TUGAS AKHIR – IS 184853

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *SELF ADAPTIVE LEARNING – GREAT DELUGE*

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING SELF ADAPTIVE LEARNING-GREAT DELUGE AGORITHM

WIDYA SAPUTRA
NRP 05211540000150

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

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

TUGAS AKHIR – IS 184853

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *SELF ADAPTIVE LEARNING – GREAT DELUGE*

WIDYA SAPUTRA
NRP 05211540000150

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

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

FINAL PROJECT – IS 184853

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING SELF ADAPTIVE LEARNING-GREAT DELUGE AGORITHM

WIDYA SAPUTRA
NRP 05211540000150

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

DEPARTMENT OF INFORMATION SYSTEMS
Faculty Of Information Technology And Communication
Institut Teknologi Sepuluh Nopember
Surabaya
2019

LEMBAR PENGESAHAN

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *SELF ADAPTIVE LEARNING – GREAT DELUGE*

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 :

Widya Saputra
NRP. 05211540000150

Surabaya, Juli 2019

Kepala
Departemen Sistem Informasi

Mahendrawati ER., S.T., M.Sc., Ph.D.
NIP. 197610112006042001

LEMBAR PERSETUJUAN

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *SELF ADAPTIVE LEARNING – GREAT DELUGE*

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 :

Widya Saputra
NRP. 05211540000150

Disetujui Tim Penguji : Tanggal Ujian : Juni 2019
Periode Wisuda : September 2019

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

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

Dr. Retno Aulia Vinarti, S.Kom., M.Kom. **(penguji II)**

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
DEPARTEMEN SISTEM INFORMASI

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN *HYPER-HEURISTIC* MENGGUNAKAN ALGORITMA *SELF ADAPTIVE LEARNING – GREAT DELUGE*

Nama mahasiswa	**: Widya Saputra**
NRP	**: 05211540000150**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing	**: Ahmad Muklason, S.Kom., M.Sc., Ph.D.**

ABSTRAK

Di literatur, hampir semua permasalahan optimasi dalam kelas NP-hard diselesaikan dengan pendekatan meta-heuristics. Akan tetapi, pendekatan ini memiliki kekurangan dimana diperlukan parameter tuning untuk setiap domain permasalahan yang berbeda maupun instance yang berbeda pada permasalahan yang sama. Pendekatan ini dirasa kurang efektif dalam penyelesaian permasalahan tersebut. Oleh karena itu, diperlukan pendekatan baru, yaitu pendekatan hyper-heuristics yang mampu menyelesaikan permasalahan lintas domain tersebut. Hyper-heuristic merupakan salah satu metode pencarian approximate dimana mampu memberikan solusi permasalahan NP-hard dimana memberikan hasil yang cukup baik dan dapat diterima dalam waktu yang relatif cepat. Metode ini memiliki dua sifat ruang pencarian, yakni pemilihan LLH dan penerimaan solusi (move acceptance). Pendekatan ini bekerja pada domain barier daripada langsung bekerja pada domain permasalahan. Dengan sifat tersebut, hyper-heuristic mampu menyelesaikan permasalahan pada domain yang berbeda. Selain itu, hyper-heuristic memiliki mekanisme pembelajaran melalui umpan balik dari solusi yang telah dihasilkan sebelumnya. Tugas akhir ini menerapkan algoritma hyper-heuristic pada enam domain permasalahan optimasi

kombinatorial, yaitu Satisfiability, Bin Packing, Flow Shop, Personnel Scheduling, TSP, dan VRP. Metode yang akan digunakan dalam pengerjaan tugas akhir ini adalah Self Adaptive – Great Deluge (SADGED). Mekanisme Self Adaptive digunakan untuk melakukan seleksi LLH yang akan digunakan, sedangkan Great Deluge digunakan dalam penentuan penerimaan solusi (move acceptance) dalam kerangka hyper-heuristic. Hasil uji coba dibandingkan dengan Algoritma yang sudah ada dan digunakan sebelumnya yaitu Simple Random – Simulated Annealing. Dari hasil pengujian, Penerapan algoritma SADGED tersebut mampu memberikan hasil yang lebih baik dan lebih optimal. Algoritma SADGED mampu memberikan kinerja yang lebih baik berdasarkan nilai median sebesar 83% dibandingkan dengan algoritma Simple Random – Simulated Annealing.

Kata Kunci : *Meta-heuristic, Hyper-heuristic, Self-adaptive Learning, Great Deluge*, Optimasi Lintas Domain

COMPARISON OF CROSS DOMAIN OPTIMIZATION PROBLEM SOLVING METHODS WITH HYPER-HEURISTIC APPROACH USING SELF ADAPTIVE LEARNING - GREAT DELUGE AGORITHM

Name	**: Widya Saputra**
NRP	**: 05211540000150**
Departmen	**: Information Systems FTIK-ITS**
Supervisor	**: Ahmad Muklason, S.Kom., M.Sc., Ph.D.**

ABSTRACT

Almost all optimization problems in NP-hard classes are solved by the meta-heuristics approach. However, this approach has the disadvantages of requiring tuning parameters for each of the different problem domains and different instances on the same problem. This approach is considered less effective in resolving these problems. Therefore, a new approach is needed, namely a hyper-heuristics approach, that is able to solve cross-domain problems. Hyper-heuristic is one of the approximate search methods which is able to provide solutions for NP-hard problems in polynomial time, and provide good enough results. This method has two search space properties, namely LLH selection and acceptance acceptance (move acceptance). This approach works on the barrier domain rather than directly working on the problem domain. With these properties, hyper-heuristic is able to solve problems in different domains. Moreover, hyper-heuristic has a learning mechanism through feedback from previous solutions. This final project tries to implement a hyper-heuristic algorithm on six combinatorial optimization problem domains, namely Satisfiability, Bin Packing, Flow Shop, Scheduling, TSP, and VRP. The method to be used in this final project is Self Adaptive - Great Deluge (SADGED). The Self Adaptive mechanism is used to conduct

LLH selection to be used, while the Great Deluge is used in determining acceptance (move acceptance) within the hyper-heuristic framework. The results then compared with Simple Random - Simulated Annealing Algorithm. From the test results, SADGED algorithm is able to provide better and more optimal results. The SADGED algorithm is able to provide better performance based on the median value of 83%.compared to Simple Random – Simulated Annealing

Keyword* : *Meta-heuristic, Hyper-heuristic, Self-adaptive Learning, Great Deluge, Cross Domain Optimization

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Allah SWT, berkat Rahmat dan Karunia-Nya penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "Perbandingan Metode Penyelesaian Permasalahan Optimasi Lintas Domain Dengan Pendekatan *Hyper-Heuristic* Menggunakan Algoritma *Self Adaptive Learning – Great Deluge*". Dalam penyelesaian laporan ini, penulis telah banyak mendapat bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ibu Mahendrawati ER., S.T., M.Sc., Ph.D. selaku Kepala Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
2. Bapak Nisfu Asrul Sani, S.Kom., M.Sc. selaku Ketua Prodi S1 Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
3. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku Pembimbing Tugas Akhir yang telah meluangkan waktu untuk memberikan bimbingan selama proses pembuatan laporan ini.
4. Bapak Edwin Riksakomara, S.Kom, M.T. dan Ibu Dr. Retno Aulia Vinarti, S.Kom., M.Kom. selaku dosen penguji sidang Tugas Akhir.
5. Pihak-pihak terkait yang telah membantu hingga selesainya penyusunan laporan Tugas Akhir ini.

Penulis menyadari bahwa laporan ini masih terdapat banyak kekurangan. Untuk itu, penulis mohon maaf jika ada kesalahan dan kekurangan dalam penyusunan laporan ini. Semoga laporan ini dapat memberikan manfaat bagi pembaca dan penulis.

Surabaya, Juni 2019

Penulis

(Halaman Ini Sengaja Dikosongkan)

DAFTAR ISI

DAFTAR TABEL

(Halaman Ini Sengaja Dikosongkan)

DAFTAR GAMBAR

(Halaman Ini Sengaja Dikosongkan)

DAFTAR ALGORITMA DAN KODE PROGRAM

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN

(Halaman Ini Sengaja Dikosongkan)

BAB I
PENDAHULUAN

Bab ini menjelaskan latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat tugas akhir, dan relevansi dari tugas akhir yang dikerjakan sehingga mampu memberi gambaran umum permasalahan dan pemecahan masalah pada tugas akhir.

1.1. Latar Belakang

Optimasi merupakan metode pencarian solusi yang layak (*feasible*) dan paling optimal dari kumpulan solusi yang telah diidentifikasi [1]. Optimasi dapat memberikan model solusi dari permasalahan dalam bentuk fungsi objektif. Optimasi berperan dalam meminimalkan maupun memaksimalkan nilai dari fungsi objektif dari setiap permasalahan. Optimasi sendiri dapat mempermudah dalam pengambilan keputusan di berbagai bidang, seperti transportasi maupun industri.

Seiring berjalannya waktu, permasalahan terkait optimasi semakin bertambah kompleks. Terdapat beragam permasalahan optimasi seperti *satisfiability, flow shop*, *timetabling, vehicle routing problem, bin packing,* maupun *travelling salesman problem* dimana berusaha mencari jarak terpendek, dari satu lokasi ke lokasi lainnya [2]. Permasalahan tersebut dapat dimasukan kedalam kelas NP-hard dimana solusi optimal susah untuk didapatkan karena Kompleksitas dari permasalahan.

Dalam menyelesaikan permasalahan yang semakin kompleks, dibutuhkan algoritma yang mampu memberikan solusi dengan waktu yang relatif cepat. Terdapat beberapa algoritma *exact* yang mampu memberikan solusi optimal yang terbaik, namun

hal ini hanya sesuai dengan tipe permasalahan dengan data yang kecil dan tidak sesuai dengan permasalahan dengan permasalahan yang kompleks karena membutuhkan waktu yang sangat lama. Oleh karena itu, diperlukan algoritma (algoritma *approximate*) yang mampu memberikan hasil yang cukup baik dengan waktu yang tidak terlalu lama. Algoritma *approximate* seperti *heuristic*, *meta-heuristic*, dan *hyper-heuristic* merupakan pilihan dalam menyelesaikan permasalahan optimasi yang kompleks. Algoritma *approximate* memberikan solusi yang tidak menjamin paling optimal, namun cukup baik dan dapat memberikan solusi dalam waktu yang cepat (*polynomial*).

Meta-heuristic merupakan salah satu metode yang mampu menyelesaikan permasalahan tersebut. Metode ini mampu memilih dan memodifikasi *heuristic* sehingga menghasilkan solusi baru atau mengubah solusi saat ini menjadi solusi lainnya [3]. Untuk banyak permasalahan kombinatorial, metode ini menjadi sangat kuat dan memberikan metode yang fleksibel. Namun, metode ini memiliki kekurangan, yaitu pendekatan ini kurang mampu beradaptasi dengan perubahan pada struktur permasalahan atau bahkan *instance* permasalahan yang berbeda dengan struktur yang sama. Pengaturan parameter pada metode ini dapat memberikan solusi yang baik pada satu permasalahan namun belum tentu untuk permasalahan lainnya.

Pada dunia industri, kemampuan adaptasi terhadap perubahan ruang permasalahan sangat penting untuk dimiliki. Pengembangan metode heuristik saat ini telah mampu menyelesaikan permasalahan yang berkembang di masyarakat [4]. *Hyper-heuristic* merupakan *high-level* metodologi dimana dapat mengkombinasikan secara efektif dari beberapa *low level heuristic* (LLH) dan *problem instance* sehingga dapat memberikan solusi dari permasalahan lintas domain. Dengan kata lain, *hyper-heuristic* dapat menentukan *low level heuristic* mana yang akan digunakan serta menentukan apakah akan menerima solusi yang dihasilkan oleh LLH (*move acceptance*). Metode ini bekerja pada ruang kerja *heuristic* sehingga tidak

perlu mengetahui pemahaman khusus terhadap permasalahan yang akan diselesaikan. Jadi, *hyper-heuristic* dapat dikatakan lebih umum untuk menyelesaikan permasalahan *hard combinatorial optimisation problem* karena tidak bergantung pada parameter permasalahan [5].

Dalam pengerjaan tugas akhir ini, penentuan penerimaan solusi dilakukan dengan mengadaptasi pendekatan *Great Deluge* (GD). GD merupakan mekanisme pencarian solusi dimana mendapatkan solusi optimal global berdasarkan nilai dari level parameter yang terus berubah, serta dapat mengatur penerimaan solusi baru yang dihasilkan. Mekanisme ini hampir sama dengan simulated annealing, namun mekanisme ini lebih tidak tergantung terhadap parameter dibandingkan dengan simulated annealing[6]. Berdasarkan beberapa penelitian GD memberikan hasil yang lebih cepat dan tahan terhadap perubahan untuk beberapa permasalahan optimasi yang susah [7]. Oleh karena itu, tugas akhir ini bertujuan menerapkan algoritma GD untuk mekanisme penerimaan solusi. Selain itu, untuk meningkatkan kinerja dari *hyper-heuristic*, dilakukan uji coba terkait mekanisme *self adaptive learning* agar memberikan solusi yang optimal pada permasalahan lintas domain berdasarkan dari solusi – solusi yang telah dihasilkan sebelumnya[8].

Berdasarkan latar belakang tersebut, maka tugas akhir ini mengangkat tema dengan judul "Perbandingan Metode Penyelesaian Permasalahan Optimasi Lintas Domain Dengan Pendekatan *Hyper-Heuristic* Menggunakan Algoritma *Self Adaptive Learning – Great Deluge*".

1.2. Perumusan Masalah

Perumusan masalah yang diangkat pada tugas akhir ini berdasarkan latar belakang masalah diatas, yaitu:

a. Bagaimana penerapan algoritma *Self Adaptif - Great Deluge* (SAD-GED) dengan menggunakan pendekatan Hyper-heuristics?

b. Bagaimana performa SAD-GED yang diusulkan dalam menyelesaikan permasalahan optimasi lintas domain, khususnya jika bandingkan dengan algoritma lainnya?

1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah:

a. Uji coba dilakukan pada enam domain permasalahan optimasi kombinatorial yang ada dalam *framework* HyFlex, yaitu *satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, TSP, dan VRP*.
b. Pengembangan metode dilakukan dengan menggunakan *framework hyper-heuristic.*
c. Aplikasi dibangun dengan menggunakan bahasa pemrograman java.

1.4. Tujuan Tugas akhir

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

a. Menerapkan algoritma SAD-GED dengan pendekatan hyper-heuristics
b. Melakukan analisa performa algoritma SAD-GED.

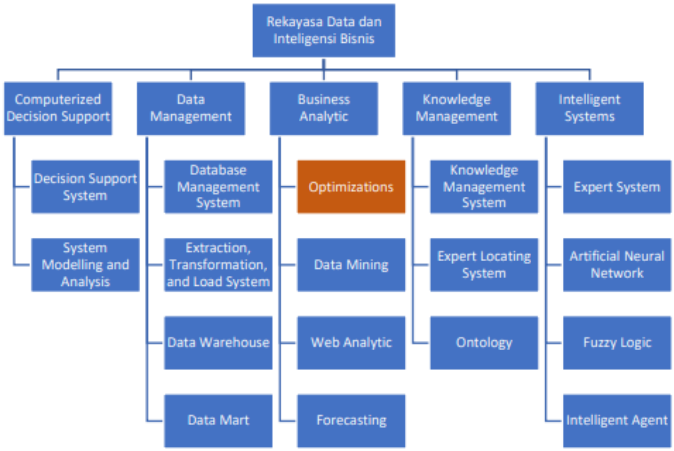
1.5. Manfaat Tugas Akhir

Manfaat yang dapat diberikan dengan adanya penelitian tugas akhir ini, yaitu:

a. Pengerjaan tugas akhir ini diharapkan dapat menambah wawasan dan pengetahuan dalam bidang keilmuan sistem informasi khususnya terkait dengan pengembangan metode *hyper-heuristic self adaptive learning – great deluge* pada permasalahan lintas domain dari HyFlex.
b. Pengerjaan tugas akhir ini diharapkan dapat menghasilkan algoritma yang dapat digunakan dalam permasalahan lintas domain.
c. Pengerjaan 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 untuk tugas akhir ini berkaitan dengan mata kuliah Optimasi Kombinatorial dan Heuristic pada Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB) pada pokok penelitian Business Analytics, khususnya di bidang optimasi. Disamping itu, penelitian ini berkaitan dengan mata kuliah Optimasi Kombinatorial dan Heuristik (OKH).



Gambar 1.1 Roadmap Penelitian Laboratorium Rekayasa Data dan Intelegensi Bisnis

(Halaman Ini Sengaja Dikosongkan)

BAB II
TINJAUAN PUSTAKA

2.1. Studi Sebelumnya

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

Tabel 2.1 Studi Sebelumnya

No	Studi Sebelumnya	
1	Nama Peneliti	Ahmad Muklason, Arif Djunaidy, Nisa Dwi Angresti
	Tahun Penelitian	2019
	Judul Penelitian	Penyelesaian Permasalahan Optimasi Lintas Domain Dari Hyflex Menggunakan *Hyper-heuristic* Yang Didasarkan Pada Metode Variable Neighborhood Search [9]
	Penjelasan Singkat	Penelitian mengkaji permasalahan HyFlex dan mengusulkan penyelesaian permasalahan dengan metode gabungan variable neighborhood search – simulated annealing.
	Hasil Penelitian	Metode yang diusulkan 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.

No	Studi Sebelumnya	
2	Nama Peneliti	Jackson et al
	Tahun Penelitian	2013
	Judul Penelitian	Late Acceptance-Based Selection Hyper-Heuristics For Cross-Domain Heuristic Search [10]
	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 *heuristic* berdasarkan pilihan roulette-wheel dan menggabungkannya dengan metode move acceptance Late Acceptance.
	Hasil Penelitian	Metode simple random – late acceptance memberikan nilai yang baik apabila diterapkan pada memori dengan jumlah yang besar.
3	Nama Peneliti	Cigdem Alabas-Uslu, Berna Dengiz
	Tahun Penelitian	2011
	Judul Penelitian	A Self-Adaptive Local Search Algorithm For The Classical Vehicle Routing Problem [11]
	Penjelasan Singkat	Mengusulkan algoritma self-adaptive local search dalam menyelesaikan permasalahan VRP. Algoritma *self-adaptive* pada metode ini bekerja dengan melakukan penambahan sebuah parameter. Parameter mengatur proses *move acceptance* pada algoritma.

No	Studi Sebelumnya	
	Hasil Penelitian	Algoritma self-adaptive local search dapat dengan mudah mengatur parameter yang digunakan. Selain itu, solusi yang dihasilkan dari algoritma ini mampu bersaing dengan beberapa algoritma *benchmark* yang digunakan dalam menyelesaikan permasalahan VRP. Penambahan *self-adaptive* pada algoritma ini mampu memberikan solusi yang lebih baik bagi algoritma local search.
4	Nama Peneliti	Nabeel R. AL-Milli
	Tahun Penelitian	2010
	Judul Penelitian	Hybrid Genetic Algorithms with Great Deluge For Course Timetabling [6]
	Penjelasan Singkat	Mengusulkan algoritma gabungan antara *Genetic Algorithm* dan *Great Deluge*. *Genetic Algorithm* berperan dalam menentukan solusi awal. Selanjutnya, solusi tadi akan diproses oleh *Great Deluge* untuk melakukan *local search* agar meningkatkan hasil dari solusi tadi.
	Hasil Penelitian	Algoritma menghasilkan solusi yang lebih baik dibandingkan dengan beberapa algoritma lain seperti *Randomised Iterative Improvement*, GA-LS. Disini, penerapan *Great Deluge* mampu melakukan optimasi lokal secara optimal serta meningkatkan solusi awal yang dihasilkan oleh Genetic Algorithm menjadi lebih baik.

2.2. Dasar Teori

Pada bagian ini akan dibahas mengenai dasar teori yang digunakan sebagai dasar informasi untuk pengerjaan tugas akhir.

2.2.1. Permasalahan optimasi kombinatorial

Permasalahan optimasi kombinatorial merupakan permasalahan yang terdapat pada bidang permesinan, perencanaan, dan perindustrian yang dapat dimodelkan dalam bentuk meminimalkan maupun memaksimalkan biaya pada variabel diskrit terbatas. Secara matematis, permasalahan ini merupakan kumpulan *instance* dari permasalahan, dimana setiap *instance* didefinisikan oleh pasangan kumpulan solusi yang dapat diterima dari *instance* tersebut (F) dan nilai dari solusi yang dihitung berdasarkan setiap kumpulan solusi *instance* dan digunakan untuk menentukan performa dari setiap solusi [12]. Dalam permasalahan optimasi, terdapat nilai fungsi tujuan yang akan dimaksimalkan maupun di minimalkan sesuai dengan tujuan yang ingin dicapai dengan berdasarkan batasan – batasan yang ada.

Penyelesaian permasalahan optimasi terdiri dari dua bagian, yaitu *polynomial* (P) dan *non-deterministic polynomial* (NP). Permasalahan *polynomial* dapat diselesaikan dengan algoritma *exact* dan dengan waktu yang relatif cepat, yaitu dalam waktu polinomial. Sedangkan *non-deterministic polynomial* merupakan permasalahan yang tidak dapat diselesaikan dengan algoritma *exact* karena dapat memakan waktu yang relatif lama. Salah satu hal yang menyebabkan permasalahan *non-deterministic polynomial* ini adalah banyak data yang akan dioptimasi. Kita tidak dapat memastikan solusi yang paling optimal pada permasalahan ini, namun kita dapat melakukan validasi solusi dalam waktu polinomial. Dalam kelas NP terdapat subset yang dinamakan *NP-complete*. Sebuah permasalahan dikatakan *NP-complete* jika dia termasuk *NP-hard* dan bagian dari kelas NP. Permasalahan masuk kedalam *NP-hard* jika paling tidak, memiliki tingkat kesusahan sama

dengan permasalahan yang terdapat pada kelas NP. Dengan kata lain, *instance* pada kelas *NP-complete* dapat direduksi ke dalam *instance NP-hard* dalam waktu polinomial. Secara teknis permasalahan *NP-complete* berhubungan dengan pengambilan keputusan, sedangkan permasalahan *NP-hard* berhubungan dengan optimasi [3].

Algoritma approximate merupakan algoritma yang mampu memberikan solusi dengan waktu yang relatif cepat. Berbeda dengan algoritma *exact* yang selalu memberikan nilai optimal dan pasti pada setiap iterasinya, algoritma *approximate* memberikan hasil yang belum tentu optimal, namun layak diterima. Karakteristik dari algoritma *approximate* ini lah yang menyebabkan layak dalam penyelesaian permasalahan *non-deterministic polynomial*, dimana membutuhkan waktu yang lama apabila diselesaikan menggunakan algoritma *exact.* Algoritma *approximate* dapat berupa *heuristic*, *meta-heuristic*, maupun *hyper-heuristic.*

2.2.2. Meta-heuristic

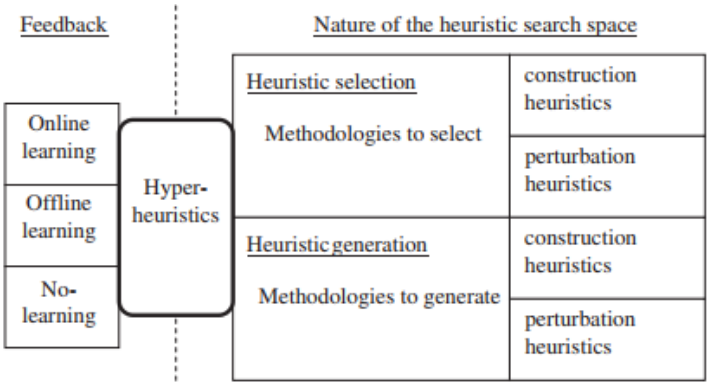
Meta-heuristic merupakan strategi utama yang memandu dan memodifikasi *heuristic* lain untuk menghasilkan solusi diluar dari pencarian optimal lokal. *Heuristic* yang dipandu oleh mekanisme ini mungkin prosedur tingkat tinggi dimana mengubah suatu solusi menjadi solusi lainnya. Metode ini diterapkan karena tidak adanya algoritma khusus yang dapat memberikan hasil yang memuaskan untuk suatu permasalahan. Metode ini mendeskripsikan seluruh proses pencarian, seperti *heuristic* mana yang akan digunakan, bahkan kriteria penerimaan solusi. Untuk banyak permasalahan kombinatorial, metode ini menjadi sangat kuat dan memberikan metode yang fleksibel. *Meta-heuristic* kebanyakan terinspirasi dari proses alam atau ilmu sains, seperti metode *Simulated Annealing, Tabu Search, Genetic Algorithm*, dan lain sebagainya [3]. Metode umumnya juga memiliki sifat menggunakan komponen *stochastic* (menghasilkan variabel acak), serta memiliki parameter yang butuh disesuaikan untuk suatu permasalahan.

