maka penulis mengangkat tema dengan judul "Penyelesaian Permasalahan Optimasi Lintas Domain dengan Pendekatan *Hyper-heuristic* Menggunakan Algoritma *Reinforcement Learning - Late Acceptance*".

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, perumusan masalah yang diambil untuk tugas akhir ini, yaitu:

1. Bagaimana penerapan algoritma *Reinforcement Learning – Late Acceptance* (RL-LA) dengan menggunakan pendekatan *Hyper-heuristic*?
2. Bagaimana performa algoritma RL-LA yang diusulkan dalam menyelesaikan permasalahan optimasi lintas domain, khususnya jika dibandingkan dengan algoritma lainnya?

1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah sebagai berikut:

1. Algoritma dibangun dengan menggunakan bahasa pemrograman java.
2. Uji coba dilakukan pada enam domain permasalahan optimasi kombinatorial yang ada dalam *framework* HyFlex, yaitu SAT, *one dimensional bin packing, permutation flow shop, personnel scheduling*, TSP, dan VRP.
3. Pengembangan metode dilakukan dengan menggunakan *framework* HyFlex.

1.4. Tujuan Tugas Akhir

Tujuan yang akan dicapai dalam pengerjaan tugas akhir ini, yaitu:

1. Menerapkan algoritma RL-LA dengan pendekatan *hyper-heuristic*.
2. Melakukan analisa performa algoritma RL-LA.

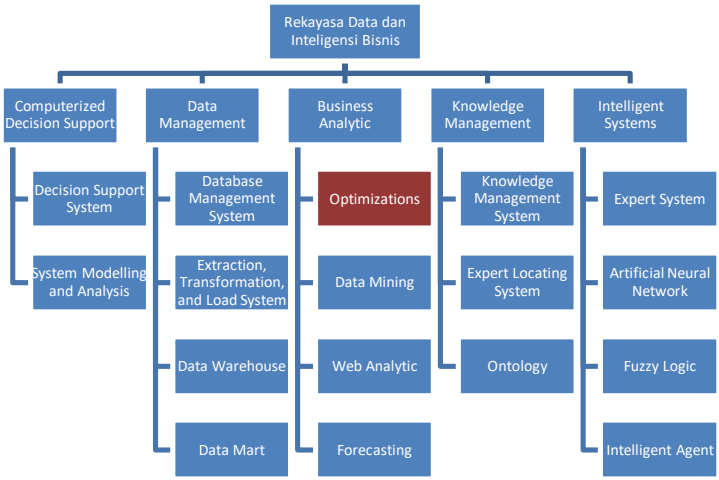
1.5. Manfaat Tugas Akhir

Manfaat yang didapatkan dengan adanya tugas akhir ini, yaitu :

1. Tugas akhir ini diharapkan dapat menambah wawasan dan pengetahuan dalam bidang keilmuan sistem informasi khususnya terkait dengan pengembangan metode *hyper-heuristic* RL-LA pada permasalahan lintas domain dari HyFlex.
2. Tugas akhir ini diharapkan dapat menghasilkan algoritma yang dapat digunakan dalam permasalahan permasalahan lintas domain.
3. Tugas akhir ini diharapkan dapat menghasilkan metode *hyper-heuristic* yang dapat menyelesaikan permasalahan optimasi *non-deterministic polynomial* (NP-hard) secara optimal.

1.6. Relevansi

Penelitian Tugas akhir ini berkaitan dengan mata kuliah Optimasi Kombinatorial dan Heuristic yang tercakup dalam laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB). Topik yang diangkat penulis dalam Tugas akhir ini adalah Optimasi, yang termasuk dalam bidang keilmuan laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB).



Gambar 1.1 Bidang Keilmuan Laboratorium Rekayasa Data dan Inteligensi Bisnis

BAB II
TINJAUAN PUSTAKA

Sebelum melakukan pengerjaan tugas akhir, penulis melakukan tinjauan pustaka dari beberapa penelitian sebelumnya dan dasar teori yang sesuai dengan topik tugas akhir.

2.1. Studi Sebelumnya

Dalam penyusunan Tugas Akhir, terdapat beberapa penelitian terkait yang sebelumnya telah dilakukan oleh pihak lain. Adapun hasil – hasil penelitian tersebut akan dijadikan sebagai referensi dalam penyusunan Tugas Akhir dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Penelitian Terdahulu	
	Nama Peneliti	Ahmad Mukhlason, Nisa Dwi Angresti, Arif Djunaidy
	Tahun Penelitian	2019
	Judul Penelitian	Penyelesaian Permasalahan Optimasi Lintas Domain Dari Hyflex Menggunakan *Hyper-heuristic* Yang Didasarkan Pada Metode *Variable Neighborhood Search*
	Penjelasan Singkat	Penelitian membahas permasalahan HyFlex dengan mengusulkan penyelesaian permasalahan dengan

No	Penelitian Terdahulu	
		metode gabungan variable neighborhood search – simulated annealing.[6]
	Hasil Penelitian	Metode yang di usulkan oleh peneliti *variable neighborhood search – simulated annealing* memberikan hasil yang paling baik pada penyelesaian *framework* HyFlex di 6 domain permasalahan dibandingkan dengan beberapa algoritma seperti *hill climbing, simulated annealing,* dan *variable neighborhood search – hill climbing.*
	Nama Peneliti	Jackson et al
	Tahun Penelitian	2013
	Judul Penelitian	*Late acceptance-based selection hyper-heuristics for cross-domain heuristic search*
	Penjelasan Singkat	Penggunaan metode *simple random* dalam pemilihan *low level heuristic* dengan berdasarkan metode *late acceptance*. Selain itu, peneliti juga memperkenalkan kelas baru metode pemilihan heuristik berdasarkan

No	Penelitian Terdahulu	
		pilihan *roulette-wheel* dan menggabungkannya dengan metode move acceptance *Late Acceptance* [7].
	Hasil Penelitian	Metode *simple random – late acceptance* memberikan nilai yang baik apabila diterapkan pada memori dengan jumlah yang besar.

2.2. Dasar Teori

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

2.2.1. *Meta-heuristic*

Meta-heuristic merupakan algoritma yang dapat menyelesaikan masalah optimasi kompleks jika diselesaikan dengan algoritma eksak. Metode heuristik adalah metode yang digunakan untuk mencari solusi suatu masalah dimana solusi yang ditemukan merupakan *feasible solution* yang terbaik [8]. Dalam pencarian solusi yang efisien dan komprehensif, metode *meta-heuristic* menggunakan mekanisme yang meniru perilaku sosial ataupun strategi yang ada di alam.

Algoritma *meta-heuristic* memiliki kecepatan pencarian solusi optimal yang lebih baik daripada metode tradisional [9]. Metode ini juga memberikan hasil yang lebih baik dibanding metode heuristik karena metode ini akan selalu berusaha untuk keluar dari solusi local

optima. Meskipun tidak ada jaminan bahwa solusi yang ditemukan merupakan solusi yang optimal, metode *meta-heuristic* yang dibangun dengan baik dapat memberikan solusi yang mendekati solusi optimal [8].

2.2.2. *Reinforcement Learning*

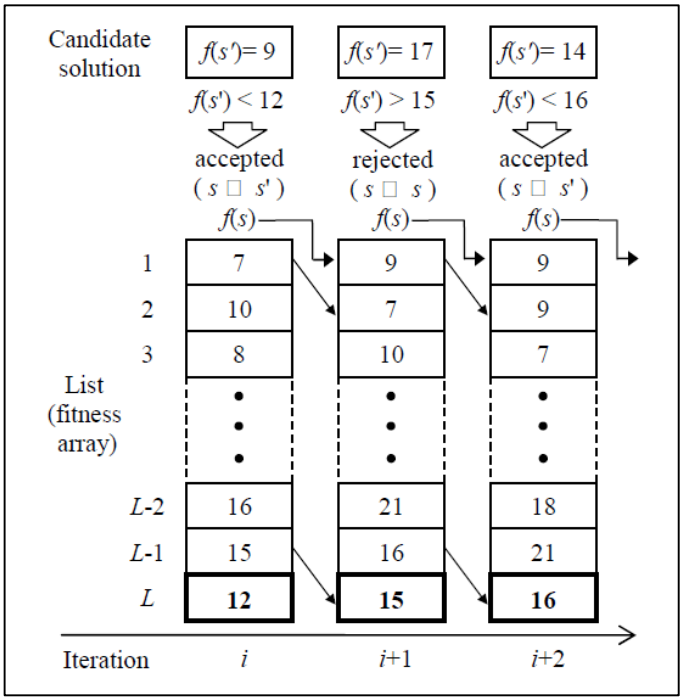
Reinforcement Learning (RL) merupakan sub area *machine learning* yang berfokus kepada cara sebuah *agent* mengambil aksi di lingkungannya. Di sini *agent* melakukan maksimalisasi tentang *reward* untuk jangka panjang [10]. Pada setiap langkah, RL memilih beberapa aksi yang mungkin dilakukan dan menerima reward dari lingkungan atas aksi spesifik yang dilakukannya. Aksi terbaik yang harus dilakukan di beberapa *state* tidak pernah diketahui sehingga agent harus mencoba beberapa aksi-aksi dan urutan-urutan aksi yang berbeda serta belajar dari pengalamannnya [11].

Berbeda dengan tipe pembelajaran lainnya, RL lebih berfokus kepada *goal-directed learning* dan interaksi yang dilakukan dengan lingkungan. *Feedback* yang diberikan kepada *agent* bersifat evaluatif, sedangkan pada tipe pembelajaran lain, misalnya pada *supervised learning feedback* yang diberikan bersifat instruksional. *Agent* RL tidak diberitahu aksi mana yang benar maupun salah melainkan diberikan *reward signal* sesuai dengan aksi yang dilakukan, yang menandakan seberapa baik aksi yang telah diambil tersebut. Karenanya dibutuhkan eksplorasi dan pembelajaran yang bersifat *trial-and-error* agar Agent dapat bertindak sesuai dengan yang seharusnya.

2.2.3. *Late Acceptance Strategy*

Late Acceptance Strategy (LAS) adalah metode baru dalam metaheuristic yang memiliki strategi membandingkan antara kandidat solusi dengan salah satu solusi yang telah muncul pada beberapa iterasi sebelumnya [12]. Hal tersebut menjadi pembeda antara LAS dengan metode pencarian metaheuristic lainnya seperti Simulated Annealing ataupun Hill Climbing yang memiliki strategi untuk membandingkan kandidat solusi dengan solusi terakhir secara langsung. Namun LAS termasuk dalam kelompok teknik pencarian perulangan meskipun memiliki tingkat mekanisme penerimaan yang lebih tinggi.

Dapat dilihat pada Gambar 2.1. bahwa kandidat solusi pada iterasi ke-i dan ke-(i+2) dapat diterima karena memiliki nilai yang lebih kecil dibandingkan dengan solusi pada urutan ke-L, sehingga kandidat solusi tersebut ditambahkan ke dalam daftar. Sedangkan untuk iterasi ke-(i+1), kandidat solusinya ditolak karena memiliki nilai yang lebih tinggi dibandingkan dengan solusi pada urutan ke-L, yang menyebabkan nilai solusi terakhir bernilai sama dengan nilai solusi sebelumnya.



Gambar 2.1 Ilustrasi iterasi pada eksekusi LAS

2.2.4. *Boolean Satisfiability Problem*

Boolean Satisfiability Problem (SAT) merupakan permasalahan yang menanyakan apakah formula logika boolean (proposisi) yang diberikan, dengan operasi boolean seperti AND, OR dan NOT, jika masing-masing variabelnya diberikan nilai true atau false dapat menghasilkan nilai akhir true.

SAT Problem telah menjadi masalah penting dalam ilmu komputer sejak Stephen Cook membuktikan bahwa penyelesaian permasalahan ini tergolong dalam Non Polynomial Complete (NP-Complete) pada tahun 1971 [13].

2.2.5. *Vehicle Routing Problem*

Vehicle routing problem (VRP) merupakan masalah penentuan rute kendaraan yang memegang peranan penting dalam dunia industri yaitu pada masalah manajemen logistik dan transportasi. VRP sebagai masalah penentuan rute optimal kendaraan dalam pendistribusian barang atau jasa dari satu atau lebih depot ke sejumlah pelanggan di lokasi yang berbeda dengan permintaan yang telah diketahui dan memenuhi sejumlah kendala. Tujuan umum dari VRP sendiri adalah :

1. meminimalkan jarak dan biaya tetap yang berhubungan dengan penggunaan kendaraan.
2. meminimalkan jumlah kendaraan yang dibutuhkan untuk melayani permintaan seluruh pelanggan.
3. menyeimbangkan rute-rute dalam hal waktu perjalanan dan muatan kendaraan.
4. meminimalkan pinalti sebagai akibat dari pelayanan yang kurang memuaskan terhadap pelanggan, seperti keterlambatan pengiriman dan lain sebagainya. [14]

2.2.6. *Personnel Scheduling*

Personnel Scheduling adalah pengalokasian sumber daya manusia pada stasiun kerja sesuai dengan kebutuhan, untuk meningkatkan produktivitas perusahaan harus menjadwalkan tenaga kerja secara optimal [15]. Penjadwalan yang baik dapat menentukan produktivitas tenaga kerja dalam melaksanakan pekerjaan, karena dapat menentukan dimana tenaga kerja harus bekerja dan beristirahat atau libur sehingga performa dan kesehatan tenaga kerja tetap terjaga.

2.2.7. *Permutation Flow Shop*

Permutation Flow Shop merupakan persoalan penjadwalan. Persoalan penjadwalan yang dimaksud adalah adalah persoalan pengalokasian pekerjaan ke mesin, pada kondisi mesin mempunyai kapasitas dan jumlah terbatas. Secara umum masalah penjadwalan dapat dijelaskan sebagai n job (J1, J2,…,Jn) yang harus diproses di m mesin (M1, M2,…,Mn) [16].

Waktu yang diperlukan untuk memproses 6 pekerjaan J1 pada mesin M adalah P setiap job harus diproses tanpa dihentikan selama waktu proses p mesin hanya dapat menangani satu job pada saat yang sama, dan secara terus menerus tersedia sejak waktu nol (time zero). Pemecahan permasalahan yang diinginkan adalah mendapatkan jadwal yang optimal, yaitu menyelesaikan semua pekerjaan dengan mendapatkan jadwal yang optimal yaitu menyelesaikan semua pekerjaan dengan adanya

keterbatasan kapasitas dan ketersediaan mesin dengan memenuhi fungsi tujuannya.

Tujuan penjadwalan adalah untuk mengurangi waktu keterlambatan dari batas waktu yang ditentukan agar dapat memenuhi batas waktu yang telah disetujui dengan konsumen, penjadwalan juga dapat meningkatkan produktifitas mesin dan mengurangi waktu menganggur. Produktifitas mesin meningkat maka waktu menganggur berkurang, secara tidak langsung perusahaan dapat mengurangi biaya produksi. Semakin baik suatu penjadwalan semakin menguntungkan juga bagi perusahaan dan bisa menjadi acuan untuk meningkatkan keuntungan dan strategi bagi perusahaan dalam kepuasan pelanggan.

2.2.8. *One Dimensional Bin Packing*

Bin Packing Problem merupakan suatu permasalahan kombinatorial (combinatorial problem) dan termasuk kedalam NP-Complete (non deterministic polynomial complete) yang membahas masalah bagaimana memaketkan barang-barang atau benda-benda yang ukurannya berbeda-beda kedalam suatu wadah atau bin yang memiliki ukuran yang tetap (tidak berubah) [17]. Sebagai contoh, diberikan sejumlah n benda dengan ukuran s1, s2, s3,…, sn, dan diberikan sejumlah bin atau wadah dengan ukuran m. Setiap benda harus ditempatkan kedalam bin sehingga besar total semua benda tidak melebihi setiap bin dan memberikan jumlah bin yang minimum.

2.2.9. *Travelling Salesman Problem*

Travelling Salesman Problem (TSP) merupakan permasalahan yang bertujuan untuk mencari jarak terdekat dalam mengunjungi masing–masing kota destinasi [18]. Destinasi kota yang dikunjungi akan membentuk sebuah pola atau rute tertentu hingga kota-kota destinasi tersebut dikunjungi tepat hanya satu kali hingga akhirnya kembali ke kota awal [19].

Penentuan rute tersebut menjadi salah satu faktor yang membuat TSP sulit untuk diselesaikan secara konvensional karena terdapat banyak sekali kemungkinan rute yang muncul dalam sebuah perjalanan. Singkatnya, TSP memberikan cara atau urutan dalam mengunjungi seluruh kota destinasi yang diinginkan. Contoh penerapan TSP yaitu Pak Pos mengambil surat di kotak pos yang tersebar pada n buah lokasi di berbagai sudut kota.

2.2.10. *Hyper-heuristic*

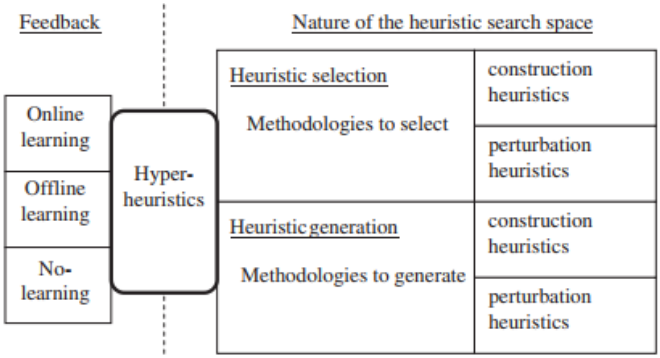
Hyper-heuristic adalah sebuah metode yang penerapan pencariannya dilakukan pada *level* heuristik, yaitu dengan melakukan pemilihan dan menggunakan heuristik secara optimal agar dapat memecahkan suatu permasalahan [4][20]. Pendekatan ini berbeda dengan pendekatan *meta-heuristic* biasa dimana ruang-pencariannya (search-space) berupa seluruh kemungkinan solusi, ruang pencarian hyperheuristic adalah semua kemungkinan heuristic yang dapat digunakan untuk memecahkan solusi [21]. *Hyper-*

heuristic melakukan proses pencarian pada domain permasalahan yang berbeda secara otomatis. Ide untuk mengotomatiskan desain heuristik telah ada pada tahun 1960, tetapi istilah *hyper-heuristic* baru diperkenalkan pada tahun 2000. *Hyper-heuristic* menggambarkan "heuristik untuk memilih heuristik" atau "heuristik untuk menghasilkan heuristik" [4].

Hyper-heuristic memberikan pendekatan yang memiliki tujuan untuk melakukan desain metode heuristik untuk menyelesaikan permasalahan komputasional yang rumit. *Hyper-heuristic* mendeskripsikan penggunaan heuristik untuk memilih heuristik lainnya pada kasus optimasi kombinatorial. Jadi, tujuan utama dari *hyper-heuristic* adalah untuk membuat desain metode umum, yang dapat memberikan solusi yang layak berdasarkan penggunaan dari LLH (*Low-Level Heuristic*).

