TUGAS AKHIR – IS 184853

OPTIMASI PENGALOKASIAN RUANG DERMAGA MENGGUNAKAN ALGORITMA *GREAT DELUGE ITERATED LOCAL SEARCH*

BERTH ALLOCATION OPTIMIZATION USING GREAT DELUGE ITERATED LOCAL SEARCH

MOHAMMAD REFI NUR GHOZI
NRP 05211640000104

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

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

TUGAS AKHIR – IS184853

OPTIMASI PENGALOKASIAN RUANG DERMAGA MENGGUNAKAN ALGORITMA *GREAT DELUGE ITERATED LOCAL SEARCH*

MOHAMMAD REFI NUR GHOZI
NRP. 05211640000104

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

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

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS – IS184853

BERTH ALLOCATION OPTIMIZATION USING GREAT DELUGE ITERATED LOCAL SEARCH

MOHAMMAD REFI NUR GHOZI
NRP. 05211640000104

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

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

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

Optimasi Pengalokasian Ruang Dermaga Menggunakan Algoritma Great Deluge Iterated Local Search

TUGAS AKHIR

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

Oleh

Mohammad Refi Nur Ghozi
05211640000104

Surabaya, 14 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Muljahidin, ST., MT.
NIP. 197010102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

OPTIMASI PENGALOKASIAN RUANG DERMAGA MENGGUNAKAN ALGORITMA *GREAT DELUGE ITERATED LOCAL SEARCH*

TUGAS AKHIR

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

Oleh :

MOHAMMAD REFI NUR GHOZI
NRP. 05211640000104

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

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)**

Halaman ini sengaja dikosongkan

OPTIMASI PENGALOKASIAN RUANG DERMAGA MENGGUNAKAN ALGORITMA *GREAT DELUGE ITERATED LOCAL SEARCH*

Nama : Mohammad Refi Nur Ghozi
NRP : 05211640000104
Departemen : Sistem Informasi FTEIC ITS
Pembimbing : Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRAK

Jasa transportasi laut yang semakin berkembang dari tahun ke tahun menyebabkan jumlah kapal peti kemas yang beroperasi meningkat pula, sehingga produktivitas proses bisnis yang dilakukan di dermaga juga harus ditingkatkan. Salah satu proses bisnis tersebut yaitu pengalokasian dermaga. Alokasi ruang dermaga dilakukan ketika kapal laut telah sampai di pelabuhan untuk melakukan bongkar muat dalam waktu seminimal mungkin agar kapal peti kemas tetap berada pada jadwal operasionalnya. Namun, permasalahan alokasi ruang dermaga atau yang biasa disebut dengan Berth Allocation Problem memiliki tingkat kompleksitas yang cukup tinggi dan tergolong ke dalam permasalahan NP-Hard sehingga membutuhkan komputasi tingkat tinggi untuk menyelesaikan permasalahan tersebut. Untuk memperoleh solusi dari permasalahan tersebut, salah satunya dapat dilakukan dengan menggunakan metode heuristic. Oleh karena itu, dalam penelitian ini membahas optimasi permasalahan alokasi ruang dermaga pada benchmark dataset pelabuhan Antwerp.

Pencarian solusi optimal dilakukan menggunakan metode Great Deluge Iterated Local Search dengan tetap memperhatikan batasan-batasan yang telah ada. Penggunaan algoritma Great Deluge Iterated Local Search dalam pencarian solusinya mendapatkan hasil yang lebih baik dibandingkan dengan algoritma Hill Climbing dengan jumlah peningkatan rata-rata biaya solusi untuk masing masing datasetnya yaitu mulai dari 0,9% hingga 6,6%. Untuk kedepannya, penelitian ini diharapkan dapat membantu pengimplementasian optimasi pengalokasian transportasi laut secara otomatis atau sebagai acuan untuk referensi penelitian selanjutnya

Kata Kunci : Optimasi, Alokasi Dermaga, Great Deluge, Iterated Local Search, Metaheuristic

BERTH ALLOCATION OPTIMIZATION USING GREAT DELUGE ITERATED LOCAL SEARCH

Name : Mohammad Refi Nur Ghozi
NRP : 05211640000104
Department : Information System FTEIC ITS
Supervisor : Ahmad Muklason, S.Kom., M.Sc., Ph.D.

ABSTRACT

Sea transportation services are gradually growing and caused an increasement of operating container ships as well, so the productivity of business processes carried out at the docks must also be increased. One such business process is the berth allocation. The berth allocation is carried out when the ships have arrived at the port to carry out loading and unloading in the minimum time possible so that container ships remain on their operational schedule. However, the problem of dock space allocation or commonly known as Berth Allocation Problem has a high level of complexity and is classified as NP-Hard problems so it requires high level computation to solve the problem. To obtain solutions of these problems, one of them can be done using the heuristic method. This research will discusses the optimization of the berth allocation problem using benchmark dataset from Antwerp port.

The searching for optimal solutions is done by using the Great Deluge Iterated Local Search method while still regarding the constraints. The use of the Great Deluge Iterated Local Search algorithm in finding the solution gets better results compared to the Hill Climbing algorithm with increasment of the cost of the solution from 0,9 up to 6,6%. In the future, this research is expected to help implement the optimization of sea

transportation allocation automatically or as a reference for further research references.

Keywords : Optimization, Berth Allocation Problem, Great Deluge, Iterated Local Search, Heuristic

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini.

Nama	: Mohammad Refi Nur Ghozi
NRP	: 05211640000104
Tempat/Tanggal lahir	: Malang, 14 Agustus 1998
Fakultas/Departemen	: FTEIC / Departemen Sistem Informasi
Nomor Telp/Hp/email	: refi16@mhs.is.its.ac.id

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

OPTIMASI PENGALOKASIAN RUANG DERMAGA MENGGUNAKAN ALGORITMA *GREAT DELUGE ITERATED LOCAL SEARCH*

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

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

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

Surabaya, 20 Juni 2020

MOHAMMAD REFI

NRP. 05211640000104

Halaman ini sengaja dikosongkan.

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan tugas akhir yang berjudul "Optimasi Pengalokasian Ruang Dermaga Menggunakan Algoritma *Great Deluge Iterated Local Search*". Adapun tugas akhir ini dibuat dan diajukan untuk memenuhi persyaratan guna memperoleh gelar sarjana pada Fakultas Teknologi Elektro dan Informatika Cerdas di Institut Teknologi Sepuluh Nopember.

Selama penulisan tugas akhir ini, penulis banyak menerima bantuan dan dukungan sehingga dapat menyelesaikan tugas akhir ini. Oleh karena itu, dalam penulisan tugas akhir ini penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Kedua orangtua dan saudara penulis yang selalu mendoakan penulis dan memberikan dukungan
2. Bapak Ahmad Muklason, S.Kom., M.Sc., Ph.D. selaku dosen pembimbing yang telah meluangkan waktu dan senantiasa membimbing penulis selama proses pengerjaan tugas akhir ini.
3. Bapak Edwin Riksakomara, S.Kom, M.T. dan Ibu Dr. Retno Aulia Vinarti, S.Kom., M.Kom. selaku dosen penguji sidang Tugas Akhir.
4. Alda Muhammad Sulaiman selaku rekan satu topik *Berth Allocation*.
5. Seluruh teman ARTEMIS yang telah menemani penulis selama perkuliahan
6. Seluruh dosen departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember yang telah memberikan ilmu kepada penulis selama perkuliahan
7. Serta semua pihak lain yang belum penulis sebutkan diatas

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karena itu, segala bentuk kritik dan saran akan penulis terima dengan besar hati. Penulis berharap agar tugas akhir ini dapat bermanfaat bagi pihak pihak yang memerlukan.

Surabaya, 20 Juni 2020

Penulis

Mohammad Refi Nur Ghozi

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongka

BAB I
PENDAHULUAN

Bab ini menjelaskan proses identifikasi masalah yang akan dipecahkan pada penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan relevansi terhadap pengerjaan tugas akhir.

1.1. Latar Belakang

Sektor transportasi di Indonesia mengalami pertumbuhan sebesar 11,59% pada tahun 2019 dan akan mengalami peningkatan pada tiap tahunnya, baik pada sektor jalur darat, udara, dan laut [1]. Walaupun peningkatan pada sektor laut masih tergolong lebih kecil dari sektor lainnya, peningkatan tersebut masih cukup signifikan. Peranan transportasi laut yang masih belum optimal merupakan salah satu penyebab kurangnya peningkatan dalam sektor transportasi laut [2]. Untuk mengoptimalkan peranannya tersebut, penyedia jasa transportasi laut perlu untuk melakukan strategi digitalisasi dan optimasi pada seluruh proses bisnis yang berlaku [3]. Karena itu, pada tugas akhir ini akan membahas optimasi salah satu proses bisnis pada jasa transportasi laut tersebut yaitu pengalokasian dermaga dengan menggunakan *benchmark* dataset pada Pelabuhan Antwerp.

Permasalahan alokasi ruang dermaga atau *Berth Allocation Problem* (BAP) adalah permasalahan mengenai menempatkan kapal ke dermaga spesifik pada waktu tertentu sehingga didapatkan jadwal kapal berlabuh di dermaga tersebut dengan waktu dan biaya dalam memproses kapal seminimal mungkin. Permasalahan alokasi ruang dermaga termasuk sebagai permasalahan yang cukup kompleks dan membutuhkan penyelesaian dengan menggunakan komputasi tingkat tinggi sehingga permasalahan ini dapat dikatakan sebagai permasalahan NP-*Hard* [4].

Pada penelitian berupa survey yang dilakukan oleh Natasa Kovac [5], penelitian penelitian yang membahas permasalahan BAP kebanyakan menggunakan metode algoritma *evolutionary* sebanyak 55%, setelah itu metode *metaheuristic local search* sebanyak 32%. Adapun metode yang digunakan untuk memecahkan permasalahan BAP dalam penelitian tugas akhir ini yaitu metode pencarian lokal *Great Deluge Iterated Local Search*. Penggunaan algoritma *metaheuristic* kombinasi *Great Deluge* dengan *Iterated Local Search* memungkinkan untuk melakukan optimasi pada ruang pencarian lokal dengan tidak terjebak pada *local optima*.

Metode kombinasi *Great Deluge Iterated Local Search* secara spesifik pernah digunakan pada penelitian sebelumnya untuk memecahkan permasalahan *orienteering problem* [6]. Sedangkan metode *Iterated Local Search* secara umum telah digunakan pada pemecahan permasalahan alokasi ruang dermaga oleh Correcher et al [7] dan berbagai permasalahan lainnya seperti penjadwalan mata kuliah [8], Flow Shop Scheduling [9] [10], dan *hub allocation problem* [11]. Kemudian, penggunaan algoritma *Great Deluge* sendiri pernah digunakan pada penelitian yang memecahkan permasalahan *channel assignment* pada komunikasi seluler [12] dan penjadwalan mata kuliah [13].

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang yang telah dijabarkan, tugas akhir ini akan menyelesaikan rumusan masalah berikut:

1. Bagaimana penerapan algoritma *Great Deluge Iterated Local Search* dalam menyelesaikan permasalahan alokasi ruang dermaga pada benchmark dataset Pelabuhan Antwerp?
2. Bagaimana performa algoritma *Great Deluge Iterated Local Search* dalam menyelesaikan permasalahan alokasi ruang dermaga pada benchmark dataset Pelabuhan Antwerp?

1.3. Batasan Masalah

Pada penyelesaian tugas akhir ini, terdapat beberapa batasan masalah. Berikut merupakan Batasan masalah yang harus diperhatikan:

1. Dataset yang digunakan untuk menyelesaikan penelitian ini adalah *benchmark* dataset Pelabuhan Antwerp versi *Realistic* yang didapatkan dari referensi [7] dengan mengambil 1 buah *instance* pada tiap jenis *problem* sehingga didapatkan hanya 10 buah dataset saja.
2. Pada studi kasus Pelabuhan Antwerp ini, terdapat batasan (*constraint*) *blocking restriction* yang disebabkan karena bentuk dermaga yang tidak biasa, tetapi batasan tersebut tidak digunakan pada pengerjaan tugas akhir ini karena keterbatasan informasi pada data tersebut.
3. Aplikasi dibangun dan dikembangkan menggunakan bahasa pemrograman Java.

1.4. Tujuan

Berdasarkan latar belakang dan permasalahan diatas, maka tujuan tugas akhir ini adalah:

1. Menerapkan metode *Great Deluge Iterated Local Search* dalam memecahkan permasalahan alokasi ruang dermaga pada *benchmark* dataset Pelabuhan Antwerp.
2. Menganalisis performa metode *Great Deluge Iterated Local Search* dalam memecahkan permasalahan alokasi ruang dermaga pada *benchmark* dataset Pelabuhan Antwerp.

1.5. Manfaat

Dengan penulisan tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

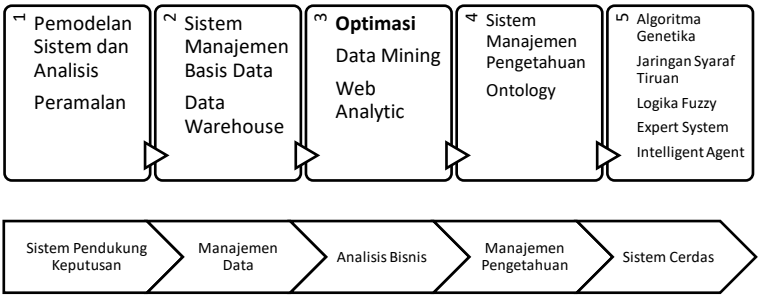
1. Bagi Peneliti
   Mampu memahami dan menerapkan metode *Great Deluge Iterated Local Search* dalam menyelesaikan permasalahan alokasi ruang dermaga.
2. Bagi Industri Pelayaran

Membantu menyelesaikan permasalahan alokasi ruang dermaga sehingga proses bisnis jasa pelayaran menjadi lebih efektif dan efisien.

3. Bagi Akademisi
   Menambah referensi dalam penggunaan metode *Great Deluge Iterated Local Search* untuk memecahkan permasalahan dengan kompleksitas NP-*Hard*, seperti permasalahan alokasi ruang dermaga yang dibahas pada tugas akhir ini.

1.6. Relevansi

Tugas Akhir ini disusun oleh penulis sebagai syarat kelulusan sarjana serta bukti dari pengimplementasian pengetahuan yang didapat selama menempuh pendidikan di Departemen Sistem Informasi ITS. Penelitian dari tugas akhir ini memiliki relevansi dengan fokus penelitian optimasi yang sesuai dengan roadmap penelitian Laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB) pada kelompok bidang analisis bisnis seperti yang tercantum pada Gambar 1.1.



Gambar 0.1 *Roadmap* Penelitian Laboratorium RDIB

Selain itu, tugas akhir ini relevan dengan mata kuliah yang pernah ditempuh oleh penulis yaitu Riset Operasi serta Optimasi Kombinatorik dan Heuristik pada kurikulum 2018 Departemen Sistem Informasi.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan studi serupa yang telah dilakukan sebelumnya sebagai acuan atau referensi penyusunan tugas akhir. Selain itu, bab ini juga menjelaskan dasar teori yang berisi informasi umum mengenai permasalahan dan metode yang digunakan dalam penelitian tugas akhir ini.

2.1. Studi Sebelumnya

Terdapat beberapa penelitian terkait dengan tugas akhir ini yang akan digunakan sebagai referensi dalam pengerjaan tugas akhir dan juga untuk menunjang penelitian pada tugas akhir. Beberapa referensi penelitian yang digunakan pada Tugas Akhir ini disajikan pada Tabel 2.1 hingga 2.3.