Meta-heuristic akan berhasil melakukan optimasi permasalahan jika dapat melakukan penyeimbangan antara eksplorasi (*diversification*) dan eksploitasi (*intensification*) [13]. Eksploitasi dibutuhkan untuk melakukan identifikasi bagian dari pencarian solusi dengan hasil kualitas yang baik. Klasifikasi solusi yang dihasilkan dari proses ini dapat berupa solusi tunggal maupun jamak.

Pendekatan ini bergantung pada nilai parameter sehingga kurang mampu beradaptasi dengan perubahan pada struktur permasalahan atau bahkan *instance* permasalahan yang berbeda dengan struktur yang sama. Pengaturan parameter pada metode ini dapat memberikan solusi yang baik pada satu permasalahan namun belum tentu untuk permasalahan lainnya.

2.2.3. *Hyper-heuristic*

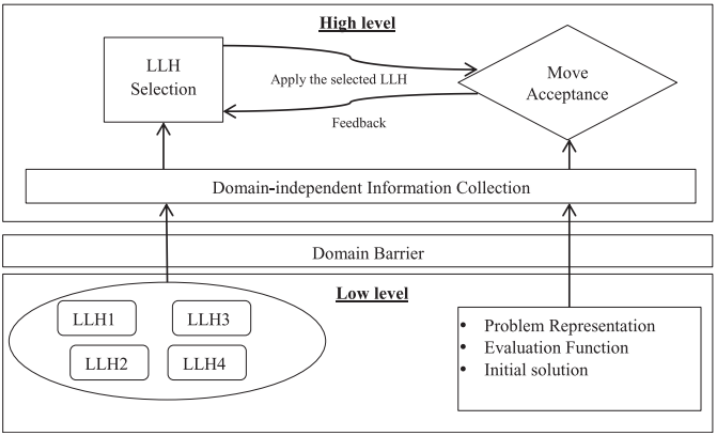
hyper-heuristic merupakan sebuah metode pencarian atau pembelajaran untuk memilih dan menggunakan *heuristic* untuk menyelesaikan permasalahan pencarian komputasi [4]. *Hyper-heuristic* meliputi kumpulan pendekatan dimana bertujuan untuk melakukan otomatisasi, dimana biasanya melakukan penggabungan dengan teknik *machine learning*, prosesnya meliputi memilih dan mengkombinasikan *heuristic* sederhana atau membuat *heuristic* baru dari komponen *heuristic* yang ada saat ini dalam rangka penyelesaian permasalahan optimasi [13]. *hyper-heuristic* adalah Sebuah metodologi yang dapat memberikan sebuah solusi Yang tidak terlalu optimal, namun solusi yang cukup baik dan dapat diterima. *hyper-heuristic* memberikan pendekatan yang bertujuan untuk melakukan desain metode *heuristic* untuk menyelesaikan permasalahan komputasional yang susah. *hyper-heuristic* mendeskripsikan penggunaan *heuristic* untuk memilih *heuristic* 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.



Gambar 2.1 Klasifikasi Pendekatan Hyper-heuristic [4]

hyper-heuristic memiliki dua klasifikasi utama, yakni sifat dari ruang pencarian, dan umpan balik yang berbeda dari sumber informasi. Sifat ruang pencarian *hyper-heuristic* sendiri dibagi menjadi dua, yakni pemilihan *heuristic* dan penerapan *heuristic*. Pada tingkatan kedua pada dimensi terdiri dari *constructive heuristics* dan *perturbative heuristics*. *Constructive heuristics* mempertimbangkan solusi kandidat secara parsial, dimana menghilangkan satu atau lebih solusi dan secara iteratif membangun solus baru, sedangkan *perturbative heuristics* merupakan perubahan terhadap solusi yang telah ada.

Hyper-heuristic dikatakan sebuah algoritma pembelajaran apabila menggunakan *feedback* dari proses pencarian solusi yang dilakukan. Berdasarkan dimensi *feedback*, terdapat 3 pembagian tipe pembelajaran. *Online learning*, pembelajaran dilakukan pada saat algoritma sedang menyelesaikan permasalahan. contoh dari penerapan *online learning* adalah *reinforcement learning* pada algoritma pemilihan *heuristic*.



Gambar 2.2 Struktur Umum Hyper-heuristic

Offline learning, dimana melakukan pengumpulan pengetahuan dalam bentuk aturan – aturan dan program, dari kumpulan *instance* latihan yang diharapkan akan menggeneralisasi untuk menyelesaikan kejadian yang tidak terlihat. Contoh dari penerapan *offline learning* adalah *learning classifier system*, dan *genetic programming* [14]. Sedangkan untuk *No-learning*, Hiperheurisik tidak melakukan pembelajaran.

Hyper-heuristic memiliki struktur umum yang terdiri dari bagian *high-level* dan *low level*. *low level heuristic* merupakan heuristic yang akan dipilih, LLH sendiri merupakan representasi dari permasalahan, fungsi evaluasi dan solusi awal yang berhubungan dengan permasalahan sehingga setiap permasalahan memiliki kumpulan LLH yang berbeda [1]. LLH merepresentasikan lingkungan pencarian lokal sederhana (*move operator*) atau abstraksi dari aturan yang berasal dari ahli untuk membangun solusi (*heuristic* konstruktif) [15]. Sedangkan untuk *high level*, memiliki mekanisme pemilihan LLH (untuk menentukan solusi baru) dan *move acceptance* (menentukan apakah akan menerima solusi atau tidak). *Move acceptance* berfungsi agar algoritma tidak terjebak pada *local optima*, sehingga metode yang baik adalah dengan menerima hasil yang

buruk juga. Selain itu pada bagian tengah *hyper-heuristic* terdapat domai barier. Domain barier bekerja sebagai letak informasi jumlah LLH dan fungsi evaluasi sehingga *hyper-heuristic* independen terhadap domain permasalahan. Selanjutnya, LLH akan melakukan eksekusi pada pencarian solusi.

Tabel 2.2 contoh Metode Seleksi Pemilihan LLH

LLH Selection Method	Description
Simple Random [10]	Melakukan pemilihan LLH secara acak pada saat proses pencarian
Reinforcement Learning [1]	Memberikan reward atau penalti kepada untuk mengukur kinerja LLH
Monte Carlo Tree Search (MCTS) [16]	Metode yang didasari pada mekanisme pohon dan melakukan analisa pada kriteria penerimaan yang paling menjanjikan

Alur proses dari *hyper-heuristic* dimulai dari masukan domain permasalahan, kemudian *high level heuristic* akan mencari LLH yang sesuai, dan solusi akan dihasilkan. Pemilihan LLH berhubungan dengan *perturbative* 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*.

2.2.4. Great Deluge

Great Deluge Algorithm (GD) merupakan algoritma umum yang digunakan untuk permasalahan optimasi [17]. GD merupakan algoritma *local search* yang menyerupai Simulated Annealing. Seperti algoritma *local search* lainnya, algoritma ini secara bertahap melakukan perubahan pada solusi saat ini dengan yang baru sampai dengan kriteria untuk mengakhiri jalannya algoritma. Algoritma ini hampir sama dengan *Simulated Annealing* dan *Hill Climbing*. Algoritma ini memiliki analogi seperti seseorang yang berada pada sebuah tempat dan terus menerus hujan yang menyebabkan banjir yang terus meningkat, maka dia akan berusaha mencari lokasi dimana tidak tergenang air.

Seperti metode *Local Search* lainnya, GD melakukan perulangan pergantian nilai solusi saat ini dengan solusi yang baru, sampai dengan kriteria untuk berhenti tercapai. Dengan kata lain, GD memiliki mekanisme penerimaan solusi dimana lebih buruk daripada solusi saat ini pada saat melakukan perulangan pencarian solusi akhir [18]. Mekanisme ini memiliki aturan jika solusi yang dihasilkan memiliki nilai sama dengan atau di bawah batas atas dari parameter L (*water level*) yang telah ditentukan maka akan diterima [19]. Pada saat melakukan perulangan pencarian solusi akhir, semakin lama waktu pencarian solusi akhir maka nilai solusi yang lebih buruk akan semakin susah untuk diterima karena nilai L yang terus berubah. Parameter L diatur memiliki nilai sama dengan nilai solusi awal yang dihasilkan sebelum melakukan iterasi algoritma dan akan berkurang pada setiap iterasi dengan pengurangan sebesar nilai dari parameter ΔL (*rain speed*) sampai dengan nilai mencapai nol atau kriteria untuk mengakhiri jalannya algoritma. Proses metode *Great Deluge* secara detail dapat dilihat pada Algoritma 2.1.

```
1   Find initial solution s
2   Sbest = f(s)
3   Scurr = f(s)
4   B = f(s)
5   Set decay rate alpha (α)
6   while !hasTimeExpired() do
7      select s′ from neighborhood structure
8      Snew = f(s′)
9      if Snew <= Scurr OR Snew <= B then
10        Scurr ← Sbest
11        If Snew < Sbest then
12           Sbest <- Snew
13        Enf if
14     End if
15     B <- B - α
16     End while
17     Return Sbest
```

Algoritma 2.1 Great Deluge

Dalam Algoritma 2.1 mulai dari baris pertama melakukan inisiasi jumlah perulangan. Baris kedua dan ketiga, menghitung nilai dari solusi awal (*fitness*) dan memasukan kedalam variabel Sbest dan Scurr. Baris keempat, menetapkan parameter level air dengan variabel B, nilai awal dari variabel ini adalah sama dengan solusi awal. Baris kelima menentukan parameter *decay rate* dimana akan mengurangkan parameter B. Baris ke tujuh, mencari solusi baru berdasarkan Neighborhood Search. Baris ke sembilan sampai dengan dua belas merupakan penerimaan solusi. Apabila solusi baru lebih baik dari Scurr atau nilai parameter B maka solusi diterima dan menggantikan nilai Scurr. Jika solusi lebih baik dari Sbest maka nilai akan menggantikan Sbest. Baris ke lima belas merupakan pengurangan parameter B dengan nilai Alpha. Pada baris terakhir, algoritma akan mengembalikan nilai terbaik sebagai solusi akhir yang dihasilkan.

2.2.5. Self-Adaptive Learning

Self-adaptive Learning merupakan metode yang digunakan dengan tujuan agar algoritma dapat secara adaptif memberikan gambaran strategi yang harus digunakan dalam melakukan proses optimasi berdasarkan dari solusi yang telah dihasilkan [8]. Metode ini membantu *hyper-heuristic* dalam menentukan LLH yang akan digunakan dalam melakukan penyelesaian permasalahan (*heuristic selection*). Self-adaptive learning ini melakukan interaksi dengan sistem dimana merespon solusi yang dihasilkan oleh LLH dengan tujuan untuk mendapatkan solusi umum yang lebih baik. Strategi Self-adaptive ini menggunakan mekanisme pencarian, dimana LLH yang memberikan solusi baik akan memiliki kemungkinan untuk lebih sering dipilih daripada LLH yang memberikan nilai solusi yang buruk.

Contoh penerapan Self-adaptive Learning dapat dilihat pada penelitian Quan-Ke et al [20]. Peneliti menentukan parameter sebagai tempat *neighbour list* (NL) dan mengisi parameter tersebut dengan metode yang dipaparkan. Selanjutnya setiap iterasi satu nilai dari NL akan dikeluarkan dan digunakan, apabila solusi yang dihasilkan baik maka akan dimasukan kedalam list parameter penyimpanan baik (WNL). Apabila NL telah kosong, maka akan diisi dengan 75% dari WNL dan sisanya akan diisi dengan metode yang telah dipaparkan oleh peneliti.

Ping-Che et all juga melakukan penelitian lain terkait *Self-adaptive* Learning berbasis *Variable Neighborhood Search* [21]. Sistem peringkat LLH dan Tabu List digunakan dalam metode seleksi LLH sehingga algoritma dapat memberikan hasil yang baik.

2.2.6. Simple Random – Simulated Annealing

Simple Random – Simulated Annealing merupakan metode *hyper-heuristic* dimana Simple Random berperan sebagai metode seleksi LLH dan Simulated Annealing sebagai metode penerimaan solusi. Simple Random merupakan metode

stokastik murni dimana memilih LLH secara acak dari kumpulan LLH yang telah tersedia.

Simulated Annealing merupakan algoritma *meta-heuristic* yang terinspirasi dari proses *annealing* yang dilakukan ahli metalurgi urutan yang baku dari energi yang minimal [13]. Teknik ini dapat dikatakan seperti membawa material dengan suhu tinggi, kemudian secara perlahan mengurangi suhunya. Algoritma ini memiliki satu parameter pengatur yaitu suhu (T) dan satu parameter pengurang suhu yakni alpha. Algoritma ini dimulai dengan membuat solusi awal. Kemudian pada setiap iterasi apabila solusi baru lebih baik daripada solusi lama maka akan diterima. Namun, apabila solusi lebih buruk, penerimaan akan dilakukan berdasarkan parameter T dengan penentuan probabilitas berdasarkan rumus ($e^{-\frac{solusi\ baru - solusi\ lama}{T}}$). Setiap iterasi, suhu akan berkurang sesuai dengan besaran perkalian alpha.

```
1    Find initial solution s randomly
2    Set initial T, @
3    functionValueCurrent = f(s)
4    while !hasTimeExpired() do
5    select s'
6    functionValueNew = f(s')
7    Δfunction = f(s') - f(s)
8    if Δfunction < 0 then
9    s ← s'
10   else if (rand(0,1) < e^(-Δfunciton/T)) then
11   s ← s'
12   f(s)= f(s')
13   end if
14   end if
15   set T = T * @
16   end while
```

Algoritma 2.2 Simulated Annealing

2.2.7. Domain permasalahan

Penerapan metode *hyper-heuristic* dalam tugas akhir ini mencoba menyelesaikan permasalahan lintas domain untuk mendapatkan solusi yang dapat diterima. Permasalahan optimasi lintas domain tersebut antara lain :

a. Boolean Satisfiability Problem (SAT)

Permasalahan ini berusaha menyelesaikan penentuan pemberian variabel *boolean* pada suatu rumus atau formula, dimana nantinya formula tersebut dapat memberikan hasil benar (*true*) [22]. Dengan kata lain, apabila sebuah formula *boolean* memiliki variabel dan dapat diganti dengan nilai benar atau salah dengan cara apapun dan menghasilkan solusi benar. Jika dalam pemberian variabel tersebut dapat memberikan nilai benar maka formula tersebut dikatakan *satisfiable*, sedangkan apabila tidak maka maka disebut dengan *unsatisfiable* dan menghasilkan nilai salah (*false*). MAX-SAT merupakan kasus khusus dari permasalahan SAT. Permasalahan tersebut merupakan NP-hard dimana mencari jumlah maksimal dari klausa yang dapat memberikan hasil yang memuaskan dari penempatan *boolean.*

Tipe SAT yang digunakan adalah klausa konjungsi (*and*) dan disjungsi (*or*). Berikut merupakan contoh persamaan dari SAT $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_2)$, dimana $x_1 = 1, x_2 = 1, x_3 = 1$. Pada persamaan tersebut, hasil memberikan nilai benar(*true*) sehingga dapat dikatakan *satisfiable.*

SAT banyak digunakan dalam permainan teka – teki logika salah satunya sudoku. Pada permainan tersebut, diharuskan mengisi satu sel dengan angka yang berbeda, serta memenuhi kriteria jumlah angka satu deret yang sama.

b. Bin Packing (BP)

Permasalahan ini merupakan pengemasan beberapa barang kedalam sebuah wadah (*bin*) sebanyak mungkin jumlahnya. Setiap barang memiliki berat atau nilai masing – masing. Dilain sisi, wadah memiliki kapasitas yang tetap sehingga tidak semua barang dapat masuk ke dalam wadah tersebut. Tujuan utama dari permasalahan ini adalah meminimalkan penggunaan bin, serta berat dari benda yang dimasukan kedalam wadah tidak melebihi kapasitas dari wadah.

Terdapat banyak variasi dari permasalahan ini, seperti *one dimensional bin packing, 2D packing, linier packing*, dan lain sebagainya. Permasalahan tersebut banyak diterapkan di dunia nyata. *Bin Packing* merupakan salah satu permasalahan *NP-Hard*.
Manajemen rantai pasok merupakan salah satu kasus pada dunia nyata yang berhubungan dengan permasalahan *Bin Packing* [23]. Manajemen ini dilakukan untuk mengatur barang yang akan disimpan maupun dikirim agar penataan barang efektif dan efisien. Penataan yang baik tentunya akan memberikan keuntungan bagi manajemen.

c. Flowshop (FS)

Permasalahan ini mencari urutan pekerjaan yang akan diproses pada sejumlah mesin secara berurutan [24]. Aturan dalam permasalahan ini adalah satu mesin hanya dapat menyelesaikan satu pekerjaan dalam satu waktu, begitu pula sebaliknya satu pekerjaan hanya dikerjakan oleh satu mesin dalam satu waktu. Semua urutan dan aturan diatur oleh sistem dalam permasalahan. setiap pekerjaan tidak dapat melompati pekerjaan lain, serta setiap mesin tidak boleh sampai tidak beroperasi apabila terdapat pekerjaan yang siap diolah.

Secara umum, urutan pekerjaan dan penggunaan mesin ini bertujuan untuk mengalokasikan sumberdaya yang terbatas sesuai dengan pekerjaan yang harus dilakukan. Tujuannya untuk melakukan optimasi satu atau banyak tujuan yang diinginkan.

Permasalahan ini dapat kita lihat pada proses produksi barang di suatu pabrik. Satu pekerjaan diselesaikan oleh mesin pertama, kemudian pekerjaan lainnya diselesaikan oleh mesin lainnya yang tersisa atau menunggu mesin pertama. Kegiatan tersebut berlangsung hingga semua operasi pekerjaan telah selesai.

d. Personnel Scheduling (PS)

Personnel Scheduling merupakan salah satu tugas penting dalam berbagai macam bisnis. Permasalahan ini mencakup area yang luas, kebanyakan dari area tersebut adalah penugasan dan pembagian jam kerja untuk sekelompok orang sesuai dengan perencanaan. Salah satu area permasalahan ini adalah menentukan kapan waktu yang tepat dan hari yang tepat (waktu kerja) untuk setiap pekerja dalam melaksanakan pekerjaan mereka dalam periode perencanaan yang telah ditentukan. Beberapa batasan yang dapat diperhatikan antara lain hari libur, jumlah waktu maksimal atau minimal jam kerja, dan lain sebagainya. Terdapat kemungkinan setiap pekerja melebihi batas waktu kerja yang ditetapkan, hal ini lah yang harus diminimalkan.

Personnel Scheduling dapat dibagi menjadi 4 klasifikasi model, yakni *permanence centered planning, fluctuation centered planning, mobility centered planning, dan project centered planning. permanence centered planning* merupakan penentuan jumlah pekerja. *fluctuation centered planning* seperti

pada manajemen gudang dan distribusi barang. *mobility centered planning* ketika berhubungan dengan transportasi. *project centered planning* dimana muncul pada perusahaan saat akan mengerjakan proyek dan terdapat pembagian pekerjaan [25].
Permasalahan ini dapat kita jumpai pada penjadwalan pekerja perawat di rumah sakit. Penjadwalan ini biasanya dibagi dalam jam kerja (*shift*) dan setiap perawat satu hari bekerja pada satu jam kerja yang telah ditentukan.

e. Traveling Salesman Problem (TSP)

Traveling Salesman Problem (TSP) merupakan sebuah permasalahan dimana mencari jarak terpendek sebuah perjalanan dari beberapa kota yang ingin dikunjungi, kota awal dan akhir adalah kota yang sama dan tepat mengunjungi satu kota pada setiap perjalanannya [3]. Permasalahan TSP ini menjadi permasalahan optimasi kombinatorial mendapat perhatian dari berbagai bidang industri maupun transportasi. Pada kasus jumlah kota sedikit, permasalahan ini dapat diselesaikan dengan mudah. Namun, apabila kota yang ingin dikunjungi ada banyak, maka permasalahan ini akan menjadi susah untuk diselesaikan sehingga permasalahan ini masuk ke dalam permasalahan NP-Complete.

Permasalahan ini dapat dimodelkan menggunakan grafik G = (V, A), di mana himpunan simpul $V=V_1,...,V_n$ yang mewakili kota-kota dan A adalah himpunan arc (*node*) $A=\{(V_i,V_j)|V_i,V_j \in V, i \neq j\}$ sesuai dengan jalur antara dua kota, dengan masing-masing sisi (i, j) memiliki biaya terkait C_{ij} yang menunjukkan jarak jalan.

Permasalahan ini dapat digambarkan dengan seorang sales melakukan perjalanan dari satu tempat ke tempat lainnya dan kembali ke tempat semula dimana ia

melakukan perjalanan. Pada permasalahan tersebut, jarak akan berusaha diminimalkan sekecil mungkin. Dalam dunia nyata, permasalahan ini muncul pada pemilihan rute antar-jemput biro travel maupun pemilihan rute penerbangan pesawat. Tentunya mereka akan memilih jarak seminimal mungkin dengan waktu dan biaya yang kecil pula. Contoh lain dari TSP adalah pengantaran surat pos.

f. Vehicle Routing Problem (VRP)

Vehicle Routing Problem (VRP) merupakan penjabaran permasalahan dari TSP dimana mengunjungi setiap pelanggan menggunakan sejumlah kendaraan dengan meminimalkan jumlah jarak yang dilalui. Setiap kendaraan memiliki kapasitas tertentu, serta berangkat dari satu tempat dan kembali ke tempat tersebut. Permasalahan ini melakukan desain optimasi pengiriman yang optimal dari satu atau banyak lokasi [23]. Selain itu, jumlah kendaraan juga dapat diminimalkan sebisa mungkin. Kapasitas tiap kendaraan tidak boleh kurang dari kebutuhan yang akan diberikan kepada pelanggan. Motivasi dari VRP adalah permasalahan transportasi pada dunia industri terutama pada bidang pengiriman barang. Beberapa batasan pada VRP seperti hanya mengunjungi satu lokasi pada satu waktu oleh satu kendaraan, serta semua kendaraan berangkat dan berhenti pada depot.

Permasalahan ini dapat dimodelkan seperti berikut $G = (V, A)$ menjadi grafik di mana $V = \{1, \dots, n\}$ adalah set titik yang menggambarkan kota dengan depot lokasi pada titik 1, dan adalah satu set busur. Sebuah matriks biaya c_{ij} dan matriks waktu perjalanan t_{ij} dikaitkan dengan A. VRP memegang peranan penting dalam bidang distribusi barang fisik dan logistik. Pengiriman barang dari gudang kepada distributor dimana terdapat

beberapa lokasi distributor dan beberapa kendaraan yang dimiliki.

