TUGAS AKHIR – TI 091324

PENGEMBANGAN ALGORITMA *SIMULATED ANNEALING* UNTUK PENYELESAIAN PERMASALAHAN ALOKASI PADA *CLOSED LOOP SUPPLY CHAIN* (CLSC)

RISAL ARSYAD MUHADDAD
NRP 2510 100 127

Dosen Pembimbing
Prof. Ir. Budi Santosa, MS., Ph.D.
NIP 19690512 199402 1001

JURUSAN TEKNIK INDUSTRI
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT – TI 091324

DEVELOPMENT OF SIMULATED ANNEALING ALGORITHM TO SOLVE CLOSED LOOP SUPPLY CHAIN (CLSC) ALLOCATION PROBLEM

RISAL ARSYAD MUHADDAD
NRP 2510 100 127

Supervisor
Prof. Ir. Budi Santosa, MS., Ph.D.
NIP 19690512 199402 1001

DEPARTMENT OF INDUSTRIAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

LEMBAR PENGESAHAN

PENGEMBANGAN ALGORITMA *SIMULATED ANNEALING* UNTUK PENYELESAIAN PERMASALAHAN ALOKASI PADA *CLOSED LOOP SUPPLY CHAIN* (CLSC)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Industri
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:
RISAL ARSYAD MUHADDAD
NRP. 2510 100 127

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Prof. Ir. Budi Santosa, MS., Ph.D.
NIP. 196905121994021001

SURABAYA, JULI 2014

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK INDUSTRI

PENGEMBANGAN ALGORITMA *SIMULATED ANNEALING* UNTUK PENYELESAIAN PERMASALAHAN ALOKASI PADA *CLOSED LOOP SUPPLY CHAIN* (CLSC)

Nama : Risal Arsyad Muhaddad

NRP : 2510100127

Pembimbing : Prof. Ir. Budi Santosa, M.S.,Ph.D

ABSTRAK

Dalam beberapa dekade ini, *supply chain management* menjadi sebuah kebutuhan mendasar bagi perusahaan. Di Indonesia sendiri, strategi mengenai *supply chain management* masih terus dikembangkan. Perkembangan *supply chain management* memunculkan model berupa *Closed Loop Supply Chain* (CLSC) yaitu penggabungan antara *forward logistics* dan *reverse logistics* yang memberikan dampak positif untuk perusahaan, salah satunya penghematan bahan baku. Namun untuk mencari solusi optimal dari alokasi CLSC yang *continues* dan kompleks menggunakan metode eksak akan memakan waktu yang banyak karena permasalahan CLSC termasuk dalam permasalahan *NP-hard*, sehingga dibutuhkan penyelesaian dengan metode metaheuristik. Metode metaheuristik yang dapat digunakan adalah *Simulated Annealing*. *Simulated Annealing* sendiri memiliki kelebihan yaitu waktu komputasi yang lebih cepat dan tidak mudah terjebak pada local optimal karena menggunakan probabilitas Boltzmann. Dalam penelitian tugas akhir ini akan dikembangkan model pada CLSC, algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search* yang diterapkan untuk permasalahan alokasi pada CLSC, sehingga menghasilkan kode program untuk *Simulated Annealing* dan *Simulated Annealing-Tabu Search* dalam menyelesaikan permasalahan alokasi pada CLSC. Dalam penelitian ini terbukti bahwa *Simulated Annealing-Tabu Search* menghasilkan hasil yang lebih baik dibandingkan dengan *Simulated Annealing.*

Kata Kunci: Alokasi, *Closed Loop Supply Chain,* Metaheuristik*, Simulated Annealing.*