Tabel 2.1 Penelitian Sebelumnya 1

Penelitian Pertama	
Judul Penelitian	*The Berth Allocation Problem in Terminals with Irregular Layouts*
Penulis; Tahun	Juan Francisco Correcher, Thomas Van den Bossche, Ramon Alvarez-Valdes, Greet Vanden Berghe; 2019
Deskripsi Umum	Penelitian ini membahas mengenai pemecahan permasalahan penjadwalan kapal pada dermaga yang memiliki bentuk tidak biasa. Solusi didapatkan dengan menggunakan pendekatan metode *Mixed Integer Linear Programming* (MILP) dan heuristik *Iterated Local Search* yang berhasil menghasilkan solusi mendekati optimal dalam menjadwalkan kapal.
Keterkaitan Penelitian	Penelitian ini akan digunakan sebagai referensi utama dalam penyusunan tugas akhir ini karena tugas akhir ini mengambil topik permasalahan yang sama dengan penelitian ini.

Tabel 2.2 Penelitian Sebelumnya 2

Penelitian Kedua	
Judul Penelitian	*Advanced Traveller Information Systems*: Optimasi Rencana Perjalanan dengan Model *Orienteering Problem* dan *Great Deluge Iterative Local Search* (Studi Kasus: Trayek Angkot Surabaya)
Penulis; Tahun	Dhamar Bagas Wisesa; 2017
Deskripsi Umum	Penelitian ini membahas tentang pemecahan permasalahan optimasi rencana perjalanan transportasi umum. Setelah itu, dilakukan optimasi dengan menggunakan metode *Great Deluge Iterative Local Search.*
Keterkaitan Penelitian	Penelitian ini akan digunakan sebagai acuan mengenai metode untuk menyusun tugas akhir ini karena telah membuktikan penggunaan metode *Great Deluge Iterated Local Search* yang berhasil dalam melakukan optimasi dan mendapatkan solusi baru yang lebih baik.

Tabel 2.3 Penelitian Sebelumnya 3

Penelitian Ketiga	
Judul Penelitian	*A Reinforcement Learning: Great-Deluge Hyper-Heuristic for Examination Timetabling*
Penulis; Tahun	Ender Özcan, Mustafa Mısır, Gabriela Ochoa, Edmund K. Burke; 2010
Deskripsi Umum	Pada penelitian ini dibahas mengenai penggunaan metode *Reinforcement Learning Great Deluge Hyper-Heuristic* untuk memecahkan permasalahan penjadwalan ujian.
Keterkaitan Penelitian	Dalam penelitian tugas akhir ini akan menggunakan algoritma *Great Deluge* dalam pengaplikasian metode *Iterated Local Search*, sehingga penelitian ini dapat menjadi referensi mengenai metode tersebut.

2.2. Dasar Teori

Sub bab ini berisi teori-teori yang mendukung serta berkaitan dengan tugas akhir yang disusun.

2.2.1. Permasalahan Alokasi Ruang Dermaga (*Berth Allocation Problem*)

Ketika kapal barang telah sampai di pelabuhan, kapal tersebut dialokasikan pada tiap dermaga untuk melakukan bongkar muat dalam waktu seminimal mungkin agar kapal tetap berada pada jadwal operasionalnya. Permasalahan BAP dapat diklasifikasikan berdasarkan waktu kedatangan kapal, tata letak kapal, dan level perencanaan.

Apabila dibedakan berdasarkan waktu kedatangan kapal, permasalahan BAP secara umum terbagi menjadi statik dan dinamik. Permasalahan BAP statik berarti jika kapal sudah ada di sekitar pelabuhan dan siap untuk dialokasikan pada saat dilakukan penjadwalan. Permasalahan BAP dinamik berarti jika terdapat kapal yang belum sampai ke pelabuhan tetapi ada perkiraan waktu kedatangannya, penjadwalan juga menggunakan perkiraan waktu tersebut.

Berdasarkan tata letaknya, secara umum BAP dibedakan menjadi diskrit, kontinyu, dan hibrid. Permasalahan BAP dikatakan sebagai diskrit apabila panjang ruang dermaga sudah ditetapkan besarnya (*fixed length*) dan dikatakan sebagai kontinyu apabila kapal dapat dialokasikan dimana saja pada sepanjang dermaga selagi panjang kapal masih memadai. Kemudian, permasalahan BAP hibrid apabila merupakan kombinasi dari permasalahan diskrit dan kontinyu.

Berdasarkan level perencanaannya, permasalahan BAP dibagi menjadi operasional, taktis, dan strategis. Operational apabila dilakukan penjadwalan untuk selama satu hingga beberapa hari kedepan. Taktis apabila dilakukan penjadwalan selama satu minggu hingga beberapa bulan kedepan. Strategis apabila

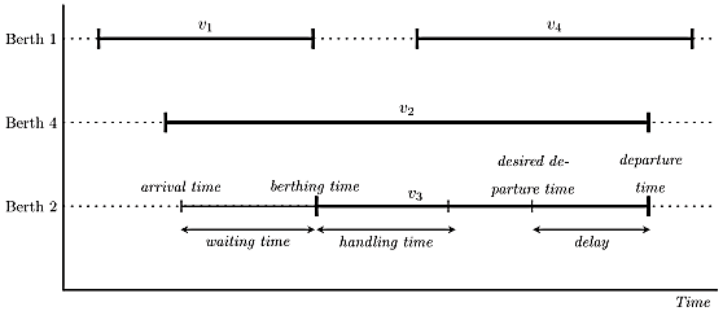
pengambilan keputusan untuk penjadwalan dalam jangka waktu tahunan [14]

Objektif dari permasalahan BAP secara general terdiri dari dua fungsi objektif, yaitu meminimalkan total waktu tunggu kapal untuk berlabuh dan meminimalkan waktu keterlambatan kapal saat akan pergi meninggalkan dermaga. Dengan waktu tunggu kapal yang minimum, hal tersebut dapat mengurangi konsumsi bahan bakar dan mengurangi emisi saat kapal sedang dalam kondisi diam [15].

Permasalahan alokasi ruang dermaga merupakan permasalahan kombinatorial yang termasuk ke dalam NP-*Hard* Problem [4].

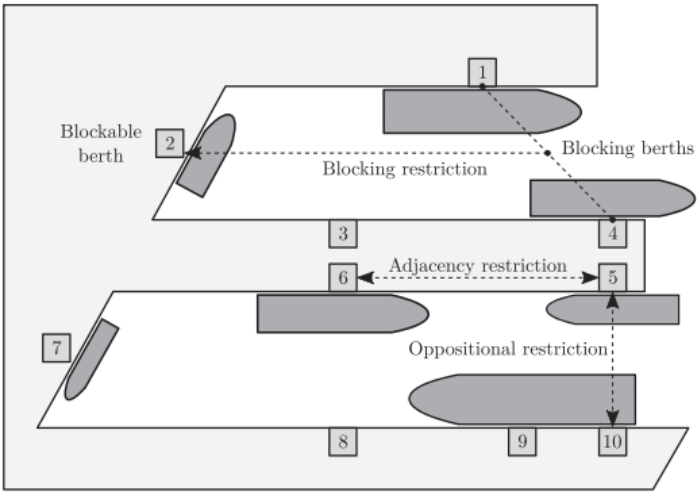
2.2.2. Dataset Pelabuhan Antwerp

Permasalahan BAP pada pelabuhan Antwerp dikategorikan sebagai permasalahan BAP dinamik dan diskrit dimana penjadwalan dilakukan untuk keseluruhan kapal baik yang sudah datang di dermaga maupun belum dengan posisi tiap dermaga yang tetap dan hanya bisa ditempati oleh satu kapal saja dalam satu waktu. Tujuan dari pemecahan permasalahan pada studi kasus dataset pelabuhan antwerp ini adalah meminimalkan total biaya pada saat pemrosesan kapal yang terdiri dari penjumlahan biaya tunggu untuk berlabuh di dermaga dan total biaya tunda saat akan meninggalkan dermaga setelah dilakukan pemrosesan kapal. Waktu pemrosesan tiap kapal pada tiap dermaga pada studi kasus ini berbeda satu sama lain. Permasalahan ini dapat digambarkan seperti pada gambar 2.1



Gambar 2.1 Skema Penjadwalan Kapal Pada Dermaga

Pelabuhan Antwerp memiliki bentuk dermaga yang tidak beraturan sehingga terdapat beberapa batasan yang diciptakan karenanya, seperti batasan dermaga yang bersebelahan dan batasan dermaga saling berhadapan. Bentuk dermaga dapat dilihat pada gambar 2.2 [7]



Gambar 2.2 Skema Pelabuhan antwerp

Adapun pada penelitian tugas akhir ini dilakukan penyederhanaan permasalahan yang mengakibatkan batasan kapal diblok oleh dermaga lain (*blockable berth*) dihilangkan,

sehingga batasan tersebut tidak digunakan pada pengerjaan tugas akhir ini.

2.2.3. Optimasi

Optimasi adalah proses pencarian kemungkinan solusi yang terbaik dari semua solusi yang tersedia. Adapun task optimasi adalah memodelkan permasalahan yang juga mengacu pada fungsi evaluasi yang merepresentasikan kualitas dari solusi, kemudian menggunakan algoritma pencarian tertentu untuk meminimalkan atau memaksimalkan fungsi objektif [16]. Optimasi dapat diaplikasikan kepada pemecahan masalah kombinatorial yang memiliki himpunan solusi layak yang terhingga. Secara prinsip, solusi dari masalah tersebut bisa didapatkan dengan enumerasi lengkap. Namun, pencarian solusi juga bisa dilakukan dengan menggunakan metode aproksimasi seperti heuristic [17].

2.2.4. *Heuristic*

Heuristic adalah metode pencarian solusi dengan kualitas yang cukup bagus dan mendekati optimal dalam waktu komputasi yang dapat diterima. Metode *Heuristic* menggunakan konsep "rule of thumb" dalam pencarian solusinya. Terdapat jenis lain dari *Heuristic*, yaitu *Metaheuristic* dan *Hyper-Heuristic*. Dalam pencarian solusi optimal pada sebuah set yang terdiri dari kelompok *initial solution*, pencarian dilakukan dengan algoritma tertentu. Tetapi dalam implementasinya, hasil pencarian dari solusi seringkali terjebak pada kondisi *local optima*. Untuk keluar dari kondisi local optima tersebut perlu memakai metode *Heuristic* yang sudah dimodifikasi yang dinamakan dengan *Metaheuristic* [16]. Metode pencarian *Metaheuristic* dapat dikatakan sebagai metodologi tingkat atas yang bisa digunakan untuk strategi perancangan *heuristic* untuk memecahkan permasalahan optimasi [18].

Sedangkan *Hyper-Heuristic* melakukan pencarian pada *search space* berupa kumpulan *Heuristic* sehingga *Hyper-Heuristic* dapat dikatakan memilih *Low-Level Heuristic* atau membuat

heuristic baru dari komponen heuristic yang sudah ada untuk kemudian dilakukan pembuatan solusi guna memecahkan permasalahan kombinatorial [18]. *Hyper-Heuristic* tidak bersinggungan langsung kepada *solution space*, tetapi bersinggungan dengan sekumpulan *low-level heuristic*. Dengan kata lain, *Hyper-Heuristic* bekerja di atas search space berupa *low-level heuristic*, sementara *Metaheuristic* bekerja diatas *search space* berupa solution space. Dengan begitu metode *Hyper-Heuristic* tidak tergantung pada *problem domain* tertentu saja seperti *Metaheuristic* [19].

2.2.5. *Great Deluge*

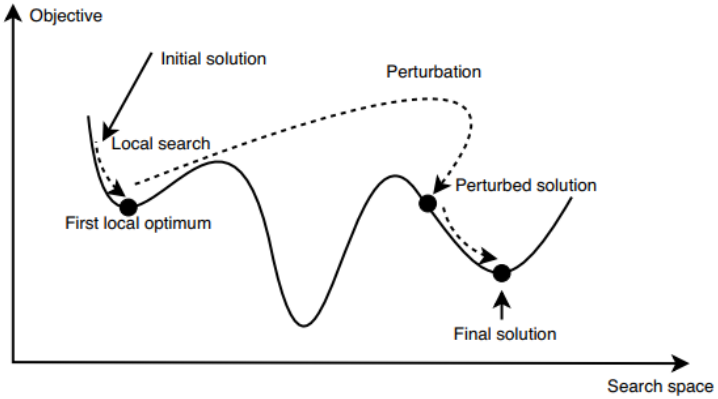
Algoritma *Great Deluge* diperkenalkan pertama kali oleh Gunter Dueck [20]. Algoritma ini berasal dari analogi pendaki bukit yang sedang melewati banjir besar (*Great Deluge*) akibat dari hujan deras dengan menjaga agar tetap melewati jalan yang tidak terkena banjir. *Global optimum* dari pencarian menggunakan algoritma ini adalah titik tertinggi dari bukit. Ketika hujan semakin deras, semakin tinggi pula ketinggian banjir tersebut. Algoritma ini berusaha agar selalu tetap di atas level air dan menyusuri area yang tidak tercakup pada *landscape* untuk menuju *global optimum*. Algoritma *Great Deluge* membutuhkan tuning hanya pada satu parameter, yaitu decay rate [21].

2.2.6. *Iterated Local Search*

Iterated Local Search merupakan metode peningkatan dari *local search*. Pada metode *local search* biasa, solusi baru dipilih secara random dan tidak berhubungan dengan local optima yang akan dihasilkan. ILS meningkatkan *local search* tersebut dengan cara menginterupsi (*perturbate*) *local optima* dan mempertimbangkan kembali *local optima* baru tersebut sebagai solusi terbaik.

Pertama-tama, *local search* pertama diterapkan pada *initial solution*. Kemudian pada setiap iterasi, dilakukan *perturbation* dan memperbarui *local optima* berdasarkan *acceptance criteria*.

Terakhir, *local search* diterapkan lagi pada solusi yang dihasilkan setelah *perturbation*. Solusi diterima sebagai solusi yang optimal apabila telah mencapai *stopping criterion*. Konsep *perturbation* digambarkan pada Gambar 2.3 [21].



Gambar 2.3 Konsep *Perturbation* pada Algoritma *Iterated Local Search*

BAB III
METODOLOGI

Bagian ini berisikan langkah penyusunan tugas akhir beserta penjelasan atas setiap tahapannya.

3.1. Tahapan Pelaksanaan Tugas Akhir

Bagian ini akan menjelaskan alur pengerjaan tugas akhir ini seperti pada Gambar 3.1.

Gambar 3.1 tahapan pengerjaan tugas akhir

3.2. Uraian Metodologi

Pada bagian ini akan dijelaskan secara lebih terperinci setiap tahapan yang dilakukan guna menyelesaikan tugas akhir.

3.2.1. Identifikasi Masalah

Pada tahap ini, dilakukan identifikasi dan pemahaman permasalahan yang akan dijadikan sebagai objek penelitian. Dalam kasus ini, permasalahan yang akan dijadikan objek

penelitian yaitu penjadwalan alokasi ruang dermaga. Setelah melakukan tahap identifikasi masalah, akan didapatkan topik permasalahan berupa rumusan masalah, batasan masalah, tujuan, dan manfaat dari penelitian ini.

3.2.2. Studi Literatur

Pada tahap ini, dilakukan pencarian studi literatur yang terkait dengan permasalahan yang telah diidentifikasi pada tahap sebelumnya dan juga mengenai metode untuk memecahkan permasalahan tersebut. Studi literatur dilakukan dengan mengumpulkan berbagai referensi dari buku, jurnal, serta paper penelitian sebelumnya. Tujuan dari tahap ini adalah untuk menambah pemahaman terkait konsep metode serta pengimplementasiannya dalam memecahkan permasalahan yang dibahas.

3.2.3. Pemahaman data

Tahap berikutnya ialah pemahaman data. Data yang dipakai merupakan *benchmark* dataset dari paper referensi penelitian pertama [7] yang membahas mengenai permasalahan alokasi dermaga pada Pelabuhan Antwerp. Dataset ini dapat diunduh pada situs https://gent.cs.kuleuven.be/bap_irregular_layouts/. Setelah itu, dilakukan pemahaman terhadap dataset tersebubt yang didalamnya tercantum informasi mengenai kapal yang akan dijadwalkan dan dermaga yang ada pada Pelabuhan Antwerp.