2.2.8. Hyflex

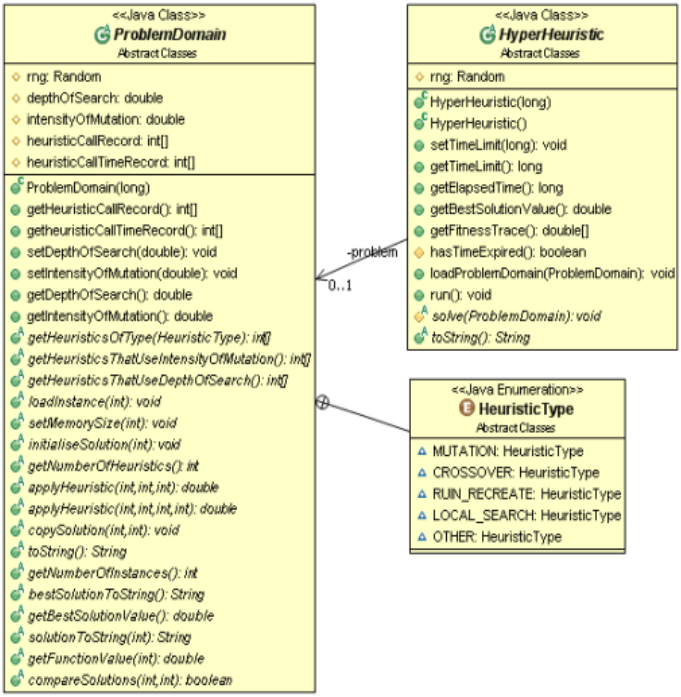
Hyflex (Hyper-heuristic Flexible Framework) merupakan kerangka kerja untuk mempermudah pengembangan, pengujian maupun perbandingan dari algoritma pencarian *heuristic* [26]. Pengujian algoritma tersebut dapat dilakukan pada beberapa domain permasalahan daripada pada satu permasalahan spesifik. Hyflex bekerja pada *library* Java untuk proses desain maupun uji coba sehingga HyFlex merupakan *benchmark* optimasi kombinatorial untuk permasalahan lintas domain.

Terdapat 2 kelas utama pada HyFlex, yaitu Hyperheuristic dan Problem domain. Pada setiap kelas utama, terdapat *method* yang dapat dipanggil dalam melakukan implementasi algoritma *hyper-heuristic*. Hyflex menyediakan beberapa modul domain permasalahan, dan setiap domain permasalahan menyimpan komponen – komponen algoritma seperti : representasi hasil, evaluasi hasil atau solusi, data *instance*, dan repositori yang berhubungan dengan *heuristic* yang dapat diterapkan dalam permasalahan. Pada kerangka kerja ini, bagian *high-level* dari *hyper-heuristic* saja yang akan dirubah untuk mendapatkan solusi yang optimal.

Pengembangan *heuristic* pada kerangka kerja ini memiliki tujuan untuk menghilangkan atau mengurangi kebutuhan dari para ahli dalam melakukan desain algoritma yang efektif untuk menyelesaikan permasalahan pencarian solusi, serta meningkatkan tingkat generalitas dari algoritma pencarian. Selain itu, algoritma dituntut menyediakan sistem yang mampu mengelola dan mengkonfigurasi dirinya sendiri pada saat melakukan pencarian solusi, serta dapat beradaptasi dengan perubahan kondisi permasalahan terutama dalam penggunaan *heuristic*. hal ini dikarenakan desain algoritma saat ini dibuat oleh para ahli dan membutuhkan biaya yang mahal. Kerangka kerja ini, mempermudah pengembangan *hyper-heuristic* dalam memilih metode atau *heuristic* yang tepat pada permasalahan

yang diselesaikan. Dengan begitu perlu adanya mengembangkan metode yang memiliki kemampuan secara otomatis bekerja dengan baik pada domain permasalahan yang berbeda.

HyFlex menyediakan sebuah aplikasi antara *hyper-heuristic* dengan domain permasalahan sehingga mempermudah komunikasi antar lapisan tersebut. Pada lapisan *hyper-heuristic*, seleksi LLH akan dilakukan dan diterapkan untuk mendapatkan solusi baru. Gambar 2.3 merupakan diagram kelas pada kerangka kerja *HyFlex* yang dapat diakses [27].



Gambar 2.3 Diagram Kelas Pada Hyflex

HyFlex telah digunakan untuk mendukung kompetisi penelitian internasional: the First Cross-domain Heuristic Search Challenge (CHeSC 2011*)*, dimana memiliki tujuan untuk mendapatkan hasil yang baik untuk setiap domain permasalahan yang disediakan. CheSC sendiri diluncurkan diluncurkan pada International Conference on Practice and Theory of Automated Timetabling (PATAT 2010).

Tabel 2.3 Domain Permasalahan Hyflex, Inisialisasi Solusi, Jumlah LLH Dan Jumlah Heuristic Sesuai Tipe

Domain	***Initialisation***	***Total***	***Mut.***	***R&R***	***X over.***	***LS.***
Max SAT	*Random bit-string*	9	4	1	2	2
Bin Packing	*Randomised first-fit heuristic*	8	3	2	1	2
Flow Shop	*Randomised NEH procedure*	15	5	2	3	4
Pers. Sched.	*Randomised hill climbing heuristic*	12	1	3	3	4
TSP	*Random permutation*	15	5	1	3	6
VRP	*Randomised constructive heuristic*	12	4	2	2	4

HyFlex memiliki enam *hard combinatorial problem* yang berhubungan dengan permasalahan pada dunia industri, yaitu *satisfiability*, *one dimensional bin packing*, *permutation flow shop*, *personnel scheduling*, *traveling salesman problem*, dan *vehicle routing problem*. Permasalahan tersebut dapat dicari

solusinya baik secara individu maupun populasi dalam penerapan *hyper-heuristic*. Selain itu, setiap domain permasalahan menyediakan contoh data nyata. Setiap modul domain permasalahan HyFlex memiliki [26]:

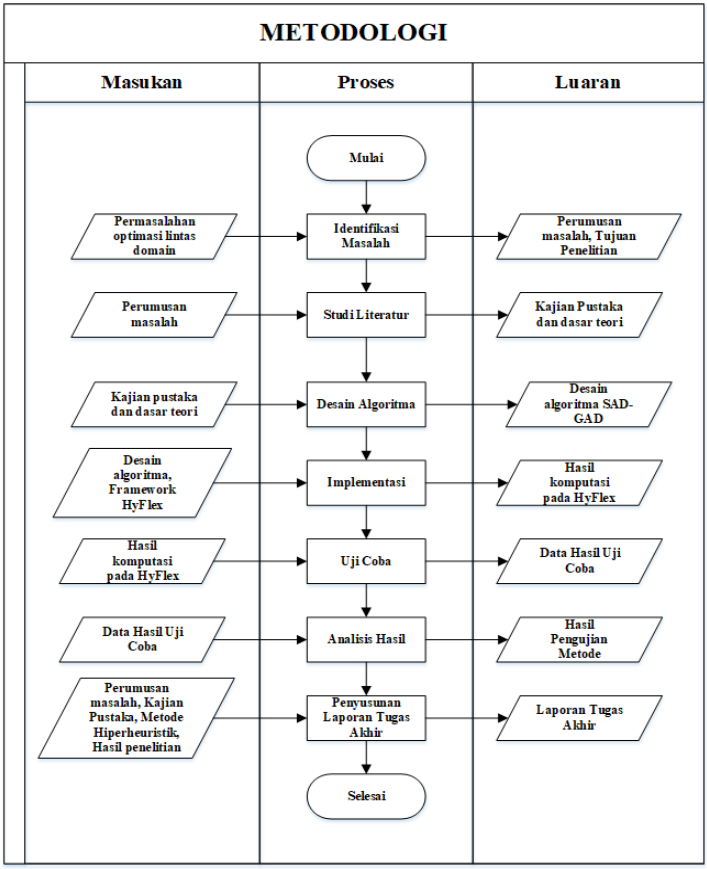
a. Aturan yang sesuai untuk melakukan inisialisasi solusi acak pada populasi
b. kumpulan *heuristic* untuk memodifikasi solusi
    - *Mutational*, membuat modifikasi secara acak pada solusi terbaru.
    - *Ruin-recreate*, menghancurkan solusi saat ini dan membuat ulang dengan prosedur *constructive.*
    - *Local search*, mencari solusi berdasarkan *neighborhood* dari solusi terkini untuk mendapatkan hasil yang lebih baik.
    - *Crossover*, mengambil dua solusi lalu menggabungkan keduanya dan mengembalikan nilai baru.
c. Variasi *instance* yang mudah dipanggil.
d. Populasi dari satu atau banyak solusi.

BAB III
METODOLOGI

Bab ini menjelaskan metodologi dalam pengerjaan tugas akhir. Metodologi ini digunakan sebagai panduan dan memberikan alur pengerjaan tugas akhir yang sistematis.

3.1. Tahapan Pelaksanaan Tugas Akhir

Alur metodologi pengerjaan tugas akhir dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Pengerjaan Tugas Akhir

3.1.1. Identifikasi masalah

Tahap identifikasi masalah merupakan langkah awal yang dilakukan dalam penelitian ini. Pada tahap ini, akan ditentukan rumusan masalah, batasan masalah, tujuan dan manfaat pengerjaan tugas akhir.

Penelitian ini bertujuan untuk Menerapkan metode *hyper-heuristic* yang mampu memberikan hasil yang optimal (*fitness*) untuk setiap domain permasalahan yang terdapat pada kerangka kerja HyFlex. Setiap permasalahan yang terdapat pada HyFlex memiliki karakteristik permasalahan yang berbeda sehingga LLH yang terdapat pada setiap permasalahan juga berbeda. *Hyper-heuristic* harus mampu mengenali setiap LLH dan menggunakan LLH pada setiap permasalahan.

Penelitian ini berusaha Menerapkan metode *high level heuristic* yang tepat sehingga dapat memilih LLH dengan tepat dan memberikan solusi yang baik pada setiap domain permasalahan. pengembangan *hyper-heuristic* ini berfokus pada seleksi LLH pertubatif berbasis *single point search* (satu hasil solusi) [4].

Tahapan yang akan dilakukan dalam penelitian ini, yakni mencari solusi baru dari mekanisme pertubatif dan dibandingkan dengan solusi awal. Selanjutnya, menentukan keputusan penerimaan solusi yang baru.

3.1.2. Studi literatur

Pada tahapan ini dilakukan studi literatur terkait materi yang akan digunakan sebagai acuan penelitian. Studi literatur mencakup konsep yang akan diterapkan dalam penelitian. Semua hasil studi literatur berupa penelitian terdahulu serta dasar teori telah dicantumkan pada Bab II.

3.1.3. Desain algoritma

Tahap ini merupakan tahap pengembangan *hyper-heuristic* sebagai strategi *high level heuristic* yang sesuai dengan permasalahan yang telah didefinisikan pada Bab II. Pada

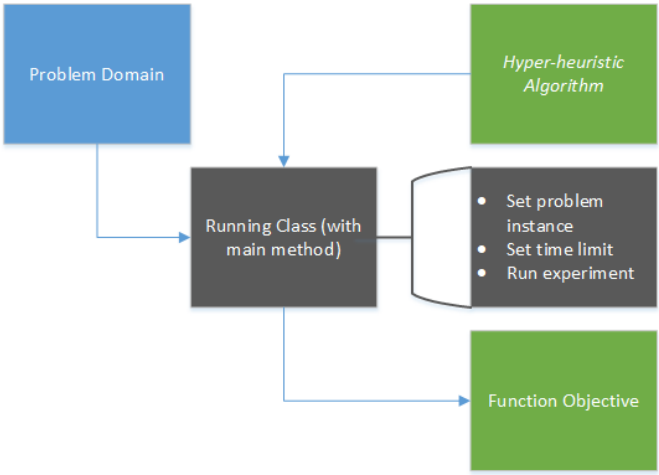
tahapan ini akan dijabarkan metode yang akan digunakan untuk melakukan seleksi LLH dan mekanisme penerimaan solusi.

3.1.4. Implementasi

Tahap ini merupakan tahap implementasi desain algoritma pada kerangka kerja *hyper-heuristic Flexible* (HyFlex). Implementasi merupakan tahap pembangunan desain algoritma kedalam bahasa program. Tahap ini dimulai dari persiapan *tools* hingga implementasi program.

3.1.5. Uji coba

Tahap uji coba merupakan pengimplementasian dan eksperimen dari algoritma *hyper-heuristic* yang telah melalui tahap desain dan telah dibangun pada tahap implementasi. Pengujian algoritma berusaha menemukan solusi yang optimal bagi enam domain permasalahan pada HyFlex. Uji coba dilakukan untuk mengetahui bagaimana kinerja dari algoritma yang telah dibangun pada enam domain permasalahan pada kerangka kerja *HyFlex*



Gambar 3.2 Skenario Uji Coba

Pada gambar 3.2, Algoritma 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, kerangka kerja akan mengembalikan nilai solusi terbaik yang dihasilkan dari proses pencarian. Algoritma 3.1 digunakan untuk menjalankan *hyper-heuristic* dengan data *input* dan *output* dari uji coba menggunakan permasalahan SAT.

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

Algoritma 3.1 Running Hyper-heuristic

Penjelasan masukan maupun luaran dari algoritma 3.1 yang digunakan untuk menjalankan *hyper-heuristic* dapat dijabarkan pada tabel 3.1.

Tabel 3.1 Masukan Dan Luaran Uji Coba

Tipe	Data	Keterangan
Input	Problem Domain	Membuat objek dari domain permasalahan yang akan diselesaikan dengan algoritma *hyper-heuristic*. Terdapat 4 domain permasalahan serta 2 domain permasalahan tersembunyi yang terdapat pada *HyFlex*, yakni *one dimensional bin packing*, *permutation flow shop*, *personnel scheduling*, *traveling salesman problem*, dan *vehicle routing problem*. Setelah itu memasukan parameter *random seed*.

Tipe	Data	Keterangan
	Hyper-heuristic object	Objek dari kelas *hyper-heuristic* yang akan digunakan untuk menyelesaikan permasalahan serta memberi parameter *random seed*.
	LoadInstance	*Instance* dari domain permasalahan yang akan diselesaikan. Terdapat 10 *instance* utama dan 2 *hidden instance* pada permasalahan *satisfiability*, *one dimensional bin packing*, *permutation flow shop*, *personnel scheduling*. Sedangkan pada 2 *hidden problem*, yakni *traveling salesman problem*, dan *vehicle routing problem* terdapat 10 *instance* permasalahan.
	Waktu *Running* Algoritma	Waktu yang dibutuhkan untuk melakukan *running* algoritma sampai dengan selesai. Waktu standar yang digunakan adalah 60000 milidetik.
	Jumlah *Heuristic*	Jumlah *heuristic* yang disediakan pada setiap domain permasalahan yang spesifik. Jumlah dari *heuristic* yang digunakan pada setiap domain permasalahan berbeda dan dapat dilihat pada tabel 2.3
Output	Solusi Terbaik	Solusi terbaik (*fitness*) yang dihasilkan dari hasil *running* algoritma.

Hasil dari tahap ini digunakan untuk menentukan strategi *high-level heuristic* yang tepat sehingga dapat memberikan solusi yang baik bagi domain permasalahan HyFlex. Uji coba dalam penelitian mengikuti aturan kerangka kerja HyFlex dengan benchmark dari kompetisi ChesSC 2011, yaitu sebagai berikut:

a. Komputer menggunakan mesin *single core* atau *dual core*.
b. Program dapat dijalankan pada sistem Windows dan Linux (32 bit atau 64 bit). Program harus dijalankan dari *console*.

c. Penetapan parameter kriteria akhir (alokasi waktu maksimum CPU, jumlah maksimum iterasi, jumlah iterasi antara dua peningkatan). Parameter 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 permasalahan [1], [10].

3.1.6. Analisis hasil

Tahap ini dilakukan setelah tahap implementasi dan uji coba selesai dilakukan. Tahap ini bertujuan untuk mengukur kinerja dari metode *hyper-heuristic self adaptive learning great deluge algorithm* dengan cara membandingkan hasil dengan metode *simple random simulated annealing*. Tahap ini akan menghitung hasil eksekusi nilai terbaik (minimum), median, dan rata-rata dari masing-masing metode terhadap enam domain permasalahan.

Perbandingan yang dilakukan mengikuti aturan yang diberlakukan oleh ChesSC 2011. Perbandingan mengacu pada solusi yang dihasilkan oleh algoritma (*fitness*), nilai median dan rata – rata serta beberapa data statistik tambahan seperti nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimum dengan eksekusi secara berulang sebanyak 31 kali pada setiap *instance* yang ada pada HyFlex (30 *instance*) dengan waktu 60000 milidetik atau 10 menit.

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.

Pada tahap ini, algoritma *self adaptive learning great deluge* diharapkan mampu memberikan hasil yang lebih baik dari algoritma pembanding *simple random simulated annealing* dimana memiliki kelebihan mampu beradaptasi dalam memilih LLH yang sesuai dengan domain permasalahan.

3.1.7. Penyusunan laporan tugas akhir

Tahap ini bertujuan untuk melaporkan seluruh hasil pekerjaan yang telah dilakukan dalam menyelesaikan permasalahan. Laporan ini mendokumentasikan rangkaian proses implementasi algoritma *Self Adaptive Learning – Great Deluge* mulai dari formulasi fungsi tujuan hingga menemukan solusi akhir yang diinginkan. Luaran dari tahap ini yaitu laporan tugas akhir.

(Halaman Ini Sengaja Dikosongkan)

BAB IV
PERANCANGAN

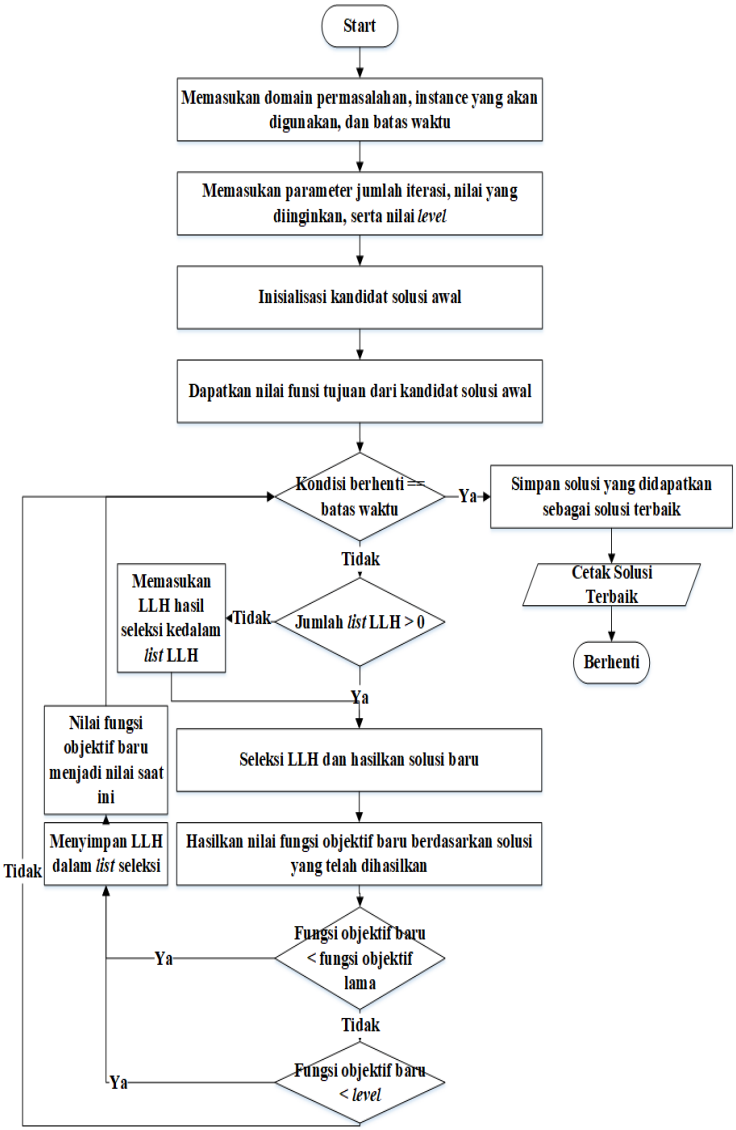
Tahapan desain dan implementasi dalam tugas akhir ini melakukan perancangan dan implementasi algoritma yang sesuai dengan identifikasi permasalahan seperti dijelaskan pada Bab I. Subbab berikut menjelaskan desain dan implementasi algoritma dalam penelitian ini.

Pada tahap ini dilakukan pendesainan algoritma *hyper-heuristic* untuk menyelesaikan permasalahan optimasi lintas domain yang terdapat pada *HyFlex.* Desain algoritma didasarkan pada kerangka kerja *hyper-heuristic* yang terdiri dari dua bagian, yakni *high level heuristic* dan rangkaian LLH.

Secara umum, proses penyelesaian permasalahan optimasi akan dimulai dengan inisialisasi solusi awal secara lengkap. Kemudian *high level heuristic* akan memilih LLH yang tersedia secara iteratif dan menghasilkan solusi baru. Solusi baru tersebut berikutnya akan ditentukan apakah diterima atau tidak. Jika solusi tersebut diterima, maka akan menjadi solusi *incumbent*, namun jika tidak diterima maka akan mencari LLH lain untuk menghasilkan solusi baru lainnya.

4.1. High Level Heuristic

Strategi *high level* yang diterapkan pada *hyper-heuristic* adalah gabungan antara *self adaptive learning* dan *great deluge* (SAD-GED). Metode *self adaptive learning* digunakan sebagai pemilihan LLH untuk menyelesaikan permasalahan, sedangkan *great deluge* digunakan untuk mekanisme *move acceptance* solusi baru yang dihasilkan dari penerapan LLH pada domain permasalahan. Mekanisme ini akan menerima solusi yang lebih baik atau dibawah batas nilai parameter B yang telah ditentukan. Desain algoritma SAD-GED dapat dilihat pada gambar 4.1.



Gambar 4.1 Desain High Level Heuristic SAD-GED

Algoritma dimulai dengan memasukan parameter – parameter pada *HyFLex* yang dibutuhkan seperti domain permasalahan, *instance*, serta batas waktu. Selanjutnya kerangka kerja akan membentuk solusi awal dan algoritma SADGED akan melakukan pencarian solusi.

4.2. Rangkaian LLH

Rangkaian LLH atau *low level heuristic* merupakan kumpulan *heuristic* pada *HyFlex* yang digunakan untuk membangkitkan solusi sehingga dapat diketahui fungsi objektif dari solusi tersebut dalam menyelesaikan permasalahan. Setiap domain permasalahan memiliki LLH masing – masing dan akan dipilih satu untuk menyelesaikan permasalahan. Pemilihan LLh tersebut dilakukan dengan memanggil method yang tersedia pada kerangka kerja *HyFlex*. Berdasarkan tabel 2.3 domain permasalahan SAT memiliki sembilan LLH. Rincian LLH dalam SAT dapat dilihat pada tabel 4.1 [22].

Tabel 4.1 Rangkaian LLH pada Domain Permasalahan SAT

No	Nama LLH	Deskripsi	Tipe LLH
1.	*GSAT*	Membalik variabel dengan nilai tertinggi dan diberhentikan secara acak.	Mutation
2.	*HSAT*	Secara fungsionalitas seperti GSAT, tetapi variabel yang diberhentikan adalah variabel yang paling lama bertahan.	Mutation
3.	*WalkSAT*	Klausa dipilih secara acak. Jika ada variabel yang memiliki nilai negatif 0, secara acak mengambil variabel dan dan membaliknya. Jika tidak ada variabel seperti itu, dipilih suatu variabel acak dengan probabilitas 0,5 atau variabel dengan keuntungan negatif minimal.	Mutation

No	Nama LLH	Deskripsi	Tipe LLH
4.	*Flip Random Variable from a Broken Clause*	Variabel rusak yang dipilih dari klausa dibalik secara acak.	Mutation
5.	*Flip Random Variable*	Membalik variabel secara acak	Mutation
6.	*Novelty*	Memiliki klausa yang rusak secara acak,membalik variabel yang memberikan hasil paling baik atau variabel dengan umur paling kecil. Pada kasus tersebut, balik variabel dengan probabilitas 0,3. Jika tidak, balik variabel dengan yang memberikan hasil terbaik kedua .	Mutation
7.	*Reinitialise Variables*	Proporsi variabel secara acak diulang kembali. Bergantung 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*	Membalik variabel secara acak dari klausa yang rusak secara acak.	Local search
9.	*Flip Random Variable*	Variabel yang dipilih, dibalikkan secara acak.	Local search
10.	*Two point crossover*	Standar dua titik crossover pada string boolean variabel.	Crossover
11.	*One point crossover*	Standar satu titik crossover pada string boolean variabel.	Crossover

Rangkaian LLH yang terdapat pada domain permasalahan one dimensional bin packing adalah sebanyak delapan LLH. Rincian LLH dalam bin packing dapat dilihat pada tabel 4.2 [28].