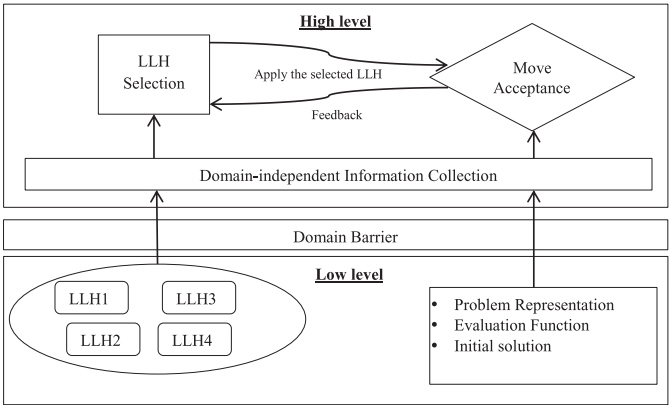
Hyper-heuristic terbagi menjadi dua dimensi klasifikasi, yaitu sifat ruang pencarian *hyper-heuristic* dan perbedaan *feedback* dari sumber informasi. Berdasarkan dimensi sifat ruang pencarian, *hyper-heuristic* terbagi lagi menjadi dua, yaitu seleksi dan generasi *hyper-heuristic*. Masing-masing seleksi dan generasi *hyper-heuristic* dibagi lagi berdasarkan sifat LLH sesuai dengan perbedaan antara paradigma pencarian konstruktif dan perturbatif. Konstruktif bekerja dengan mempertimbangkan solusi kandidat parsial (satu atau lebih komponen solusi hilang) dan secara iteratif membangun inisiasi solusi yang baru. Sementara itu, perturbatif mempertimbangkan inisiasi solusi lengkap

dan sudah ada dan mengubahnya dengan memodifikasi satu komponen solusi atau lebih.



Gambar 2.2 Klasifikasi Pendekatan *Hyper-heuristic* [20]

Hyper-heuristic terbagi menjadi 3 berdasarkan dimensi *feedback* sumber informasi pembelajaran, yaitu *online, offline*, dan *no-learning*. Dalam pembelajaran *hyper-heuristic online*, pembelajaran berlangsung ketika algoritma sedang menyelesaikan sebuah instance dari sebuah permasalahan, contohnya penerapan *reinforcement learning* pada seleksi heuristik, sedangkan dalam pembelajaran *hyper-heuristic offline*, mengumpulkan pengetahuan dalam bentuk peraturan atau program dari kumpulan *instance* pelatihan yang diharapkan akan menggeneralisasi untuk menyelesaikan kejadian yang tidak terlihat , contohnya penerapan *learning classifier system* dan *genetic programming* [22][3]. Sedangkan *no-learning* tidak menggunakan pembelajaran apapun.



Gambar 2.3 Struktur Umum *Hyper-heuristic*

Pada *high level heuristic* memiliki dua mekanisme, yaitu mekanisme seleksi LLH (*LLH selection*) dan mekanisme penerimaan solusi (*move acceptance*). Mekanisme seleksi LLH berfungsi menentukan LLH mana yang diterapkan untuk solusi pada tahap tertentu, sedangkan move acceptance menentukan apakah solusi diterima atau tidak. Sedangkan pada bagian *low level* berisi representasi dari permasalahan, fungsi evaluasi, dan kumpulan LLH dari permasalahan. LLH merepresentasikan lingkungan pencarian lokal sederhana (*move operator*) atau abstraksi dari aturan yang berasal dari ahli untuk membangun solusi (heuristik konstruktif). Selain itu terdapat domain barrier pada bagian tengah yang berfungsi sebagai letak informasi jumlah LLh dan fungsi evaluasi sehingga *hyper-heuristic* independent terhadap domain permasalahan. Kemudian LLH akan melakukan pencarian solusi [20][23][24]. Proses *hyper-heuristic* dimulai dari masukan domain permasalahan,

kemudian *high level heuristic* akan mencari LLH yang terbaik, dan solusi akan dihasilkan. Pemilihan LLH berhubungan dengan *pertubative* LLH dimana melakukan perubahan pada solusi saat ini. Oleh karena itu, metode pemilihan LLH dan *move acceptance* memiliki dampak yang besar terhadap nilai yang akan dihasilkan oleh *Hyper-heuristic* [25]. Beberapa metode dalam menyeleksi LLH adalah *Simple Random* dan *Reinforcement Learning*. Penjelasan metode tersebut dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh Metode dalam Seleksi LLH

Metode seleksi LLH	Deskripsi
Simple Random [7]	Melakukan pemilihan LLH secara acak pada saat proses pencarian
Reinforcement Learning [5]	Memberikan reward atau penalti kepada untuk mengukur kinerja LLH

2.2.11. *HyFlex*

HyFlex (Hyper-heuristics Flexible framework) adalah *software* kerangka kerja yang dirancang untuk memungkinkan pengembangan, pengujian, dan perbandingan algoritma pencarian heuristik tujuan umum iteratif seperti *hyper-heuristic*s. Kerangka kerja ini dibangun dengan menggunakan bahasa Java dan biasa digunakan oleh banyak peneliti [5].

HyFlex pertama kali digunakan untuk mendukung kompetisi penelitian internasional pada tahun 2011 dan dikenal sebagai CheSC (*Cross-Domain Heuristic Search Challenge*) 2011 [26].Di dalam Hyflex saat ini terdapat enam modul domain permasalahan yang diimplementasikan, yaitu *one dimensional bin packing, vehicle routing problem, permutation flow shop, personnel scheduling, traveling salesman problem,* dan *satisfiability.*

HyFlex telah ada pada Agustus 2010 saat peluncuran CheSC pada *International Conference on Practice and Theory of Automated Timetabling* (PATAT 2010). Saat ini telah banyak artikel implementasi *hyper-heuristic* yang dipublikasikan dengan HyFlex. *Cross-Domain Heuristic Search Challenge* bertujuan untuk melakukan pencarian dan pengoptimalan pada beberapa domain permasalahan [27]. Setiap domain permasalahan terdiri dari:

a. Rutinitas untuk menginisialisasi solusi secara acak dalam populasi.

b. Suatu rangkaian LLH untuk memodifikasi solusi:

    1. *Mutational*, memodifikasi solusi saat ini secara acak

    2. *Ruin-recreate*, menghancurkan solusi dan membangunnya kembali dengan prosedur konstruktif.

    3. *Local search*, mencari solusi berdasarkan neighborhood dari solusi terkini untuk mendapatkan hasil yang lebih baik.

4. *Crossover*, mengambil dua solusi dan menggabungkannya kemudian mengembalikan solusi baru.

c. Variasi *instance* yang mudah dipanggil.

d. Populasi dari satu atau banyak solusi.

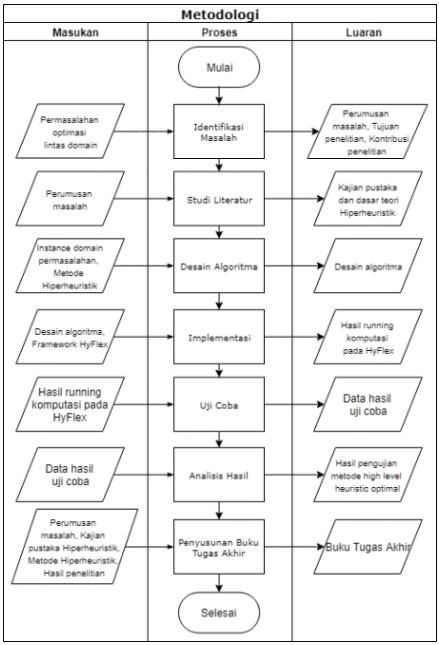
HyFlex terbagi menjadi dari tiga lapisan, yaitu *hyper-heuristic*, domain *barrier*, dan domain permasalahan. Lapisan *hyper-heuristic* menyeleksi kumpulan low level heuristic (LLH) dan menerapkan LLH pada solusi yang baru, setelah itu solusi baru akan disimpan ke dalam urutan solusi yang terdapat pada domain permasalahan. Lapisan domain barrier merupakan tantangan antara *hyper-heuristic* dan *low-level heuristic* yang independen terhadap informasi domain permasalahan. Lapisan domain permasalahan merupakan rumusan permasalahan optimasi yang spesifik Strategy (LAS) adalah metode baru dalam metaheuristic yang memiliki strategi membandingkan antara kandidat solusi dengan salah satu solusi yang telah muncul pada beberapa iterasi sebelumnya [12]. Hal tersebut menjadi pembeda antara LAS dengan metode pencarian metaheuristic lainnya seperti Simulated Annealing ataupun Hill Climbing yang memiliki strategi untuk membandingkan kandidat solusi dengan solusi terakhir secara langsung. Namun LAS termasuk dalam kelompok teknik pencarian perulangan meskipun memiliki tingkat mekanisme penerimaan yang lebih tinggi.

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai alur metodologi yang akan dilakukan dalam tugas akhir ini.

3.1. Metodologi Penelitian

Diagram metodologi pengerjaan Tugas Akhir dapat dilihat pada Gambar 3.1.



Gambar 3.1 Metodologi Pengerjaan Tugas Akhir

3.2. Tahapan Pelaksanaan Tugas Akhir

Tahapan pelaksanaan tugas akhir akan menjelaskan terkait segala sesuatu yang akan dikerjakan oleh penulis atau merupakan langkah-langkah pengerjaan tugas akhir.

3.2.1. Identifikasi Masalah

Tahap ini merupakan tahap untuk mendalami permasalahan yang diangkat dalam pengerjaan tugas akhir ini. Permasalahan yang dijadikan topik dalam pengerjaan tugas akhir ini yaitu permasalahan optimasi lintas domain yang akan diselesaikan dengan menerapkan algoritma *Reinforcement Learning – Late Acceptance* (RL-LA) dengan menggunakan pendekatan *Hyper-heuristic* pada framework Hyflex. Di dalam Hyflex saat ini terdapat enam modul domain permasalahan yang diimplementasikan, yaitu *one dimensional bin packing, vehicle routing problem, permutation flow shop, personnel scheduling, traveling salesman problem,* dan *satisfiability.*

3.2.2. Studi Literatur

Tahap ini bertujuan untuk memperdalam pengetahuan dan pemahaman terkait permasalahan yang akan diteliti. Studi literatur yang dilakukan mengacu kepada beberapa referensi untuk lebih memahami permasalahan. Studi literatur juga digunakan untuk melakukan pemilihan metode dan algoritma untuk menyelesaikan permasalahan lintas domain. Studi literatur mencakup konsep yang akan diterapkan dalam penelitian. Semua

hasil studi literatur berupa penelitian terdahulu serta dasar teori telah dicantumkan pada Bab II.

3.2.3. Desain Algoritma

Tahap ini merupakan tahap penggabungan antara *algoritma Reinforcement Learning* untuk melakukan seleksi LLH dan *Late Acceptence* (LA) sebagai penerima solusi yang terbaik sesuai dengan permasalahan yang telah di definisikan pada Bab II. Pada tahapan ini akan dijabarkan metode yang akan digunakan untuk melakukan seleksi LLH dan mekanisme penerimaan solusi.

3.2.4. Implementasi

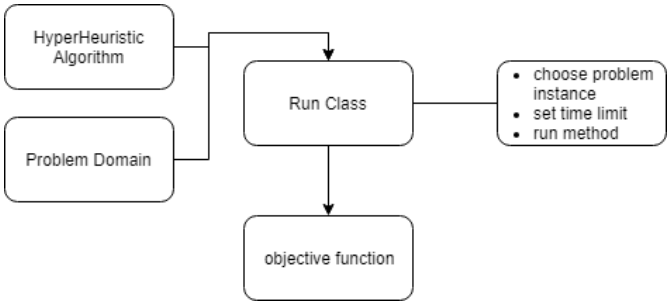
Langkah ini merupakan langkah implementasi desain algoritma yang sudah dibuat pada kerangka kerja HyFlex. Algoritma RL-LA yang telah didesain akan dibangun ke dalam HyFlex dalam bahasa pemrograman Java sehingga menjadi program yang siap digunakan untuk uji coba. Tahap ini dimulai dari persiapan *tools* hingga implementasi program. Mekanisme algoritma *Reinforcement Learning* secara sederhana memberikan *reward* dan *penalty* pada setiap LLH. Pada tahap awal, setiap LLH diberi skor awal yang sudah ditentukan. Jika LLH yang dipilih dan hasil solusinya dapat diterima, maka skor akan meningkat sampai skor mencapai batas yang ditentukan. Sebaliknya jika hasil ditolak, maka skor akan diturunkan hingga mencapai batas bawah. Pada setiap iterasi, LLH dengan skor tertinggi akan dipilih. Tetapi jika ada lebih dari satu LLH yang memiliki skor sama akan dipilih secara *random*. Untuk mekanisme

algoritma *Late Acceptence* sebagai move acceptance dasarnya sama seperti yang sudah dijelaskan pada bagian 2.2.3 yaitu dengan membandingkan pada beberapa iterasi sebelumnya.

3.2.5. Uji Coba

Tahap ini merupakan tahap pengimplementasian dan eksperimen dari implementasi algoritma *hyper-heuristic* yang dilakukan. Uji coba dilakukan untuk mengetahui kinerja dari metode *hyper-heuristic* yang diimplementasikan. Hasil dari uji coba digunakan untuk mencari strategi *high level heuristic* yang tepat untuk menyelesaikan masalah lintas domain. Tahap uji coba ini dilakukan pada kombinasi metode yang sesuai pada desain algoritma *hyper-heuristic*.

Pada gambar 3.2, algoritma *HyperHeuristic* yang telah dibuat dijalankan dengan memasukan beberapa data masukan yang diperlukan. Selanjutnya, kerangka kerja akan melakukan pencarian solusi berdasarkan data masukan yang telah didefinisikan. Setelah kriteria akhir dari *running* algoritma selesai, framework akan menampilkan nilai solusi terbaik yang dihasilkan dari proses pencarian solusi.



Gambar 3.2 Skenario Uji Coba

Uji coba algoritma tersebut dilakukan dengan menggunakan *framework* HyFlex. Uji coba dilakukan dengan beberapa ketentuan sebagai berikut:

a. Komputer yang digunakan memiliki *processor core* i5 8250U.

b. Program dapat dijalankan pada sistem Windows dan Linux (32 bit atau 64 bit). Program dijalankan dari *console*.

c. Penetapan parameter kondisi berhenti (alokasi waktu maksimum CPU, jumlah maksimum iterasi, jumlah iterasi antara dua peningkatan). Paramater ini digunakan untuk membatasi banyak pencarian dengan batasan waktu adalah 10 menit (60000 milidetik) untuk setiap *instance*. Kecepatan pencarian yang tepat dari komputer tergantung pada sejumlah faktor, termasuk memori, sistem operasi, dan kecepatan waktu.

d. Uji coba terhadap implementasi algoritma dieksekusi sebanyak 31 kali pada 30 *instance* domain.

Berikut adalah Java code untuk running hyper-heuristic pada suatu domain permasalahan :

```
1 ProblemDomain     problem     =     new
  SAT(seed1);
2 HyperHeuristic    HHObject    =     new
  ExampleHyperHeuristic1(seed2);
3 problem.loadInstance(0);
4 HHObject.setTimeLimit(60000);
5 HHObject.loadProblemDomain(problem);
6 HHObject.run();
7 System.out.println(HHObject.getBestSo
  lutionValue());
```

Kode Program 3.1 Java *Code* untuk *Running* Domain Permasalahan

Pada *line* pertama, digunakan untuk mengganti atau memilih domain permasalahan dengan memasukkanya pada *object* yang bernama *problem* (pada contoh ini adalah domain SAT). *Line* kedua digunakan untuk membuat *object* berisi algoritma yang sudah dibuat. Kemudian *line* ketiga digunakan untuk memanggil problem instance pada (pada contoh ini problem instance yang digunakan adalah index ke 0). *Line* keempat digunakan untuk mengatur batasan waktu yang digunakan untuk running selama 60 detik atau 60000 milisekon pada contoh ini. Lalu *line* kelima, memanggil domain permasalahan yang sudah dipilih. *Line* keenam digunakan untuk menjalankan program, kemudian *line*

terakhir digunakan untuk memunculkan hasi solusinya (output).

3.2.6. Analisis Hasil

Tahap analisis hasil dikerjakan setelah tahap implementasi dan uji coba pada HyFlex selesai. Analisis hasil digunakan untuk mengkur kinerja metode *hyper-heuristic*. Nilai fungsi fitness menjadi nilai patokan yang akan dibandingkan setelah dieksekusi secara berulang kali (31 kali) dari setiap instance pada setiap domain permasalahan (total 30 instance) dengan waktu 60000 milidetik (10 menit). Kriteria yang perlu diukur untuk membandingkan adalah nilai fungsi fitness terbaik (minimum), median, dan rata-rata, dan beberapa data statistik tambahan seperti nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimum untuk setiap 30 instance domain permasalahan.

Proses pengujian akan dilakukan dengan menggunakan sistem Formula One Point Scoring. Sistem ini menggunakan data median sebagai perbandingan hasil. Median pada masing – masing *fitness* pada setiap *instance* diberi poin, setelah itu point dari setiap *instance* dijumlahkan. Dalam sistem *Formula One*, performa delapan peringkat terbaik, diberikan poin 10, 8, 6, 5, 4, 3, 2, 1. Poin 0 akan diberikan kepada hasil yang melebihi dari 8 peringkat tersebut. Jika terdapat peserta dengan peringkat yang sama, maka peserta akan diberikan poin yang sama berdasarkan peringkatnya. Nantinya, poin setiap *instance* akan dibandingkan dengan algoritma pembanding

3.2.7. Penyusunan Buku Tugas Akhir

Tahap ini bertujuan untuk melaporkan seluruh hasil pekerjaan yang telah dilakukan dalam menyelesaikan permasalahan. Laporan ini mendokumentasikan rangkaian proses implementasi algoritma *reinforcement learning – late acceptance* dari formulasi fungsi tujuan hingga menemukan solusi akhir yang diinginkan. Luaran dari tahap ini yaitu Buku tugas akhir.

BAB IV
DESAIN

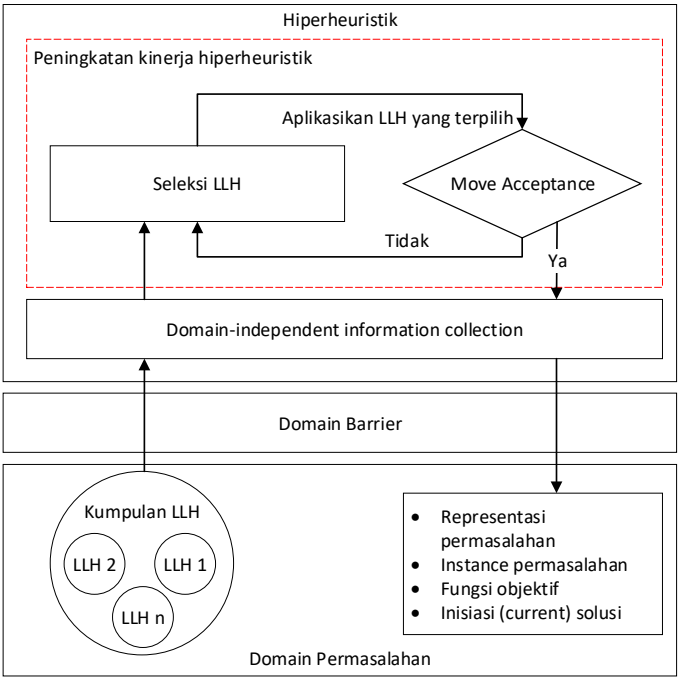
Desain pada penelitian ini merupakan tahapan untuk melakukan perancangan algoritma yang sesuai dengan identifikasi permasalahan seperti dijelaskan pada pendahuluan. Subbab berikut ini menjelaskan desain dan implementasi algoritma dalam penelitian ini.

4.1. Desain Hiperheuristik

Pada tahapan desain ini, perlu dilakukan perancangan algoritma untuk strategi *high level heuristic* yang tepat dalam penyelesaian permasalahan optimasi lintas domain dengan menggunakan pendekatan hiperheuristik. Desain algoritma ini didasarkan pada *framework* hiperheuristik yang terdiri dari dari dua *level* proses pencarian, yaitu *high level heuristic* dan kumpulan LLH.