3.2.4. Pemodelan

Pada tahap ini dilakukan pemahaman dari model permasalahan dalam bentuk model matematis untuk mengetahui fungsi tujuan dan batasan masalah. Hal ini dilakukan untuk mendapatkan gambaran yang lebih spesifik dari permasalahan, sehingga pengerjaan tahap selanjutnya akan lebih terarah. Model matematis pada permasalahan ini tercantum pada bab sebelumnya.

3.2.5. Perancangan Algoritma

Pada tahap ini dilakukan perancangan algoritma yang digunakan untuk mendapatkan solusi awal. Kemudian dilakukan perancangan algoritma optimasi dengan menggunakan algoritma *Great Deluge Iterated Local Search* dari solusi awal tersebut.

3.2.6. Pengimplementasian Algoritma

Setelah dilakukan perancangan algoritma, algoritma tersebut diimplementasikan ke semua *instance* dataset agar diperoleh hasil solusi optimal dari masing masing *instance* dataset.

3.2.7. Analisis Hasil dan Kesimpulan

Pada tahap ini dilakukan analisis pada solusi yang dihasilkan pada tahap sebelumnya. Selain itu juga dilakukan analisis terhadap performa algoritma dalam melakukan optimasi pencarian solusi pada setiap *instance* dataset. Setelah itu dilakukan penarikan kesimpulan dari hasil analisis tersebut.

3.2.8. Penyusunan Laporan Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan tugas akhir. Setiap tahapan akan didokumentasikan serta membuat kesimpulan seitap langkah serta keputusan yang diambil dalam proses pembuatannya.

Tahapan penulisan laporan penelitian tugas akhir ini dijelaskan sebagai berikut:

1. BAB I PENDAHULUAN
   Bab ini mendokumentasikan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian serta manfaat pengerjaan tugas akhir. Tak lupa untuk mendokumentasikan relevansi penelitian dengan mata kuliah terkait.
2. BAB II TINJAUAN PUSTAKA
   Dijelaskan mengenai penelitian serupa yang telah dilakukan, serta teori-teori yang membantu penulis untuk menyusun tugas akhir.

3. BAB III METODOLOGI
   Dalam bab ini dijelaskan mengenai tahapan penyusunan tugas akhir yang harus dilakukan.
4. BAB IV PERANCANGAN
   Bab ini berisi tentang bagaimana rancangan yang akan dilakukan untuk implementasi metode yang digunakan.
5. BAB V IMPLEMENTASI
   Bab ini membahas tentang setiap langkah yang dilakukan dalam implementasi metodologi yang digunakan pada tugas akhir.
6. BAB VI HASIL DAN PEMBAHASAN
   Bab ini berisi tentang hasil pembahasan dalam penyelesaian permasalahan yang diangkat dalam tugas akhir.
7. BAB VII KESIMPULAN DAN SARAN
   Bab ini akan berisi kesimpulan dan saran sebagai pelengkap tugas akhir.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan aktivitas yang akan digunakan untuk diimplementasikan dalam tugas akhir. Bab ini berisikan Pemahaman Data, Pemodelan Matematis, Perancangan Algoritma dan Perancangan Skenario Uji Coba.

4.1. Pemahaman data

Dataset terdiri dari 10 buah *instance* yang terbagi menjadi 10 jenis yang merepresentasikan jumlah kapal yang tersedia. Seluruh *instance* beserta jumlah kapalnya dapat dilihat pada tabel 4.1. Untuk mempersingkat penyebutan nama *instance*, selanjutnya penamaan tiap *instance* akan dituliskan dengan jumlah kapalnya.

Tabel 4.1 Instance pada Dataset Pelabuhan Antwerp

Instance	Jumlah kapal
problem_10_vessels_0.txt	10
problem_20_vessels_0.txt	20
problem_30_vessels_0.txt	30
problem_40_vessels_0.txt	40
problem_50_vessels_0.txt	50
problem_60_vessels_0.txt	60
problem_70_vessels_0.txt	70
problem_80_vessels_0.txt	80
problem_90_vessels_0.txt	90
problem_100_vessels_0.txt	100

Setiap file *instance* data memiliki format yang seragam. Pada bagian *general data* menjelaskan jumlah kapal, jumlah dermaga, jarak aman kapal saling bersebelahan, dan jarak aman kapal saling berhadapan. Untuk keseluruhan dataset, jumlah dermaga, jarak aman kapal saling bersebelahan dan jarak aman

kapal saling berhadapan adalah sama, yakni masing masing 11, 10, dan 30.

Tabel 4.2 General Data

Data	Deskripsi
Jumlah dermaga	11
Jarak aman antar kapal di dermaga yang saling bersebelahan	10
Jarak aman antar kapal di dermaga yang saling berhadapan	30

Bagian selanjutnya adalah *Berths* yang menjelaskan mengenai karakteristik tiap dermaga yang bisa menampung tipe kapal tertentu dengan kapasitas maksimum *draft* kapal, kapasitas maksimum panjang kapal, kapasitas maksimum lebar kapal, dan maksimum DWT kapal yang tertera pada masing masing baris data.

Tabel 4.3 Data Berth

Data	Deskripsi
Type	Berisi tipe kapal yang dapat disandarkan pada dermaga
Max_Draft	Batas maksimum draft kapal untuk bisa bersandar di dermaga
Max_Length	Batas maksimum panjang kapal untuk bisa bersandar di dermaga
Max_Width	Batas maksimum lebar kapal untuk bisa bersandar di dermaga
Max_DWT	Batas maksimum DWT (*Deadweight Tonnage*) kapal untuk bisa bersandar di dermaga

Selanjutnya, terdapat *matrix* yang menjelaskan tentang jarak antar dermaga yang saling berhadapan. Angka 0 pada *matrix* berarti kedua berth memiliki jarak yang tidak terbatas atau

dengan kata lain tidak saling berhadapan satu sama lain. Dari *matrix* tersebut didapatkan dermaga yang saling berhadapan.

Tabel 4.4 Jarak Antar Dermaga yang Saling Berhadapan

Dermaga yang saling berhadapan	Jarak antara dermaga
1-3	92
1-4	92
2-3	92
2-4	92
5-8	129
5-9	196
5-10	196
6-8	129
6-9	196
6-10	196

Kemudian, terdapat *matrix* yang menjelaskan mengenai jarak antar dermaga yang saling bersebelahan. Sama seperti *matrix* sebelumnya, angka 0 berarti kedua berth memiliki jarak yang tidak terbatas atau tidak saling bersebelahan.

Tabel 4.5 Jarak Antar Dermaga yang saling Bersebelahan

Dermaga yang saling bersebelahan	Jarak antara dermaga
1-2	140
3-4	187
5-6	187
8-9	75
9-10	160

Bagian terakhir adalah *Ships* yang menjelaskan mengenai atribut dari tiap kapal seperti tipe kapal, draft kapal, panjang kapal, lebar kapal, dan DWT kapal. Arrival adalah waktu kedatangan kapal di dermaga. Des_depart adalah waktu pergi yang diinginkan yang bernilai sama dengan waktu kedatangan. Cost_wait dan cost_delay masing masing adalah koefisien *cost*

tunggu untuk bersandar dan *cost* tunda untuk pergi kapal. Lalu *process times in berths* berupa array yang menjelaskan waktu proses kapal pada masing masing dermaga dengan nilai 0 berarti kapal tersebut tidak cocok untuk bersandar di dermaga tersebut. *Cost wait* dan *cost delay* pada dataset ini bernilai 1 dan 0 karena objective function untuk meminimalkan waktu kapal selama berada di dermaga, sehingga hanya menghitung lama waktu di dermaga saja.

Tabel 4.6 Data Ship

Data	Deskripsi
Type	Tipe kapal untuk disandarkan pada dermaga dengan tipe yang sesuai
Draft	Besar draft kapal
Length	Panjang kapal
Width	Lebar kapal
DWT	DWT kapal
Arrival	Waktu kedatangan kapal
Des_depart	Waktu kepergian yang diinginkan kapal
Cost_wait	Biaya tunggu kapal
Cost_delay	Biaya tunda kapal
Process_times	Waktu proses kapal di dermaga.

4.2. Pemodelan Matematis

Pemodelan matematis dilakukan untuk mendapatkan gambaran mengenai permasalahan yang dijabarkan dalam bentuk simbol matematis seperti parameter, variabel keputusan, fungsi tujuan dan batasan.

4.2.1. Parameter dermaga

Berikut merupakan parameter yang mendeskripsikan atribut atribut pada dermaga.

Set dermaga: $B.N_B = |B|$
Release time berth k dengan k e B: rel_k

Set of adjacent berth:
$B^a = \{(f,k) \in BxB \mid berth\ f\ dan\ k\ saling\ berdekatan\}$
Set of opposite berth:
$B^o = \{(f,k) \in BxB \mid berth\ f\ dan\ k\ saling\ berhadapan\}$
Untuk setiap pasangan berth yang saling bersebelahan $(f,k) \in B^a$, jarak kedekatan antar dermaga adalah d_{fk}^a dan clearance antar kapal yang berlabuh di berth adalah c_{fk}^a
Untuk setiap pasangan berth yang saling bersebelahan $(f,k) \in B^o$, jarak kedekatan antar dermaga adalah d_{fk}^o dan clearance antar kapal yang berlabuh di berth adalah c_{fk}^o

4.2.2. Parameter Kapal

Di bawah ini merupakan parameter yang mendeskripsikan atribut atribut yang dimiliki oleh kapal.

Set kapal**:** $V.N_V = |V|$
Untuk tiap kapal $i \in V$,
Panjang kapal*:* l_i
Lebar kapal*:* w_i
Waktu kedatangan kapal*:* a_i
Waktu pergi kapal yang diinginkan*:* s_i
Set dermaga yang cocok*:* $B_i \subseteq B$
Waktu proses kapal di dermaga $k \in B_i$*:* h_i^k
Biaya tunggu kapal untuk bersandar di dermaga*:* C_i^w
Biaya tunda kapal saat akan meninggalkan dermaga*:* C_i^d

4.2.3. Variabel keputusan

Berikut ini merupakan variabel keputusan untuk menentukan dermaga, waktu bersandar kapal serta waktu kepergiannya

t_i = waktu bersandar kapal i
r_i = waktu kepergian kapal i
u_i = waktu penundaan kapal i ketika akan pergi

$$m_i^k = \begin{cases} 1, & apabila\ kapal\ i\ berlabuh\ di\ dermaga\ k \\ 0, & sebaliknya \end{cases}$$

$$\sigma_{ij} = \begin{cases} 1, & apabila\ r_i \leq t_j \mid i,j \in V \\ 0, & sebaliknya \end{cases}$$

4.2.4. Fungsi tujuan

Fungsi tujuan dari permasalahan ini adalah meminimalkan jumlah biaya tunggu dan biaya tunda yang diformulasikan pada persamaan 1.

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i \quad (4.1)$$

4.2.5. Batasan

Batasan 1 pada persamaan 4.2 memastikan kapal bersandar di dermaga ketika atau setelah kapal sudah sampai di dermaga.

$$t_i \geq a_i, \forall i \in V \quad (4.2)$$

Batasan 2 pada persamaan 4.3 memastikan kapal disandarkan pada dermaga.

$$\sum_{k \in B_i} m_i^k = 1, \forall i \in V \quad (4.3)$$

Batasan 3 pada persamaan 4.4 memastikan kapal baru bisa bersandar di suatu dermaga apabila dermaga tersebut tersedia dan sedang tidak ditempati oleh kapal lain.

$$t_i \geq \sum_{k \in B_i} m_i^k\, rel_k, \forall i \in V \quad (4.4)$$

Batasan 4 pada persamaan 4.5 memastikan kapal baru bisa pergi meninggalkan dermaga setelah waktu pemrosesan selesai.

$$r_i \geq t_i + \sum_{k \in B_i} m_i^k\, h_i^k, \forall i \in V \quad (4.5)$$

Batasan 5 pada persamaan 4.6 menjelaskan mengenai waktu tunda kapal yang memiliki nilai lebih dari/sama dengan waktu pergi kapal dikurangi waktu pergi yang diinginkan.

$$u_i \geq r_i - s_i, \forall i \in V \quad (4.6)$$

Batasan 6 pada persamaan 4.7 memastikan kapal bisa disandarkan di suatu dermaga setelah kapal lain telah pergi meninggalkan dermaga yang sama.

$$t_j \geq r_i - M(1 - \sigma_{ij}), \forall i, j \in V, i \neq j \quad (4.7)$$

Batasan 7 pada persamaan 4.8 menjelaskan mengenai batasan pada dermaga yang saling bersebelahan.

$$\frac{l_i}{2} + \frac{l_j}{2} + c_{fk}^a > d_{fk}^a,$$
$$i, j \in V; f \in B_i; k \in B_j; i < j; f \neq k \quad (4.8)$$

Batasan 8 pada persamaan 4.9 menjelaskan mengenai batasan pada dermaga yang saling berhadapan.

$$w_i + w_j + c_{fk}^o > d_{fk}^o,$$
$$i, j \in V; f \in B_i; k \in B_j; i < j; f \neq k \quad (4.9)$$

4.3. Perancangan Algoritma

4.3.1. Algoritma initial solution

Pencarian *initial solution* dilakukan untuk mendapatkan solusi awal yang layak dengan menggunakan algoritma pada gambar 4.1.

Input: instance data
1: M = 0
2: Sort V by increasing arrival time
3: for i ∈ V, according to the ordering do
4: Update rel_b
5: Sort B_i lexicographically by 1) increasing release time and 2) increasing $h_i^{k \in B_i}$
6: b = first element in B_i
7: $m_i^b = 1$
8: $t_i = \max\{rel_b, a_i, M\}$
9: $r_i = t_i + h_i^b$
10: M = r_i
11: $rel_b = r_i$
12: end for
Output: feasible solution

Kode 4.1 Algoritma Pembuatan Initial Solution

Algoritma diatas menghasilkan solusi layak yang mana setiap kapal diproses secara berurutan dengan waktu proses seminimal mungkin. Hal itu menyebabkan biaya yang diperlukan sangat besar, namun dapat digunakan sebagai batas atas untuk strategi *branch and bound* untuk proses pengoptimasian solusi. Selain itu, algoritma ini sekaligus dapat melakukan penghitungan waktu maksimum kepergian seluruh kapal pada tiap *instance* atau variabel M. Output dari algoritma ini berupa *arraylist* berisi keseluruhan kapal pada instance dataset yang berisi informasi mengenai waktu dan biaya pemrosesan kapal.

4.3.2. Low Level Heuristic

Low level heuristic adalah operator yang digunakan untuk mencari bentuk solusi (*neighborhood*) baru. Terdapat 3 LLH yang digunakan yaitu *shift, swap*, dan *ruin & recreate*.

4.3.2.1. *Shift*

Memindahkan satu kapal random ke dermaga lain yang sesuai dan tersedia, serta memiliki biaya terkecil. Apabila dermaga

yang terpilih terdapat kapal lain, maka kapal tersebut hanya bisa ditambatkan pada saat kapal lain tersebut selesai diproses.

4.3.2.2. *Swap*
Menukar dermaga pada masing masing dua kapal random dengan tetap memperhatikan kesesuaian kapal pada dermaga dan ketersediaan dermaga. Sama seperti shift, apabila pemindahan kapal pada dermaga yang terpilih masih ditempati oleh kapal lain, kapal tersebut akan dimasukkan setelah kapal lain tersebut selesai diproses.

4.3.2.3. *Ruin & Recreate*
Mengambil sejumlah kapal random, lalu menempatkannya kembali ke dermaga berbeda yang sesuai dan tersedia. Dengan begitu akan tercipta jadwal baru yang sedikit banyak berbeda dengan solusi sebelumnya. Seperti namanya, terdapat dua tahap pada metode ini yaitu *ruin* dan *recreate*.

