UNIVERSITAS INDONESIA

OPTIMASI PENENTUAN RUTE DAN JADWAL KAPAL DALAM MENDISTRIBUSIKAN LPG DENGAN MENJAGA INVENTORI MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

GABRIELA SABAKTANI PARDEDE
0806337623

FAKULTAS TEKNIK
PROGRAM TEKNIK INDUSTRI
DEPOK
JUNI 2012

UNIVERSITAS INDONESIA

OPTIMASI PENENTUAN RUTE DAN JADWAL KAPAL DALAM MENDISTRIBUSIKAN LPG DENGAN MENJAGA INVENTORI MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

GABRIELA SABAKTANI PARDEDE
0806337623

FAKULTAS TEKNIK
PROGRAM TEKNIK INDUSTRI
DEPOK
JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	**:**	**Gabriela Sabaktani Pardede**
NPM	**:**	**0806337623**
Tanda Tangan	**:**	
Tanggal	**:**	**14 Juni 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Gabriela Sabaktani Pardede
NPM	: 0806337623
Program Studi	: Teknik Industri
Judul Skripsi	: Optimasi Penentuan Rute dan Jadwal Kapal dalam Mendistribusikan LPG dengan Menjaga Inventori Menggunakan Algoritma *Tabu Search*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Industri, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	: Ir. Amar Rachman, MEIM	( )
Penguji	: Dr. Akhmad Hidayatno, S.T.,MBT	( )
Penguji	: Armand OM., S.T.,M.Sc.	( )
Penguji	: Romadhani Ardi,S.T.,M.T	( )

Ditetapkan di : Depok
Tanggal : 20 Juni 2012

KATA PENGANTAR

Puji syukur kepada Tuhan Yesus atas segala kasih, berkat dan penyertaanNya yang melimpah sehingga penulis dapat menyelesaikan skripsi ini dengan tepat waktu. Penulis menyadari bahwa skripsi ini tidak akan terselesaikan dengan baik tanpa bantuan pihak-pihak yang membimbing dan mendukung penulis dalam pengerjaan skripsi ini. Untuk itu penulis mengucapkan terima kasih kepada:

1. Bapak Ir. Amar Rachman selaku dosen pembimbing yang selalu setia membimbing, mengarahkan dan dengan sabar mengajari penulis dalam menyelesaikan setiap tahap dalam pengerjaan skripsi ini. Semoga Tuhan selalu memberkati bapak dan keluarga.
2. Bapak Ir. Sumarsono selaku dosen pembimbing II yang memberikan masukan dalam pengerjaan skripsi ini.
3. Bapak Antoni Akbar dan bapak Eko Ricky atas bantuan dan waktunya dalam mengajari penulis untuk mengerjakan skripsi ini.
4. Keluarga terkasih, yaitu Mama, Papa adik-adik penulis (Christy, Ruth dan Hana), Opung, Bundo dan Tante atas segala perhatian, doa, semangat, dan kasih sayang yang tak ternilai. Skripsi ini penulis persembahkan sebagai ungkapan terima kasih atas segala kasih sayang dan kerja keras keluarga pada penulis.
5. Samuel Guswindo (TE'08) yang dengan sabar dan setia membantu penulis dalam mengerjakan dan menyelesaikan program Matlab pada skripsi ini di tengah kesibukannya dalam mengerjakan skripsi juga. Semoga Tuhan membalas kebaikannya.
6. Sahabat-sahabat SAROHA (*one heart*), Kristina, Eltina, Ana, Funi, Jessica, Frihot, Anda, Roro, Pol, Andrew, Andreas dan Rizal sebagai sesama pejuang skripsi yang saling membangun, menyemangati dan mendoakan. Terima kasih untuk persahabatan ini.
7. Teman-teman POFT khususnya angkatan 2008 yang sama-sama berjuang dalam pengerjaan skripsi, dan AKK penulis (Connie, Monika dan Thaza)

yang selalu mendukung dan mendoakan penulis. Terima kasih untuk setiap keceriaan dan semangatnya.

8. Teman-teman angkatan 2008 yang telah bersama dengan penulis selama 4 tahun di Departemen Teknik Industri Universitas Indonesia, yang sama-sama berjuang dalam pengerjaan skripsi dan saling membantu.
9. Pihak-pihak yang tidak bisa penulis sebutkan satu-persatu di sini.

Akhir kata, penulis berharap semoga Tuhan membalas segala kebaikan semua pihak yang telah membantu dan mendukung penulis dalam menyelesaikan skripsi ini. Penulis menyadari bahwa skripsi ini tidak sempurna, oleh karena itu penulis terbuka terhadap saran dan kritik. Semoga skripsi ini bermanfaat bagi pengembangan ilmu.

Depok, 14 Juni 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sitivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Gabriela Sabaktani Pardede

NPM : 0806337623

Program Studi : Teknik Industri

Departemen : Teknik Industri

Fakultas : Teknik

Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalty-Free Right***) atas karya ilmiah saya yang berjudul:

Optimasi Penentuan Rute dan Jadwal Kapal dalam Mendistribusikan LPG dengan Menjaga Inventori Menggunakan Algoritma *Tabu Search*

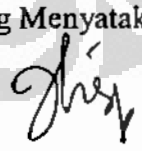
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 14 Juni 2012

Yang Menyatakan



(Gabriela Sabaktani Pardede)

ABSTRAK

Nama	:Gabriela Sabaktani Pardede
Program Studi	:Teknik Industri
Judul	:Optimasi Penentuan Rute dan Jadwal Kapal dalam Mendistribusikan LPG dengan Menjaga Inventori Menggunakan Algoritma *Tabu Search*

Skripsi ini membangun suatu model integrasi antara inventori di pelabuhan *loading* dan ketersediaan muatan di pelabuhan *unloading* melalui penentuan rute dan penjadwalan kapal VLGC yang berfungsi sebagai pengangkut dari pelabuhan *loading* dan sekaligus sebagai *floating storage* di pelabuhan *unloading.* Untuk menentukan rute penjadwalan yang optimal harus ditetapkan berapa jumlah produk yang akan diangkut, kapan waktunya, menggunakan kapal yang mana, memastikan ketersediaan muatan di pelabuhan *unloading*, dan level inventori produk yang tidak melebihi batas kapasitas pelabuhan. Model yang dikembangkan bertujuan untuk meminimalkan biaya dengan dasar algoritma *Tabu Search* dengan *tools* Matlab. Dari hasil *running* program optimasi disimpulkan bahwa model yang dikembangkan memiliki performansi yang baik dibandingkan dengan kondisi *existing*.

Kata Kunci : Rute dan penjadwalan kapal, inventori, algoritma *tabu search, time window.*

ABSTRACT

Name :Gabriela Sabaktani Pardede
Study Program :Industrial Engineering
Title :Ship routing and scheduling optimization in distributing LPG by keeping inventory using Tabu Search algorithm

This research presents a model of integration of inventory at the loading port and the cargo availability at unloading port through routing and ship scheduling VLGC vessel that serves as a transporter of cargo from the loading port as well as floating storage at unloading port. The optimal routing schedule should specify how much of each product to carry, at what time, on which ship, ensure the cargo availability in unloading port, and the stock levels of the products cannot exceed the inventory capacity of loading port. The model has objective function to minimize cost of ship that developed with Tabu Search algorithm using Matlab. From optimization running program conclude that model has good performance compared existing conditions.
Keywords : routing and ship scheduling, inventory, *tabu search algorithm, time window.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1. Latar Belakang Permasalahan

Pelaksanaan Program Peningkatan Pemakaian LPG Domestik menggantikan (konversi) Minyak Tanah untuk Rumah Tangga telah menyebabkan kebutuhan LPG (*Liquid Petroleum Gasses*) dalam negeri meningkat dengan cepat. Hal ini dilihat dari perkiraan peningkatan konsumsi LPG sebesar 3-3.9 juta MT pada tahun 2010-2014 (Data Pertamina 2010). Program ini dilaksanakan pemerintah karena diperkirakan akan memberikan penghematan yang cukup besar pada pos anggaran belanja pemerintah, terutama pengurangan subsidi bahan bakar minyak. Untuk itu, saat ini PT Pertamina (Persero) sebagai perusahaan minyak dan gas negara yang harus dapat menjamin ketersediaan bahan bakar di seluruh nusantara, terus mengupayakan ketahanan persediaan LPG nasional karena meningkatnya permintaan gas tersebut.

Pengupayaan ketersediaan LPG bagi masyarakat Indonesia bukanlah merupakan suatu hal yang sederhana karena wilayah Indonesia secara geografis sangat luas, sehingga diperlukan infrastruktur penyediaan LPG yang cukup kompleks. LPG ini diperoleh dari pengilangan gas alam. Beberapa kilang LPG diantaranya, Natuna, Tj. Santan, Jabung, Bontang dan Sorong.

Gambar 1.1 Peta Lokasi Kilang LPG di Indonesia

Hasil produksi kilang-kilang minyak dan gas ini didistribusikan ke titik-titik permintaan yang dinamakan *End Depot* melalui *transshipment storage* dengan menggunakan kapal LPG yaitu VLGC (*Very Large Gas Carier*), dimana pendistribusian LPG 90% dilakukan dengan menggunakan kapal (Data Pertamina 2009). Dari terminal *End Depot* ini, BBG tersebut kemudian didistribusikan ke depot tujuan melalui jaringan pipa dan truk. Kapal yang digunakan dalam pendistribusian bahan bakar di PT Pertamina (Persero) memiliki suatu fungsi menganut sistem *industrial operating*, yaitu menyiapkan sarana angkut kapal guna **menjamin seluruh kargo yang ada dapat terangkut** dengan **biaya minimum**.

Dua target operasi perkapalan PT Pertamina (Persero) adalah seluruh kargo yang ada dapat terangkut dan biaya operasi minimum. Namun realisasinya pada tahun 2009 tercatat 84 kali terjadi depot kritis, dimana hanya 88,74% LPG yang dapat diangkut disebabkan oleh tidak adanya kargo. Kemudian meningkatnya biaya operasi kapal sebanyak 102%.

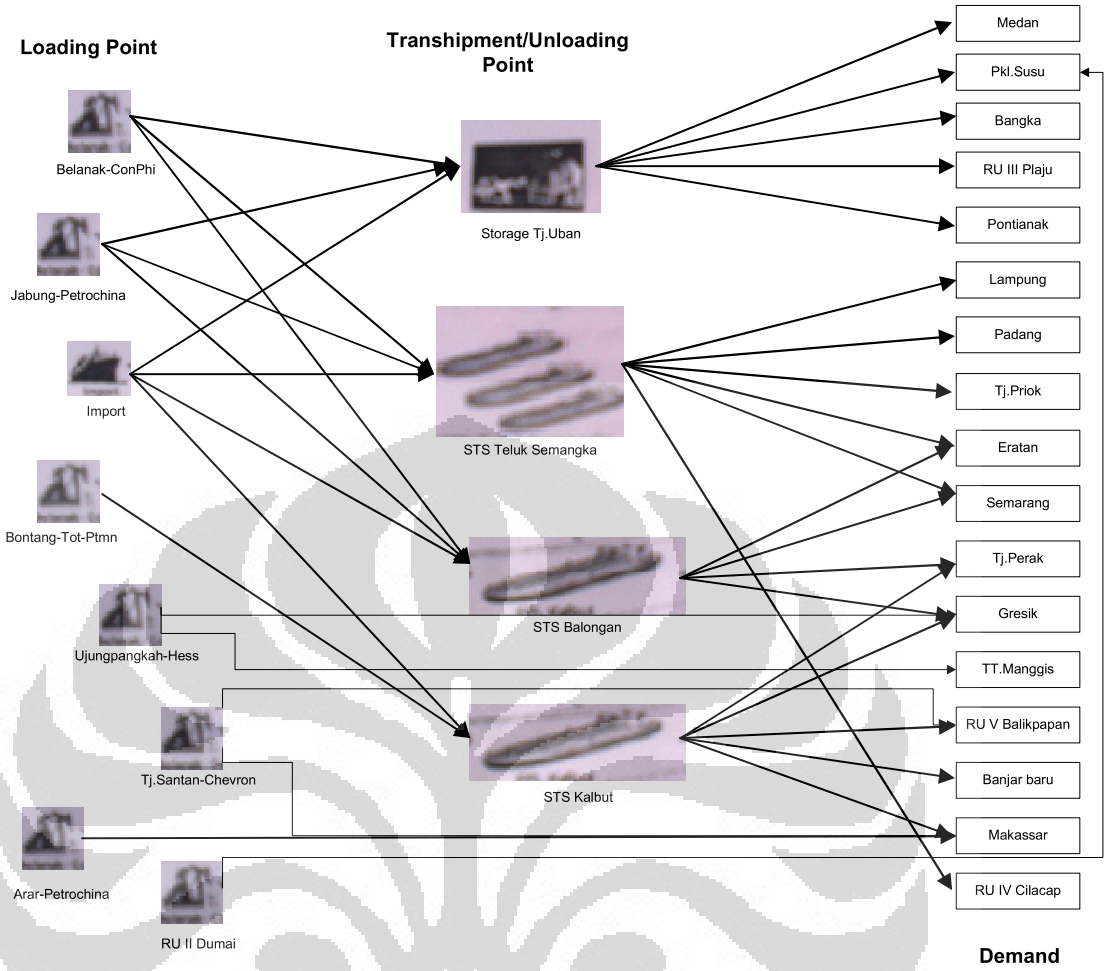
Kekurangan tonase kapal ini disebabkan oleh beberapa hal, yaitu keterlambatan kapal karena proses pengadaan kapal yang lama dan adanya kapal rusak yang merupakan 5% penyebab terjadinya depot kritis. Pada awal perhitungan, jumlah tonase kapal lebih besar dibanding material yang akan diangkut. Namun kenyataannya banyak material yang tidak bisa diangkut karena ketidaktersediaan tonase kapal. Hal ini mengindikasikan adanya masalah penjadwalan kapal yang tidak optimal.

Dari sisi biaya operasi kapal, target biaya tahun 2009 tidak terpenuhi. Target biaya operasi tahun 2009 adalah 89,174 Rp/liter namun terealisasi mencapai 91,086 Rp/liter. Realisasi biaya operasi ini melebihi target sekitar 102%, dimana kondisi operasi perkapalan dipengaruhi oleh *Round Trip Days* (RTD), biaya sewa kapal (*charter rate*), konsumsi bahan bakar, biaya pelabuhan (*port charges*), dan volume angkutan. RTD adalah waktu yang dibutuhkan kapal mulai dari *loaded* (muat), berlayar, *discharge* (bongkar), sampai kembali ke pelabuhan *loading.* RTD ini dipengaruhi oleh jarak, kecepatan kapal, dan *port time* (waktu yang dihabiskan di pelabuhan). Biaya sewa kapal ditentukan oleh pasar, konsumsi bahan bakar ditentukan oleh *performance* mesin kapal, sedangkan biaya pelabuhan tergantung dari bendera kapal tersebut.

Terdapat tiga hal yang menjadi indikator tingginya biaya operasi kapal. Pertama, penambahan kapal melalui *spot charter* (tempat sewa). Kedua, deviasi rute kapal yang rata-ratanya 12 call tiap bulan menjadi 23 call yang berimplikasi pada peningkatan konsumsi bahan bakar. Ketiga, waktu *idle* kapal (kapal tidak beroperasi) dikarenakan *waiting jetty* (kapal bersandar di dermaga), tangki kilang yang masih kosong (*ullage*), dan *daylight* (waktu operasi pelabuhan yang terbatas hanya sampai siang hari), dimana *daylight* adalah 23% penyebab deviasi yang meningkatkan *port time* kapal. Ketiga hal ini berkontribusi pada peningkatan biaya operasi kapal.

Selain masalah tonase kapal dan biaya operasi kapal, *cargo lifting* juga sangat berpengaruh terhadap *Supply Chain Management* secara keseluruhan. Jika kilang sampai berhenti karena tidak ada ruang kosong tangki (*ullage*) dapat menyebabkan biaya *start up* yang tinggi dan juga potensi depot kritis yang tinggi. Karena itu dibutuhkan kapal yang dapat datang tepat waktu (tidak terlambat) untuk mengambil muatan agar tangki di kilang tersebut tidak sampai penuh. Di lain sisi, kebutuhan kapal VLGC sebagai *storage* (tempat penyimpanan) sekaligus *transporter* (pengangkut) di daerah *transshipment* sangat dibutuhkan guna menjamin ketersediaan muatan dan juga menghindari waktu tempuh kapal *shuttle* yang lebih lama yang dapat meningkatkan biaya.

Aliran kapal pada pendistribusian LPG dilakukan dari *supply point* kemudian ke *transhipment* dan berakhir di *discharging port* seperti pada gambar 1.3. Terdapat 8 buah *supply point (loading point)* yaitu 1) Belanak (*Conoco Philips*), 2) Jabung (*Petrochina*), 3) Import (Arab, Australia, Iran dan Singapore), 4) Bontang (Total/Pertamina), 5) Ujung Pangkah (*Hess*), 6) Tj.Santan (*Chevron*), 7) RU II Dumai, 8) Arar (*Petrochina*). Kemudian terdapat 4 *transhipment, 3* berupa *floating storage* yang berada di Teluk Semangka, Balongan, Kalbut, dan 1 berupa *storage* darat yang berada di Tj.Uban. *Floating storage* ini adalah kapal VLGC yang menjadi pengangkut dari s*upply point* ke *transshipment* sekaligus sebagai *storage* seperti yang telah disebutkan sebelumnya. Kemudian di *transshipment* ini akan datang kapal *shuttle* untuk mengambil muatan (berupa LPG) untuk disalurkan ke setiap *discharging port*. Pemindahan muatan dari VLGC ke *shuttle* ini disebut dengan STS (*ship to ship*).



Gambar 1.2 Pola Distribusi LPG

Terdapat lima kapal VLGC, tiga di Teluk Semangka, satu di Balongan, dan satu di Kalbut. Jika muatan di kapal ini habis, maka kapal ini dapat bergerak ke kilang (*supply point*).

Jika tangki di kilang sudah penuh, kapal VLGC harus digerakkan ke kilang (Belanak, Jabung, Bontang) untuk digunakan sebagai tempat penyimpanan (*storage*) dari LPG yang belum dimuat. Hal ini dilakukan untuk mencegah terjadinya pemberhentian produksi di kilang karena tidak ada tangki kosong. Namun, ketersediaan muatan di *unloading point* akan terancam karena tidak adanya *floating storage* di *transshipment*.

Dari semua latar belakang di atas, sangat jelas diperlukan integrasi inventori di kilang dan ketersediaan muatan di *storage* (*transshipment*) dengan penjadwalan kapal dan biaya yang minimum. Untuk menentukan rute penjadwalan yang

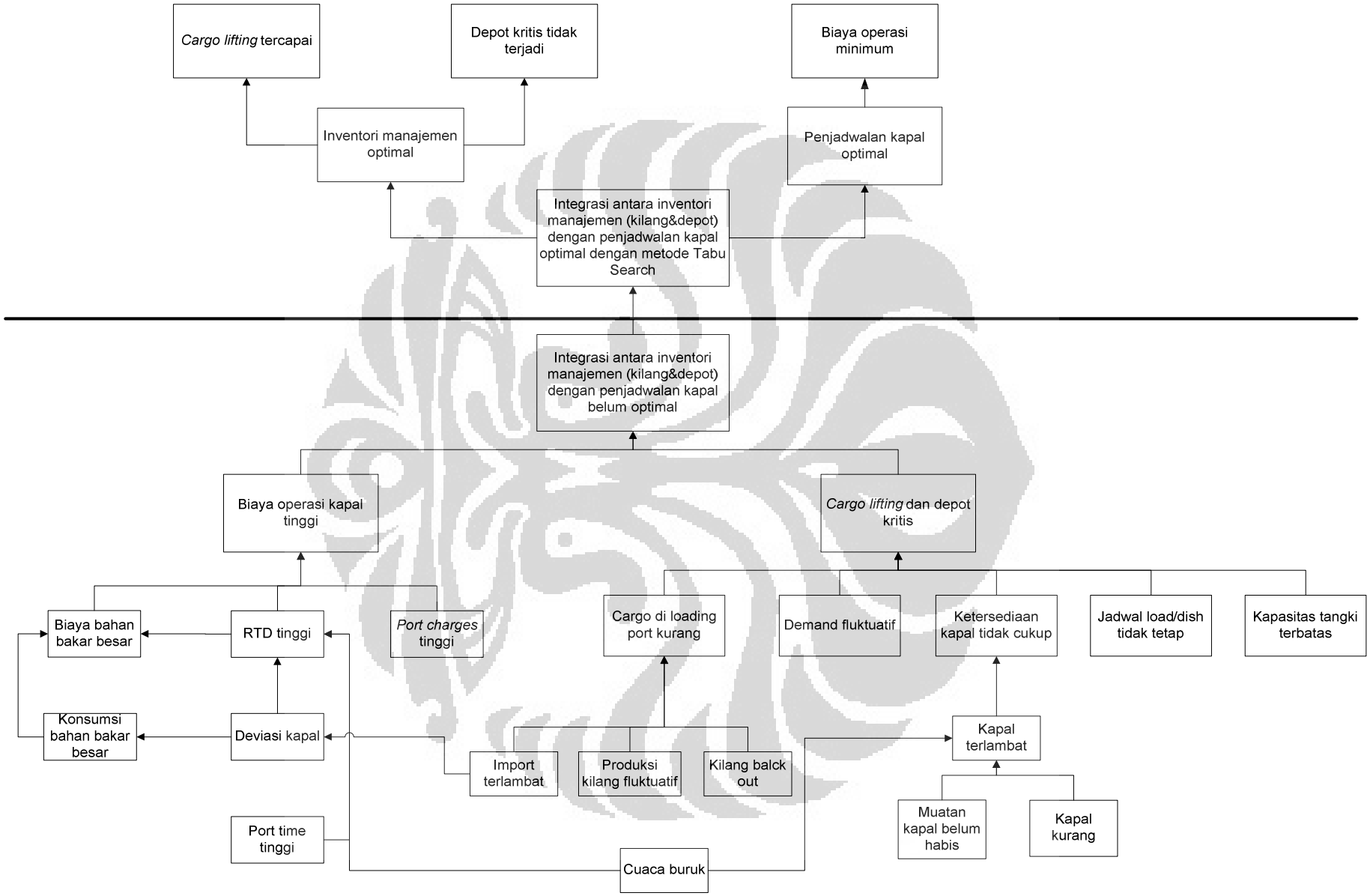
optimal haruslah menetapkan berapa jumlah produk yang akan diangkut dari satu titik ke titik lain, kapan waktunya, menggunakan kapal yang mana, memastikan semua titik/pelabuhan memiliki produk yang cukup untuk dikonsumsi, dan level inventori produk yang tidak melebihi batas kapasitas pelabuhan (Khayyal et al, 2005). Selain itu, dalam transportasi perkapalan, muatan yang banyak biasanya dimuat dan dibongkar di setiap pemberhentian di pelabuhan. Operasi loading dan pengangkutan memerlukan waktu dan biaya yang mahal. Oleh karena itu perencanaan rute dan manajemen inventori memiliki potensi besar jika kedua hal ini dapat disinkronisasikan (Christiansen et al, 2010). Fagerholt et al, 2009 menggambarkan dasar dari MIRP (*Maritime Inventory Routing Problem*) yaitu sistem transportasi dari *single product* yang diproduksi di pelabuhan muat dan dikonsumsi di pelabuhan bongkar, dimana setiap pelabuhan memiliki kapasitas tangki persediaan dan kecepatan produksi dan konsumsi.

Salah satu metode yang dapat digunakan untuk mengoptimalkan jaringan logistik tersebut adalah dengan pendekatan model *Mixed Integer Problem* yang diselesaikan dengan Algoritma *Tabu Search*, dimana algoritma ini menuntun setiap tahapannya agar dapat menghasilkan criteria apsirasi yang paling optimum tanpa terjebak ke dalam solusi awal yang ditemukan selama tahapan berlangsung.

Dari hasil penelitian sebuah jurnal ditunjukkan bahwa Tabu Search heuristic menyediakan solusi optimal atau *near-optimal* dalam waktu yang layak untuk permasalahan rute dan jadwal kapal. Metode ini dapat memberikan solusi yang lebih baik dari masalah dengan kendala yang besar dengan waktu komputasi yang lebih singkat dari metode lain seperti *multi-local search* (Korsvik et al, 2010). Dengan metode ini akan dilakukan eksplorasi yang lebih dalam dengan kualitas solusi yang jauh lebih baik dan metode ini cocok untuk permasalahan dengan variabel yang tak terbatas.

1.2. Diagram Keterkaitan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dibuat diagram keterkaitan masalah yang menampilkan permasalahan secara visual dan sistematis. Diagram keterkaitan masalah adalah seperti yang ditunjukkan pada Gambar1.3.



Gambar 1.3 Diagram Keterkaitan Masalah

1.3. Rumusan Masalah

Penjadwalan kapal bertujuan untuk meminimalisasi biaya operasi kapal (konsumsi bahan bakar, biaya pelabuhan, dll) dengan permasalahan menjaga inventori di kilang dan juga menjaga ketersediaan muatan di STS (*ship to ship*) agar suplai ke konsumen terjamin. Jadi, pokok permasalahan yang akan dibahas adalah bagaimana mengatur penjadwalan kapal dengan menjaga *ullage* (dalam hal ini inventori di tangki kilang) dan menjaga ketersediaan muatan di STS dengan biaya transportasi yang minimum.

1.4. Tujuan dan Manfaat Penelitian

Adapun tujuan dari penelitian ini adalah memperoleh suatu model terintegrasi antara penjadwalan kapal dengan inventori level di *storage* kilang dengan biaya minimum menggunakan Algoritma Tabu Search.

Manfaat yang diperoleh dari penelitian ini adalah perbaikan penjadwalan kapal, khususnya kapal pengangkut LPG untuk memperoleh inventori yang optimal di *storage* kilang, yang dapat digunakan sebagai masukan pada Pertamina dalam mengatur penjadwalan kapal.

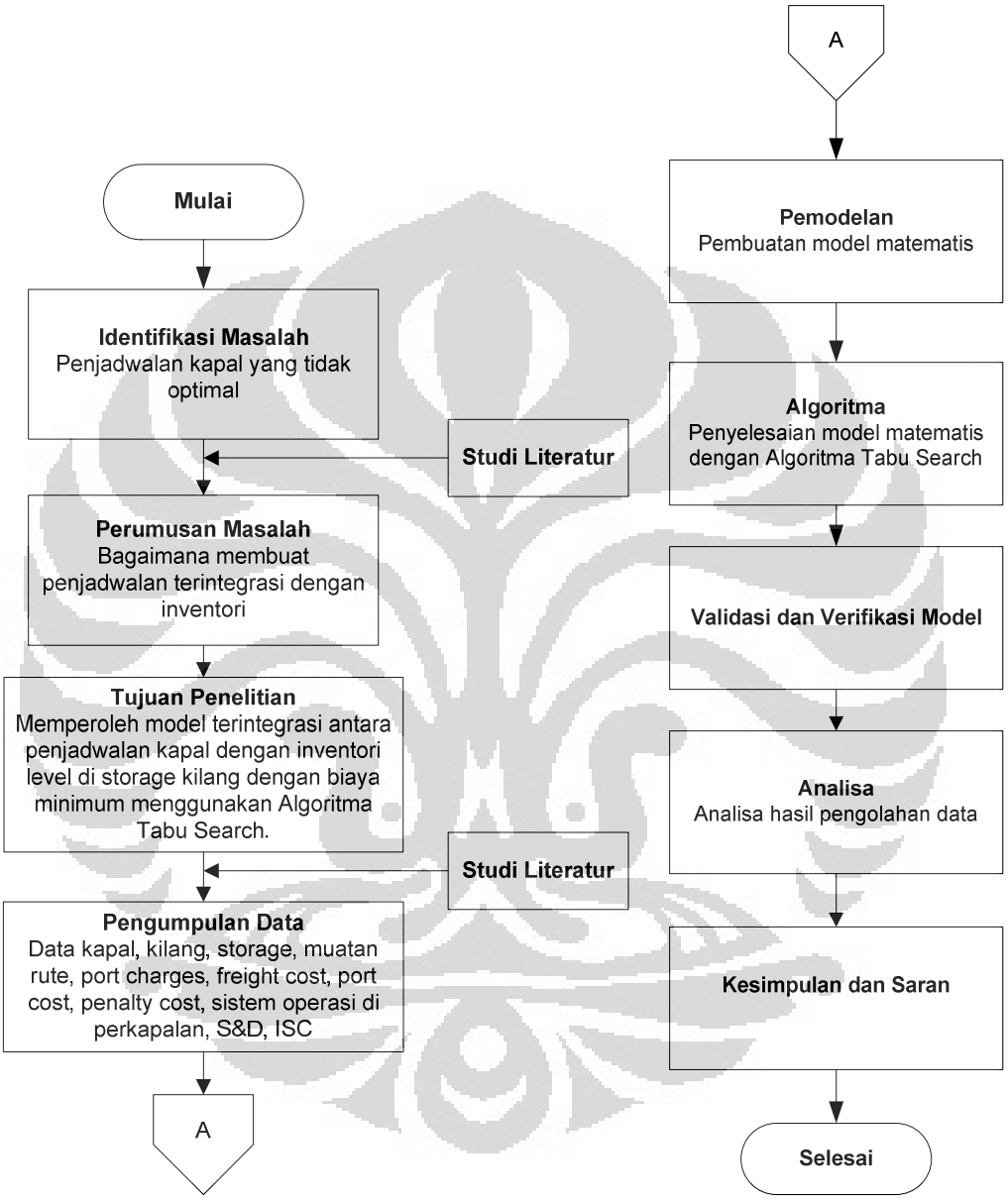
1.5. Ruang Lingkup Penelitian

Untuk mendapatkan hasil penelitian yang spesifik dan terarah sesuai dengan tujuan pelaksanaannya, dan agar memudahkan formula dalam program nantinya, maka dilakukan pembatasan masalah sebagai berikut:

a. Rute dan jarak ditentukan pada data realisasi seluruh pelayaran kapal PT Pertamina (Persero).
b. Kapasitas produksi tiap kilang dan *demand* tiap *storage* dianggap konstan selama periode perhitungan.
c. *Single product* (LPG Mix).
d. Pergerakan kapal dibatasi dari pelabuhan *loading* ke pelabuhan *unloading* atau sebaliknya.
e. Waktu perencanaan selama 60 hari.

1.6. Metodologi Penelitian

Metodologi penelitian ini bertujuan untuk mempermudah dan membantu penulis dalam memecahkan permasalahan tersebut. Metodologi penelitian ini dijelaskan secara sistematis pada Gambar 1.4 dengan uraian sebagai berikut:



Gambar 1.4 Diagram Alir Metodologi Penelitian

1. Identifikasi permasalahan

   Pada tahap pertama, dilakukan pencarian tema-tema yang mungkin untuk diangkat terkait tidak tercapainya kinerja operasional dan *financial* divisi operasi perkapalan dalam mengoperasikan kapal-kapal LPG. Teridentifikasi adanya permasalahan penjadwalan kapal.