Penyelesaian solusi pada permasalahan lintas domain, dilakukan secara bergantian, sesuai dengan *input* domain permasalahan yang digunakan. Tetapi, teknik pencarian hanya menggunakan satu metode saja untuk berbagai domain permasalahan optimasi. Pada dasarnya pencarian di dalam metode hiperheuristik dimulai dari solusi awal yang lengkap sebagai inisiasi solusi dan kemudian *high level heuristic* secara iteratif memilih LLH dari kumpulan LLH yang tersedia sampai kondisi berhenti terpenuhi, LLH terpilih akan membentuk solusi yang baru, solusi tersebut akan dipertimbangkan untuk diterima atau ditolak. Apabila diterima, maka solusi tersebut akan menjadi

inisiasi solusi pada iterasi berikutnya, tetapi apabila ditolak maka akan diulangi untuk mencari LLH. Strategi *high level heuristic* penelitian ini dapat pada Gambar 4.1.



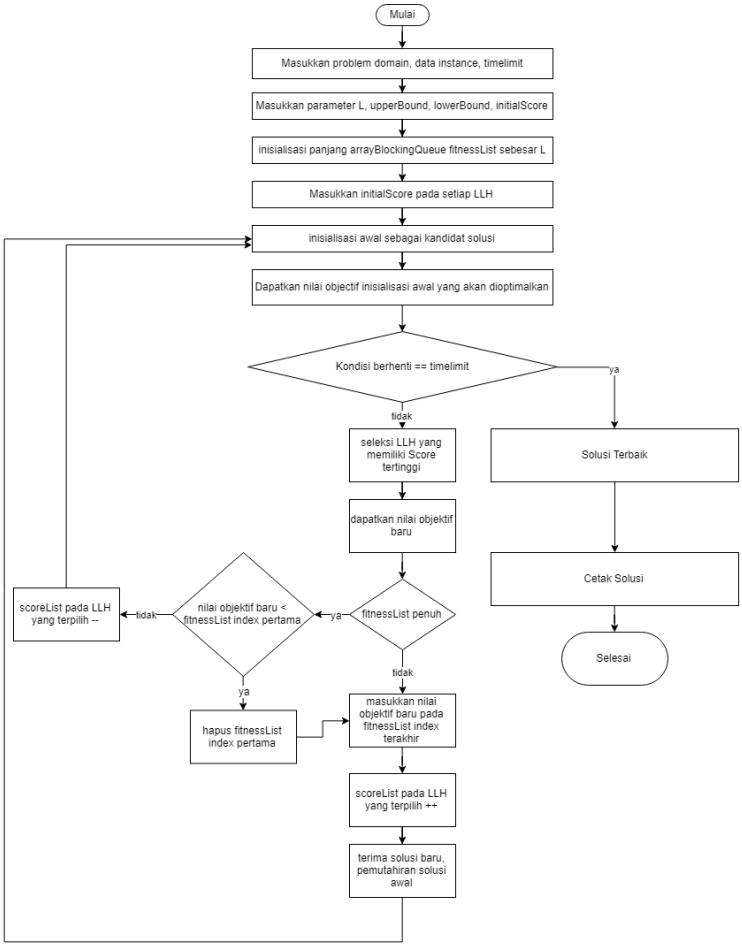
Gambar 4.1 Kerangka Usulan Strategi Hiperheuristik

Strategi *high level heuristic* merupakan gabungan dua mekanisme metode, yakni seleksi LLH dan *move acceptance* yang ditingkatkan dalam penelitian ini. Subbagian berikut ini menjelaskan proses strategi *high level heuristic* dan rangkaian LLH.

4.1.1. Desain *High Level Heuristic* RL-LA

Untuk membentuk strategi *high level heuristic* dilakukan dengan menggabungkan algoritma untuk seleksi LLH dengan *move acceptance* yang terdiri dari algoritma-algortima yang diadaptasi. Dalam penelitian ini, algoritma RL sebagai metode untuk seleksi LLH dikombinasikan dengan metode *move acceptance* LA.Desain algoritma RL-LA dapat dilihat pada Gambar 4.2.

Dalam desain algoritma ini, *Reinforcement Learning* dalam pencarian LLH dilakukan dengan memilih LLH yang memiliki score tertinggi dan setiap LLH yang terpilih akan membangkitkan satu solusi. Untuk *move acceptance*, algoritma *Late Acceptance* hanya membandingankan dengan solusi yang telah muncul pada L iterasi sebelumnya sehingga solusi yang buruk pun dapat diterima asalkan solusi tersebut lebih baik dari solusi yang muncul sebelumnya. Pada desain *high level heuristic* RL-LA ini dimulai dengan inisialisasi solusi untuk mendapatkan solusi awal. Inisialisasi solusi dilakukan bergantung terhadap domain permasalahan. Dari inisialisasi solusi ini didadapatkan nilai fungsi objektif yang menjadi perbandingan untuk pemilihan solusi selama iterasi.



Gambar 4.2 Desain High Level Heuristic RL-LA

4.1.2. Kumpulan LLH

Low level heuristic adalah sebuah heuristik yang berfungsi untuk membangkitkan atau membangun solusi dan mendapatkan nilai fungsi objektif dari solusi tersebut. Kumpulan LLH terdapat dalam setiap domain permasalahan. Dari kumpulan LLH tersebut, dipilih salah satu LLH yang akan digunakan untuk membangun solusi. LLH yang digunakan pada penelitian ini adalah LLH yang telah ada di dalam kerangka kerja HyFlex [5]. Pemakaian LLH yang ada pada HyFlex dilakukan dengan cara memanggil fungsi yang ada. Kumpulan LLH dalam HyFlex dikelompokkan menjadi empat kelompok, yaitu *ruin-recreate*, mutasi, *crossover*, dan *local search*.

Kelompok *ruin-recreate* perubahan pada solusi dilakukan dengan membongkar sebagian solusi dan membangun atau membuatnya kembali. Pada kelompok heuristik mutasi mengubah secara acak pada solusi dengan cara menukar, mengubah, membuang, menambah, atau menghapus komponen solusi. Kelompok heuristik *crossover* melakukan perubahan dengan mengambil dua solusi, menggabungkannya dan mengembalikan solusi baru. Pada kelompok heuristik *local search* secara iteratif membuat perubahan kecil pada solusi, dan hanya menerima solusi yang meningkat sampai optimum lokal ditemukan atau kondisi berhenti terpenuhi. Heuristik ini berbeda dari kelompok heuristik mutasi karena heuristik menggabungkan proses perbaikan berulang dan mereka menjamin bahwa solusi

yang baik akan dihasilkan. Tabel 4.1 merupakan jumlah LLH yang terdapat dalam HyFlex berdasarkan kelompok heuristik setiap domain permasalahan.

Tabel 4.1 Jumlah LLH pada Setiap Domain Permasalahan

No.	Domain Permasalahan	Mutation	Ruin-Recret	Crossover	Local Search	Total
1.	Satisfiability	6	1	2	2	11
2.	One Dimensional Bin Packing	3	2	1	2	8
3.	Permutation Flow Shop	5	2	3	4	15
4.	Personnel scheduling	1	3	3	4	12
5.	Traveling salesman Problem	5	1	3	2	13
6.	Vehicle Routing Problem	3	2	2	3	10

Berdasarkan table 4.1, dari setiap domain permasalahan memiliki berbagai macam LLH. LLH tergantung pada domain permasalahan. Domain permasalahan *satisfiability* (SAT) terdapat sebelas LLH [28]. Rincian LLH dalam SAT dapat dilihat pada Tabel 4.2.

Tabel 4.2 Kumpulan LLH pada Domain Permasalahan SAT

No	Nama LLH	Deskripsi	Tipe LLH
1.	GSAT	Variabel dengan nilai paling besar akan diberhentikan secara *random*.	Mutation
2.	HSAT	Sama seperti GSAT, tetapi variabel yang diberhentikan adalah variabel dengan umur tertua.	Mutation
3.	WalkSAT	Klausa dipilih random. Apabila ada variabel yang memiliki keuntungan negatif 0, salah satu dari ini dipilih	Mutation

		random. Jika tidak ada variabel seperti itu, dipilih suatu variabel acak dengan probabilitas 50% atau variabel dengan keuntungan negatif minimal.	
4.	Flip Random Variable from a Broken Clause	Variabel rusak yang dipilih dari klausa dibalikkan random.	Mutation
5.	Flip Random Variable	Seluruh variabel yang dipilih dibalikkan random.	Mutation
6.	Novelty	Klausa yang rusak dipilih random. Jika ada variabel memiliki keuntungan negatif nol, maka pilih salah satu dari itu secara random. Jika tidak ada, pilih klausa random. Kemudian, balikkan variabel dengan perolehan nilai tertinggi, kecuali variabel yang memiliki umur minimal. Jika tidak ada, variabel dengan probabilitas 0,3 diterima. Jika tidak variabel dengan keuntungan tertinggi kedua diterima.	Mutation
7.	Reinitialise Variables	Variabel diulang kembali secara random. Bergantug pada nilai parameter/intensitas mutasi, baik 0,2, 0,4, 0,6, atau 0,8 dari solusi yang diulang.	Ruin-recreate
8.	Flip Random Variable from a Broken Clause	Variabel dipilih klausa yang rusak secara random, kemudian dibalikkan secara random.	Local search
9.	Flip Random Variable	Variabel yang dipilih, dibalikkan secara random.	Local search
10.	Twopoint crossover	Standar dua titik crossover pada string boolean variabel.	Crossover
11.	Onepoint crossover	Standar satu titik crossover pada string boolean variabel.	Crossover

Kumpulan LLH pada domain permasalahan one dimensional bin packing adalah sebanyak delapan LLH [29]. Rincian LLH dalam bin packing dapat dilihat pada Tabel 4.3.

Tabel 4.3 Kumpulan LLH pada Domain Permasalahan Bin Packing

No	Nama LLH	Deskripsi	Tipe LLH

1.	Swap	Dua bagian berbeda dipilih secara acak, kemudian diganti dengan bagian terpilih jika ada ruang. Jika salah satu bagian tidak sesuai dengan bin baru, masukkan ke dalam bin kosong	Mutation
2.	Split a Bin	Heuristik ini memilih bin pembuangan secara acak dari yang memiliki lebih banyak bagian daripada rata-rata. Kemudian membagi bin ini menjadi dua tempat pembuangan, masing-masing berisi setengah bagian dari bin asalnya.	Mutation
3.	Repack the Lowest Filled Bin	Semua bagian terendah dari bin pembuangan dihapus, dan kembalikan ke dalam bin lainnya jika memungkinkan, dengan heuristik terbaik.	Mutation
4.	Destroy x Highest Bins	Semua bagian terbesar dihapus dari bin pembuangan, di mana x adalah bilangan bulat yang ditentukan oleh parameter `intensitas mutasi'.	Ruin-recreate
5.	Destroy x Lowest Bins	Semua bagian terendah dihapus dari bin pembuangan, di mana x adalah bilangan bulat yang ditentukan oleh parameter `intensitas mutasi'.	Ruin-recreate
6.	Exon Shuffling Crossover	bagian dipindah silang	Crossover
7.	Swap	Dua bagian berbeda dipilih secara acak, dan ditukar dengan mereka jika ada ruang, dan jika itu akan menghasilkan peningkatan kebugaran.	Local search
8.	Swap from Lowest Bin	Bagian terbesar diambil dari bin sampah dengan bagian terkecil, dan ditukar dengan bagian yang lebih kecil dari bin yang dipilih secara acak. Jika tidak ada bagian yang menghasilkan pengepakan yang valid setelah swap, maka tukarkan bagian pertama dengan dua bagian yang memiliki ukuran total lebih kecil. Jika tidak ada bagian seperti itu maka heuristik tidak melakukan apa-apa.	Local search

Rangkain LLH yang terdapat pada domain permasalahan permutation flow shop sebanyak lima belas LLH [30]. Rincian LLH dalam Flow Shop dapat dilihat pada Tabel 4.4.

Tabel 4.4 Kumpulan LLH pada Domain Permasalahan Flow Shops

No	Nama LLH	Deskripsi	Tipe LLH

1.	randomReinsertion	Pekerjaan yang dipilih secara acak dimasukkan kembali ke posisi yang dipilih secara acak dalam permutasi, menggeser sisa pekerjaan sesuai kebutuhan.	Mutation
2.	swapTwo	Dua pekerjaan yang dipilih secara acak ditukarkan di permutasi.	Mutation
3.	shuffle	Pekerjaan dipilih secara acak dalam permutasi.	Mutation
4.	shuffleSubSequence	Elemen k dipilih secara acak dalam permutasi, di mana k=2+[α × (n-2)], dan α adalah parameter intensitas mutasi.	Mutation
5.	useNEH	Solusi baru dibuat menggunakan NEH dan menggunakan permutasi saat ini untuk menentukan peringkat pekerjaan.	Mutation
6.	iteratedGreedy	Elemen *l* dihapus, *l*=[α × (n-2)], pekerjaan yang dipilih secara acak dan dimasukkan kembali dalam mode NEH. Heuristik ini menyerupai komponen utama dari heuristik greedy.	Ruin-recreate
7.	deepIteratedGreedy	Elemen l dihapus seperti h6. Pekerjaan yang dipilih secara acak dimasukkan kembali dalam mode NEH, tetapi setiap iterasi dari prosedur NEH terbaik q, $q = \lfloor \beta * (l - 1) \rfloor + 1$, urutan yang dihasilkan dipertimbangkan untuk reintegrasi.	Ruin-recreate
8.	localSearch	LLH ini adalah pencarian lokal keturunan paling curam. Pada setiap iterasi setiap pekerjaan dihapus dari posisi saat ini dan ditugaskan ke semua posisi yang tersisa. Pekerjaan itu diarahkan ke posisi yang mengarah pada jadwal terbaik. Ini diulang sampai tidak ada perbaikan yang diamati.	Local search
9.	fImpLocalSearch	LLH ini adalah pencarian lokal pertama yang ditingkatkan. Setiap iterasi, pekerjaan dihapus dari posisi saat ini dan ditugaskan ke posisi yang tersisa. Jika ada perbaikan, akan diterima, dan pencarian dilanjutkan dengan pekerjaan berikutnya. Ini	Local search

		diulang sampai tidak ada perbaikan yang diamati.	
10.	randomLocalSearch	LLH ini adalah pencarian lokal tunggal secara acak. Dalam hal ini, $r = \lfloor \beta * (n-1) \rfloor + 1$ pekerjaan yang dipilih secara acak diuji (satu per satu) pada semua posisi dan ditetapkan ke tempat terbaik yang mungkin. Ini hanya dilakukan sekali.	Local search
11.	fImpLocalSearch	LLH ini adalah peningkatan satu kali pencarian lokal tunggal secara acak pertama. Sama seperti h_9, tetapi pekerjaan ditugaskan ke tempat pertama yang meningkatkan jadwal saat ini, yaitu pekerjaan tidak perlu diuji di semua posisi. Ini hanya dilakukan sekali.	Local search
12.	Ox	Permintaan crossover	Crossover
13.	ppx	Precedence preservative crossover	Crossover
14.	pmx	Crossover yang dipetakan sebagian	Crossover
15.	oneX	Crossover satu titik	Crossover

Rangkain LLH yang terdapat pada domain permasalahan personnel scheduling sebanyak dua belas LLH [31]. Rincian LLH dalam personnel scheduling dapat dilihat pada Tabel 4.5.

Tabel 4.5 Kumpulan LLH pada Domain Permasalahan Personnel Scheduling

No	Nama LLH	Deskripsi	Tipe LLH
1.	Mutation heuristic 1	Sejumlah shift dibatalkan secara acak, menjaga solusi yang layak. Jumlah pergeseran yang tidak diberikan operator sebanding dengan intensitas parameter mutasi.	Mutation
2.	Ruin and Recreate Heuristic 1	Shift tidak ditetapkan dalam satu atau lebih jadwal karyawan dipilih secara acak sebelum disusun kembali. Jadwal dibangun kembali oleh tujuan pertama yang memuaskan terkait dengan permintaan untuk bekerja di hari-hari	Ruin-recreate

		tertentu atau shift dan kemudian dengan tujuan yang memuaskan terkait dengan akhir pekan.	
3.	Ruin and Recreate Heuristic 2	LLH kedua ini memberikan perubahan yang lebih besar ke solusi dengan menetapkan x.	Ruin-recreate
4.	Ruin and Recreate Heuristic 3	Varian ketiga dari LLH ini menciptakan gangguan kecil dalam solusi dengan menggunakan x = 1.	Ruin-recreate
5.	Crossover heuristic 1	LLH Ini beroperasi dengan mengidentifikasi tugas terbaik x di setiap parent. Penugasan terbaik diidentifikasi dengan mengukur perubahan dalam fungsi obyektif ketika setiap pergeseran k sementara waktu tidak ditugaskan dalam daftar. Penetapan terbaik adalah yang menyebabkan peningkatan terbesar dalam nilai fungsi obyektif ketika mereka tidak ditetapkan. Parameter x berkisar antara 4-20 dan dihitung menggunakan intensitas parameter mutasi.	Crossover
6.	Crossover heuristic 2	LLH ini menciptakan daftar baru dengan menggunakan semua tugas yang dibuat. Hal ini semua tugas yang umum bagi kedua parent pertama dan kemudian secara bergantian memilih penugasan dari masing-masing parent dan membuatnya dalam keturunan kecuali tujuan sampul sudah terpenuhi.	Crossover
7.	Crossover heuristic 3	LLH ini membuat daftar baru dengan membuat tugas yang hanya umum bagi kedua parent.	Crossover
8.	*Vertical* swaps	Pergeseran antara dua karyawan dipindahkan sehingga pergeseran bergerak secara vertikal dalam daftar.	Local search
9.	Horizontal swaps	Pergeseran dalam pola kerja karyawan tunggal dipindahkan sehingga pergeserannya bergerak secara horizontal dalam daftar.	Local search
10.	New swaps	Perubahan baru diperkenalkan ke dalam daftar (atau menghapus shift secara terbalik).	Local search
11.	Local search heuristics 1-3	LLH ini seperti “hill climbers” yang masing-masing menggunakan salah satu dari jenis operator lingkungan ini.	Local search

12.	Local search heuristics 4 and 5	Variasi dari variable depth search	Local search

Rangkain LLH yang terdapat pada domain permasalahan *travelling salesman problem* (TSP) sebanyak tiga belas LLH. Rincian LLH dalam TSP dapat dilihat pada Tabel 4.6.

Tabel 4.6 Kumpulan LLH pada Domain Permasalahan TSP

No	Nama LLH	Deskripsi	Tipe LLH
1.	randomReinsertion	LLH ini memasukkan kembali kota pada posisi yang berbeda dalam tur secara acak	Mutation
2.	swapTwo	LLH ini menukarkan dua kota yang dipilih secara acak	Mutation
3.	shuffle	LLH ini mengacak tur	Mutation
4.	shuffleSubSequence	LLH ini mengacak tur berikutnya	Mutation
5.	nOptMove		Mutation
6.	iteratedGreedy	Kota-kota dari sub-urutan yang dihapus dimasukkan kembali	Ruin-recreate
7.	twoOptLocalSearch	Pilih dua pencarian lokal, dan pilih peningkatan pertama	Local search
8.	bestImpTwoOptLocalSearch	Pilih dua pencarian lokal dan pilih peningkatan terbaik	Local search
9.	threeOptLocalSearch	Pilih tiga pencarian lokal dan pilih peningkatan pertama	Local search
10.	Ox	Permintaan crossover	Crossover
11.	pmx	Precedence preservative crossover	Crossover
12.	ppx	Crossover yang dipetakan sebagian	Crossover
13.	oneX	Crossover satu titik	Crossover

Rangkain LLH yang terdapat pada domain permasalahan *vehicle routing problem* (VRP) sebanyak sepuluh LLH [32]. Rincian kumpulan LLH yang terdapat di dalam VRP dapat dilihat pada Tabel 4.7.