Tahap *ruin* berarti mengambil sejumlah kapal dari jadwal di dermaga untuk kemudian kapal yang telah terambil tersebut ditempatkan lagi pada tahap *recreate*. Jumlah kapal yang diambil pada tahap *ruin* relatif pada besar instance, dengan jumlah 20% dari total kapal pada tiap *instance*. Hal ini dilakukan agar besar jumlah kapal yang diambil seimbang antar *instance*.

Kemudian, pada tahap *recreate*, kapal akan ditempatkan bukan pada dermaga yang memiliki waktu pemrosesan paling kecil, tetapi pada dermaga terkecil kedua. Hal ini memungkinkan untuk mendapatkan jadwal yang sedikit lebih jelek karena kapal akan berpeluang ditempatkan pada dermaga dengan waktu pemrosesan yang sedikit lebih besar, sehingga solusi yang dihasilkan lebih beragam. Algoritma tahap *ruin* tertulis pada gambar 4.2, sementara tahap recreate tertulis pada gambar 4.3.

```
Input: list ships
1:    Initiate list L
2:    ruinfactor R = 20% * numberofShips
3:    while R > 0 do
4:        Pick random number r=U[1,R]
5:        Pick random berth b
6:        Remove r ships from berth b schedule
7:        Add removed ships to list L
8:        R=R-r
9:    end while
Output: list L contained removed ships from each berth
```

Kode 4.2 Algoritma Ruin

```
Input: list of removed ships L
1:    sort L by increasing ship's arrival time
2:    for each ship in L do
3:          Initiate list feasible F
4:          for each b in berth do
5:              if feasible then
6:                  put ship at berth b to list feasible
7:          end if
8:        end for
          Sort list feasible by increasing ship's handling time
          Insert 2nd ship at list F to schedule
11:   end for
Output: list ships
```

Kode 4.3 Algoritma Recreate

4.3.3. Algoritma *Great Deluge Iterated Local Search*

Algoritma *Great Deluge Iterated Local Search* digunakan untuk melakukan optimasi *initial solution* yang telah dilakukan. Terdapat dua hal utama pada algoritma ini yaitu *local search* dan *perturbation*. *Local search* dilakukan untuk melakukan pencarian solusi baru yang optimal hingga menemukan *local optima*. Selanjutnya, *perturbation* dilakukan untuk merubah *local optima* pada *local search* yang telah dilakukan sebelumnya, memungkinkan untuk mendapatkan solusi baru

yang berbeda. Setelah itu dilanjutkan melakukan *local search* lagi pada solusi baru yang dihasilkan dari perturbation tersebut. Hal ini diulangi hingga kondisi penyelesaian tercapai.

Local search yang digunakan adalah *Great Deluge*. Algoritma *Great Deluge* memerlukan parameter *decay rate* yang harus ditentukan terlebih dahulu. Algoritma *Great Deluge* tertulis pada gambar 4.4.

```
Input: s = solution
1:    Inisiasi nilai level = cost initial solution
2:    Inisiasi nilai decay rate
3:    for i < maximum iteration, do
4:        s' = LLH(s)
5:        if cost s' < s then
6:            s=s'
7:        if cost s' < level then
8:            s=s'
9:            level = level - decayrate
10:   end for
Output: new best solution
```

Kode 4.4 Algoritma Great Deluge

Adapun algoritma *Great Deluge Iterated Local Search* yang diimplementasikan pada penelitian ini tertera pada gambar 5.3

```
Input: So = initial solution
1:    So ← Arraylist Initial Solution
2:    S* ← Great Deluge(So)
3:    while termination condition unsatisfied do
4:        S' ← Perturbation(S*)
5:        S*' ← Great Deluge(S')
6:        if S*' is better than S* then
8:            S* ← S*'
10:   end while
Output: S*
```

Kode 4.5 Algoritma *Great Deluge Iterated Local Search*

Dalam melakukan pencarian solusi baru baik pada saat *local search* atau *perturbation*, pencarian dilakukan dengan menggunakan *Low Level Heuristics*. Untuk *local search*, LLH yang digunakan adalah *shift* dan *swap*. Sedangkan untuk perturbation menggunakan *ruin* & *recreate*.

4.4. Perancangan Skenario uji coba

Pada algoritma *Great Deluge Iterated Local Search*, terdapat parameter yang harus ditentukan terlebih dahulu yaitu parameter *decay rate* yang digunakan pada tahap local search *Great Deluge*. Adapun parameter *decay rate* tersebut didapatkan dari perhitungan seperti pada persamaan 4.11 [18] yang dilambangkan dengan DR.

$$DR = \frac{biaya\ solusi\ awal - estimated\ quality}{jumlah\ iterasi} \quad (4.11)$$

Pada persamaan 2.2, terdapat *estimated quality* yang harus ditentukan terlebih dahulu. Untuk itu, pada skenario uji coba ini akan melakukan pengujian pada parameter decay rate dengan mengubah nilai alfa (α) pada perhitungan *estimated quality*. *Estimated quality* disini merupakan nilai biaya solusi yang diharapkan, sehingga dalam perhitungannya akan menggunakan persentase dari hasil solusi awal.

$$estimated\ quality = \alpha * biaya\ solusi \quad (4.12)$$

Maka dari itu, seperti pada persamaan 4.12 terdapat variabel α yang melambangkan jumlah persentase. Variabel α inilah yang akan diubah ubah nilainya pada uji coba parameter algoritma *Great Deluge Iterated Local Search* ini. Skenario uji coba nilai α dapat dilihat pada tabel 2.2

Tabel 4.7 Skenario Uji Coba Nilai α

Skenario	Nilai α
1	0.1
2	0.2
3	0.3
4	0.4
5	0.5
6	0.6
7	0.7
8	0.8
9	0.9

Algoritma *Great Deluge Iterated Local Search* yang digunakan pada penelitian ini menggunakan local search *Great Deluge* yang akan dilakukan sebanyak 200 dan 500 iterasi, sedangkan untuk *iterated local searchnya* sendiri akan dilakukan sebanyak 5000 iterasi.

Tabel 4.8 Skenario Jumlah Iterasi *Great Deluge*

No	Algoritma	Jumlah iterasi local search
1	*Great Deluge Iterated Local Search*	200
2	*Great Deluge Iterated Local Search*	500

Selanjutnya algoritma *Great Deluge Iterated Local Search* akan dijalankan sebanyak 5000 kali iterasi dengan tiap local searchnya yaitu *Great Deluge* akan dijalankan sebanyak 500 kali. Kemudian algoritma tersebut akan dibandingkan terhadap algoritma lainnya untuk kemudian dianalisis performanya. Adapun algoritma pembanding tersebut yaitu algoritma *Hill Climbing* sebanyak 2500000 iterasi seperti yang tertera pada tabel 2.2

Tabel 4.9 Skenario Jumlah Iterasi Perbandingan Algoritma

No	Algoritma	Jumlah iterasi
1	*Hill Climbing*	2500000
2	*Great Deluge Iterated Local Search*	5000x500

Tiap algoritma yang dilakukan uji coba akan dijalankan sebanyak 10 kali pada tiap instance.

BAB V
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai proses implementasi pencarian solusi terbaik dengan menggunakan algoritma *Great Deluge Iterated Local Search*. Hal ini dilakukan berdasarkan perancangan yang sudah dilakukan pada bab Perancangan.

5.1. Pembacaan dataset

Dataset terdiri dari 10 buah instance file txt dengan format yang sama. Pembacaan dilakukan sekali untuk selanjutnya disimpan pada variabel tertentu sesuai dengan tipe data yang sesuai. Bagian pertama yang dilakukan pembacaan adalah bagian general data yang berisi jumlah kapal, jumlah dermaga, safety adjacent dan safety opposite. Selanjutnya, matrix adjacent dan matrix opposite yang menjelaskan mengenai jarak kedua dermaga yang saling bersebelahan atau saling berhadapan. Dan yang terakhir adalah bagian data kapal. Data kapal disimpan kedalam bentuk objek untuk memudahkan penyimpanan data tiap kapal. Data dermaga tidak dilakukan pembacaan karena pilihan dermaga yang sesuai sudah tercantum secara signifikan pada data tiap kapal. Objek dermaga dibuat untuk membantu pemilihan dalam penempatan kapal pada dermaga yang sesuai.

Pada kode 5.1 merupakan potongan kode kelas objek Ship yang merepresentasikan kapal dan memiliki masing masing atribut dan *method* yang digunakan dalam penyelesaian permasalahan alokasi dermaga pelabuhan Antwerp. Kemudian, untuk kelas objek BerthTrans pada kode 5.2 yang merepresentasikan dermaga untuk dilakukan pemilihan dalam penempatan seluruh kapal.

```
1.    public class Ship implements Cloneable{
2.    private int shipId;
3.    private String type;
4.    private double draft;
5.    private double length;
6.    private double width;
7.    private double DWT;
8.    private double arrival;
9.    private double desDepart;
10.   private int manouvTime;
11.   private int costWait;
12.   private int costDelay;
13.   private int [] processTimes=new int[11]
14.   private int ti;
15.   private int ri;
16.   private int hi;
17.   private int ui;
18.   private int berth;
19.   public Ship(int shipId) {
20.   this.shipId = shipId;
21.   }
22.   ...
23.   }
```

Kode 5.1 Kode Kelas Objek Ship

```
1.    public class BerthTrans {
2.    private int id;
3.    private int releasetime;
4.    private int handlingtime;
5.    public BerthTrans(int id) {
6.    this.id = id;
7.    }
8.    ...
9.    }
```

Kode 5.2 Kode Kelas Objek BerthTrans

5.2. Pembuatan solusi awal

Setelah data data yang diperlukan sudah tersimpan pada masing masing variabel, pembuatan solusi awal dapat dilakukan. Solusi awal dibuat dengan mengikuti algoritma pembuatan solusi awal yang telah tercantum sebelumnya pada bab perancangan. Solusi berbentuk arraylist yang berisikan masing masing kapal yang kini telah tersimpan data mengenai waktu selama kapal disandarkan pada dermaga hingga meninggalkan dermaga. Nilai biaya solusi juga dihitung dengan menggunakan perhitungan yang sesuai dengan algoritma 4.1. Solusi awal ini akan secara mudah memenuhi seluruh batasan kapal ketika bersandar di dermaga. Adapun kode pembuatan solusi awal dapat dilihat pada kode 5.3 hingga 5.5.

```
1.    public    InitSolution(ArrayList<Ship>
      listship){
2.    this.listship=listship;
3.    ArrayList<BerthTrans> listbertha = new
      ArrayList<>();
4.    for (int i = 0; i < 11; i++) {
5.    BerthTrans bertha = new BerthTrans(i);
6.    bertha.setReleasetime(0);
7.    bertha.setHandlingtime(0);
8.    listbertha.add(bertha);
9.    }
10.   int[]       berthi       =       new
      int[listship.size()];
11.   int ti = 0;
12.   int ri = 0;
13.   int M = 0;
14.   int ui = 0;
15.   listship.sort(Comparator.comparing(Shi
      p::getArrival));
```

Kode 5.3 Kode Pembuatan Solusi Awal 1

```
1.    for (int i = 0; i < listship.size();
      i++) {
2.    Ship thisship = listship.get(i); //i
3.    berthi = thisship.getProcessTimes();
4.    for (int j = 0; j <
      listbertha.size(); j++) {
5.    listbertha.get(j).setHandlingtime(
      berthi[j]);
6.    }
7.    for (int j = 0; j <
      listbertha.size(); j++) {
8.    if(thisship.getArrival()>=
      listbertha.get(j).getReleasetime())
9.    listbertha.get(j).setReleasetime(0);
10.   }
11.   List<BerthTrans> filterberth =
      listbertha.stream().filter(p ->
      p.getHandlingtime()>
      0).collect(Collectors.toList());
12.  filterberth.sort(Comparator.comparing(
     BerthTrans::getReleasetime).thenCompari
     ng(BerthTrans::getHandlingtime));
```

Kode 5.4 Kode Pembuatan Solusi Awal 2

```
1.    BerthTrans pilih = filterberth.get(0);
2.    ti =
      Math.max((int)thisship.getArrival(),
      Math.max(pilih.getReleasetime(),M));
3.    thisship.setTi(ti);
4.    ri = ti + pilih.getHandlingtime();
5.    thisship.setHi(
      pilih.getHandlingtime());
6.    thisship.setRi(ri);
7.    thisship.setUi(ri-
      (int)thisship.getDesDepart());
8.    thisship.setBerth(pilih.getId());
9.    M = ri;
10.   this.M=M;
11.   listbertha.get(
      pilih.getId()).setReleasetime(ri);
12.   }
13.   this.initialsol = new
      ArrayList<>(listship);
14.   }
```

Kode 5.5 Kode Pembuatan Solusi Awal 3

5.3. Pengecekan Batasan

Batasan 1 menjelaskan mengenai waktu berlabuh kapal yang harus lebih besar atau sama dengan waktu kedatangan kapal. Karena itu, batasan ini akan mengecek waktu berlabuh kapal dan waktu kedatangan kapal. Batasan 2 menjelaskan mengenai kapal yang disandarkan pada dermaga. Batasan ini secara tidak langsung sudah otomatis terpenuhi dengan melihat keseluruhan kapal yang memiliki dermaga untuk disandarkan.

Batasan 3 menjelaskan mengenai kapal yang hanya bisa bersandar apabila dermaga kosong atau tidak sedang ditempati kapal lain. Oleh karena itu, batasan ini akan mengecek waktu bersandar kapal dan waktu pelepasan dermaga (*release time*). Batasan 4 menjelaskan mengenai waktu kepergian kapal yang

berlangsung setelah selesainya waktu pemrosesan kapal di dermaga. Pengecekan dilakukan dengan melihat waktu kepergian yang harus bernilai lebih besar atau sama dengan jumlah waktu bersandar kapal dan waktu pemrosesan kapal di dermaga.

Batasan 5 menjelaskan mengenai waktu tunda kapal yang memiliki nilai lebih dari atau sama dengan selisih dari waktu kepergian kapal dan waktu kepergian kapal yang diinginkan yang mana bernilai sama dengan waktu kedatangan. Batasan 6 menjelaskan mengenai urutan kapal yang berlabuh di satu dermaga yang berlangsung secara tidak bersamaan. Batasan ini melakukan pengecekan dengan melihat waktu bersandar kapal yang harus bernilai lebih besar atau sama dengan waktu kepergian kapal sebelumnya.

Batasan 7 menjelaskan mengenai kapal kapal yang bersandar di dermaga yang saling bersebelahan. Batasan ini akan melihat panjang masing masing kapal, jarak aman kedekatan kapal, dan jarak kedua dermaga. Jarak kedua dermaga didapatkan dengan melihat matrix jarak antar dermaga yang saling bersebelahan. Batasan 8 menjelaskan mengenai kapal kapal yang bersandar di dermaga yang saling berhadapan. Batasan ini akan melihat lebar masing masing kapal, jarak aman antar kapal, dan jarak kedua dermaga. Jarak kedua dermaga didapatkan dengan melihat matrix jarak antar dermaga yang saling berhadapan.

Dalam pencarian solusi layak pada tahap local search, batasan 7 dan 8 merupakan batasan yang paling sering dilanggar. Hal itu dikarenakan pada saat dilakukannya pencarian solusi baru dengan *low level heuristic* baik *shift*, *swap*, maupun *ruin* & *recreate*, solusi yang sudah layak akan diubah. Kemudian, dikarenakan batasan 7 dan 8 yang saling berhubungan dengan batasan 6 mengakibatkan apabila batasan 7 atau 8 terpenuhi akan tetapi batasan 6 tidak, maka batasan 7 atau 8 akan secara otomatis menjadi tidak terpenuhi juga.