2. Perumusan masalah

   Berdasarkan identifikasi masalah yang diperoleh, dapat dirumuskan permasalahan dalam penelitian ini yaitu belum optimalnya integrasi antara inventori level dengan penjadwalan kapal, sehingga diperlukan model penjadwalan kapal yang dapat mengintegrasikan dua kepentingan pokok dan berbeda antara *lifting* muatan di *loading port* agar selalu ada volume tangki (*ullage*), agar kilang dapat terus berproduksi dengan ketersediaan muatan di *unloading port*, dimana hal ini terintegrasi oleh penjadwalan kapal VLGC yang berfungsi sebagai *transporter* dan *floating storage*.

3. Penentuan tujuan penelitian

   Setelah masalah teridentifikasi, tujuan penelitian dapat ditentukan, yaitu membangun model integrasi antara inventori dengan penjadwalan kapal.

4. Pengumpulan data

   Dilakukan pengumpulan data primer dari database divisi operasi perkapalan antara lain:

   - Data kapal, mulai Januari 2009 s/d Desember 2010.
   - Data pelabuhan terkait termasuk fasilitas pendukung yang terkait dengan operasi perkapalan.
   - Data operasi perkapalan termasuk, rute, *fuel oil consumption*, *loading time*, *unloading time*, dan *port charges*.
   - Sistem operasi dalam tata kerja oganisasi fungsi-fungsi terkait.

5. Pemodelan

   Pada tahap ini dilakukan pembuatan model awal optimasi (fungsi tujuan, kendala, konstanta dan variabel).

6. Pengolahan data

   Pada tahap ini dilakukan pengolahan dari data yang diperoleh. Pengolahan data ini dilakukan sesuai model matematis dengan bantuan algoritma *Tabu Search* dan *tools* perangkat lunak VBA (*Visual Basic Application*).

7. Menganalisis hasil

   Dalam tahap ini dilakukan analisis terhadap hasil output untuk dilihat dan dibandingkan dengan keadaan yang selama ini berlangsung.

8. Menarik kesimpulan

   Pada tahap ini akan dihasilkan kesimpulan mengenai keseluruhan penelitian yang merupakan inti sari dari hasil penelitian dan analisis yang telah dilakukan sebelumnya.

1.7. Sistematika Penulisan

Untuk memudahkan penulisan skripsi ini, maka secara garis besar akan disusun dalam penulisan yang sistematis. Skripsi ini dibagi menjadi 5 bab.

BAB I : PENDAHULUAN

Berisi penjelasan secara singkat dari isi keseluruhan skripsi ini. Bab ini berisikan latar belakang permasalahan, diagram keterkaitan permasalahan, rumusan permasalahan, batasan masalah, tujuan dan manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II : LANDASAN TEORI

Berisikan teori, konsep, dan prinsip yang berkaitan dengan masalah yang dihadapi dan menjadi landasan konseptual sebagai pedoman pemecahan masalah.

BAB III : PENGUMPULAN DAN PENGOLAHAN DATA

Menyajikan data yang diperoleh baik berupa rute pelayaran maupun biaya transportasi yang akan diolah dan dikembangkan ke dalam model matematis sehinga diperoleh penjadwalan kapal yang terintegrasi dengan inventori di kilang dengan minimum biaya.

BAB IV : HASIL DAN PEMBAHASAN

Berisi pembahasan tentang hasil dan analisis dari pengumpulan dan pengolahan data terhadap berbagai aspek.

BAB V : KESIMPULAN DAN SARAN

Bab ini merupakan tahap terakhir dari penyusunan skripsi yang berisikan kesimpulan dari hasil penelitian dan saran-saran yang dapat menjadi masukan bagi objek penelitian.

BAB 2

DASAR TEORI

Logistik mempunyai pengaruh yang signifikan terhadap biaya dan keputusan perusahaan. Logistik juga berpengaruh untuk menghasilkan level pelayanan kepada konsumen yang berbeda-beda. Tujuan akhir manajemen logistik adalah mendapatkan sejumlah barang atau jasa yang tepat pada tempat dan waktu yang tepat, serta kondisi yang diinginkan dengan memberikan kontribusi terbesar bagi perusahaan[1].

Untuk mencapai tujuan akhir manajemen logistik, diperlukan suatu sistem distribusi produk yang:

- Memastikan bahwa produk yang tersedia pada waktu dan jumlah yang tepat sesuai permintaan konsumen
- Memiliki kualitas yang terjamin
- Memperhatikan tingkat keselamatan dalam pendistribusiannya.

Suatu perusahaan harus dapat mengoptimalkan sistem distribusinya agar dapat bersaing dengan perusahaan sejenis lainnya. Salah satu caranya adalah dengan pengoptimalan transportasi. Salah satu permasalahan dalam transportasi adalah *Vehicle Routing Problem* (VRP), yaitu merancang *m* set rute kendaraan dengan biaya rendah dimana tiap kendaraan berawal dan berakhir di depot dan setiap konsumen hanya dilayani sekali oleh sebuah kendaraaan, serta total permintaan yang dibawa tidak melebihi kapasitas kendaraan. Transportasi ini memberikan kontribusi biaya 1/3 sampai 2/3 dari total biaya distribusi[1].

2.1 *Vehicle Routing Problem*

Vehicle Routing Problem (VRP) adalah suatu metode yang digunakan untuk menentukan rute pada suatu armada kendaraan, baik dari *single depot* ataupun *multiple depot* sehingga dapat melayani pelanggan yang tersebar secara geografis[2]. Pada umumnya tujuan yang ingin dicapai dalam VRP adalah

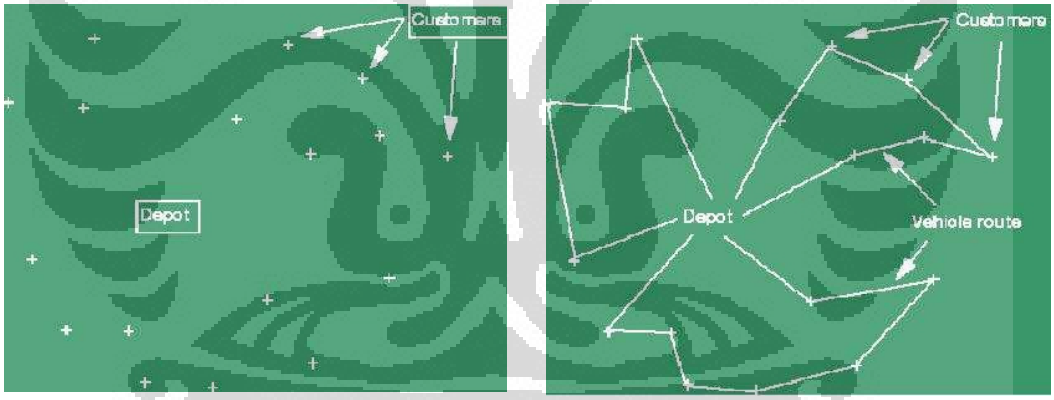
[1] Ronald H. Ballou, 2004

[2] Berbane Dorronsoro Diaz. *What is VRP?* The VRP Web. Malaga: Auren. 2002:1

meminimalkan jarak tempuh kendaraan dan biaya transportasi dalam melakukan pengiriman ke pelanggan sesuai dengan jumlah permintaannya masing-masing.

Vehicle Routing Problem (VRP) pertama kali dikenalkan oleh Dantzig dan Ramser pada tahun 1959. VRP memegang peranan penting pada manajemen distribusi dan telah menjadi salah satu permasalahan dalam optimalisasi kombinasi yang dipelajari secara luas. VRP merupakan manajemen distribusi barang yang memperhatikan pelayanan, periode waktu tertentu, sekelompok konsumen dengan sejumlah kendaraan yang berlokasi pada satu atau lebih depot, yang dijalankan oleh sekelompok pengendara dengan menggunakan *road network* yang sesuai. Solusi dari sebuah VRP yaitu menentukan sejumlah rute yang masing-masing dilayani oleh suatu kendaraan yang berasal dan berakhir pada depotnya, sehingga kebutuhan pelanggan terpenuhi, semua permasalahan operasional terselesaikan dan biaya transportasi secara umum diminimalkan.

Dua gambar berikut menunjukkan masukan yang biasa ada pada VRP dan salah satu kemungkinan keluaran yang ada:



Gambar 2.1 Contoh *Input* dan *Output* dalam VRP

Gambar 2.1 menunjukkan contoh *input* dari sebuah permasalahan VRP dimana terdapat satu depot dan sejumlah pelanggan yang tersebar di berbagai daerah, dan *output* menggambarkan contoh hasil yang diperoleh dari VRP yaitu rute distribusi yang optimal untuk melakukan pengiriman ke pelanggan.

Karakteristik konsumen dalam VRP:

- Menempatkan *road graph* dimana konsumen berada
- Adanya permintaan dalam berbagai tipe dan harus diantarkan ke tempat konsumen

- Terdapat periode waktu (*time window*) dimana konsumen dapat dilayani
- Waktu yang dibutuhkan untuk mengantarkan barang ke lokasi konsumen, hal ini dapat berhubungan dengan jenis kendaraan
- Sekelompok kendaraan tersedia digunakan untuk melayani konsumen

Terdapat empat tujuan umum VRP[3], yaitu:

- Meminimalkan biaya transportasi global, terkait dengan jarak dan biaya tetap yang berhubungan dengan kendaraan
- Meminimalkan jumlah kendaraan yang dibutuhkan untuk melayani semua konsumen
- Menyeimbangkan rute untuk waktu perjalanan dan muatan kendaraan
- Meminimalkan penalti akibat pelayanan yang kurang memuaskan dari konsumen

Pada VRP yang terjadi di lapangan, terdapat beragam batasan yang membuat permasalahan ini menjadi kompleks sehingga terdapat beberapa jenis VRP sesuai dengan batasan yang dimiliki. Menurut Toth dan Vigo (2002) ditemukan variasi permasalahan utama VRP yaitu:

- *Capacitated* VRP (CVRP)

  Faktor: Setiap kendaraan punya kapasitas yang terbatas.
- VRP *with Time Windows* (VRPTW)

  Faktor: Setiap pelanggan harus disuplai dalam jangka waktu tertentu.
- *Mulitple Depot* VRP (MDVRP)

  Faktor: Distributor memiliki banyak depot untuk menyuplai pelanggan.
- VRP *with Pick-Up and Delivering* (VRPPD)

  Faktor: Pelanggan dapat mengembalikan barang pada depot asal.
- *Split Delivery* VRP (SDVRP)

  Faktor: Pelanggan dilayani dengan kendaraan berbeda.
- *Stochastic* VRP (SVRP)

  Faktor: Munculnya '*random values*' (seperti jumlah pelanggan, jumlah permintaan, waktu pelayanan atau waktu perjalanan).
- *Periodic* VRP

  Faktor: Pengantaran hanya dilakukan di hari tertentu.

[3] Toth and Vigo, 2002

VRP merupakan sebuah problem pemrograman integer yang masuk kategori *NP-Hard Problem*, yang berarti usaha komputasi yang digunakan akan semakin sulit dan banyak seiring dengan meningkatnya ruang lingkup masalah. Untuk masalah-masalah seperti ini, biasanya yang dicari adalah aproksimasi solusi yang terdekat, karena solusi tersebut dapat dicari dengan cepat dan cukup akurat. Biasanya masalah ini terselesaikan dengan menggunakan berbagai variasi dari metode heuristik yang memerlukan sedikit pengamatan pada ruang lingkup masalah.

Beberapa metode yang digunakan untuk menyelesaikan VRP antara lain adalah dengan pendekatan eksak, heuristic dan metaheuristik. Dibandingkan dengan heuristic klasik, metaheuristik menunjukkan pencarian solusi yang lebih teliti. Penelitian dalam metaheuristik ini lebih menunjukkan perkembangan yang hebat dalam deckde terkahir dan telah menghasilkan heuristic VRP yang lebih efektif dan fleksibel[4]. *Tabu search* (TS) merupakan metode terbaik yang dapat diimplementasikan pada VRP dibanding metaheuristik lain seperti *Simulated Annealing, Genetic Search, Ant Colony* dan *Neural Network.*

2.2 *Vehicle Routing* dan *Scheduling*

Vehicle routing dan *scheduling* merupakan perluasan dari VRP. Beberapa batasan yang realistis yang termasuk di dalamnya adalah sebagai berikut[1]:

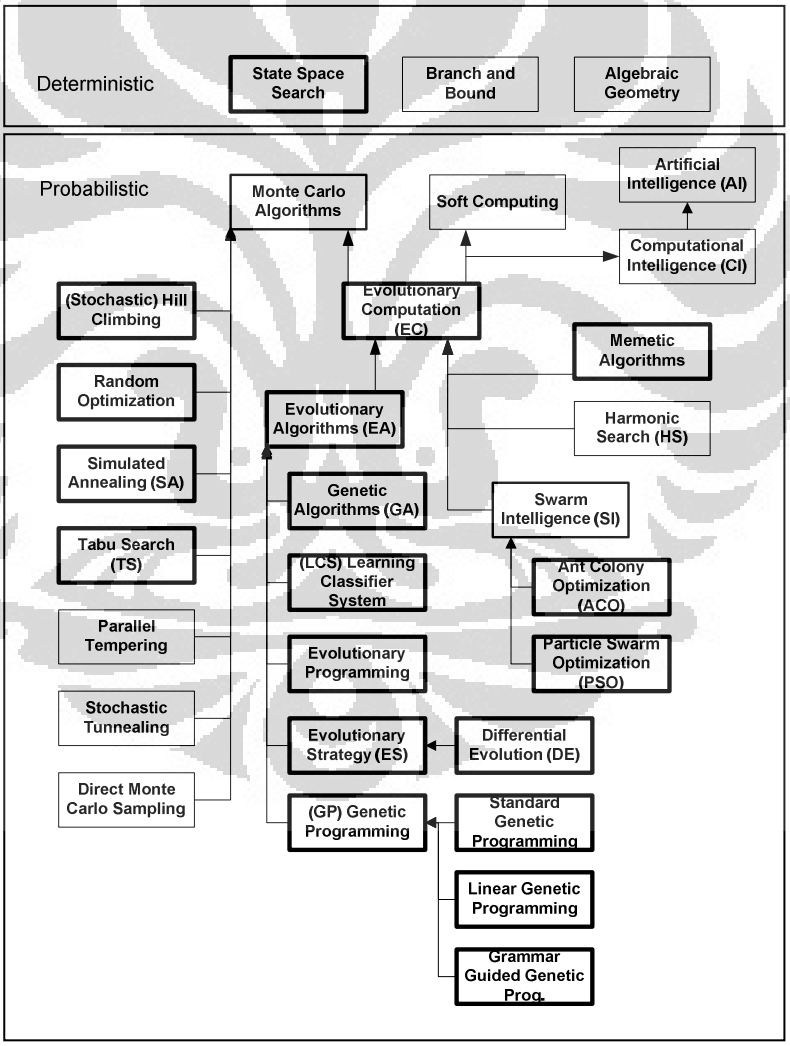
1. Dalam setiap titik pemberhentian ada sejumlah volume yang diambil dan dikirim.
2. Jumlah kendaraan yang diupayakan minimal namun mampu mengirim semua kebutuhan produk.
3. Maksimum total waktu kerja operator kendaraan untuk melakukan pengiriman sesuai dengan ketentuan jam kerja depot maupun konsumen.
4. Titik pemberhentian pelanggan hanya memperbolehkan pengiriman dan/atau pengambilan produk pada waktu tertentu (disebut: *time window*).
5. Pengambilan hanya boleh dilakukan setelah dilakukan pengiriman.
6. Titik pemberhentian tidak dapat menerima kendaraan pada saat bersamaan.

[4] Gendreau M, Laporte G and Potvin JY. Metaheuristic for the capacitated VRP. 2002.

7. Pengiriman selanjutnya ke titik pemberhentian harus mempunyai interval waktu dengan pengiriman sebelumnya.

2.3 Metode Penyelesaian VRP

Secara umum VRP dapat diselesaikan dengan menggunakan dua jenis pendekatan, yaitu pendekatan eksak dan heuristic. Kemudian secara umum kelas penyelesaian heuristic dalam VRP dapat dibagi menjadi dua, yaitu heuristic klasik dan heuristic modern (*metaheuristic*). Heuristic klasik dikembangkan antara tahun 1960 hingga 1990, dilanjutkan dengan pengembangan algoritma metaheuristik hingga kini. Kerangka algoritma dapat dilihat pada gambar diagram berikut:



Gambar 2.2 Kerangka Algoritma

2.3.1 Pendekatan Eksak

Pada solusi eksak, dilakukan pendekatan dengan menghitung setiap solusi yang mungkin sampai satu terbaik dapat diperoleh. Terdapat beberapa algoritma eksak utama penyelesaian VRP, yaitu:

- *Branch and Bound*
- *Branch and Cut*
- *Set Covering Based*

Salah satu pendekatan eksak yaitu *Branch and Bound*. Metode ini dapat memecahkan masalah pemrograma integer. Prinsip dasar metode ini adalah memecah daerah fisibel suatu masalah program linear-relaksasi dengan membuat subproble-subproblem. Ada dua konsep dasar dalam algoritma *branch and bound.*

✓ Cabang (*Branch*)

Membuat partisi daerah solusi dari masalah utama dengan membentuk subproblem yang tujuannya adalah menghapus daerah solusi yang tidak fisibel. Hal ini dicapai dengan menentukan kendala yang penting untuk menghasilkan solusi programa integer, secara tidak langsung titik integer yang tidak fisibel tehapus. Karena sifat partisi tersebut, maka prosedur ini dinamakan percabangan.

✓ Batas (*Bound*)

Jika permasalahan utama berupa maksimisasi, nilai objektif yang optimal untuk setiap subproblem dibuat dengan membatasi pencabangan dengan batas atas dari nilai objektif yang dihubungkan dengan sembarang nilai integer yang fisibel. Hal ini sangat penting untuk mengatur dan menempatkan solusi optimal. operasi pmbatasan ini dinamakan pembatasan (*bounding*) (Taha, 1975).

Metode *branch and bound* diawali dari menyelesaikan program linier (PL) dari suatu *integer programming* (IP). Jika semua nilai variabel keputusan solusi optimal sudah berupa integer, maka solusi tersebut merupakan solusi optimal IP. Jika tidak, dilakukan pencabangan dan penambahan batasan pada PL dan kemudian diselesaikan.

Secara umum penggunaan metode eksak untuk penyelesaian VRP akan menghabiskan waktu yang lama. Hal tersebut dikarenakan VRP termasuk

permasalahan *NP-hard* (*Non Polynominal-hard*), dimana kompleksitas penyelesaian permasalahan akan meningkat secara eksponensial dengan semakin besarnya permasalahan. Hingga kini, belum ada algoritma eksak yang mampu menyelesaikan kasus-kasus yang terdiri lebih dari 50 pelanggan secara konsisten[5]. Oleh karena itu, berbagai penelitian terhadap algoritma heuristic telah dilakukan untuk menyederhanakan penyelesaian VRP.

2.3.2 Heuristik

Metode heuristic memberikan suatu cara untuk menyelesaikan permasalahan optimasi yang lebih sulit dan dengan kualitas dan waktu penyelesaian yang lebih cepat daripada solusi eksak. Contoh metode heiristik antara lain *Saving Based, Matching Based, Multiroute improvement heuristic,* dll.

2.3.3 Metaheuristik

Metaheuristik adalah suatu metode untuk melakukan eksplorasi yang lebih dalam pada daerah daerah yang menjanjikan dari ruang solusi yang ada. Kualitas solusi yang dihasilkan dari metode ini jauh lebih baik daripada yang diperoleh dari heuristic klasik. Contoh metaheuristik adalah *genetic algorithm, simulated annealing, ant colony, tabu searchi,* dll.

2.4 Algoritma *Tabu Search*

Tabu search dikembangkan oleh Glover pada pertengahan 1980-an. Beberapa ide dasarnya dikenalkan oleh Hansen dan selanjutnya diformalkan oleh Glover dan De Wera and Hertz. Tabu search berasal dari Tongan, suatu bahasa Polinesia yang digunakan oleh suku Aborigin Pulau Tonga untuk mengindikasikan suatu hal yang tidak boleh "disentuh" karena sakralnya. Menurut kasus Webster, Tabu berarti larangan yang dipaksakan oleh kebudayaan social sebagai suatu tindakan pencegahan atau sesuatu yang dilarang karena berbahaya. Bahaya yang harus dihindari dalam *Tabu Search* adalah penjadwalan yang tidak layak dan terjebak tanpa ada jalan keluar.

Tabu search merupakan algoritma yang menuntun setiap tahapannya agar dapat menghasilkan kriteria aspirasi yang paling optimum tanpa terjebak ke dalam solusi awal yang ditemukan selama tahapannya berlangsung. Algoritma ini

[5] Toth, P & Vigo, D. (1998). Exact solution of the vehicle routing problem. *Fleet Management and Logistic*

mencegah terjadinya perulangan dan ditemukannya solusi yang sama pada suatu iterasi yang akan digunakan lagi pada iterasi selanjutnya (dikatakan tabu). *Tabu search* adalah sebuah metode optimasi yang berbasis pada *local search*. Proses pencarian bergerak dari satu solusi ke solusi berikutnya dengan cara memilih solusi terbaik *neighborhood* solusi sekarang (*current*) yang tidak tergolong solusi terlarang.

Ide dasar dari algoritma *tabu search* adalah mencegah proses pencarian dari *local search* agar tidak melakukan pencarian ulang pada ruang solusi yang sudah pernah ditelusuri, dengan memanfaatkan suatu struktur memori yang mencatat sebagian jejak proses pencarian yang telah dilakukan. Struktur memori fundamental dalam *tabu search* dinamakan *tabu list*.

Tabu list menyimpan atribut dari sebagian *move* (transisi solusi) yang telah diterapkan pada iterasi-iterasi sebelumnya. *Tabu search* menggunakan *tabu list* untuk menolak solusi-solusi yang memenuhi atribut tertentu guna mencegah proses pencarian mengalami *cycling* pada daerah solusi yang sama dan menuntun proses pencarian menelusuri daerah solusi yang belum dikunjungi. Tanpa menggunakan strategi ini, *local search* yang sudah menemukan solusi optimum local dapat terjebak pada daerah solusi optimum local tersebut pada iterasi-iterasi berikutnya. List ini mengikuti aturan LIFO dan biasanya sangat pendek (panjangnya biasanya sebesar O ($\sqrt{N}$), dimana *N* adalah jumlah total dari operasi).

Perekaman solusi secara lengkap dalam sebuah *forbidden list* dan pengecekan apakah sebuah kandidat solusi tercatat dalam list tersebut merupakan cara yang tidak mudah, baik dari sisi kebutuhan memori maupun kebutuhan waktu komputasi. Jadi, *tabu list* hanya menyimpan langkah transisi (*move*) yang merupakan lawan atau kebalikan langkah yang telah digunakan dalam iterasi sebelumnya untuk bergerak dari satu solusi ke solusi berikutnya. Dengan kata lain tabu list berisi langkah-langkah yang membalikkan solusi yang baru ke solusi yang lama.

Pada tiap iterasi dipilih solusi baru yang merupakan solusi terbaik dalam *neighborhood* dan yang tidak tergolong sebagai tabu. Kualitas solusi baru ini tidak harus lebih baik dari kualitas solusi sekarang. Apabila solusi baru ini memiliki

nilai fungsi objektif lebih baik dibandingkan solusi terbaik yang telah dicapai sebelumnya, maka solusi baru ini dicatat sebagai solusi terbaik yang baru.

Sebagai tambahan dari *tabu list*, dikenal adanya *aspiration criteria*, yaitu suatu penanganan khusus terhadap *move* yang dinilai dapat menghasilkan solusi yang baik, namun move tersebut berstatus tabu. Dalam hal ini, jika move tersebut memenuhi criteria aspirasi yang telah ditetapkan sebelumnya, maka move tersebut dapat digunakan untuk membentuk solusi berikutnya (status tabunya dibatalkan).

2.4.1 Memori

2.4.1.1 *Tools*

Untuk menunjang sistematis dari tujuan *tabu search*, digunakan dua macam *tools* yaitu *adaptive memory* dan *responsive exploration*. *Adaptive memory* pada *tabu search* ini akan menuntun suatu prosedur yang mampu melakukan pencarian solusi yang diinginkan dengan lebih ekonomis dan efektif. *Responsive exploration* lebih menekankan pada tahapan tiap proses yang harus dilalui selama proses pencarian itu berlangsung, dimana pada setiap tahapan tersebut memiliki suatu variabel keputusan yang akan menuntun pada tahapan selanjutnya sampai proses pencarian dihentikan.

2.4.1.2 Sifat

Memori yang ada pada *tabu search* memiliki dua sifat, yaitu:

1. *Explicit Memory* bersifat menyimpan *complete solution* yang umumnya menghabiskan alokasi ruang memori dan waktu, sehingga untuk menghindarinya maka *complete solution* dikurangi sehingga hanya terdiri dari *elite solution* yang dikunjungi selama pencarian.
2. *Attributive Memory* bersifat menyimpan informasi tentang atribut dari solusi yang ditemukan yang mungkin dapat berubah dari satu solusi ke solusi lainnya.

2.4.2 Intensifikasi dan Diversifikasi

Dua komponen yang sangat penting dari *tabu search* adalah strategi intensifikasi dan diversifikasi. Strategi intensifikasi berdasarkan modifikasi aturan-aturan pilihan untuk memacu kombinasi pergerakan dan fitur-fitur solusi yang terbukti baik. Ini juga berarti mulainya pencarian daerah yang menarik secara lebih menyeluruh. Karena solusi elit harus dicatat untuk mencari solusi-

solusi tetangga, memori eksplisit sangat berhubungan dengan implementasi dari strategi intensifikasi. Perbedaan utama antara intensifikasi dan diversifikasi adalah bahwa selama masa intensifikasi pencarian difokuskan pada pemeriksaan untuk sebuah elit.

Strategi intensifikasi membutuhkan cara untuk mengidentifikasi suatu set solusi elit sebagai dasar untuk menggabungkan atribut-atribut yang baik menjadi solusi yang baru. Keanggotaan dalam suatu set elit sering ditentukan dengan menentukan ambang batas yang dihubungkan dengan nilai fungsi objektif dari solusi terbaik yang ditemukan selama pencarian.

2.4.3 Penyelesaian *Tabu Search*

Tabu search adalah salah satu metode yang tergabung dalam satu kelas yang disebut metaheuristik. Metode *tabu search* ini terbukti sukses dalam memecahkan permasalahan kombinatorial terkait dengan masalah optimasi. Dasar dari TS metaheuristik adalah dengan menggunakan strategi pengawalan yang agresif untuk memotong prosedur pencarian local untuk membawa keluar eksplorasi dari himpunan solusi dalam rangka menghindari keterjebakan dalam *local optima*. Ketika *local optima* ditemui, strategi agresif bergerak ke solusi terbaik di setiap tetangga walaupun mungkin akan mengakibatkan penurunan dalam nilai tujuan. Untuk menghindari pencarian ke tempat yang baru saja diperoleh, TS menggunakan struktur memori untuk menyimpan atribut dari solusi yang diterima, yang baru saja ditemui dalam *tabu list*. Atribut yang disimpan dalam *tabu list* disebut *tabu-active*, dan solusi-solusi yang memiliki elemen *tabu active* dikatakan sebagai tabu. Sebuah atribut tetap *tabu active* selama durasi tt, dikenal sebagai tabu tenure. Algoritma TS melanjutkan pencariannya sampai iterasi tertentu sebelum ini diakhiri.

TS metaheuristik membutuhkan:

- Solusi awal
- Mekanisme pembentukan solusi tetangga
- Data struktur manajemen
- Set komponen untuk algoritma TS

2.4.3.1 Solusi Awal

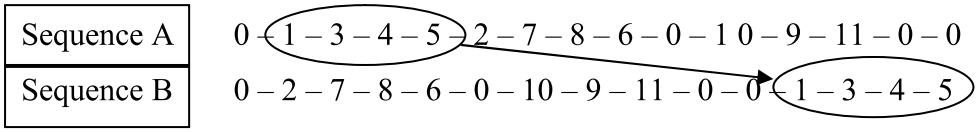
Solusi awal yang digunakan untuk algoritma TS adalah solusi dari kelas yang lebih rendah dalam hal ini dapat melalui pendekatan eksak, *random*, ataupun metode heuristic. Prosedur VRP digunakan untuk mendapatkan solusi awal secara cepat dan selanjutnya diperbaiki menggunakan algoritma TS. Untuk setiap kendaraan tipe *t,* beberapa solusi dihasilkan. Kemudian dipilih solusi awal terbaik dengan mempertimbangkan semua jenis kendaraan yang ada. Kendaraan dengan karakteristik yang berbeda-beda diatur untuk sekelompok rute terbaik dalam solusi awal untuk meminimalkan biaya tetap total dan biaya variabel namun tetap dapat memenuhi permintaan.

2.4.3.2 Mekanisme Pembentukan Solusi Tetangga

Anggap S= $\{R_1, , , ,R_2, , , ,R_v\}$ adalah merupakan solusi VRP dimana *v* adalah jumlah total kendaraan. Mekanisme pembentukan solusi tetangga menentukan suatu set operator yang dapat diaplikasikan pada S untuk menghasilkan *move* ke solusi S' yang lain sebagai tetangga S, N(S). Untuk implementasinya diadopsi mekanisme *λ-interchange* oleh Osman untuk masalah *routing* dan *grouping*. Misalkan sepasang rute (R_p, R_q) dalam S, *λ-interchange* yang dapat digunakan adalah:

- 1-*interchange mechanism*

  Proses pada mekanisme ini ada dua yaitu proses pindah (*shift*) dan proses tukar (*exchange*). Proses pindah berdasarkan operator (1,0) dan (0,1), sedangkan proses tukar berdasarkan operator (1,1). Operator pindah (1,0) memindahkan satu konsumen dari rute R_p ke R_q sedangkan operator pindah (0,1) memindahkan satu konsumen dari rute R_q ke R_p. Operator (1,1) menukar masing-masing satu konsumen dari rute R_p ke R_q secara serentak.

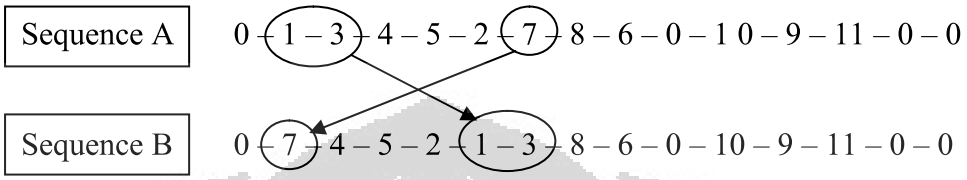


Gambar 2.3 *Move* pada 1-*interchange mechanism*

- 2-*Consecutive node interchange mechanism*

  Mekanisme ini menggunakan semua operator pada 1-*interchange mechanism* ditambah operator pindah (2,0) dan (0,2) dan operator tukar

(2,1), (1,2) dan (2,2). Cara kerja operator-operator tersebut sama dengan cara kerja pada 1-*interchange mechanism*, hanya saja jumlah konsumen yang dipindahkan atau dipertukarkan berbeda. Pada operator tambahan tadi yang berpindah atau dipertukarkan sebanyak dua konsumen, yang mana dua konsumen tersebut berurutan atau tidak berurutan.