Tabel 4.7 Kumpulan LLH pada Domain Permasalahan VRP

No	Nama LLH	Deskripsi	Tipe LLH
1.	twoOpt	Heuristik ini menukarkan pelanggan dengan pelanggan yang mendahuluinya dalam rute.	Mutation
2.	orOpt	Dua pelanggan yang berurutan dipilih dalam satu rute, dan dipindahkan ke lokasi lain dalam rute yang sama.	Mutation
3.	locRR	Heuristik ini mengitung perhitungan nilai kedekatan sebagai jarak euclidean antara benchmark pelanggan dan pelanggan yang sedang dipertimbangkan.	Ruin-recreate
4.	timeRR	Heuristik ini menghitung nilai kedekatan perbedaan antara waktu kedatangan benchmark pelanggan saat ini dan pelanggan yang sedang dipertimbangkan. Di mana nilai ini jika berada di bawah batas, maka pelanggan dihapus dari solusi.	Ruin-recreate
5.	combine	Heuristik ini memilih kombinasi rute dari 2 solusi induk untuk membentuk solusi generasi.	Local search
6.	combineLong	Heuristik ini memilih rute dari kedua solusi sebelum mencoba memasukkan pelanggan yang tersisa.	Crossover
7.	interchange	Heuristik ini menerima solusi baru hanya jika dapat meningkatkan nilai fungsi obyektif.	Crossover
8.	shiftMutate	Heuristik ini menghapus pelanggan dari satu rute dan dimasukkan ke rute lain. Informasi dari solusi digunakan untuk membantu memilih pelanggan untuk dihapus.	Mutation
9.	twoOptStar	Heuristik ini menukar bagian akhir dari dua rute untuk membuat dua rute baru.	Local search
10.	GENI	Heuristik ini menghilangkan pelanggan dari satu rute dan memindahkan pelanggan tersebut ke rute lain yang dipilih secara acak.	Local search

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

5.1. Implementasi Hasil

Tahap Implementasi dilakukan pada algoritma yang telah dirancang. Tahap ini diawali dengan mempersiapkan *tools* yang akan dipakai sampai dengan membuat kode program dan eksekusi (*running*). Tahap ini dibahas pada sub bagian berikut ini. Pada penelitian ini, metode seleksi SR diimplementasikan ulang untuk dibandingkan terhadap metode seleksi berbasis RL yang diusulkan.

5.1.1. Kebutuhan Implementasi

Implementasi dalam penelitian ini membutuhkan *tools* sebagai media untuk implementasi desain algoritma yang dirancang. Implementasi dilakukan pada komputer dengan prosesor Intel® Core™ i5-8250U dan memori 8 GB. Hasil desain algoritma diimplementasikan pada *framework* HyFlex [5] menggunakan *tools* NetBeans IDE 8.2. Implementasi pada HyFlex dilakukan dengan cara memanggil fungsi-fungsi (*methods*) dan *libraries chesc.jar* yang ada pada HyFlex.

5.1.2. Fungsi-fungsi HyFlex yang Digunakan

Desain dari metode hiperheuristik yang akan diimplementasikan pada kelas baru turunan dari kelas abstrak *HyperHeuristic* dilakukan dengan cara memanggil fungsi-fungsi dan *libraries* yang ada pada HyFlex. Fungsi-

fungsi dan *libraries* yang dipakai dalam penelitian ini adalah sebagai berikut:

a. *toString*() untuk menampilkan nama metodologi yang dirancang. Setiap kelas harus berisi *method* ini.

b. *initialiseSolution*(*j*), *j* merupakan indeks dari solusi array pada memori. Fungsi ini digunakan sebagai prosedur inisialisasi yang rutin untuk setiap solusi secara *random*.

c. *hasTimeExpired*() mencatat dan melihat berapa lama waktu yang telah berjalan. Algoritma baru yang dibuat harus dijalankan dalam fungsi *hasTimeExpired*().

d. *getNumberOfHeuristics*() berfungsi untuk mendapatkan jumlah LLH.

e. *applyHeuristic* (*i*, *j*, *k*) merupakan fungsi yang memanggil dan menggunakan LLH pada masalah spesifik untuk memodifikasi solusi, dimana *i* adalah indeks LLH untuk meminta solusi, *j* adalah indeks solusi pada memori untuk modifikasi, dan *k* adalah indeks pada memori dimana hasil solusi dapat ditempatkan. Solusi *j* tidak dapat dimodifikasi pada operasi ini.

f. *copySolution*(*k*,*j*) adalah pengaturan memori solusi untuk memindah tempat penyimpanan solusi yang dimodifikasi pada memori dalam

iterasi. Solusi dipindahkan dari indeks memory *k* ke indeks memori *j*.

g. *loadInstance* (*a*), memuat suatu instance dataset di mana *a* adalah indeks dari *instance* yang akan dimuat.

h. *solve*() adalah fungsi yang berisi iterasi sampai batas waktu yang telah ditentukan. Di dalam iterasi, harus menyediakan mekanisme untuk memilih LLH antara domain-spesifik yang tersedia dan memilih solusi mana dalam memori untuk menerapkan LLH. Memori dapat dengan mudah ditentukan dan dipelihara melalui metode panggilan kelas *ProblemDomain*, tempat memori disimpan. Metode *solve*() adalah satu-satunya metode yang harus diterapkan.

i. *setTimeLimit*(60000) merupakan fungsi untuk mengatur waktu yang digunakan untuk iterasi dalam satuan milidetik.

j. *loadProblemDomain*() merupakan fungsi untuk memanggil objek dari kelas domain permasalahan.

k. *run*() merupakan fungsi untuk menjalankan metode hiperheuristik yang telah dibuat.

l. *getBestSolutionValue*() merupakan fungsi untuk mendapatkan nilai *fitness function* solusi terbaik dalam proses pencarian.

m. *bestSolutionToString*() merupakan fungsi untuk mengubah tipe nilai *fitness function* solusi menjadi tipe *string*.

5.1.3. Pengaturan Parameter

Implementasi rancangan algoritma pada penelitian ini, dibutuhkan beberapa konfigurasi nilai parameter. Tabel 5.1 berikut adalah konfigurasi parameter yang dipakai.

Tabel 5.1 Pengaturan Parameter

Variabel	Deskripsi	Nilai	Referensi
initialiseSolution	Paramater untuk indeks array solusi pada memori.	0	[27]
Problem domain	Membuat objek dari kelas domain permasalahan dengan seed acak	1234	[7]
HyperHeuristic	Membuat objek metode hiperheuristik dengan seed acak	5678	[26]
loadInstance	Parameter indeks dari instance domain permasalahan yang digunakan	0-4	[26]
setTimeLimit	parameter kondisi berhenti untuk menghentikan proses pencarian dengan batasan waktu milidetik.	60000 (ms)	[27]

5.1.4. Implementasi RL-LA dan SR-LA

Algoritma hiperheuristik RL-LA dan SRLA diimplementasikan dan diuji coba pada *framework* HyFlex. Untuk kode program dalam implementasi ini dibagi menjadi bagian agar memudahkan dalam memahami alur dari tahap ini. Detail kode program dapat dilihat pada Kode Program 5.1, Kode Program 5.2, Kode Program 5.3 dan Kode Program 5.4.

```
1  package RLLA;
2  import AbstractClasses.HyperHeuristic;
3  import AbstractClasses.ProblemDomain;
4  import SAT.SAT;
5  import BinPacking.BinPacking;
6  import FlowShop.FlowShop;
7  import
   PersonnelScheduling.PersonnelScheduling;
8  import travelingSalesmanProblem.TSP;
9  import VRP.VRP;
10 /**
11  *
12  * @author Anang Firdaus - Sistem Informasi
   - ITS -2019
13  */
```

Kode Program 5.1 Deklarasi Pemanggilan *Library*

Kode Program 4.1 adalah bagian kode yang pertama. Pada bagian ini di jelaskan deklarasi untuk memanggil *library* yang dibutuhkan untuk uji coba ini. Baris pertama

adalah nama *package* yang dipakai sebagai tempat menyimpan kode program hiperheuristik di dalam HyFlex. Baris kedua merupakan pemanggilan kelas *HyperHeuristic* berisi *method* hiperheuristik yang akan digunakan dalam uji coba. Baris ketiga berfungsi untuk pemanggilan kelas domain permasalahan yang terletak yang berisi method pada kelas tersebut.

Baris keempat sampai kesembilan merupakan pemanggilan domain permasalahan sebagai objek yang digunakan dalam uji coba ini. Baris keempat adalah pemanggilan fungsi domain SAT. Baris kelima merupakan pemanggilan domain *BinPacking*. Baris keenam merupakan pemanggilan domain *FlowShop*. Baris ketujuh merupakan pemanggilan domain *PersonnelScheduling*. Baris kedelapan merupakan pemanggilan domain TSP. Baris kesembilan merupakan pemanggilan domain VRP. Untuk kode program algoritma hiperheuristik dapat dilihat pada Kode Program 5.2 dan Kode Program 5.3.

Kode program 5.2 merupakan bagian kode program utama untuk proses hiperheuristik RL-LA. Baris kelimabelas merupakan deklarasi kelas *RLLA* yang merupakan *extends* dari kelas *HyperHeuristic*, artinya semua fungsi-fungsi atau *method* yang terdapat pada kelas *HyperHeuristic* dapat digunakan oleh kelas *RLLA*.

```
14  public class RLLA extends HyperHeuristic {
15      public RLLA(long seed){
16              super(seed);
17      }
18  public void solve(ProblemDomain problem) {
19  int number_of_heuristics =
    problem.getNumberOfHeuristics();
20  int[] scoreList = new int
    [number_of_heuristics];
21  int L = 1100;
22  int upperBound = 10;
23  int lowerBound = 0;
24  int initialScore = 5;
25
26  for (int i = 0; i < number_of_heuristics;
    i++) {
27  scoreList[i]=initialScore;
28  }
29
30  double current_obj_function_value =
    Double.POSITIVE_INFINITY;
31  problem.initialiseSolution(0);
32  BlockingQueue<Double> fitnessList = new
    ArrayBlockingQueue<>(L);
33  fitnessList.add(current_obj_function_value);
34  int heuristic_to_apply =
    rng.nextInt(number_of_heuristics);
35  int max;
36  List<Integer> maxValsList = new
    ArrayList<>();
37
38  while (!hasTimeExpired()) {
39  max = Integer.MIN_VALUE;
40  maxValsList.clear();
41  for (int i=0; i < scoreList.length; ++i) {
42  if (scoreList[i] == max) {
43  maxValsList.add(i);
44  }
45  else if (scoreList[i] > max) {
46  maxValsList.clear();
47  maxValsList.add(i);
48  max = scoreList[i];
49  }
50  }
51  heuristic_to_apply =
    maxValsList.get(rng.nextInt(maxValsList.size(
    )));
```

```
52  double new_obj_function_value =
    problem.applyHeuristic(heuristic_to_apply, 0,
    1);
53
54  if (new_obj_function_value <
    fitnessList.peek()) {
55  problem.copySolution(1, 0);
56  scoreList[heuristic_to_apply]++;
57  if(scoreList[heuristic_to_apply]>=upperBound)
    {
58  scoreList[heuristic_to_apply]=lowerBound;
59  }
60  if (fitnessList.size()==L) {
61  fitnessList.poll();
62  fitnessList.add(new_obj_function_value);
63  }
64  else
65  fitnessList.add(new_obj_function_value);
66  }else
67  {
68  scoreList[heuristic_to_apply]--;
69  if(scoreList[heuristic_to_apply]<=lowerBound)
    {
70  scoreList[heuristic_to_apply]=lowerBound;
71  }
72  }
73
74  }
75  }
```

Kode Program 5.2 Hyper-heuristic RL-LA

Baris dua puluh dua adalah *method* untuk menuliskan strategi hiperheuristik yang dirancang. Parameter *problem* merupakan domain permasalahan yang akan diselesaikan. Baris kedua puluh sembilan merupakan penetapan parameter L untuk jumlah iterasi kriteria penerimaan solusi pada *Late Acceptance*, nilai parameter L didapatkan dari hasil uji coba pada lampiran D. Baris ketiga puluh sampai tiga puluh dua merupakan penetapan

parameter score yang menentukan batas atas , batas bawah dan inisialisasi *score* pada setiap LLH [33], nilai parameter ini juga didapatkan dari hasil uji coba pada lampiran D. Pemanggilan *low level heuristic* yang akan digunakan pada setiap domain permasalahan untuk menghasilkan solusi. Baris ketiga puluh sembilan dilakukan pencarian inisial solusi yang ditempatkan pada memory 0. Baris tiga puluh delapan adalah nilai fungsi objektif dari inisial solusi. Baris empat puluh enam merupakan iterasi yang dilakukan untuk mencari solusi sampai batas waktu yang ditentukan.

Baris keempat puluh tujuh sampai bari kelima puluh sembilan adalah proses memilih LLH yang memiliki score tertinggi. Baris keenam puluh merupakan hasil *generate* nilai fungsi objektif solusi baru dari LLH yang terpilih. Nilai ini diletakkan pada indeks memory 1. Baris keenam puluh dua sampai ke delapan puluh merupakan proses penerimaan atau penolakan solusi serta pemberian reward dan penalty dimana reward dan penalty tersebut didapatkan dari hasil uji coba pada lampiran D.

Kode Program 5.3 merupakan bagian kode program utama untuk proses hiperheuristik SR-LA

```
202 public class SRLA extends HyperHeuristic {
203  public SRLA(long seed){
204          super(seed);
205  }
206  public void solve(ProblemDomain problem) {
207             int number_of_heuristics =
    problem.getNumberOfHeuristics();
208             double current_obj_function_value
    = Double.POSITIVE_INFINITY;
209             problem.initialiseSolution(0);
210             int L = 500;
211             BlockingQueue<Double> fitnessList
    = new ArrayBlockingQueue<>(L);
212
    fitnessList.add(current_obj_function_value);
213             while (!hasTimeExpired()) {
214
215                     int heuristic_to_apply =
    rng.nextInt(number_of_heuristics);
216                     double
    new_obj_function_value =
    problem.applyHeuristic(heuristic_to_apply, 0,
    1);
217                     if
    (new_obj_function_value < fitnessList.peek())
    {
218
    problem.copySolution(1, 0);
219                         if
    (fitnessList.size()==L) {
220
    fitnessList.poll();
221
    fitnessList.add(new_obj_function_value);
222                         }else
223
    fitnessList.add(new_obj_function_value);
224                     }
225
226             }
227         }
```

Kode Program 5.3 Hyper-heuristic SR-LA

Untuk menjalankan kode hiperheuristik yang telah dibuat dijelaskan pada Kode Program 5.4.

```
228 public static void main(String[] args) {
229 ProblemDomain problem = new SAT(1234);
230 HyperHeuristic HH_Object = new RLLA(5678);
231 System.out.println("Metode "+HH_Object );
232 problem.loadInstance(0);
233 HH_Object.setTimeLimit(60000);
234 HH_Object.loadProblemDomain(problem);
235 HH_Object.run();
236 System.out.println("BEST SOLUTION : " +
    HH_Object.getBestSolutionValue());   }}
```

Kode Program 5.4 Running Hyperheuristic

Dalam Kode Program 4.4, baris kedua ratus tiga merupakan pembuatan objek untuk domain permasalahan yang akan digunakan. Baris dua ratus empat merupakan pembuatan objek untuk metode hiperheuristik yang akan digunakan yang digunakan dengan *seed* acak. Baris kedua ratus lima digunakan untuk menampilkan nama dari hiperheuristik yang digunakan. Baris kedua ratus enam merupakan indeks *instance* yang digunakan untuk dijalankan. *Instance* 0 artinya *instance* indeks 0, *instance* dimulai dari indeks 0. Baris kedua ratus tujuh adalah batas waktu untuk menjalankan metode hiperheuristik yang digunakan dalam mencari solusi. Disini menggunakan batasan waktu 60000 milidetik. Baris kedua ratus delapan adalah objek metode hiperheuristik yang diberikan acuan kepada objek domain permasalahan. Baris kedua ratus sembilan adalah pemanggilan method *run*() untuk menjalankan proses hiperheuristik dalam melakukan pencarian solusi. Baris kelima puluh empat menampilkan nilai fungsi objektif solusi terbaik.

Halaman ini sengaja dikosongkan

BAB VI
UJI COBA DAN ANALISIS HASIL

Bab ini menjelaskan proses uji coba dan analisis dari metode untuk hiperheuristik RL-LA dengan algoritma pembanding yaitu *Simple Random Late Acceptance* (SR-LA) terhadap enam domain permasalahan optimasi yang telah dijelaskan sebelumnya. Setelah itu dilakukan evaluasi kinerja dari algoritma RL-LA.

6.1 Hasil Uji Coba

Algoritma *high level heuristic* yang digunakan, dieksekusi pada enam domain permasalahan optimasi. Di setiap domain permasalahan optimasi terdapat lima *instance* data, sehingga secara keseluruhan terdapat 30 *instance* data. Dari setiap *instance* data diuji coba sebanyak 31 kali eksekusi dengan waktu *running* 60000 milidetik [7][25]. Dari hasil eksekusi, didapatkan kumpulan data nilai fungsi *fitness* setiap *instance* data. Hasil uji coba algoritma *high level heuristic* yang digunakan dijelaskan pada sub bagian berikut ini.

6.1.1 Hasil Uji Coba SR-LA

Hasil *running* dari SR-LA yang dilakukan pada 30 variasi instance pada enam domain permasalahan dapat dilihat pada Lampiran A. Data hasil perhitungan nilai terbaik (minimum), kuartil pertama, median, kuartil ketiga, nilai maksimum, dan rata-rata dari nilai fungsi *fitness* pada setiap *instance* masing-masing domain permasalahan

dapat dilihat pada Tabel 6.1 berikut ini. Hasil dari ekseksusi metode ini menjadi perbandingan untuk algoritma RLLA.

Tabel 6.1 Data Hasil Ekseskusi SR-LA

Domain Permasalahan	Instance	Min	Q1	Median	Q3	Maks
SAT	3	7	9	13	16	25
	5	9	13.5	18	27	243
	4	4	8	9	12.5	143
	10	8	14	15	18.5	40
	11	10	13	14	18.5	46
BinPacking	7	0.173680	0.180048	0.182416	0.187891	0.194771
	1	0.071094	0.075646	0.076787	0.080524	0.087946
	9	0.039695	0.042880	0.044649	0.046317	0.053082
	10	0.126133	0.127281	0.127499	0.127594	0.127915
	11	0.066186	0.071703	0.074517	0.078318	0.092808
FlowShop	5	19	28	30	35.5	43
	9	10539	16738.5	31713	39867	87457
	8	3273	3356.5	3491	3636.5	4300
	10	1558	1807.5	2015	2183.5	3165
	11	320	347.5	370	390	1720
Personnel Scheduling	1	6340	6369	6380	6388	6400
	8	26903	26982.5	27000	27022.5	27051
	3	6408	6426	6441	6449.5	6467
	10	11517	11554	11562	11575.5	11602
	11	26743	26782.5	26798	26826	26897
TSP	0	49632.41	50608.08	51327.52	51733.98	52282.16
	8	2118964	2166099	2252034	2497337	253257
	2	7003.405	7056.466	7080.019	7104.737	7128.255
	7	69915.85	72313.34	75357.49	79189.34	80365.06
	6	55139.36	58240.24	60191.69	61102.09	65880.69
VRP	6	102113.8	106354.4	108810.6	112265.0	119641.3
	2	15878.64	17003.22	17978.32	18111.23	19190.11
	5	332666.3	349761.5	360081.0	367069.6	391686.2
	1	22953.57458	25777.5594	27485.61043	28593.90306	30599.98395
	9	200472.3	219097.8727	224333.0392	230779.8011	236232.6

6.1.2 Hasil Uji Coba RL-LA

Hasil *running* dari SR-LA yang dilakukan pada 30 variasi instance pada enam domain permasalahan dapat dilihat pada Lampiran A. Data hasil perhitungan nilai terbaik (minimum), kuartil pertama, median, kuartil ketiga, nilai maksimum, dan rata-rata dari nilai fungsi *fitness* pada setiap *instance* masing-masing domain permasalahan dapat dilihat pada Tabel 6.2 berikut ini. Hasil dari ekseksusi metode ini menjadi perbandingan untuk algoritma SR-LA.