Tabel 4.2 Rangkaian 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 maka dimasukan ke dalam bin yang kosong	Mutation
2.	*Split a Bin*	Heuristik ini memilih bin secara acak dari bin yang memiliki bagian lebih banyak daripada rata-rata bin yang tersedia. Kemudian membagi bin ini menjadi dua dan masing-masing berisi setengah bagian dari bin asalnya.	Mutation
3.	*Repack the Lowest Filled Bin*	Membuang semua bagian dari bin terendah yang telah terisi, dan mengemasnya kembali pada wadah lain jika memungkinkan, dengan *heuristic* yang paling sesuai.	Mutation
4.	*Destroy x Highest Bins*	menghapus semua bagian dari *x bin terbesar*. dimana x adalah nilai integer yang ditentukan dengan parameter '*intensity of mutation*'	Ruin-recreate
5.	*Destroy x Lowest Bins*	menghapus semua bagian dari *x bin terkecil*. dimana x adalah nilai integer yang ditentukan dengan parameter '*intensity of mutation*'	Ruin-recreate
6.	*Exon Shuffling Crossover*	bagian di pindah silang	Crossover

No	Nama LLH	Deskripsi	Tipe LLH
7.	*Swap*	Memilih dua bagian secara acak, lalu menukar posisi mereka apabila terdapat tempat dan menghasilkan *fitness* yang lebih baik	Local search
8.	*Swap from Lowest Bin*	Mengambil bagian terbesar dari bin terkecil yang terisi, dan ditukar dengan bagian yang lebih kecil dari bin yang dipilih secara acak. Jika tidak ada bagian yang menghasilkan hasil valid setelah penukaran, 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. Rincian LLH dalam Flow Shop dapat dilihat pada tabel 4.3 [24].

Tabel 4.3 Rangkaian LLH pada Domain Permasalahan Flow Shop

No	Nama LLH	Deskripsi	Tipe LLH
1.	*randomReinsertion*	Memasukan kembali pekerjaan secara acak pada posisi yang acak pula.	Mutation
2.	*swapTwo*	Menukar dua pekerjaan secara acak pada permutasi	Mutation
3.	*shuffle*	Pekerjaan dipilih secara acak dalam permutasi.	Mutation

No	Nama LLH	Deskripsi	Tipe LLH
4.	*shuffleSubSequence*	Elemen k dipilih secara acak dalam permutasi, dimana k=2+[α × (n-2)], dan α adalah parameter intensitas mutasi.	Mutation
5.	*useNEH*	Solusi baru dibuat berdasarkan 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.	Ruin-recreate
7.	*deepIteratedGreedy*	Elemen *l* dihapus seperti langkah sebelumnya, kemudian memilih pekerjaan 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 *steepest descent local search*. Pada setiap iterasi pekerjaan dihapus dari posisi saat ini dan ditugaskan ke semua	Local search

No	Nama LLH	Deskripsi	Tipe LLH
		posisi yang tersisa. Pekerjaan itu diarahkan ke posisi yang mengarah pada jadwal terbaik. Ini diulang sampai tidak ada perbaikan yang diamati.	
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 diulang sampai tidak ada perbaikan yang diamati.	Local search
10.	*randomLocalSearch*	LLH ini adalah pencarian lokal tunggal secara acak. Dengan aturan $r = \lfloor \beta * (n - 1) \rfloor + 1$, pekerjaan yang dipilih secara acak diuji (satu per satu) pada semua posisi dan ditetapkan ke tempat terbaik yang mungkin.	Local search
11.	*fImpLocalSearch*	LLH ini adalah peningkatan satu kali pencarian lokal tunggal secara acak pertama. Pekerjaan tidak perlu diuji pada setiap posisi.	Local search

No	Nama LLH	Deskripsi	Tipe LLH
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. Rincian LLH dalam personnel scheduling dapat dilihat pada tabel 4.4 [26].

Tabel 4.4 Rangkaian LLH pada Domain Permasalahan Personnel Scheduling

No	Nama LLH	Deskripsi	Tipe LLH
1.	*Mutation heuristic 1*	*Heuristic* ini membatalkan sejumlah shift secara acak berdasarkan nilai parameter.	Mutation
2.	*Ruin and Recreate Heuristic 1*	*Heuristic* ini bekerja dengan membatalkan penugasan satu atau banyak pekerja. Kemudian mereka di jadwalkan kembali berdasarkan kepuasan terhadap hari, waktu jaga maupun minggu.	Ruin-recreate
3.	*Ruin and Recreate Heuristic 2*	Membatalkan jadwal pekerja dan membangun ulang hanya pada dua sampai dengan 6 pola kerja pada satu waktu. Heuristic ini dipengaruhi oleh variabel x, yakni x = Round(intensityOfMutation * 4) + 2	Ruin-recreate

No	Nama LLH	Deskripsi	Tipe LLH
4.	*Ruin and Recreate Heuristic 3*	Menyediakan perubahan yang besar pada solusi menggunakan persamaan x = Round(intensityOfMutation*number of employee)	Ruin-recreate
5.	*Ruin and Recreate Heuristic 3*	*Heuristic* ini membuat gangguan kecil pada solusi dengan menggunakan nilai x=1	Ruin-recreate
6.	*Crossover heuristic 1*	LLH Ini beroperasi dengan mengidentifikasi tugas terbaik x di setiap parent. Penugasan terbaik diidentifikasi dengan mengukur perubahan dalam fungsi objektif ketika setiap pergeseran sementara tidak dijadwalkan.	Crossover
7.	*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
8.	*Crossover heuristic 3*	LLH ini membuat daftar baru dengan membuat tugas yang hanya umum bagi kedua parent.	Crossover
9.	*Local Search Heuristic 1*	Menambahkan *shift* pada jadwal karyawan.	Local search

No	Nama LLH	Deskripsi	Tipe LLH
10.	*Local Search Heuristic 2*	Menukar jadwal antara dua karyawan.	Local search
11.	*Local Search Heuristic* 3	Menukar jam kerja satu karyawan.	Local search
12.	*Local Search Heuristic* 4	Variasi dari variable depth search.	Local search
13.	*Local Search Heuristic* 5	*Heuristic* ini mengacu pada algoritma greedy dimana menciptakan jadwal baru untuk satu pekerja.	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 4.5.

Tabel 4.5 Rangkaian LLH pada Domain Permasalahan TSP

No	Nama LLH	Deskripsi	Tipe LLH
1.	*twoOptLocalSearch*	Pilih dua pencarian lokal, dan pilih peningkatan pertama	Local search
2.	*bestImpTwoOptLocalSearch*	Pilih dua pencarian lokal dan pilih peningkatan terbaik	Local search
3.	*threeOptLocalSearch*	Pilih tiga pencarian lokal dan pilih peningkatan pertama	Local search
4.	*Ox*	Permintaan crossover	Crossover

No	Nama LLH	Deskripsi	Tipe LLH
5.	*pmx*	Precedence preservative crossover	Crossover
6.	*ppx*	Crossover yang dipetakan sebagian	Crossover
7.	*oneX*	Crossover pada satu titik	Crossover
8.	*randomReinsertion*	LLH ini memasukkan kembali kota pada posisi yang berbeda dalam tur secara acak	Mutation
9.	*swapTwo*	LLH ini menukarkan dua kota yang dipilih secara acak	Mutation
10.	*shuffle*	LLH ini mengacak tur	Mutation
11.	*shuffleSubSequence*	LLH ini mengacak tur berikutnya	Mutation
12.	*nOptMove*	Memindahkan Kota	Mutation
13.	*iteratedGreedy*	Kota-kota dari sub-urutan yang dihapus dimasukkan kembali	Ruin-recreate

Rangkain LLH yang terdapat pada domain permasalahan *vehicle routing problem* (VRP) sebanyak sepuluh LLH. Rincian rangkaian LLH yang terdapat di dalam VRP dapat dilihat pada tabel 4.6.

Tabel 4.6 Rangkaian LLH pada Domain Permasalahan VRP

No	Nama LLH	Deskripsi	Tipe LLH
1.	*twoOptStar*	Heuristik ini menukar bagian akhir dari dua rute, kemudian membuat dua rute baru.	Local search
2.	*GENI*	Heuristik ini menghilangkan pelanggan dari satu rute dan memindahkan pelanggan tersebut ke rute lain secara acak.	Local search
3.	*combine*	Heuristik ini memilih kombinasi rute dari 2 solusi induk untuk membentuk solusi generasi.	Local search
4.	*locRR*	Heuristik ini menghitung perhitungan nilai kedekatan sebagai jarak euclidean antara *benchmark* pelanggan dan pelanggan yang sedang dipertimbangkan.	Ruin-recreate
5.	*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
6.	*twoOpt*	Heuristik ini menukarkan pelanggan dengan pelanggan yang mendahuluinya dalam rute.	Mutation
7.	*orOpt*	Dua pelanggan yang berurutan dipilih dalam satu rute, dan dipindahkan ke lokasi lain dalam rute yang sama.	Mutation

8.	*shiftMutate*	Heuristik ini menghapus pelanggan dari satu rute dan dimasukkan ke rute lain.	Mutation
9.	*combineLong*	Heuristik ini memilih rute dari kedua solusi sebelum mencoba memasukkan pelanggan yang tersisa.	Crossover
10.	*interchange*	Heuristik ini menerima solusi baru hanya jika dapat meningkatkan nilai fungsi objektif.	Crossover

BAB V
IMPLEMENTASI

Implementasi dilakukan pada algoritma yang telah dirancang pada tahap sebelumnya. Tahap ini dilakukan mulai dari persiapan aplikasi yang akan digunakan sampai dengan pembangunan algoritma dan eksekusi algoritma.

5.1. Kebutuhan Implementasi

Implementasi dalam pengerjaan tugas akhir ini dilakukan dengan prosesor core i5 3.2 Ghz dan memori 4096 MB. Desain algoritma akan diimplementasikan pada kerangka kerja HyFlex menggunakan aplikasi NetBeans IDE 8.2. implementasi dilakukan dengan memanggil *method* yang terdapat pada *library chesc.jar.*

5.2. Fungsi – fungsi yang digunakan

Fungsi – fungsi yang digunakan pada implementasi *hyper-heuristic* mengacu pada library yang tersedia pada kerangka kerja HyFlex. fungsi – fungsi tersebut dapat dijabarkan sebagai berikut :

a. *toString*(), memberi nama metodologi yang akan dirancang.
b. *initialiseSolution*(), dimana melakukan inisialisasi solusi awal secara acak
c. *hasTimeExpired*(), fungsi untuk mencatat dan memantau lamanya waktu yang saat menjalankan program (batasan waktu).
d. *getNumberOfHeuristics*() digunakan untuk mendapatkan index LLH.
e. *applyHeuristic* (i, j, k), merupakan fungsi untuk menerapkan LLH (i) pada array (k) dengan pendahulu solusi sebelumnya pada array (j).
f. *copySolution*(k,j), Solusi dipindahkan dari indeks memory k ke indeks memori j.

g. *loadInstance* (a), memanggil *instance* a.
h. *solve*(), fungsi yang digunakan untuk melakukan pencarian nilai fungsi objektif.
i. *loadProblemDomain*(), digunakan untuk memanggil kelas ProblemDomain.
j. *run*(), fungsi yang digunakan untuk menjalankan proses pencarian nilai fungsi objektif.
k. *getBestSolutionValue*(), merupakan fungsi untuk menemukan nilai fungsi objektif terbaik dari setiap *instance* pada permasalahan yang berbeda.
l. *bestSolutionToString*(), merupakan fungsi untuk mengubah nilai fungsi objektif menjadi tipe *string*.

5.3. Pengaturan parameter

Pada implementasi desain algoritma yang akan diterapkan terdapat beberapa pengaturan nilai parameter yang akan digunakan. Tabel 4.7 berikut ini adalah detail dari pengaturan parameter algoritma yang digunakan.

Tabel 5.1 Pengaturan Parameter

Variabel	Deskripsi	Nilai	Referensi
T	Suhu awal pada algoritma *Simulated Annealing*	100	[29]
Alpha	Faktor penurun suhu pada *Simulated Annealing*	0.95	[29]

5.4. Implementasi *Simple Random Simulated Annealing*

Pada tahap ini, dilakukan implementasi algoritma *Simple Random Simulated Annealing* dimana berperan dalam *high level heuristic*. Algoritma ini digunakan sebagai pembanding dari algoritma usulan SAD-GED. Implementasi algoritma ini mengunakan bahasa pemrograman Java.

```
1  import AbstractClasses.HyperHeuristic;
2  import AbstractClasses.ProblemDomain;
3  import SAT.SAT;
4  import BinPacking.BinPacking;
```

```
5  import FlowShop.FlowShop;
6  import
   PersonnelScheduling.PersonnelSchedulin
   g;
7  import travelingSalesmanProblem.TSP;
8  import VRP.VRP;
```

Kode Program 5.1 Simple Random Simulated Annealing 1

Baris algoritma 4.1 merupakan algoritma untuk memanggil *library* yang akan digunakan dalam penyelesaian permasalahan optimasi lintas domain. Baris pertama digunakan untuk memanggil kelas *hyperheuristic* yang berisi fungsi – fungsi untuk menjalankan program. Baris kedua memanggil kelas permasalahan. baris ketiga hingga delapan memanggil setiap domain permasalahan yang terdapat pada HyFlex. berikutnya akan dilakukan implementasi algoritma *Simulated Annealing.*

```
9  public class SA extends HyperHeuristic
   {
10     double T = 100;
11     double alpha = 0.95;
12     double p;
13     public SA(long seed){
14             super(seed);
15       }
16 public void solve(ProblemDomain
   problem) {
17 int numLLH=
   problem.getNumberOfHeuristics();
18 double
   current_obj_function_value=Double.POSI
   TIVE_INFINITY;
19 problem.initialiseSolution(0);
20 while (!hasTimeExpired()) {
21 int LLH_apply = rng.nextInt(numLLH);
22 double new_obj_function_value =
   problem.applyHeuristic(LLH_apply, 0,
   1);
```

```
23 double delta_obj_function =
   current_obj_function_value -
   new_obj_function_value;
24 if (delta_obj_function > 0) {
25 problem.copySolution(1, 0);
26 current_obj_function_value =
   new_obj_function_value}
27 else{
28 p = Math.exp(-
   (Math.abs(delta_obj_function)/T));
29 if(p >rng.nextDouble() ){
30 problem.copySolution(1, 0);
31 current_obj_function_value =
   new_obj_function_value; }}
32 T = alpha * T}}
```

Kode Program 5.2 Simple Random Simulated Annealing 2

Pada baris kesepuluh sampai dengan duabelas merupakan inisialisasi variabel suhu awal, faktor pengurang suhu, serta probabilitas sebagai penerimaan solusi yang dihasilkan. Baris berikutnya merupakan pembuatan objek SA dengan *seed* acak.

Baris ke enam belas merupakan fungsi yang akan digunakan untuk mencari nilai fungsi objektif dari *hyperheuristic*. Pada baris ke dua puluh lima, solusi baru yang dihasilkan akan diterima apabila memiliki nilai yang lebih rendah dari pada nilai yang disimpan sebelumnya. Namun, apabila nilai lebih tinggi dari nilai yang disimpan, maka nilai tersebut akan dibandingkan dengan probabilitas seperti pada baris ke dua puluh delapan. Berikutnya suhu saat ini akan berkurang sebanyak perkalian dengan variabel alpha.

```
33 ProblemDomain problem = new
   BinPacking(1234);
34 HyperHeuristic HH_Object = new
   SA(5678);
35 System.out.println("Metode "+HH_Object
   );
36 HH_Object.setTimeLimit(60000);
```

```
37 problem.loadInstance(8);
38 HH_Object.loadProblemDomain(problem);
39 HH_Object.run();
40 System.out.println("BEST FUNCTION
   SOLUTION : "
   +HH_Object.getBestSolutionValue());
```

Kode Program 5.3 Simple Random Simulated Annealing 3

Pada algoritma di atas, baris ke tujuh puluh empat digunakan untuk memanggil domain permasalahan yang akan digunakan. Dilanjutkan dengan baris ke tujuh puluh lima dimana memanggil *hyperheuristic* yang akan digunakan untuk menyelesaikan permasalahan. dari algoritma dapat diketahui bahwa waktu menjalankan program adalah 60000 milidetik dengan memanggil *instance* ke delapan. Baris terakhir digunakan untuk mencetak hasil yang didapatkan.

5.5. Implementasi *Self Adaptive Learning Great Deluge*

Implementasi algoritma gabungan *Self Adaptive Learning Great Deluge* dilakukan pada kerangka kerja dengan mengikuti aturan yang terdapat pada *HyFlex*. Algoritma ini nantinya akan dibandingkan dengan kinerja dari *Simple Random Simulated Annealing*. Algoritma dapat dilihat seperti berikut.

```
1  import AbstractClasses.HyperHeuristic;
2  import AbstractClasses.ProblemDomain;
3  import SAT.SAT;
4  import BinPacking.BinPacking;
5  import FlowShop.FlowShop;
6  import
   PersonnelScheduling.PersonnelSchedulin
   g;
7  import travelingSalesmanProblem.TSP;
8  import VRP.VRP;
```

Kode Program 5.4 Self Adaptive Learning Great Deluge 1

Baris algoritma 4.4 merupakan algoritma untuk memanggil *library* yang akan digunakan dalam penyelesaian permasalahan optimasi lintas domain. Baris pertama digunakan untuk

memanggil kelas *hyperheuristic* yang berisi fungsi – fungsi untuk menjalankan program. Baris kedua memanggil kelas permasalahan. baris ketiga hingga delapan memanggil setiap domain permasalahan yang terdapat pada HyFlex.

```
 9  public class SADGED extends
    HyperHeuristic{
10 double desired_value=0;
11 double beta = 0;
12 double level = 0;
13 int LLH_apply=0;
14 static double
    current_obj_function_value=0;
15 double new_obj_function_value = 0;
16 int llh = 0;
17 static int batas_llh=10;
18 int batas_best = batas_llh;
19 int penanda_LLH = 0;
20 int penanda_best = 0;
21 int LLH [] = new int[batas_llh];
22 int best [] = new int [batas_best];
23 static int numLLH;
24 public SADGEDtraintest11(long seed){
25 super(seed);
26 }
27 public void solve(ProblemDomain
    problem) {
28 numLLH=
    problem.getNumberOfHeuristics();
29 System.out.println(numLLH + " " +
    batas_llh);
30 problem.initialiseSolution(0);
31 LLH_apply = rng.nextInt(numLLH);
32 current_obj_function_value =
    problem.applyHeuristic(LLH_apply, 0,
    1);
33 level = current_obj_function_value;
34 desired_value = 0.1 *
    current_obj_function_value;
```

```
35 beta = (level-desired_value)/N;
36 for(int i =0; i<batas_llh ; i++){
37 LLH[i]=rng.nextInt(numLLH);
38 }
39 while (!hasTimeExpired()) {
40 if(penanda_LLH<batas_llh){
41 llh = LLH[penanda_LLH];
42 penanda_LLH++;
43 }else{
44 penanda_LLH=0;
45 for(int i=0 ; i<batas_llh; i++){
46 if(penanda_best>0) {
47 if(i<0.75*batas_llh){
48 int temp = rng.nextInt(penanda_best);
49 LLH[i] = best[temp];
50 }else{
51 LLH[i]=rng.nextInt(numLLH);
52 }
53 }else{
54 LLH[i]=rng.nextInt(numLLH);
55 }
56 }
57 penanda_best=0;
58 llh = LLH[penanda_LLH];
59 penanda_LLH++;
60 }
61 new_obj_function_value =
   problem.applyHeuristic(llh, 0, 1);
62 double delta_obj_function =
   current_obj_function_value -
   new_obj_function_value;
63 if (delta_obj_function >= 0 ||
   new_obj_function_value <= level) {
64 problem.copySolution(1, 0);
65 current_obj_function_value =
   new_obj_function_value;
66 best[penanda_best] = llh;
67 penanda_best++;
68 }
```

```
69 level = level - beta;
70 }}
71 public String toString() {
72 return "Great Deluge";
73 }
```

Kode Program 5.5 Self Adaptive Learning Great Deluge 2

Baris sepuluh sampai dengan dua puluh tiga merupakan inisialisasi variabel yang akan digunakan untuk pencarian solusi menggunakan algoritma ini. Berikutnya, pada baris dua puluh empat membuat instance *hyperheuristic* dengan *seed* acak. Nilai dari variabel *desired value* diatur menjadi 10 persen dari nilai inisial solusi. Nilai parameter *level* diawal atau faktor penerimaan solusi diatur sama dengan nilai solusi awal, sedangkan nilai *decay rate* atau faktor pengurang suhu diatur sesuai dengan persamaan 4.1 [6].Kemudian pada algoritma, *self adaptive learning* berperan dalam melakukan pemilihan LLH yang menghasilkan nilai terbaik. Ketika LLH yang dipilih telah terpakai semua, maka algoritma akan mengisi LLH yang akan digunakan dengan komposisi 75% dari LLH yang menghasilkan nilai yang dapat diterima oleh metode *move acceptance*, sedangkan 25% dari LLH yang terdapat pada domain permasalahan [20]. berikutnya apabila nilai yang dihasilkan lebih kecil dari nilai yang disimpan saat ini, maka nilai tersebut diterima dan menjadi solusi saat ini. Namun, apabila nilai lebih dari nilai saat ini, maka akan dibandingkan dengan variabel level. Jika nilai lebih kecil dari variabel level, maka nilai akan diterima. Berikutnya pada baris ke enam puluh sembilan, nilai variabel level akan dikurangkan dengan variabel beta.

$$\frac{(solusi\ awal - nilai\ yang\ diinginkan)}{jumlah\ iterasi} \qquad (5.1)$$

```
74 ProblemDomain problem = new
   BinPacking(1234);
75 HyperHeuristic HH_Object = new
   SA(5678);
76 System.out.println("Metode "+HH_Object
   );
77 HH_Object.setTimeLimit(60000);
78 problem.loadInstance(8);
79 HH_Object.loadProblemDomain(problem);
80 HH_Object.run();
81 System.out.println("BEST FUNCTION
   SOLUTION : "
   +HH_Object.getBestSolutionValue());
```

Kode Program 5.6 Self Adaptive Learning Great Deluge 3

Pada algoritma di atas, baris ke tujuh puluh empat digunakan untuk memanggil domain permasalahan yang akan digunakan. Dilanjutkan dengan baris ke tujuh puluh lima dimana memanggil *hyperheuristic* yang akan digunakan untuk menyelesaikan permasalahan. dari algoritma dapat diketahui bahwa waktu menjalankan program adalah 60000 milidetik dengan memanggil *instance* ke delapan. Baris terakhir digunakan untuk mencetak hasil yang didapatkan.