A DEVELOPMENT OF SIMULATED ANNEALING ALGORITHM TO SOLVE CLOSED LOOP SUPPLY CHAIN (CLSC) ALLOCATION PROBLEM

Name : Risal Arsyad Muhaddad

Student Number : 2510100127

Supervisor : Prof. Ir. Budi Santosa, M.S.,Ph.D

ABSTRACT

In the last coupe decades, supply chain management has become a fundamental managerial aspects need to be considered by a company to succeed. In Indonesia, studies related to supply chain management have been popular among scholars and researchers. One of developments emerged in this field is closed loop supply chain (CLSC), which is a combination of forward logistics and reverse logistics. Raw material reduction is one of positive impacts brought by the application of this concept, among many others. However, to find the optimal solution from continuous and complex allocation problem in CLSC, for instance, requires a tremendous amount of time due to its NP-Hard characteristic. Therefore, a metaheuristics method needs to be applied to solve this issue. While there are many methods available, Simulated Annealing has its own advantages among others to provide shorter computation time and its ability to hinder from local optimum as it uses Boltzmann probability in its procedure. In this study, Simulated Annealing and Simulated Annealing -Tabu Search model will be deployed on CLSC allocation problem. This will generate a sequence of codes capable to utilize Simulated Annealing and Simulated Annealing - Tabu Search algorithms to solve a CLSC allocation problem. The results suggest that Simulated Annealing - Tabu Search generates a better solution compared to Simulated Annealing.

Keywords: Allocation, Closed Loop Supply Chain, Metaheuristics, Simulated Annealing.

KATA PENGANTAR

Segala puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul **"Pengembangan Algoritma *Simulated Annealing* untuk Penyelesaian Permasalahan Alokasi pada *Closed Loop Supply Chain* (CLSC)"**

Penyusunan Tugas Akhir ini dilakukan untuk memenuhi persyaratan menyelesaikan studi strata satu dan memperoleh gelar sarjana Teknik Industri di Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Dalam penyusunan Tugas Akhir ini tidak terlepas dari bantuan banyak pihak yang telah menberikan bantuan kepada penulis. Oleh karena itu penulis mengucapkan terimakasih kepada:

1. Ibu Mustlah dan Bapak Surama selaku kedua orang tua penulis serta Rikza Azka Mumtaz selaku adik penulis yang senantiasa memberikan dukungan, inspirasi, semangat dan kepercayaan yang luar biasa besar kepada penulis. Terima kasih untuk tetap dan selalu percaya, terima kasih.
2. Prof. Ir. Budi Santosa, M.S., Ph.D selaku Ketua Jurusan Teknik Industri, serta selaku dosen pembimbing atas semua pembelajaran yang telah diberikan selama ini.
3. Ibu Anny Maryani, S.T., M.T. selaku dosen wali. Serta Bapak Yudha Prasetyawan, S.T., M.Eng., Ibu Syarifah Hanoum S.T., M.T., dan Bapak Nugroho Priyo Negoro, S.T., S.E., M.T. yang pernah menjadi dosen wali selama perkuliahan di TI ITS. Termakasih atas bimbingan dan arahan yang diberikan selama ini.
4. Bapak Yudha Andrian, S.T., M.BA. selaku koordinator Tugas Akhir atas kelancaran birokrasi selama pengerjaan Tugas Akhir.