Gambar 2.4 *Insert move* pada 2-*Consecutive mode interchange mechanism*

2.4.3.3 Data Struktur Manajemen

Untuk meningkatkan kecepatan heuristic dikembangkan sebuah manajemen struktur data untuk merekam jarak masing-masing *route delivery*, permintaan dan kendaraan yang dialokasikan. Ketika sebuah solusi tetangga dibentuk dengan satu *move* tetangga, hanya dua dari rute yang terlibat yang dihitung ulang.

2.4.3.4 Komponen *Tabu Search*

- ✓ *Tabu list*

  Tabu list adalah memori jangka pendek yang digunakan untuk menyimpan beberapa atribut dari *move* yang sedang dilakukan untuk menentukan status tabu di *move* selanjutnya.

- ✓ *Tabu restriction*

  Tabu restriction adalah criteria untuk menentukan status *move* yang tabu. Ada beberapa arahan yang dapat digunakan untuk membuat *tabu restriction* ini. Misalnya untuk mekanisme 2-*consecutive node interchange*, *move* dikatakan tabu jika konsumen *i* dan *j* dan konsumen *l* dan *s* kembali ke rute semulanya.

- ✓ *Aspiration criteria*

  Aspiration criteria mengesampingkan status tabu dari sebuah *move* yang merupakan *tabu-active* dan membuat *move* tersebut diizinkan jika *move* tersebut menghasilkan solusi terbaik baru.

- ✓ *Stopping rule*

  Merupakan aturan atau criteria untuk menghentikan seluruh proses *tabu search.*

- ✓ Skema *tabu tenure*

  Tabu tenure adalah durasi suatu atribut dikatakan *tabu active*, setelah melewati nilai *tabu tenure* ini maka atribut tersebut tidak lagi *tabu active*. Skema yang digunakan untuk melakukan kontrol *tabu tenure* adalah:

  - Skema *Fixed TS* (*F-tabu*)

    Merupakan cara yang paling pertama dikenal dan digunakan yaitu menerapkan nilai *tt* selama proses.

  - Skema *Robust TS* (*Rb-tabu*)

    Menggunakan nilai *tt* secara acak pada kisaran tertentu. Selama pencarian nilai *tt* secara periodic berubah setelah melakukan sebanyak *m* iterasi. Nilai *m* pun diambil secara acak.

  - Skema *Periodic TS* (*P-tabu*)

    Menerapkan nilai *tt* yang berubah secara periodic dari nilai yang kecil, sedang hingga besar. Perubahan tersebut dilakukan setelah melakukan iterasi sebanyak *m* iterasi.

  - Skema *Reversed deterministic TS* (*Rd-tabu*)

    Merupakan skema baru yang diusulkan dengan mekanisme mengubah dan membalikkan nilai *tt* selama pencarian. Skema ini diawali dari penggunaan beberapa nilai *tt* yang tetap, yang sebelumnya telah ditentukan, dimana *tt=n/p* dan nilai *p* diambil dari kisaran 2 hingga 7. Skema Rd-tabu secara dinamis mengubah nilai *tt* selama proses pencarian. Nilai *tt* diambil dari perhitungan dengan membagi jumlah konsumen (n) dengan suatu nilai *p*, setelah iterasi sejumlah *m*, yaitu *tt=n/p* dan *m=T_itr/n x p*, *T_itr* adalah jumlah total iterasi yang harus diselesaikan selama pencarian. Pada awal pencarian, nilai *p* adalah 9 lalu diturunkan satu setelah melakukan iterasi sejumlah *m*. setelah mencapai nilai 1, lalu nilai *p* diulang lagi dari 9, proses berlanjut hingga pencarian dihentikan. Dengan cara ini kita dapat melakukan kontrol terhadap proses diversifikasi dengan memberikan nilai *p* yang kecil dan

melakukan kontrol terhadap proses intensifikasi dengan memberikan nilai *p* yang besar.

2.4.4 Mekanisme Algoritma *Tabu Search*

Berikut adalah tahapan-tahapan umum dalam proses algoritma *tabu search.*

1. **Menentukan solusi awal**, diperoleh dari hasil pengolahan data dengan metode *random.* Acuan awal ini digunakan sebagai pembanding ketika proses *tabu search* dimulai. Pada tahap ini ditentukan S_{best} = S, C_itr = 0 *(current iteration counter).*
2. **Inisialisasi *tabu search***, menentukan skema *tabu tenure* seta nilai untuk tiap parameternya. Kemudian menentukan jumlah total iterasi *T_itr* dan *B_itr* = 0 (*best iteration counter*).
3. **Melakukan iterasi,**

   Lakukan *move* untuk membuat solusi tetangga, ada beberapa macam *move* yang dapat dipilih selama proses pencarian berjalan yaitu:

   a. *Local search* yang terdiri dari dua macam yaitu:
      i. *Insertion*: memilih secara acak satu bagian struktur untuk dipindah ke bagian yang lain.
      ii. *Swap*: memilih secara acak dua bagian struktur untuk selanjutnya ditukar posisinya.

   b. Neighborhood Search

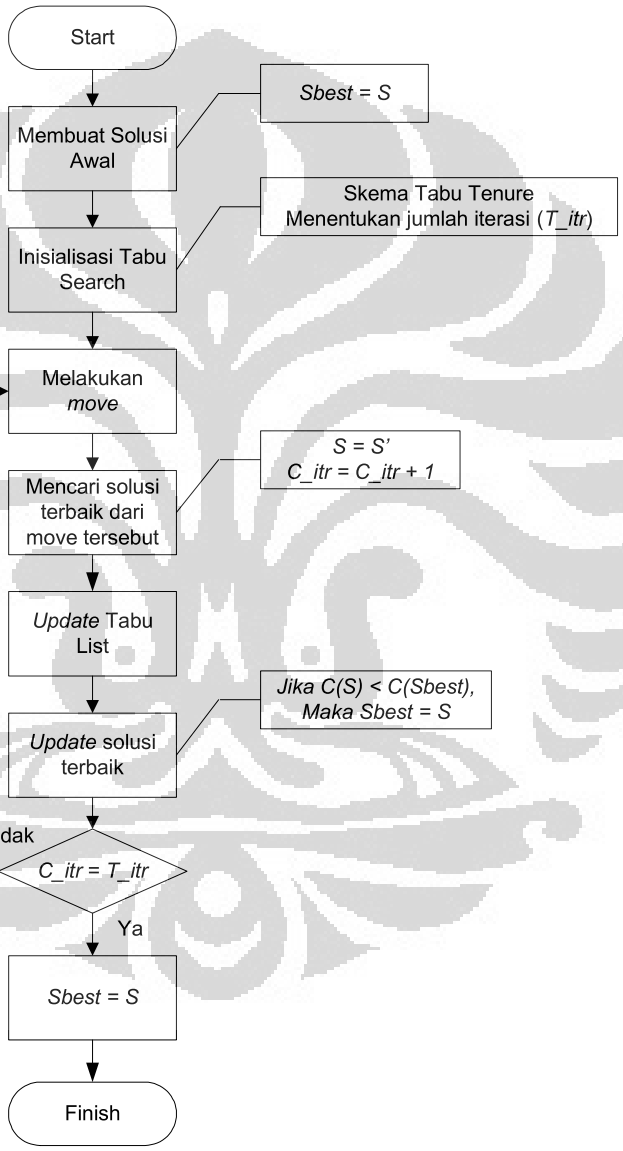
   Pada pencarian dengan teknik ini, setiap kemungkinan atribut dari struktur dapat dipindah-pindah. Permutasi *n-change neighborhood* mengambil *n* elemen dari matriks solusi.

   Kemudian menentukan solusi saat ini (*current solution*) S menjadi S', *C_itr=C_itr+1.*
4. Untuk menghindari terulangnya langkah yang diambil, maka dilakukan *tabu test. Tabu test* memanfaatkan *tabu list* yang sudah ada. Tujuan sebenarnya dari *tabu list* bukan untuk mencegah terulangnya langkah-langkah yang telah diambil, teapi lebih kepada agar tidak mundur untuk mencegah perulangan, daftar solusi yang telah dicapai disimpan dalam sebuah table.
5. *Alternative move* yang lolos *tabu test* harus melewati *aspiration test*. Jika tidak dapat melewati *aspiration test*, maka tidak akan diteruskan ke iterasi

berikutnya. Jika alternative move mempunyai aspiration criteria yang lebih baik daripada *aspiration threshold*, maka dilakukan eksekusi terhadap *alternative move* tersebut dan memperbaharui memori yang tidak relevan.

6. Jika aturan pemberhentian sudah memenuhi syarat pemberhentian, maka pencarian berhenti.

Mekanisme pengerjaan *tabu search* dapat digambarkan melalui *flowchart* berikut:



Gambar 2.5 *Flowchart Tabu Search*

BAB 3

DATA DAN PENGOLAHAN DATA

3.1 Pendahuluan

Pada bab ini akan dibahas mengenai data yang diperlukan dalam penyelesaian penelitian ini beserta dengan model matematikanya. Data yang digunakan dalam penelitian ini merupakan data realisasi yang didapatkan langsung dari fungsi perkapalan, distribusi dan fungsi persediaan di depot tempat dilaksanakan penelitian. Data tersebut merupakan data bulan Januari 2009 sampai dengan Desember 2010, sedangkan model matematika diverifikasi dan divalidasi dengan kondisi aktual.

Data yang digunakan dalam penyusunan penelitian tugas akhir ini berupa:

- Data pelabuhan
- Data kapal
- Data realisasi operasi kapal
- Data biaya; *bunker price* dan *port cost*
- *Production* dan *consumption rate* tiap bulan

3.2 Profil Perusahaan

Penelitian ini dilakukan di PT Pertamina (Persero). Perusahaan ini adalah perusahaan minyak dan gas bumi yang dimiliki pemerintah Indonesia (*National Oil Company*) yang berdiri sejak tanggal 10 Desember 1957. Pada tahun 2001 diterbitkan UU Migas No.22 tahun 2001 yang akhirnya mengantar PT Pertamina menjadi perusahaan perseroan terbatas.

Adapun tujuan perusahaan perseroan adalah untuk:

1. Mengusahakan keuntungan berdasarkan prinsip pengelolaan perseroan secara efektif dan efisien.
2. Memberikan kontribusi dalam meningkatkan kegiatan ekonomi utnuk kesejahteraan dan kemakmuran rakyat.

Untuk mencapai maksud dan tujuan tersebut, perseroan melaksanakan kegiatan usaha sebagai berikut:

1. Menyelenggarakan usaha di bidang minyak dan gas bumi beserta hasil olahan dan turunannya.
2. Menyelenggarakan kegiatan usaha di bidang panas bumi yang ada pada saat pendiriannya, termasuk Pembangkit Listrik Tenaga Panas Bumi (PLTP) yang telah mencapai tahap akhir negosiasi dan berhasil menjadi milik perseroan.
3. Melaksanakan pengusahaan dan pemasaran *Liquified Natural Gas* (LNG) dan produk lain yang dihasilakn dari kilang LNG.
4. Menyelenggarakan kegiatan usaha lain yang terkait atau menunjang kegiatan usaha sebagaimana dimaksud dalam nomor 1, 2 dan 3.

Pengusahaan gas pada perusahaan ini dilakukan oleh Unit Gas Domestik. Unit Gas Domestik merupakan salah satu unit bisnis di Pertamina yang memasarkan LPG dan produk-produk gas lainnya di Indonesia. Sejak tahun 1968, Unit Gas Domestik telah berkomitmen untuk melayani seluruh masyarakat Indonesia dengan menyediakan LPG sebagai bahan baku dan bahan bakar Industri, Rumah Tangga, dan Komersial dengan menggunakan *brand* "Elpiji". Akhir-akhir ini, Elpiji menjadi lebih dikenal dan dekat dengan masyarakat karena adanya program Pemerintah untuk mengkonversi Minyak Tanah dengan Elpiji, yang ternyata telah terbukti lebih ekonomis, efisien dan ramah lingkungan.

Sesuai dengan ketentuan dalam Undang-Undang Migas baru, perusahaan ini tidak lagi menjadi satu-satunya perusahaan yang memonopoli industry Migas dimana kegiatan usaha minyak dan gas bumi diserahkan kepada mekanisme pasar.

3.3 Sistem *Scheduling* Kapal LPG

PT Pertamina (Persero) melayani kebutuhan LPG untuk seluruh nusantara melalui kapal dalam pendistribusiannya yang dikendalikan dengan sistem koordinasi dengan fungsi-fungsi terkait, antara lain ISC (*Integrated Supply Chain*), perkapalan dan S&D, dimana terdapat perwakilan setiap fungsi dalam koordinasi tersebut. Pergerakan kapal ditentukan dari hasil koordinasi. Setiap bulan dilakukan rapat bulanan dari seluruh Indonesia guna evaluasi periode

sebelumnya dan menganalisa rencana pengadaan muatan oleh ISC berdasarkan kebutuhan dari fungsi S&D. Selanjutnya fungsi perkapalan akan mengatur jadwal kapal guna menjembatani persediaan dan permintaan yang disebut *master program.* Dari sini ditemukan kendala jarak dan penjadwalan distribusi masih belum optimal. Penjadwalan kapal sejauh ini hanya berdasarkan pada nilai permintaan saat ini saja dan intuisi dari perencana penjadwalan kapal. Perusahaan hanya melakukan estimasi kebutuhan kapal secara *aggregate* yaitu dengan membandingkan total ECC (*Effective Cargo Capacity*) pada keperluan tonase dan total ECC pada *tanker available*.

3.4 Data

Data pelabuhan, data kapal, dan data operasional kapal merupakan data utama dalam penelitian ini. Data-data ini dijelaskan sebagai berikut:

3.4.1 Pelabuhan

Dari gambaran Bab I terdapat delapan pelabuhan muat LPG, akan tetapi hanya empat pelabuhan muat yang dipakai dalam penelitian ini karena hanya LPG dari empat pelabuhan ini yang memakai *transshipment* dahulu sebelum sampai ke *end depot.* Empat (4) pelabuhan muat ini adalah Balanak (BLN), Jabung (TJB), Bontang (BON) dan Import yang hanya bisa di Teluk Semangka (TLS IMP). Sedangkan tiga (3) pelabuhan bongkar dalam penelitian ini adalah tempat kapal VLGC sebagai *floating storage* yaitu Teluk Semangka (TLS), Balongan (BAL) dan Kalbut Situbondo (XPN). Gambaran posisi pelabuhan muat dan bongkar dapat dilihat pada gambar berikut ini.

Gambar 3.1 Posisi Pelabuhan

Sumber: PT Pertamina (Persero), Operasi Perkapalan

Tiga pelabuhan muat (*loading*) memiliki kendala inventori yang harus dijaga yaitu BLN, TJB, dan BON. Sedangkan pelabuhan TLS (Import) tidak memiliki kendala inventori karena kapasitas tangki pelabuhan dianggap tidak terbatas (selalu terdapat muatan). Perbedaan pelabuhan *loading* dan *unloading* dapat dilihat dari notasinya, jika ganjil adalah pelabuhan *loading* dan genap adalah pelabuhan *unloading.* Himpunan pelabuhan dapat dilihat pada tabel berikut:

Tabel 3.1 Himpunan Pelabuhan

Pelabuhan	Notasi	Keterangan
BLN	n_1	Pelabuhan muat (*loading*)
TJB	n_3	Pelabuhan muat (*loading*)
BON	n_5	Pelabuhan muat (*loading*)
TLS (IMPORT)	n_7	Pelabuhan muat (*loading*)
TLS	n_2	Pelabuhan bongkar (*unloading*)
BAL	n_4	Pelabuhan bongkar (*unloading*)
XPN	n_6	Pelabuhan bongkar (*unloading*)

Kapal akan muat di 4 pelabuhan muat dan akan kembali ke 3 pelabuhan bongkar sebagai *floating storage*. Kemudian akan datang kapal *shuttle* yang akan sandar di kapal untuk muat dan selanjutnya muatan akan dibawa ke *end depot.*

- Pelabuhan Balanak (BLN)

  Pelabuhan ini berada pada posisi 05^0 01’ 28” LU/ 105^0 57’ 02” BT. Pada pelabuhan ini terdapat dua *wellhead platforms* sebagai unit eksplorasi, satu unit *Floating Production Storage and Offloading unit* (FPSO) sebagai unit pengolahan dan *storage,* satu unit *Catenaries Anchor Leg Mooring* (CALM), satu unit *Crude Oil Offloading Buoy* (OOB) dan satu *unit Floating Storage and Offloading* (FSO) sebagai *storage.* Kapasitas produksi terminal ini untuk LPG adalah 14 ribu barrel propane per hari dan 10 ribu barrel butane per hari untuk diolah menjadi LPG.

- Pelabuhan Jabung (TJB)

  Kapal VLGC yang akan muat di pelabuhan ini akan berlabuh pada posisi 00^0 55’ 05.1” LU/ 104^0 04’ 02” 59.7 BT. Secara keseluruhan produksi gas di terminal ini adalah 8500 barel propane per hari dan 5200 barel butane per hari, sehingga digabungkan produksi gas adalah 20 ribu barel per hari.

- Pelabuhan Bontang (BON)

  Kapal VLGC yang akan muat di pelabuhan ini akan berlabuh pada posisi 00^0 05' 51" LU/ 117^0 28' 47" BT. Secara keseluruhan total kapasitas tangki LPG adalah 200,000 m^3 per hari.

- STS (*Ship to Ship*) *Floating Storage*

  Ketiga lokasi STS ditempatkan berdasarkan kondisi lingkungan yang memungkinkan sebagai *floating storage*, dimana pada daerah ini telah terdapat fasilitas sebagi pelabuhan.

Berikut merupakan tingkat persediaan dalam interval minimum dan maksimum yang tergantung kapasitas tangki muatan tiap pelabuhan.

Table 3.2 Kapasitas Pelabuhan

Pelabuhan (N)		Total Capacity (m^3)	Contracted Capacity (m^3)	Upper Tank Cap (S_{MXi}) (mt)	Lower Tank Cap (S_{MNi}) (mt)
BLN	n_1	80025	80025	44064.52	235.68
TJB	n_3	83070	83070	41589	244.64
BON	n_5	200000	82000	47332.04	241.5
TLS (IMPORT)	n_7	NA	NA	47000	NA

Khusus pelabuhan *unloading,* kapasitas tangki tergantung dari kapal yang masuk pada pelabuhan tersebut.

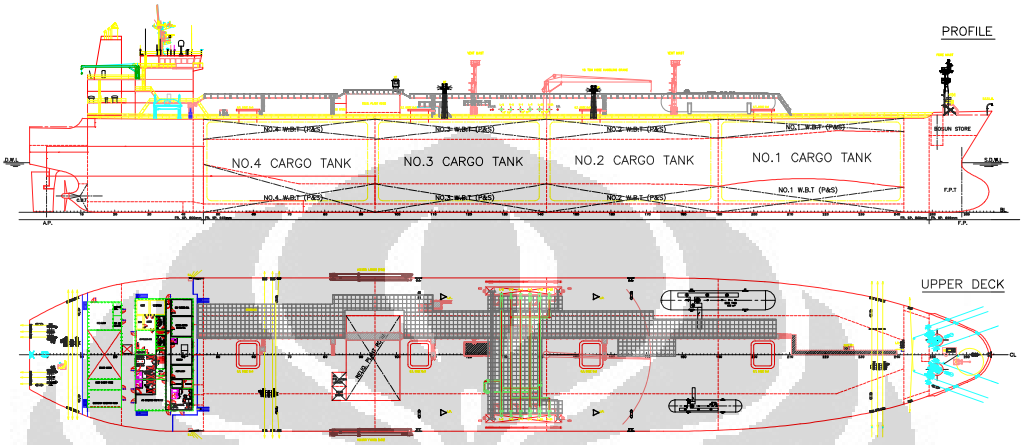
Pada tangki pelabuhan *loading* terdapat *soft inventory alarm* yang merupakan *soft inventory interval,* dimana level inventori muatan berada diantara *upper alarm* dan kapasitas tangki tertinggi (*upper tank capacity*). Jika muatan pada pelabuhan *loading* berada pada interval ini (> *upper alarm*), maka akan dikenakan biaya pinalti setiap unit muatan.

Table 3.3 *Soft Inventory Alarm*

Pelabuhan		Upper Tank Cap (S_{MXi})	Upper Alarm (A_{MXi})
BLN	n_1	44064.52	38061.294
TJB	n_3	41589	39509.55
BON	n_5	47332.04	44965.438
TLS (IMPORT)	n_7	Unlimited	NA

3.4.2 Kapal

Kapal yang digunakan perusahaan ini adalah kapal dengan sistem sewa jangka panjang dan tidak ada perubahan komposisi selama periode perencanaan. Tipe kapal VLGC (*Very Large Gas Carrier*) yang digunakan memiliki kapasitas dan karakter kapal yang berbeda-beda.



Gambar 3.2 Kapal VLGC

Sumber: PT Pertamina (Persero), Koordinator Pembangunan Kapal

Kapal ini digunakan untuk mengambil muatan di pelabuhan muat dan sekaligus sebagai *storage* di pelabuhan bongkar. Kapal dirancang agar bisa menjaga muatan tetap pada suhu terbatas yang diinginkan dan menjaga struktur dari dinginnya muatan. Perancangan kapal ini telah memenuhi aturan dari *International Maritime Organization* (IMO) melalui *International Gas Carrier* (IGC) *Code.* Seluruh kapal VLGC memiliki *vapour pressure design* tidak lebih dari 0.7 bar dengan tekanan yang hampir mendekati tekanan lingkungan. Suhu muatan kapal ini selalu di bawah *boiling point temperature* (dipertahankan pada posisi -50^0). Table berikut memperlihatkan himpunan kapal dan karakter dari masing-masing kapal VLGC:

Table 3.4 Himpunan Kapal

Kapal (V)	Notasi
BCH	v_1
MVT	v_2
MHS	v_3
BCL	v_4
GKM	v_5

Tabel 3.5 Karakteristik Kapal
Sumber: PT Pertamina (Persero), Operasi Perkapalan (telah diolah)

Karakteristik	Kapal				
	BCH	MVT	MHS	BCL	GKM
Tahun	1992	2008	2008	1992	1995
LOA (m)	223.99	225.48	225.49	223.99	230
LBP (m)	212.00	215	217.57	212.0	219
Breadth (m)	36.00	36.6	36.63	36.00	36.60
Depth (m)	21.80	22	22.039	21.80	20.40
Draft (m)	12.41	12.574	12.574	12.42	10.836
No. 1 tank					
- 98 % (m^3)	15.250	17542.3	17534.769	15256	17151.404
- Butane 0.596,-2^{0}C (mt)	9`.083	10.473	10301.5	9086	10454.00
-Propane, 0.581,-41.5^{0}C (mt)	8.843	10.175	10106.5	8846	9926.00
No. 2 tank					
- 98 % (m^3)	20570	21393.6	21417.866	20571	19972.064
- Butane 0.596,-2^{0}C (mt)	12251	12772	12582.8	12252	12172.00
- Propane, 0.581,-41.5^{0}C (mt)	11927	12408	12344.81	11928	11558.00
No. 3 tank					
- 98 % (m^3)	20623	21393	21403.987	20625	19968.448
- Butane 0.596,-2^{0}C (mt)	12282	12772	12574.6	12284	12170.00
- Propane, 0.581,-41.5^{0}C (mt)	11957	12408	12336.8	11959	11556.00
No. 4 tank					
- 98 % (m^3)	20525	20278,7	20300,229	20525	19817.56
- Butane 0.596,-2^{0}C (mt)	12224	12106	11925.8	12224	12078.00
- Propane, 0.581,-41.5^{0}C (mt)	11901	11762	11700.3	11901	11468.00
Cargo Tank Capacity Total					
- 98 % (m^3)	76968	80607,6	80656.851	76977	76909.476
- Butane 0.596,-2^{0}C (mt)	45839	48121,74	47384	45845	46974.00
- Propane, 0.581,-41.5^{0}C (mt)	44628	46752,41	46487	44633	44508,00
Cargo pumps (no x m^3/jam)	8 x 530	8 x 600	8 x 600	8 x 530	8 x 530
Discharging Cap. (mt/jam)	4240	4800	4800	4240	4240
Discharging Time (jam)	Abt. 20	Abt. 18	Abt. 18	Abt. 20	Abt. 20
Loading Capacity (mt/jam)	4200	3050	3050	4200	4200
Loading Time (jam)	Abt. 20	Abt. 18	Abt. 18	Abt. 20	Abt. 20

Karakteristik tiap kapal berbeda-beda baik dari segi performance maupun dari segi biaya operasi. Kelima kapal tersebut dapat masuk ke seluruh pelabuhan.

Berikut adalah besar kapasitas tangki muatan tiap kapal dengan kapasitas kapal yang berbeda-beda. Pada penelitian ini diasumsikan bahwa *unpumpable* muatan (bagian dasar tangki yang tidak bisa dipompa lagi) dapat kosong, walaupun dalam kenyataannya selalu ada sisa muatan untuk pemeliharaan suhu kargo tangki.

Table 3.6 Kapasitas Kapal

Kapal (V)	**Volume (m^3)**	**Kapasitas 100% (MT)**	***CAPv* (*mt*)**
BCH (CAP_1)	78539.00	46259.47	45334.28
MVT (CAP_2)	82252.70	48446.84	47477.90
MHS (CAP_3)	82302.91	48476.41	47506.89
BCL (CAP_4)	78549.00	46265.36	45340.05
GKM (CAP_5)	78479.06	46224.17	45299.68

3.4.3 Realisasi Pengoperasian Kapal

Data operasi kapal mulai Januari 2009 s/d Desember 2010 diperoleh dari data VMIS (*Vessel Management Information System*) yang meliputi data pergerakan kapal dan muatan, yang merupakan dasar perhitungan dalam penelitian ini, termasuk: *port time* tiap pelabuhan, *sea time* tiap kapal tiap rute, *fuel oil consumption* tiap kapal tiap rute dan *cargo consumption rate* tiap pelabuhan bongkar.

a. Waktu *un(loading)* muatan

Adalah waktu yang diperlukan kapal ketika berada di *un(loading) port* untuk memuat atau membongkar muatan di pelabuhan pemberangkatan dengan satuan hari per metric ton. Pada saat memompa muatan ke dalam kapal, yang digunakan adalah fasilitas pompa pelabuhan yang tergantung dari jumlah muatan. Sedangkan pada saat membongkar muatan dari kapal VLGC, yang digunakan adalah fasilitas pompa kapal VLGC yang sekaligus berfungsi sebagai *storage* untuk memompa muatan ke dalam kapal *shuttle.*

Table 3.7 Waktu *Un(Loading*) Muatan

Notasi	Pelabuhan	T_{Qi} (*hari/mt*)
n_1	BLN	0.000059
n_3	TJB	0.000050
n_5	BON	0.000047
n_7	TLS (IMPORT)	0.0000145
n_2	TLS	0.000137153
n_4	BAL	0.000497314
n_6	XPN	0.000546969

b. *Sea Time*

Sea Time adalah waktu yang dibutuhkan tiap-tiap kapal untuk berlayar dari pelabuhan pemberangkatan ke pelabuhan tujuan, mulai dari waktu berangkat actual di pelabuhan sebelumnya sampai dengan waktu tiba di pelabuhan berikutnya. Antara waktu dari pelabuhan A ke B dan waktu dari B ke A berbeda, dapat dilihat pada table berikut:

Table 3.8 *Sea Time*

Route	*Sea time* (jam)
BLN - TLS	43.025
TLS - BLN	45.49
BLN - BAL	47.625
BAL - BLN	49.44
BLN - XPN	68.2
XPN - BLN	68.13
TJB - TLS	41.35
TLS - TJB	41.613
TJB - BAL	35.8
BAL - TJB	33.75
TJB - XPN	55.85
XPN - TJB	55.35
BON - TLS	76.45
TLS - BON	72.62
BON - BAL	54.43
BAL - BON	54.43
BON - XPN	43.8
XPN - BON	43.16
TLS (IMPORT) - BAL	22.3
BAL - TLS (IMPORT)	19.32
XPN - TLS (IMPORT)	51.64
TLS (IMPORT) - XPN	51.64
TLS (IMPORT) - TLS	0.9
TLS - TLS (IMPORT)	0.9

c. Tingkat produksi/konsumsi

Tingkat produksi bernilai positif jika berada pada pelabuhan muat (produksi) dan bernilai negatif pada pelabuhan bongkar (konsumsi) dengan satuan mt per hari. Table tingkat produksi/konsumsi dapat dilihat sebagai berikut:

Table 3.9 *Production/Consumption Rate*

Port	**Produstion/Consumption Rate**		
	(Barrel/hari)	(mt/hari)	(mt/jam)
BLN	12000	1032	35.83
TJB	6850	589	18.63
BON	13762	1184	49.32
TLS (IMPORT)	NA	NA	NA
TLS	-	-7291	-303.798
BAL	-	-2011	-83.783
XPN	-	-1828	-76.177

3.4.4 Biaya Kapal

Biaya tetap kapal (*fixed cost*) yang merupakan pengeluaran tetap berupa *charter rate* dan *port charges* tidak dimasukkan dalam penelitian ini. Biaya variabel (*variable cost*) yang dihitung dalam penelitian ini adalah *bunker consumption* dengan harga *bunker* yang ditetapkan sebesar USD 532 per metric ton dan biaya pelabuhan (*port cost*).

a. Biaya Transportasi (*transportation cost*)

Biaya transportasi ini diperoleh dari waktu *sea time* dikali dengan *fuel oil consumption* tiap satuan waktu kemudian dikali dengan harga *bunker* per satuan berat *bunker* (US$ 532 per metric ton).