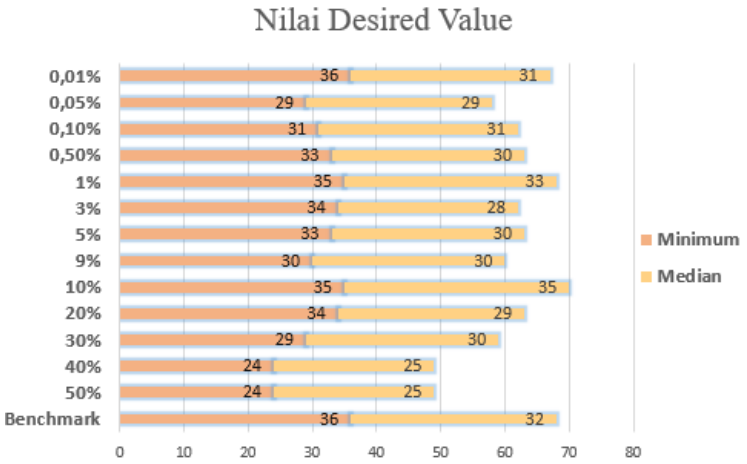
(Halaman Ini Sengaja Dikosongkan)

BAB VI
HASIL DAN PEMBAHASAN

Dalam tahap uji coba dan analisis hasil akan ditampilkan hasil uji coba metode *Self Adaptive Learning Great Deluge* sebagai strategi *high level heuristic* yang diusulkan terhadap enam domain permasalahan optimasi lintas domain. Selanjutnya, dilakukan evaluasi kinerja terhadap strategi *high level heuristic* yang digunakan tersebut. Detail dari uji coba dan analisis hasil dijelaskan pada sub bab di bawah ini.

6.1. Hasil Uji Coba

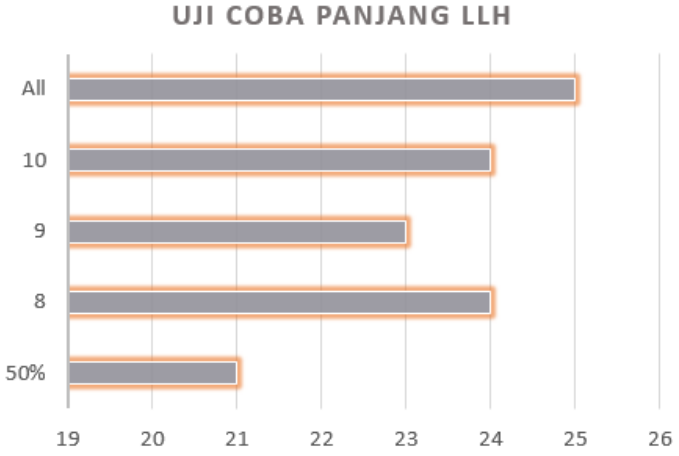
Berdasarkan hasil uji coba parameter *desired value* pada metode SADGED, nilai paling optimal yang didapatkan yaitu sebesar 10 persen dari nilai *initial solution*. Uji coba dilakukan dengan membandingkan jumlah nilai median dan minimum *fitness* yang lebih unggul dari metode SRSA. Eksekusi dilakukan terhadap persentase dari nilai *initial solution*. Perbandingan dapat dilihat pada gambar 6.1.



Gambar 6.1 Perbandingan Jumlah Nilai Median dan Nilai Minimum yang Lebih Baik SADGED Dibandingkan dengan SRSA

Berdasarkan uji coba pada tabel 5.1, nilai 10% unggul pada posisi pertama dimana menang pada 35 *instance* berdasarkan nilai minimum dan 35 *instance* pada nilai median. Berikutnya terdapat nilai parameter sebesar 1% dimana menang pada 68 *instance* , dimana selisih dua instance dengan nilai peringkat pertama. Oleh karena itu, parameter yang digunakan adalah 10% dari solusi awal.

Berdasarkan uji coba terhadap panjang LLH yang digunakan, penggunaan panjang LLH yang sesuai dengan banyaknya LLH pada setiap domain permasalahan memiliki hasil yang lebih baik. Perbandingan dilakukan dengan membandingkan nilai minimum pada algoritma SRSA. Penggunaan panjang sesuai jumlah LLH memberikan solusi menang sebanyak 25 *instance* menang dibandingkan dengan algoritma SRSA. Perbandingan dilakukan menggunakan parameter uji coba dengan *random seed* 1234 untuk metode *Hyper-heuristics* yang digunakan, sedangkan 5678 untuk domain permasalahan yang digunakan. Grafik uji coba dapat dilihat pada gamber 6.2.



Gambar 6.2 Grafik Uji Coba Panjang LLH yang Digunakan Pada Mekanisme Self Adaptive Learning

Algoritma *Simple Random Simulated Annealing* dan *Self Adaptive Learning Great Deluge* diterapkan pada enam domain permasalahan yang terdapat pada kerangka kerja HyFlex, yaitu *yaitu* SAT, Bin Packing, Flow Shop, Personnel Scheduling, TSP, dan VRP. Pada setiap domain permasalahan, masing – masing digunakan lima *instance* yang telah ditentukan untuk proses uji coba sehingga keseluruhan *instance* yang digunakan dalam uji coba berjumlah 30 *instance*. Setiap *instance* diuji sebanyak 31 kali dengan waktu 60000 milidetik. Algoritma *Simple Random Simulated Annealing* berperan sebagai algoritma pembanding dari *Self Adaptive Learning Great Deluge* dimana akan digunakan dalam perhitungan kinerja algoritma. Berdasarkan algoritma yang telah dibangun pada tahap sebelumnya. hasil uji metode SADED dan SRSA dapat dilihat pada LAMPIRAN E dan LAMPIRAN F. tabel 5.1 merupakan contoh hasil uji coba dari SRSA pada domain permasalahan SAT.

Tabel 6.1 Hasil Uji Coba Simple Random Simulated Annealing

Problem	Ins	Q1	Med	Q3	Average	Max	Min
SAT	3	26	29	32	28,64516	39	19
	4	38	40	42,5	39,74194	48	29
	5	54	57	59	55,90323	66	35
	10	26	29	32	28,83871	40	21
	11	12	14	14,5	13,29032	19	10

6.2. Analisis Hasil

pada bagian ini akan dilakukan analisis hasil atau evaluasi kinerja dari *hyper-heuristic Self Adaptive Learning Great Deluge*. Evaluasi ini bertujuan untuk mengetahui kinerja algoritma *Self Adaptive Learning Great Deluge* pada enam domain permasalahan yang terdapat pada kerangka kerja *HyFlex*. Analisis dilakukan dengan cara membandingkan algoritma SADGED dengan algoritma SRSA. Perbandingan dilakukan dengan mengacu pada nilai terbaik (minimum), kuartil pertama, median, kuartil ketiga, nilai maksimum, dan rata-rata nilai fungsi fitness setiap *instance* yang telah ditentukan. Perbandingan akan dilakukan dengan empat tahap, yakni :

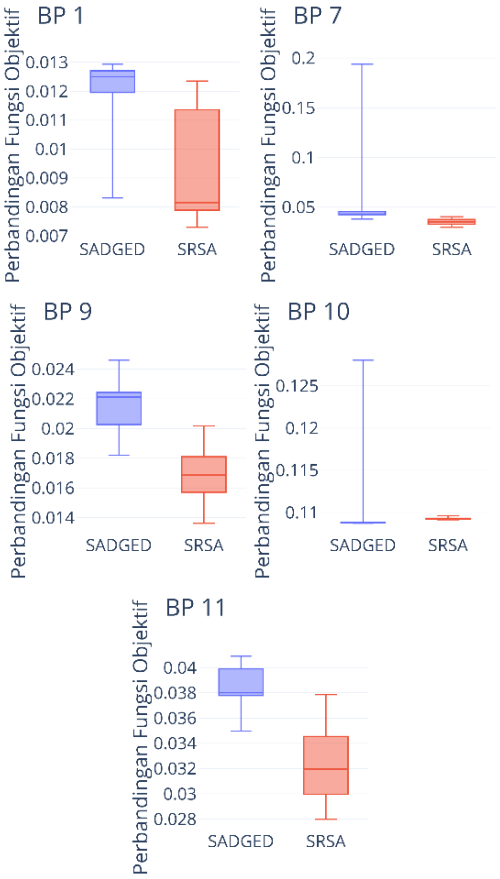
a) Pertama, membandingkan distribusi nilai yang dihasilkan oleh algoritma dengan menggunakan diagram *boxplot*.
b) Kedua, membandingkan nilai median untuk mengukur pemusatan data yang menjadi nilai tengah dari data.
c) Ketiga, membandingkan nilai fungsi objektif dari masing – masing *instance* untuk mengukur kinerja algoritma dalam menghasilkan nilai fungsi objektif.

Pada tahap pertama, analisis kinerja dilakukan dengan membandingkan diagram boxplot untuk mengetahui distribusi nilai dalam data pada masing – masing domain permasalahan. Diagram dengan nilai fungsi objektif lebih kecil menunjukan kinerja algoritma yang lebih baik. Hasil dari perbandingan kemudian diberikan nilai untuk mengetahui hasil pengujian. Setiap metode yang memiliki nilai fungsi objektif lebih kecil akan diberikan satu nilai. Penilaian ini memiliki jumlah maksimal sebanyak 150 dengan 25 poin pada setiap domain permasalahan.

Gambar 6.3 menunjukan SADGED memiliki kinerja yang lebih baik dibandingkan dengan SRSA pada setiap *instance* domain permasalahan SAT. Total nilai yang didapatkan oleh SADGED pada domain permasalahan SAT adalah 25, sedangkan SRSA mendapatkan poin 0 karena memiliki nilai fungsi objektif yang lebih rendah pada kelima *instance* yang diujikan. Pada domain permasalahan tersebut SADGED unggul pada kelima *instance* yang diujikan.

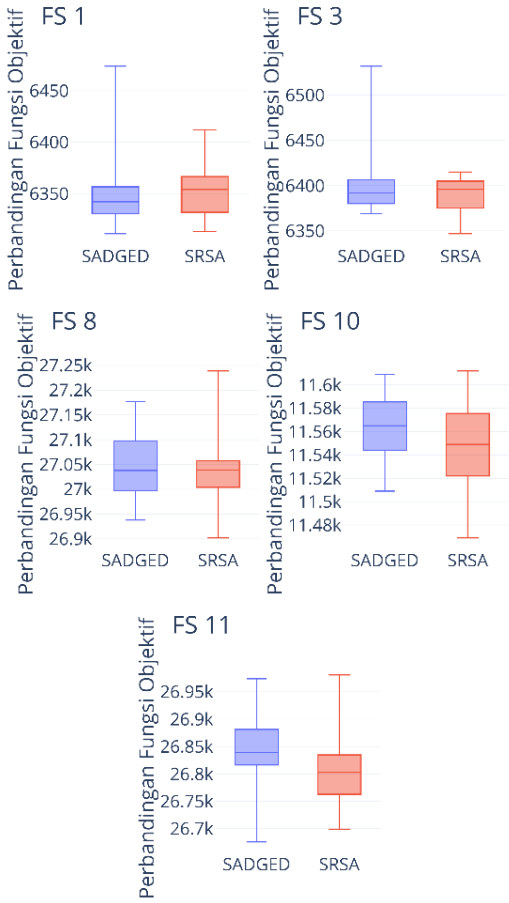
Gambar 6.3 Grafik Boxplot Pengujian pada domain SAT

Pada gambar 6.4 domain permasalahan *bin packing*, algoritma SADGED kalah sebesar 17 poin dari algoritma SRSA dengan perolehan nilai 21 poin untuk SRSA dan 4 poin untuk SADGED. Algoritma SADGED hanya mampu unggul pada *instance* 10 saja dan kalah pada keempat *instance* lainnya.



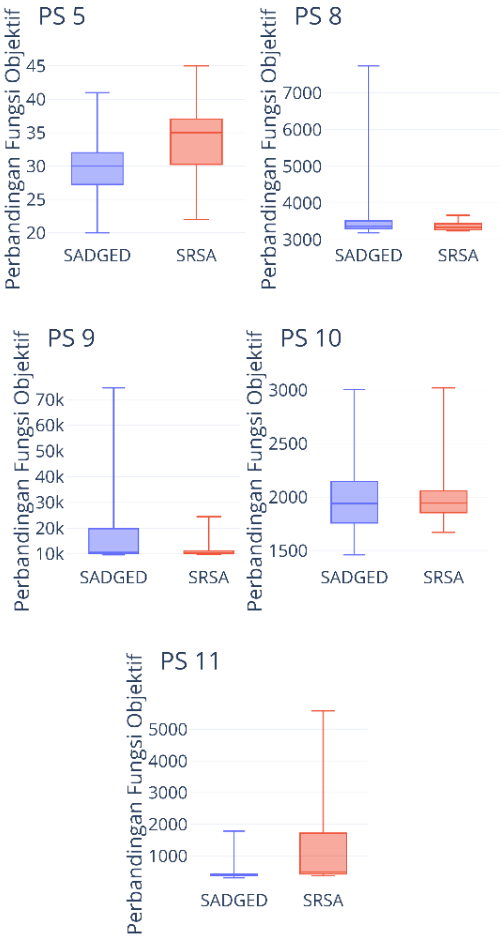
Gambar 6.4 Grafik Boxplot Pengujian pada domain Bin Packing

Pada gambar 6.5 domain permasalahan flow shop, SADGED hanya mampu unggul pada dua *instance* yang diujikan yakni *instance* 1 dan *instance* 8, sedangkan kalah pada tiga *instance* sisanya. SADGED mendapatkan skor sebesar 12 poin dan SRSA mendapatkan skor sebesar 13 poin dengan perbedaan 1 poin dimenangkan oleh SRSA.



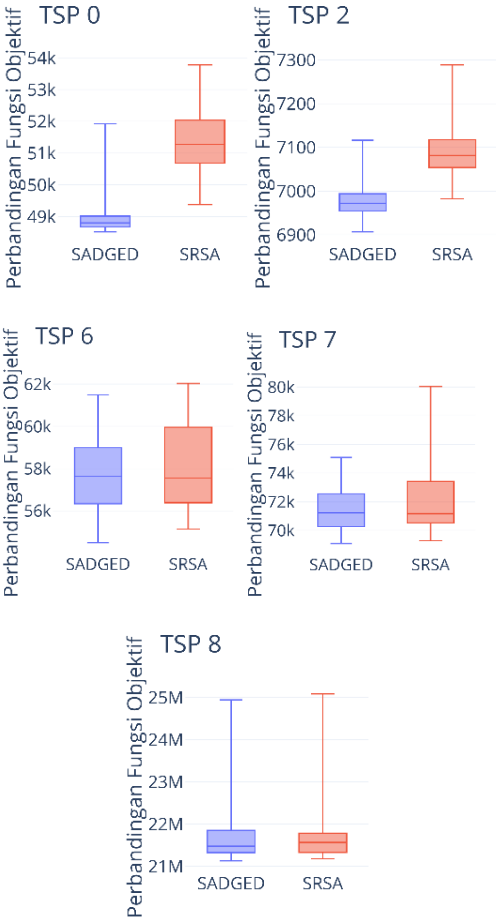
Gambar 6.5 Grafik Boxplot Pengujian pada domain Flowshop

Pada gambar 6.6 dimana menunjukan domain permasalahan personnel scheduling, SADGED unggul pada tiga *instance* yang diujikan , yakni *instance* 5, *instance* 10, dan *instance* 11, sedangkan kalah pada dua lainnya. Algoritma SADGED unggul 5 poin dari SRSA dengan mendapatkan poin sebesar 16.



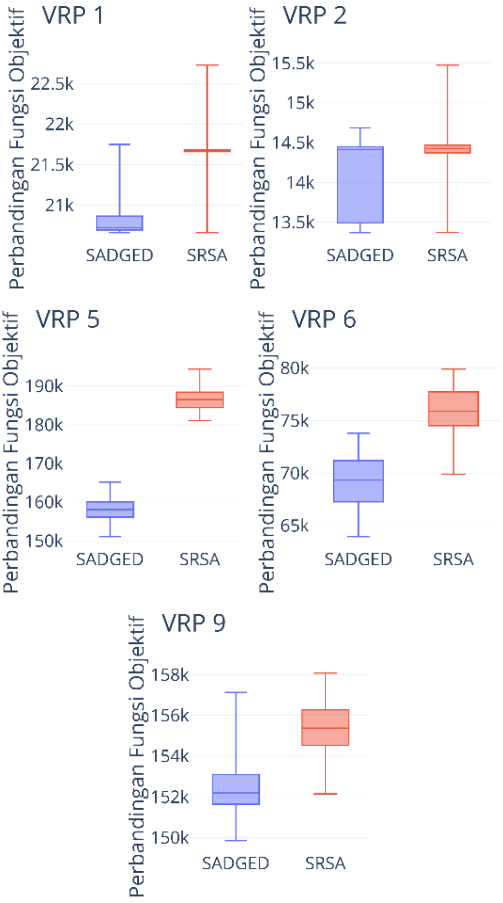
Gambar 6.6 Grafik Boxplot Pengujian pada domain Personnel Scheduling

Pada gambar 6.7, domain permasalahan tsp, SADGED unggul pada kelima *instance* dengan mendapatkan nilai 21, dimana unggul sebanyak 19 poin dari SRSA yang mendapatkan nilai sebesar 4.



Gambar 6.7 Grafik Boxplot Pengujian pada domain TSP

Pada gambar 6.8, domain permasalahan vrp, SAGDED unggul pada kelima *instance* yang ada dengan mendapatkan poin sebesar 25, sedangkan SRSA mendapatkan poin 0.



Gambar 6.8 Grafik Boxplot Pengujian pada domain VRP

Berdasarkan hasil pengujian dengan diagram box plot, algoritma SADGED unggul sebanyak 56 poin dibandingkan dengan SRSA dimana mendapatkan poin sebesar 103. Dari hasil diagram, dapat diketahui bahwa algoritma SADGED memiliki kinerja yang lebih baik daripada algoritma SRSA. Detail dari perbandingan kedua metode tersebut dapat dilihat pada LAMPIRAN G.

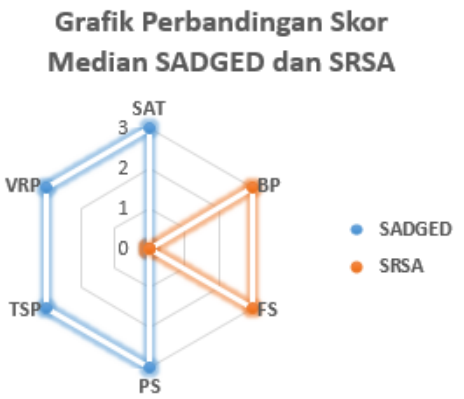
Pada tahap kedua, nilai median yang telah didapatkan diukur untuk mengetahui pemusatan data yang didapatkan. Pengukuran kedua ini menggunakan metode bola FIFA dengan melihat nilai dari median pada setiap *instance*. pada algoritma yang dianggap menang akan diberikan nilai tiga, kemudian bagi algoritma yang berimbang akan mendapatkan nilai satu dan algoritma yang kalah akan mendapatkan nilai kosong. Penentuan tersebut dihitung berdasarkan penjumlahan perubahan nilai dari algoritma. Perubahan nilai sendiri dapat dihitung dengan persamaan 5.1.

$$\Delta(\%) = \left(\frac{x-y}{y}\right) X100 \qquad (6.1)$$

Dalam persamaan 6.1, variabel x merupakan nilai fungsi objektif dari algoritma pembanding, dalam kasus ini merupakan algoritma SRSA. Sementara, variabel y merupakan nilai fungsi objektif dari algoritma SADGED. Apabila hasil positif, maka menunjukan peningkatan kinerja. Namun, apabila hasil negatif, maka algoritma menunjukan penurunan kinerja. Jadi, apabila hasil menunjukan nilai positif, maka metode SADGED dianggap menang dan mendapatkan poin sebesar tiga.

Berdasarkan perhitungan pada LAMPIRAN H menggunakan sistem bola FIFA, metode SADGED menang pada empat domain permasalahan, yaitu sat, personnel scheduling, tsp, dan vrp. Namun, metode ini kalah pada dua domain permasalahan lainnya, yaitu *Bin Packing* dan *Flowshop*. Hal ini menandakan bahwa algoritma SADGED tidak mampu memberikan hasil yang lebih baik pada dua domain permasalahan tersebut apabila

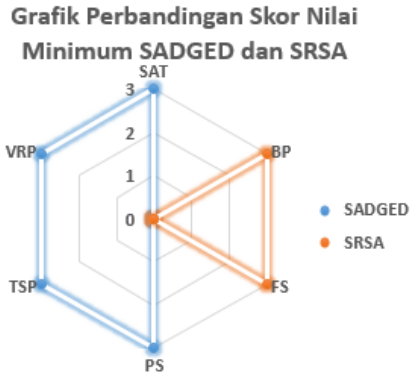
dibandingkan dengan algoritma SRSA dimana mengalami kekalahan yang cukup banyak. Grafik pengujian kinerja dapat dilihat pada gambar 6.9.



Gambar 6.9 Grafik Perbandingan Skor Median SADGED dan SRSA

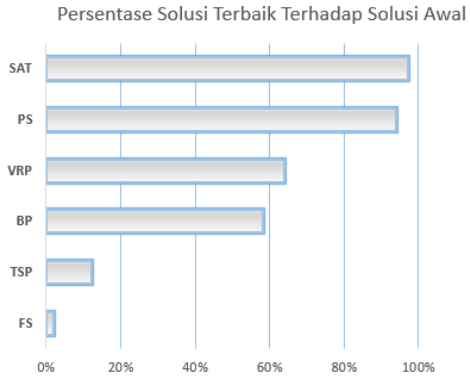
Pada tahap ketiga, nilai minimum dari hasil 31 kali eksekusi dilakukan perbandingan dengan menggunakan metode bola FIFA. Pengujian nilai minimum dapat dilihat pada LAMPIRAN I. metode yang memberikan persentase perubahan positif atau lebih tinggi dari nilai perubahan metode lainnya dianggap lebih baik dan diberikan skor tiga. Metode yang memberikan nilai imbang akan mendapatkan skor masing – masing satu dan metode yang kalah akan mendapatkan nilai kosong.

Berdasarkan data perhitungan pada LAMPIRAN I, nilai minimum SADGED secara keseluruhan menang pada empat domain permasalahan, yakni SAT, Personnel Scheduling, TSP, dan VRP. Namun, pada domain permasalahan Bin Packing dan Flowshop metode SADGED kalah bersaing dengan SRSA. Kekalahan yang paling signifikan terdapat pada domain permasalahan Bin Packing dimana hanya dapat memenangkan pada satu *instance* saja untuk nilai minimum. Grafik pengujian kinerja dapat dilihat pada gambar 6.10.



Gambar 6.10 Grafik Perbandingan Skor Minimum SADGED dan SRSA

Domain permasalahan SAT mengalami peningkatan kualitas solusi akhir terbaik dibandingkan dengan domain permasalahan lainnya. peningkatan pada domain permasalahan ini sebesar 97% dari solusi awal pada kelima *instance* yang diujikan. Sedangkan pada domain permasalahan FS, peningkatan solusi akhir atau solusi terbaik mendapatkan persentase paling rendah, yakni 2% dari solusi awal yang dihasilkan. Grafik persentase peningkatan kualitas solusi dapat dilihat pada gambar 6.11 serta LAMPIRAN J.



Gambar 6.11 Persentase Kenaikan Kualitas Solusi Berdasarkan Solusi Awal

Selanjutnya melakukan evaluasi kinerja dari algoritma dengan menggunakan sistem Formula One. Sistem ini digunakan untuk menjaga persaingan algoritma yang adil. Sistem ini menggunakan data median dalam proses evaluasinya. Nilai median ini berdasarkan hasil dari nilai fungsi objektif yang telah didapatkan, kemudian diberi poin dan dijumlahkan keseluruhan poin tersebut. Dalam sistem ini, kinerja delapan peringkat terbaik diberikan poin 10,8,6,5,4,3,2,1. Jika terdapat lebih dari delapan algoritma, maka akan diberikan peringkat 0 untuk algoritma dengan urutan lebih dari delapan. Jika peserta memiliki nilai yang sama, maka akan diberikan nilai yang sama sesuai dengan peringkatnya [1]. Namun, karena peserta pada persaingan ini hanya terdiri dari dua algoritma, poin tertinggi yang diberikan adalah 2 untuk peringkat pertama dan 1 untuk peringkat kedua. Skor maksimum untuk setiap domain permasalahan adalah 120. Algoritma yang memiliki jumlah poin yang lebih banyak dianggap memiliki kinerja yang lebih baik dari algoritma lainnya. Tabel 6.2 merupakan perhitungan skor untuk menentukan peringkat algoritma berdasarkan sistem Formula One dengan mengacu pada data median. Warna hijau menandakan nilai yang memberikan hasil lebih baik dari pada algoritma lainnya.