Solusi yang baru ditemukan akan dilakukan pengecekan dan harus memenuhi semua batasan yang telah disebutkan sebelumnya. Apabila saat pengecekan batasan diperoleh hasil solusi yang tidak memenuhi salah satu atau beberapa batasan, maka solusi akan ditolak. Dengan begitu, solusi yang dihasilkan akan selalu dalam kondisi memenuhi semua batasan.

5.4. Penghitungan fungsi objektif

Seperti yang sudah tertulis pada bab sebelumnya, fungsi objektif dihitung sesuai dengan persamaan 4.1. Karena biaya tunggu pada dataset yang bernilai 0, maka penghitungan fungsi objektif hanya melakukan penjumlahan pada biaya tunda yang merupakan selisih waktu kepergian kapal dan rencana waktu kepergian kapal yang bernilai sama dengan waktu kedatangan. Dengan begitu, didapatkan biaya selama kapal berada di dermaga hingga kapal meninggalkan dermaga.

5.5. Optimasi Solusi

Optimasi menggunakan algoritma *Great Deluge Iterated Local Search* diterapkan pada solusi awal agar menemukan solusi baru yang lebih optimal.

5.5.1. Low Level heuristic

Low level heuristic digunakan pada saat melakukan *local search* dan *perturbation*. *Local search* untuk mendapatkan solusi baru yang lebih baik menggunakan *low level heuristic shift* dan *swap*. Karena ada dua low level heuristic yang dipakai, kedua LLH tersebut dirandom pada setiap kali melakukan iterasi dalam local search. Kemudian pada saat melakukan *perturbation* digunakan LLH *ruin* & *recreate*. Hal ini memungkinkan untuk melakukan diversifikasi solusi yang dibutuhkan dalam algoritma *Iterated Local Search* untuk pindah ke *local optima* yang baru.

Setelah melakukan ketiga LLH diatas dan didapatkan solusi baru, solusi tersebut terlebih dahulu dilakukan pengecekan batasan untuk mengecek kelayakan solusi. Apabila solusi

ternyata tidak layak dalam pengecekan, maka solusi tersebut akan ditolak, dan LLH dijalankan kembali hingga mendapatkan solusi yang layak.

5.5.2. Penerapan Algoritma *Great Deluge Iterated Local Search*

Algoritma *Iterated Local Search* diterapkan dengan menggunakan *Great Deluge* sebagai metode *local search*nya. Pada kode 5.6 menjelaskan mengenai deklarasi variabel yang digunakan dalam algoritma *Great Deluge Iterated Local Search*.

```
1.    public void ilsgd(String filename,
      int run, double desired){
2.    ArrayList<Ship> sbest =
      Util.cloneList(initsol);
3.    ArrayList<Ship> stemp =
      Util.cloneList(initsol);
4.    ArrayList<Ship> bestes =
      Util.cloneList(initsol);
5.    double penaltybest =
      Util.cost(bestes);
6.    double penalty1 = Util.cost(sbest);
7.    double penalty2 = 0;
8.    Random rn = new Random();
9.    int gditer = 500;
10.   double cost1 = 0;
11.   double cost2 = 0;
12.   double costils = 0;
13.   double costgd = 0;
14.   double costbest = 0;
15.   double bbest = 0;
16.   double level = Util.cost(sbest);
17.   double desiredvalue = desired*level;
18.   double decayrate =
      desiredvalue/gditer;
```

Kode 5.6 Kode Program Great Deluge Iterated Local Search 1

Selanjutnya melakukan *Local search Great Deluge* yang pertama yang kode programnya dapat dilihat pada Kode 5.7.

```
1.    //great deluge
2.    for (int j = 0; j < gditer; j++) {
3.    int numb=rn.nextInt(1);
4.    switch(numb){
5.    case(0):
6.    do {
7.    shift(stemp);
8.    } while (!Util.cekhc(stemp));
9.    break;
10.   case(1):
11.   do {
12.   swap(stemp);
13.   } while (!Util.cekhc(stemp));
14.   break;
15.   }
16.   penalty2 = Util.cost(stemp);
17.   if (penalty2<penalty1) {
18.   penalty1=penalty2;
19.   sbest = Util.cloneList(stemp);
20.   level = penalty2;
21.   }else if (penalty2<=level) {
22.   penalty1=penalty2;
23.   sbest = Util.cloneList(stemp);
24.   }else
25.   stemp = Util.cloneList(sbest);
26.   if (penalty2<penaltybest) {
27.   penaltybest=penalty2;
28.   bestes = Util.cloneList(stemp);
29.   }
30.   level=level-decayrate;
31.   }
32.   this.gdsol=Util.cloneList(bestes);
```

Kode 5.7 Kode Program Great Deluge Iterated Local Search 2

Local search dilakukan sebanyak 1000 kali. Setelah itu, hasil dari *local search Great Deluge* dilakukan *perturbation* seperti yang tertera pada kode 5.8 kemudian dilanjutkan dengan *local search Great Deluge* lagi. Setelah itu dilakukan perbandingan hasil *local search Great Deluge* dengan solusi optimal yang ditemukan sebelumnya seperti pada kode 5.9 dan hal ini dilakukan sebanyak iterasi algoritma *Iterated Local Search* yaitu sebanyak 5000 kali iterasi.

```
1.    ArrayList<Ship> perturb =
      Util.cloneList(bestes);
2.    ArrayList<Ship> bestperturb =
      Util.cloneList(bestes);
3.    costils = Util.cost(perturb);
4.    costbest = Util.cost(bestperturb);
5.    for (int i = 0; i < 5000; i++) {
6.    //perturbation
7.    do {
8.    try {
9.    ruincreate2(perturb);
10.   } catch (CloneNotSupportedException
      ex) {
11.   Logger.getLogger(
      Heuristic.class.getName()).log(Level.
      SEVERE, null, ex);
12.   }
13.   } while (!Util.cekhc(perturb));
```

Kode 5.8 Kode Program Great Deluge Iterated Local Search 3

```
1.    costgd = Util.cost(bestest);
2.    if (costgd<costils) {
3.    costils=costgd;
4.    perturb = Util.cloneList(bestest);
5.    }
6.    if (costils<costbest) {
7.    costbest=costils;
8.    bestperturb=Util.cloneList(perturb);
9.    }
10.   }
11.   this.ilssol =
      Util.cloneList(bestperturb);}
```

Kode 5.9 Kode Program Great Deluge Iterated Local Search 4

Halaman ini sengaja dikosongkan.

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan mengenai hasil dan analisis proses uji coba algoritma algoritma yang telah diimplementasikan dalam pencarian solusi pada bab sebelumnya.

6.1. Lingkungan Uji Coba

Pengimplementasian penelitian tugas akhir ini menggunakan lingkungan uji coba dengan spesifikasi perangkat keras maupun perangkat lunak seperti yang tertera pada tabel 6.1 dan 6.2.

Tabel 6.1 Spesifikasi Perangkat Keras yang Digunakan pada Penelitian

PERANGKAT KERAS	SPESIFIKASI
Jenis	Laptop
Processor	Intel Core i5-4210U CPU @ 1.70GHz (4CPUs), ~ 2.4GHz
RAM	8192 MB
Hard Disk Drive	HDD 500GB

Tabel 6.2 Spesifikasi Perangkat Lunak yang Digunakan pada Penelitian

PERANGKAT LUNAK	SPESIFIKASI
Sistem Operasi	Windows 10 Edu 64-bit
Compiler Pemrograman	Java JDK version 8u102
Aplikasi Pengembangan	Netbeans
Aplikasi Pengolahan Data	Microsoft Excel

6.2. Hasil Pembuatan Solusi Awal

Solusi awal dibuat dengan menggunakan algoritma yang telah dijelaskan pada bab perancangan. Solusi yang dihasilkan memiliki total biaya solusi yang cukup besar. Hal ini dikarenakan algoritma ini membuat seluruh kapal diproses secara FIFO untuk keseluruhan dermaga. Sehingga hanya satu kapal saja yang diproses pada satu waktu dan kapal selanjutnya baru bisa bersandar setelah kapal sebelumnya meninggalkan dermaga. Namun, algoritma ini secara mudah menghasilkan

solusi yang telah memenuhi keseluruhan batasan yang ada. Adapun rata-rata biaya solusi yang dihasilkan pada tiap *instance* dapat dilihat pada tabel 6.3.

Tabel 6.3 Hasil Solusi Awal

Instance	Biaya Solusi
problem_10_vessels_0	685
problem_20_vessels_0	2411
problem_30_vessels_0	5108
problem_40_vessels_0	8725
problem_50_vessels_0	13757
problem_60_vessels_0	19743
problem_70_vessels_0	26584
problem_80_vessels_0	34484
problem_90_vessels_0	43222
problem_100_vessels_0	52892

6.3. Hasil Penentuan Parameter Decay Rate

Penentuan parameter *decay rate* didapatkan dari uji coba nilai α *estimated quality* pada algoritma *Great Deluge Iterated Local Search*. Uji coba dilakukan sesuai dengan jumlah skenario yang tertulis pada bab sebelumnya. Adapun algoritma *Great Deluge Iterated Local Search* tersebut dijalankan sebanyak 10 kali dengan jumlah iterasi sebanyak 5000 kali dan iterasi local searchnya sebanyak 200 kali dan 500 kali. Dari uji coba tersebut dipilih *estimated quality* dengan nilai α yang menghasilkan hasil solusi terbaik pada tiap dataset. Parameter yang menghasilkan rata-rata biaya solusi terbaik seperti yang tertera pada tabel 6.4

Tabel 6.4 Hasil Uji Coba Parameter

Instance	Parameter α yang menghasikan hasil terbaik	
	200 iterasi	500 iterasi
10	0,1 – 0,9	0,1 – 0,9
20	0,1 – 0,9	0,1 – 0,9
30	0,6 – 0,9	0,1 – 0,9
40	0,1	0,4
50	0,2	0,3 & 0,7
60	0,2	0,9
70	0,4	0,6
80	0,8	0,8
90	0,3	0,4
100	0,5	0,2

Pada dataset 10 dan 20 baik pada percobaan dengan 200 iterasi maupun 500 iterasi, seluruh skenario parameter menghasilkan rata rata biaya solusi yang sama. Begitu juga pada dataset 30 dengan 500 iterasi, seluruh skenario parameter memberikan hasil yang sama. Kemudian, pada dataset 30 dengan jumlah iterasi sebanyak 200, terdapat beberapa parameter yang menghasilkan solusi terbaik dengan nilai yang sama yaitu 0,6 hingga 0,9. Selain itu, hasil solusi terbaik pada dataset 50 dengan jumlah iterasi 500 juga dihasilkan oleh beberapa parameter yaitu 0,3 dan 0,7. Untuk dataset lainnya, solusi terbaik hanya dihasilkan oleh satu buah parameter saja seperti yang tertera pada tabel 2.2. Hasil optimasi masing masing parameter pada tiap dataset dapat dilihat di Lampiran A.

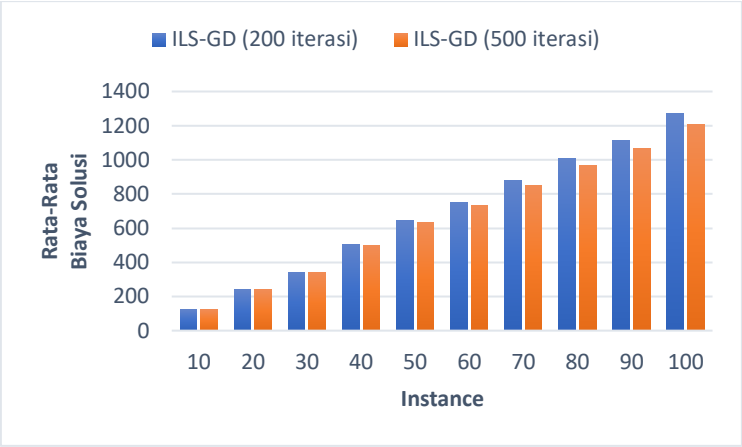
6.4. Hasil Optimasi *Great Deluge Iterated Local Search*

Setelah parameter *decay rate* ditentukan, selanjutnya melakukan skenario percobaan algoritma dengan menggunakan *local search Great Deluge* sebanyak 200 dan 500 kali iterasi.

Hal ini dilakukan untuk mengetahui pengaruh banyak iterasi terhadap nilai biaya solusi yang dihasilkan. Hasilnya, algoritma dengan menggunakan iterasi *local search* sebanyak 500 cenderung mengalami peningkatan rata-rata biaya solusi yang lebih baik dari algoritma dengan iterasi *local search* sebanyak 200 iterasi pada dataset 30 hingga 100. Hal tersebut membuktikan bahwa pada algoritma *Iterated Local Search*, jumlah iterasi pada *local search* berpengaruh terhadap biaya solusi yang dihasilkan dengan adanya proses intensifikasi yang dilakukan dalam *Iterated Local Search*. Oleh karena itu algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yang akan dipilih.

Tabel 6.5 Hasil Optimasi Algoritma *Great Deluge Iterated Local Search* dengan Iterasi *Great Deluge* Sebanyak 200 Kali dan 500 Kali

Instance	ILS-GD (200 iterasi)	ILS-GD (500 iterasi)
10	126	126
20	240	240
30	340	340.7
40	500.2	503.9
50	633.5	643
60	732	750.6
70	850.9	880.8
80	967.1	1009
90	1063.5	1115.1
100	1209.1	1273.8



Gambar 6.1 Perbandingan hasil rata-rata biaya solusi algoritma ILS-GD dengan iterasi *Great Deluge* sebanyak 200 dengan iterasi 500

6.5. Performa algoritma *Great Deluge Iterated Local Search*

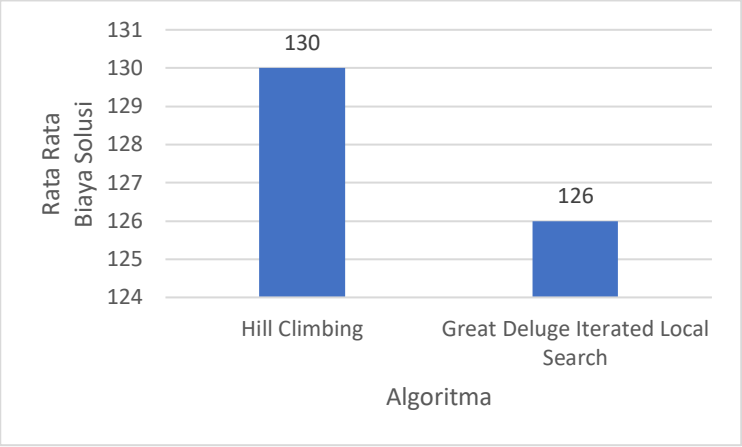
Untuk mengetahui performa dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* yang dilakukan pada penelitian ini akan dibandingkan dengan algoritma *Hill Climbing*. Kedua algoritma dijalankan masing masing sebanyak 10 kali, menghasilkan rata rata total biaya solusi yang akan dibandingkan antar algoritma. Hasilnya, solusi yang dihasilkan oleh algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali cenderung lebih baik dari algoritma *Hill Climbing*.

6.5.1. Instance 10

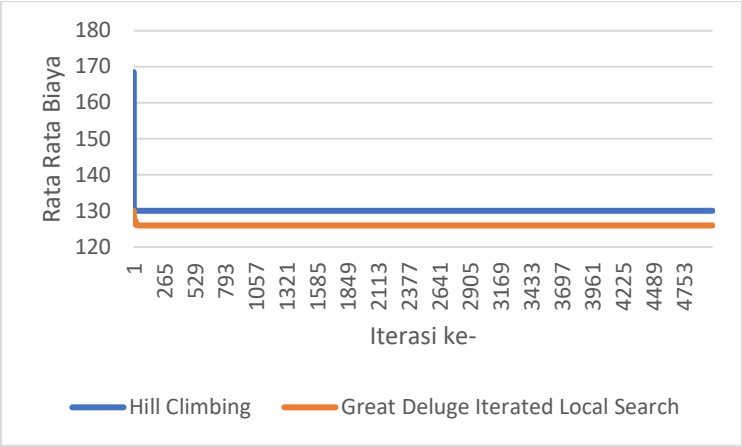
Pada dataset 10, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali sebesar 126. Hasil tersebut lebih unggul sebanyak 3,1% apabila dibandingkan dengan algoritma *Hill Climbing* yang memiliki nilai rata-rata total biaya solusi sebesar 130. Perbandingan nilai rata rata total biaya

solusi tersebut dapat dilihat pada gambar 6.2 yang disajikan dalam bentuk *bar chart*.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.3.