5. Bapak Dody Hartanto, S.T., M.T. selaku dosen penguji pada Tugas Akhir atas masukan dan bimbingannya selama di Teknik Industri ITS.
6. Bapak Ahmad Rusdiansyah, Bapak Stefanus, Ibu Effi, Bapak Gunarta, Bapak Adit, Ibu Dhana, Ibu Niniet, Bapak Nyoman, Bapak Ibnu, Bapak Suparno, Bapak Sritomo, Bapak Udi, Bapak Imam, Bapak Hary, Bapak Suef, Bapak Iwan, dan segenap dosen Teknik Industri ITS yang tidak lelah membimbing dan memberi inspirasi pada penulis selama menempuh masa studi di Teknik Industri ITS.
7. Bapak Bas dosen Teknik Informatika, Bapak Putu dosen Mesin, dan Bapak Sutarsis dosen Metalurgi, yang turut mengajar dalam perkuliahan di Teknik Industri ITS atas bimbingan dan inspirasi yang diberikan kepada penulis selama menempuh perkuliahan.
8. Bapak Ibu Tata Usaha Jurusan Teknik Industri ITS, Ibu Ira, Ibu Ima, Ibu Partinah, Bapak Kimin, Mas Buchori, Mas Hanif, Mas Tegus, Mas Nanang, atas kemudahan administrasi dan birokrasi selama penulis menempuh pendidikan di Jurusan Teknik Industri ITS.
9. Sahabat terdekat penulis Meli, Retha, Aish, Ida, Lona, Sofi, Fajar, Ucup, dan Derry, yang selalu memberikan dukungan dan semangat serta keceriaan selama pengerjaan Tugas Akhir.
10. Teman dari MABA Riri, Desi, Galuh, Hasyim, Ratri, Vita, Kartika, Alo, Fira, Vega, Be, Wiwid, Jimbo, Imam, Agyl, dan Tatsa yang selalu mau mendengar keluh kesah serta selalu memberikan dukungan, bantuan, semangat, dan dorongan agar penulis menyelesaikan Tugas Akhir dengan baik dan benar serta untuk berbagai pengalaman selama perkuliahan.
11. Mansur, Hajar, dan Gusti yang telah banyak membantu penulis dalam proses pengerjaan dan penyelesaian Tugas Akhir ini. Terima kasih yang sebesar-besarnya, jika tidak ada kalian mungkin Tugas Akhir ini tidaklah bisa selesai seperti ini.
12. Teman makan dan main dikala semester tengah Puhenk, Jumi, Gaciel, Nizar, Hendy, Dela, Rara, Karin, Lita, Dini atas dukungan, bantuan, dan semangat yang diberikan kepada penulis untuk menyelesaikan Tugas Akhir ini.

13. Bu Kos Nadiya yang telah banyak membantu dalam segala hal selama pengerjaan Tugas Akhir, serta teman-teman yang sudah lulus pada wisuda 110 Kay, Anissa, Rida, Snups, Prima, dan Intan yang selalu memberi inspirasi dan semangat kepada penulis.
14. Teman SD, SMP, SMA penulis Dedek, Desta, Yola, Tito, Rizka, Dipta, Citra, Ira, Iil, Marfian, Afam, Illa, Dany, Itsna, Vivi, Mona, dan Ari Suci. Seluruh warga E'07, dan EL-SCIEVEN, yang selalu memberikan dukungan dan semangat selama pengerjaan Tugas Akhir.
15. Teman-teman Lab. LSCM Evi, Ketut, Issam, Adit, Dewie, Pocong, Niken, Irma, Revi; Lab. PSMI Zulvah, Chika, Adis, Atikah, Billy, Yolla, Saras, Ajeng; Lab. KOI Apul, Andrew, Layli, Bina, Dewi; Lab. EPSK Nigel, Subi, Arip; Lab. Siman Esty, Namek, Yaniks atas dukungan dan kebaikannya selama pengerjaan Tugas Akhir. Serta adik-adik LSCM Aul, Gio, Putek, Reby, Troy, Willy, Reika, dan Gane atas semangat dan dukungannya.
16. Rekan-rekan Provokasi 2010 atas pengalaman dan kebersamaan yang telah dijalani selama menempuh studi di Jurusan Teknik Industri.
17. Mbak Mas Argent25, Mbak Nilla, Mbak Deliza, Mas SM, dkk atas pengalaman dan kebaikannya selama menempuh studi di TI ITS.
18. Adik-adik Veresis dan Kavaleri, Devin, Didi, Muti, Triyoga, Aan, Ayu, Joshua, Vincentia, dll atas dukungannya dalam pengerjaan Tugas Akhir.
19. Teman-teman Manajemen Unair, Vida, Poce, Hapsari, Ika, Akbar, dan Demi, serta teman-teman Brawijaya, Chea dan Dek Mar, dan teman SCC KCP Rista, Donny, Rizal, Sieta, Karin, Farah, dan Shafira yang telah banyak memberikan inspirasi selama perkuliahan dan Tugas Akhir ini.
20. Bapak Mio dan Bapak Budi yang selalu rela menjaga kendaraan sampai malam sehingga kampus Jurusan Teknik Indutri selalu aman. Serta seluruh See and Go Jurusan Teknik Industri yang membuat kampus terasa bersih dan nyaman.