Table 3.10 *Transportation Cost* (US$)

	Kapal	BLN	TJB	BON	TLS (IMP)	TLS	BAL	XPN
BLN	V_1					47961.51	53089.3	76024.99
	V_2					45974.23	50889.54	72874.89
	V_3					48953.12	54186.92	77596.81
	V_4					52223.44	57806.89	82780.67
	V_5					48687.19	53892.56	77175.28
TJB	V_1					46094.33	39907.55	62258
	V_2					44184.41	38253.98	59678.34
	V_3					47047.33	40732.63	63545.18
	V_4					50190.34	43453.79	67790.33
	V_5					46791.76	40511.36	63199.99
BON	V_1					85221.56	60675.08	48825.43
	V_2					81690.4	58161	46802.35
	V_3					86983.52	61929.53	49834.9
	V_4					92794.47	66066.75	53164.13
	V_5					86511	61593.11	49564.18
TLS (IMPORT)	V_1					1003.26	24858.61	57568.15
	V_2					961.69	23828.59	55182.81
	V_3					1024	25372.56	58758.37
	V_4					1092.41	27067.58	62683.73
	V_5					1018.44	25234.73	58439.18
TLS	V_1	48772.7	44618.66	77866.19	965.02			
	V_2	48397.81	44275.7	77267.68	957.6			
	V_3	49528.96	45310.5	79073.57	979.98			
	V_4	52739.37	48247.8	84199.03	1043.5			
	V_5	49784.01	45543.83	79480.76	985.03			

Sea Times	Kapal	BLN	TJB	BON	TLS (IMP)	TLS	BAL	XPN
BAL	V_1	53011.63	36188.16	58362.12	20714.69			
	V_2	52604.16	35910	57193.52	20555.47			
	V_3	53833.62	36749.28	59267.07	21035.89			
	V_4	57323.05	39131.33	63108.69	22399.41			
	V_5	54110.83	36938.52	59572.26	21144.21			
XPN	V_1	73055.4	59348.58	46277.95	55373.63			
	V_2	72493.87	58892.4	45922.24	54948			
	V_3	74188.18	60268.82	46995.53	56232.24			
	V_4	78996.98	64175.38	50041.72	59877.15			
	V_5	74570.21	60579.18	47237.53	56521.8			

Terlihat bahwa tiap-tiap kapal dengan rute yang sama memiliki biaya yang berbeda-beda. Biaya dari pelabuhan A ke B maupun dari B ke A berbeda, hal ini terkait dengan jumlah *fuel oil consumption* yang digunakan pada rute tersebut juga berbeda.

b. Biaya FOC pada pelabuhan

Dari table 3.7 (waktu *un(loading)* muatan) dikali dengan harga *bunker,* diperoleh biaya bahan bakar yang tergantung dari muatan yang akan dimuat atau dibongkar.

Table 3.11 *Fuel Oil Cost* pada *un(loading) times*

Pelabuhan	Time (hari/mt cargo)	FOC (US$ per mt cargo)				
		BCH	MVT	MHS	BCL	GKM
BLN	0.000059	0.34268	0.31388	0.287465	0.315649	0.329497
TJB	0.000050	0.290407	0.266	0.243614	0.267499	0.279235
BON	0.000047	0.272982	0.25004	0.228998	0.251449	0.26248
TLS (IMP)	0.0000145	0.470459	0.43092	0.394655	0.433348	0.45236
TLS	0.000137153	0.525782	0.656689	0.542594	0.444665	0.59602
BAL	0.000497314	1.906473	2.381139	1.967435	1.612348	2.161155
BON	0.000546969	2.096828	2.618888	2.163876	1.773335	2.376939

c. Biaya Pelabuhan

Biaya pelabuhan adalah biaya yang dibebankan kepada pihak kapal oleh *port authority* karena pemakaian jasa dan fasilitas pelabuhan. Biaya pelabuhan yang dibebankan kepada tiap kapal berbeda-beda.

Table 3.12 Biaya Pelabuhan

Kapal	Port Cost (US$)
BCH	6448.277
MVT	6714.618
MHS	6693.261
BCL	6448.616
GKM	6448.616

d. Biaya Pinalti

Biaya pinalti adalah biaya yang dibebankan ke penjadwalan kapal karena inventori di pelabuhan *loading* berada di atas *soft inventory alarm.* Biaya yang dikenakan adalah US$ 734 per metric ton. Perhitungan pembebanan biaya pinalti ini adalah ketika kapal datang di pelabuhan *loading,* pada saat itu kondisi inventori di pelabuhan ini berada di atas *soft inventory alarm*. Sehingga kapal akan dikenakan biaya tambahan sebesar jumlah muatan yang melebihi batas dikali dengan US$ 734 per metric ton. Perhitungan biaya ini tidak berlaku di pelabuhan *unloading*.

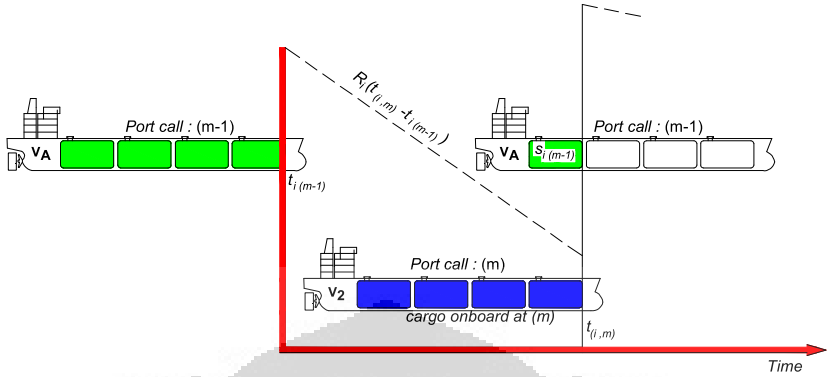
3.5 Formulasi Permasalahan dan Asumsi

Penjadwalan kapal yang terintegrasi dengan inventori akan dirumuskan sebagai masalah minimasi biaya deterministic melalui minimalisasi biaya transportasi kapal (*fuel oil consumption, port cost*) dan *penalty cost.*

Dalam deskripsi matematis permasalahan, *N* dianggap sebagai himpunan pelabuhan denga indeks *i* (pelabuhan pemberangkatan) dan *j* (pelabuhan tujuan) dan *V* sebagai himpunan kapal yang tersedia dengan indeks *v*. Setiap pelabuhan dapat dikunjungi kapal beberapa kali selama periode perencanaan yang disebut *port call*. *Mi* adalah himpunan *port call* yang mungkin pada pelabuhan *i* dan *j* dengan indeks *m* dan *n.* Secara spesifik *port call* terdiri atas pelabuhan dan *call* yang ditentukan oleh (*i,m*), dimana $i \in N$ dan $m \in Mi$. Semua kapal *v* dapat masuk ke semua pelabuhan.

Waktu yang diperlukan untuk *un(loading)* muatan setiap unitnya di pelabuhan dinotasikan sebagai T_{Qi}, sedangkan T_{ijv} merupakan waktu berlayar dari pelabuhan *i* ke *j* dengan kapal *v*. Kapasitas kapal *v* dinotasikan dengan *CAPv.* Khusus pada pelabuhan *unloading* (N_D), kapal berfungsi sebagai *mobile floating*

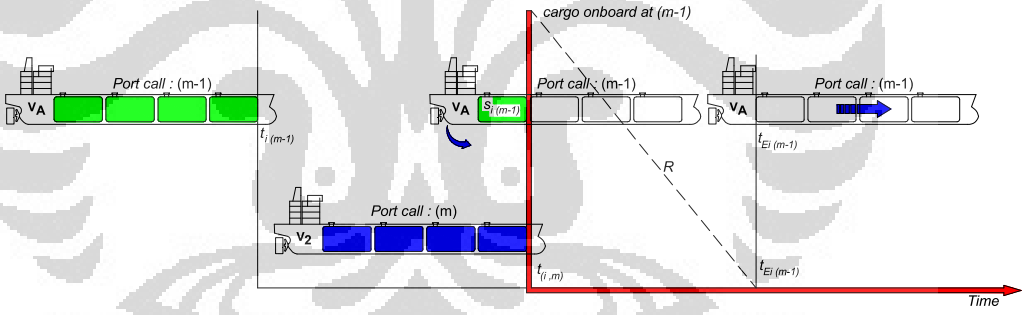
storage. Pada pelabuhan *unloading* ini *port call (i,m)* harus *overlapping* dengan *port call* sebelumnya *(i,(m-1).*



Gambar 3.3 Waktu tiba *port call* (t_{im})

Kapal berikutnya harus tiba sebelum jumlah muatan di kapal sebelumnya habis. Formula ini menjadi salah satu kendala utama.

Kapal pada *port call* sebelumnya diharuskan berangkat meninggalkan pelabuhan *unloading* $t_{Ei(m-1)}$ sampai muatan di atas kapal s_{im-1} habis, dengan ilistrasi gambar sebagai berikut:



Gambar 3.4 Waktu berangkat *port call* (t_{im})

Khusus kasus di pelabuhan *unloading*, permasalahan *overlapping time* di atas disederhanakan menjadi waktu kedatangan kapal *port call* (t_{im}) lebih besar atau sama dengan 0 hari.

Tingkat persediaan (*level of inventory*) ditetapkan pada interval yang tergantung kapasitas fisik dan sistem persediaan di pelabuhan [S_{MNi}, S_{MXi}]. Namun, guna meningkatkan hubungan penjadwalan kapal dengan inventori, digunakan batasan *soft inventory alarm level*, dimana untuk *upper alarm interval* [A_{MXi}, S_{MXi}] akan dikenakan *penalty cost* setiap unit muatan C_{Pi}^{+} jika level inventori berada pada interval tersebut pada saat kapal datang.

Tingkat produksi R_i bernilai positif jika berada pada pelabuhan *loading* (produksi), dan negatif pada pelabuhan *unloading* (konsumsi). Selanjutnya, konstanta *Ii* sama dengan 1 jika *i* adalah pelabuhan *loading* dan -1 jika *i* adalah pelabuhan *unloading*. Oleh karena itu himpunan pelabuhan (*N*) dapat dibagi menjadi satu himpunan pelabuhan *loading* dengan notasi N_p dan satu himpunan pelabuhan *unloading* dengan notasi N_D.

Total biaya variabel C_{ijv} dalam hal ini adalah biaya bahan bakar dari pelabuhan *i* ke *j* dan C_w yaitu bahan bakar di pelabuhan *i* untuk kapal *v*.

Formula *arc flow* melibatkan dua jenis variabel biner, yaitu variabel *arc flow* dengan x_{imjnv}, $v \in V$, $(i, j) \in N$, $(m,n) \in Mi$ sama dengan 1 jika kapal *v* berlayar dari *port call* (*i,m*) langsung menuju *port call* (*j,n*) dan 0 sebaliknya. Variabel *slag* y_{im}, $i \in N$, $m \in Mi$ adalah sama dengan 1 jika kapal tidak ada singgah di pelabuhan (*i,m*) dan 0 jika sebaliknya.

Selain itu, digunakan variabel kontinu. Variabel waktu t_{im}, $i \in N$, dan $m \in Mi$ merupakan waktu dimulainya pelayanan (*service*) pada *port call* (*i,m*), dalam hal ini disebut *start up time* (t_{im}) atau waktu kapal tiba. Waktu pemberangkatan kapal setelah selesai proses *un(loading)* di *port i* disebut t_{Eim}. Variable l_{imv}, $v \in V$, $i \in N$, $m \in Mi$ memberikan total muatan (*onboard*) kapal *v* setelah proses *un(loading)* selesai pada *port call* (*i,m*). Variabel s_{im} (s_{Eim}), $i \in N$, $m \in Mi$ merupakan tingkat persediaan saat layanan mulai (berakhir) di *port call* (*i,m*), sementara (a_{im}^{+}) adalah jumlah muatan pada level inventori di atas *soft inventory alarm* (A_{MXi}).

Kondisi awal yang didefenisikan meliputi posisi awal kapal (i_v,m_v), inventori awal muatan di tiap pelabuhan (IS_i) dan muatan awal kapal (Q_v). Posisi awal kapal ini merupakan salah satu kendala dimana model awal harus berada di pelabuhan (pada kondisi actual ada kemungkinan posisi kapal berada di tengah laut). Data kondisi awal kapal dapat dilihat pada table berikut:

- Jumlah awal muatan kapal (Q_v)

  Pada kondisi awal perencanaan jadwal, setiap kapal sudah berisi muatan tertentu. Kapal v_1 pada kondisi awal berisi muatan Q_1 = 43000 mt, namun masih dapat dimuati tergantung *running* model. Q_2 = 0 mt, kapal in diharuskan muat terlebih dahulu sesuai dengan kebutuhan muatan. Sedangkan Q_3, Q_4, dan

Q_5 berada di pelabuhan *unloading* dan akan bongkar sesuai kebutuhan muatan pelabuhan tersebut.

Table 3.13 Muatan Awal Kapal

Kapal (V)	Notasi	Qv (*mt*)
BCH	Q_1	43000
MVT	Q_2	0
MHS	Q_3	43000
BCL	Q_4	21000
GKM	Q_5	22000

- *Stock* awal pelabuhan (IS_i)

  Berikut adalah *stock* awal muatan di pelabuhan *i* pada awal perencanaan

Table 3.14 *Stock* Awal Pelabuhan

Pelabuhan		*ISi*
BLN	n_1	43000
TJB	n_3	22000
BON	n_5	36000
TLS (IMPORT)	n_7	Unlimited
TLS	n_2	$*V_3$
BAL	n_4	$*V_4$
XPN	n_6	$*V_5$

Catatan: *: jumlah inventori di pelabuhan *unloading* sama dengan jumlah muatan kapal yang masuk di pelabuhan tersebut.

3.5.1 Notasi dan Asumsi

Notasi yang digunakan dalam permasalahan penelitian ini adalah:

Himpunan

N	:	Himpunan pelabuhan dengan lambang *i* dan *j*.
N_P	:	Himpunan pelabuhan *loading*.
N_D	:	Himpunan pelabuhan *unloading*
V	:	Himpunan kapal dengan indeks *v*.
M_i	:	Himpunan *port calls* (jumlah kedatangan) pada pelabuhan *i* dengan lambang *m* dan *n*

S_T : Himpunan semua pelabuhan kedatangan (i,m) untuk i Є N and m Є M_i.

S_0 : Himpunan posisi awal $\{(i_v, m_v)|v \in V\}$.

S_N : Himpunan semua posisi yang mungkin ditempati oleh kapal setelah meninggalkan posisi awal (S_0)

A_v : Himpunan *feasible arc* untuk kapal *v*.

Konstanta dan Parameter

Parameter waktu

T_{Qi} : Waktu yang dibutuhkan untuk *un(load)* setiap unit muatan di pelabuhan *i*, (hari/mt)

T_{ijv} : Waktu yang dibutuhkan untuk berlayar dari pelabuhan *i* ke pelabuhan *j* dengan kapal *v* (hari).

Parameter untuk aliran *network*

i_v : Pelabuhan awal untuk kapal *v* (pemberangkatan).

m_v : Urutan kedatangan (*port call*) kapal *v* di pelabuhan i_v.

Parameter untuk *loading* dan *unloading*

J_i : Konstanta bernilai +1 jika berada di pelabuhan *loading* (N_P) dan -1 jika di pelabuhan *unloading* (N_D)

Q_v : Jumlah muatan di kapal *v* pada saat awal *planning horizon* (mt)

CAP_v : Kapasitas muat tangki kapal *v* (mt)

Parameter untuk *inventory*

IS_i : *Stock* awal muatan di pelabuhan *i*.

R_i : Rata-rata konsumsi atau produksi muatan di pelabuhan *i*.

S_{MXi} : Tingkat persediaan (*level of inventory*) dalam interval minimum dan maksium yang tergantung kapasitas phisik dan sistem inventory di pelabuhan *i* (mt).

A_{Mxi} : *Soft alarm inventory interval*, maksimum level dengan satuan mt.

T : Panjang waktu perencanaan (*Time Horizon*).

Parameter untuk biaya

C_{ijv} : Biaya kapal *v* berlayar dari pelabuhan *i* ke pelabuhan *j* (US$)

C_{Pi}^{+} : *Penalty cost* pada inventory level pada interval *soft inventory alarm* maksimum dengan satuan US$/mt.

Variabel

x_{imjnv} : *Binary Variable*, $v \in V$, $(i,j) \in N$, $(m,n) \in M_i$ sama dengan 1 jika kapal *v* berlayar dari *port call* (*i, m*) langsung menuju *port call* (*j, n*) dan 0 jika sebaliknya.

o_{imv} : *Binary Variable*, $v \in V$, $(i,j) \in N$, $(m,n) \in M_i$ sama dengan 1 jika muatan di un(loading) pada *port call* (*i, m*) di kapal v, dan 0 jika sebaliknya.

y_{im} : *Binary Variable,* $i \in N$, $m \in M_i$ adalah sama dengan 1 jika kapal tidak ada yang singgah di pelabuhan (*i,m*) dan 0 jika sebaliknya.

t_{im} : *Start up time*; merupakan waktu di mana dimulai pelayanan (*service*) sebelum proses bongkar muat pada *port call* (*i, m*).

t_{Eim} : *Departure time*; merupakan waktu berakhirnya pelayanan (*service*) pada *port call* (*i, m*).

l_{imv} : Variable jumlah muatan di kapal, $v \in V$, $i \in N$, $m \in M_i$ yaitu total muatan (*onboard*) kapal *v* ketika kapal meninggalkan pelabuhan, satuan (mt).

q_{Vimv} : variable jumlah muatan, $v \in V$, $i \in N$, $m \in M_i$ merupakan jumlah muatan yang dibongkar atau muat di pelabuhan pada *port call* (*i, m*) ketika kapal *v* berkunjung *port call* (*i, m*) dengan satuan (mt).

s_{im} (s_{Eim}) : Variable persediaan, $i \in N$, $m \in M_i$ merupakan tingkat persediaan saat layanan proses bongkar muat mulai (berakhir) sewaktu di *port call* (*i, m*) dengan satuan (mt)

a_{im}^{+} : jumlah *inventory level* diatas *soft alarm inventory* A_{MXi} dengan satuan mt.

3.5.2 Fungsi Tujuan dan Kendala

Meminimumkan total biaya operasi, *transportation cost* C_{ijv}, *FOC un(loading) times* (C_{fiv}) dan biaya *penalty* C_{Pi}^{+},).

$$\boldsymbol{Min\ Z} = \sum_{v \epsilon V} \sum_{(i,m,j,n) \epsilon\, A_v} \boldsymbol{C_{ijv}\, x_{imjnv}} + \sum_{i \in N_P} \sum_{m \in M_i} \sum_{v \in V} \boldsymbol{C_{fiv}\, o_{imv}} + \sum_{iEN} \sum_{mEMi} (C_{Pi}^{+} a_{im}^{+}) \quad (3.1)$$

dengan C_{ijv} = biaya dari port i ke j oleh kapal v, (fuel oil consumption)

C_{fiv} = biaya pelabuhan di port i, tergantung jumlah muatan yang di *un(loading)* (fuel oil consumption)

C_{Pi}^{+} = biaya penalty, karena keterlambatan kapal

x_{imjnv} = 1 , jika terdapat kapal dari i ke j

0 , sebaliknya

O_{imv} = *variable* cargo *un(loading)* pelabuhan,bernilai 1, jika di *un(loading*), dan 0 jika sebaliknya.

Fungsi integer di atas menunjukkan terhubungkan atau tidaknya titik *i* dan *j* oleh kapal *v* pada *port call* awal (*i,m*) dan *port* tujuan (*j,n*). Nilai 0 menunjukkan tidak terhubungkannya titik *i* dan *j*, sedangkan nilai 1 menunjukkan bahwa *port i* dan *port j* terhubungkan oleh kapal *v* pada periode *port call* tersebut dengan :

i = 1,2,3,4,5,6,7

v = 1,2,3,4,5

j = 1,2,3,4,5,6,7

Subject to :

- *Initial constraints*
  - *Initial position constraint*
    Setiap kapal *v* harus berangkat dari posisi awal,

$$\sum_{(j,n) \in S_N} x_{i_v m_v jnv} = 1\,, \text{untuk setiap } v \in V \quad (3.2)$$

  - *Initial ship load constraint*
    Jumlah muatan *onboard* kapal *v* saat berangkat dari posisi awal (i_v, m_v), $l_{i_v m_v v}$, harus sama dengan jumlah *onboard* awal Q_v ditambah muatan,

$q_{i_v m_v v}$, yang dimuat, $J_{i_v} = +1$, (sebaliknya di bongkar, $J_{i_v} = -1$) pada posisi awal.

$$Q_v + J_i q_{i_v m_v v} - l_{i_v m_v v} = 0, untuk\ setiap\ v \in V \tag{3.3}$$

- *Initial inventory constraint*
  Pada saat awal periode perencanaan, bagi pelabuhan yang tidak memiliki kapal maka level stok s_{i1} pada pelabuhan *i* saat kedatangan kapal pertama adalah IS_i jumlah muatan pelabuhan *i* pada awal horison perencanaan ditambah jumlah yang diproduksi (J_i = +1) atau minus jumlah yang dikonsumsi (J_i = -1) sampai t_{i1} kedatangan kapal pertama.

$$s_{i1} = IS_i + J_i R_i t_{i1}, untuk\ setiap\ i \in N \tag{3.4}$$

- Tambahan : tidak ada pergerakan kapal antar pelabuhan *loading* dan antar pelabuhan *unloading*

- *Routing Constraints*
  - *Flow conservation constraint*
    Kedatangan kapal ke *m* di pelabuhan *i* harus juga meninggalkan pelabuhan tersebut atau mengakhiri rutenya disana. Z benilai 0 jika (*i,m*) adalah posisi *intermediate* dan harus sama dengan 1 jika merupakan posisi akhir *schedule* kapal *v*.

$$\sum_{(j,n)\in S_T} x_{jnimv} - \sum_{(j,n)\in S_N} x_{imjnv} - Z_{imv} = 0\,, \quad \text{untuk setiap } (v,i,m) \in V \times S_T \tag{3.5}$$

  - *Route finishing constraint*
    Di akhir periode perencanaan, setiap kapal harus berada di pelabuhan

$$\sum_{(i,m)\in S_N} z_{imv} = 1, \text{untuk setiap } v \in V \tag{3.6}$$

  - *One time visit constraint*
    Memastikan bahwa setiap port call *(i, m)* dikunjungi paling banyak sekali, y_{im} adalah binary variable = 1 jika posisi (*i,m*) tidak dikunjungi.

$$\sum_{v\in V} \sum_{(j,n)\in S_T} x_{jnimv} + y_{im} = 1\,, \text{untuk setiap } (i,m) \in S_T \tag{3.7}$$

- *Arrival sequence constraint*
  Jika pelabuhan tidak memiliki kedatangan ke *(m-1),* maka ia tidak akan mempunyai kedatangan ke *m*, sebaliknya, jika terdapat kedatangan ke *m*, maka secara pasti terdapat kedatangan *(m-1)*

$$y_{im} - y_{i\,(m-1)} \geq 0, \text{untuk setiap } (\mathrm{i,m}) \in \mathrm{S_N} \tag{3.8}$$

- *Un(loading) Constraints*
  - *Ship load constraint*
    Jika kapal *v* berlayar dari *(i, m)* ke *(j, n)* ; yaitu $x_{imjnv} = 1$. Maka jumlah muatan *onboard*, l_{jnv} ,pada saat kapal meninggalkan *(j, n)* harus sama dengan jumlah *onboard* saat berangkat dari *(i, m)*, l_{imv}, ditambah, $J_j = +1$ jika dimuat sejumlah kargo, q_{jnv}, (sebaliknya, minus, $J_j = -1$ jika bongkar) di *(j, n)*.

$$x_{imjnv}\left[\, l_{imn} + J_j\, q_{jnv} - \, l_{jnv} \,\right] = 0, \quad \text{untuk setiap } \mathrm{v} \in \mathrm{V} \text{ dan } (\mathrm{i,m,j,n}) \in \mathrm{A_V} \tag{3.9}$$

    Karena persamaan diatas tidak linear, maka dilinearkn dengan persamaan sebagai berikut :

$$l_{imv} + J_j\, q_{jnv} - l_{jnv} + CAP_V \;\; x_{pimjnv} \leq CAP_V \quad v \in V,\;\; (i,m,j,n) \in A_v \tag{3.10}$$

$$l_{imv} + J_j\, q_{jnv} - l_{jnv} - CAP_V \;\; x_{imjnv} \geq -CAP_V \quad v \in V,\;\; (i,m,j,n) \in A_v \tag{3.11}$$

  - *Compartment capacity constraint*
    Jumlah muatan onboard kapal *v* saat berangkat dari posisi *(i, m)*, $l_{imv,}$ tidak dapat melebihi kapasitas kompartemen *CAPv* , Namun, ini akan bermakna hanya jika kapal *v* mengunjungi *(i,m)* yaitu,$\sum_{(j,n)\in S_T} x_{jnimv} = 1$, jika tidak maka quantity l_{imv} *=0*

$$l_{imv} \leq \sum_{(j,n)\in S_T} CAP_v x_{jnimv}, \quad untuk\ setiap\ v \in V\ dan\ setiap\ (i,m) \in\ S_N \tag{3.12}$$

  - *Servicing product constraint*
    memastikan bahwa kuantitas q_{imv} dimuat ke kapal *v* pada posisi *(i, m)* tidak akan melebihi kapasitas kompartemen *CAPv* kapal *v.*
    Pelabuhan *loading*

$$q_{imv} = \; CAP_v\, o_{imjnv}, \quad untuk\ setiap\ v \in V\ dan\ setiap\ (i,m) \in\ N_P\, xM_i \tag{3.13}$$

Pelabuhan *unloading*

$$q_{imv} \leq CAP_v \, o_{imjnv},$$
$$untuk\ setiap\ v \in V\ dan\ setiap\ (i,m) \in N_D\ xM_i \quad (3.14)$$

- *Time Constraints*
  - *Service time sequence constraint*

    Sangat jelas bahwa waktu kedatangan t_{im} selalu lebih besar dari waktu kedatangan sebelunya $t_{(im-1)}$.

    Untuk semua pelabuhan (N)

$$t_{im} - t_{i(m-1)} > 0 \ \ untuk\ setiap\ (i,m) \in S_N \quad (3.15)$$

    Untuk pelabuhan unlaoding (N_D)

$$t_{Ei(m-1)} - t_{im} \geq 1$$
$$untuk\ setiap\ v \in V\ dan\ setiap\ (i,m) \in N_D\ xM_i \quad (3.16)$$

  - *Route and schedule compatibility constraint*

    Jika kapal *v* berlayar dari posisi *(i, m)* ke *(j, n)* -dimana, $x_{imjnv} = 1$-maka waktu kedatangan di *(j, n)*, t_{jn}, adalah jumlah waktu keberangkatan dari *(i, m)* setelah kapal melaksanakan service, t_{Eim}, dan waktu perjalanan dari pelabuhan *i* ke pelabuhan *j* dengan kapal *v*, T_{ijv}.

$$x_{imjnv}\left[t_{Eim} + T_{ijv} - t_{jn}\right] \leq 0,$$
$$untuk\ setiap \in V dan\ tiap\ (i,m,j,n) \in A_v \quad (3.17)$$

    Kendala ini tidak linear, sehingga perlu diformulasi ulang menjadi linier setara, sebagai berikut :

$$\left[t_{Eim} + T_{ijv} - t_{jn} + 2T x_{imjnv}\right] \leq 2T, untuk\ setiap\ v \in V dan\ tiap\ (i,m,j,n) \in A_v \quad (3.18)$$

  - *Service finishing constraint*

    Pelabuhan *loading*

    Waktu keberangkatan kapal di suatu pelabuhan t_{Eim}, adalah waktu kedatangan t_{im} ditambah dengan waktu yang diperlukan untuk layanan proses loading $T_{Qi} q_{i(m-1)v}$

$$t_{Eim} = t_{im} + \sum_{v \in V} T_{Qi}\ q_{imv}\ o_{imv}\ ,$$
$$untuk\ setiap\ (i,m) \in N \quad x\ Mi \quad (3.19)$$

- *The inventories Constraints*
  - Inventory level constraint

    Stock level pada saat kapal datang s_{im} sama dengan *stock level* pada saat selesai *service* pada ketangan kapal sebelumnya $s_{Ei(m-1)}$ ditambah dengan dengan konsumsi R_i selama periode kedatangan kapal t_{im} dengan pemberangkatan kapal sebelumnya $t_{Ei(m-1)}$.

    Stock level pada saat kedatangan kapal harus lebih kecil dari kapasitas tangki maksimun (S_{MXi})

$$s_{im} + \sum_{v \in V} J_i q_{imv} - J_i R_i (t_{Eim} - t_{im}) - s_{Eim} = 0 \quad untuk\ setiap\ (i,m) \in N_P\ x\ M_i \tag{3.20}$$

  - *Stock level bound*

    Untuk *stock level bound* dipelabuhan loading sebagai berikut :

$$s_{im} - a_{im}^{+} \leq \mathrm{A_{MXi}}, \quad \forall i \in N_p, \quad m \in M_i \tag{3.21}$$

$$s_{(E)im} - a_{im}^{+} \leq \mathrm{A_{MXi}}, \quad \forall i \in N_p, \quad m \in M_i \tag{3.22}$$

- Sign and Integrality Constraint

$$\mathrm{x_{imjnv}} \in \{0,1\}, \quad \forall_v \in V, \ (i,m,j,n) \in \mathrm{A_v} \tag{3.23}$$

$$\mathrm{w_{im}} \in \{0,1\}, \quad \forall_i \in N, \ m \in \mathrm{M_i} \tag{3.24}$$

$$t_{im}, s_{im}, s_{Eim} \geq \{0,1\}, \quad \forall_i \in N, \quad m \in \mathrm{M_i} \tag{3.25}$$

$$l_{im}, qv_{imv} \geq 0 \quad \forall_v \in V, \quad i \in N, \quad m \in \mathrm{M_i} \tag{3.26}$$

3.6 Pengolahan Data

Pada sub bab ini akan dijabarkan mengenai pengolahan data dengan memasukkan data ke dalam persamaan matematika model fingsi tujuan dan kendala. Permasalahn ini akan diselesaikan dengan menggunakan metode *Tabu Search* yang diolah dengan menggunakan *software* Matlab.

Dari hasil pengolahan data akan diperoleh *output* berupa rute dan jadwal kapal yang optimal dalam mendistribusikan LPG dari pelabuhan *loading* ke pelabuhan *unloading.*

3.6.1 Penyusunan Algoritma

Penyelesaian masalah penjadwalan kapal yang harus menjaga inventori pada kasus ini menggunkan algoritma *Tabu Search.* Untuk menuliskan algoritma ini sekaligus melakukan pencarian solusi optimal, digunakan bahasa

pemrograman pada *software* Matlab. *Source Code* program Matlab yang digunakan dalam penelitian ini dapat dilihat pada bagian lampiran.

Pembuatan program optimasi untuk penjadwalan kapal yang optimal menggunakan algoritma *Tabu Search* dengan Matlab ini didasarkan pada fungsi objektif yaitu meminimumkan total biaya distribusi, dan output yang diharapkan dari program ini berupa rute dan jadwal pendistribusian yang optimal dari setiap kapal dengan jumlah muatan yang dimuat/dibongkar dan biaya yang diperlukan.

3.6.1.1 Penetapan Parameter Kontrol

Ada dua parameter kontrol yang nilainya harus ditetapkan dalam penyusunan algoritma ini, yaitu jumlah solusi tetangga dan panjang *tabu list*. Untuk menetapkan kedua parameter tersebut dilakukan percobaan dengan menggunkan beberapa sampel nilai jumlah solusi tetangga dan panjang *tabu list.* Untuk menetapkan jumlah solusi tetangga tersebut dilakukan percobaan dengan nilai antara 10 – 60. Sementara untuk menetapkan panjang *tabu list* dilakukan percobaan nilai antara 10 – 40.