Gambar 6.2 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 10

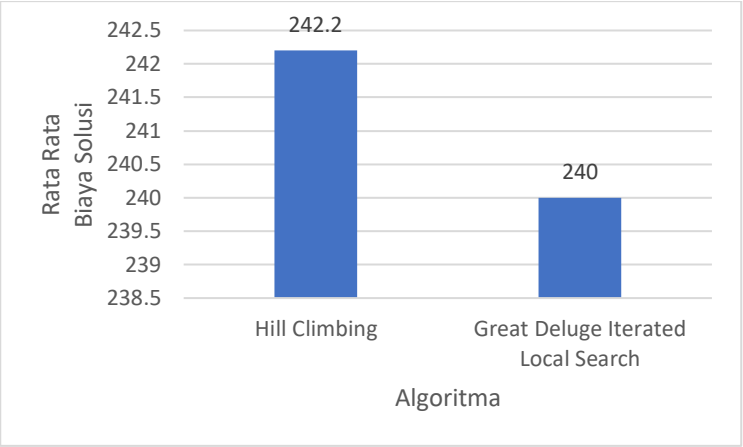


Gambar 6.3 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 10

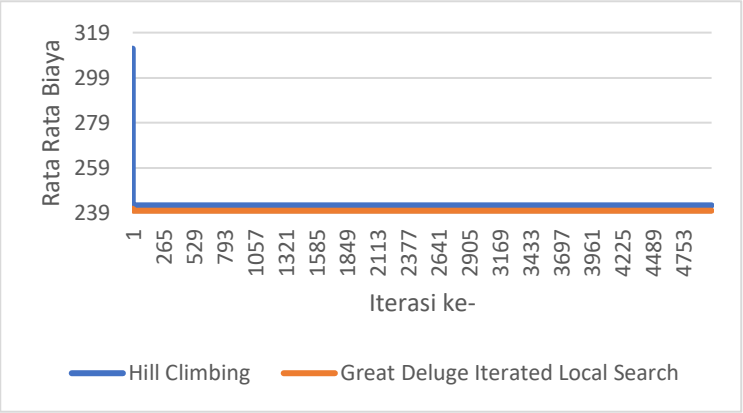
6.5.2. Instance 20

Pada instance 20, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali sebesar 240. Hasil tersebut lebih unggul sebanyak 0,9% dibandingkan dengan algoritma *Hill Climbing* yang memiliki nilai rata-rata total biaya solusi sebesar 242,2. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.4 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.5.



Gambar 6.4 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 20



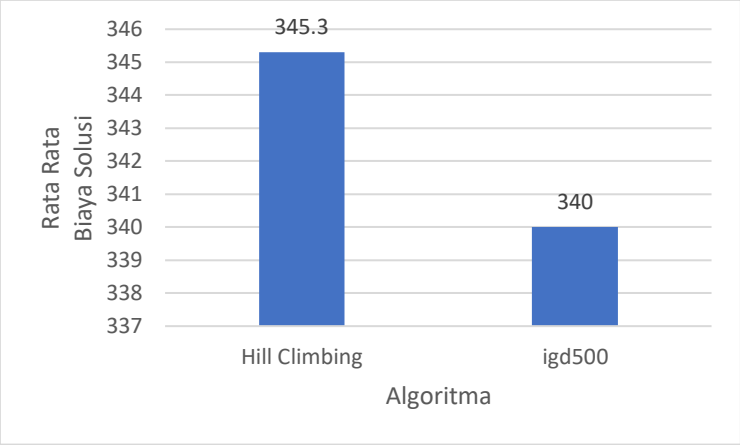
Gambar 6.5 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 20

6.5.3. Instance 30

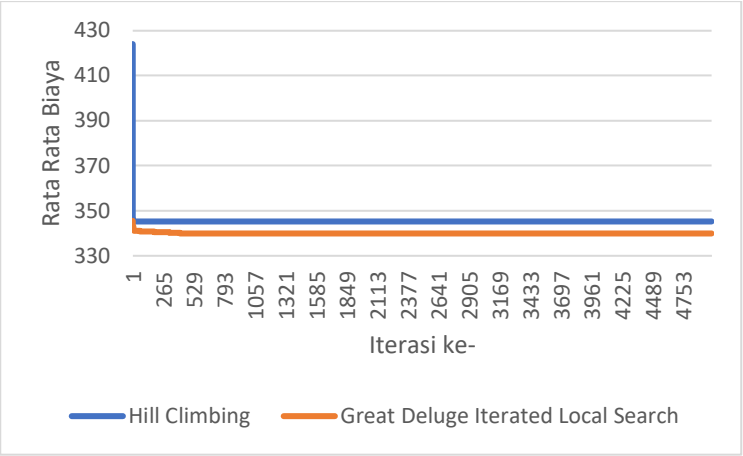
Pada instance 30, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 340. Hasil tersebut lebih unggul sebanyak 1,5% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi

sebesar 345,3. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.6 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.7.



Gambar 6.6 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 30

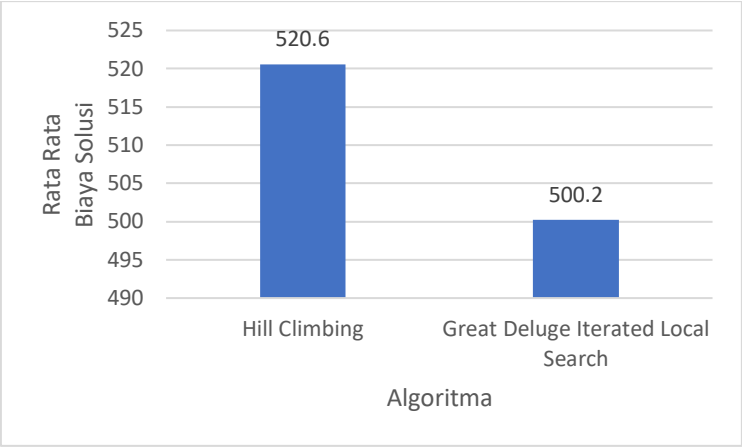


Gambar 6.7 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 30

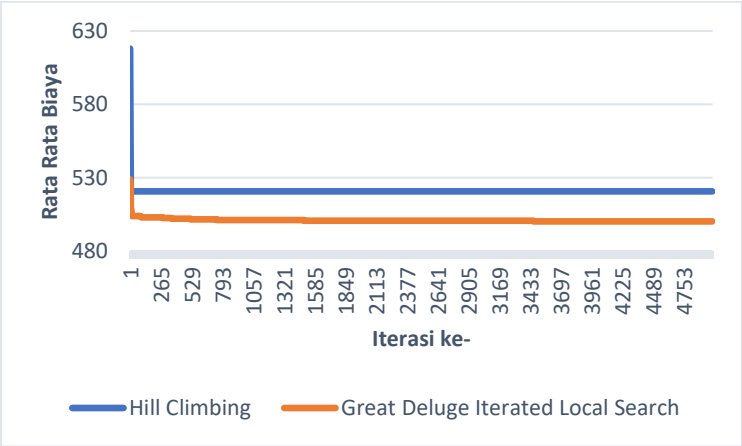
6.5.4. Instance 40

Pada instance 40, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 500,2. Hasil tersebut lebih unggul sebanyak 3,9% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 520,6. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.8 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.9.



Gambar 6.8 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 40



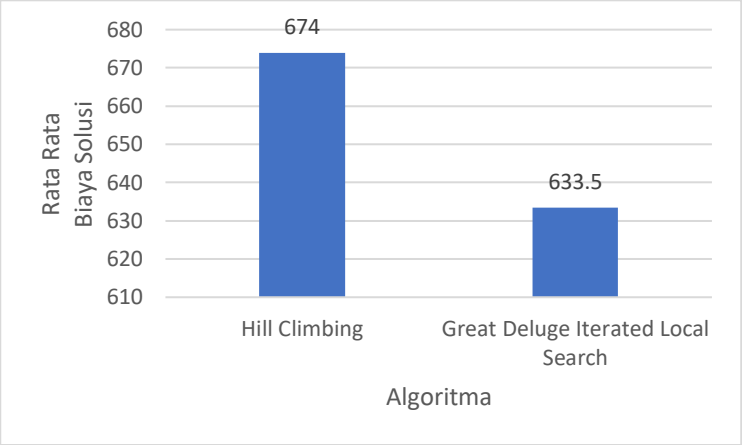
Gambar 6.9 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 40

6.5.5. Instance 50

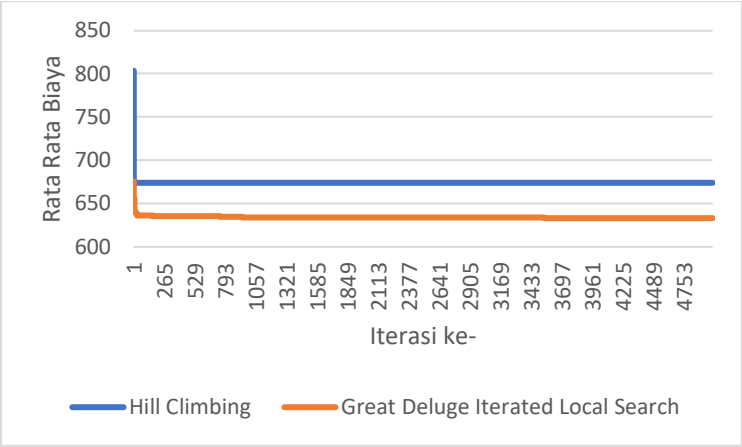
Pada instance 50, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 633,5. Hal

tersebut lebih unggul sebanyak 6% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 674. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.10 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.11.



Gambar 6.10 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 50

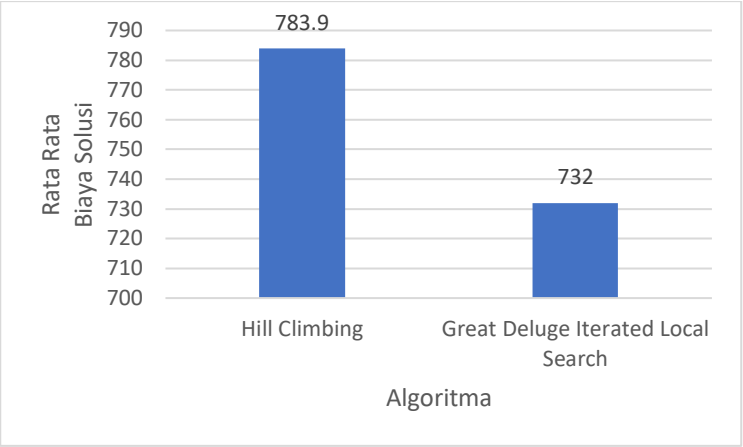


Gambar 6.11 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 50

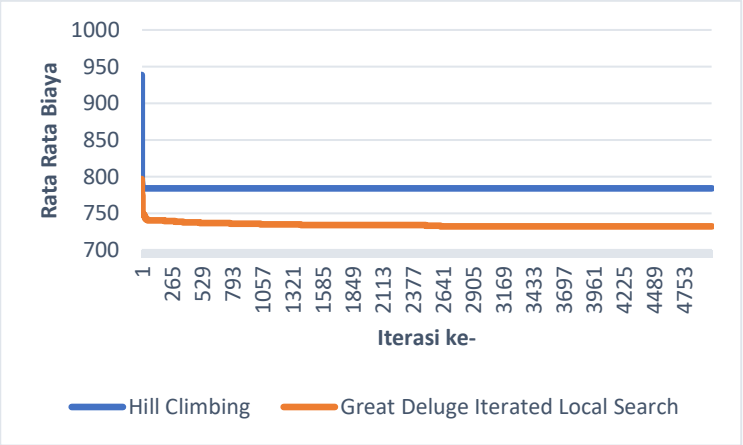
6.5.6. Instance 60

Pada instance 60, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 732. Hal tersebut lebih unggul sebanyak 6,6% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 783,9. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.12 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.13.



Gambar 6.12 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 60



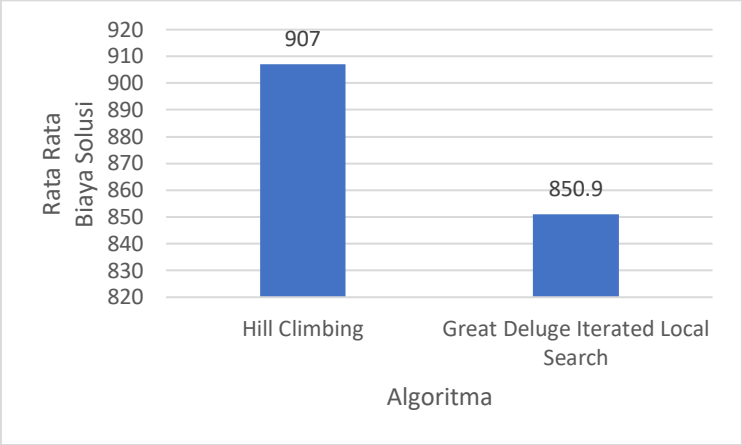
Gambar 6.13 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 60

6.5.7. Instance 70

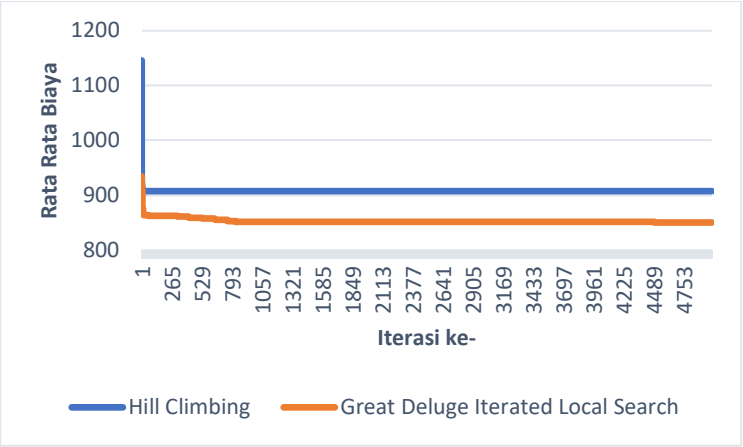
Pada instance 70, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali sebesar 850,9. Hal tersebut

lebih unggul sebanyak 6,2% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 907. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.14 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.15.