Serta berbagai pihak yang tidak dapat penulis sebutkan satu per satu, semoga Allah SWT membalas semua kebaikan yang telah dilakukan.

Penulis menyadari bahwa pengerjaan Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan sebagai motivasi dalam rangka pengembangan diri menjadi lebih baik. Penulis berharap Tugas akhir ini dapat memberikan manfaat bagi para pembacanya. Sekian yang bisa penulis sampaikan, mohon maaf jika ada yang kurang berkenan, terima kasih.

Surabaya, 4 Agustus 2014

Penulis

DAFTAR ISI

ABSTRAK i

ABSTRACT iii

KATA PENGANTAR v

DAFTAR ISI ix

DAFTAR TABEL xiii

DAFTAR GAMBAR xv

BAB 1 PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Perumusan Masalah 3

1.3 Ruang Lingkup Penelitian 3

1.3.1 Batasan 3

1.3.2 Asumsi 3

1.4 Tujuan Penelitian 3

1.5 Manfaat Penelitian 4

1.6 Sistematika Penulisan 4

BAB 2 TINJAUAN PUSTAKA 7

2.1 *Supply Chain Management* 7

2.2 *Reverse Logistics* 8

2.2.1 *Recycling* 9

2.2.2 *Re-use* 10

2.2.3 *Remanufakturing* 10

2.3 *Closed Loop Supply Chain* (CLSC) 10

2.4 Optimasi 13

2.5 Metaheuristik 14
2.5.1 Algoritma *Simulated Annealing* 16
2.5.2 Algoritma *Tabu Search* 17
2.6 Posisi Penelitian 18
BAB 3 METODOLOGI PENELITIAN 21
3.1 Tahap Pengumpulan Data 22
3.2 Tahap Pengembangan Model Matemastis 23
3.3 Tahap Pengembangan Algoritma 23
3.3.1 Pembuatan Kode Program untuk SA 24
3.3.2 Pembuatan Kode Program untuk SA-TS 27
3.3.3 Verifikasi dan Validasi Algoritma SA-TS 28
3.3.4 Eksperimen 28
3.4 Tahap Analisis dan Kesimpulan 28
3.4.1 Analisis dan Interpretasi 28
3.4.2 Penarikan Kesimpulan dan Saran 29
BAB 4 PENGEMBANGAN MODEL DAN ALGORITMA 31
4.1 Pengembangan Model *Closed Loop Supply Chain* 31
4.1.1 Model Matematis *Closed Loop Supply Chain* 31
4.2 Pengembangan Algoritma *Simulated Annealing* untuk *Closed Loop Supply Chain* 34
4.3 Verifikasi dan Validasi 40
4.3.1 Verifikasi dan Validasi Algoritma *Simulated Annealing* 41
4.3.2 Verifikasi dan Validasi Algoritma *Simulated Annealing - Tabu Search* 42
BAB 5 EKSPERIMEN DAN ANALISIS 45
5.1 Deskripsi Data Uji 45

5.2 Eksperimen 45
5.2.1 Eksperimen dengan Algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search* 46
5.2.1.1 Eksperimen Uji Parameter 46
5.2.1.2 Eksperimen SA dan SA-TS pada Data Uji Ke-2 49
5.2.1.3 Eksperimen SA dan SA-TS pada Data Uji Ke-3 53
5.1 Analisis Hasil Eksperimen 62
5.1.1 Analisis Uji Parameter Algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search* 62
5.1.2 Analisis Hasil SA dan SA-TS untuk Data Uji 2 63
5.1.3 Analisis Hasil SA dan SA-TS untuk Data Uji 3 64
BAB 6 KESIMPULAN DAN SARAN 65
6.1 Kesimpulan 65
6.2 Saran 65
DAFTAR PUSTAKA 67
LAMPIRAN 69
BIOGRAFI PENULIS 81