Tabel 6.2 Data Hasil Eksekusi RL-LA

Domain Permasalahan	Instance	Min	Q1	Median	Q3	Maks
SAT	3	5	8	10	12.5	28
	5	4	7.5	12	18.5	76
	4	2	4	6	8	48
	10	8	10.5	14	20.5	56
	11	8	11.5	14	18	25
BinPacking	7	0.159912832	0.17457095	0.179959314	0.187096142	0.193885074
	1	0.061829329	0.07152568	0.076116693	0.080092461	0.085174843
	9	0.044619446	0.048976783	0.049960718	0.050949972	0.054059372
	10	0.125825114	0.127043067	0.127199467	0.127409333	0.127761956
	11	0.078909524	0.083891228	0.085738756	0.087778248	0.09226123
FlowShop	5	17	26	28	33.5	41
	9	9749	10087	10232	10765	22675

Domain Permasalahan	Instance	Min	Q1	Median	Q3	Maks
	8	3203	3324.5	3376	3438	3698
	10	1664	1955	2090	2218.5	3080
	11	320	385	410	447.5	1760
Personnel Scheduling	1	6342	6394	6403	6413	6456
	8	26975	27020.5	27050	27056.5	27201
	3	6400	6429	6448	6461.5	6512
	10	11530	11579.5	11595	11613	11677
	11	26753	26816.5	26827	26868	26913
TSP	0	49112.41331	50456.84897	50880.82775	51351.51569	53109.5144
	8	21190320.6	22201045.56	23530136.49	25024671.54	25337432.72
	2	7004.931891	7081.128996	7105.80593	7196.001718	7331.887875
	7	6996.708622	7077.509522	7092.303222	7123.227885	7162.379838
	6	55952.70202	59774.81309	60549.20105	61390.94534	64288.84671
VRP	6	90569.70358	96548.40185	99856.26228	102476.8138	109688.4909
	2	15759.30365	16783.96179	17060.6299	17947.94307	18243.56636
	5	255160.9657	316756.0244	340975.3792	348450.8122	380042.0352
	1	21916.35463	24022.66622	25122.85746	26319.95871	28588.63084
	9	178289.3463	197146.8922	203516.7574	215239.3759	227753.4051

5.2. Analisa Hasil

Pada bagian analisa hasil ini dilakukan evaluasi kinerja terhadap algoritma yang digunakan. Evaluasi kinerja ini

bertujuan untuk menguji pengaruh dari metode seleksi LLH menggunakan RL dalam meningkatkan kinerja (solusi optimal) pada enam permasalahan lintas domain. Analisis hasil ini dimulai dari pengukuran, penilaian, dan pengujian. Pengukuran kinerja dilakukan dengan cara membandingkan hasil uji coba metode yang diusulkan dengan metode lainnya. Disini, metode seleksi LLH menggunakan *Reinforcement Learning* dibandingkan dengan metode Simple Random yang dikombinasikan dengan metode *move acceptance* yang sama untuk melihat kinerja metode yaitu *Late Acceptance*. Langkah-langkah dalam analisis hasil diuraikan pada sub bagian berikut ini.

6.2.1. Pengujian RL-LA

Pengujian ini digunakan untuk mengukur kinerja dari metode RL yang dikombinasikan dengan *Late Acceptance* (RL-LA) sebagai strategi *high level heuristic*. Dalam mengukur kinerja metode RL-LA ini dibandingkan data hasil uji coba RL-LA dengan data hasil uji coba metode Simple Random yang dikombinasikan dengan metode *move acceptance* yang sama yaitu *Simple Random - Late Acceptance* (SR-LA). Perbandingan ini dilakukan ke dalam tiga tahap:

a. Pertama, membandingkan penyebaran data secara keseluruhan dari hasil eksekusi masing-masing strategi menggunakan diagram boxplot.

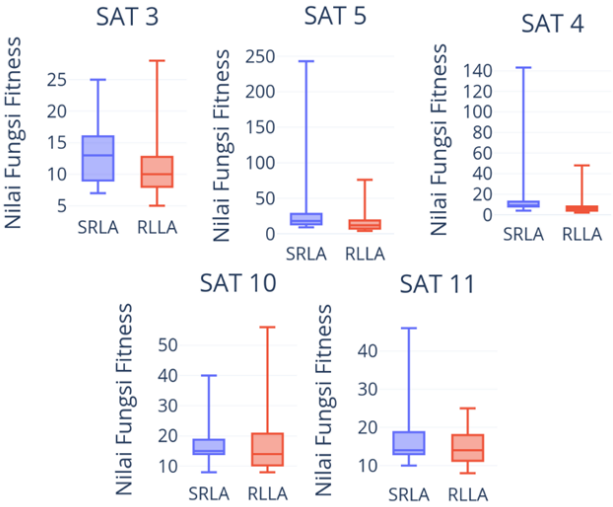
b. Kedua, secara spesifik membandingkan nilai median untuk mengukur pemusatan data yag menjadi nilai tengah dari sekolompok data.

c. Ketiga, membandingkan nilai minimum fungsi *fitness* hasil eksekusi masing-masing strategi untuk mengukur kualitas solusi dari fungsi objektif untuk meminimalkan, maka kemampuan strategi *high level heuristic* yang diusulkan dalam menghasilkan nilai minimum fungsi objektif perlu diuji.

Pada tahap pertama, perbandingan kinerja metode dianalis menggunakan diagram boxplot untuk membandingkan penyebaran atau distribusi data hasil ekseskusi dari nilai minimum, kuartil pertama, median, kuartil ketiga, dan nilai maksimum yang dilakukan terhadap masing-masing domain permasalahan. *Box* mewakili strategi SR-LA dan RL-LA. Letak posisi (atas dan bawah) *box* menyatakan kinerja masing-masing strategi.

Box yang terletak pada nilai fungsi *fitness* lebih kecil (bawah), maka kinerja strategi tersebut semakin baik. Hasil dari perbandingan diberikan penilaian (poin) untuk menyimpulkan hasil pengujian. Setiap metode yang memiliki nilai fungsi *fitness* lebih kecil akan diberikan satu poin. Jumlah poin (skor) maksimum untuk semua domain permasalahan adalah 150 poin. Setiap domain permasalahan memiliki total skor 25 poin. Jumlah poin (skor) semua domain permasalahan adalah hasil dari pengujian terhadap metode RL-LA.

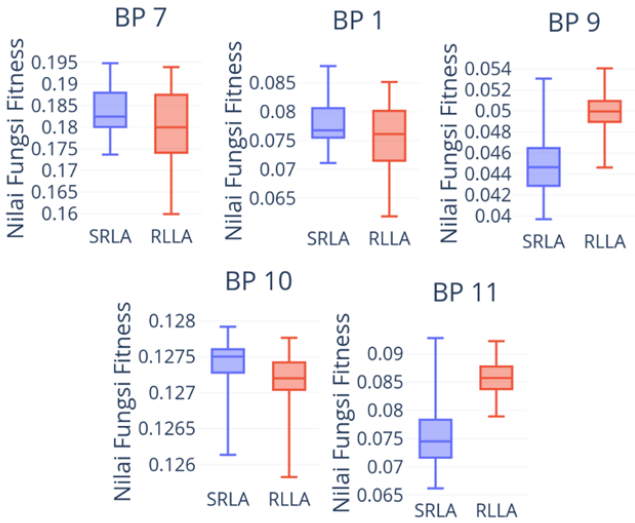
Hasil penilaian skor tersebut dapat dilihat pada Lampiran C. Pada domain permasalahan satisfiability (SAT), RL-LA memiliki kinerja lebih baik daripada SR-LA, terutama pada *instance* 3, 5, 4, dan 11 seperti yang dapat dilihat pada diagram boxplot pada Gambar 6.1. Skor RL-LA unggul 17 poin dengan total skor RL-LA 20 poin sedangkan SR-LA 3 poin.



Gambar 6.1 Pengujian RL-LA Domain SAT

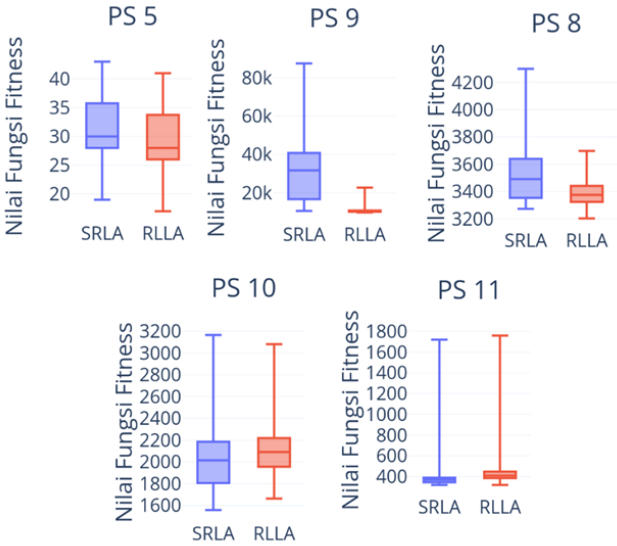
Pada domain permasalahan bin packing (BP), RL-LA juga memiliki kinerja lebih baik daripada SR-LA, terutama pada *instance* 7, 1, dan 10 seperti yang dapat

dilihat pada diagram boxplot pada Gambar 6.2. Skor RL-LA unggul 7 poin dengan total skor RL-LA 16 poin sedangkan SR-LA 9 poin.



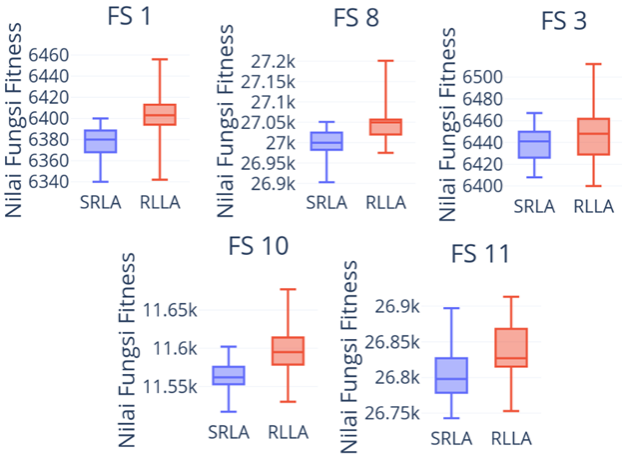
Gambar 6.2 Pengujian RL-LA Domain BP

Pada domain permasalahan personnel scheduling (PS), RL-LA juga memiliki kinerja lebih baik daripada SR-LA, terutama pada *instance* 5, 9, dan 8 seperti yang dapat dilihat pada diagram boxplot pada Gambar 6.3. Skor RL-LA unggul 8 poin dengan total skor RL-LA 16 poin sedangkan SR-LA 8 poin.



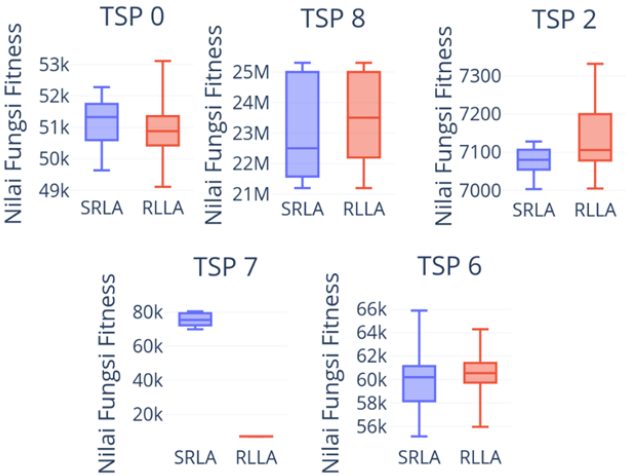
Gambar 6.3 Pengujian RL-LA Domain PS

Sedangkan pada domain permasalahan FlowShop (FS), RL-LA memiliki kinerja jauh lebih buruk daripada SR-LA di semua instance seperti yang dapat dilihat pada diagram boxplot pada Gambar 6.4. Total skor RL-LA hanya 1 poin sedangkan skor SR-LA unggul 23 poin dengan total skor 24 poin.



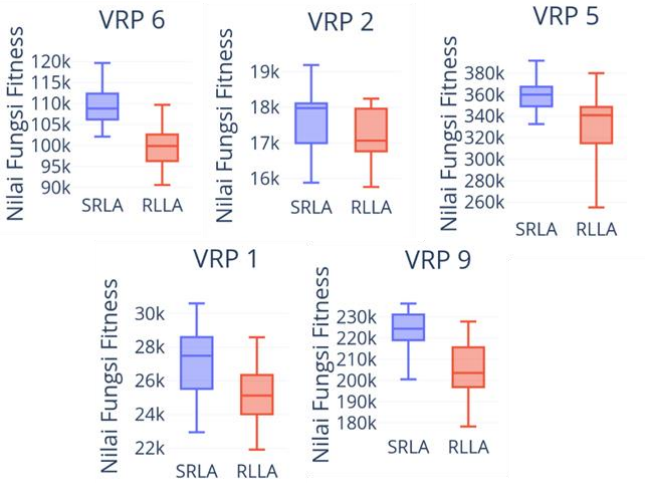
Gambar 6.4 Pengujian RL-LA Domain FS

Pada domain permasalahan TSP, RL-LA juga memiliki kinerja lebih buruk daripada SR-LA, terutama pada instance 8, 2, dan 6 seperti yang dapat dilihat pada diagram boxplot pada Gambar 6.5. Total skor RL-LA 10 poin sedangkan skor SR-LA unggul 5 poin dengan total skor 15 poin.



Gambar 6.5 Pengujian RL-LA Domain TSP

Pada domain permasalahan VRP, RL-LA memiliki kinerja jauh lebih baik daripada SR-LA pada semua instance seperti yang dapat dilihat pada diagram boxplot pada Gambar 6.6. Total skor RL-LA unggul telak sebanyak 25 poin sedangkan skor SR-LA tidak mendapatkan skor sama sekali.



Gambar 6.6 Pengujian RL-LA Domain VRP

Dari hasil pengujian dengan perbandingan dan penilaian pada diagram boxplot terhadap nilai minimum, kuartil pertama, median, kuartil ketiga, dan nilai maksimum hasil eksekusi, algoritma RL-LA memiliki kinerja yang lebih baik 29 poin daripada SR-LA dengan total skor RL-LA 88 poin sedangkan SR-LA 59 poin. Dari hasil pengujian tersebut, algoritma *Reinforcement Learning* yang diusulkan dapat meningkatkan kinerja hiperheuristik dalam menghasilkan solusi yang lebih optimal terutama pada 4 domain permasalahan yaitu SAT, Bin Packing, Personnel Scheduling, dan VRP. Sedangkan untuk domain permasalahan Flow Shop dan TSP belum optimal. Untuk detail dari perbandingan dan penilaian dari kedua metode dapat dilihat pada Lampiran C.

Pada tahap kedua dilakukan perbandingan nilai median untuk mengukur pemusatan data yag menjadi nilai tengah dari sekolompok data. Untuk pengukuran kedua metode, digunakan sistem poin bola FIFA untuk melihat persaingan dari metode tersebut. Data nilai median pada hasil eksekusi setiap domain permasalahan pada setiap stretagi yang bersaing akan diberikan tiga (3) poin untuk strategi yang menang, satu (1) point untuk hasil imbang, dan tidak ada (0) poin untuk strategi yang kalah. Menang, kalah, atau imbang tersebut dihitung dari penjumlahan perubahan nilai (peningkatan dan penurunan) pada kedua metode. Perubahan nilai dihitung dengan perhitungan pada persamaan 6.1.

$$\Delta\,(\%) = ((\boldsymbol{a} - \boldsymbol{b})/\boldsymbol{b}) \times \mathbf{100} \quad (5.$$

Persamaan 6.1 Perubahan Nilai

Dalam persamaan 6.1, *a* adalah nilai fungsi fitness dari metode perbandingan dan *b* adalah nilai fungsi fitness hasil eksekusi metode yang diusulkan. Hasil dari perubahan nilai jika bernilai positif, maka terjadi peningkatan tetapi jika perubahan bernilai negatif (-) maka terjadi penurunan kinerja. Persentase perubahan nilai pada setiap *instance* setiap domain permasalahan dijumlahkan. Jika total persentase perubahan bernilai positif, maka metode RL-LA dianggap menang dan diberikan tiga poin. Pengujian ini bertujuan untuk mencari strategi *high level heuristic* yang paling optimal untuk menyelesaikan masalah lintas domain.

Tabel 6.3 berikut ini adalah perbandingan nilai median dari kedua metode menggunakan sistem poin FIFA.