Gambar 6.14 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 70

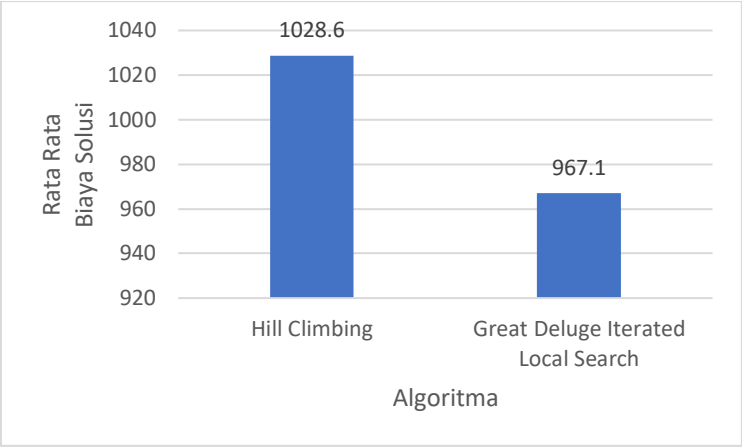


Gambar 6.15 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 70

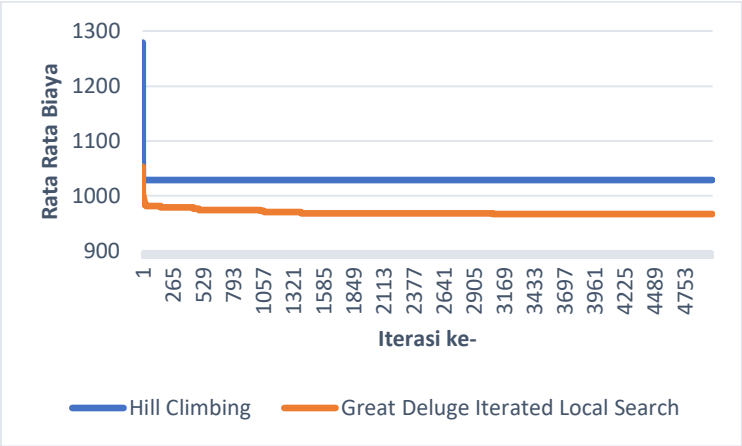
6.5.8. Instance 80

Pada instance 80, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 967,1. Hal tersebut lebih unggul sebanyak 6% dari algoritma Hill Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 1028,6. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.16 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.17.



Gambar 6.16 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 80



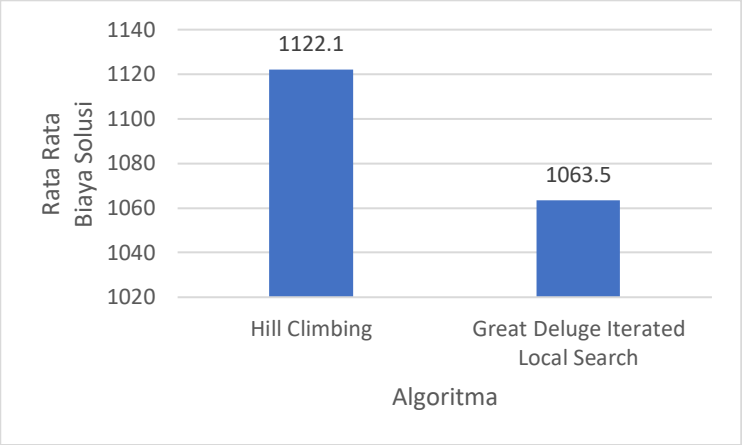
Gambar 6.17 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 80

6.5.9. Instance 90

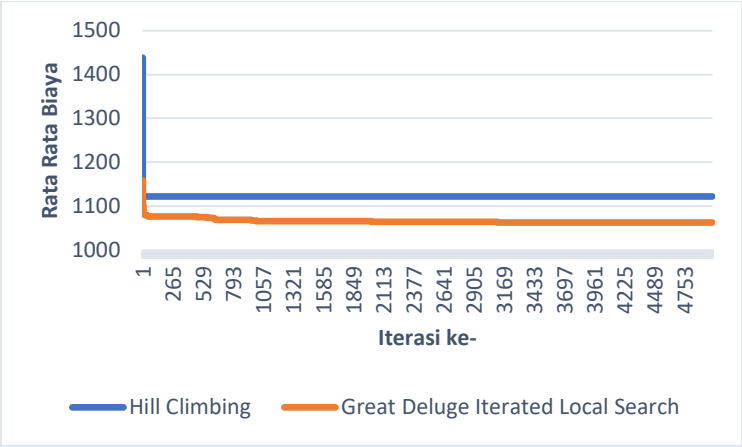
Pada instance 90, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali yaitu sebesar 1063,5. Hal tersebut lebih unggul sebanyak 5.2% dari algoritma Hill

Climbing yang menghasilkan nilai rata-rata total biaya solusi sebesar 1122,1. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.18 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.19.



Gambar 6.18 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 90

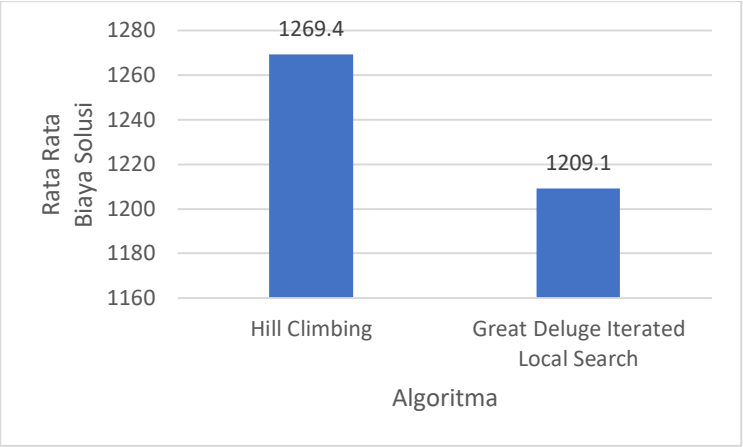


Gambar 6.19 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 90

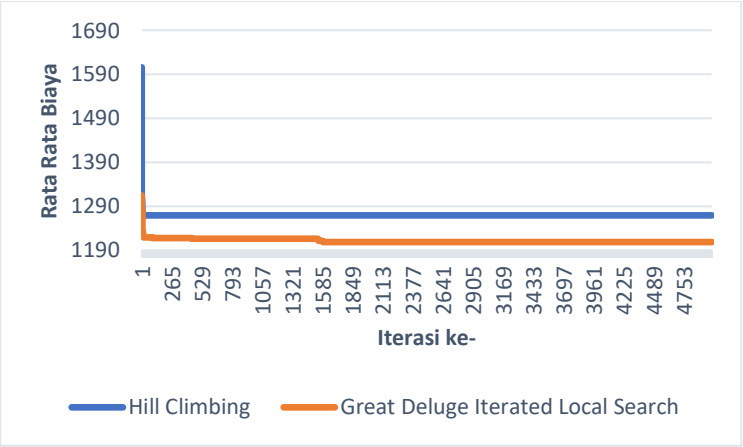
6.5.10. Instance 100

Pada instance 100, rata-rata total biaya solusi yang dihasilkan algoritma *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* sebanyak 500 kali sebesar 1209,1. Hal tersebut lebih unggul sebanyak 4,8% dari algoritma *Hill Climbing* yang menghasilkan nilai rata-rata total biaya solusi sebesar 1269,4. Perbandingan nilai rata rata total biaya solusi tersebut dapat dilihat pada gambar 6.20 yang disajikan dalam bentuk bar chart.

Dalam pencarian solusinya, algoritma *Great Deluge Iterated Local Search* menemukan solusi terbaik lebih cepat daripada algoritma *Hill Climbing* yang dapat dilihat melalui grafik *trajectory* pada gambar 6.21.



Gambar 6.20 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 100



Gambar 6.21 Grafik Perbandingan Trajectory *Great Deluge Iterated Local Search* dengan *Hill Climbing* pada Instance 100

Penggunaan algoritma pembuatan initial solution yang digunakan pada penelitian ini secara mudah dapat memenuhi seluruh batasan yang ada, namun menghasilkan ketimpangan nilai yang sangat besar dengan solusi yang dihasilkan setelah dilakukan optimasi

6.6. Ringkasan Hasil Uji Coba

Algoritma Great Deluge Iterated Local Search berhasil melakukan optimasi solusi awal dengan menghasilkan rata-rata biaya solusi yang lebih baik. Adapun algoritma dengan jumlah iterasi local search Great Deluge sebanyak 500 menghasilkan hasil yang lebih baik dari algoritma dengan jumlah iterasi local search Great Deluge sebanyak 200. Perbandingan hasil optimasi tersebut dapat dilihat pada tabel 6.6.

Tabel 6.6 Hasil Solusi Awal dan Hasil Optimasi ILS-GD Sebanyak 200 dan 500 Iterasi

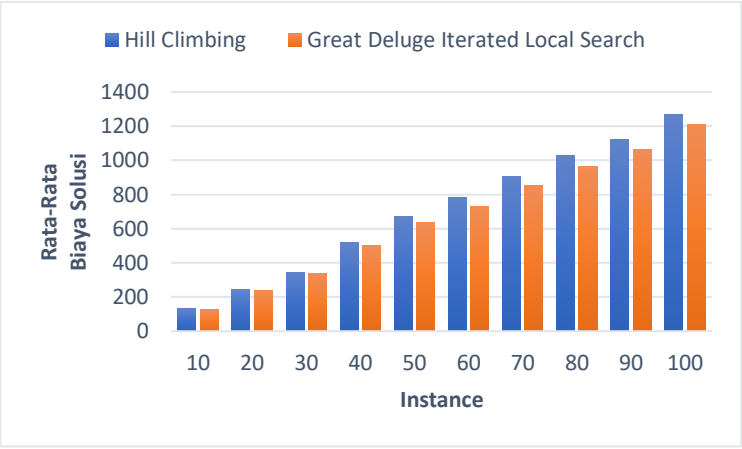
Instance	Solusi Awal	ILS-GD (200 iterasi)	ILS-GD (500 iterasi)
10	685	126	126
20	2411	240	240
30	5108	340	340.7
40	8725	500.2	503.9
50	13757	633.5	643
60	19743	732	750.6
70	26584	850.9	880.8
80	34484	967.1	1009
90	43222	1063.5	1115.1
100	52892	1209.1	1273.8

Selanjutnya untuk melihat performanya, algoritma *Great Deluge Iterated Local Search* dibandingkan dengan algoritma *Hill Climbing*. Hasilnya, rata-rata total biaya solusi yang dihasilkan algoritma Great Deluge Iterated Local Search lebih baik daripada algoritma Hill Climbing dengan jumlah peningkatan rata-rata biaya solusi seperti yang tertera pada pada tabel 6.6. Selain itu, dalam pencarian solusinya, algoritma Great Deluge Iterated Local Search juga lebih cepat dalam mendapatkan solusi optimalnya daripada algoritma Hill

Climbing seperti yang terlihat pada grafik trajektori di gambar 6.22.

Tabel 6.7 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing*

Instance	*Hill Climbing*	ILS-GD (500 iterasi)	Peningkatan
10	130	126	3.1%
20	242.2	240	0.9%
30	345.3	340	1.5%
40	520.6	500.2	3.9%
50	674	633.5	6.0%
60	783.9	732	6.6%
70	907	850.9	6.2%
80	1028.6	967.1	6.0%
90	1122.1	1063.5	5.2%
100	1269.4	1209.1	4.8%



Gambar 6.22 Perbandingan Hasil Optimasi *Great Deluge Iterated Local Search* dengan *Hill Climbing*

BAB VII
KESIMPULAN DAN SARAN

Bagian ini akan menjelaskan kesimpulan yang didapatkan dari penelitian tugas akhir ini serta saran untuk penelitian selanjutnya yang relevan dengan penelitian ini.

7.1. Kesimpulan

Berdasarkan hasil uji coba yang dihasilkan pada tugas akhir ini dapat ditarik kesimpulan sebagai berikut:

a. Algoritma *Great Deluge Iterated Local Search* dapat menyelesaikan permasalahan alokasi ruang dermaga dengan total biaya solusi yang cukup optimal pada 10 instance dataset yang digunakan pada penelitian ini.
b. Algoritma *Great Deluge Iterated Local Search* dengan iterasi *local search* sebanyak 500 menghasilkan rata-rata biaya solusi yang lebih baik dari algoritma dengan iterasi local search sebanyak 200, hal tersebut disebabkan oleh adanya intensifikasi pencarian solusi dimana semakin banyak iterasi akan dapat menemukan solusi yang lebih baik lagi.
c. Algoritma Great Deluge Iterated Local Search menghasilkan solusi yang lebih baik daripada algoritma *Hill Climbing* pada seluruh instance dengan peningkatan nilai rata-rata biaya solusi sebesar 0,9% hingga 6,6%

7.2. Saran

Kemudian, saran yang dapat penulis berikan untuk penelitian selanjutnya yaitu:

a. Menggunakan algoritma pembuatan *initial solution* yang berbeda, seperti metode *greedy* atau algoritma *branch and bound* lainnya.
b. Menggunakan metode lain dalam penentuan parameter *decay rate* seperti metode *force decay rate* [23] atau *non-linear decay rate* [24]

c. Melakukan uji coba algoritma dengan menggunakan iterasi yang lebih besar.

Halaman ini sengaja dikosongkan.

DAFTAR PUSTAKA

[1] Setijadi, “Supply chain Indonesia,” 19 March 2019. [Online]. Available: http://supplychainindonesia.com/new/sektor-transportasi-diprediksi-tumbuh-1115-pada-2019/. [Diakses 5 November 2019].

[2] R. W. Online/Ant, "Warta ekonomi," 2014 August 4. [Online]. Available: https://www.wartaekonomi.co.id/read32979/peranan-transportasi-laut-indonesia-masih-belum-optimal.html. [Accessed 5 November 2019].

[3] F. Wuryasti, "Media Indonesia," 1 July 2019. [Online]. Available: https://mediaindonesia.com/read/detail/244313-digitalisasi-pelabuhan-ciptakan-efisiensi. [Accessed 5 November 2019].

[4] A. Lim, "The berth planning problem," *Operations Research Letters*, vol. 22, no. 2, pp. 105-110, 1998.

[5] N. Kovač, "Metaheuristic approaches for the berth allocation problem," *Yugoslav Journal of Operations Research*, vol. 27, no. 3, pp. 265-289, 2017.

[6] D. B. Wisesa, "Advanced traveller information systems: optimasi rencana perjalanan dengan model orienteering problem dan great deluge iterative local search (studi kasus: trayek angkot surabaya)," Institut Teknologi Sepuluh Nopember, Surabaya, 2017.

[7] J. F. Correcher, T. V. d. Bossche, R. Alvarez-Valdes and G. V. Berghe, "The berth allocation problem in terminals

with irregular layouts," *European Journal of Operational Research,* vol. 272, no. 3, pp. 1096-1108, 2019.

[8] J. A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall and M. Kendall, "Iterated local search using an add and delete hyper-heuristic for university course timetabling," *Applied Soft Computing,* vol. 40, pp. 581-593, 2016.

[9] S. Sánchez-Herrera, J. R. Montoya-Torres and E. L. Solano-Charrisa, "Flow shop scheduling problem with position-dependent processing times," *Computers & Operations Research,* vol. 111, pp. 325-345, 2019.

[10] A. Juan, H. Lourenço, M. Mateo, R. Luo and Q. Castella, "Using iterated local search for solving the flow-shop problem: parallelization, parametrization, and randomization issues," *International Transactions in Operational Research,* vol. 21, no. 1, pp. 103-126, 2013.

[11] J. Guan, G. Lin and H.-B. Feng, "A multi-start iterated local search algorithm for the uncapacitated single allocation hub location problem," *Applied Soft Computing,* vol. 73, pp. 230-241, 2018.

[12] G. Kendall and M. Mohamad, "Channel assignment in cellular communication using a great deluge hyper-heuristic," in *12th IEEE International Conference on Networks (ICON)*, Singapore, 2014.

[13] E. K. Burke and J. P. Newall, "Enhancing timetable solutions with local search methods," in *International Conference on the Practice and Theory of Automated Timetabling*, Berlin, 2002.

[14] R. Marti, P. M. Pardalos and M. G. C. Resende, Handbook of heuristics, Cham: Springer, 2018.

[15] M. M. Golias, G. K. Saharidis, M. Boile, S. Theofanis and M. G. Ierapetritou, "The berth allocation problem: optimizing vessel arrival time," *Maritime Economics & Logistics*, vol. 11, no. 4, pp. 358-377, 2009.

[16] E. K. Burke and G. Kendall, Search methodologies: introductory tutorials in optimization and decision support techniques, New York: Springer, 2005.

[17] F. K. Rambe, Pendekatan pencarian lokal dalam optimisasi kombinatorik, Medan: University of Sumatera Utara, 2016.

[18] E.-G. Talbi, Metaheuristics from design to implementation, Hoboken: Wiley, 2009.