DAFTAR TABEL

Tabel 2.1 Posisi Penelitian 19
Tabel 4.1 Data *Demand,* Botol Balik ke *Warehouse,* Botol Balik ke *Manufacture.* 34
Tabel 4.2 Jarak dari *Manufacture* ke *Warehouse, Warehouse* ke *Manufacture, Warehouse* ke *Distributor, Distributor* ke *Warehouse.* 35
Tabel 4.3 Data Kapasitas Produksi dan *Warehouse* 35
Tabel 5.1 Data Uji 45
Tabel 5.2 Uji Parameter Faktor Pereduksi Temperatur 46
Tabel 5.3 Uji Parameter Faktor Pereduksi Temperatur (Lanjutan)..... 47
Tabel 5.4 Uji Parameter Temperatur Awal 48
Tabel 5.5 Uji Parameter Temperatur Awal (Lanjuan) 49
Tabel 5.6 Hasil Eksperimen Data Uji 2 dengan *Simulated Annealing* 50
Tabel 5.7 Hasil Eksperimen Data Uji 2 dengan *Simulated Annealing-Tabu Search* 51
Tabel 5.8 Perbandingan Nilai Perubahan Total Biaya pada Data Uji 2..... 52
Tabel 5.9 Perbandingan Waktu Komputasi pada Data Uji 2 52
Tabel 5.10 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 53
Tabel 5.11 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan).. 54
Tabel 5.12 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan 2)55
Tabel 5.13 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan 3)56
Tabel 5.14 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 57
Tabel 5.15 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 1)..... 58
Tabel 5.16 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 2)..... 59
Tabel 5.17 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 3)..... 60

Tabel 5.16 Perbandingan Nilai Perubahan Total Biaya pada Data Uji 3 61
Tabel 5.17 Perbandingan Waktu Komputasi pada Data Uji 3..... 61

DAFTAR GAMBAR

Gambar 3.1 Tahapan Metodologi Penelitian 21
Gambar 3.2 Tahapan Metodologi Peneletian (Lanjutan) 22
Gambar 3.3 Alur Distribusi 23

BAB 1
PENDAHULUAN

Bab ini merupakan dasar dari dilakukannya penelitian yang meliputi latar belakang masalah, perumusan masalah, tujuan penelitian, dan manfaat penelitian, ruang lingkup penelitian.

1.1 Latar Belakang

Dalam beberapa dekade ini, *supply chain management* menjadi sebuah kebutuhan mendasar bagi perusahaan. Di Indonesia sendiri, strategi mengenai *supply chain management* masih terus dikembangkan. Menurut Pujawan, (2005) *supply chain* merupakan hubungan jaringan antar perusahaan-perusahaan yang memiliki kerjasama untuk menciptakan dan mendistribusikan suatu produk kepada *customer* akhir.

Pada *supply chain,* terdapat beberapa jenis aliran yang harus dikelola. Pertama, aliran barang yang mengalir dari hulu (*upstream*) ke hilir (*downstream*). Contohnya, bahan baku yang dikirim dari *supplier* ke pabrik, produk jadi yang diproduksi oleh pabrik lalu dikirim ke distributor, kemudian oleh distributor dikirim ke pengecer sampai pada pemakai akhir. Kedua adalah aliran uang dan sejenisnya yang mengalir dari hilir ke hulu. Ketiga adalah aliran informasi yang bisa terjadi dari hulu ke hilir ataupun sebaliknya.

Banyak hal yang menjadi pertimbangan dalam pengambilan keputusan untuk penentuan aliran material. Aliran material dari hulu ke hilir atau biasa disebut *forward logistics* akan berdampak pada penentuan alur distribusi serta penentuan *distribution center* yang mampu melakukan penyimpanan dan mampu mencakup daerah *customer* dengan *cost* yang minimum. Namun sekarang banyak perusahaan yang menjual produk yang dalam bagian produk tersebut dapat diambil kembali manfaatnya, sehingga perlu mempertimbangkan bagaimana aliran bagian produk tersebut kembali ke perusahaan atau biasa disebut *reverse*

logistics. *Reverse logistics* dapat diterapkan pada pengadaan botol untuk teh dan minuman bersoda, gallon pada air mineral galon, dan tabung LPG.