Percobaan untuk menetapkan jumlah solusi tetangga yang terbaik dilakukan dengan melakukan simulasi dengan mengubah-ubah nilai parameter jumlah solusi tetangga, sedangkan parameter lainnya (panjang *tabu list* dan jumlah iterasi) dibuat tetap. Dengan menggunakan nilai panjang *tabu list* = 20 serta jumlah iterasi = 100, didapatkan hasil percobaan seperti table berikut:

Table 3.15 Hasil Percobaan Jumlah Solusi Tetangga

Jumlah Solusi tetangga	Total Biaya (US$)	Waktu Iterasi (detik)
10	833640	6.330238
20	833640	6.109676
30	833640	5.786949
40	833640	5.654354
50	833640	5.611844

Berdasarkan hasil percobaan yang dilakukan diperoleh total biaya yang sama di setiap jumlah solusi tetangga. Hal ini dikarenakan rute selama 30 hari yang sangat pendek sehingga *move* yang terjadi tidak memiliki perbedaan yang jauh setiap iterasi (jumlah *move* yang terjadi tidak banyak). Sehingga untuk pengolahan data selanjutnya akan digunakan parameter jumlah solusi tetangga sebesar 50 berdasarkan waktu iterasi tercepat.

Untuk menetapkan panjang *tabu list* dilakukan simulasi seperti parameter jumlah solusi tetangga. Dengan menggunakan jumlah solusi tetangga = 50 dan jumlah iterasi sebesar 100, didapatkan hasil percobaan yang sama. Total biaya yang diperoleh dari setiap panjang *tabu list* sama, yaitu sebesar US$ 833640. Maka panjang *tabu list* yang digunakan dalam penelitian selanjutnya adalah 10, mengingat *move* yang terjadi pada iterasi tidak banyak.

Selain dua parameter di atas, pada tahap ini juga ditentukan criteria terminasi program. Criteria terminasi dapat ditentukan berdasarkan jumlah iterasi maksimum ataupun waktu proses. Pada kasus ini criteria terminasi yang digunakan adalah jumlah iterasi maksimum. Dalam hal ini ditentukan bahwa program akan mencapai terminasi apabila telah melakukan 500 iterasi.

3.6.1.2 Langkah-langkah Algoritma dalam Program

Seperti telah dinyatakan dalam bab sebelumnya, untuk menyelesaikan permasalahan penelitian ini, digunakan algoritma metaheuristik *Tabu Search.* Langkah-langkah dalam menyelesaikan algoritma *Tabu Search* dapat dijelaskan sebagai berikut:

1. **Menentukan solusi awal**, diperoleh dari hasil pengolahan data dengan metode *random.* Acuan awal ini digunakan sebagai pembanding ketika proses *tabu search* dimulai. Pada tahap ini ditentukan $S_{best} = S$, *C_itr = 0 (current iteration counter).*
2. **Inisialisasi *tabu search***, menentukan skema *tabu tenure* seta nilai untuk tiap parameternya. Kemudian menentukan jumlah total iterasi *T_itr* dan *B_itr* = 0 (*best iteration counter*).
3. **Melakukan iterasi,**

   Lakukan *move* untuk membuat solusi tetangga, ada beberapa macam *move* yang dapat dipilih selama proses pencarian berjalan yaitu:

   a. *Local search* yang terdiri dari dua macam yaitu:
      i. *Insertion*: memilih secara acak satu bagian struktur untuk dipindah ke bagian yang lain.
      ii. *Swap*: memilih secara acak dua bagian struktur untuk selanjutnya ditukar posisinya.

b. Neighborhood Search

Pada pencarian dengan teknik ini, setiap kemungkinan atribut dari struktur dapat dipindah-pindah. Permutasi *n-change neighborhood* mengambil *n* elemen dari matriks solusi.

Kemudian menentukan solusi saat ini (*current solution*) S menjadi S', *C_itr=C_itr+1*.

4. Untuk menghindari terulangnya langkah yang diambil, maka dilakukan *tabu test. Tabu test* memanfaatkan *tabu list* yang sudah ada. Tujuan sebenarnya dari *tabu list* bukan untuk mencegah terulangnya langkah-langkah yang telah diambil, teapi lebih kepada agar tidak mundur untuk mencegah perulangan, daftar solusi yang telah dicapai disimpan dalam sebuah table.
5. *Alternative move* yang lolos *tabu test* harus melewati *aspiration test*. Jika tidak dapat melewati *aspiration test*, maka tidak akan diteruskan ke iterasi berikutnya. Jika alternative move mempunyai aspiration criteria yang lebih baik daripada *aspiration threshold*, maka dilakukan eksekusi terhadap *alternative move* tersebut dan memperbaharui memori yang tidak relevan.
6. Jika aturan pemberhentian sudah memenuhi syarat pemberhentian, maka pencarian berhenti.

3.6.2 Pengerjaan Solusi Awal

3.6.2.1 *Input*

Input data yang diperlukan untuk pengolahan data awal ialah data kapal (kapasitas kapal, waktu *un(loading),* dan FOC yang dibutuhkan), data pelabuhan (kapasitas tangki pelabuhan, stok awal pelabuhan dan *sea time*) dan kondisi awal pelabuhan, berupa posisi kapal dan inventori di setiap kapal dan pelabuhan.

- Pelabuhan

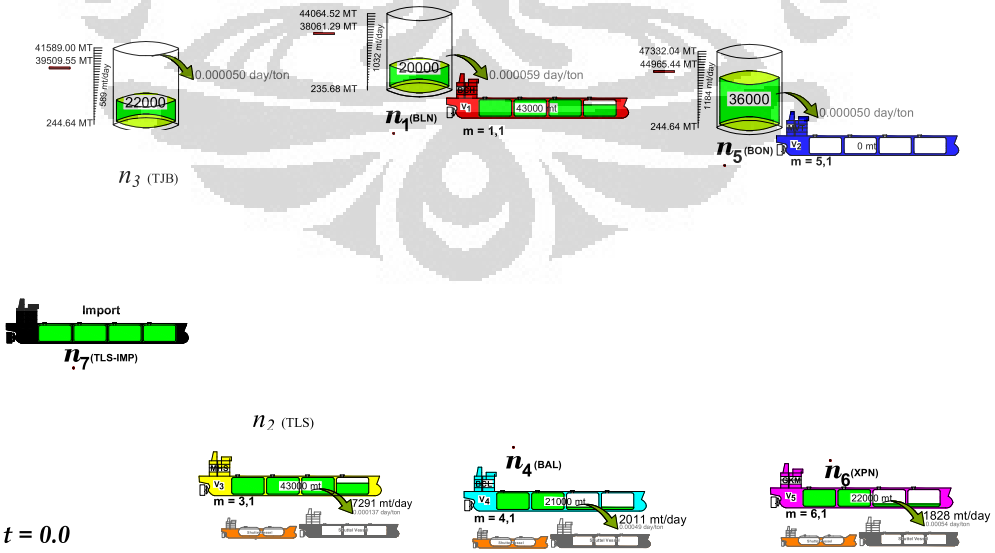
Table 3.16 Kondisi Awal Pelabuhan

Pelabuhan (n_i)	Inventory (IS_i)	Produksi/ Konsumsi (R_i)	Keterangan
BLN (n_1)	20000	1032	Pelabuhan *Loading*
TJB (n_3)	22000	589	Pelabuhan *Loading*
BON (n_5)	36000	1184	Pelabuhan *Loading*
IMPORT (n_7)	NA	NA	Pelabuhan *Loading*
TLS (n_2)	43000	-7291	Pelabuhan *Unloading*
BAL (n_4)	21000	-2011	Pelabuhan *Unloading*
XPN (n_6)	22000	-1828	Pelabuhan *Unloading*

- Kapal

Table 3.17 Posisi Awal Kapal

Kapal (v_i)	Muatan (Q_v)	Posisi	Keterangan
BCL (v_1)	43000	BLN (n_1)	*Loading*
MVT (v_2)	0	BON (n_5)	*Loading*
MHS (v_3)	43000	TLS (n_2)	*Unloading*
BCL (v_4)	21000	BAL (n_4)	*Unloading*
GKM (v_5)	22000	XPN (n_6)	*Unloading*



Gambar 3.5 Kondisi Awal Perencanaan

3.6.2.2 Langkah Pengerjaan

Pengerjaan solusi awal disini dimulai dengan memasukkan data yang telah dikumpulkan ke dalam Matlab. Data ini berbentuk matrix yang dibuat dalam *Microsoft Excel* dan telah diolah sehingga program Matlab dapat membacanya. Pengerjaan solusi awal ini tidak menggunakan metode random ataupun metode heuristic lainnya. Fungsi tujuan dan setiap kendala (*constraint*) dari model matematika yang telah diperoleh sebelumnya di masukkan ke dalam *software* Matlab. *Constraint* ini diterjemahkan dalam bentuk *coding* program berupa *logic.* Dengan data yang telah di*input* sebelumnya, maka program akan bekerja secara otomatis untuk mencari solusi yang paling optimal. *Source Code* dapat dilihat di lampiran.

3.6.2.3 Output

Hasil dari tahap pengerjaan awal ini berupa rute dan jadwal kapal yang optimal, yang merupakan solusi awal untuk tahap pengerjaan selanjutnya. Solusi awal ini sudah merupakan solusi yang sangat optimal karena telah memenuhi seluruh *constraint* yang ada, termasuk *time window* dengan menjaga inventori di setiap pelabuhan *loading.* Dari solusi awal ini diperoleh jumlah muatan yang dimuat/dibongkar dan total biaya distribusi yang minimal. Solusi awal ini pun siap diolah dengan menggunakan algoritma *Tabu Search.*

3.6.3 Verifikasi dan Validasi Program

Sebelum menggunakan program untuk mengolah data solusi awal, perlu dilakukan verifikasi dan validasi terhadap program. Varifikasi adalah proses pemeriksaan apakah logika operasional model telah sesuai dengan logika diagram alir. Beberapa proses verifikasi yang dilakukan dalam proses pembuatan program ini adalah:

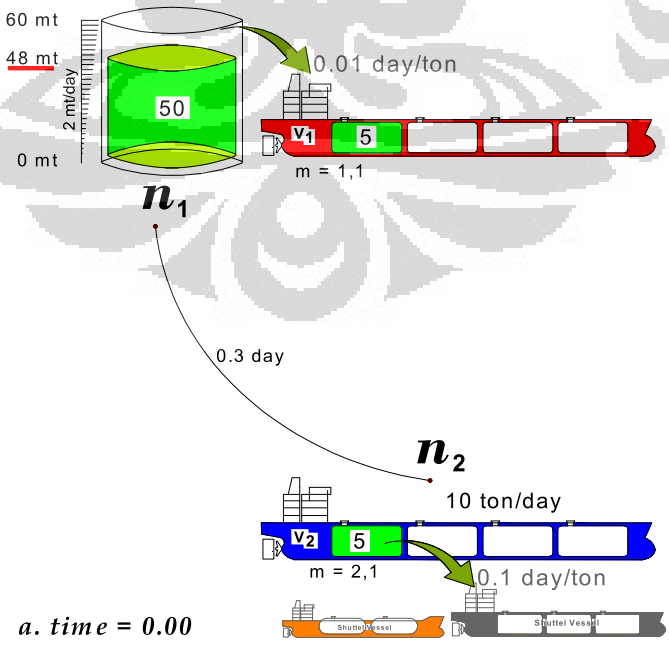
- Memastikan kebenaran logika pemikiran, yaitu kesesuaian penulisan *source code* dengan konsep algoritma *Tabu Search* dan dapat dijalankan dengan baik.
- Memastikan bahwa program sudah memasukkan semua batasan masalah yang ada.
- Memastikan fungsi tujuan telah sesuai.

Setelah proses verifikasi program, langkah selanjutnya adalah melakukan validasi. Validasi dilakukan untuk memastikan bahwa program menghasilkan output yang benar. Hasil perhitungan yang dihasilkan oleh program harus bernilai sama dengan perhitungan manual. Validasi dapat dilakukan dengan menggunkan data *dummy* untuk permasalahan yang sederhana.

Untuk melakukan validasi digunakan data *dummy* yaitu dua pelabuhan (*loading* dan *unloading*) dengan dua kapal, sebagai berikut:

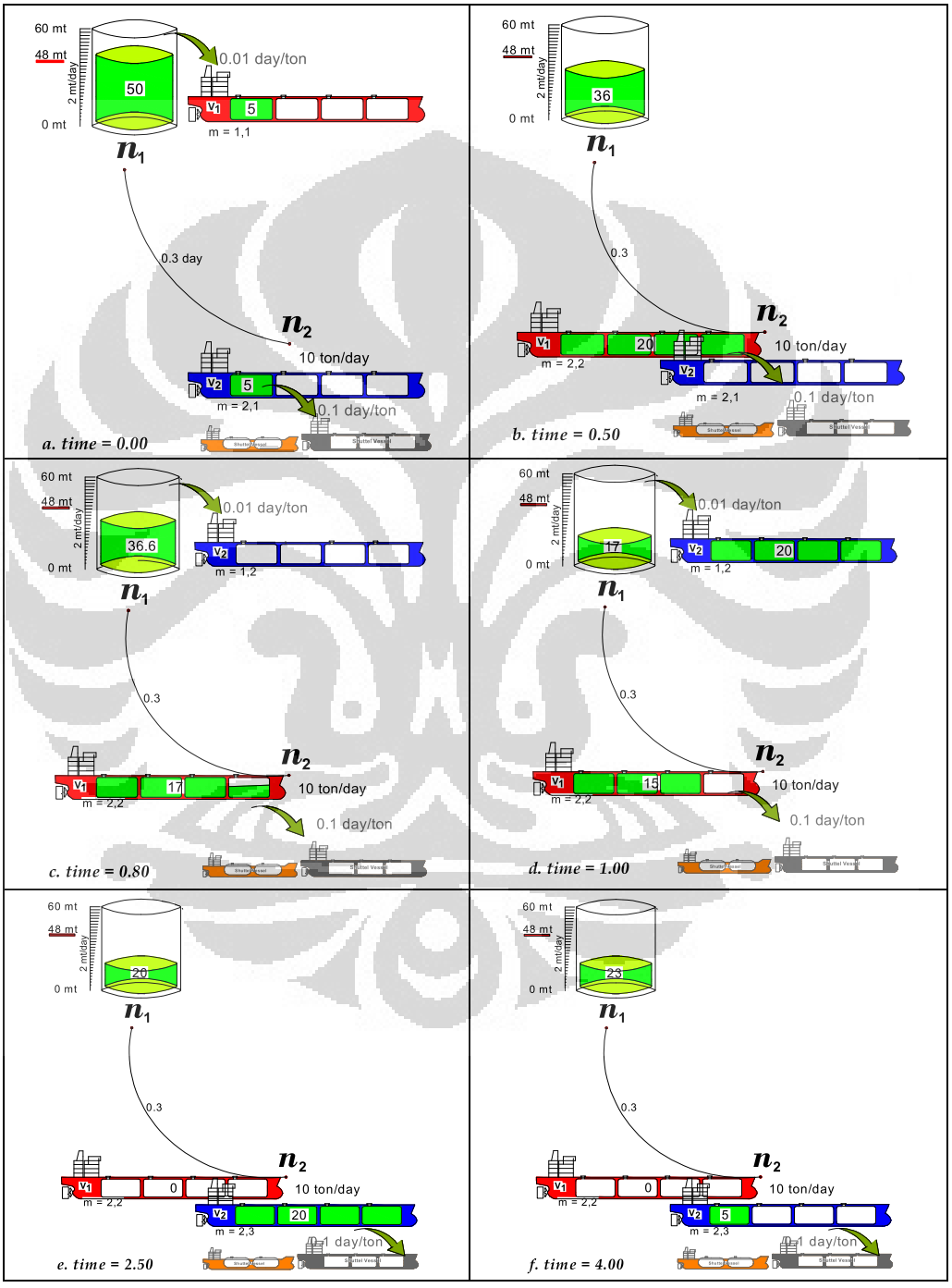
Terdapat dua kapal (v_1 dan v_2) dengan kapasitas mutan kapal (CAP_v) 1 dan 2 =20 mt. kedua kapal ini digunakan untuk mengangkut muatan dari pelabuhan *loading* (n_1) ke pelabuhan *unloading* (n_2). Pelabuhan n_1 memiliki kapasitas maksimum tangki S_{MXi} = 60 mt dengan A_{MXi} = 48 mt, kecepatan produksi R_1 = 2 mt/hari dan kecepatan *loading* T_{Q1} = 0.01 hari/mt. sedangkan pelabuhan *unloading* n_2 memiliki kecepatan konsumsi R_2 = 10 mt/hari, kecepatan *unloading* T_{Q2} = 0.1 hari/mt, dan *sea time* dari n_1 ke n_2 adalah 0.3 hari.

Pada awal perencanaan, kapal v_1 berada di n_1 dengan inventori awal Q_1 = 5 mt, sedangkan posisi inventori di pelabuhan n_1, IS_1 = 50 mt. Kapal v_2 berada di pelabuhan *unloading* n_2 dengan sisa muatan Q_2 = 5 mt. Biaya berlayar (C=US$1). Biaya pelabuhan ($C_F$ = US$2), biaya *un(loading)* (C_w = 0) dan biaya pinalti (C_P =US$ 0.5). Gambaran kondisi awal dapat dilihat sebagai berikut:



Gambar 3.6 Kondisi Awal Perencanaan (verifikasi & validasi)

Model matematika di *running* dalam program Matlab. Model matematika yang diinput telah mendapatkan hasil berupa rute dan jadwal yang dapat dilihat pada Gambar 3.7.



Gambar 3.7 Solusi penjadwalan kapal (verifikasi & validasi)

Dari hasil yang digambarkan dapat dilakukan perhitungan manual berikut.

a) Pada *time* = 0, *port call* (1,1) , kapal v_1 memiliki *cargo onboard* (l_{imv}) sebesar 5 mt dan pada *port call* (2,1), kapal v_2 telah memiliki muatan kapal (l_{imv}) sebesar 5 mt.

b) Pada *time* = 0.5, sisa muatan di kapal v_2 habis (5-0.5*10) dan mengharuskan *port call* (2,2) diisi oleh kapal v_1 dengan muatan 20 mt (5mt+(0.15day/0.01day/mt) dengan jarak tempuh 0.3 hari sehingga v_1 sampai di n_2 pada hari ke 0.45.

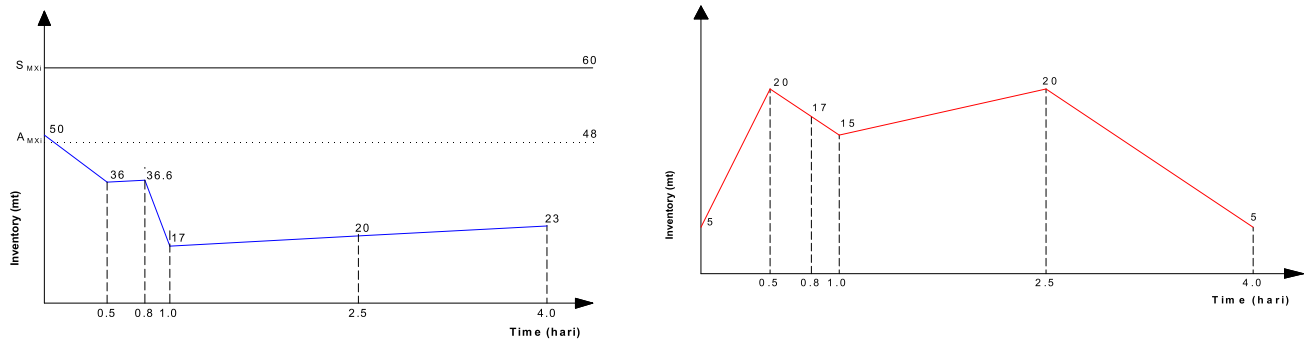
c) Pada *time* = 0.8, kapal v_2 telah sampai di pelabuhan n_1 pada *port call* (1,2), sedangkan muatan kapal v_1 di pelabuhan n_2 sebesar 17 mt (20-0.3*10) dan inventori di pelabuhan n_1 menjadi 36.6 mt (36+0.3*20).

d) Pada *time* = 1, posisi kapal v_2 di pelabuhan n_1 telah terisi penuh sebesar 20 mt (0.2/0.01), sedangkan posisi kapal v_1 di pelabuhan n_2 tinggal 15 mt (17-0.2*10).

e) Pada *time* = 2.5 merupakan awal *port call* (2,3), dimana jumlah muatan kapal v_1 pada *port call* sebelumnya (2,2) telah habis (15-1.5*10), sedangkan kapal v_2 akan mengisi *port call* (2,3) dengan muatan 20 mt.

f) Pada *time* = 4 yang merupakan batas periode perencanaan, kapal v_1 tetap di pelabuhan n_2 dengan posisi muatan v_1 tinggal 5 mt.

Biaya kapal v_1 = US$ 6, dimana C = US$ 1, C_F = US$ 4 (ke pelabuhan n_1 dan n_2) dan biaya pinalti pada *port call* (1,1) sebesar 2 mt sehingga C_P=US$ 1. Biaya kapal v_2 = US$ 8, dimana C = US$ 2 dan CF = US$ 6, sehingga total biaya sebesar US$ 14, sama dengan output *running program*.



Gambar 3.8 Inventori tiap pelabuhan (verifikasi & validasi)

Dari output program dibandingkan dengan perhitungan manual didapatkan hasil yang sama, sehingga program dikatakan telah terverifikasi dan valid.

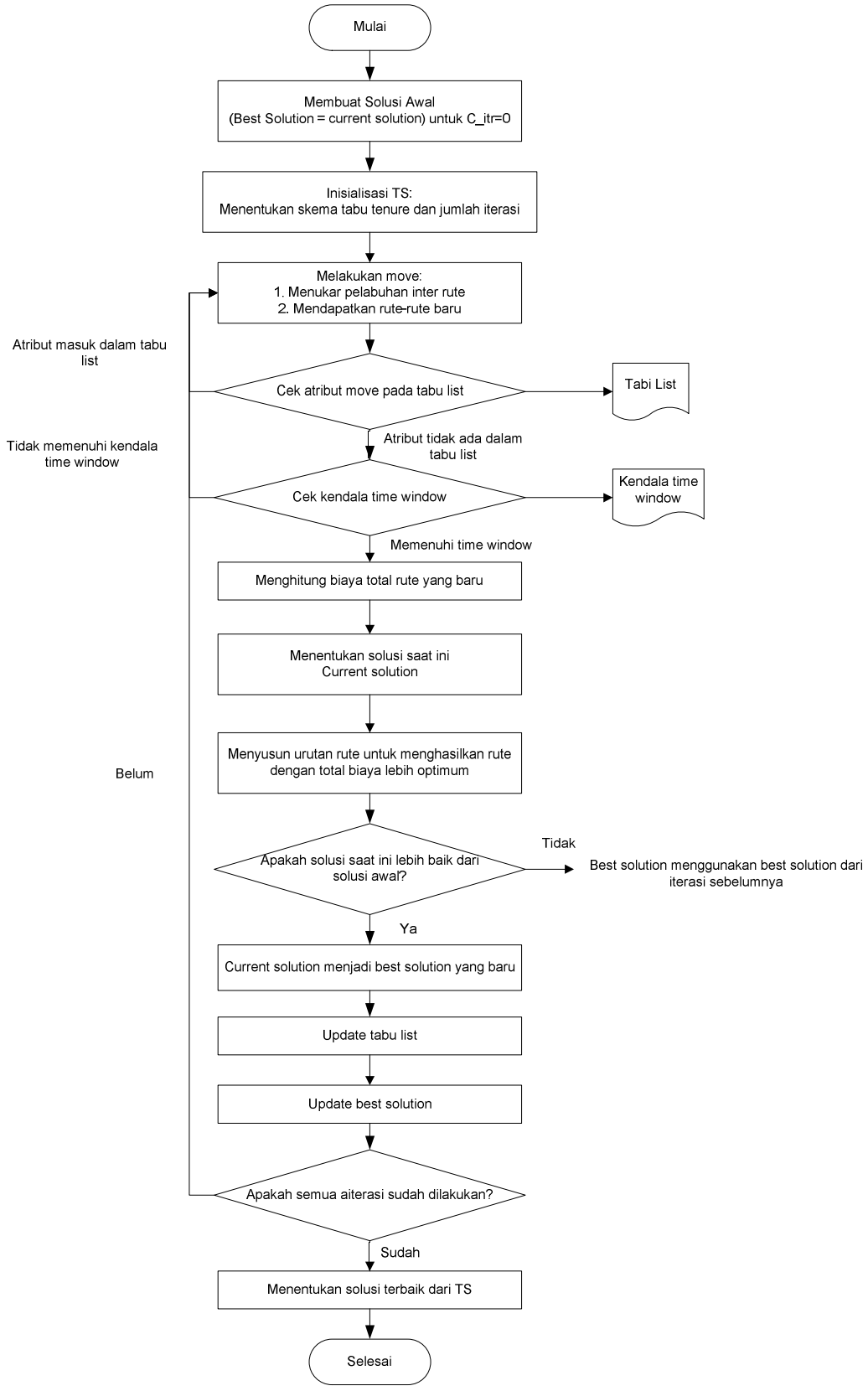
3.6.4 Pengolahan Solusi Awal dengan Algoritma*Tabu Search*

Untuk melakukan pengolahan data lebih lanjut terhadap solusi awal yang sudah diperoleh, dibuatlah program dengan menggunakan *software* Matlab dengan menerapkan algoritma *tabu search.* Data yang diperlukan untuk membuat program ini sama dengan matrix yang dibuat untuk mencari solusi awal. Data ini dibuat dalam bentuk matrix di *Microsoft Excel.*

Pada setiap proses pengerjaan atau *run* program, data yang perlu dimasukkan adalah data rute dan jadwal kapal dari solusi awal yang akan dioptimalkan lagi. Selanjutnya data rute dan jadwal kapal yang dimasukkan pada proses *run* program ini akan diolah sesuai dengan tahap algoritma *tabu search* yang secara skematis terlihat pada gambar 3.6.

Prosedur pemilihan atribut perpindahan dilakukan dengan metode *swap,* artinya adanya pertukaran dua rute di setiap rute kapal. Karena adanya pertukaran rute secara acak, maka banyak solusi yang diperoleh dengan biaya minimum namun tidak memenuhi *time window.*

Agar program dapat mengolah data rute dan jadwal kapal yang telah dimasukkan, perlu ditentukan jumlah iterasi, besar *tabu tenure*, panjang *tabu list* yang menggambarkan berapa panjang iterasi suatu atribut *move* dikatakan tabu atau tidak boleh dilakukan. Jumlah iterasi dan panjang *tabu list* harus ditentukan setelah memasukkan data rute dan jadwal kapal karena program Matlab tidak bisa menentukannya sendiri. Kedua faktor ini juga sangat mempengaruhi waktu proses *run* program dan output yang dihasilkan. Untuk itu, jumlah iterasi, panjang *tabu list,* dan jumlah solusi tetangga telah ditentukan sebelumya.



Gambar 3.9 Diagram alir optimasi menggunakan algoritma *tabu search*

3.6.4.1 Pengerjaan Algoritma *Tabu Search*

Pada tahap awal, program akan meminta *input* berupa rute dan jadwal distribusi berdasarkan solusi awal yang telah diperoleh sebelumnya. Kemudian program akan mengakses *database* pada *spreadsheet* yang berupa data matrix rute kapal, inventori kapal dan pelabuhan, total biaya dan jadwal keberangkatan dan kepergian kapal. Rute dari solusi awal ini akan dijadikan solusi terbaik saat ini, dimana solusi ini akan digantikan jika ditemukan solusi yang lebih optimum baik dari segi biaya maupun pemenuhan kendala.

Selanjutnya dilakukan tahapan inisialisasi yaitu menentukan jumlah iterasi dan penggunaan *tabu tenure*. Jumlah iterasi yang digunakan adalah 500, panjang *tabu list* adalah 10 dan jumlah solusi tetangga sebanyak 50. Sedangkan dalam penelitian ini, skema *tabu tenure* yang digunakan adalah *fix tabu tenure*, artinya selama iterasi, maksimal *tabu tenure* adalah tetap.

Di tahapan selanjutnya adalah masuk ke pengulangan solusi tetangga, dimana pada solusi saat ini (solusi awal), *move* urutan rute tujuan yang dikunjungi dilakukan dengan metode *swap,* yaitu dengan menukar secara random dua pelabuhan pada satu rute. Dalam hal ini diasumsikan satu *move* sama dengan 1 solusi tetangga pada 1 iterasi. Di setiap iterasi dilakukan pengecekan apakah atribut *move* yang digunakan masuk ke dalam *tabu list* atau tidak. Jika ada, maka *move* tersebut tidak boleh melanjutkan proses selanjutnya, sedangkan jika *move* yang digunakan tidak terdapat dalam daftar tabu, maka solusi yang dihasilkan harus dicek apakah sudah memenuhi semua kendala. Jika tidak memenuhi, maka *move* tersebut tidak dapat melanjutkan ke proses berikutnya, namun jika memenuhi, maka solusi tersebut menjadi solusi yang dipilih.

Jika solusi yang dipilih tersebut memiliki biaya yang lebih rendah daripada solusi terbaik pada iterasi saat ini, maka solusi tersebut menjadi solusi terbaik yang baru dan akan menjadi solusi saat ini, yang akan diproses pada iterasi-iterasi selanjutnya. Atribut *move* yang menghasilkan solusi terbaik yang baru tersebut direkam dalam *tabu list* sehingga dalam beberapa iterasi selanjutnya atribut *move* tersebut dilarang untuk dilakukan.

3.6.4.2 Output

Output hasil pengolahan data dengan algoritma *tabu search* adalah *port call* setiap kapal dengan biaya yang minimum dari masing-masing rute kapal. Dengan penetapan parameter perhitungan seperti yang disebutkan di atas, diperoleh hasil total biaya US$ 907,008.23, dengan waktu perhitungan yang dibutuhkan selama 5.611844 detik dengan menggunakan PC MacBook dengan spesifikasi Intel Core 2 Duo Processor, RAM 2 GHz dan *software programming* MATLAB versi R2009b.