[19] N. Pillay and R. Qu, Hyper-Heuristics: Fundamentals and Theory, Cham: Springer, 2018.

[20] A. Muklason, "Solver penjadwal ujian otomatis dengan algoritma maximal clique dan hyper-heuristics," in *Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI)*, Pekanbaru, 2017.

[21] G. Dueck, "New optimization heuristics the great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, vol. 104, pp. 86-92, 1993.

[22] N. Al-Milli, "Hybrid Genetic Algorithms with Great Deluge For Course Timetabling," *International Journal of Computer Science and Network Security*, vol. 10, 2010.

[23] H. Turabieh, S. Abdullah and B. Mccolum, "Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem," in *4th International Conference, RSKT 2009*, Gold Coast, 2009.

[24] J. Obit, Y. Kuan, R. Alfred, J. Bolongkikit and O. Sheng, "An Investigation towards Hostel Space Allocation," in *5th ICCST 2018*, Kota Kinabalu, 2018.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis dengan nama Mohammad Refi Nur Ghozi yang lahir di Kota Malang pada tanggal 14 Agustus 1998, merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh jenjang pendidikan formal di beberapa sekolah, yaitu SD Negeri Dinoyo 2 Malang, SMP Negeri 8 Malang, dan SMA Negeri 1 Malang. Penulis melanjutkan pendidikan sarjana di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2016 yang terdaftar sebagai mahasiswa dengan NRP 05211640000104. Selama masa perkuliahan, penulis ikut berpartisipasi dalam organisasi kemahasiswaan sebagai pengurus Himpunan Mahasiswa Sistem Informasi. Penulis juga ikut serta dalam kepanitiaan *Information Systems Expo* 2017 dan 2018. Karena ketertarikannya dalam bidang *Data Science*, penulis mengambil bidang minat Rekayasa Data dan Intelegensi Bisnis (RDIB) di Departemen Sistem Informasi ITS. Penulis dapat dihubungi melalui email refighozi@gmail.com.

Halaman ini sengaja dikosongkan.

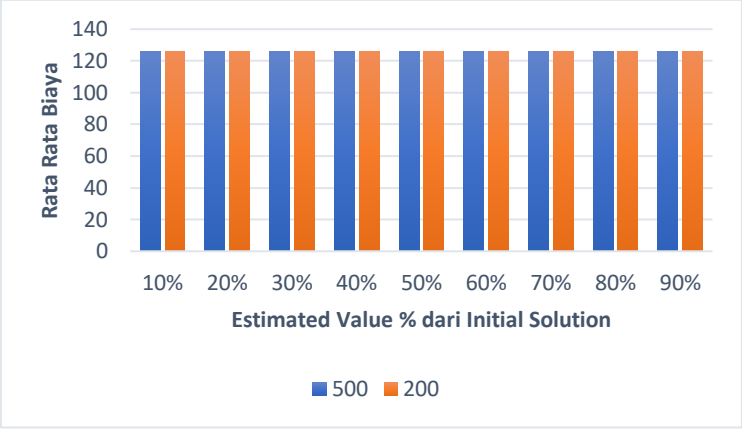
LAMPIRAN A.

Hasil uji coba paramater estimated value *Great Deluge Iterated Local Search*.

INSTANCE 10

Tabel A.1 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 10

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	126	126
20%	126	126
30%	126	126
40%	126	126
50%	126	126
60%	126	126
70%	126	126
80%	126	126
90%	126	126



Gambar A.1 Hasil Uji Coba Parameter Pada Instance 10

INSTANCE 20

Tabel A.2 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 20

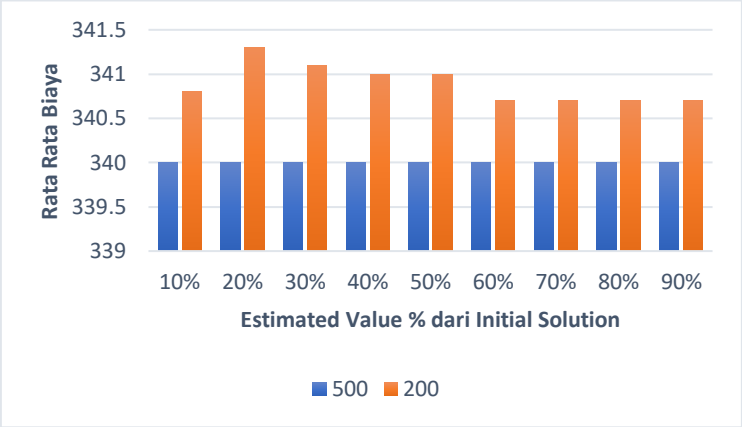
Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	240	240
20%	240	240
30%	240	240
40%	240	240
50%	240	240
60%	240	240
70%	240	240
80%	240	240
90%	240	240

Gambar A.2 Hasil Uji Coba Parameter Pada Instance 20

INSTANCE 30

Tabel A.3 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 30

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	340	340.8
20%	340	341.3
30%	340	341.1
40%	340	341
50%	340	341
60%	340	**340.7**
70%	340	**340.7**
80%	340	**340.7**
90%	340	**340.7**

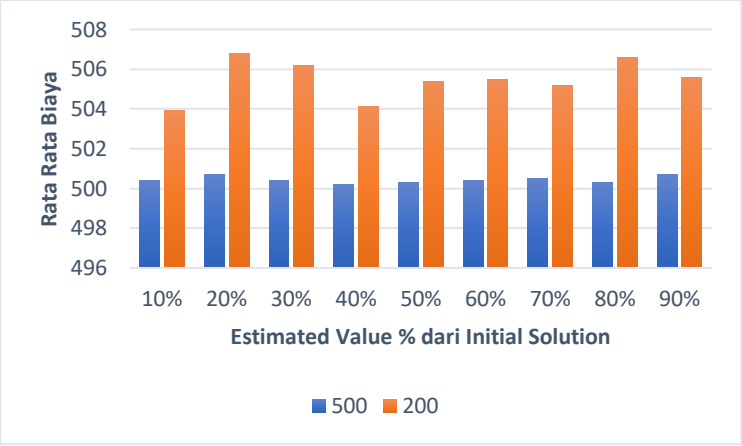


Gambar A.3 Hasil Uji Coba Parameter Pada Instance 30

INSTANCE 40

Tabel A.4 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 40

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	500.4	**503.9**
20%	500.7	506.8
30%	500.4	506.2
40%	**500.2**	504.1
50%	500.3	505.4
60%	500.4	505.5
70%	500.5	505.2
80%	500.3	506.6
90%	500.7	505.6

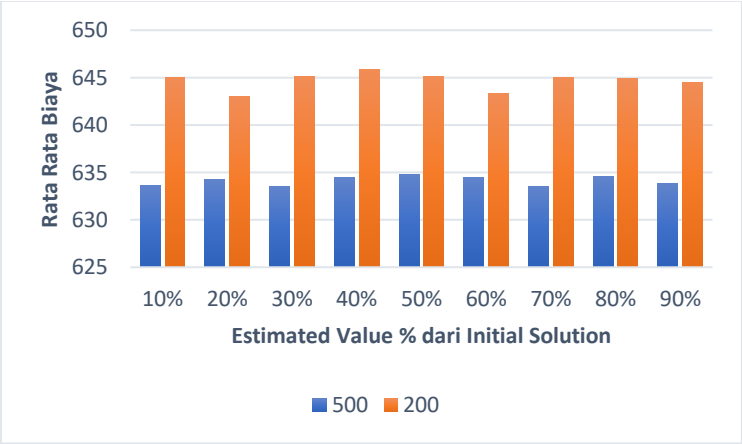


Gambar A.4 Hasil Uji Coba Parameter Pada Instance 40

INSTANCE 50

Tabel A.5 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 50

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	633.6	645
20%	634.3	**643**
30%	**633.5**	645.1
40%	634.5	645.9
50%	634.8	645.1
60%	634.5	643.3
70%	**633.5**	645
80%	634.6	644.9
90%	633.8	644.5

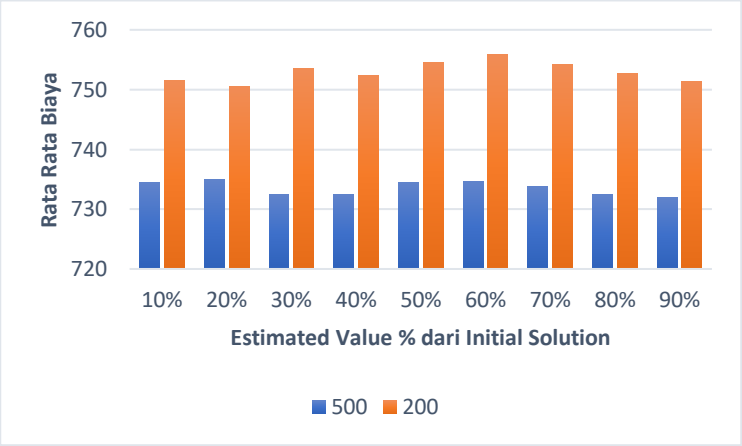


Gambar A.5 Hasil Uji Coba Parameter Pada Instance 50

INSTANCE 60

Tabel A.6 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 60

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	734.5	751.5
20%	735	**750.6**
30%	732.5	753.5
40%	732.5	752.3
50%	734.5	754.5
60%	734.6	755.8
70%	733.8	754.2
80%	732.4	752.7
90%	**732**	751.4

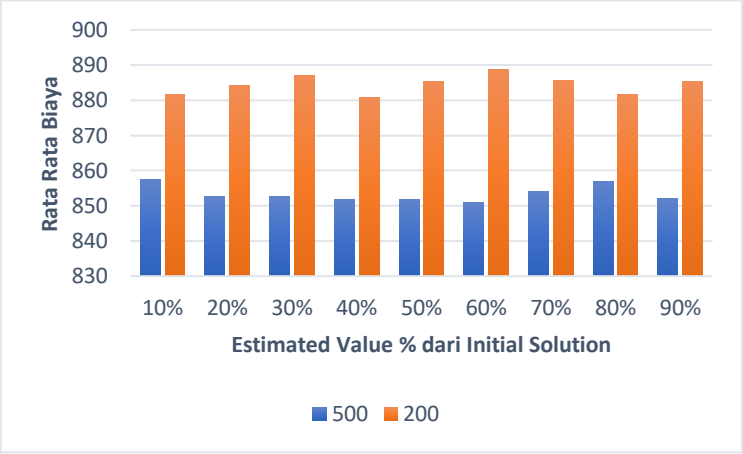


Gambar A.6 Hasil Uji Coba Parameter Pada Instance 60

INSTANCE 70

Tabel A.7 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 70

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	857.4	881.7
20%	852.7	884.1
30%	852.7	887
40%	851.8	**880.8**
50%	851.8	885.4
60%	**850.9**	888.6
70%	854.2	885.7
80%	856.9	881.6
90%	852.2	885.2

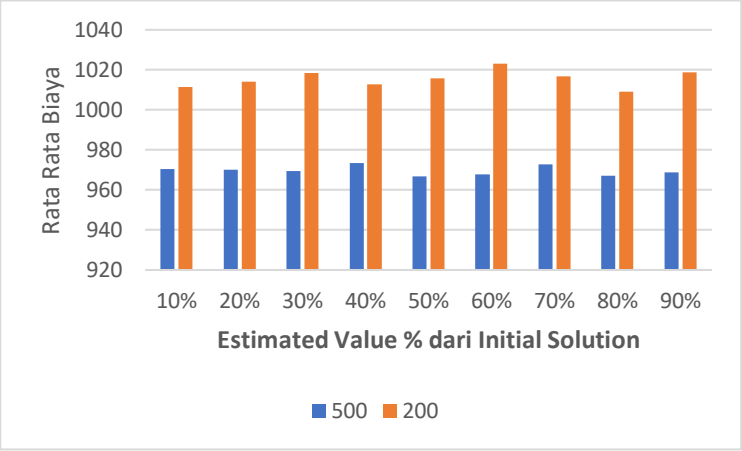


Gambar A.7 Hasil Uji Coba Parameter Pada Instance 70

INSTANCE 80

Tabel A.8 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 80

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	970.5	1011.3
20%	970	1014.2
30%	969.5	1018.4
40%	973.3	1012.8
50%	966.9	1015.9
60%	967.8	1023
70%	972.8	1016.6
80%	**967.1**	**1009**
90%	968.8	1018.7

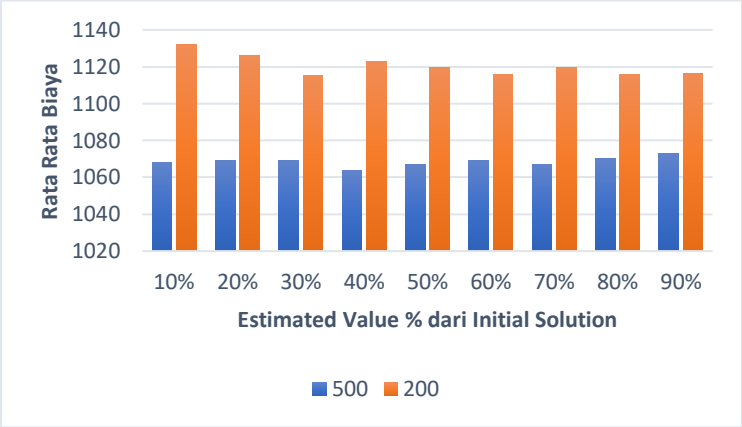


Gambar A.8 Hasil Uji Coba Parameter Pada Instance 80

INSTANCE 90

Tabel A.9 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 90

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	1067.9	1132.2
20%	1069.4	1125.9
30%	1069.2	**1115.1**
40%	**1063.5**	1122.6
50%	1066.7	1119.3
60%	1069.2	1116
70%	1067.2	1119.7
80%	1070.1	1115.9
90%	1072.9	1116.2

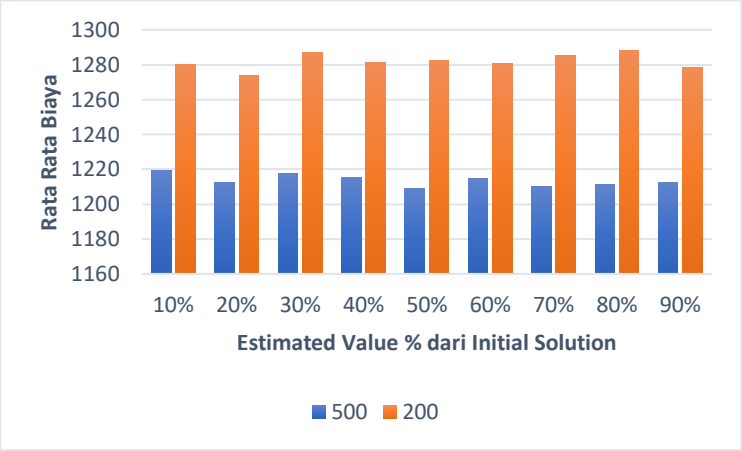


Gambar A.9 Hasil Uji Coba Parameter Pada Instance 90

INSTANCE 100

Tabel A.10 Hasil optimasi tiap parameter *Great Deluge Iterated Local Search* dengan iterasi *Great Deluge* 500 iterasi dan 200 iterasi pada instance 100

Estimated Value	ILS-GD (500 Iterasi)	ILS-GD (200 Iterasi)
10%	1280	1219.1
20%	**1273.8**	1212.7
30%	1287	1217.4
40%	1281.4	1215.2
50%	1282.3	**1209.1**
60%	1280.8	1214.9
70%	1285.5	1210.4
80%	1288.1	1211.1
90%	1278.3	1212.7



Gambar A.10 Hasil Uji Coba Parameter Pada Instance 100

LAMPIRAN B.

Hasil solusi pengalokasian kapal seluruh instance dapat dilihat pada https://intip.in/BAPILSGD

Halaman ini sengaja dikosongkan