Rogers dan Tibben-Lembke (1999) mendefinisikan *reverse logistics* sebagai aktivitas untuk merencanakan, mengaplikasikan, dan mengendalikan proses agar tercapai efisiensi dalam aliran material, persediaan, produk jadi, dan informasi terkait dari konsumen kembali ke manufaktur dengan tujuan untuk mendapatkan kembali nilai ekonomis produk atau untuk melakukan proses pembuangan yang tepat. Aktivitas utama dari *reverse logistics* adalah melakukan distribusi untuk pengumpulan bagian produk yang dapat diambil manfaatnya dan melakukan distribusi untuk produk yang diperbaharui.

Reverse logistics ini banyak memberikan keuntungan bagi perusahaan. Konsep ini menjadi salah satu alternatif terbaik yang dapat dipertimbangkan untuk mengurangi keterbatasan sumber daya bahan baku. Selain itu, *reverse logistics* terbukti dapat memberikan nilai ekonomi bagi para pelakunya (Rivera dan Ertel, 2008). Dilain pihak, isu lingkungan menjadi salah satu motivasi terkuat untuk melakukan *reverse logistic*.

Seperti yang kita ketahui bahwa akan lebih mudah apabila *reverse distribution* untuk produk tersebut mengacu pada *forward distribution*. Namun terkadang terdapat batasan-batasan dalam keadaan yang nyata yang menyebabkan perusahaan perlu memperhitungkan kapasitas dan kemampuan setiap *distribution center* serta pabrik dalam menangani produk asli (produk *forward disribution*) dan *reverse* produk. Penentuan keputusan secara manajerial untuk kasus *forward* dan *reverse logistics* biasa disebut dengan *Closed Loop Supply Chain* (CLSC) yang berarti aliran tersebut akan terus berlangsung dengan barang .

Pada dasarnya permasalahan CLSC dapat diselesaikan menggunakan *software* khusus seperti Lindo, Lingo, Gams dan Cplex. CLSC dapat pula diselesaikan menggunakan model MILP, robust optimization, serta branch and bound. Namun terdapat kekurangan dalam metode-metode tersebut, seperti jumlah perhitungan yang besar dan lama. Dari kekurangan itulah maka dapat dikembangkan model yang lebih handal dan cepat untuk menyelasaikan masalah *Closed Loop Supply Chain* (CLSC) dengan menggunakan metode metaheuristik. Salah satu metode metaheuristik yang dapat digunakan adalah *Simulated*

Annealing. *Simulated Annealing* sendiri memiliki kelebihan yaitu waktu komputasi yang lebih cepat dan tidak mudah terjebak pada lokal optimal karena menggunakan probabilitas Boltzmann yaitu dapat menerima hasil yang tidak lebih baik.

1.2 Perumusan Masalah

Berdasarkan uraian pada latar belakang yang telah dipaparkan sebelumnya, permasalahan pada penelitian ini adalah bagaimana mengembangkan algoritma *Simulated Annealing* untuk menyelesaikan permasalahan *Closed Loop Supply Chain.*

1.3 Ruang Lingkup Penelitian

Dalam ruang lingkup penelitian akan dijelaskan batasan-batasan serta asumsi yang digunakan. Batasan-batasan yang ada digunakan agar penelitian ini tidak meluas dan tetap dalam lingkup penelitian, sedangkan asumsi digunakan untuk memberikan pembenaran atas kondisi penelitian terhadap kondisi realitasnya.

1.3.1 Batasan

Batasan yang digunakan dalam penelitian ini antara lain:

1. Menggunakan multi eselon yang terdiri dari supplier, manufacturing plants, warehouse, dan distributor
2. Data yang digunakan adalah data sekunder dari perusahaan X.

1.3.2 Asumsi

Asumsi yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Backorder tidak diperhitungkan dalam model CLSC ini.
2. Seluruh input data bersifat deterministik selama periode perencanaan.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Mengembangkan model *Closed Loop Supply Chain* (CLSC).

2. Mengembangkan algoritma dan menghasilkan kode program *Simulated Annealing* dan *Simulated Annealing-Tabu Search* untuk diterapkan pada kasus *Closed Loop Supply Chain* (CLSC).