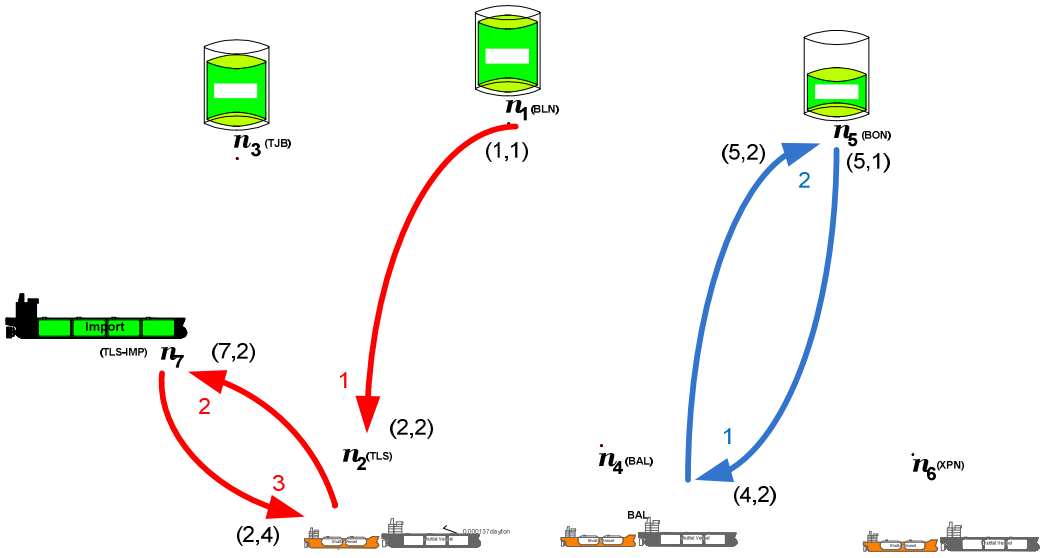
Hasil dari *running* program rute dan penjadwalan kapal dapat dilihat pada bab selanjutnya.

BAB 4
HASIL DAN ANALISIS

Penyelesaian fungsi tujuan dalam permasalahan minimalisasi biaya transportasi, biaya pelabuhan, biaya *un(loading)*, dan biaya pinalti dengan rute dan jadwal kapal yang telah memenuhi seluruh kendala telah diperoleh dari pengolahan data menggunakan algoritma *Tabu Search* dengan *software* MATLAB.

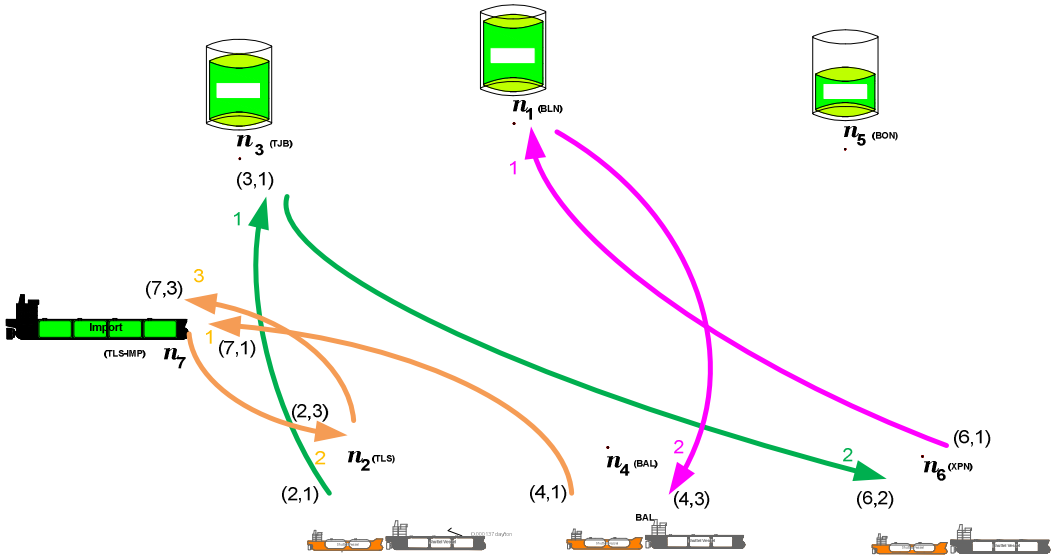
4.1 Hasil

Setelah program tervalidasi, maka data di-*run* untuk memperoleh hasil optimal. *Output* dari program dengan perangkat lunak Matlab adalah berupa penjadwalan kapal (Tabel 4.1), dengan visualisasi sebagai berikut.



Gambar 4.1 Pergerakan Kapal BCH (v_1) dan MVT (v_2)

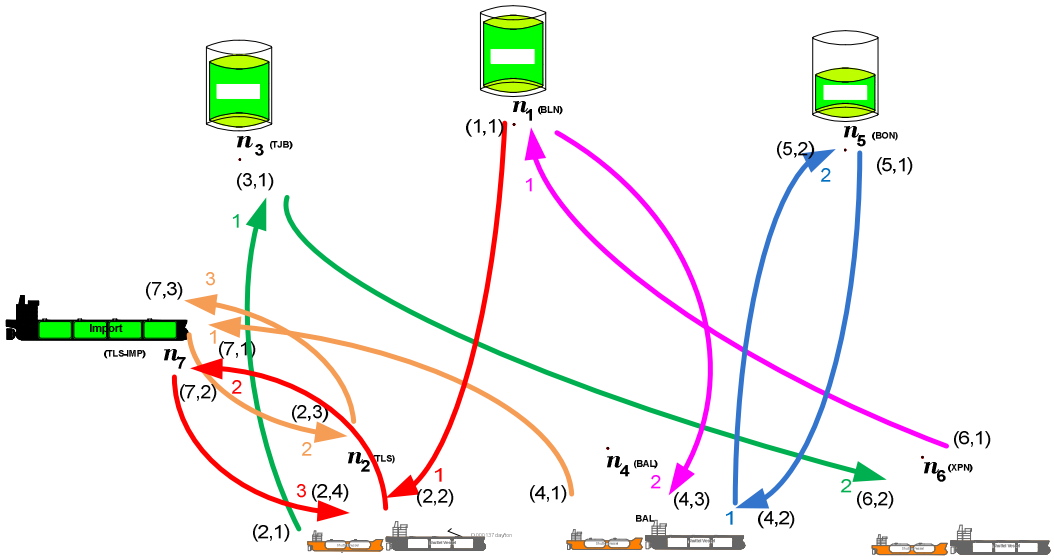
Pergerakan kapal selama ± 30 hari pada kapal BCH (merah) dan MVT (biru) membentuk *clustering*, dimana kapal BCH cenderung dari pelabuhan TLS ke pelabuhan TLS-IMP. Sedangkan kapal MVT *loading* di pelabuhan Bontang lalu unloading di pelabuhan BAL dan kembali lagi ke pelabuhan BON untuk *loading*. Pergerakan kapal pada penjadwalan selama 30 hari ini tidak terlalu banyak, sebab waktu lebih banyak dihabiskan di pelabuhan *unloading* untuk bongkar muatan kapal.



Gambar 4.2 Pergerakan kapal MHS (v_3), BCL (v_4), dan GKM (v_5)

Pergerakan kapal MHS (hijau), BCL (jingga), dan GKM (merah muda) dapat dilihat dalam gambar di atas. Dapat dilihat bahwa kapal MHS cenderung *loading* di pelabuhan TLS-IMP. Pergerakan kapal MHS dari TLS ke TLS-IMP lalu bongkar di XPN. Kapal BCL bergerak dari pelabuhan BAL untuk *loading* ke TLS-IMP, kemudian bongkar di pelabuhan TLS dan kembali lagi ke TLS-IMP. Pergerakan kapal GKM dimulai dari pelabuhan XPN lalu *loading* ke pelabuhan BLN dan kemudian bongkar di pelabuhan BAL.

Berikut dapat digambarkan gabungan dari pergerakan kelima kapal:



Gambar 4.2 Pergerakan Lima Kapal

Table 4.1 Penjadwalan Kapal ±30 hari

Kapal	Q_v	Port Call (i,m)											
		t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)
BCH (v_1)	43000	1,1		2334	2,2		45334	7,2		45334	2,4		45334
		0.00	3.1048		4.8976	12.1153		14.8096	15.4669		17.334	24.552	
MVT (v_2)	0	5,1		38121	4,2		38121	5,2		35376			
		0.00	7.1757		9.4436	29.4019		31.6698	33.332				
MHS (v_3)	43000	2,1		43000	3,1		22868	6,2		22868			
		0.00	5.8976		7.6314	8.706		11.0333	24.543				
BCL (v_4)	21000	4,1		21000	7,1		45340	2,3		45340	7,3		45340
		0.00	10.4436		11.248	12.1153		12.1153	18.334		18.3715	24.5145	
GKM (v_5)	22000	6,1		22000	1,2		36014	4,3		36014			
		0.00	12.033		14.872	26.417		28.4019	46.31				

Table 4.2 Biaya Operasi Kapal

Kapal	Total Cost (US$)	Port Call (i,m)											
		C	Cf	Cw	C	Cf	Cw	C	Cf	Cw	C	Cf	Cw
BCH (v_1)	145522.479	1,1			2,2			7,2			2,4		
		-	6448.277	799.9111	47961.51	6448.277	23835.95	965.02	6448.277	21327.92	1003.26	6448.277	23835.8012
MVT (v_2)	175306.776	5,1			4,2			5,2					
		-	6714.618	9531.869	45974.23	6714.618	90772.3	957.6	6714.618	8845.42			
MHS (v_3)	144756.285	2,1			3,1			6,2					
		-	6693.261	23331.54	45310.5	6693.261	5570.96	979.98	6693.261	49483.52			
BCL (v_4)	173421.311	4,1			7,1			2,3			7,3		
		-	6448.616	33859.31	52223.44	6448.616	19647.99	1043.5	6448.616	20161.111	1043.5	6448.616	19647.998
GKM (v_5)	236892.086	6,1			1,2			4,3					
		-	6448.616	52292.66	74570.21	6448.616	11866.5	985.03	6448.616	77831.84			
Total =	876817.436												

Table 4.3 Biaya Operasi Kapal tanpa Biaya *Un(Loading)*

Kapal	Total Cost (US$)	Port Call (i,m)							
		C	Cf	C	Cf	C	Cf	C	Cf
BCH (v_1)	75722.898	1,1		2,2		7,2		2,4	
		-	6448.277	47961.51	6448.277	965.02	6448.277	1003.26	6448.277
MVT (v_2)	67075.684	5,1		4,2		5,2			
		-	6714.618	45974.23	6714.618	957.6	6714.618		
MHS (v_3)	66370.263	2,1		3,1		6,2			
		-	6693.261	45310.5	6693.261	979.98	6693.261		
BCL (v_4)	80104.904	4,1		7,1		2,3		7,3	
		-	6448.616	52223.44	6448.616	1043.5	6448.616	1043.5	6448.616
GKM (v_5)	94901.088	6,1		1,2		4,3			
		-	6448.616	74570.21	6448.616	985.03	6448.616		
Total =	384174.837								

4.2 Analisis

Setelah dilakukan tahap pengolahan data dan diperoleh hasil, selanjutnya adalah tahap analisis. Dalam hal ini akan dilakukan analisis terhadap metode/program yang digunakan serta analisis hasil optimasi rute dan penjadwalan kapal.

4.2.1 Analisis Metode dan Program

Pada permasalahan ini, penyelesaian masalah rute dan jadwal kapal dilakukan dengan menggunakan program Matlab dengan algoritma *Tabu Search*. Secara umum program dapat berjalan dengan baik. Hal ini dibuktikan dengan solusi akhir yang ditawarkan masuk akal dan dapat diaplikasikan pada kondisi actual. Selain itu sudah dilakukan validasi terlebih dulu antara perhitungan manual dengan program.

Dengan menggunakan algoritma ini diharapakan diperoleh solusi global optimum yang seharusnya lebih baik dari solusi awal. Namun pada kenyataannya solusi akhir (global optimum) yang telah diperoleh tidak jauh lebih baik daripada solusi awal yang telah diperoleh sebelumnya dengan memasukkan seluruh kendala (*constraint*) menggunakan program Matlab. Hal ini dikarenakan pada metode algoritma *Tabu Search* diperlukan *move* yang dijadikan sebagai solusi tetangga untuk diiterasi. *Move* ini dibuat dengan metode *swap,* dimana dua pelabuhan dalam satu rute kapal dipertukarkan. Namun metode ini bertentangan dengan salah satu kendala dalam permasalahan ini, yaitu *time window.* Dimana dua pelabuhan yang dipertukarkan akan memiliki kemungkinan mengalami pergeseran waktu, baik dari segi waktu berlayar, dan juga waktu *un(loading)* karena muatannya pun akan berubah. Sehingga ada kemungkinan tidak tersedianya kapal(muatan) di pelabuhan *unloading.*

Pada saat *running* program, iterasipun terus berjalan. Setiap solusi yang memiliki total biaya yang lebih rendah dari hasil iterasi sebelumnya akan disimpan di *tabu list*. Namun setelah rute kapal diperiksa, ternyata ada pelabuhan *unloading* yang tidak disinggahi kapal selama beberapa hari, sehingga solusi inipun dibuang. Iterasi pun berjalan seperti itu seterusnya.

Dari hasil *run* program yang berjalan, diperoleh penyelesaian terbaik. Dimana solusi sudah sangat optimal, dapat dilihat dari segi biaya dan selalu tersedianya muatan di pelabuhan *unloading.*

Permasalahan seperti ini sudah pernah diteliti sebelumnya dengan menggunakan metode *Branch and Bound* dengan program LINGO. Hasil penjadwalan yang diperoleh dapat dilihat pada tabel 4.4.

Dengan menggunakan metode *Branch and Bound* dengan LINGO dapat dilihat perbedaanya. Total biaya operasi kapal (tanpa biaya pelabuhan) selama $\pm$30 hari menggunakan algoritma *Tabu Search* dengan Matlab adalah sebesar US$ 384174.837, sedangkan total biaya operasi kapal (tanpa biaya pelabuhan) menggunakan *Branch* and *Bound* dengan LINGO adalah sebesar US$ 440820.994. dari total biaya dapat dilihat bahwa metode algoritma *Tabu Search* dengan Matlab ini jauh lebih baik karena bisa menghemat biaya sebesar 12.85%

Perbedaan hasil penelitian yang dilakukan menggunakan *Branch and Bound* dengan LINGO dan algoritma *Tabu Search* dengan matlab juga dapat dilihat dari segi ketersediaan muatan di pelabuhan *unloding.* Hasil *running* program LINGO menunjukkan tidak adanya aktivitas *unloading* pada pelabuhan TLS *port call* (3,1), dapat dilihat dari l_{imv} (sisa muatan setelah kapal meninggalkan pelabuhan) = 4300 mt. Hal yang sama juga terjadi pada *port call* (3,3) dan (3,5).

Jika ditinjau dari segi waktu *running* program (*time elapsed*), terdapat perbedaan yang sangat jauh antara LINGO dan Matlab. Pengolahan data yang menggunakan LINGO memerlukan waktu *running* selama kurang lebih dua jam. Sedangkan waktu *running* program Matlab dengan jumlah iterasi 500 hanya memerlukan waktu 11-12 detik.

Table 4.4 Penjadwalan Kapal ±30 hari (dengan LINGO)

Kapal	Q_v	Port Call (i,m)														
		t_{im}	t_{Eim}	l_{imv}	t_{im}	t_{Eim}	l_{imv}	t_{im}	t_{Eim}	l_{imv}	t_{im}	t_{Eim}	l_{imv}	t_{im}	t_{Eim}	l_{imv}
BCH (v_1)	43000	1	1	(mt)	3	2	(mt)	7	3	(mt)	3	4	(mt)	7	6	(mt)
		0.00	1.26	45334	5.90	12.12	1862	12.16	18.09	45334	18.13	24.29	1454	24.33	24.33	1454
MVT (v_2)	0	5	1		6	2		5	3							
		0.00	1.91	22444	10.88	24.74	0	27.59	29.84	44965						
MHS (v_3)	43000	3	1		7	1		4	2		7	5		3	5	
		0.00	5.90	43000	5.94	9.36	25938	10.29	22.43	5245	23.36	24.05	47506	24.09	30.60	47506
BCL (v_4)	21000	4	1		7	2		3	3		7	4		4	4	
		0.00	10.29	306	11.22	11.87	45340	12.02	18.23	45340	18.27	18.93	45340	20.39	29.83	26070
GKM (v_5)	22000	6	1		1	2		4	3		5	2		6	3	
		0.00	12.03	320	16.78	17.55	10476	20.39	20.39	10476	22.66	22.81	13523	24.64	31.94	13523

Table 4.5 Biaya Operasi Kapal tanpa Biaya *Un(Loading)* (dengan LINGO)

Kapal	Total Cost (US$)	Port Call (i,m)											
		C	CF	total	C	CF	total	C	CF	total	C	CF	total
BCH (v1)	76687.918	3	2		7	3		3	4		7	6	
		47962	6448.277	54410	965.02	6448.277	7413	1003.3	6448.277	7451.5	965	6448	7413.3
MVT (v2)	61189.186	6	2		5	3							
		46802	6714.618	53517	957.6	6714.618	7672						
MHS (v3)	75185.474	7	1		4	2		7	5		3	5	
		980	6693.261	7673	25372.56	6693.261	32066	21036	6693.261	27729	1024	6693	7717.3
BCL (v4)	77397.364	7	2		3	3		7	4		4	4	
		22399	6448.616	28848	1092.41	6448.616	7541	1043.5	6448.616	7492.1	27068	6449	33516
GKM (v5)	161309.042	1	2		4	3		5	2		6	3	
		74570	6448.616	10476	53892.56	6448.616	60341	59572	6448.616	66021	49564	6449	13523
Total =	440820.994												

4.2.2 Analisis Hasil

Pada tahap analisis hasil ini akan dianalisis hasil yang menyangkut operasional berupa tingkat utilisasi ruang muat kapal, ketersediaan muatan di pelabuhan, inventori di pelabuhan *unloading,* jaminan *lifting,* rute dan biaya.

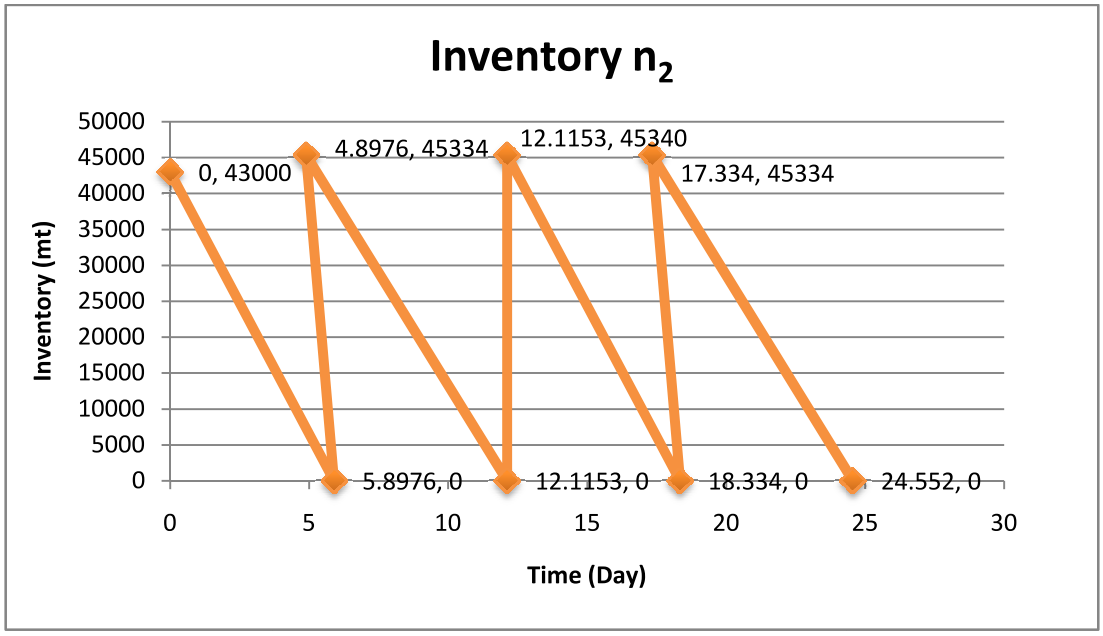
a) *Effective Load Factor* (ELF)

Effective Load Factor adalah tingkat utilisasi ruang muat kapal (perbandingan antara kargo yang diangkut dengan kapasitas maksimal ruang muat kapal). ELF kapal MVT, MHS dan GKM belum mencapai 100%. Kapal MVT mengangkut muatan rata-rata 34912 mt, dimana kapasitas kapal MVT adalah 47477 mt, sehingga ELF dari kapal ini sebesar 73.6 %. Begitu juga dengan kapal MHS dan GKM yang memiliki ELF sebesar 69.3% dan 64.1%. Kondisi ini diakibatkan oleh beberapa hal, yaitu:

- Kapal mengangkut kargo dengan tidak *full tank* untuk menangani terjadinya depot kritis di pelabuhan *unloading*, kondisi ini terjadi pada *port call* (3,1) dan (5,1).
- *Production rate* di pelabuhan *loading* yang cenderung kecil, sehingga pada saat kapal datang untuk muat, inventori di pelabuhan *loading* belum penuh (masih kurang dari kapasitas kapal).

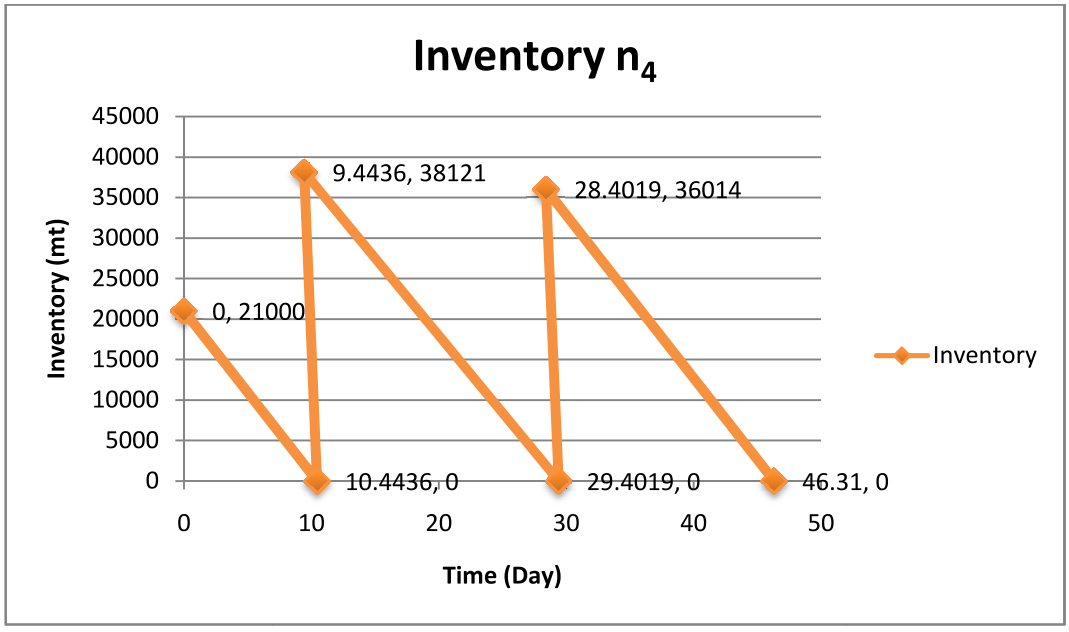
b) Ketersediaan (*availability*) muatan

Jaminan ketersediaan muatan di pelabuhan *unloading* selama periode perencanaan dapat dipenuhi. Dibandingkan dengan penjadwalan actual kapal, dimana terjadi depot kritis sebanyak 84 kali. Jaminan ketersediaan ini dibuktikan dengan terpenuhinya *time window*, dimana terdapat *overlapping* selama 1 hari (kapal sekarang datang 1 hari sebelum kapal sebelumnya pergi). Dengan adanya jaminan ketersediaan muatan di pelabuhan *unloading* (*floating storage*) ini, maka ketersediaan muatan di *end depot* akan terjamin, sehingga *demand* akan selalu dapat terpenuhi. Gambaran ketersediaan muatan di pelabuhan *unloading* dapat dilihat pada grafik berikut:



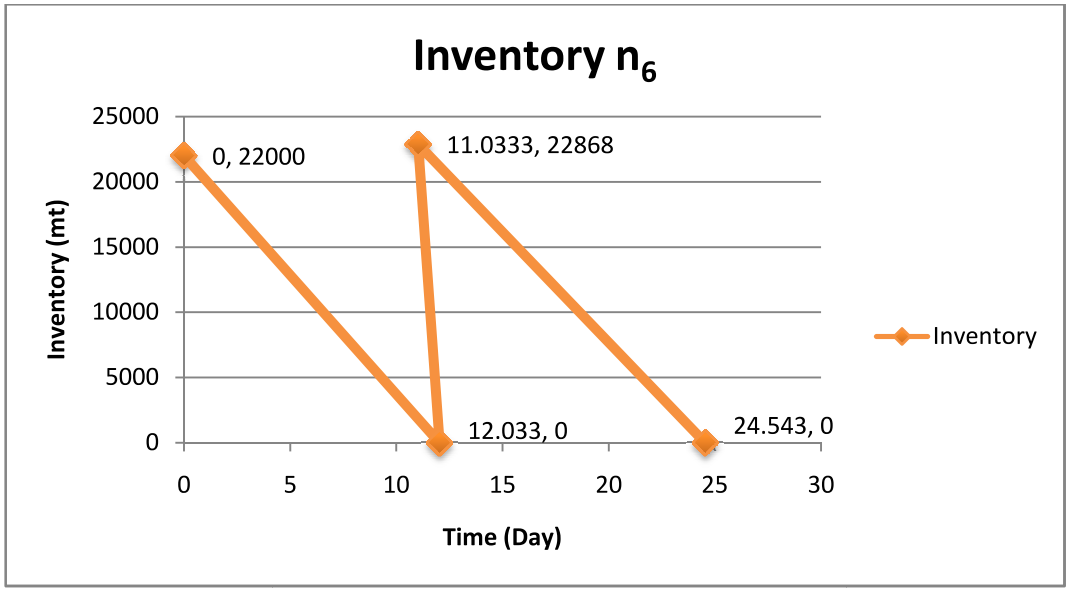
Gambar 4.3 Grafik Inventori di Pelabuhan TLS

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Tetapi pada *port call* (2,3) kapal datang tepat pada hari ke 12.1153. Hal ini dikarenakan waktu kapal yang sangat singkat untuk *loading* di pelabuhan TLS-IMP. Namun secara umum, muatan selalu tersedia di pelabuhan TLS dengan memenuhi kendala *time window.*



Gambar 4.4 Grafik Inventori di Pelabuhan BAL

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Maka disimpulkan bahwa muatan selalu tersedia di pelabuhan BAL dengan memenuhi kendala *time window*.

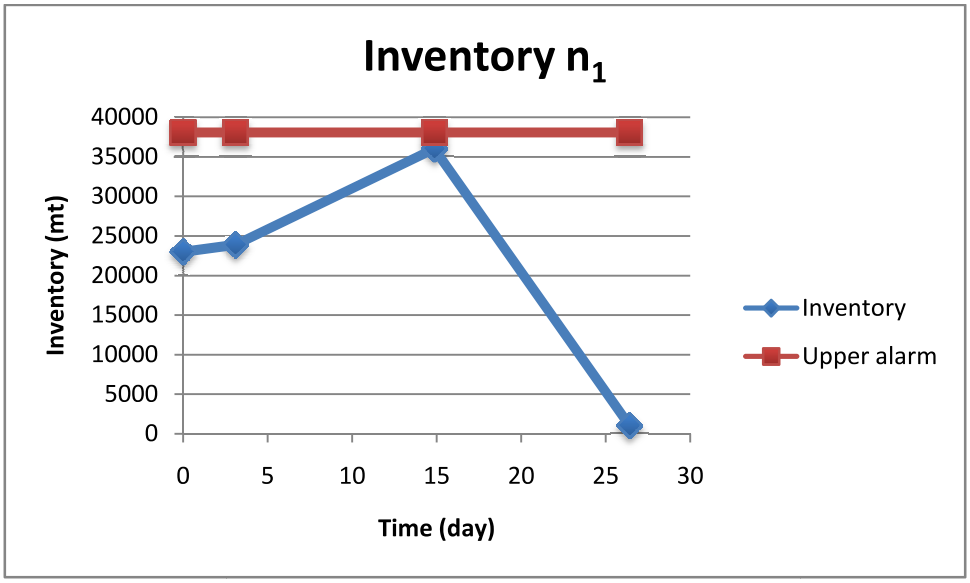


Gambar 4.5 Grafik Inventori di Pelabuhan XPN

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Maka disimpulkan bahwa muatan selalu tersedia di pelabuhan XPN dengan memenuhi kendala *time window*.

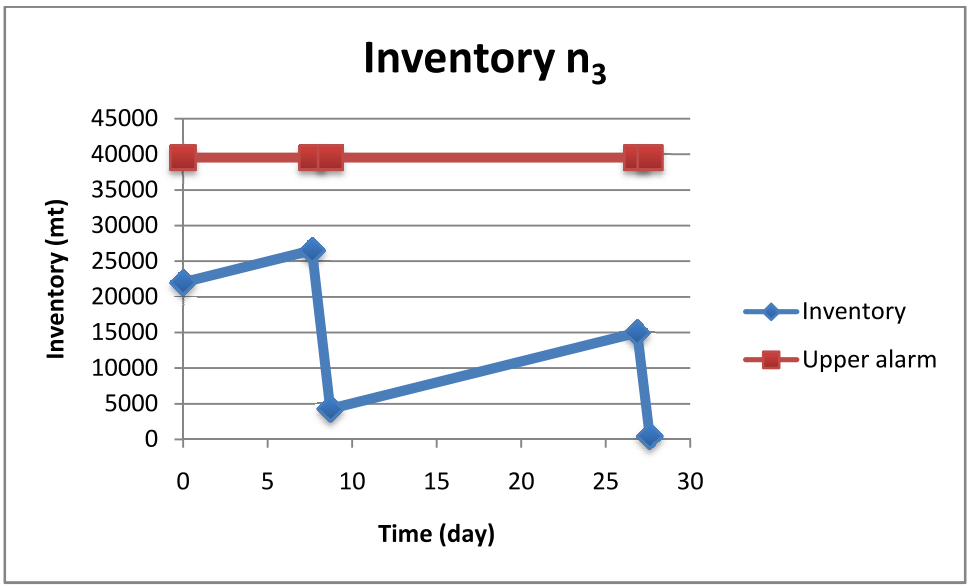
c) Inventori di pelabuhan *loading*

Salah satu kendala yang harus dihindari dari permasalahan ini adalah jika inventori pelabuhan *loading* melebihi *soft inventory level*. Kendala ini pun berhasil dihindari sehingga kapal tidak dikenai biaya pinalti. Level inventori setiap kunjungan dapat dilihat pada grafik berikut:



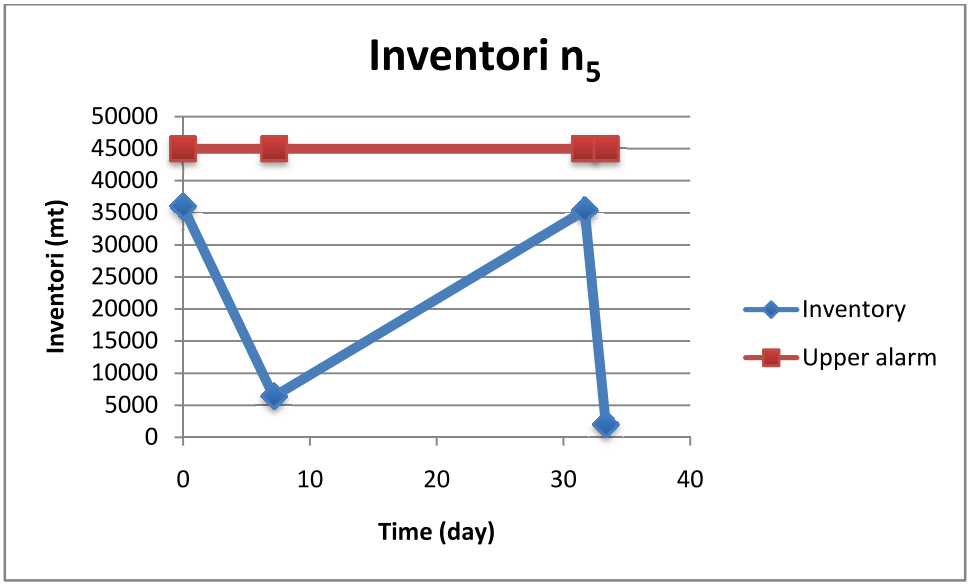
Gambar 4.6 Grafik Inventori di Pelabuhan BLN

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan BLN tidak pernah mencapai batas *soft inventori level* yaitu sebesar 38061.294 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.