Tabel 6.2 Hasil Perbandingan Nilai Median Algoritma dengan Sistem Formula One

Problem	Instance	SADGED	Poin	SRSA	Poin
SAT	3	6	2	29	1
	4	9	2	40	1
	5	8	2	57	1
	10	9	2	29	1
	11	9	2	14	1
BP	1	0,012499	1	0,008143	2
	7	0,043092	1	0,035139	2
	9	0,022124	1	0,016872	2

Problem	Instance	SADGED	Poin	SRSA	Poin
	10	0,10876	2	0,109267	1
	11	0,038005	1	0,031955	2
FS	1	6342	2	6354	1
	3	6392	2	6396	1
	8	27038	2	27039	1
	10	11565	1	11549	2
	11	26839	1	26803	2
PS	5	30	2	35	1
	8	3362	1	3335	2
	9	10576	1	10362	2
	10	1939	2	1945	1
	11	395	2	490	1
TSP	0	48805,37	2	51271,54	1
	2	6972,085	2	7081,726	1
	6	57642,37	1	57560,79	2
	7	71236	1	71170,41	2
	8	21477414	2	21566445	1
VRP	1	20719,33	2	21672,03	1
	2	14413,72	2	14426,11	1
	5	158067,4	2	186522,5	1
	6	69325,76	2	75886,81	1
	9	152188,7	2	155372,1	1
Total Poin		50		40	
Rata - Rata Poin		1,666666667		1,333333333	
% Poin		83%		67%	

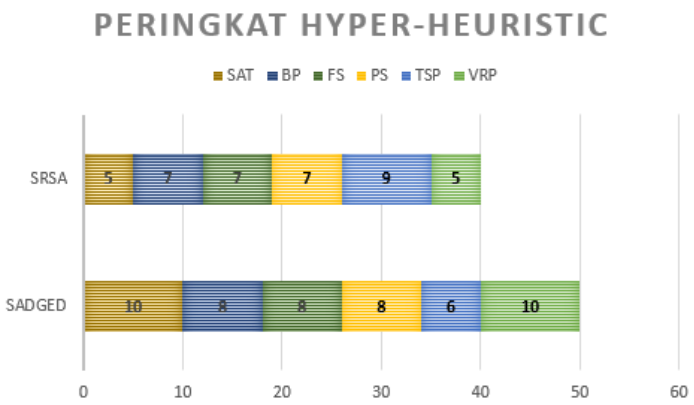
Skor setiap *instance* selanjutnya dijumlahkan untuk mengetahui persentase total skor seluruh *instance*. Perhitungan persentase total skor dapat dilihat pada persamaan 5.2.

$$Total\ Skor(\%) = \left(\frac{X}{Y}\right) x\ 100 \qquad (6.2)$$

Dimana, variabel X adalah total skor, sedangkan variabel Y adalah jumlah skor maksimal pada satu algoritma. Berdasarkan hasil perhitungan skor seluruh *instance* diatas, berikut diketahui peringkat dari metode yang digunakan :

a. Peringkat pertama diperoleh oleh metode SADGED dengan perolehan skor 50 poin dari total skor yang dapat diperoleh sebesar 60 poin. Metode SADGED mendapatkan persentase poin sebesar 83 % dengan rata – rata poin yang didapatkan pada setiap *instance* adalah 1,6.
b. Peringkat kedua diperoleh oleh SRSA dengan perolehan skor 40 poin dari total skor 60 poin. Metode ini mendapatkan persentase poin sebesar 67% dengan rata – rata poin pada setiap *instance* sebesar 1,3.

Berdasarkan hasil pengujian dengan menggunakan sistem Formula One, metode SADGED mampu memberikan kinerja yang lebih baik dalam memberikan solusi yang lebih optimal, khususnya pada empat domain permasalahan yang ada yakni SAT, Personnel Scheduling, TSP, dan VRP. Urutan peringkat dari perbandingan metode berdasarkan kinerja pada seluruh domain permasalahan dapat dilihat pada gambar 6.12.



Gambar 6.12 Peringkat Strategi Hyper-heuristic berdasarkan Sistem Formula One Menggunakan Data Median

(Halaman Ini Sengaja Dikosongkan)

BAB VII
KESIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan dan saran yang dapat diambil berdasarkan seluruh proses pengerjaan tugas akhir ini.

7.1. Kesimpulan

Berdasarkan hasil uji coba dan analisis hasil yang telah dilakukan pada bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut :

a. Kombinasi *Self Adaptive Learning Great Deluge* mampu memberikan hasil yang lebih baik dibandingkan dengan algoritma *Simple Random Simulated* Annealing. SADGED mampu memberikan hasil yang lebih baik pada empat domain permasalahan, yakni *SAT, Bin Packing, TSP,* dan *VRP*.
b. Metode SADGED yang digunakan kurang baik dalam mencari solusi optimal bagi permasalahan *Bin Packing*. SADGED hanya mampu memberikan solusi optimal pada satu dari lima *instance* dalam domain permasalahan tersebut dan mendapatkan penurunan perubahan sebesar sekitar 78% apabila dilihat dari nilai minimum dan 92% pada nilai median.
c. Walaupun terjadi penurunan pada domain permasalahan *flowshop*, penurunan tersebut dianggap tidak terlalu signifikan. SADGED mampu memberikan solusi optimal bagi dua *instance* pada domain permasalahan tersebut apabila dilihat dari sisi minimum, dan dapat memberikan solusi optimal pada tiga *instance* apabila dilihat dari nilai median.
d. Berdasarkan hasil pengujian menggunakan sistem Formula One, Metode SADGED memiliki kinerja yang lebih baik dibandingkan dengan algoritma SRSA. Hal ini dapat dilihat dari skor persentase kinerja yang didapatkan sebesar 83% dengan poin sebesar 50 dari

total 60 poin yang diuji pada 30 *instance* di enam domain permasalahan yang berbeda. Metode SADGED unggul sebesar 16% dari metode SRSA sebagai algoritma pembanding.

7.2. Saran

Berdasarkan hasil pengerjaan tugas akhir, pada penelitian selanjutnya dapat dilakukan eksperimen dalam menentukan parameter jumlah batas LLH yang digunakan dalam mencari nilai fungsi objektif yang optimal pada metode *Self Adaptive Learning*. Dengan penentuan jumlah yang tepat, diharapkan mampu memberikan hasil yang lebih baik dalam mencari solusi yang optimal, khususnya bagi domain permasalahan *Bin Packing* dan *Flowshop*.

DAFTAR PUSTAKA

[1] 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.

[2] 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.

[3] E. K. Burke and G. Kendall, *Search Methodologies*, 2nd ed. .

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

[5] J. D. Walker, "Design of Vehicle Routing Problem Domains for a Hyper-Heuristic Framework," 2015.

[6] R. A.-M. Nabeel, "Hybrid genetic algorithms with great deluge for course timetabling," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 4, pp. 283–288, 2010.

[7] A. Acan and A. Ünveren, "A two-stage memory powered Great Deluge algorithm for global optimization," *Soft Comput.*, vol. 19, no. 9, pp. 2565–2585, 2015.

[8] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Inf. Sci. (Ny).*, vol. 181, no. 20, pp. 4515–4538, 2011.

[9] A. Angresti, Nisa Dwi; Muklason, Ahmad; Djunaedy,

"Penyelesaian Permasalahan Optimasi Lintas Domain Dari Hyflex Menggunakan Hiperheuristik Yang Didasarkan Pada Metode Variable Neighborhood Search," Institut Teknologi Sepuluh Nopember, 2019.

[10] W. G. Jackson, E. Ozcan, and J. H. Drake, "Late acceptance-based selection hyper-heuristics for cross-domain heuristic search," *2013 13th UK Work. Comput. Intell. UKCI 2013*, no. September, pp. 228–235, 2013.

[11] C. Alabas-Uslu and B. Dengiz, "A self-adaptive local search algorithm for the classical vehicle routing problem," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8990–8998, 2011.

[12] P. Franq, "Optimization Problem," *Paul Otlet Institute*, 2012. [Online]. Available: http://www.otlet-institute.org/wikics/Optimization_Problems.html. [Accessed: 21-Feb-2019].

[13] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci. (Ny).*, vol. 237, pp. 82–117, 2013.

[14] E. K. Burke *et al.*, "A Classification of Hyper-heuristic Approaches," 2009.

[15] E. Burke, T. Curtois, M. Hyde, G. Ochoa, and J. A. Vazquez-Rodriguez, "HyFlex: A Benchmark Framework for Cross-domain Heuristic Search," no. January, 2011.

[16] N. R. Sabar and G. Kendall, "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Inf. Sci. (Ny).*, vol. 314, pp. 225–239, 2015.

[17] P. Roblem, "Hyper Heuristic Based on Great Deluge and Its Variants for Exam Timetabling."

[18] N. Nahas and M. Nourelfath, “Iterated great deluge for the dynamic facility layout problem,” *Oper. Res. Comput. Sci. Interfaces Ser.*, vol. 60, pp. 57–92, 2016.

[19] P. Eles, K. Kuchcinski, and Z. Peng, “Optimization Heuristics,” *System Synthesis with VHDL*. pp. 137–163, 2013.

[20] Q. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, “A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem,” *Inf. Sci. (Ny).*, vol. 181, no. 12, pp. 2455–2468, 2011.

[21] P. C. Hsiao, T. C. Chiang, and L. C. Fu, “A VNS-based hyper-heuristic with adaptive computational budget of local search,” *2012 IEEE Congr. Evol. Comput. CEC 2012*, pp. 1–8, 2012.

[22] M. Hyde, G. Ochoa, V. Antonio, and T. Curtois, “A HyFlex Module for the MAX-SAT Problem,” no. January, 2015.

[23] U. Eliiyi and D. Tursel, “Applications of Bin Packing Models Through the Supply,” *Int. J. Bus. Manag.*, vol. 1, no. 1, pp. 11–19, 2009.

[24] G. Ochoa and T. Curtois, “A HyFlex Module for the Permutation Flow Shop Problem,” pp. 1–4.

[25] P. Brucker, R. Qu, E. Burke, and C. P. Brucker, “Personnel Scheduling : Models and Complexity Personnel Scheduling : Models and Complexity,” 2009.

[26] E. Burke, T. Curtois, H. Gabriela, and O. Jos, “HyFlex: A Benchmark Framework for Cross-domain Heuristic Search,” pp. 1–28, 2011.

[27] G. Ochoa *et al.*, “HyFlex : A Benchmark Framework for Cross-domain Heuristic Search,” no. May 2014, 2011.

[28] M. Hyde, G. Ochoa, V. Antonio, and T. Curtois, “A HyFlex Module for the One Dimensional Bin Packing Problem,” pp. 1–5.

[29] G. N. Gan, T. L. Huang, and S. Gao, “Genetic simulated annealing algorithm for task scheduling based on cloud computing environment,” *Proc. - 2010 Int. Conf. Intell. Comput. Integr. Syst. ICISS2010*, pp. 60–63, 2010.

BIOGRAFI PENULIS

Widya Saputra, lahir di Magelang, 9 Mei 1996. Penulis menempuh pendidikan formal di SD Muhammadiyah 01 Sidoarjo, SMP Muhammadiyah 1 Sidoarjo, dan SMA Negeri 2 Sidoarjo. Pada tahun 2015 penulis melanjutkan pendidikan ke jenjang S1 di Program Studi Sistem Informasi Institut Teknologi Sepuluh Nopember, Surabaya. Penulis sempat mengikuti kegiatan PASKIBRAKA, Pramuka dan Kerohanian semasa menjalani pendidikan SMA. Selama menjalani perkuliahan penulis aktif berpartisipasi dalam organisasi keislaman Kajian Islam Sistem Informasi (KISI) serta Himpunan Mahasiswa Sistem Informasi. Pada Pengerjaan Tugas Akhir ini penulis mengambil konsentrasi Rekayasa Data dan Inteligen Bisnis dengan topik *Hyper-heuristic*. Untuk kritik dan saran yang membangun dapat disampaikan melalui email widyasaputra@engineer.com.

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN A HASIL EKSEKUSI *SIMPLE RANDOM SIMULATED ANNEALING*

1. SAT

ins	RUN																															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	17	16	12	15	20	16	11	21	14	12	13	15	11	13	15	18	12	13	8	12	20	9	9	17	15	14	11	9	11	11	15
1	29	27	27	27	26	24	25	30	26	35	28	28	26	27	39	26	28	28	37	29	31	26	26	26	26	32	29	32	32	30	27
2	23	18	19	31	21	21	23	21	25	22	23	23	21	25	24	25	29	19	24	29	25	25	27	26	27	25	21	17	24	23	25
3	23	34	24	20	31	29	25	32	27	33	21	39	23	33	23	30	30	32	19	35	31	32	28	27	29	31	27	33	30	28	29
4	40	42	43	46	39	33	38	40	40	38	42	32	39	44	38	48	44	35	35	43	41	42	34	29	41	41	40	35	40	45	45
5	62	46	58	51	58	56	55	62	65	57	61	66	51	55	54	60	57	35	51	57	47	64	54	55	55	56	57	58	51	62	57
6	9	9	8	8	13	11	10	11	11	11	10	11	10	10	9	11	8	9	7	12	8	9	9	8	9	9	10	8	13	13	8
7	11	12	9	12	12	11	9	8	9	7	9	10	11	10	8	9	9	13	8	8	13	11	8	7	9	12	10	11	9	8	14

8	12	15	14	12	13	14	15	15	13	10	13	13	12	12	11	13	14	15	13	9	11	12	17	13	14	16	11	14	8	13	10
9	213	215	215	215	223	221	213	216	217	221	223	229	213	219	213	215	215	215	219	217	217	215	221	215	225	215	215	215	221	213	213
10	33	33	33	26	26	24	29	27	21	27	31	33	29	22	34	28	27	26	28	40	29	30	25	34	25	25	30	30	25	29	35
11	15	12	14	13	12	14	12	14	12	15	11	13	14	14	10	19	14	14	16	11	15	11	10	11	15	15	15	13	13	14	11
2. Bin Packing																																
ins	RUN																															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	0,0107555	0,0100952	0,0107844	0,0075445	0,0113669	0,0077006	0,0110266	0,0107227	0,010777	0,0121447	0,0112222	0,0076691	0,0,0108	0,0112889	0,0070966	0,0076688	0,0109833	0,0112444	0,0065887	0,0075211	0,0074944	0,0110001	0,0113008	0,0076227	0,0108077	0,0110762	0,0119333	0,0075366	0,0070884	0,0075822	0,0107688

1	0,00761	0,008041	0,007699	0,007613	0,008244	0,007293	0,012256	0,008205	0,007717	0,008266	0,007852	0,008243	0,00804	0,011422	0,007882	0,011538	0,012353	0,00812	0,011334	0,007949	0,008263	0,00809	0,008143	0,007692	0,011874	0,011397	0,008204	0,011406	0,007918	0,007898	0,011365
2	0,023094	0,023839	0,022355	0,025258	0,024904	0,022406	0,023854	0,026778	0,027305	0,023742	0,024091	0,023799	0,023911	0,026064	0,023291	0,022337	0,026732	0,023137	0,023099	0,025183	0,024438	0,02343	0,023415	0,023474	0,02231	0,022801	0,023335	0,024706	0,024705	0,024664	0,023082
3	0,026652	0,024124	0,026201	0,026831	0,027186	0,025788	0,026012	0,023418	0,023469	0,022439	0,024874	0,024563	0,024241	0,024752	0,025017	0,029352	0,022222	0,024211	0,025851	0,024794	0,023947	0,022956	0,024227	0,025896	0,025777	0,028107	0,026739	0,024187	0,023154	0,022227	0,02401

4	0,00456	0,00503	3,43E-04	0,00456	0,00503	3,47E-04	3,43E-04	0,00456	0,00456	0,00456	0,00456	0,00457	0,00456	0,00508	0,00456	0,00504	0,00456	0,00478	0,00456	0,00457	0,00456	0,00456	0,00460	0,00506	0,00456	0,00456	3,45E-04	0,00508	0,00460	0,00456	0,00456	0,00456	0,00456
5	0,00340	0,00033	0,00034	0,00036	0,00036	0,00031	0,00035	0,00035	0,00035	0,00033	0,00033	0,00035	0,00033	0,00034	0,00035	0,00032	0,00031	0,00033	0,00032	0,00032	0,00033	0,00033	0,00030	0,00033	0,00033	0,00033	0,00033	0,00033	0,00033	0,00033	0,00035	0,00034	0,00035	0,00035
6	0,01107	0,01164	0,01615	0,01646	0,01518	0,01524	0,01642	0,01534	0,01638	0,01604	0,01530	0,01072	0,01648	0,01596	0,01598	0,01645	0,01596	0,01123	0,01648	0,01599	0,01182	0,01532	0,01642	0,02042	0,01634	0,02053	0,01630	0,01515	0,01608	0,01614	0,01606	0,01633	0,01636	0,01636

7	0,04	0,03997	0,035327	0,035468	0,035139	0,037784	0,034878	0,032923	0,032916	0,035377	0,035402	0,03772	0,029717	0,038034	0,030315	0,032629	0,037307	0,03538	0,030229	0,040377	0,032912	0,032914	0,035372	0,032555	0,032152	0,032893	0,037776	0,032829	0,032574	0,037718	0,030143
8	0,063139	0,063067	0,056635	0,059037	0,05997	0,058289	0,063901	0,062467	0,063223	0,057512	0,058323	0,060601	0,05993	0,062172	0,061564	0,059215	0,062117	0,063213	0,060853	0,055989	0,06004	0,062329	0,06152	0,060002	0,063112	0,061428	0,062227	0,062943	0,06401	0,063106	0,06222
9	0,013621	0,016032	0,015801	0,014559	0,01783	0,01577	0,019056	0,016034	0,018143	0,015632	0,020171	0,019008	0,017955	0,018048	0,01705	0,016755	0,018007	0,016868	0,016872	0,019091	0,014965	0,017113	0,015688	0,018128	0,018133	0,017995	0,015868	0,014688	0,014584	0,020113	0,015674

10	0,1092773	0,1092733	0,1092933	0,1095331	0,1092911	0,1092511	0,1092234	0,1092224	0,1092772	0,1094776	0,1092234	0,1091555	0,1092211	0,1092243	0,1092555	0,1091197	0,1093334	0,1091955	0,1096033	0,1092555	0,1093755	0,1092779	0,1092199	0,1093975	0,1092207	0,1091014	0,1092377	0,1093	0,1092667	0,1093115
11	0,0300644	0,0299455	0,0320223	0,0339355	0,0378556	0,0310888	0,0289922	0,0319449	0,0319882	0,0350339	0,03559	0,0347966	0,0329411	0,0328115	0,0299222	0,0319555	0,0288522	0,0300744	0,0319581	0,0350911	0,0347452	0,0299559	0,0330411	0,0290244	0,0318511	0,0318744	0,0359552	0,0299888	0,0358587	0,0299844	0,0279782

3. Flowshop																																	
ins	RUN																																
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	64211	63776	64332	63999	64224	64003	64211	64119	63779	63900	64115	63800	63385	63888	63886	63365	63955	63888	64211	63942	63382	63361	63655	64004	63800	63668	63362	64100	63933	64002	63638
1	63700	63365	63362	63663	63233	63553	63324	63361	63399	63382	63346	63413	63362	63183	63300	63331	63355	63379	63341	63361	63374	63443	63375	63611	63300	64112	63200	63700	63667	63433	63543
2	64333	64002	64005	64233	64007	63996	64015	63981	64311	64442	64437	64425	64428	64456	63999	64007	64430	64423	64430	64404	64463	64431	64422	64414	63844	64008	64444	63999	64311	64419	63933
3	64005	63900	64005	63647	63977	64008	63866	63396	63669	63370	63777	64008	63775	64008	63988	63969	63375	63988	63900	63666	63395	63933	64115	64415	64004	63977	63378	64006	64408	63700	63999
4	64	64	64	64	65	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	65	65	64	64	64	64	64	64	64	64

	66	15	45	58	08	67	58	55	54	80	53	81	47	18	51	42	50	33	41	79	77	06	00	69	33	59	53	97	73	81	73
5	10572	10556	10569	10565	10559	10538	10546	10553	10565	10569	10546	10562	10556	10579	10552	10560	10575	10544	10550	10543	10593	10553	10631	10556	10557	10572	10549	10581	10577	10572	10577
6	11017	10965	11017	10965	11017	10965	10974	10944	11017	11017	11017	10965	10976	10972	11017	10972	11017	10957	11017	10980	10963	10965	10946	10933	10965	10944	10965	10956	11017	10965	10965
7	26507	26520	26456	26479	26544	26594	26514	26545	26672	26508	26436	26511	26531	26592	26588	26489	26619	26422	26515	26484	26601	26494	26533	26512	26535	26410	26607	26571	26602	26673	26536
8	27033	27005	27069	27046	27077	27239	27101	27121	26960	27004	27039	26945	27058	27056	27032	27002	27040	27011	26959	27046	27018	26998	27052	27091	27004	27014	26999	27015	27079	27052	26902
9	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26

	827	808	835	953	831	867	843	808	742	900	874	794	874	851	796	885	854	897	891	862	848	803	842	798	855	849	831	815	794	891	897
10	11519	11564	11566	11581	11528	11576	11488	11541	11602	11612	11519	11500	11554	11576	11555	11570	11530	11520	11574	11469	11546	11548	11590	11549	11576	11536	11604	11519	11566	11502	11543
11	26727	26769	26878	26866	26803	26834	26799	26735	26981	26894	26780	26737	26699	26835	26786	26808	26807	26723	26844	26811	26876	26770	26789	26848	26761	26815	26780	26717	26813	26821	26718

4. Personnel Scheduling

ins	RUN																													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	3344	3362	3410	3358	3408	3410	3352	3383	3311	3369	3389	3391	3320	3311	3343	3418	3338	3383	3359	3355	3314	3345	3356	3391	3410	3420	3361	3370	3320	3383	3423
1	2812	2820	3293	3240	2523	2420	2479	2425	2210	2665	2398	2530	2220	2735	2705	2620	2285	2315	2697	2580	2770	2205	2669	2260	2265	3399	2600	2534	2299	2525	3610
2	475	485	1735	465	410	445	500	475	1730	1820	2020	420	1935	525	490	485	440	1800	565	1735	470	430	405	1810	1945	590	525	535	520	1805	435
3	33	22	30	22	28	34	24	20	23	26	29	21	26	24	21	27	26	32	23	21	37	28	30	30	29	29	29	30	39	33	30
4	24	27	35	40	26	28	24	27	31	30	26	30	37	36	33	40	29	28	51	35	26	30	30	33	29	37	31	27	33	22	28
5	37	34	25	30	38	35	41	34	45	35	35	31	32	28	29	24	31	35	37	22	41	38	36	32	37	30	42	36	26	38	34

6	1309	1123	1235	1419	1106	1329	1128	1313	1416	1450	1411	1426	1297	1210	1209	1834	1199	1285	1324	1123	1220	1575	1426	1135	1122	1226	1329	1219	1425	1414	1222
7	2279	2282	2285	2463	2304	2373	2278	2179	2387	2474	2286	2367	2309	2302	2495	2389	2488	2404	2294	2211	2283	2402	2608	2372	2372	2585	2571	2192	2223	2212	2385
8	3525	3266	3335	3335	3320	3259	3340	3271	3267	3460	3354	3238	3437	3311	3346	3555	3282	3364	3498	3377	3293	3470	3266	3275	3505	3254	3276	3418	3665	3239	3259
9	10362	11231	98555	97666	11147	10009	10940	10749	12081	24479	10192	12094	11379	12415	10057	10307	97833	97733	10837	99741	10820	11031	10696	10177	11215	10212	10135	10245	99683	98355	10615
10	1865	1885	2217	1853	2065	1823	2030	2855	2038	2140	2395	1963	1955	3020	2230	1905	1740	1940	1814	1900	2183	1853	1950	1670	1858	2020	1983	1880	1840	1748	1945
11	17	415	380	410	18	30	17	395	395	17	490	17	16	450	16	495	430	420	505	55	18	480	435	440	17	445	440	17	16	450	395