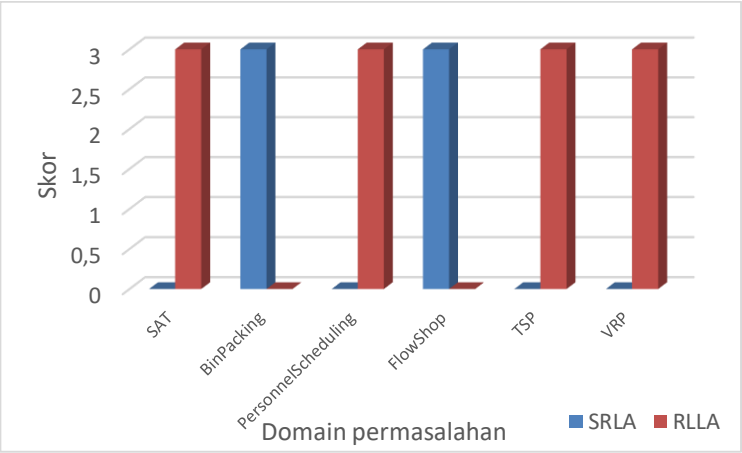
Tabel 6.3 Pengujian Nilai Median RL-LA

Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
SAT	3	13	10	10	3	30.00%	137.1%	0	3
	5	18	12	12	6	50.00%			
	4	9	6	6	3	50.00%			
	10	15	14	14	1	7.14%			
	11	14	14	14	0	0.00%			
BinPacking	7	0.182416281	0.179959314	0.179959314	0.002456967	1.37%	-21.2%	3	0
	1	0.076787151	0.076116693	0.076116693	0.000670458	0.88%			
	9	0.044649806	0.049960718	0.044649806	-0.005310912	-10.63%			
	10	0.127499378	0.127199467	0.127199467	0.000299911	0.24%			
	11	0.074517505	0.085738756	0.074517505	-0.011221251	-13.09%			

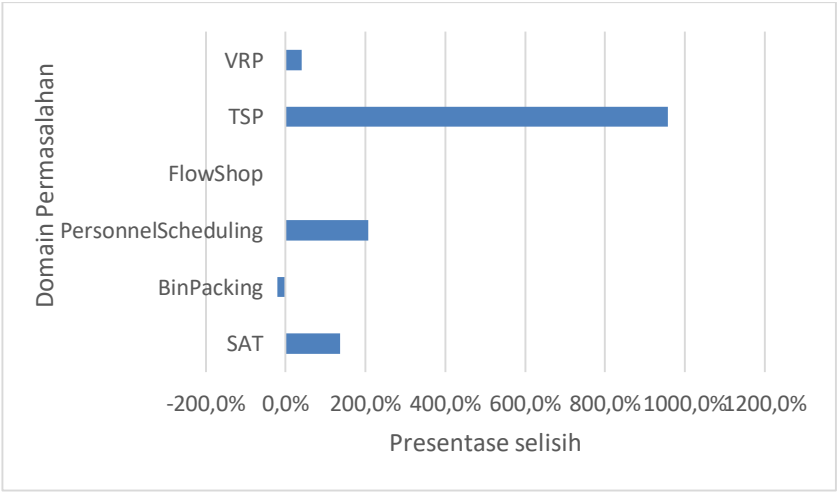
Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
PersonnelScheduling	5	30	28	28	2	7.14%	207.1%	0	3
	9	31713	10232	10232	21481	209.94%			
	8	3491	3376	3376	115	3.41%			
	10	2015	2090	2015	-75	-3.59%			
	11	370	410	370	-40	-9.76%			
FlowShop	1	6380	6403	6380	-23	-0.36%	-1.0%	3	0
	8	27000	27050	27000	-50	-0.18%			
	3	6441	6448	6441	-7	-0.11%			
	10	11562	11595	11562	-33	-0.28%			
	11	26798	26827	26798	-29	-0.11%			
TSP	0	51327.52573	50880.82775	50880.82775	446.6979846	0.88%	958.2%	0	3
	8	22520346.55	23530136.49	22520346.55	-1009789.942	-4.29%			
	2	7080.019716	7105.80593	7080.019716	-25.78621492	-0.36%			
	7	75357.49123	7092.303222	7092.303222	68265.18801	962.52%			

Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
	6	60191.69561	60549.20105	60191.69561	-357.5054485	-0.59%			
VRP	6	108810.6632	99856.26228	99856.26228	8954.400942	8.97%	39.6%	0	3
	2	17978.32745	17060.6299	17060.6299	917.6975493	5.38%			
	5	360081.0111	340975.3792	340975.3792	19105.63191	5.60%			
	1	27485.61043	25122.85746	25122.85746	2362.752972	9.40%			
	9	224333.0392	203516.7574	203516.7574	20816.28186	10.23%			
Jumlah terbaik	11		18	Total Skor Perubahan				6	12
Persentase (%)	38%		62%	Persentase				33%	67%

Berdasarkan data pengujian terhadap nilai median, RL-LA menang pada empat domain permasalahan, yaitu pada domain permasalahan SAT, personnel scheduling, TSP dan VRP sedangkan pada domain permasalahan bin packing dan flowshop, metode RL-LA kalah karena hanya mendapatkan nol poin. Metode seleksi LLH RL secara sistematis dapat menghasilkan solusi yang lebih optimal untuk empat domain permasalahan, tetapi dua domain permasalahan memiliki hasil yang lebih buruk daripada metode seleksi random. Hal ini menandakan metode RL belum bisa meningkatkan kinerja pada domain permasalahan bin packing dan flowshop, karena kinerja hiperheuristik tidak saja dipengaruhi oleh metode seleksi LLH, tetapi juga dipengaruhi oleh metode move acceptance. Untuk melihat grafik skor dan nilai selisih pengujian kinerja RL-LA dari nilai median pada setiap domain permasalahan, dapat dilihat pada Gambar 6.7 dan Gambar 6.8.



Gambar 6.7 Grafik Perbandingan Skor Median SR-LA dan RL-LA



Gambar 6.8 Grafik Presentase Selisih Nilai Median Fungsi Fitness

Pada tahap ketiga, nilai minimum dari hasil 31 kali eksekusi masing-masing hiperheuristik dibandingkan untuk mengukur kinerja algoritma dalam meningkatkan kualitas solusi optimal (nilai fungsi *fitness* minimum). Penilaian untuk mengukur metode sama dengan sistem poin penilaian hasil nilai median yaitu menggunakan sistem poin bola FIFA. Pengujian nilai minimum metode RL-LA dapat dilihat pada Tabel 6.4.

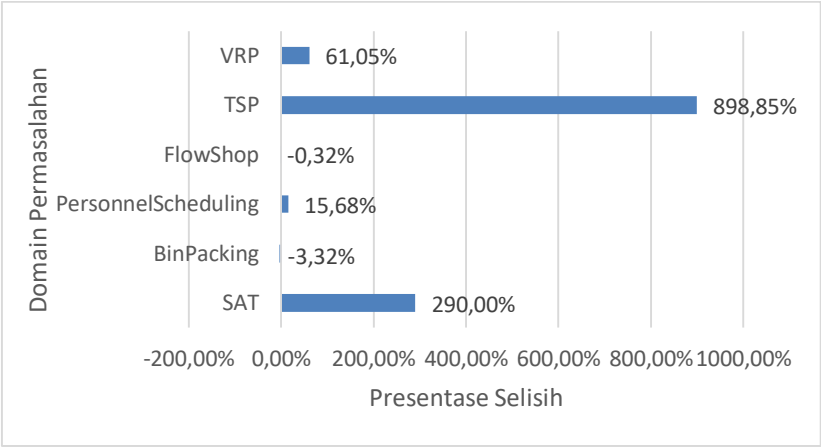
Berdasarkan data pengujian terhadap nilai minimum, RL-LA menang pada empat domain permasalahan, yaitu pada domain permasalahan SAT, personnel scheduling, TSP dan VRP sedangkan pada domain permasalahan bin packing dan flowshop, metode RL-LA kalah karena hanya mendapatkan nol poin. Metode seleksi LLH RL secara sistematis dapat menghasilkan solusi yang lebih optimal untuk empat domain permasalahan, tetapi dua domain permasalahan memiliki hasil yang lebih buruk daripada metode seleksi random. Hal ini menandakan metode RL belum bisa meningkatkan kinerja pada domain permasalahan bin packing dan flowshop, karena kinerja hiperheuristik tidak saja dipengaruhi oleh metode seleksi LLH, tetapi juga dipengaruhi oleh metode move acceptance. Untuk melihat grafik perubahan nilai dan skor pengujian kinerja RL-LA dari nilai median pada setiap domain permasalahan, dapat dilihat pada Gambar 6.9 dan Gambar 6.10

Tabel 6.4 Pengujian Nilai Minimum RL-LA

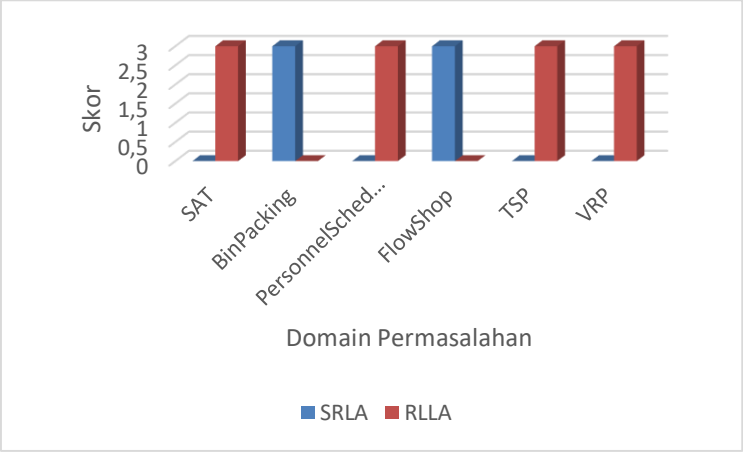
Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
SAT	3	7	5	5	**2**	**40.00%**	290.0%	0	3
	5	9	4	4	**5**	**125.00%**			
	4	4	2	2	**2**	**100.00%**			
	10	8	8	8	**0**	**0.00%**			
	11	10	8	8	**2**	**25.00%**			
BinPacking	7	0.173680589	0.159912832	0.159912832	**0.013767757**	**8.61%**	-3.3%	3	0
	1	0.07109422	0.061829329	0.061829329	**0.009264891**	**14.98%**			
	9	0.039695278	0.044619446	0.039695278	**-0.004924168**	**-11.04%**			
	10	0.126133333	0.125825114	0.125825114	**0.000308219**	**0.24%**			
	11	0.066186746	0.078909524	0.066186746	**-0.012722778**	**-16.12%**			
PersonnelScheduling	5	19	17	17	**2**	**11.76%**	15.7%	0	3
	9	10539	9749	9749	**790**	**8.10%**			

Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
	8	3273	3203	3203	**70**	**2.19%**			
	10	1558	1664	1558	**-106**	**-6.37%**			
	11	320	320	320	**0**	**0.00%**			
FlowShop	1	6340	6342	6340	**-2**	**-0.03%**	-0.3%	3	0
	8	26903	26975	26903	**-72**	**-0.27%**			
	3	6408	6400	6400	**8**	**0.13%**			
	10	11517	11530	11517	**-13**	**-0.11%**			
	11	26743	26753	26743	**-10**	**-0.04%**			
TSP	0	49632.41258	49112.41331	49112.41331	**519.9992689**	**1.06%**	898.8%	0	3
	8	21189643.04	21190320.6	21189643.04	**-677.5582596**	**0.00%**			
	2	7003.4058	7004.931891	7003.4058	**-1.526091529**	**-0.02%**			
	7	69915.8509	6996.708622	6996.708622	**62919.14227**	**899.27%**			
	6	55139.36884	55952.70202	55139.36884	**-813.3331789**	**-1.45%**			
VRP	6	102113.8192	90569.70358	90569.70358	**11544.11562**	**12.75%**	61.1%	0	3

Domain Permasalahan	Ins	SRLA	RLLA	Solusi Terbaik	% Nilai Perubahan		Total % Perubahan	Skor	
								SRLA	RLLA
	2	15878.64252	15759.30365	15759.30365	**119.3388703**	**0.76%**			
	5	332666.3338	255160.9657	255160.9657	**77505.36814**	**30.38%**			
	1	22953.57458	21916.35463	21916.35463	**1037.219947**	**4.73%**			
	9	200472.334	178289.3463	178289.3463	**22182.98771**	**12.44%**			
Jumlah terbaik		**10**	**18**		**Total Skor Perubahan**			6	12
Persentase (%)		**36%**	**64%**		**Persentase**			**33%**	**67%**



Gambar 6.9 Grafik Presentase Selisih Nilai Minimum Fungsi Fitness



Gambar 6.10 Grafik Perbandingan Skor Nilai Minimum SR-LA dan RL-LA

6.2.2. Evaluasi Kinerja RL-LA

Evaluasi kinerja ini bertujuan untuk memilih satu metode dari semua hasil uji coba yang memiliki kinerja terbaik dalam menghasilkan solusi optimal pada permasalahan optimasi lintas domain. Evaluasi dilakukan pada hasil pengujian RL-LA yang dibandingkan dengan metode yang menjadi patokan dalam penelitian ini (SR-LA). Pengujian metode tersebut menggunakan sistem Formula One. Sistem Formula One Point digunakan untuk memelihara persaingan yang fair. Pada sistem ini, hasil median digunakan sebagai data perbandingan. Median pada masing-masing fungsi *fitness* minimum setiap *instance* diberikan poin berdasarkan peringkat, dan kemudian poin dari semua *instance* ditotalkan.

Dalam sistem Formula One, kinerja delapan peringkat terbaik, diberikan poin 10, 8, 6, 5, 4, 3, 2, 1. Jika ada peserta lebih dari delapan, maka semua peserta dengan hasil yang lebih buruk dari peringkat delapan teratas, akan diberi poin nol. Jika beberapa peserta memiliki hasil yang sama, maka poin akan diberikan sama pada setiap peserta berdasarkan peringkatnya [5]. Namun, karena peserta pada persaingan ini hanya terdiri dari dua peserta, maka akan dilakukan perubahan poin pada setiap peringkat, yaitu peringkat pertama mendapat 2 poin sedangkan peringkat kedua mendapat 1 poin. Poin setiap *instance* pada domain permasalahan dimasukkan ke dalam tabel perbandingan. Total skor (poin) pada seluruh *instance* menjadi hasil pengujian

untuk menentukan metode terbaik. Skor maksimum untuk semua domain permasalahan adalah 60 poin. Metode yang memiliki total skor yang lebih besar memiliki kinerja yang lebih baik. Tabel 6.5 berikut merupakan perhitungan skor untuk menentukan peringkat pada sistem formula one.

Tabel 6.5 Perbandingan Sistem Formula One

Domain Permasalahan	Ins	SRLA		RLLA	
		Median	Poin	Median	Poin
SAT	3	13	1	10	2
	5	18	1	12	2
	4	9	1	6	2
	10	15	1	14	2
	11	14	2	14	2
BinPacking	7	0.182416281	1	0.179959314	2
	1	0.076787151	1	0.076116693	2
	9	0.044649806	2	0.049960718	1
	10	0.127499378	1	0.127199467	2
	11	0.074517505	2	0.085738756	1
PersonnelScheduling	5	30	1	28	2
	9	31713	1	10232	2
	8	3491	1	3376	2
	10	2015	2	2090	1
	11	370	2	410	1
FlowShop	1	6380	2	6403	1
	8	27000	2	27050	1

Domain Permasalahan	Ins	SRLA		RLLA	
		Median	Poin	Median	Poin
	3	6441	2	6448	1
	10	11562	2	11595	1
	11	26798	2	26827	1
TSP	0	51327.52573	1	50880.82775	2
	8	22520346.55	2	23530136.49	1
	2	7080.019716	2	7105.80593	1
	7	75357.49123	1	7092.303222	2
	6	60191.69561	2	60549.20105	1
VRP	6	108810.6632	1	99856.26228	2
	2	17978.32745	1	17060.6299	2
	5	360081.0111	1	340975.3792	2
	1	27485.61043	1	25122.85746	2
	9	224333.0392	1	203516.7574	2
Total Poin		43		48	
Rata-Rata Poin		7.166666667		8	
% Poin		**72%**		**80%**	

Untuk memudahkan memberi peringkat setiap metode pada *instance* yang sama, maka nilai fungsi *fitness* setiap *instance* diberikan warna sesuai dengan peringkat, warna hijau merupakan peringkat pertama yang mendapat skor 2 poin sedangkan warna merah merupakan peringkat kedua yang mendapat skor 1 poin.

Skor setiap *instance* pada masing-masing metode dijumlahkan untuk mendapatkan total skor. Dari total

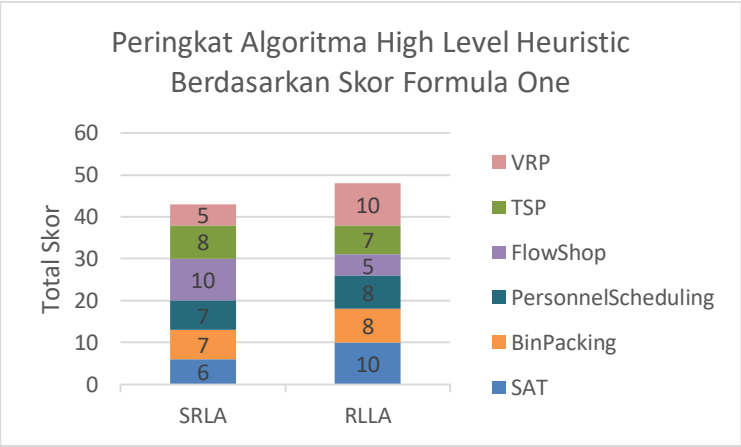
skor dihitung persentase total skor dengan perhitungan seperti persamaan 6.2.

$$\textbf{Total Skor}\ (\%) = (\textbf{total skor}/\textbf{jumlah skor maksimal}) \times \textbf{100}$$

Persamaan 6.2 Perhitungan Total Skor

Setiap metode diberi peringkat berdasarkan hasil perhitungan total skor seluruh *intance*. Peringkat pertama diperoleh oleh metode RL-LA dengan total skor tertinggi yaitu **48** poin dari 60 poin dengan persentase **80%**. Peringkat kedua diperoleh oleh metode SR-LA dengan total skor **43** dari 60 poin dengan persentase **72%**.

Berdasarkan hasil pengujian menggunakan sistem *formula one*, metode yang memiliki kinerja terbaik dalam menghasilkan solusi optimal pada permasalahan optimasi lintas domain adalah RL-LA. Gabungan antara *Reinforcement Learning* sebagai pemilihan LLH dan *Late Acceptance* sebagai move aceptance merupakan algoritma *high level heuristic* yang tepat untuk meningkatkan peforma hiperheuristik dalam penyelesaian permasalahan optimasi lintas domain. Urutan peringkat dari perbandingan algoritma berdasarkan dari hasi perhitungan skor *formula one* pada seluruh domain permasalahan dapat dilihat pada Gambar 6.11. Berdasarkan hasil tersebut algoritma RL-LA terutama pada domain permasalahan SAT, *Bin Packing*, *Personnel Scheduling* dan VRP mendapatkan skor yang lebih tinggi, sedangkan *Flow Shop* dan TSP mendapatkan skor lebih rendah.



Gambar 6.11 Peringkat Algoritma High Level Heuristic Berdasarkan Skor *Formula One*

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan rangkuman singkat yang dapat disimpulkan dari penelitian ini. Terdapat saran dari penulis yang diharapkan dapat membantu dalam meningkatkan hasil pada penelitian selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil yang telah diuraikan pada bagian sebelumnya, kesimpulan yang dapat diambil adalah,

1) Berdasarkan penilaian boxplot dan Formula One, algoritma seleksi LLH dengan menggunakan *Reinforcement Learning* dan *Late Acceptance* sebagai *move acceptance* dapat meningkatkan kinerja hiperheuristik dalam menyelesaikan permasalahan optimasi lintas domain. Terdapat 4 dari enam domain permasalahan yang mengalami peningkatan kinerja, yaitu SAT, *bin packing*, *personnel scheduling*, dan VRP, pada domain lainnya seperti flow shop dan TSP mengalami penurunan. Berbeda dengan penilaian berdasarkan bola FIFA, domain permasalahan yang mengalami penurunan yaitu bin packing dan flowshop sedangkan pada SAT, personnel scheduling, TSP, dan VRP mengalami peningkatan.
2) Berdasarkan hasil penilaian dengan system bola FIFA, seleksi LLH dengan menggunakan Reinforcement Learning pada dua domain yang telah diuji (bin packing

dan flow shop) terjadi penurunan kinerja, tetapi penurunan yang terjadi tidak signifikan.

3) Berdasarkan penilaian *Formula One*, algoritma RL-LA dipilih sebagai strategi *high level heuristic* dalam menyelesaikan permasalahan optimasi lintas domain, karena RL-LA menang dengan skor persentase kinerja sebesar 80% yang diuji coba pada 30 variasi *instance* data dalam enam domain permasalahan. RL-LA unggul 8% dibandingkan dengan SR-LA.

7.2 Saran

Penelitian berikutnya sebaiknya dilakukan uji coba RL-LA dengan menggunakan parameter yang lain agar dapat meningkatkan kinerja pada domain permasalahan yang belum mengalami peningkatan seperti domain bin packing dan flow shop serta dilakukan pada komputer yang memiliki spesifikasi lebih tinggi.