Gambar 4.7 Grafik Inventori di Pelabuhan TJB

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan TJB tidak pernah mencapai batas *soft inventori level* yaitu sebesar 39509.55 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.



Gambar 4.8 Grafik Inventori di Pelabuhan BON

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan BON tidak pernah mencapai batas *soft inventori level* yaitu sebesar 44965.438 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.

d) Jaminan Lifting

Seluruh kargo *lifting* di pelabuhan *loading* tercapai. Pada periode ini persebaran permintaan kapal di pelabuhan *loading* cukup merata, maka jumlah inventori di pelabuhan *loading* pun merata, sehingga tidak ada pelabuhan yang inventorinya terlalu besar yang mengakibatkan pelabuhan terkena biaya pinalti.

e) Analisis rute

Adanya kecenderungan pelabuhan TLS untuk muat di pelabuhan TLS-IMP, hal ini wajar karena inventori pelabuhan TLS-IMP yang tidak terbatas dan juga waktu berlayar yang pendek. Jika diamati, pergerakan kapal dari pelabuhan *unloading* ke *loading* pada hasil program ini didasarkan pada

waktu berlayar yang disinkronisasikan dengan inventori pelabuhan yang lebih penuh.

f) Analisis Biaya

Berdasarkan hasil *output* program, diperoleh biaya operasi kapal sebesar US$ 867817.436 dan biaya operasi kapal tanpa biaya *un(loading)* adalah sebesar US$ 384174.837. jika dibandingkan dengan hasil *run* program menggunakan LINGO maka total biaya ini mengalami penghematan sebesar 12.85%.

g) Analisis Sensitifitas

Analisis sensitifitas pada permasalahan ini dapat dilakukan dengan mengubah-ubah jumlah kapal dan jumlah pelabuhan *loading* dan *unloading.* Namun setelah melihat hasil penjadwalan kapal pada penelitian ini, diprediksi bahwa tidak mungkin mengurangi kapal dengan jumlah pelabuhan seperti pada permasalahan ini. Hal ini dikarenakan kondisi pelabuhan *unloading* sebagai *floting storage*, dimana harus selalu ada ketersediaan kapal di ketiga pelabuhan ini. oleh karena itu hanya ada dua kapal yang bisa muat ke pelabuhan *loading* untuk menggantikan kapal di pelabuhan *unloading* nantinya. Jika kapal ditambahkan dalam kasus ini, hal ini memang tidak akan berpengaruh pada pergerakan kapal, justru sangat membantu karena utilitas kapasitas kapal bisa mencapai 100 % karena kapal tidak perlu datang dalam kondisi tidak *full tank* ke pelabuhan *unloading.* Namun akan terbeban pada biaya investasi yang cukup besar. Sehingga dengan kelima kapal ini sudah cukup untuk memenuhi *demand* di setiap pelabuhan *loading* dengan menjaga inventori agar tidak melebihi batas *soft inventory level.*

Jika analisis sensitifitas dilakukan dengan menambahkan pelabuhan *unloading*, maka harus dilakukan penambahan kapal sehingga *demand* di setiap pelabuhan *unloading* bisa terpenuhi.

4.2.3 Analisis Hasil Penjadwalan Kapal VLGC ±60 hari

Kelebihan metode *Tabu Search* dengan *tools* Matlab bisa membuat penjadwalan rute kapal sesuai dengan yang dibutuhkan. Sesuai dengan perencanaan awal, maka berikut adalah rute penjadwalan kapal VLGC ±60 hari.

Table 4.6 Penjadwalan Kapal ±60 hari

Kapal	Q_v	Port Call (i,m)											
		t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)
BCH (v_1)	43000	1,1		2334	2,2		45334	7,2		45334	2,4		45334
		0.00	3.1048		4.8976	12.1153		12.0778	17.296		17.334	24.55	
MVT (v_2)	0	5,1		38121	4,2		38121	7,5		47477	4,4		47477
		0.00	7.1757		9.4436	29.4019		30.206	44.38		45.31	69.92	
MHS (v_3)	43000	2,1		43000	7,1		47506	6,2		47506	5,2		38649
		0.00	5.8976		5.9351	8.88		11.0333	38.021		39.819	41.635	
BCL (v_4)	21000	4,1		21000	7,2		45340	2,3		45340	7,3		45340
		0.00	10.4436		11.248	12.1153		12.1153	18.334		18.3715	21.39	
GKM (v_5)	22000	6,1		22000	1,2		36014	4,3		36014	3,1		38000
		0.00	12.033		14.872	26.417		28.4019	46.31		47.716	49.616	

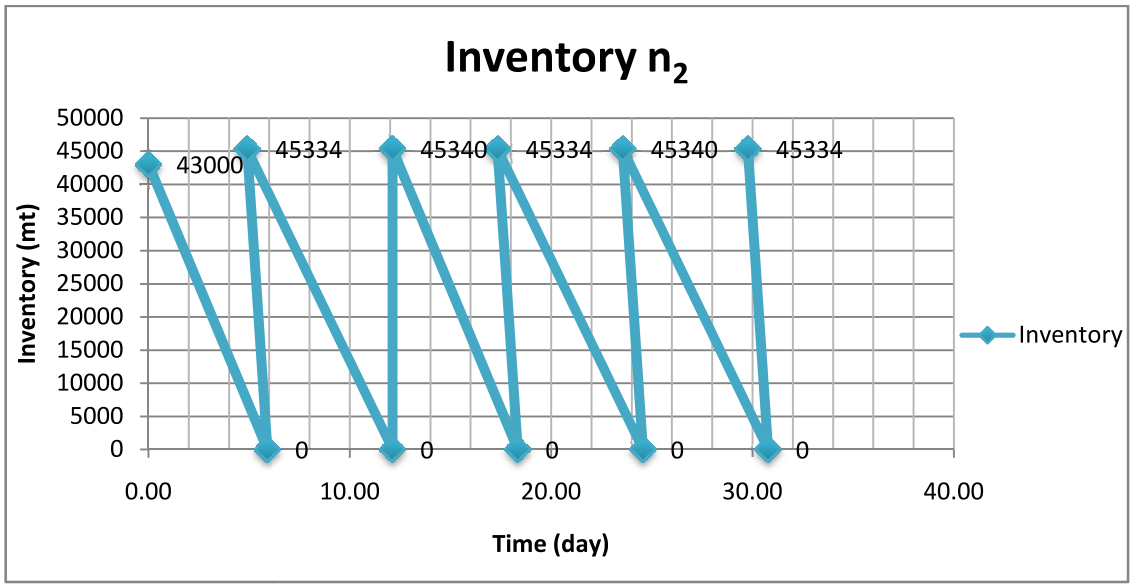
Kapal	Port Call (i,m)								
	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)	t_{im}	t_{Eim}	q_{imv} (mt)
BCH (v_1)	7,4		45334	2,6		45334	1,3		30000
	24.59	27.996		29.769	36.984		38.879	50	
MVT (v_2)									
MHS (v_3)									
BCL (v_4)	2,5		45340	7,5		45340	6,3		45340
	23.55	30.769		30.81	34.869		37.021	61.824	
GKM (v_5)									

Table 4.7 Biaya Operasi Kapal

Kapal	Total Cost (US$)	Port Call (i,m)											
		C	Cf	Cw	C	Cf	Cw	C	Cf	Cw	C	Cf	Cw
BCH (v_1)	145537.284	1,1			2,2			7,2			2,4		
		-	6448.277	799.8151	47961.51	6448.277	23835.8	965.02	6448.277	21327.78831	1018.44	6448.277	23835.8012
MVT (v_2)	363214.832	5,1			4,2			7,5			4,4		
		-	6714.618	9531.775	58161	6714.618	90771.4	20555.47	6714.618	20458.78884	23828.59	6714.618	113049.336
MHS (v_3)	287234.583	2,1			7,1			6,2			5,2		
		-	6693.261	23331.54	979.98	6693.261	18748.48	58758.37	6693.261	102797.0933	46995.53	6693.261	8850.5437
BCL (v_4)	143646.200	4,1			7,2			2,3			7,3		
		-	6448.616	33859.31	22399.41	6448.616	19648	1092.41	6448.616	20161.111	1043.5	6448.616	19647.998
GKM (v_5)	343797.683	6,1			1,2			4,3			3,1		
		-	6448.616	52292.66	74570.21	6448.616	11866.5	53892.56	6448.616	77831.83617	36938.52	6448.616	10610.93
Total =	1283430.582												

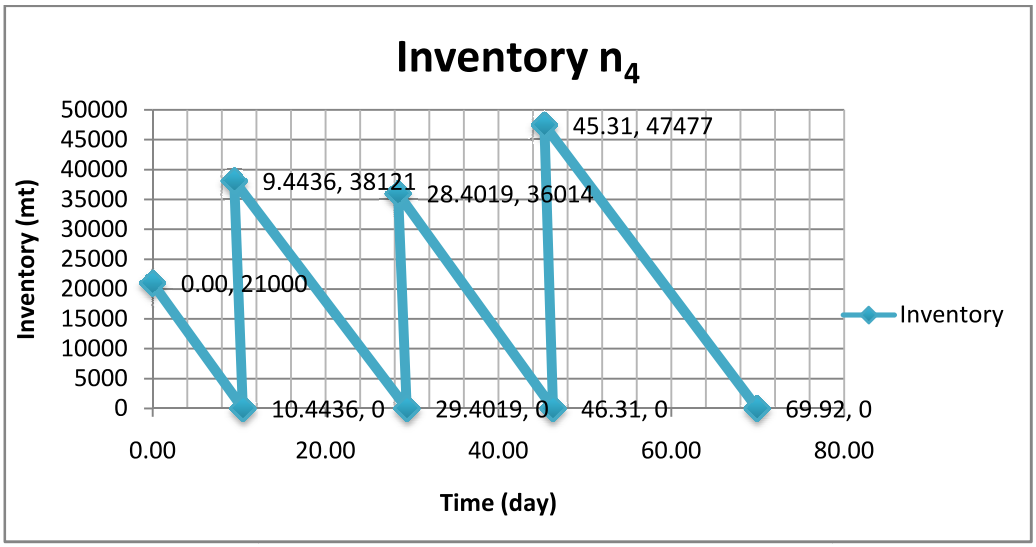
Kapal	Total Cost (US$)	Port Call (i,m)								
		C	Cf	Cw	C	Cf	Cw	C	Cf	Cw
BCH (v_1)	125544.980	7,4			2,6			1,3		
		965.02	6448.277	21327.79	1018.44	6448.277	23835.8	48772.7	6448.277	10280.4
MVT (v_2)										
MHS (v_3)										
BCL (v_4)	204377.606	2,5			7,5			6,3		
		1092.41	6448.616	20161.11	1043.5	6448.616	19648	62683.73	6448.616	80403.009
GKM (v_5)										
Total =	329922.587									
Total =	**1613353.169**									

❖ Gambaran inventori di pelabuhan *Unloading*



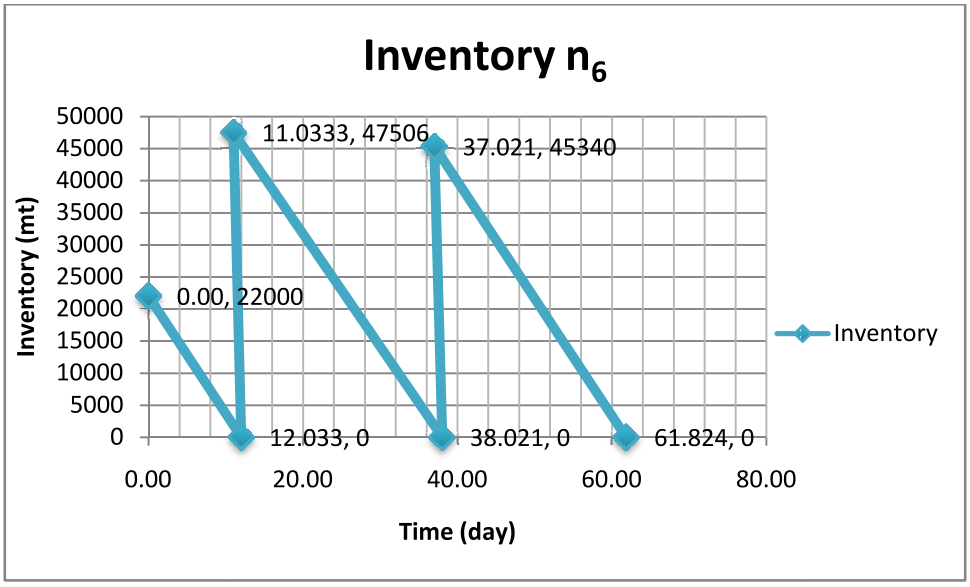
Gambar 4.9 Grafik Inventori di Pelabuhan TLS

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Tetapi pada *port call* (2,3) kapal datang tepat pada hari ke 12.1153. Hal ini dikarenakan waktu kapal yang sangat singkat untuk *loading* di pelabuhan TLS-IMP. Namun secara umum, muatan selalu tersedia di pelabuhan TLS dengan memenuhi kendala *time window*.



Gambar 4.10 Grafik Inventori di Pelabuhan BAL

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Maka disimpulkan bahwa muatan selalu tersedia di pelabuhan BAL dengan memenuhi kendala *time window*.

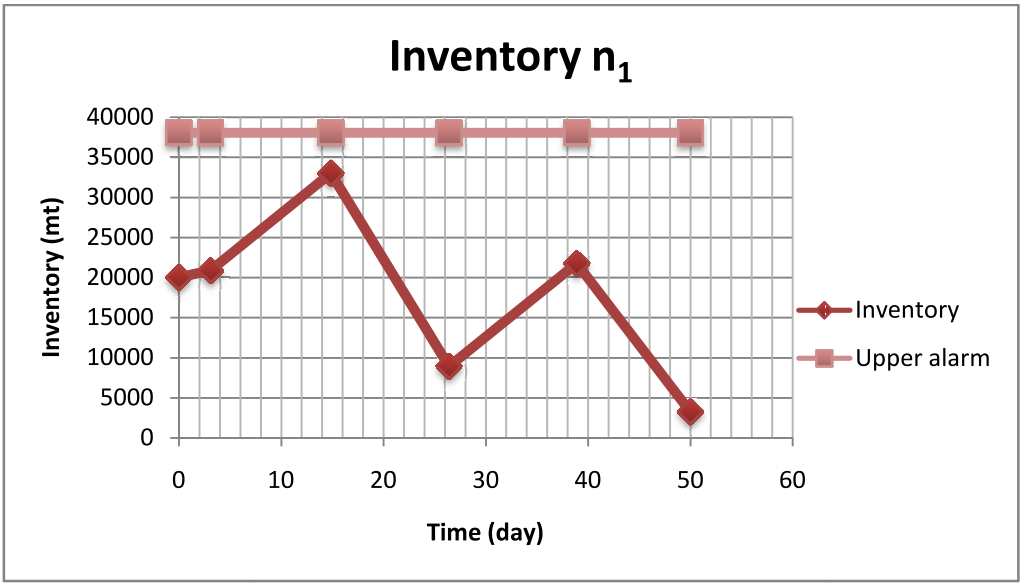


Gambar 4.11 Grafik Inventori di Pelabuhan XPN

Dari grafik di atas digambarkan bahwa kapal datang satu hari sebelum kapal sebelumnya pergi. Maka disimpulkan bahwa muatan selalu tersedia di pelabuhan XPN dengan memenuhi kendala *time window*.

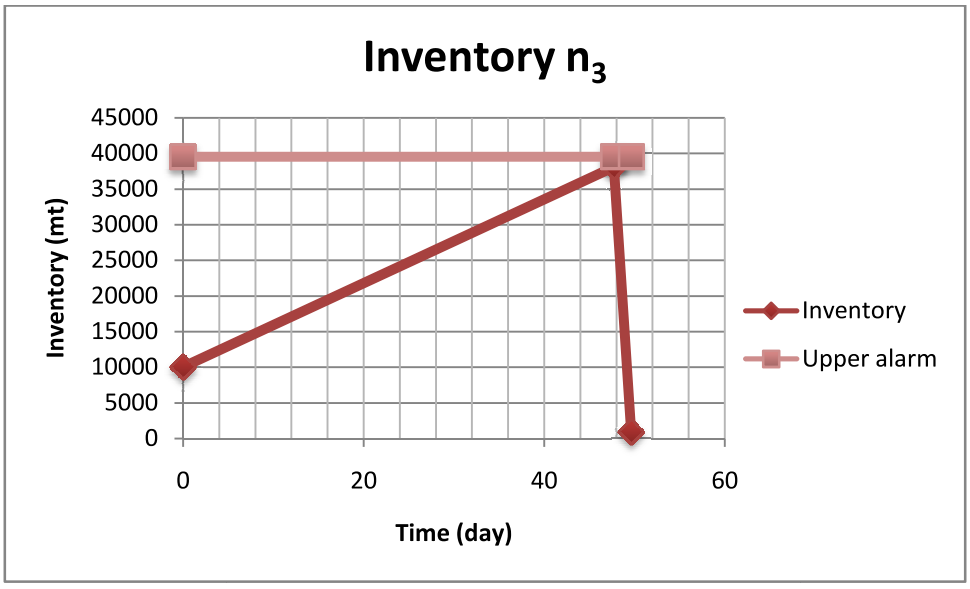
- Gambaran inventori di pelabuhan *loading*

  Salah satu kendala yang harus dihindari dari permasalahan ini adalah jika inventori pelabuhan *loading* melebihi *soft inventory level*. Kendala ini pun berhasil dihindari sehingga kapal tidak dikenai biaya pinalti. Level inventori setiap kunjungan dapat dilihat pada grafik berikut:



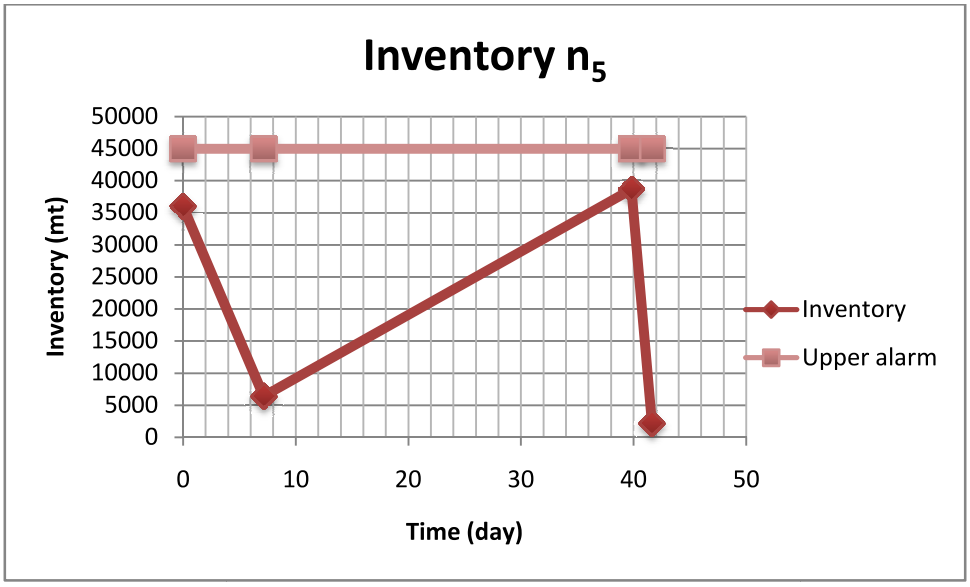
Gambar 4.12 Grafik Inventori di Pelabuhan BLN

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan BLN tidak pernah mencapai batas *soft inventori level* yaitu sebesar 38061.294 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.



Gambar 4.13 Grafik Inventori di Pelabuhan TJB

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan TJB tidak pernah mencapai batas *soft inventori level* yaitu sebesar 39509.55 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.



Gambar 4.14 Grafik Inventori di Pelabuhan BON

Pada grafik di atas dapat dilihat bahwa inventori pelabuhan BON tidak pernah mencapai batas *soft inventori level* yaitu sebesar 44965.438 mt. Oleh karena itu selama periode perencanaan, pelabuhan ini tidak pernah terkena biaya pinalti.

BAB 5

KESIMPULAN DAN SARAN

Penelitian ini menghasilkan suatu rute dan jadwal kapal yang berbeda dengan yang diterapkan perusahaan saat ini. Berdasarkan rute dalan jadwal kapal yang diperoleh melalui pengolahan data menggunakan algoritma *Tabu Search,* diperoleh kesimpulan sebagai berikut:

- Diperoleh model optimasi penentuan rute dan jadwal kapal yang optimal, yang dapat memenuhi setiap permintaan pada pelabuhan *unloading* dan menjaga inventori pelabuhan *loading* sehingga tidak terkena biaya pinalti dan didapatkan biaya yang minimum.
- Hasil pengolahan data dengan menggunakan algoritma *tabu search* dan bahasa pemrograman pada Matlab memperoleh hasil dengan biaya yang lebih minimum dibandingkan dengan hasil penelitian sebelumnya yang menggunakan metode *Branch and Bound* dengan LINGO, dimana waktu *running* program yang dibutuhkan jauh lebih cepat. Selain itu karena keterbatasan jumlah variabel integer, program LINGO harus 2 kali *running* untuk memperoleh periode perencanaan 30 hari (2x15 hari), sedangkan dengan Matlab, periode perencanaan dapat dibuat sesuai dengan yang diinginkan karena tidak adanya keterbatasan variabel integer.
- Dari hasil *running* program optimasi yang dilakukan, dapat disimpulkan bahwa model yang dikembangkan memiliki performansi yang cukup baik dibandingkan dengan kondisi *existing*.

Adapun saran perbaikan dan pengembangan untuk penelitian selanjutnya adalah sebagai berikut:

- Mengembangkan penelitian lebih lanjut dengan menggunakan metode algoritma yang lain untuk mengetahui perbandingan terhadap hasilnya, yaitu berupa sistem distribusi yang efektif dan efisien meliputi penentuan rute distribusi dan penjadwalan kapal sehingga menghasilkan biaya yang minimum dengan memenuhi semua kendala.

DAFTAR REFERENSI

Antoni, Akbar. (2011). Integrasi *Inventory* dan *Ship Scheduling* dengan *Time Window Constraint* pada *Mobile Floating Storage* dengan Model *Mix Integer Programming.*

Ballou, Ronald H. (1992). *Business Logistics Management* 4th ed. Prentice-Hall. Inc. New Jersey.

Christiansen et al. (2010). *Operation Research: Maritime inventory routing with multiple products: A case study from the cement industry.*

Fagerholt et al. (2009). *Operations Research: Maritime Inventory Routing Problems.*

Khayyal et al. (2005). *Operational Research: Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model.*

Korsvik et al. (2010). *Operational Research: A Tabu Search Heuristic for Ship Routing and Scheduling.*

www.pertamina.com, 1 Maret 2012. 15.00.