	80				35	70	25			15		15	65		90					80	06				35			55	95		
5. TSP																															
ins	RUN																														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	512532,32	528043,33	537763,29	505806,65	510020,09	517155,65	512715,54	502763,32	521233,97	517866,91	517166,52	502044,76	516044,36	522222,44	512666,22	506566,28	509815,25	522855,84	493783,85	523133,39	495592,2	520743,43	502048,86	521388,62	518488,08	512755,45	507493,14	519176,19	501628,28	503188,53	510992,72
1	116360	116134	118423	114684	116549	115875	120020	118468	119642	119578	115494	116820	116354	114984	114509	114536	114274	117237	115635	117828	118676	117122	115449	117393	119311	112135	117437	114899	115560	114559	116086

	,1		,8			,1	,7	,7	,1	,1	,5	,2	,2	,3	,9	,3	,5	,8	,5	,8	,5	,3	,6	,5	,2		,5		,7	,2	,4
2	7195,48	7201,947	7015,55	7102,991	7052,164	7040,021	7041,788	7081,726	7082,02	7125,672	7114,011	6982,597	7118,617	7097,752	7087,145	7029,022	7059,878	7061,23	6991,389	7069,512	7076,427	7176,012	7189,832	7077,958	7050,641	7100,735	7082,578	7288,849	7072,608	7170,397	7066,523
3	44243,77	45874,63	43736,71	43818,57	43378,31	44445,93	44547,46	43870,71	44431,47	44024,52	43724,9	43960,28	44007,15	45006,25	43960,63	43965,37	43874,54	44131,69	43618,57	44871,68	43887,62	43856,35	43818,66	43505,16	44225,6	44152,37	44538,24	44524,44	44436,36	43851,85	43916,25
4	9193,1	9332,1	9318,2	9253,98	9207,5	9321,9	9257,8	9194,21	9152,9	9306,9	9208,8	9175,99	9227,9	9387,2	9278,0	9365,0	9294,1	9247,1	9286,1	9281,7	9232,1	9284,7	9396,8	9301,4	9261,8	9491,6	9414,5	9322,6	9359,49	9273,38	9265,2

	21	82	23		47	33	34		84	28	82		19	76	82	32	93	59	01	55	59	32	79	78	03	36	02	91			14
5	617922,23	611622,46	606955,05	60661,42	60854,8	612222,78	611122,49	64294,65	613667,63	60660,21	62083,53	61586,37	60804,06	61278,04	60399,77	60305,37	60521,6	61859,51	60804,19	60608,09	60537,59	63134,18	60958,6	60531,5	63007,45	61559,87	61199,27	61923,36	61811,84	61154,23	617179,52
6	603448,45	597244,08	600088,58	56446,06	56549,9	568655,35	56660,7	555569,84	551154,13	60502,32	587444,46	60044,84	55559,27	55925,34	57589,04	581296,26	56363,35	57180,58	551355,68	57764,54	58418,45	56334,78	59843,62	56661,95	61486,69	55617,32	57532,79	57560,79	60256,35	60781,07	620420,65
7	708581,,	712558,,	708211,,	80040,,	79313,,	722275,5	70584,,	70311,,	69741,,	70510,,	72706,,	70330,,	78912,,	79356,,	72029,,	78543,,	708531,1	73254,,	69289,,	70877,,	72009,,	77073,,	69405,,	71190,,	70724,,	79825,,	71170,,	69794,,	70592,,	704466,1	734880,,

	82	85	36	27	51		22	78	42	58	79	72	97	75	82	13		39	35	46	39	49	02	01	38	17	41	06	12		67
8	2,17E+07	2,18E+07	2,20E+07	2,14E+07	2,16E+07	2,16E+07	2,17E+07	2,49E+07	2,13E+07	2,51E+07	2,16E+07	2,48E+07	2,18E+07	2,18E+07	2,14E+07	2,15E+07	2,15E+07	2,14E+07	2,12E+07	2,12E+07	2,13E+07	2,12E+07	2,16E+07	2,13E+07	2,15E+07	2,17E+07	2,13E+07	2,13E+07	2,13E+07	2,51E+07	2,16E+07
9	792951,4	681425,3	677502,1	791959,3	677609	682642,4	678953,6	679446,7	789254,2	679296,9	675390,4	677427,9	676124,5	679273,5	681050,8	683895,1	676269,3	680187,1	679257,6	774921,1	680100,8	676454,2	678636,6	677167,1	678237,9	681433,4	676970,3	798795,3	681711,9	680515,9	677775,7

6. VRP																																		
ins	RUN																																	
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	620,525	615,537	709,682	614,858	715,736	619,504	621,428	721,253	617,063	717,326	606,437	715,285	620,797	613,862	612,274	616,753	615,584	709,087	616,283	526,744	719,869	722,427	612,243	616,475	815,596	616,378	720,166	716,565	613,899	809,339	713,597
1	2006,211	2166,735	2165,206	2172,419	2166,881	2166,786	2166,704	2171,995	2166,421	2006,941	2165,784	2268,471	2166,958	2006,01	2165,914	2169,612	2167,736	2164,97	2166,641	2272,607	2167,693	2165,963	2168,67	2166,311	2069,135	2168,689	2166,482	2171,387	2167,82	2167,203	2167,872
2	14	14	13	14	14	14	14	14	14	15	14	14	13	13	14	14	13	13	14	14	14	15	14	14	14	15	14	14	15	14	13

	431,59	472,68	397,66	422,28	405,22	432,81	522,68	370,15	387,4	461,35	472,87	446,37	397,74	406,57	441,1	390,24	369,48	374,39	365,13	421,07	473,56	474,93	390,51	426,11	433,36	435,72	427,3	453,2	458,46	374,21	403,19
3	830 6, 135	630 0, 918	730 1, 038	833 7, 623	826 6, 554	740 0, 923	739 0, 48	734 1, 314	732 0, 604	823 0, 727	724 5, 835	724 4, 046	733 1, 205	733 6, 587	733 03, 872	827 9, 621	733 0, 003	733 0, 609	733 01, 201	733 60, 672	733 15, 221	728 6, 906	103 51, 13	831 12, 405	733 15, 616	733 81, 601	728 9, 723	722 76, 522	827 4, 508	730 9, 156	833 8, 113
4	1432 4, 22	1533 47, 03	1533 45, 95	1637 8, 73	1529 0, 2	1432 6, 13	1533 19, 33	1535 57, 85	1535 318, 48	1535 31, 01	1429 8, 72	1535 331, 07	1423 93, 07	1535 325, 27	1433 39, 95	1534 44, 67	1529 0, 99	1428 83, 16	1538 88, 07	1535 349, 59	1535 324, 32	1637 5, 47	1429 7, 24	1431 12, 57	1633 93, 03	1633 43, 16	1534 348, 49	1433 445, 65	1433 46, 02	1532 73, 73	1535 354, 51	1433 144, 29
5	191	185	186	181	186	186	183	188	187	189	186	186	184	185	184	189	181	186	184	184	183	194	191	188	187	190	190	183	183	186	186

	548,5	120,8	092,5	031,8	522,5	851,2	068,8	017,3	943,4	079,2	910,4	039,3	834,8	141	755,7	006,2	533,5	624,5	339,7	128	461,6	384,3	015	580,5	183,4	877,4	978,5	893	502,4	810,9	312,1
6	731922,43	780088,21	758988,7	751266,26	755749,92	761422,78	796005,32	763406,81	744499,92	781085,4	742154,97	768704,22	768846,02	781311,46	797433,23	728544,21	755198,31	782066,71	798871,57	791211,24	765889,91	746855,26	755969,15	750188,5	720069,3	747344,11	755886,81	698891,9	711911,37	730500,82	751185,8
7	701088,9	171802,4	171533,8	169045,5	171459,9	167658,6	171522,7	170363,6	170746,1	172362,3	168706,2	171533,5	172182,7	170029,9	173622,3	170406,8	168049,3	168003,3	167006	168815,6	168822,4	171456,1	167900,4	169567,3	168609,3	169999	169577,3	168950,5	168356	169668,9	171864,7
8	1805	1728	1761	1840	1778	1789	1755	1776	1756	1806	1750	1732	1787	1739	1744	1755	1785	1753	1770	1801	1753	1725	1759	1800	1770	1799	1766	1788	1776	1783	1765

	96,2	78,6	70,4	13,4	01,6	35,7	91,8	23,6	32,5	43,5	33,3	58	28,2	63,7	81,1	15,5	21,3	49,6	82,6	56	74,6	78,9	67,2	90,2	60,5	04,2	65,3	78,4	98,4	41,8	13,8
9	1560647,7	1546779,8	1534791,1	1547327,7	1552526,6	155835	1562298,8	1580743,3	155973	1530969,9	1521400	1539000,7	1570600,3	1544944,5	1549255,5	1538047	1555596,6	1553533,7	1528188,6	1549888,6	1562822,3	15632323	1553721,1	1573633,2	1545597,7	154146	155833	1572047,9	1554214,4	1578633,7	1565500,8

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN B HASIL EKSEKUSI *SELF ADAPTIVE LEARNING GREAT DELUGE*

1. SAT																																	
ins	RUN																																
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	21	13	16	12	22	7	13	6	26	15	12	16	16	9	14	21	10	21	11	25	18	18	6	12	12	12	8	19	11	12	14
1	32	30	29	31	34	29	35	32	30	27	30	37	37	28	34	29	31	30	36	32	31	26	38	34	36	27	29	37	31	32	36
2	20	30	22	29	27	25	27	23	29	27	24	33	31	22	31	30	26	27	31	22	29	27	28	27	26	25	29	25	32	25	26
3	5	7	6	8	6	5	3	8	9	4	7	6	6	5	6	4	8	7	6	6	8	8	6	5	6	5	6	9	7	5	7
4	10	16	4	11	10	14	18	37	14	10	15	11	11	22	8	14	14	3	30	12	10	14	14	18	8	13	7	31	26	9	6
5	10	9	8	10	5	10	5	13	8	5	6	6	5	8	9	8	7	7	9	10	5	13	6	7	8	9	6	12	11	8	8
6	5	8	6	6	5	6	7	6	6	6	7	5	7	5	7	6	5	5	6	5	6	5	6	5	5	7	6	5	7	6	7
7	6	5	7	6	7	5	6	5	6	5	6	6	6	6	6	5	5	6	6	6	6	5	6	6	8	6	7	7	6	7	6

8	8	8	5	5	8	8	8	9	9	9	8	10	9	9	8	5	8	9	9	8	5	8	8	6	9	7	9	5	5	8	5
9	213	211	213	209	217	214	214	216	215	219	211	215	213	213	215	211	213	217	213	217	215	215	213	213	215	213	211	221	217	213	213
10	6	17	7	7	9	12	19	8	13	13	14	13	11	8	13	11	14	7	12	15	10	10	20	15	7	14	11	9	9	11	11
11	7	10	8	7	10	8	9	9	10	7	10	10	7	9	7	10	7	9	11	9	10	11	9	10	10	8	8	10	8	9	9
2. Bin Packing																															
ins	RUN																														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	0,0117	0,0120	0,0113	0,01201	0,0123	0,0123	0,0108	0,01156	0,0120	0,0123	0,0120	0,01209	0,0119	0,0113	0,0107	0,0117	0,0114	0,0119	0,0109	0,0119	0,0113	0,0119	0,01231	0,0120	0,0117	0,0116	0,0117	0,0120	0,0119	0,01236	0,0120

	12	36	79		28	35	12		45	32	51		09	43	96	81	35	86	67	93	28	43		47	02	26	96	84	06		55
1	0,012587	0,011454	0,011871	0,012321	0,012927	0,012293	0,008307	0,012946	0,012447	0,01282	0,012041	0,012334	0,012738	0,016284	0,012679	0,012596	0,012963	0,012431	0,012591	0,011451	0,011992	0,012183	0,008393	0,012636	0,012637	0,013066	0,012862	0,01302	0,011727	0,016108	0,011912
2	0,02466	0,022803	0,024895	0,023238	0,022742	0,025563	0,025265	0,024475	0,023132	0,023355	0,024613	0,022831	0,022889	0,022171	0,026848	0,023896	0,025115	0,024786	0,022782	0,023748	0,024802	0,024753	0,024388	0,024782	0,025389	0,026421	0,024889	0,023854	0,024181	0,025575	0,025334
3	0,02460	0,024867	0,022667	0,022727	0,02250	0,026565	0,025353	0,024848	0,025454	0,024	0,024241	0,024949	0,026262	0,08633	0,024848	0,023939	0,023636	0,024646	0,026565	0,024848	0,025353	0,024040	0,026969	0,026261	0,026060	0,025757	0,023131	0,026261	0,024343	0,026	0,024

	62	63	97	12	55	81	93	85	18	91	27	21	07	01	26	57	15	91	13	13	25	82	66	13	55	47	43	43	37	03	42
4	0,006649	0,000662	0,000580	0,000678	0,000721	0,000708	0,000594	0,000627	0,000736	0,000693	0,000702	0,000645	0,000675	0,000676	0,000635	0,000611	0,000692	0,000568	0,000690	0,000614	0,000714	0,000663	0,000650	0,000705	0,000734	0,000636	0,000685	0,000647	0,000644	0,000695	0,000689
5	0,004099	0,000422	0,000840	0,000842	0,000420	0,000420	0,000410	0,000418	0,000420	0,000420	0,000414	0,000849	0,000396	0,001641	0,000842	0,000841	0,000419	0,000847	0,000422	0,000843	0,000421	0,000900	0,000416	0,000420	0,000415	0,000415	0,000416	0,000469	0,000839	0,000840	0,000410
6	0,00274	0,00271	0,00269	0,00313	0,00273	0,00260	0,00270	0,00321	0,00315	0,00270	0,00314	0,00313	0,00270	0,00314	0,00274	0,00276	0,00271	0,00314	0,00271	0,00222	0,00221	0,00271	0,00316	0,00192	0,00314	0,00263	0,00260	0,00269	0,00218	0,00271	0,00271

	79	55	24	38	38		22	01	08	08	71	51	37		82	05	06		84	46	25		12	23	03	25	95	29	32	16	23
7	0,04587	0,045601	0,045491	0,042906	0,038157	0,04519	0,045133	0,048153	0,043184	0,186159	0,047771	0,045455	0,040554	0,045884	0,040077	0,042704	0,045756	0,050613	0,043028	0,037999	0,04571	0,045811	0,048113	0,193837	0,045582	0,050611	0,048274	0,040495	0,04806	0,047836	0,047571
8	0,060533	0,05572	0,057993	0,059774	0,056731	0,06316	0,059845	0,058324	0,06136	0,061219	0,056549	0,061152	0,061104	0,061462	0,060941	0,06066	0,061045	0,059641	0,057223	0,058296	0,059902	0,060373	0,062045	0,05953	0,061352	0,063209	0,061391	0,058894	0,058147	0,058927	0,061277
9	0,02439	0,0214	0,0234	0,0221	0,0223	0,0223	0,0224	0,0246	0,0214	0,0213	0,0213	0,0191	0,0212	0,0233	0,0212	0,0234	0,0203	0,0203	0,0235	0,0223	0,0222	0,0245	0,0212	0,0223	0,0234	0,0223	0,0256	0,0243	0,0232	0,0213	0,0221

		42	73	26	78	84	48	36	46	59	88	04	23	32	38	44	05	69	56	79	76	89	92	61	18	82	45	06	59	27	75
10	0,10887575	0,1008857	0,1008838	0,1008958	0,1088664	0,1090544	0,1088448	0,1088813	0,1088667	0,1091887	0,1088666	0,1088644	0,1089822	0,1088686	0,1089334	0,1089555	0,1089445	0,1088445	0,1280446	0,1089012	0,1088777	0,1089022	0,1089081	0,1091244	0,1089053	0,1089188	0,1091688	0,1088988	0,1088872	0,1088871	0,1088686
11	0,0390226	0,0369999	0,0409161	0,0397611	0,0390688	0,0417211	0,0409005	0,0407073	0,0407222	0,0418711	0,0391103	0,0407229	0,0420344	0,0408485	0,0436565	0,0408687	0,0397555	0,0408788	0,0400933	0,0397833	0,0397445	0,0397544	0,0390887	0,0410366	0,0399911	0,0397707	0,0408688	0,0418896	0,0418117	0,0410033	0,0426671

3. Flowshop

ins	RUN																															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	6389	6369	6420	6391	6396	6415	6438	6394	6407	6450	6401	6411	6392	6400	6410	6389	6393	6368	6370	6388	6413	6356	6399	6414	6365	6422	6262	6382	6382	6417	6390
1	6353	6330	6387	6474	6311	6342	6331	6345	6341	6370	6411	6320	6337	6366	6355	6557	6347	6341	6350	6363	6321	6372	6203	6339	6344	6349	6331	6320	6312	6340	6348
2	6438	6389	6376	6430	6387	6401	6404	6413	6407	6454	6407	6543	6398	6438	6422	6415	6423	6398	6395	6411	6423	6400	6423	6430	6392	6490	6415	6408	6413	6413	6414
3	6408	6390	6378	6375	6388	6398	6402	6408	6369	6401	6378	6377	6408	6408	6532	6384	6398	6371	6502	6390	6408	6379	6388	6369	6408	6395	6392	6394	6392	6388	6397

4	6450	6475	6461	6476	6482	6484	6464	6455	6474	6508	6474	6479	6458	6480	6477	6460	6461	6509	6462	6494	6473	6504	6480	6449	6559	6435	6512	6470	6484	6464	6490
5	10619	10569	10593	10584	10553	10586	10568	10588	10589	10594	10581	10612	10610	10555	10557	10588	10571	10586	10588	10551	10584	10556	10636	10582	10568	10620	10603	10558	10587	10623	10581
6	11017	10965	11017	11007	11001	10957	10987	11017	10956	10987	11017	11017	10989	10962	11017	10998	10992	11017	10965	10981	10972	11017	10981	11017	11017	11017	10999	10965	10981	11017	11045
7	26589	26657	26596	26461	26528	26562	26658	26639	26490	26676	26611	26655	26537	26574	26660	26532	26674	26560	26649	26691	26594	26562	26573	26475	26537	26523	26510	26559	26581	26554	26615
8	26974	27059	26995	27105	27103	27172	27023	27066	26994	27057	27177	27070	27033	27032	27170	26976	26992	27003	26987	27069	27133	27025	27038	26938	27038	27115	26955	27036	27015	27146	27080

9	26819	26786	26814	26804	26823	26783	26804	26974	26900	26807	26915	26860	26874	26801	26924	26883	26810	26858	26855	26844	26816	26814	26836	26868	26784	26853	26813	26751	26854	26897	26753
10	11598	11544	11576	11509	11586	11577	11565	11563	11597	11544	11566	11555	11519	11590	11544	11541	11600	11548	11564	11519	11609	11571	11551	11596	11519	11572	11540	11586	11584	11519	11581
11	26743	26934	26901	26833	26874	26884	26836	26697	26845	26947	26796	26839	26766	26843	26884	26899	26676	26870	26833	26914	26840	26834	26815	26842	26837	26827	26974	26859	26742	26761	26821

4. Personel Scheduling

ins	RUN																															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	3361	3343	3343	3342	3328	3410	3372	3367	3388	3319	3358	3375	3337	3330	3363	3351	3303	3321	3464	3348	3373	3422	3403	3387	3359	3312	3356	3349	3341	3315	3347
1	2490	2495	2535	2564	2600	2260	2320	2410	4633	2565	2243	3176	2733	2407	2150	2405	2650	2335	2160	2510	2580	2555	2683	2230	2555	2413	2448	2615	2502	2904	2325
2	441	575	440	545	3900	450	415	375	1865	565	425	1741	3065	1740	410	7200	5400	515	455	1745	410	455	435	420	3125	415	425	355	460	425	19115
3	31	23	29	31	27	28	26	36	36	27	28	26	33	36	29	27	33	25	27	26	17	27	22	28	27	30	28	31	26	23	119

4	31	28	33	29	31	26	26	36	29	30	32	28	27	32	26	24	33	30	35	39	41	38	31	39	37	27	28	30	30	30	25
5	36	32	28	28	20	22	32	26	31	32	36	31	28	30	33	41	27	31	22	26	38	33	28	36	23	30	29	29	32	25	30
6	1299	1342	1222	1635	1406	1220	1222	1121	1420	1304	1522	1414	1213	1212	1307	1330	1117	1384	1303	1395	1306	1220	1331	1126	2191	1330	1388	1436	1205	1509	1120
7	2397	2297	2237	2489	2308	2386	2579	2294	2618	2270	2434	2362	2303	2374	2402	2263	2303	2377	2369	2458	2296	2299	2435	2262	2534	2365	2470	2305	2430	2368	2324
8	3459	4329	3431	3399	3265	3488	3294	5164	3706	3301	3224	3371	3346	3545	3531	3418	3258	3385	3702	3453	3358	3275	3489	3569	3225	3563	3422	3391	3345	3181	3384
9	12086	9693	10506	9633	10127	20039	10203	67865	12054	12191	10147	15301	10330	10083	9940	10399	60702	10149	10645	12419	11466	10245	10711	32782	13139	10411	17008	40327	57724	9918	11527
10	20	16	20	18	18	20	15	18	19	16	17	30	19	20	17	22	26	24	21	18	16	21	18	17	18	20	22	20	16	14	27

	55	50	48	10	15	58	25	58	40	40	70	05	39	90	53	40	05	45	70	05	80	63	05	20	34	35	40	40	15	60	50
11	450	1695	445	375	390	360	495	360	415	415	380	355	355	530	405	450	435	360	390	390	345	340	1705	370	405	400	350	375	385	400	425
5. TSP																															
ins	RUN																														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	519233,8	486565,47	490815,43	489556,33	487555,17	486933,35	493523,03	486684,04	488585,76	486656,56	488848,64	488523,09	488805,37	487327,17	486381,11	491014,45	488791,14	486746,76	488803,57	486911,01	485433,85	490072,28	487311,04	495633,68	490213,34	489222,45	487493,52	488142,57	505144,46	490606,48	486745,35
1	1166	1188	1177	1166	1155	1111	1177	1166	1188	1111	1155	1177	1133	1188	1133	1155	1133	1166	1111	1110	1155	1166	1122	1111	1122	1166	1133	1110	1188	1111	1144

	875,7	416	410,8	769,7	532,9	881,7	219,1	421,3	811,9	546	307,5	373,4	161	748,7	699,2	365	502,1	867,4	288,2	054,6	063,5	922,3	941,5	509,2	362,6	149,1	103,5	731,2	714,3	949,7	761,6
2	6990,728	6974,469	6956,057	6976,851	6953,095	6984,948	6976,94	6979,092	6975,334	7022,587	7000,263	6952,798	6976,653	6978,944	7076,196	7080,223	6961,868	7065,444	6991,594	6987,888	6958,429	6960,809	6980,013	6961,215	6973,737	6948,711	6991,365	7042,44	7100,21	6987,559	6992,537
3	43318,47	43523,68	43797,55	42964,11	43125,38	43220,24	43013,51	43953,32	43135,46	43078,31	43011,26	42937,42	42846,44	43094,65	42904,31	42913,81	42946,16	43059,31	42877,69	42976,42	43795,83	43195,47	42785,98	43129,97	43037,61	44344,15	42664,41	42306,77	43196,59	42984,04	43013,3
4	9088	9080	9275	9129	9088	9240	9244	9117	9131	9177	9047	9344	9226	9112	9119	9093	9066	9074	9111	9091	9121	9125	9102	9210	9094	9149	9062	9115	9110	9101	9069