BIODATA PENULIS

Penulis yang memiliki nama lengkap Anang Firdaus ini, lahir di Lamongan pada tanggal 1 Desember 1996. Penulis merupakan anak Pertama dari orang tua bernama Najikh dan Maisyaroh. Penulis menempuh pendidikan dasar di SD Muhammadiyah 18 Surabaya pada tahun 2003 hingga 2009. Setelah lulus dari sekolah dasar, penulis melanjutkan pendidikan formal di bangku sekolah menengah pertama di SMP Negeri 6 Surabaya dan lulus pada tahun 2012. Pada tahun yang sama, penulis melanjutkan pendidikan formal di SMA Negeri 2 Surabaya dan lulus pada tahun 2015. Pada tahun 2015, penulis melanjutkan pendidikan tinggi di Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Insitut Teknologi Sepuluh Nopember (ITS) Surabaya, melalui jalur Seleksi Bersama Masuk Perguruan Tinggi Negeri (SBMPTN) tahun 2015.

Selama menjalani pendidikan tinggi di ITS, penulis aktif dalam berbagai kegiatan, baik organisasi maupun kepanitiaan. Penulis pernah menjadi staf departemen syiar KISI ITS pada tahun 2017. Pada tahun 2017, penulis pernah menjadi staf ahli divisi administrasi Information System Expo (ISE! 2017). Selain aktif dalam organisasi dan kepanitiaan, penulis juga mengikuti beberapa pelatihan diantaranya SAP University Alliance Course yang diadakan Departemen Sistem Informasi. Penulis

juga telah mendapatkan sertifikasi IC3 (Internet Core Competency Certification) pada tahun 2017.

Untuk mengetahui informasi lebih lanjut mengenai penelitian ini maupun terkait dengan penulis, dapat menghubungi melalui email anangfirdaus05@gmail.com.

DAFTAR PUSTAKA

[1] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *Eur. J. Oper. Res.*, vol. 211, no. 3, pp. 427–441, 2011.

[2] G. Ochoa and n.d., *Search-based Approaches and Hyper-heuristics*. .

[3] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Handbook of Metaheuristics," *Handb. Metaheuristics*, 2006.

[4] E. K. Burke and G. Kendall, *Search Methodologies*. 2013.

[5] G. Ochoa *et al.*, "HyFlex: A benchmark framework for cross-domain heuristic search," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7245 LNCS, pp. 136–147, 2012.

[6] A. Mukhlason, A. Djunaidy, and N. Angresti, "Penyelesaian permasalahan optimasi lintas domain dari hyflex menggunakan hiperheuristik yang didasarkan pada metode variable neighborhood search," Institut Teknologi Sepuluh Nopember, 2019.

[7] W. G. Jackson, E. Ozcan, and J. H. Drake, "Late acceptance-based selection hyper-heuristics for cross-domain heuristic search," in *2013 13th UK Workshop on Computational Intelligence, UKCI 2013*, 2013.

[8] F. Hillier and G. Lieberman, *Introduction to Operation Research*. 2010.

[9] M. Madi, D. Markovi, and M. Radovanovi,

"Comparison of Meta-Heuristic Algorithms for," *Facta Univ.*, vol. 11, no. 1, pp. 29–44, 2013.

[10] C. Szepesvári, "Algorithms for Reinforcement Learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.

[11] R. Sutton and A. Barto, *Reinforcement Learning, An Introduction*. .

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

[13] M. Y. Vardi, "Boolean satisfiability," *Commun. ACM*, vol. 57, no. 3, pp. 5–5, 2014.

[14] P. Toth and D. Vigo, "1. An Overview of Vehicle Routing Problems," *Veh. Routing Probl.*, pp. 1–26, 2011.

[15] J. Van Den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *Eur. J. Oper. Res.*, 2013.

[16] Iwan Halim Sahputra, Tanti Octavia, and Agus Susanto Chandra, "Tabu Search Sebagai Local Search Pada Algoritma Ant Colony Untuk Penjadwalan Flowshop," *J. Tek. Ind.*, 2009.

[17] A. Perdana, "Analisis Perbandingan Metode Genetic Algorithm dan Particle Swarm Optimization dalam Menilai Tingkat Optimasi Hasil Pada Bin Packing Problem Satu Dimensi," *Pros. SNASTIKOM 2017*, pp. 1–6, 2017.

[18] E. K. Burke and Y. Bykov, "The late acceptance Hill-

Climbing heuristic," *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 70–78, 2017.

[19] S. Puspitorini, "Penyelesaian Masalah Traveling Salesman Problem dengan Jaringan Saraf Self Organizing," *Media Inform.*, vol. 6, no. 1, pp. 39–55, 2017.

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

[21] M. Harman and F. Chicano, "Search Based Software Engineering (SBSE)," *J. Syst. Softw.*, vol. 103, p. 266, 2015.

[22] E. K. Burke *et al.*, "A Classification of Hyper-heuristic Approaches A Classification of Hyper-heuristic Approaches," *Origins*, pp. 1–54, 2009.

[23] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intell. Data Anal.*, vol. 12, no. 1, pp. 3–23, 2018.

[24] N. R. Sabar, M. Ayob, G. Kendall, S. Member, R. Qu, and S. Member, "Automatic Design of a Hyper-Heuristic Framework With Gene Expression Programming for Combinatorial Optimization Problems," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 309–325, 2015.

[25] S. S. Choong, L. P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Inf. Sci. (Ny).*, vol. 436–437, pp. 89–107, 2018.

[26] M. H. G. Ochoa, "The Cross-domain Heuristic Search Challenge (CHeSC 2011)," 2011. [Online]. Available:

http://www.asap.cs.nott.ac.uk/chesc2011/.

[27] B. McCollum *et al.*, “The Cross-Domain Heuristic Search Challenge – An International Research Competition,” 2011.

[28] M. Hyde and G. Ochoa, “A HyFlex Module for the MAX-SAT Problem,” *Univ. Nottingham ...*, pp. 3–6, 2011.

[29] M. Hyde and G. Ochoa, “A hyflex module for the one dimensional bin-packing problem,” *Sch. Comput. ...*, pp. 1–5, 2009.

[30] J. Vázquez-Rodrıguez and G. Ochoa, “A hyflex module for the permutation flow shop problem,” *Sch. Comput. ...*, pp. 1–4, 2009.

[31] T. Curtois, G. Ochoa, M. Hyde, and J. A. Vazquez-Rodriguez, “A HyFlex Module for the Personnel Scheduling Problem,” *Sch. Comput. Sci. Univ. Nottingham*, pp. 1–12, 2009.

[32] T. Nottingham and N. E. User, “Walker , James D . (2015) Design of vehicle routing problem domains for a hyper-heuristic framework . PhD Design of Vehicle Routing Problem Domains for a Hyper-Heuristic Framework,” 2015.

[33] A. Muklason, “Hyper-heuristics and fairness in examination timetabling problems,” 2017.

LAMPIRAN A: Hasil Eksekusi SR-LA

Tabel A.1 Hasil Eksekusi SR-LA pada Domain SAT (60000ms)

Domain		SAT				
instance		3	5	4	10	11
run ke	1	18	10	11	11	10
	2	12	24	9	17	13
	3	9	16	6	15	14
	4	23	29	8	15	18
	5	20	19	4	19	19
	6	9	12	13	14	23
	7	13	44	9	17	14
	8	8	15	16	40	16
	9	9	18	7	8	13
	10	14	243	12	15	22
	11	21	46	10	12	10
	12	7	18	9	19	19
	13	19	41	15	9	11
	14	16	25	8	23	19
	15	8	16	9	16	12
	16	9	9	11	18	12
	17	16	11	15	12	19
	18	11	55	13	22	13
	19	12	13	10	16	13
	20	9	49	7	16	13

	21	9	12	9	16	12
	22	17	22	8	15	14
	23	14	12	6	14	17
	24	13	15	13	19	19
	25	15	20	7	23	11
	26	15	10	39	9	15
	27	7	21	11	15	14
	28	15	45	143	15	46
	29	25	19	10	14	16
	30	11	14	8	15	13
	31	9	15	9	25	15
min		7	9	4	8	10
q1		9	13.5	8	14	13
med		13	18	9	15	14
q3		16	27	12.5	18.5	18.5
max		25	243	143	40	46

Tabel A.2 Hasil Eksekusi SR-LA pada Domain BinPacking (60000ms)

Domain		BinPacking				
instance		7	1	9	10	11
run ke	1	0.181919179	0.07671722	0.04869602	0.127539556	0.074517505
	2	0.177403486	0.075286783	0.044935346	0.127265422	0.078070748
	3	0.180071959	0.071586382	0.049063074	0.127271556	0.075480298
	4	0.180768211	0.076762085	0.042888575	0.127291378	0.072531809
	5	0.194771922	0.075929858	0.05308235	0.127560622	0.080214585
	6	0.188329441	0.072430087	0.044632853	0.127471111	0.072604779
	7	0.180024041	0.085352201	0.041884628	0.127915289	0.077093542
	8	0.184283876	0.076843181	0.044920942	0.127578489	0.075280708
	9	0.188209847	0.075206951	0.04772948	0.127417244	0.084486626
	10	0.188351366	0.080720567	0.043912916	0.127236089	0.072531912
	11	0.178117237	0.072670995	0.041647336	0.127642933	0.071418252
	12	0.182416281	0.076316377	0.040858245	0.127705244	0.078239805
	13	0.188018685	0.079809194	0.048856661	0.127697067	0.091357753
	14	0.178221532	0.076580937	0.039695278	0.127416711	0.083176966

	15	0.184280137	0.076729226	0.044614903	0.127531644	0.068236321
	16	0.183248911	0.084168973	0.046634932	0.1275376	0.078683401
	17	0.177507478	0.081252009	0.045999502	0.126133422	0.071831208
	18	0.186104872	0.072600897	0.044649806	0.127610222	0.075381683
	19	0.184499291	0.087946458	0.04185	0.127082044	0.092808075
	20	0.173680589	0.085135639	0.045942027	0.127420711	0.070536368
	21	0.181770024	0.075364086	0.049071144	0.127720267	0.078268289
	22	0.188203293	0.076787151	0.043921341	0.127612444	0.072723998
	23	0.187815484	0.080023956	0.044598504	0.126133333	0.066243682
	24	0.174916217	0.079855004	0.044771136	0.127499378	0.0715752 7
	25	0.180178973	0.080219806	0.044947091	0.127573689	0.072437256
	26	0.1775499	0.080328658	0.042872268	0.127231733	0.066186746
	27	0.186613935	0.0710942 2	0.044914183	0.127568533	0.069008805
	28	0.182399941	0.080980 93	0.040490894	0.1274696	0.092302 49
	29	0.187967 22	0.079794904	0.041686848	0.127771467	0.068020041
	30	0.188638782	0.084859119	0.042927288	0.127476356	0.073372522
	31	0.182105798	0.076365456	0.048697347	0.126987911	0.078369314

min	0.173681	0.071094	0.039695	0.126133	0.066187
q1	0.180048	0.075647	0.04288	0.127281	0.071703
med	0.182416	0.076787	0.04465	0.127499	0.074518
q3	0.187891	0.080525	0.046317	0.127594	0.078319
max	0.194772	0.087946	0.053082	0.127915	0.092808

Tabel A.3 Hasil Eksekusi SR-LA pada Domain Personnel Scheduling (60000ms)

Domain		**Personnel Scheduling**				
instance		5	9	8	10	11
run ke	1	42	10539	3491	2900	375
	2	35	14755	3775	2310	385
	3	36	16835	4256	2124	370
	4	33	13849	3644	2533	345
	5	36	31922	3575	2230	340
	6	32	49350	4102	1708	400
	7	43	22143	3598	2200	360
	8	33	42547	3378	2120	395
	9	36	34751	3896	1558	390
	10	36	15923	3458	2187	330
	11	24	29974	3621	1815	415
	12	29	18870	3573	1935	320
	13	36	37521	3629	1860	375
	14	23	67572	3833	1759	360
	15	32	41653	4300	1799	395
	16	29	87457	3874	2015	1720

	17	41	11888	3618	2080	345
	18	28	19157	3376	1805	371
	19	29	16642	3432	1830	355
	20	21	59719	3273	1720	370
	21	30	53995	3479	2005	350
	22	28	15974	3327	2180	415
	23	28	29017	3292	3165	390
	24	30	12235	3387	2055	330
	25	30	34849	3352	2150	390
	26	27	31713	3541	2000	390
	27	32	36521	3347	2955	365
	28	29	35682	3303	1654	390
	29	19	38081	3287	1810	330
	30	23	51095	3361	1695	365
	31	25	26108	3300	2163	345
min		19	10539	3273	1558	320
q1		28	16738.5	3356.5	1807.5	347.5
med		30	31713	3491	2015	370
q3		35.5	39867	3636.5	2183.5	390
max		43	87457	4300	3165	1720

Tabel A.4 Hasil Eksekusi SR-LA pada Domain FlowShop (60000ms)

Domain		FlowShop				
instance		1	8	3	10	11
run ke	1	6372	26997	6463	11519	26796
	2	6391	27008	6448	11559	26831
	3	6380	26917	6463	11539	26802
	4	6383	27048	6467	11570	26828
	5	6380	27004	6450	11576	26817
	6	6374	26990	6429	11593	26812
	7	6380	27007	6427	11572	26833
	8	6384	26951	6441	11565	26795
	9	6360	26982	6459	11542	26764
	10	6381	26989	6426	11534	26888
	11	6375	27000	6452	11602	26816
	12	6390	27035	6422	11583	26797
	13	6394	26903	6449	11584	26799
	14	6380	26967	6408	11561	26854
	15	6352	26943	6429	11595	26775
	16	6389	26948	6409	11562	26743
	17	6364	26988	6454	11559	26837
	18	6395	27051	6452	11559	26746
	19	6387	27018	6426	11560	26792
	20	6382	27002	6443	11568	26790
	21	6340	27035	6442	11560	26897
	22	6385	27003	6444	11552	26770
	23	6396	27027	6438	11571	26863
	24	6400	26988	6427	11556	26750

	25	6390	27041	6423	11529	26794
	26	6371	26963	6415	11575	26824
	27	6387	27050	6412	11564	26823
	28	6349	27013	6444	11540	26772
	29	6356	27029	6408	11578	26798
	30	6367	26983	6445	11586	26772
	31	6356	26983	6430	11517	26790
min		6340	26903	6408	11517	26743
q1		6369	26982.5	6426	11554	26782.5
med		6380	27000	6441	11562	26798
q3		6388	27022.5	6449.5	11575.5	26826
max		6400	27051	6467	11602	26897

Tabel A.5 Hasil Eksekusi SR-LA pada Domain TSP (60000ms)

Domain		TSP				
instance		0	8	2	7	6
run ke	1	50062.14	2.12E+07	7063.076	70856.33	57544.37
	2	50246.94	2.53E+07	7084.929	80249.32	62103.7
	3	50653.08	2.49E+07	7083.101	77496.96	60817.21
	4	52282.16	2.50E+07	7041.072	79530.01	59864.93
	5	49796.16	2.12E+07	7034.58	72018.65	59273.18
	6	50581.81	2.52E+07	7088.951	79382.6	61917.72
	7	51074.05	2.24E+07	7080.02	79822.45	59660.33
	8	51146.65	2.50E+07	7064.263	79486.51	62452.12
	9	52120.87	2.21E+07	7125.723	74064.19	57596.3
	10	51403.51	2.50E+07	7060.865	78829.72	60218.68
	11	52068.67	2.15E+07	7079.342	72976.2	61029.86
	12	51511.11	2.13E+07	7081.375	70617.5	58060.68
	13	49632.41	2.52E+07	7052.069	78532.88	62180.67
	14	51327.53	2.24E+07	7079.387	75357.49	57026.08
	15	50634.36	2.50E+07	7080.479	80337.03	60771.73
	16	51044.32	2.15E+07	7061.703	71399.92	55139.37
	17	51586.98	2.12E+07	7040.221	69915.85	56245.27
	18	51481.59	2.48E+07	7111.381	77978.06	65880.69
	19	51754.96	2.23E+07	7117.6	72923.52	58419.82
	20	52025.94	2.31E+07	7108.543	78809.49	59969.26
	21	51058.25	2.48E+07	7117.677	79880.42	63087.88
	22	51617.51	2.23E+07	7079.831	73824	60381.49
	23	51870.71	2.53E+07	7092.211	78164.92	61355.8
	24	51052.24	2.48E+07	7128.255	78996.09	60549.2

	25	50293.19	2.14E+07	7003.406	72500.18	56736.54
	26	50084.32	2.15E+07	7026.727	72126.5	57750.64
	27	50492.81	2.50E+07	7121.956	80365.07	59896.94
	28	51480.98	2.18E+07	7034.914	70796.81	59062.43
	29	51842.15	2.49E+07	7100.932	74917.15	61174.33
	30	51713.01	2.25E+07	7128.203	73955.83	60191.7
	31	51849.91	2.24E+07	7037.003	71632.12	60627.63
min		49632.41	21189643	7003.406	69915.85	55139.37
q1		50608.09	21660991	7056.467	72313.34	58240.25
med		51327.53	22520347	7080.02	75357.49	60191.7
q3		51733.98	24973373	7104.737	79189.34	61102.1
max		52282.16	25325737	7128.255	80365.07	65880.69

Tabel A.6 Hasil Eksekusi SR-LA pada Domain VRP (60000ms)

Domain		VRP				
instance		6	2	5	1	9
run ke	1	112056.9	19089.62	351125.7	29568.5	231523.1
	2	117070.1	18066.2	391686.3	28778.44	227613.7
	3	113721.6	18030.98	362525.7	26408.37	219618.3
	4	114264	15991.4	357425.7	27476.58	215462
	5	110545.8	17048.3	383604.8	24176.85	222619.6
	6	109062.6	18016.41	362268.5	27466.04	219345.6
	7	119641.4	17022.06	359771.5	28554.33	228853
	8	107444.5	17223.48	334160.8	30599.98	218850.2
	9	108694.8	16957.88	359933.3	28750.2	235618
	10	103023.4	16984.39	364625.7	25209.45	209705.6
	11	105999.9	18109.88	366363.2	27353.5	231416.1
	12	112473.2	17969.93	346762.4	22953.57	221623.7
	13	103807.6	16059.53	358859.4	27556.59	226745.8
	14	107110.2	17962.69	361263	27669.81	209813.2
	15	115153.4	15878.64	387377.4	27485.61	227652.4
	16	111324.8	16882.01	348752.2	28597.24	214493.1
	17	102113.8	19099.11	360099.7	23059.33	208221.2
	18	114146.8	15983.83	359902.4	25209.2	223081.6
	19	106709	18037.73	372677.6	28557.49	231850.7
	20	107741.9	19190.12	374666.5	25284.92	225743
	21	105143.3	16923.61	332666.3	27515.31	230199.3
	22	105215.1	18112.6	362879.4	29681.67	220797.8
	23	112629.4	18128.43	341219.2	28590.57	224333
	24	103647.9	18115.61	343916.3	29665.77	223320.7

	25	109016.5	18010.71	350770	25228.55	228697.7
	26	108947.5	17075.6	360081	27465.26	233933.8
	27	110274.5	18239.41	371444.5	26270.2	231360.3
	28	108730.6	17978.33	367776.1	26366.73	234352.5
	29	108145.8	17101.82	348753.1	27567.29	200472.3
	30	105941.3	18120.97	373225.1	28607.96	236232.6
	31	108810.7	18001.35	344729.2	24157.27	214362.5
min		102113.8	15878.64	332666.3	22953.57	200472.3
q1		106354.5	17003.23	349761.6	25777.56	219097.9
med		108810.7	17978.33	360081	27485.61	224333
q3		112265	18111.24	367069.7	28593.9	230779.8
max		119641.4	19190.12	391686.3	30599.98	236232.6

LAMPIRAN B: Hasil Eksekusi RL-LA

Tabel B.1 Hasil Eksekusi RL-LA pada Domain SAT (60000ms)