LAMPIRAN

Bahasa Program MATLAB

```
% function
[Data_Muatan_Baru,Data_Isi_Pelabuhan_Setelah_Loading,Data_Waktu_Ba
ru,Biaya_FOC_Baru,Sea_Time_Baru,Schedule_Pelabuhan,Time_Elapsed_Ba
ru] =
Inverse_Solusi(XCL_Kapasitas_Kapal,XCL_Muatan_Awal_Kapal,XCL_Data_
Sea_Time,XCL_Stock_Awal_Pelabuhan,XCL_Waktu_Load_Unload,XCL_Data_F
OC,XCL_Production_Consumption_Rate,Solusi_Baru,rute,kapal)
Solusi_Baru = Solusi_Awal;

Data_Muatan_Baru = [];
Sea_Time_Baru = zeros(kapal,1);
Data_Waktu_Baru = zeros(kapal,1);

Kunjungan_Pelabuhan = ones(7,1);
Schedule_Pelabuhan = zeros(7,2);
Rute_Finish = ones(5,1);

Isi_Tangki_Kapal_Sekarang       = XCL_Muatan_Awal_Kapal;
Isi_Pelabuhan_Loading_Sekarang  = XCL_Stock_Awal_Pelabuhan;
Time_Elapsed_Baru               = zeros(5,1);
Time_Window                     = 1;

Waktu_Terakhir_Loading_Baru     = zeros(3,1);

Data_Isi_Pelabuhan_Setelah_Loading      = zeros(4,rute);
Data_Isi_Pelabuhan_Setelah_Loading(4,:) = inf;

Task = (rute-1)*kapal;

for r = 1
    for k = 1:kapal
        % Hitung Muatan & Biaya
        % ========================================

        % 1)Pelabuhan Loading BLN ----------------------------
----------------
        if  Solusi_Baru(k,r) == 1
            % Cek sisa ruang kosong di tangki kapal

Production_Time                     = Time_Elapsed_Baru(k,1) -
Waktu_Terakhir_Loading_Baru(1,1);
            Isi_Pelabuhan_Loading_Sekarang(1,1) =
Isi_Pelabuhan_Loading_Sekarang(1,1) +
XCL_Production_Consumption_Rate(1,1)*Production_Time;
            Masih_Kosong                        =
XCL_Kapasitas_Kapal(k,1) - Isi_Tangki_Kapal_Sekarang(k,1);

            if  Isi_Pelabuhan_Loading_Sekarang(1,1) < Masih_Kosong
                Waktu_Sinergis                          =
```

```
(XCL_Waktu_Load_Unload(1,1)*Isi_Pelabuhan_Loading_Sekarang(1,1))/(
1-
XCL_Waktu_Load_Unload(1,1)*XCL_Production_Consumption_Rate(1,1));
                Isi_Pelabuhan_Loading_Sekarang(1,1)     =
Isi_Pelabuhan_Loading_Sekarang(1,1) +
XCL_Production_Consumption_Rate(1,1)* Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) +
Isi_Pelabuhan_Loading_Sekarang(1,1);
                Biaya_FOC_Baru(k,r)                     =
Isi_Pelabuhan_Loading_Sekarang(1,1)*XCL_Data_FOC(1,k);
                Data_Muatan_Baru(k,r)                   =
Isi_Pelabuhan_Loading_Sekarang(1,1);
                Isi_Pelabuhan_Loading_Sekarang(1,1)     = 0;
            else
                Waktu_Sinergis                          =
Masih_Kosong*XCL_Waktu_Load_Unload(1,1);
                Isi_Pelabuhan_Loading_Sekarang(1,1)     =
Isi_Pelabuhan_Loading_Sekarang(1,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(1,1)*Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) + Masih_Kosong;
                Biaya_FOC_Baru(k,r)                     =
Masih_Kosong*XCL_Data_FOC(1,k);
                Data_Muatan_Baru(k,r)                   =
Masih_Kosong;
            end
            Data_Waktu_Baru(k,r)                        =
Waktu_Sinergis;
            Schedule_Pelabuhan(1,2*Kunjungan_Pelabuhan) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(1,1) = Kunjungan_Pelabuhan(1,1) +
1;
            Data_Isi_Pelabuhan_Setelah_Loading(k,r)     =
Isi_Pelabuhan_Loading_Sekarang(1,1);
            Waktu_Terakhir_Loading_Baru(1,1)            =
Time_Elapsed_Baru(k,1) + Waktu_Sinergis;
            Time_Elapsed_Baru(k,1)                      =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
        end

        % 2)Pelabuhan Unloading TLS ------------------------------
------------------
        if  Solusi_Baru(k,r) == 2
            unloading                               =
Isi_Tangki_Kapal_Sekarang(k,1);
            Biaya_FOC_Baru(k,r)                     =
unloading*XCL_Data_FOC(5,k);
            Isi_Tangki_Kapal_Sekarang(k,1)          = 0; % Karena di-
unload sampai habis
            Data_Muatan_Baru(k,r)                   = -unloading;
            Data_Waktu_Baru(k,r)                    =
abs(unloading*XCL_Waktu_Load_Unload(5,1));
            Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(2,1)                    =
Kunjungan_Pelabuhan(2,1) + 1;
            Time_Elapsed_Baru(k,1)                      =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
```

```
            unloading = [];
        end

        % 3)Pelabuhan Loading TJB ------------------------------
-----------------
        if  Solusi_Baru(k,r) == 3
            % Cek sisa ruang kosong di tangki kapal
            Production_Time                     =
Time_Elapsed_Baru(k,1) - Waktu_Terakhir_Loading_Baru(2,1);
            Isi_Pelabuhan_Loading_Sekarang(2,1) =
Isi_Pelabuhan_Loading_Sekarang(2,1) +
XCL_Production_Consumption_Rate(2,1)*Production_Time;
            Masih_Kosong                        =
XCL_Kapasitas_Kapal(k,1) - Isi_Tangki_Kapal_Sekarang(k,1);

            if  Isi_Pelabuhan_Loading_Sekarang(2,1) < Masih_Kosong
                Waktu_Sinergis                          =
(XCL_Waktu_Load_Unload(2,1)*Isi_Pelabuhan_Loading_Sekarang(2,1))/(
1-
XCL_Waktu_Load_Unload(2,1)*XCL_Production_Consumption_Rate(2,1));
                Isi_Pelabuhan_Loading_Sekarang(2,1)     =
Isi_Pelabuhan_Loading_Sekarang(2,1) +
XCL_Production_Consumption_Rate(2,1)* Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) +
Isi_Pelabuhan_Loading_Sekarang(2,1);
                Biaya_FOC_Baru(k,r)                     =
Isi_Pelabuhan_Loading_Sekarang(2,1)*XCL_Data_FOC(2,k);
                Data_Muatan_Baru(k,r)                   =
Isi_Pelabuhan_Loading_Sekarang(2,1);
                Isi_Pelabuhan_Loading_Sekarang(2,1)     = 0;
            else
                Waktu_Sinergis                          =
Masih_Kosong*XCL_Waktu_Load_Unload(2,1);
                Isi_Pelabuhan_Loading_Sekarang(2,1)     =
Isi_Pelabuhan_Loading_Sekarang(2,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(2,1)*Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) + Masih_Kosong;
                Biaya_FOC_Baru(k,r)                     =
Masih_Kosong*XCL_Data_FOC(2,k);
                Data_Muatan_Baru(k,r)                   =
Masih_Kosong;
            end
            Data_Waktu_Baru(k,r)                = Waktu_Sinergis;
            Schedule_Pelabuhan(3,2*Kunjungan_Pelabuhan(3,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(3,1)                    =
Kunjungan_Pelabuhan(3,1) + 1;
            Data_Isi_Pelabuhan_Setelah_Loading(k,r) =
Isi_Pelabuhan_Loading_Sekarang(2,1);
            Waktu_Terakhir_Loading_Baru(2,1)    =
Time_Elapsed_Baru(k,1) + Waktu_Sinergis;
            Time_Elapsed_Baru(k,1)              =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
        end
```

```
        % 4)Pelabuhan Unloading BAL  ----------------------------
--------------------
        if  Solusi_Baru(k,r) == 4
            unloading                           =
Isi_Tangki_Kapal_Sekarang(k,1);
            Biaya_FOC_Baru(k,r)                 =
unloading*XCL_Data_FOC(6,k);
            Isi_Tangki_Kapal_Sekarang(k,1)      = 0;
            Data_Muatan_Baru(k,r)               = -unloading;
            Data_Waktu_Baru(k,r)                =
abs(unloading*XCL_Waktu_Load_Unload(6,1));
            Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(4,1)                    =
Kunjungan_Pelabuhan(4,1) + 1;
            Time_Elapsed_Baru(k,1)              =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
        end

        % 5)Pelabuhan Loading BON  -------------------------------
------------------
        if  Solusi_Baru(k,r) == 5
            Production_Time                     =
Time_Elapsed_Baru(k,1) - Waktu_Terakhir_Loading_Baru(3,1);
            Isi_Pelabuhan_Loading_Sekarang(3,1) =
Isi_Pelabuhan_Loading_Sekarang(3,1) +
XCL_Production_Consumption_Rate(3,1)*Production_Time;
            Masih_Kosong                        =
XCL_Kapasitas_Kapal(k,1) - Isi_Tangki_Kapal_Sekarang(k,1);

            if  Isi_Pelabuhan_Loading_Sekarang(3,1) < Masih_Kosong
                Waktu_Sinergis                          =
(XCL_Waktu_Load_Unload(3,1)*Isi_Pelabuhan_Loading_Sekarang(3,1))/(
1-
XCL_Waktu_Load_Unload(3,1)*XCL_Production_Consumption_Rate(3,1));
                Isi_Pelabuhan_Loading_Sekarang(3,1)     =
Isi_Pelabuhan_Loading_Sekarang(3,1) +
XCL_Production_Consumption_Rate(3,1)* Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) +
Isi_Pelabuhan_Loading_Sekarang(3,1);
                Biaya_FOC_Baru(k,r)                     =
Isi_Pelabuhan_Loading_Sekarang(3,1)*XCL_Data_FOC(3,k);
                Data_Muatan_Baru(k,r)                   =
Isi_Pelabuhan_Loading_Sekarang(3,1);
                Isi_Pelabuhan_Loading_Sekarang(3,1)     = 0;
            else
                Waktu_Sinergis                          =
Masih_Kosong*XCL_Waktu_Load_Unload(3,1);
                Isi_Pelabuhan_Loading_Sekarang(3,1)     =
Isi_Pelabuhan_Loading_Sekarang(3,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(3,1)*Waktu_Sinergis;
                Isi_Tangki_Kapal_Sekarang(k,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) + Masih_Kosong;
                Biaya_FOC_Baru(k,r)                     =
Masih_Kosong*XCL_Data_FOC(3,k);
                Data_Muatan_Baru(k,r)                   =
Masih_Kosong;
```

```
            end
            Data_Waktu_Baru(k,r)                  = Waktu_Sinergis;
            Schedule_Pelabuhan(5,2*Kunjungan_Pelabuhan(5,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(5,1)                      =
Kunjungan_Pelabuhan(5,1) + 1;
            Data_Isi_Pelabuhan_Setelah_Loading(k,r) =
Isi_Pelabuhan_Loading_Sekarang(1,1);
            Waktu_Terakhir_Loading_Baru(3,1)      =
Time_Elapsed_Baru(k,1) + Waktu_Sinergis;
            Time_Elapsed_Baru(k,1)                =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
        end
        % 6)Pelabuhan Unloading XPN   ----------------------------
---------------------
        if  Solusi_Baru(k,r) == 6
            unloading                             =
Isi_Tangki_Kapal_Sekarang(k,1);
            Biaya_FOC_Baru(k,r)                   =
unloading*XCL_Data_FOC(7,k);
            Isi_Tangki_Kapal_Sekarang(k,1)        = 0;
            Data_Muatan_Baru(k,r)                 = -unloading;
            Data_Waktu_Baru(k,r)                  =
abs(unloading*XCL_Waktu_Load_Unload(7,1));
            Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(6,1)                      =
Kunjungan_Pelabuhan(6,1) + 1;
            Time_Elapsed_Baru(k,1)                =
Data_Waktu_Baru(k,r);
        end

        % 7)Pelabuhan Loading TLS (IMPORT) -----------------------
---------------------
        if  Solusi_Baru(k,r) == 7
            loading = XCL_Kapasitas_Kapal(k,1) -
Isi_Tangki_Kapal_Sekarang(k,1);
            if  loading > Isi_Pelabuhan_Loading_Sekarang(4,1)
                loading = Isi_Pelabuhan_Loading_Sekarang(4,1);
                Isi_Pelabuhan_Loading_Sekarang(4,1) = 0;
            else
                Isi_Pelabuhan_Loading_Sekarang(4,1) =
Isi_Pelabuhan_Loading_Sekarang(4,1) - loading;
            end
            Data_Isi_Pelabuhan_Setelah_Loading(k,r) = inf;
            Biaya_FOC_Baru(k,r)                   =
loading*XCL_Data_FOC(4,k);
            Isi_Tangki_Kapal_Sekarang(k,1)        =
Isi_Tangki_Kapal_Sekarang(k,1) + loading;
            Data_Muatan_Baru(k,r)                 = loading;
            Data_Waktu_Baru(k,r)                  =
abs(loading*XCL_Waktu_Load_Unload(4,1));
            Schedule_Pelabuhan(7,2*Kunjungan_Pelabuhan(7,1)) =
Data_Waktu_Baru(k,r);
            Kunjungan_Pelabuhan(7,1)                      =
Kunjungan_Pelabuhan(7,1) + 1;
            Time_Elapsed_Baru(k,1)                =
Time_Elapsed_Baru(k,1) + Data_Waktu_Baru(k,r);
        end
```

```
        if r < rute
            for b=1:24
                if XCL_Data_Sea_Time(b,2:3) ==
Solusi_Baru(k,r:r+1);
                    lama = XCL_Data_Sea_Time(b,1);
                end
            end
            Sea_Time_Baru(k,r) = lama/24;
            Time_Elapsed_Baru(k,1) = Time_Elapsed_Baru(k,1) +
Sea_Time_Baru(k,r);
        end
    end
end

% Create rute kedua dan seterusnya ====================
% Karena rute kedua sangat kompleks ====================


for T = 1:Task
    Belum_Selesai        = find(Rute_Finish ~= rute);
    Waktu_Tercepat       =
min(Time_Elapsed_Baru(Belum_Selesai,1));
    Kapal_Selesai        = find(Time_Elapsed_Baru ==
Waktu_Tercepat);
    Cek_Finished_Route   = Rute_Finish(Kapal_Selesai,:);
    Cek_Next_Rute        =
Solusi_Baru(Kapal_Selesai,Cek_Finished_Route+1);
    Rute_Finish(Kapal_Selesai,1) = Rute_Finish(Kapal_Selesai,1) +
1;
    Rute_Sekarang        = Rute_Finish(Kapal_Selesai,1);

    % 1)Pelabuhan Loading BLN ----------------------------------
-------------
    if  Cek_Next_Rute == 1
        % Cek sisa ruang kosong di tangki kapal
        Schedule_Pelabuhan(1,2*Kunjungan_Pelabuhan(1,1)-1)    =
Time_Elapsed_Baru(Kapal_Selesai,1);
        Production_Time                    =
Time_Elapsed_Baru(Kapal_Selesai,1) -
Waktu_Terakhir_Loading_Baru(1,1);
        Isi_Pelabuhan_Loading_Sekarang(1,1) =
Isi_Pelabuhan_Loading_Sekarang(1,1) +
XCL_Production_Consumption_Rate(1,1)*Production_Time;
        Masih_Kosong                       =
XCL_Kapasitas_Kapal(Kapal_Selesai,1) -
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);

        if  Isi_Pelabuhan_Loading_Sekarang(1,1) < Masih_Kosong
            Waktu_Sinergis                                 =
(XCL_Waktu_Load_Unload(1,1)*Isi_Pelabuhan_Loading_Sekarang(1,1))/(
1-
XCL_Waktu_Load_Unload(1,1)*XCL_Production_Consumption_Rate(1,1));
            Isi_Pelabuhan_Loading_Sekarang(1,1)         =
Isi_Pelabuhan_Loading_Sekarang(1,1) +
XCL_Production_Consumption_Rate(1,1)* Waktu_Sinergis;
```

```
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)  =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) +
Isi_Pelabuhan_Loading_Sekarang(1,1);
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)
=
Isi_Pelabuhan_Loading_Sekarang(1,1)*XCL_Data_FOC(1,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)
= Isi_Pelabuhan_Loading_Sekarang(1,1);
            Isi_Pelabuhan_Loading_Sekarang(1,1)     = 0;
        else
            Waktu_Sinergis                                  =
Masih_Kosong*XCL_Waktu_Load_Unload(1,1);
            Isi_Pelabuhan_Loading_Sekarang(1,1)             =
Isi_Pelabuhan_Loading_Sekarang(1,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(1,1)*Waktu_Sinergis;
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)      =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) + Masih_Kosong;
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)     =
Masih_Kosong*XCL_Data_FOC(1,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)   =
Masih_Kosong;
        end
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)
= Waktu_Sinergis;
        Schedule_Pelabuhan(1,2*Kunjungan_Pelabuhan(1,1))
= Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(1,1)
= Kunjungan_Pelabuhan(1,1) + 1;

Data_Isi_Pelabuhan_Setelah_Loading(Kapal_Selesai,Rute_Sekarang) =
Isi_Pelabuhan_Loading_Sekarang(1,1);
        Waktu_Terakhir_Loading_Baru(1,1)
= Time_Elapsed_Baru(Kapal_Selesai,1) + Waktu_Sinergis;
        Time_Elapsed_Baru(Kapal_Selesai,1)
= Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end

    % 2)Pelabuhan Unloading TLS ---------------------------------
---------------
    if  Cek_Next_Rute == 2
        % Apply Time Window
        if Time_Elapsed_Baru(Kapal_Selesai,1) <
Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)-2)
            Time_Elapsed_Baru(Kapal_Selesai,1) =
Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)-2);
            Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1) - Time_Window;
        else
%             Percepat = Time_Elapsed_Baru(Kapal_Selesai,1) -
Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)-2)
            Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1);
        end
        unloading                                           =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);
        Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)         =
unloading*XCL_Data_FOC(5,Kapal_Selesai);
```

```
        Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)           = 0; %
Karena di-unload sampai habis
        Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)        = -
unloading;
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)         =
abs(unloading*XCL_Waktu_Load_Unload(5,1));
        Schedule_Pelabuhan(2,2*Kunjungan_Pelabuhan(2,1))     =
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(2,1)                             =
Kunjungan_Pelabuhan(2,1) + 1;
        Time_Elapsed_Baru(Kapal_Selesai,1)                   =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
        unloading = [];
    end

    % 3)Pelabuhan Loading TJB ----------------------------------
-------------
    if  Cek_Next_Rute == 3
        Schedule_Pelabuhan(3,2*Kunjungan_Pelabuhan(3,1)-1)   =
Time_Elapsed_Baru(Kapal_Selesai,1);
        Production_Time                                      =
Time_Elapsed_Baru(Kapal_Selesai,1) -
Waktu_Terakhir_Loading_Baru(2,1);
        Isi_Pelabuhan_Loading_Sekarang(2,1)                  =
Isi_Pelabuhan_Loading_Sekarang(2,1) +
XCL_Production_Consumption_Rate(2,1)*Production_Time;
        Masih_Kosong                                         =
XCL_Kapasitas_Kapal(Kapal_Selesai,1) -
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);

        if  Isi_Pelabuhan_Loading_Sekarang(2,1) < Masih_Kosong
            Waktu_Sinergis                                   =
(XCL_Waktu_Load_Unload(2,1)*Isi_Pelabuhan_Loading_Sekarang(2,1))/(
1-
XCL_Waktu_Load_Unload(2,1)*XCL_Production_Consumption_Rate(2,1));
            Isi_Pelabuhan_Loading_Sekarang(2,1)      =
Isi_Pelabuhan_Loading_Sekarang(2,1) +
XCL_Production_Consumption_Rate(2,1)* Waktu_Sinergis;
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)           =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) +
Isi_Pelabuhan_Loading_Sekarang(2,1);
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)
=
Isi_Pelabuhan_Loading_Sekarang(2,1)*XCL_Data_FOC(2,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)
= Isi_Pelabuhan_Loading_Sekarang(2,1);
            Isi_Pelabuhan_Loading_Sekarang(2,1)      = 0;
        else
            Waktu_Sinergis                                   =
Masih_Kosong*XCL_Waktu_Load_Unload(2,1);
            Isi_Pelabuhan_Loading_Sekarang(2,1)          =
Isi_Pelabuhan_Loading_Sekarang(2,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(2,1)*Waktu_Sinergis;
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)           =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) + Masih_Kosong;
```

```
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)
= Masih_Kosong*XCL_Data_FOC(2,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)
= Masih_Kosong;
        end
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)
= Waktu_Sinergis;
        Schedule_Pelabuhan(3,2*Kunjungan_Pelabuhan(3,1))
= Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(3,1)
= Kunjungan_Pelabuhan(3,1) + 1;

Data_Isi_Pelabuhan_Setelah_Loading(Kapal_Selesai,Rute_Sekarang) =
Isi_Pelabuhan_Loading_Sekarang(2,1);
        Waktu_Terakhir_Loading_Baru(2,1)
= Time_Elapsed_Baru(Kapal_Selesai,1) + Waktu_Sinergis;
        Time_Elapsed_Baru(Kapal_Selesai,1)
= Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end

    % 4)Pelabuhan Unloading BAL  ---------------------------------
----------------
    if  Cek_Next_Rute == 4
        if Time_Elapsed_Baru(Kapal_Selesai,1) <
Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1)-2)
            Time_Elapsed_Baru(Kapal_Selesai,1) =
Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1)-2);
            Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1) - Time_Window;
        else
            Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1);
        end
        unloading                                           =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);
        Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)         =
unloading*XCL_Data_FOC(6,Kapal_Selesai);
        Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)          = 0;
        Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)       = -
unloading;
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)        =
abs(unloading*XCL_Waktu_Load_Unload(6,1));
        Schedule_Pelabuhan(4,2*Kunjungan_Pelabuhan(4,1))    =
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(4,1)                            =
Kunjungan_Pelabuhan(4,1) + 1;
        Time_Elapsed_Baru(Kapal_Selesai,1)                  =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end

    % 5)Pelabuhan Loading BON  -----------------------------------
--------------
    if  Cek_Next_Rute == 5
```

```
        Schedule_Pelabuhan(5,2*Kunjungan_Pelabuhan(5,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1);
        Production_Time                                     =
Time_Elapsed_Baru(Kapal_Selesai,1) -
Waktu_Terakhir_Loading_Baru(3,1);
        Isi_Pelabuhan_Loading_Sekarang(3,1)                 =
Isi_Pelabuhan_Loading_Sekarang(3,1) +
XCL_Production_Consumption_Rate(3,1)*Production_Time;
        Masih_Kosong                                        =
XCL_Kapasitas_Kapal(Kapal_Selesai,1) -
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);

        if  Isi_Pelabuhan_Loading_Sekarang(3,1) < Masih_Kosong
            Waktu_Sinergis                                  =
(XCL_Waktu_Load_Unload(3,1)*Isi_Pelabuhan_Loading_Sekarang(3,1))/(
1-
XCL_Waktu_Load_Unload(3,1)*XCL_Production_Consumption_Rate(3,1));
            Isi_Pelabuhan_Loading_Sekarang(3,1)     =
Isi_Pelabuhan_Loading_Sekarang(3,1) +
XCL_Production_Consumption_Rate(3,1)* Waktu_Sinergis;
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)          =
Isi_Tangki_Kapal_Sekarang(k,1) +
Isi_Pelabuhan_Loading_Sekarang(3,1);
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)
=
Isi_Pelabuhan_Loading_Sekarang(3,1)*XCL_Data_FOC(3,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)
= Isi_Pelabuhan_Loading_Sekarang(3,1);
            Isi_Pelabuhan_Loading_Sekarang(3,1)     = 0;
        else
            Waktu_Sinergis                                  =
Masih_Kosong*XCL_Waktu_Load_Unload(3,1);
            Isi_Pelabuhan_Loading_Sekarang(3,1)     =
Isi_Pelabuhan_Loading_Sekarang(3,1) - Masih_Kosong +
XCL_Production_Consumption_Rate(3,1)*Waktu_Sinergis;
            Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)          =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) + Masih_Kosong;
            Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)
= Masih_Kosong*XCL_Data_FOC(3,Kapal_Selesai);
            Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)
= Masih_Kosong;
        end
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)    =
Waktu_Sinergis;
        Schedule_Pelabuhan(5,2*Kunjungan_Pelabuhan(5,1))    =
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(5,1)                            =
Kunjungan_Pelabuhan(5,1) + 1;

Data_Isi_Pelabuhan_Setelah_Loading(Kapal_Selesai,Rute_Sekarang) =
Isi_Pelabuhan_Loading_Sekarang(1,1);
        Waktu_Terakhir_Loading_Baru(3,1)                =
Time_Elapsed_Baru(Kapal_Selesai,1) + Waktu_Sinergis;
        Time_Elapsed_Baru(Kapal_Selesai,1)              =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end
```

```
    % 6)Pelabuhan Unloading XPN   ------------------------------
-----------------
    if  Cek_Next_Rute == 6
        if Time_Elapsed_Baru(Kapal_Selesai,1) <
Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1)-2)
            Time_Elapsed_Baru(Kapal_Selesai,1) =
Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1)-2);
            Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1) - Time_Window;
        else
            Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1);
        end
        unloading                                           =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);
        Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)         =
unloading*XCL_Data_FOC(7,Kapal_Selesai);
        Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)          = 0;
        Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)       = -
unloading;
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)        =
abs(unloading*XCL_Waktu_Load_Unload(7,1));
        Schedule_Pelabuhan(6,2*Kunjungan_Pelabuhan(6,1))    =
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(6,1)                            =
Kunjungan_Pelabuhan(6,1) + 1;
        Time_Elapsed_Baru(Kapal_Selesai,1)                  =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end

    % 7)Pelabuhan Loading TLS (IMPORT) ---------------------------
-----------------
    if  Cek_Next_Rute == 7
        Schedule_Pelabuhan(7,2*Kunjungan_Pelabuhan(7,1)-1)  =
Time_Elapsed_Baru(Kapal_Selesai,1);
        loading = XCL_Kapasitas_Kapal(Kapal_Selesai,1) -
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1);
        if  loading > Isi_Pelabuhan_Loading_Sekarang(4,1)
            loading = Isi_Pelabuhan_Loading_Sekarang(4,1);
            Isi_Pelabuhan_Loading_Sekarang(4,1) = 0;
        else
            Isi_Pelabuhan_Loading_Sekarang(4,1) =
Isi_Pelabuhan_Loading_Sekarang(4,1) - loading;
        end
        Data_Isi_Pelabuhan_Setelah_Loading(Kapal_Selesai,r) = inf;
        Biaya_FOC_Baru(Kapal_Selesai,Rute_Sekarang)     =
loading*XCL_Data_FOC(4,Kapal_Selesai);
        Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1)      =
Isi_Tangki_Kapal_Sekarang(Kapal_Selesai,1) + loading;
        Data_Muatan_Baru(Kapal_Selesai,Rute_Sekarang)   = loading;
        Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang)    =
abs(loading*XCL_Waktu_Load_Unload(4,1));
        Schedule_Pelabuhan(7,2*Kunjungan_Pelabuhan(7,1))=
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang) +
Time_Elapsed_Baru(Kapal_Selesai,1);
        Kunjungan_Pelabuhan(7,1)                        =
Kunjungan_Pelabuhan(7,1) + 1;
```

```
        Time_Elapsed_Baru(Kapal_Selesai,1)              =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Data_Waktu_Baru(Kapal_Selesai,Rute_Sekarang);
    end
    if Rute_Sekarang <6
        for b=1:24
            if XCL_Data_Sea_Time(b,2:3) ==
Solusi_Baru(Kapal_Selesai,Rute_Sekarang:Rute_Sekarang+1);
                lama = XCL_Data_Sea_Time(b,1);
            end
        end
        Sea_Time_Baru(Kapal_Selesai,Rute_Sekarang) = lama/24;
        Time_Elapsed_Baru(Kapal_Selesai,1) =
Time_Elapsed_Baru(Kapal_Selesai,1) +
Sea_Time_Baru(Kapal_Selesai,Rute_Sekarang);
    end
end



clear;
clc;
kapal = 5;
rute = 3;

tic;
disp('======== ~(^0^ ~) Program Skripsi Gaby (~ ^0^)~ ==========')

% 1) Create Solusi Awal
% => IMPORT DATA EXCEL ----------------------------------------
---------
XCL_Data_Sea_Time               = xlsread('data.xlsx','Sea time
(jam)','A4:D27');
XCL_Kapasitas_Kapal             = xlsread('data.xlsx','Kapasitas
kapal (CAPv)','E4:E8');
XCL_Muatan_Awal_Kapal           = xlsread('data.xlsx','Muatan awal
kapal (Qv)','D4:D8');
XCL_Data_FOC                    =
xlsread('data.xlsx','FOC','C3:G9');
XCL_Stock_Awal_Pelabuhan        = xlsread('data.xlsx','Stock awal
pelabuhan (ISi)','D4:D7');
XCL_Stock_Awal_Pelabuhan(4,1)   = inf;   %Karena stok import tak
terhingga
XCL_Waktu_Load_Unload           = xlsread('data.xlsx','Waktu
un(loading) muatan (TQi)','C4:C10');
XCL_Production_Consumption_Rate =
xlsread('data.xlsx','Ri','D5:D7');

% Generate Solusi Rute 1: -------------------------------------
---------
[solusi_awal_terbaik,Isi_Tangki_Kapal_Sekarang,Isi_Pelabuhan_Loadi
ng_Sekarang,Biaya_FOC,Data_Muatan,Data_Waktu,Waktu_Occupied_Pelabu
han,Waktu_Terakhir_Loading,Time_Elapsed] =
Get_Solusi_Rute_1(XCL_Kapasitas_Kapal,XCL_Muatan_Awal_Kapal,XCL_Da
ta_FOC,XCL_Stock_Awal_Pelabuhan,XCL_Waktu_Load_Unload,XCL_Producti
on_Consumption_Rate,rute,kapal)
Solusi_Awal     = solusi_awal_terbaik;
Datang_Pergi(:,1:2) = Waktu_Occupied_Pelabuhan(:,1:2);
```

```
% Generate Solusi Rute 2 dst: ----------------------------------
---------
Rute_Sekarang = 1;
for r = 2:rute
    Rute_Sekarang = Rute_Sekarang + 1;
    disp(['======================== Rute Berikutnya
======================== '])

[solusi_rute,Sea_Time_Rute,Waktu_Occupied_Pelabuhan,Data_Waktu_Seb
elumnya_ADJUSTED,Data_Muatan_BARU,Data_Waktu_BARU,Biaya_FOC_BARU,I
si_Tangki_Kapal_Sekarang,Isi_Pelabuhan_Loading_Sekarang,Time_Elaps
ed,Waktu_Terakhir_Loading] =
Create_Smart_Solution_Solver(XCL_Data_Sea_Time,XCL_Production_Cons
umption_Rate,XCL_Data_FOC,XCL_Waktu_Load_Unload,XCL_Kapasitas_Kapa
l,Data_Waktu(:,r-1),Data_Muatan(:,r-1),kapal,Solusi_Awal(:,r-
1),Waktu_Occupied_Pelabuhan,Waktu_Terakhir_Loading,Isi_Pelabuhan_L
oading_Sekarang,Isi_Tangki_Kapal_Sekarang,Time_Elapsed);
    Solusi_Awal(:,r)     = solusi_rute
    Sea_Time(:,r-1)      = Sea_Time_Rute
    Data_Waktu(:,r-1)    = Data_Waktu_Sebelumnya_ADJUSTED
    Data_Waktu(:,r)      = Data_Waktu_BARU
    Data_Muatan(:,r)     = Data_Muatan_BARU
    Biaya_FOC(:,r)       = Biaya_FOC_BARU
    Datang_Pergi(:,r*2-1:r*2) = Waktu_Occupied_Pelabuhan(:,1:2)
end

% 4) Hitung Total Waktu Semua Kapal
for k = 1:kapal
    Kapal_Sekarang = find(abs(Waktu_Occupied_Pelabuhan(:,3)) ==
k);
    Total_Time(k,1) = Waktu_Occupied_Pelabuhan(Kapal_Sekarang,2);
end

% 5) Hitung Biaya Total
% b. Biaya Sea Time ============================================
Biaya_Transportasi1 = xlsread('data.xlsx','Sheet2','H4:P23');
Biaya_Transportasi2 = xlsread('data.xlsx','Sheet2','D24:O38');
[Transportation_Cost] =
Get_Transportation_Cost(Biaya_Transportasi1,Biaya_Transportasi2,So
lusi_Awal,rute,kapal);

% b. Biaya Pelabuhan ===========================================
XCL_Data_Port_Cost = xlsread('data.xlsx','Port_Cost','C4:C8');
[Port_Cost] =
Get_Port_Cost(XCL_Data_Port_Cost,Solusi_Awal,kapal,rute);

% ========================BIAYA TOTAL ============================

Z = sum(sum(Transportation_Cost,1),2) + sum(sum(Biaya_FOC,1),2) +
sum(sum(Port_Cost,1),2)

% ========================TABU SEARCH============================

Max_Iterasi         = 50;
Jumlah_Tetangga     = 10;
Panjang_Tabu_List   = 10;
Tabu_List           = zeros(kapal,rute,Panjang_Tabu_List);
```

```
Posisi_Tabu_List    = 1;
Solusi_Agak_Baru = Solusi_Awal;
Solusi_Terbaik = Solusi_Awal;

Z_Awal = Z;
Z_Lama = Z_Awal;
Z_Terbaik = Z_Awal;
Z_Terbaik_Solusi_Tetangga = Z_Awal;
toc;

tic;
for iterasi = 1:Max_Iterasi
    for Iterasi_Tetangga = 1:Jumlah_Tetangga
        [Solusi_Tetangga]        =
Perform_Swap_Tiga(Solusi_Agak_Baru,kapal)
        [Result]                 =
Inspector(Solusi_Tetangga,rute);
        if Result == 'OK'

[Data_Muatan_Baru,Data_Isi_Pelabuhan_Setelah_Loading,Data_Waktu_Ba
ru,Biaya_FOC_Baru,Sea_Time_Baru,Jadwal_Baru,Time_Elapsed_Baru] =
Inverse_Solusi(XCL_Kapasitas_Kapal,XCL_Muatan_Awal_Kapal,XCL_Data_
Sea_Time,XCL_Stock_Awal_Pelabuhan,XCL_Waktu_Load_Unload,XCL_Data_F
OC,XCL_Production_Consumption_Rate,Solusi_Tetangga,rute,kapal);
            [Transportation_Cost]    =
Get_Transportation_Cost(Biaya_Transportasi1,Biaya_Transportasi2,So
lusi_Tetangga,rute,kapal);
            [Port_Cost]              =
Get_Port_Cost(XCL_Data_Port_Cost,Solusi_Tetangga,kapal,rute);
            Total_Time               = sum(Sea_Time_Baru,2) +
sum(Data_Waktu_Baru,2);
            [Z_Lokal_Baru]           =
Get_Z(Solusi_Tetangga,Transportation_Cost,Biaya_FOC_Baru,Port_Cost
,rute,kapal);
            Z_Solusi_Tetangga(Iterasi_Tetangga,1) = Z_Lokal_Baru;
            Solusi_Tetangga_Terbaik(:,:,Iterasi_Tetangga) =
Solusi_Tetangga;
        end
    end

    Z_Terbaik_Solusi_Tetangga    = min(Z_Solusi_Tetangga);
    if Z_Terbaik_Solusi_Tetangga < Z_Terbaik
        Posisi_Z_Solusi_Tetangga    = find(Z_Solusi_Tetangga ==
Z_Terbaik_Solusi_Tetangga);
        if numel(Posisi_Z_Solusi_Tetangga > 1)
            Posisi_Z_Solusi_Tetangga  =
randsample(Posisi_Z_Solusi_Tetangga,1);
            Solusi_Terbaik =
Solusi_Tetangga_Terbaik(:,:,Posisi_Z_Solusi_Tetangga);
            Tabu_List(:,:,Posisi_Tabu_List) = Solusi_Terbaik;
        else
            Solusi_Terbaik =
Solusi_Tetangga_Terbaik(:,:,Posisi_Z_Solusi_Tetangga);
            Tabu_List(:,:,Posisi_Tabu_List) = Solusi_Terbaik;
        end
        if Posisi_Tabu_List == 10
            disp('Tabu List Sudah Penuh...')
        end
```

```
        Posisi_Tabu_List = Posisi_Tabu_List + 1;
        Z_Terbaik = Z_Terbaik_Solusi_Tetangga;
    end
    [Solusi_Baru]       =
Perform_Swap_Tiga(Solusi_Agak_Baru,kapal);
    Solusi_Agak_Baru    = Solusi_Baru;
    iterasi
end
Z_Awal
Z_Terbaik

Solusi_Awal
Solusi_Terbaik   = Tabu_List(:,:,Posisi_Tabu_List-1)
[Data_Muatan_TERBAIK,Data_Isi_Pelabuhan_Setelah_Loading_TERBAIK,Da
ta_Waktu_Baru_TERBAIK,Biaya_FOC_Baru_TERBAIK,Sea_Time_Baru_TERBAIK
,Jadwal_Baru_TERBAIK,Time_Elapsed_Baru_TERBAIK] =
Inverse_Solusi(XCL_Kapasitas_Kapal,XCL_Muatan_Awal_Kapal,XCL_Data_
Sea_Time,XCL_Stock_Awal_Pelabuhan,XCL_Waktu_Load_Unload,XCL_Data_F
OC,XCL_Production_Consumption_Rate,Solusi_Terbaik,rute,kapal)
[Transportation_Cost]    =
Get_Transportation_Cost(Biaya_Transportasi1,Biaya_Transportasi2,So
lusi_Terbaik,rute,kapal)
[Port_Cost]              =
Get_Port_Cost(XCL_Data_Port_Cost,Solusi_Terbaik,kapal,rute)
Total_Time               = sum(Sea_Time_Baru,2) +
sum(Data_Waktu_Baru,2)

toc;
```