	,382	,379	,225	,321	,572	,65	,592	,362	,276	,443	,928	,781	,67	,279	,138	,578	,289	,468	,802	,871	,208	,687	,113	,768	,562	,423	,178	,972	,536	,582	,121
5	595571,02	590800,45	588887,8	588455,72	615669,54	610664,34	591033,35	592800,07	588823,61	591075,45	606493,37	594666	591255,69	595569,01	592669,22	592107,07	591577,78	587977,8	588559,43	592387,75	591400,31	590900,77	610188,56	592755,93	597543,03	592557,45	593633,17	587881,58	592255,59	609444,57	607266,1
6	5,71E+04	5,45E+04	6,15E+04	5,92E+04	5,67E+04	5,62E+04	5,80E+04	6,10E+04	5,76E+04	5,96E+04	5,94E+04	5,50E+04	5,84E+04	5,95E+04	5,85E+04	5,53E+04	5,67E+04	5,71E+04	5,77E+04	5,58E+04	5,54E+04	5,78E+04	5,86E+04	5,52E+04	5,67E+04	5,47E+04	5,91E+04	5,85E+04	5,98E+04	5,76E+04	5,74E+04
7	737885	690887	706061	702661	726666	712336	708336	701448	732887	702661	705996	702711	731888	709886	717955	726044	732999	702686	727886	750992	723993	698666	720888	708002	722005	702997	714999	712955	701003	713388	709968

	,63	,12	,11	,94	,37		,32	,4	,11	,84	,37	,95	,75	,35	,34	,62	,84	,73	,29	,61	,29	,74	,91	,88	,93	,83	,48	,56	,58	,43	,82
8	2,20E+07	2,14E+07	2,22E+07	2,12E+07	2,23E+07	2,13E+07	2,13E+07	2,15E+07	2,49E+07	2,12E+07	2,25E+07	2,15E+07	2,15E+07	2,13E+07	2,19E+07	2,12E+07	2,13E+07	2,15E+07	2,15E+07	2,24E+07	2,16E+07	2,11E+07	2,13E+07	2,16E+07	2,12E+07	2,19E+07	2,18E+07	2,16E+07	2,14E+07	2,13E+07	2,16E+07
9	675832,9	677903,1	678745,4	677859,7	677930,9	678001,6	678151,6	679434,8	677148,7	677203,7	683879,7	680070,6	678387	674906,8	675815	678084,8	679787,8	674646,6	675640	679651	677580,7	678494,1	674982,7	676780,8	676007,2	675898,9	674640,5	679251,5	675951,3	677601,8	675534,1

6. VRP

ins	RUN																													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	5215,335	5255,055	5177,805	5358,6	5323,86	5297,738	5472,639	5447,45	5457,858	5439,958	5263,173	5494,248	5462,442	5244,182	5226,588	5458,345	5267,271	5291,322	5203,223	5558,208	5215,692	5231,592	5296,819	5281,87	5271,507	5267,374	5478,927	5197,355	5450,044	5455,403	5257,744
1	21702,83	20687,92	20660,35	20706,27	20667,52	20736,57	21779,39	20687,42	20655,36	20672,53	20746,51	20696,94	20769,32	20770,47	20696,37	21659,67	20672,29	20733	20748,68	21671,62	20674,78	20657,48	20723,38	20665,15	20700,52	21693,07	20804,96	20718,08	21669,36	20775,84	20740,4
2	13	14	14	13	14	14	13	14	14	14	14	14	14	14	14	13	13	13	13	13	14	13	13	14	13	14	14	13	14	13	13

	626,81	423,89	441,35	367,11	686,8	437,5	640,8	446,48	470,55	478,83	459,26	429,08	415,72	452,48	418,78	585,06	393	408,1	641,51	379,02	625,04	459,26	371,26	406,1	582,97	436,7	413,72	399,89	454,21	594,69	379,19
3	54664,482	53888,224	55288,317	55988,461	547 6,4	58955,336	55933,648	54911,828	58022,239	58722,833	56811,175	56255,741	56933,213	57222,23	58133,742	60066,071	57355,614	55044,467	57773,821	56223,492	55544,547	55887,754	54900,991	54527,982	54667,021	59999,569	56227,942	56667,662	57889,339	59444,955	56211,459
4	14306,59	14429,607	14435,876	14430,845	14437,888	14431,938	14494,052	14590,006	15328,228	15323,336	14334,109	14335,114	15351,107	14496,046	14431,081	15324,447	15344,467	14325,92	14552,862	14336,52	14298,48	14326,29	14342,207	15316,111	15344,118	14351,185	14382,235	15330,51	14514,97	14338,56	14437,97
5	162	168	166	165	166	164	172	166	173	168	167	166	160	171	162	163	168	170	165	171	171	170	167	166	160	167	160	165	162	165	164

	907,8	439,1	357,9	415,7	023,9	501,1	896,1	395,8	531,7	475,9	691,4	723	548,9	319,8	601,5	450,8	118,4	011,3	952,2	747,1	530,5	620,7	482,2	548,5	608,3	471,4	714,3	893,8	893,7	107,7	218,1
6	67203,88	73917,53	74049,01	68964,55	69164,57	67423,52	73301,83	68973,5	68325,24	73250,86	73534,31	66905,17	69146,36	68461,34	70592,41	74731,82	72633,22	67321,07	67262,22	73713,11	69274,81	68411,34	69698,15	70180,14	71876,46	70724,02	72598,94	74061,97	72231,07	77812,05	71638
7	164584,3	164480,2	166501,7	165082,2	166966,4	165450,2	165386,6	165716,6	166150,4	166449,6	166748,4	165325	169871,1	166861,1	166192,6	166894,5	166491,3	165253,2	167118,7	167020,4	165238,5	163088,6	167103,9	164434	165669,8	165389,2	166658,7	165344,3	16529,91	166488,1	165231,5
8	17166	16951	16811	16800	17799	16822	17178	17000	16944	16999	17009	17116	17044	16766	17226	16688	17150	16800	17166	16866	17000	17166	17003	17411	16733	17227	16559	16888	17150	17333	17223

	95	86,2	57,8	02,3	40	14,1	86,3	95,1	00,9	87,4	41,8	54,2	53,6	96,7	49,8	88,2	29,9	58	10,9	72,3	51,1	98	13,7	45,3	34,8	21,3	07,5	86,7	64,6	41,4	22,1
9	1522087	1535933	1532234	1537988	1527794	1520636	1547046	1534347	1526352	1552769	1525588	1506236	1550339	1556567	1524927	1530666	1531491	1550422	1509200	1559493	1518226	1542407	1521303	1522457	1547122	1520057	1545726	1526684	1519979	1529753	1530317

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN C PERBANDINGAN UJI COBA NILAI MINIMUM HASIL EKSEKUSI

Minimum															
Problem	Benchmark	50 %	40 %	30 %	20 %	10 %	9 %	8 %	5 %	3 %	1 %	0,50 %	0,10 %	0,05 %	0,01 %
SAT	8	10	7	8	8	9	7	8	8	8	8	8	8	6	7
BP	3	0	2	2	4	4	4	4	5	4	4	5	3	4	3
FS	5	2	0	2	1	1	1	1	0	0	1	1	1	0	7
PS	7	2	4	6	7	6	8	9	10	10	8	7	9	9	5
TSP	9	7	7	8	8	8	8	8	8	8	8	8	8	8	8
VRP	4	3	4	3	6	7	2	4	2	4	6	4	2	2	6
JUMLAH	36	24	24	29	34	35	30	34	33	34	35	33	31	29	36

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN D PERBANDINGAN UJI COBA NILAI MEDIAN HASIL EKSEKUSI

Median															
Problem	Benchmark	50 %	40 %	30 %	20 %	10 %	9 %	8 %	5 %	3 %	1 %	0,50 %	0,10 %	0,05 %	0,01 %
SAT	9	8	8	8	9	9	8	10	8	8	8	9	9	8	8
BP	1	0	0	3	2	3	3	3	4	3	2	3	3	3	3
FS	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
PS	9	4	4	7	5	7	7	9	9	6	8	6	8	7	2
TSP	7	7	7	6	6	7	6	7	7	6	7	6	6	6	7
VRP	5	5	5	5	6	8	5	7	1	4	7	5	4	4	4
JUMLAH	32	25	25	30	29	35	30	37	30	28	33	30	31	29	31

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN E HASIL UJI COBA *SIMPLE RANDOM SIMULATED ANNEALING*

Problem	instance	Q1	Med	Q3	Average	Max	Min
SAT	3	26	29	32	28,64516	39	19
	4	38	40	42,5	39,74194	48	29
	5	54	57	59	55,90323	66	35
	10	26	29	32	28,83871	40	21
	11	12	14	14,5	13,29032	19	10
BP	1	0,00789	0,008143	0,01135	0,00903	0,012353	0,007293
	7	0,032729	0,035139	0,037513	0,034733	0,040377	0,029717
	9	0,015729	0,016872	0,018088	0,016944	0,020171	0,013621
	10	0,109229	0,109267	0,109307	0,109288	0,109603	0,10914

Problem	instance	Q1	Med	Q3	Average	Max	Min
	11	0,029974	0,031955	0,034344	0,032171	0,037856	0,027982
FS	1	6332,5	6354	6366	6351,645	6412	6313
	3	6376	6396	6405	6390,742	6415	6347
	8	27004	27039	27057	27035,94	27239	26902
	10	11524	11549	11575	11549,13	11612	11469
	11	26765	26803	26834,5	26800,77	26981	26699
PS	5	30,5	35	37	33,80645	45	22
	8	3269	3335	3427,5	3357,419	3665	3238
	9	10033	10362	11089	11044,48	24479	9766
	10	1855,5	1945	2051,5	2018,161	3020	1670
	11	432,5	490	1720	1149,548	5580	380

Problem	instance	Q1	Med	Q3	Average	Max	Min
TSP	0	50702,71	51271,54	51995,81	51343,43	53776,29	49378,85
	2	7056,021	7081,726	7116,314	7093,647	7288,849	6982,597
	6	56404,71	57560,79	59924,23	57960,07	62042,65	55135,68
	7	70547,4	71170,41	73367,53	72824,41	80040,27	69289,35
	8	21330057	21566445	21752469	21956172	25086687	21178322
VRP	1	21658,49	21672,03	21682,71	21613,27	22726,07	20660,1
	2	14372,18	14426,11	14462,94	14359,33	15474,93	13369,48
	5	184547,7	186522,5	188298,9	186631,9	194384,3	181031,8
	6	74567,59	75886,81	77439,22	75740,99	79887,57	69891,9
	9	154546,1	155372,1	156256,1	155337,3	158074,3	152140

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN F HASIL UJI COBA *SELF ADAPTIVE LEARNING GREAT DELUGE*

Problem	instance	Q1	Med	Q3	Average	Max	Min
SAT	3	5	6	7	5,935484	9	2
	4	8	9	13,5	11,58065	30	4
	5	7	8	10	8,83871	20	3
	10	8	9	12,5	10,41935	16	5
	11	8	9	9	8,709677	11	7
BP	1	0,011972	0,012499	0,012706	0,012126	0,012936	0,008307
	7	0,042482	0,043092	0,045456	0,053021	0,193837	0,037932
	9	0,020272	0,022124	0,022435	0,021522	0,024606	0,018217
	10	0,108742	0,10876	0,108804	0,109408	0,128046	0,10872
	11	0,037787	0,038005	0,039884	0,038518	0,040904	0,034971

Problem	instance	Q1	Med	Q3	Average	Max	Min
FS	1	6330,5	6342	6356	6348,355	6474	6311
	3	6381,5	6392	6405	6399,194	6532	6369
	8	26999	27038	27091,5	27050,84	27177	26938
	10	11544	11565	11585	11562,35	11609	11509
	11	26818	26839	26879	26837,61	26974	26676
PS	5	27,5	30	32	29,83871	41	20
	8	3302,5	3362	3505,5	3533,258	7738	3185
	9	10190,5	10576	19670,5	16118,29	74534	9626
	10	1761,5	1939	2126,5	1985,903	3005	1460
	11	377,5	395	427,5	602,7419	1785	310

Problem	instance	Q1	Med	Q3	Average	Max	Min
TSP	0	48682,89	48805,37	49014,31	48997,17	51923,8	48523,09
	2	6954,885	6972,085	6992,279	6984,171	7116,431	6906,831
	6	56451,24	57642,37	58874,79	57606,24	61503,91	54488,53
	7	70284,89	71236	72498,95	71483,36	75092,61	69087,12
	8	21327788	21477414	21832246	21689000	24945109	21130422
VRP	1	20694,05	20719,33	20855,74	20916,02	21748,23	20660,1
	2	13521,12	14413,72	14443,92	14023,39	14686,8	13367,11
	5	156227,1	158067,4	159990,7	158014,7	165161,8	151071,6
	6	67330,9	69325,76	71136,08	69116,76	73785,72	63945,88
	9	151679,3	152188,7	153089,5	152555,5	157127,3	149842,9

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN G PERBANDINGAN SADGED DAN SRSA

Prob	Ins	Metode	Min	Q1	Med	Q3	Max	Skor	SRSA	SADGED
SAT	3	SADGED	2	5	6	7	9	5		
		SRSA	19	26	29	32	39	0		
	4	SADGED	4	8	9	13,5	30	5		
		SRSA	29	38	40	42,5	48	0		
	5	SADGED	3	7	8	10	20	5		
		SRSA	35	54	57	59	66	0		
	10	SADGED	5	8	9	12,5	16	5		
		SRSA	21	26	29	32	40	0		
	11	SADGED	7	8	9	9	11	5		
		SRSA	10	12	14	14,5	19	0	0	25
BP	1	SADGED	0,008	0,012	0,012	0,013	0,013	0		
		SRSA	0,007	0,008	0,008	0,011	0,012	5		
	7	SADGED	0,038	0,042	0,043	0,045	0,194	0	21	4

		SRSA	0,030	0,033	0,035	0,038	0,040	5		
		SADGED	0,018	0,020	0,022	0,022	0,025	0		
	9	SRSA	0,014	0,016	0,017	0,018	0,020	5		
		SADGED	0,109	0,109	0,109	0,109	0,128	4		
	10	SRSA	0,109	0,109	0,109	0,109	0,110	1		
		SADGED	0,035	0,038	0,038	0,040	0,041	0		
	11	SRSA	0,028	0,030	0,032	0,034	0,038	5		
FS		SADGED	6311	6330,5	6342	6356	6474	4		
	1	SRSA	6313	6332,5	6354	6366	6412	1		
		SADGED	6369	6381,5	6392	6405	6532	2		
	3	SRSA	6347	6376	6396	6405	6415	3		
		SADGED	26938	26999	27038	27091,5	27177	3		
	8	SRSA	26902	27004	27039	27057	27239	2		
		SADGED	11509	11544	11565	11585	11609	1		
	10	SRSA	11469	11524	11549	11575	11612	4	13	12

		SADGED	26676	26818	26839	26879	26974	2		
	11	SRSA	26699	26765	26803	26834,5	26981	3		
PS		SADGED	20	27,5	30	32	41	5		
	5	SRSA	22	30,5	35	37	45	0		
		SADGED	3185	3302,5	3362	3505,5	7738	1		
	8	SRSA	3238	3269	3335	3427,5	3665	4		
		SADGED	9626	10190,5	10576	19670,5	74534	1		
	9	SRSA	9766	10033	10362	11089	24479	4		
		SADGED	1460	1761,5	1939	2126,5	3005	4		
	10	SRSA	1670	1855,5	1945	2051,5	3020	1		
		SADGED	310	377,5	395	427,5	1785	5		
	11	SRSA	380	432,5	490	1720	5580	0	9	16
TSP		SADGED	48523,09	48682,89	48805,37	49014,31	51923,8	5		
	0	SRSA	49378,85	50702,71	51271,54	51995,81	53776,29	0		
		SADGED	6906,831	6954,885	6972,085	6992,279	7116,431	5		
	2	SRSA	6982,597	7056,021	7081,726	7116,314	7288,849	0	4	21

		SADGED	54488,53	56451,24	57642,37	58874,79	61503,91	3		
	6	SRSA	55135,68	56404,71	57560,79	59924,23	62042,65	2		
		SADGED	69087,12	70284,89	71236	72498,95	75092,61	4		
	7	SRSA	69289,35	70547,4	71170,41	73367,53	80040,27	1		
		SADGED	21130422	21327788	21477414	21832246	24945109	4		
	8	SRSA	21178322	21330057	21566445	21752469	25086687	1		
VRP		SADGED	20660,1	20694,05	20719,33	20855,74	21748,23	5		
	1	SRSA	20660,1	21658,49	21672,03	21682,71	22726,07	0		
		SADGED	13367,11	13521,12	14413,72	14443,92	14686,8	5		
	2	SRSA	13369,48	14372,18	14426,11	14462,94	15474,93	0		
		SADGED	151071,6	156227,1	158067,4	159990,7	165161,8	5		
	5	SRSA	181031,8	184547,7	186522,5	188298,9	194384,3	0	0	25

		SADGED	63945,88	67330,9	69325,76	71136,08	73785,72	5		
	6	SRSA	69891,9	74567,59	75886,81	77439,22	79887,57	0		
		SADGED	149842,9	151679,3	152188,7	153089,5	157127,3	5		
	9	SRSA	152140	154546,1	155372,1	156256,1	158074,3	0		
Total Skor									47	103

(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN H HASIL PENGUJIAN NILAI MEDIAN METODE SISTEM BOLA FIFA

Problem	ins	SADGED	SR-SA	% Nilai Perubahan		Total % Perubahan	Skor	
							SADGED	SRSA
SAT	3	6	29	23	383,33%	1618,06%	3	0
	4	9	40	31	344,44%			
	5	8	57	49	612,50%			
	10	9	29	20	222,22%			
	11	9	14	5	55,56%			
BP	1	0,012499	0,008143	-0,00436	-34,85%	-92,50%	0	3
	7	0,043092	0,035139	-0,00795	-18,46%			
	9	0,022124	0,016872	-0,00525	-23,74%			
	10	0,10876	0,109267	0,000507	0,47%			
	11	0,038005	0,031955	-0,00605	-15,92%			
FS	1	6342	6354	12	0,19%	-0,02%	0	3
	3	6392	6396	4	0,06%			
	8	27038	27039	1	0,00%			
	10	11565	11549	-16	-0,14%			

	11	26839	26803	-36	-0,13%			
PS	5	30	35	5	16,67%	38,20%	3	0
	8	3362	3335	-27	-0,80%			
	9	10576	10362	-214	-2,02%			
	10	1939	1945	6	0,31%			
	11	395	490	95	24,05%			
TSP	0	48805,37	51271,54	2466,163	5,05%	6,81%	3	0
	2	6972,085	7081,726	109,641	1,57%			
	6	57642,37	57560,79	-81,5782	-0,14%			
	7	71236	71170,41	-65,5889	-0,09%			
	8	21477414	21566445	89031,59	0,41%			
VRP	1	20719,33	21672,03	952,6962	4,60%	34,24%	3	0
	2	14413,72	14426,11	12,38959	0,09%			
	5	158067,4	186522,5	28455,07	18,00%			
	6	69325,76	75886,81	6561,05	9,46%			
	9	152188,7	155372,1	3183,37	2,09%			
jumlah terbaik		20	10	Total Skor Perubahan			12	6
persentase (%)		67%	33%	Persentase			67%	33%

LAMPIRAN I HASIL PENGUJIAN NILAI MINIMUM METODE SISTEM BOLA FIFA

Problem	Instance	SADGED	SRSA	% Nilai Perubahan		Total % Perubahan	Skor	
							SADGED	SRSA
SAT	3	2	19	17	850%	2904,52%	3	0
	4	4	29	25	625%			
	5	3	35	32	1066,67%			
	10	5	21	16	320,00%			
	11	7	10	3	42,86%			
BP	1	0,008307	0,007293	-0,00101	-12,21%	-78,69%	0	3
	7	0,037932	0,029717	-0,00822	-21,66%			
	9	0,018217	0,013621	-0,0046	-25,23%			
	10	0,10872	0,10914	0,00042	0,39%			
	11	0,034971	0,027982	-0,00699	-19,98%			
FS	1	6311	6313	2	0,03%	-0,71%	0	3
	3	6369	6347	-22	-0,35%			
	8	26938	26902	-36	-0,13%			

	10	11509	11469	-40	-0,35%			
	11	26676	26699	23	0,09%			
PS	5	20	22	2	10,00%	50,08%	3	0
	8	3185	3238	53	1,66%			
	9	9626	9766	140	1,45%			
	10	1460	1670	210	14,38%			
	11	310	380	70	22,58%			
TSP	0	48523,09	49378,85	855,7581	1,76%	4,57%	3	0
	2	6906,831	6982,597	75,76559	1,10%			
	6	54488,53	55135,68	647,1466	1,19%			
	7	69087,12	69289,35	202,2317	0,29%			
	8	21130422	21178322	47900,06	0,23%			
VRP	1	20660,1	20660,1	0	0,00%	30,68%	3	0
	2	13367,11	13369,48	2,375557	0,02%			
	5	151071,6	181031,8	29960,17	19,83%			
	6	63945,88	69891,9	5946,029	9,30%			
	9	149842,9	152140	2297,087	1,53%			
jumlah terbaik		23	7	Total Skor Perubahan			12	6
persentase (%)		77%	23%	Persentase			67%	33%

LAMPIRAN J PERSENTASE PENINGKATAN KUALITAS SOLUSI

Problem	**instance**	Avg Initial	Avg Best	Rata – rata Perubahan
SAT	SAT 3	322,8387	6,258065	
	SAT 4	340,4839	14,19355	
	SAT 5	455,0968	8,096774	
	SAT 10	402,8065	11,48387	
	SAT 11	147,9677	8,903226	97%
BP	BP 1	0,094974	0,012407	
	BP 7	0,190487	0,054438	
	BP 9	0,057402	0,022434	
	BP 10	0,128835	0,109543	
	BP 11	0,094625	0,040581	58%
FS	FS 1	6537,194	6348,355	
	FS 3	6603,194	6399,194	
	FS 8	27242,42	27050,84	
	FS 10	11727,35	11562,35	
	FS 11	27046,94	26837,61	2%
PS	PS 5	1403,194	29,83871	
	PS 8	53092,97	3492,613	
	PS 9	118553,6	18057,1	
	PS 10	57343,84	1985,903	
	PS 11	190751,2	482,2581	94%
TSP	TSP 0	62177,96	48921,81	
	TSP 2	8072,164	6992,226	
	TSP 6	61347,01	57606,24	
	TSP 7	78122,06	71483,36	12%

	TSP 8	24429000	21689000	
VRP	VRP 1	45204,99	20901,42	
	VRP 2	32231,9	14023,39	
	VRP 5	576571,7	166651,6	
	VRP 6	327715	70991,21	
	VRP 9	387382,2	153284,5	64%

LAMPIRAN K JUMLAH ITERASI SADGED

Problem	instance	Iterasi
SAT	0	13268
	1	7670
	2	8088
	3	26637
	4	3887
	5	16935
	6	186509
	7	178531
	8	122476
	9	69984
	10	20863
	11	85517
Bin Packing	0	84475
	1	84097
	2	494573
	3	461707
	4	58845
	5	51708
	6	105418
	7	49342
	8	17151
	9	17078
	10	47866
	11	21693
FlowShop	0	47882

	1	62756
	2	75217
	3	55884
	4	78704
	5	10021
	6	16271
	7	399
	8	424
	9	601
	10	634
	11	638
Personnel Scheduling	0	64
	1	52
	2	77
	3	73
	4	62
	5	73
	6	65
	7	59
	8	85
	9	25
	10	57
	11	78
TSP	0	226591
	1	474717
	2	207028
	3	128376
	4	151414
	5	99140

	6	98544
	7	52381
	8	1402
	9	115
VRP	0	283374
	1	949551
	2	939564
	3	452360
	4	1202282
	5	11957
	6	2770
	7	10469
	8	14993
	9	15098