Domain		SAT				
instance		3	5	4	10	11
run ke	1	22	5	6	18	9
	2	11	10	12	16	10
	3	14	7	3	8	20
	4	5	17	10	11	18
	5	10	9	12	8	17
	6	23	4	3	56	17
	7	12	6	13	11	22
	8	8	6	8	27	12
	9	12	30	4	30	8
	10	5	58	45	8	19
	11	10	11	6	19	13
	12	7	18	4	12	18
	13	7	19	5	10	18
	14	5	7	6	10	17
	15	9	12	5	15	13
	16	11	6	6	14	10
	17	5	5	8	17	13
	18	6	14	3	27	13
	19	9	48	3	9	14
	20	12	10	3	38	10
	21	8	76	48	12	25

	22	28	57	2	22	10
	23	13	13	4	11	11
	24	19	8	8	21	10
	25	21	12	9	35	23
	26	8	13	5	12	14
	27	12	61	6	20	19
	28	19	18	6	20	12
	29	8	13	7	9	21
	30	12	36	6	10	14
	31	10	11	7	13	14
min		5	4	2	8	8
q1		8	7.5	4	10.5	11.5
med		10	12	6	14	14
q3		12.5	18.5	8	20.5	18
max		28	76	48	56	25

Tabel B.2 Hasil Eksekusi RL-LA pada Domain BinPacking (60000ms)

Domain		BinPacking				
instance		7	1	9	10	11
run ke	1	0.173571427	0.061829329	0.044619446	0.127710311	0.083069918
	2	0.175514656	0.080606882	0.052783618	0.127427111	0.086205252
	3	0.177532947	0.072245975	0.0509582	0.127391556	0.08958285
	4	0.187937019	0.070955925	0.049993094	0.127172889	0.084672027
	5	0.193885074	0.076143655	0.050055746	0.127560622	0.088634267
	6	0.187853261	0.071455688	0.052869333	0.127363289	0.085657369
	7	0.181462953	0.071833413	0.04974779	0.127509689	0.085006378
	8	0.185642638	0.067953651	0.052798952	0.126988178	0.08777947
	9	0.177583253	0.079976254	0.048067091	0.127110311	0.085483622
	10	0.166755717	0.076616114	0.048929961	0.125825114	0.086564781
	11	0.177926509	0.071595672	0.049955856	0.127199467	0.085973738
	12	0.188531215	0.076116693	0.049043062	0.1274528	0.089466921
	13	0.185524802	0.075411269	0.047834569	0.1269864	0.084152782

	14	0.180497992	0.084772327	0.049814751	0.127228356	0.083629673
	15	0.189794846	0.083971069	0.046004596	0.127050133	0.079552866
	16	0.179603524	0.072186646	0.051826766	0.127117244	0.085738756
	17	0.159912832	0.076046133	0.050941745	0.127255289	0.085096888
	18	0.179959314	0.063438239	0.049841934	0.127500889	0.083394944
	19	0.188036188	0.079626793	0.046944162	0.126910044	0.078909524
	20	0.17114419	0.084624738	0.051111817	0.127345778	0.085798827
	21	0.173627259	0.076683096	0.054059372	0.127076978	0.092261223
	22	0.180196447	0.080556239	0.050046464	0.127036	0.088854423
	23	0.188568591	0.085174843	0.049952541	0.127253867	0.089748448
	24	0.169011943	0.071399736	0.050627333	0.127674578	0.088563187
	25	0.161538723	0.067162328	0.045723029	0.127071467	0.080462558
	26	0.175692962	0.079438928	0.048082024	0.126721422	0.085406173
	27	0.189032523	0.071152072	0.050113646	0.127104622	0.082272305
	28	0.186339023	0.080519044	0.053919618	0.127306844	0.086016165
	29	0.185873836	0.080208668	0.049023604	0.126957867	0.087737596
	30	0.179795959	0.072127949	0.049960718	0.127014311	0.087777025

	31	0.164180813	0.076153765	0.050861016	0.127761956	0.082528922
min		0.159913	0.061829	0.044619	0.125825	0.07891
q1		0.174571	0.071526	0.048977	0.127043	0.083891
med		0.179959	0.076117	0.049961	0.127199	0.085739
q3		0.187096	0.080092	0.05095	0.127409	0.087778
max		0.193885	0.085175	0.054059	0.127762	0.092261

Tabel B.3 Hasil Eksekusi RL-LA pada Domain Personnel Scheduling (60000ms)

Domain		**Personnel Scheduling**				
instance		5	9	8	10	11
run ke	1	40	10236	3336	1673	1760
	2	33	9852	3323	2425	370
	3	34	10784	3520	2047	395
	4	31	10075	3385	2110	425
	5	34	10271	3241	2010	410
	6	30	11981	3400	2473	1740
	7	41	12004	3339	2090	385
	8	31	11184	3634	1862	320
	9	34	10186	3538	2095	445
	10	34	10672	3698	1910	1710
	11	22	10651	3294	2180	400
	12	27	9992	3374	1978	460
	13	34	10221	3538	2725	420
	14	21	11067	3203	2120	380
	15	30	10215	3454	1955	410

	16	27	10101	3342	2228	440
	17	39	10746	3222	2219	375
	18	26	9907	3422	2365	390
	19	27	10831	3261	2218	405
	20	19	10193	3411	1900	405
	21	28	9749	3475	2018	385
	22	26	11391	3342	1825	455
	23	26	22675	3421	2090	380
	24	28	10099	3445	3080	410
	25	28	10243	3272	1955	516
	26	25	10063	3272	2125	455
	27	30	9968	3431	1664	385
	28	27	10232	3376	2065	450
	29	17	10134	3385	2230	395
	30	21	9787	3372	2150	375
	31	23	10653	3326	1893	445
min		17	9749	3203	1664	320
q1		26	10087	3324.5	1955	385
med		28	10232	3376	2090	410
q3		33.5	10765	3438	2218.5	447.5
max		41	22675	3698	3080	1760

Tabel B.4 Hasil Eksekusi RL-LA pada Domain FlowShop (60000ms)

Domain		FlowShop				
instance		1	8	3	10	11
run ke	1	6412	27085	6506	11602	26821
	2	6407	27034	6455	11588	26819
	3	6377	27091	6400	11624	26813
	4	6409	27056	6429	11616	26820
	5	6361	27021	6439	11576	26808
	6	6389	27050	6445	11575	26821
	7	6398	27098	6458	11586	26830
	8	6398	27026	6431	11615	26868
	9	6436	26994	6501	11611	26824
	10	6376	26983	6500	11595	26868
	11	6408	26998	6434	11552	26876
	12	6401	26999	6458	11581	26775
	13	6429	27043	6478	11677	26807
	14	6456	26981	6408	11591	26859
	15	6431	27052	6416	11608	26888
	16	6406	27035	6448	11602	26913
	17	6342	27108	6451	11530	26819
	18	6413	27030	6438	11595	26753
	19	6394	27055	6449	11555	26870
	20	6403	27020	6451	11532	26813
	21	6398	27077	6419	11585	26897
	22	6399	26983	6429	11544	26823
	23	6359	27201	6512	11604	26863

	24	6435	27051	6484	11581	26827
	25	6413	27054	6461	11611	26843
	26	6394	27044	6421	11578	26811
	27	6394	27057	6448	11620	26866
	28	6405	27051	6425	11665	26904
	29	6436	27054	6462	11629	26840
	30	6450	26975	6418	11634	26889
	31	6402	27163	6471	11605	26814
min		6342	26975	6400	11530	26753
q1		6394	27020.5	6429	11579.5	26816.5
med		6403	27050	6448	11595	26827
q3		6413	27056.5	6461.5	11613	26868
max		6456	27201	6512	11677	26913

Tabel B.5 Hasil Eksekusi RL-LA pada Domain TSP (60000ms)

Domain		**TSP**				
instance		0	8	2	7	6
run ke	1	50734.6	2.23E+07	7206.139	7080.353	61342.6
	2	52583.32	2.53E+07	7105.806	7056.348	62103.7
	3	51168.8	2.49E+07	7092.282	7126.5	60817.21
	4	49811.07	2.50E+07	7097.752	7135.63	59864.93
	5	50520.03	2.51E+07	7203.99	7153.595	64288.85
	6	50249.86	2.24E+07	7155.243	7056.352	60386.6

	7	49112.41	2.50E+07	7060.189	7053.248	60353.51
	8	50834.95	2.50E+07	7260.983	7162.38	62452.12
	9	51661.28	2.22E+07	7105.576	7115.9	57833.54
	10	49902.77	2.50E+07	7331.888	7092.303	60218.68
	11	51288	2.13E+07	7094.406	7094.406	59308.67
	12	51363.26	2.22E+07	7093.137	7069.915	60010.92
	13	50995.86	2.52E+07	7238.102	7075.825	62180.67
	14	50199.58	2.52E+07	7210.832	7099.635	59684.69
	15	53109.51	2.35E+07	7130.058	7139.08	60771.73
	16	51339.77	2.53E+07	7050.212	7103.485	60880.63
	17	50776.03	2.35E+07	7156.131	7079.194	61426.09
	18	52064.8	2.12E+07	7094.406	7107.764	62207.96
	19	50880.83	2.29E+07	7075.671	7126.538	60592.97
	20	50160.31	2.50E+07	7188.013	7075.769	59969.26
	21	52148.71	2.24E+07	7028.791	7091.439	60947.7
	22	51142.09	2.19E+07	7135.653	7135.291	62389.75
	23	51322.02	2.53E+07	7253.735	7073.573	61355.8
	24	51587.49	2.48E+07	7004.932	7118.191	60549.2
	25	49780.6	2.23E+07	7041.808	7084.446	56736.54
	26	50621.65	2.15E+07	7106.789	7084.917	55952.7
	27	50393.66	2.50E+07	7069.174	7125.432	59896.94
	28	51264.49	2.18E+07	7086.587	7086.587	59062.43
	29	50525.48	2.15E+07	7061.945	6996.709	56157.14
	30	52500.77	2.53E+07	7295.178	7084.618	63328.78
	31	50632.16	2.14E+07	7160.859	7121.023	59395.54
min		49112.41	21190321	7004.932	6996.709	55952.7
q1		50456.85	22201046	7081.129	7077.51	59774.81

med	50880.83	23530136	7105.806	7092.303	60549.2
q3	51351.52	25024672	7196.002	7123.228	61390.95
max	53109.51	25337433	7331.888	7162.38	64288.85

Tabel B.6 Hasil Eksekusi RL-LA pada Domain VRP (60000ms)

Domain		VRP				
instance		6	2	5	1	9
run ke	1	102186.1	15797.92	326629.3	22881.73	207773.7
	2	102200.8	15995.16	339123.2	24036.5	198156.7
	3	104446.1	16912.52	377480.3	23136.43	227753.4
	4	90569.7	17155.11	323516.6	26261.33	199741.6
	5	103511.8	16880.07	347123.3	22894.07	219571.8
	6	95501.32	16737.9	375938.7	23156.26	181683.9
	7	100066.2	18088.84	255161	25151.16	227228.8
	8	94661.38	17104.88	333346.2	25113.48	197981.4
	9	99856.26	16927.58	320376.9	24148.34	220332.9
	10	97430.6	16968.78	348277.6	27508.21	181740
	11	93975.63	18111.67	284131.3	22085.12	203516.8
	12	97887.35	17076.62	349650.3	28513.45	180603.9
	13	108467.1	18114.05	347967.6	21916.35	223751.7
	14	97091.97	18243.57	333164.7	25558.01	178383.3
	15	109688.5	17100.33	311709.7	26146.77	207393.9
	16	99660.82	17995.73	340975.4	26463.96	198553.7
	17	98626.2	17921.82	380042	27385.82	185798.8
	18	109351.8	18235.85	377409.2	22994.92	215979.7
	19	106386.2	16830.03	355090.8	24235.14	201732.7
	20	96004.84	18051.74	348624	26183.73	224315.6
	21	98353.92	15759.3	284074.1	24308.82	205962.9
	22	95665.17	17974.07	341958.8	27444.62	207111.9
	23	102109.3	17157.78	313135.1	25122.86	196555.7

	24	101518.1	15800.72	304578.6	26378.58	197738.1
	25	94608.33	15805.96	344402.5	24015.7	178289.3
	26	95154.55	17060.63	347716	25199.55	206325.5
	27	105853.4	16969.3	311139.3	25298.23	202689.6
	28	101328.5	15846.73	348225.7	26436.09	216315.6
	29	102752.8	16911.83	330538	24957.26	210222.9
	30	97777.2	15928.66	283784.9	24029.63	183991.1
	31	100342.9	17198.11	359842.9	28588.63	214499.1
min		90569.7	15759.3	255161	21916.35	178289.3
q1		96548.4	16783.96	316756	24022.67	197146.9
med		99856.26	17060.63	340975.4	25122.86	203516.8
q3		102476.8	17947.94	348450.8	26319.96	215239.4
max		109688.5	18243.57	380042	28588.63	227753.4

LAMPIRAN C: PERBANDINGAN SR-LA DAN RL-LA

Tabel C.1 Perbandingan SR-LA dan RL-LA

Domain Permasalahan	Ins	Metode HH	Min	Q1	Median	Q3	Maks	Skor	SRLA	RLLA
SAT	3	SRLA	7	9	13	16	25	1	3	20
		RLLA	5	8	10	12.5	28	4		
	5	SRLA	9	13.5	18	27	243	0		
		RLLA	4	7.5	12	18.5	76	5		
	4	SRLA	4	8	9	12.5	143	0		
		RLLA	2	4	6	8	48	5		
	10	SRLA	8	14	15	18.5	40	2		
		RLLA	8	10.5	14	20.5	56	2		
	11	SRLA	10	13	14	18.5	46	0		

		RLLA	8	11.5	14	18	25	4		
BinPacking	7	**SRLA**	0.173680589	0.180048	0.182416281	0.187891352	0.19477192	0	9	16
		RLLA	0.159912832	0.174570957	0.179959314	0.187096142	0.19388507	5		
	1	**SRLA**	0.07109422	0.075646972	0.076787151	0.080524613	0.08794646	0		
		RLLA	0.061829329	0.07152568	0.076116693	0.080092461	0.08517484	5		
	9	**SRLA**	0.039695278	0.042880422	0.044649806	0.046317211	0.05308235	5		
		RLLA	0.044619446	0.048976783	0.049960718	0.050949972	0.05405937	0		
	10	**SRLA**	0.126133333	0.127281467	0.127499378	0.127594356	0.12791529	0		
		RLLA	0.125825114	0.127043067	0.127199467	0.127409333	0.12776196	5		

	11	SRLA	0.066186746	0.071703239	0.074517505	0.078318801	0.09280808	4		
		RLLA	0.078909524	0.083891228	0.085738756	0.087778248	0.09226123	1		
Personnel Scheduling	5	SRLA	19	28	30	35.5	43	0	8	16
		RLLA	17	26	28	33.5	41	5		
	9	SRLA	10539	16738.5	31713	39867	87457	0		
		RLLA	9749	10087	10232	10765	22675	5		
	8	SRLA	3273	3356.5	3491	3636.5	4300	0		
		RLLA	3203	3324.5	3376	3438	3698	5		
	10	SRLA	1558	1807.5	2015	2183.5	3165	4		
		RLLA	1664	1955	2090	2218.5	3080	1		
	11	SRLA	320	347.5	370	390	1720	4		
		RLLA	320	385	410	447.5	1760	0		
FlowShop	1	SRLA	6340	6369	6380	6388	6400	5	24	1
		RLLA	6342	6394	6403	6413	6456	0		

	8	SRLA	26903	26982.5	27000	27022.5	27051	5		
		RLLA	26975	27020.5	27050	27056.5	27201	0		
	3	SRLA	6408	6426	6441	6449.5	6467	4		
		RLLA	6400	6429	6448	6461.5	6512	1		
	10	SRLA	11517	11554	11562	11575.5	11602	5		
		RLLA	11530	11579.5	11595	11613	11677	0		
	11	SRLA	26743	26782.5	26798	26826	26897	5		
		RLLA	26753	26816.5	26827	26868	26913	0		
TSP	0	SRLA	49632.41258	50608.08642	51327.52573	51733.9836	52282.1627	1	15	10
		RLLA	49112.41331	50456.84897	50880.82775	51351.51569	53109.5144	4		
	8	SRLA	21189643.04	21660990.88	22520346.55	24973373.46	25325736.7	5		
		RLLA	21190320.6	22201045.56	23530136.49	25024671.54	25337432.7	0		

	2	SRLA	7003.4058	7056.466869	7080.019716	7104.737161	7128.25536	5		
		RLLA	7004.931891	7081.128996	7105.80593	7196.001718	7331.88788	0		
	7	SRLA	69915.8509	72313.34057	75357.49123	79189.34283	80365.0668	0		
		RLLA	6996.708622	7077.509522	7092.303222	7123.227885	7162.37984	5		
	6	SRLA	55139.36884	58240.247	60191.69561	61102.09654	65880.6927	4		
		RLLA	55952.70202	59774.81309	60549.20105	61390.94534	64288.8467	1		
VRP	6	SRLA	102113.8192	106354.4605	108810.6632	112265.042	119641.392	0	0	25
		RLLA	90569.70358	96548.40185	99856.26228	102476.8138	109688.491	5		
	2	SRLA	15878.64252	17003.2267	17978.32745	18111.23811	19190.1176	0		

		RLLA	15759.30365	16783.96179	17060.6299	17947.94307	18243.5664	5		
	5	**SRLA**	332666.3338	349761.5624	360081.0111	367069.6513	391686.288	0		
		RLLA	255160.9657	316756.0244	340975.3792	348450.8122	380042.035	5		
	1	**SRLA**	22953.57458	25777.5594	27485.61043	28593.90306	30599.984	0		
		RLLA	21916.35463	24022.66622	25122.85746	26319.95871	28588.6308	5		
	9	**SRLA**	200472.334	219097.8727	224333.0392	230779.8011	236232.639	0		
		RLLA	178289.3463	197146.8922	203516.7574	215239.3759	227753.405	5		
Total Skor									**59**	**88**

LAMPIRAN D: HASIL UJI COBA PARAMETER RL-LA

Tabel D.1 Hasil Uji Coba Parameter RL-LA

Uji Coba	Parameter						Fungsi Fitness Domain					
	L	upperBound	initialScore	lowerBound	reward	pinalty	SAT	BP	PS	FS	TSP	VRP
1	500	500	250	0	1	-1	42	0.0767172	3491	6372	50062.137	6389.2321
2	1100	20	17	0	1	-2	35	0.0752868	3775	6391	50246.94	6364.8971
3	1100	20	19	0	1	-1	36	0.0715864	4256	6380	50653.079	6395.3928
4	500	15	7	0	1	/2	33	0.0767621	3644	6383	52282.163	6387.5312

5	700	20	19	0	1	-1	36	0.0759299	3575	6380	49796.158	6382.7321
6	1100	20	10	0	5	sqrt(score)	32	0.0724301	4102	6374	50581.808	6340.9021
7	500	20	5	0	1	-1	43	0.0853522	3598	6380	51074.055	6385.6721
8	1100	10	5	0	1	-1	20	0.0668432	3378	6384	51146.655	6296.2321
9	1100	20	17	0	1	-6	36	0.075207	3396	6360	52120.867	6400.4521
10	1100	20	17	0	1	-1	36	0.0807206	3458	6381	51403.513	6390.8751
11	500	512	2	0	1	/2	24	0.072671	3621	6375	52068.666	6371.3181
12	700	20	9	0	1	-1	29	0.0763164	3573	6320	51511.111	6387.1321
13	1100	20	10	0	5	sqrt(score)	36	0.0798092	3629	6394	49632.413	6349.9412