1.5 Manfaat Penelitian

Penelitian tugas akhir ini diharapkan dapat memberikan kontribusi dalam bidang keilmuan optimasi, yaitu dengan adanya penyelesaikan permasalahan *Closed Loop Supply Chain* (CLSC) dengan menggunakan algoritma *Simulated Annealing*.

1.6 Sistematika Penulisan

Laporan Tugas Akhir ini akan terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bab pendahuluan ini berisi tentang hal-hal yang mendasari dilakukannya penelitian serta pengidentifikasian masalah penelitian. Bagian-bagian yang terdapat dalam bab pendahuluan ini meliputi latar belakang masalah, perumusan masalah, ruang lingkup penelitian, tujuan penelitian, serta manfaat penelitian.

BAB II TINJAUAN PUSTAKA

Tinjauan pustaka menguraikan teori, temuan, dan bahan penelitian lain yang diperoleh dari acuan yang akan dijadikan landasan untuk melakukan kegiatan penelitian yang akan dijadikan tugas akhir.

BAB III METODOLOGI PENELITIAN

Bab ini menguraikan metodologi penelitian yang dilakukan dalam mengembangkan algoritma *Simulated Annealing* untuk menyelesaikan permasalahan *Closed Loop Supply Chain* (CLSC).

BAB IV PENGEMBANGAN MODEL DAN ALGORITMA

Pada bab pengembangan model dan algoritma ini akan dijelaskan tentang tahap-tahap yang dilakukan dalam mengembangkan model *Closed Loop Supply*

Chain (CLSC) dan algoritma *Simulated Annealing* yang digunakan dalam penyelesaian permasalahan.

BAB V EKSPERIMEN DAN ANALISIS

Pada bab eksperimen dan analisis akan dijabarkan mengenai analisis hasil eksperimen dari bab pengujian algoritma. Di samping itu juga dilakukan analisis perbandingan dari algoritma yang diusulkan dengan metode eksak.

BAB VI KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan hasil penelitian dan saran-saran yang dapat dikembangkan untuk penelitian selanjutnya

BAB 2
TINJAUAN PUSTAKA

Pada bagian ini akan diuraikan mengenai konsep, teori, temuan, dan hasil penelitian terdahulu yang diperoleh dari acuan yang akan dijadikan landasan dalam menyelesaikan permasalahan *Closed Loop Supply Chain* (CLSC) menggunakan *Simulated Annealing*.

2.1 *Supply Chain Management*

Supply chain management merupakan metode, alat atau pendekatan pengelolaan yang melibatkan perusahaan – perusahaan yang terlibat dalam memasok bahan baku, memproduksi barang, maupun mengirimkannya ke final consumer. *Supply chain management* tidak hanya berorientasi pada urusan internal, melainkan juga urusan eksternal yang menyangkut hubungan dengan perusahaan – perusahaan yang terkait.

Definisi *supply chain* menurut Beamon (1998) adalah proses manufaktur yang terintegrasi mulai dari bahan baku yang diproses menjadi produk jadi kemudian didistribusikan ke konsumen. Pada tingkatan tertinggi, proses terintegrasi pada *supply chain* dapat dibagi menjadi dua yaitu : (1) *production planning and inventory control process*; (2) *distribution and logistic process*.

Production Planning and Inventory Control Process secara spesifik merencanakan produksi yang menggambarkan desain dan manajemen dari keseluruhan proses manufaktur, termasuk dari penjadwalan order bahan baku hingga penerimaan, desain dan penjadwalan proses manufaktur, dan desain *material handling*. *Inventory control* menggambarkan pengaturan dan desain dari kebijakan dan prosedur penyimpanan bahan baku, *work-in-process inventories*, dan yang terakhir produk jadi.

Distribution and Logistic Process menentukan bagaimana produk didistribusikan dari gudang ke retailer. Suatu produk dapat didistribusikan secara langsung ke retailer, atau dikirimkan ke sub-warehouse dahulu baru kemudian didistribusikan ke retailer.

Pada awalnya desain, model dan analisa dari *supply chain* terfokus pada pengoptimalan pengadaan bahan baku dari suplier dan pengiriman produk ke *end-costumer*. Akan tetapi Beamon (1998) mengungkapkan bahwa ruang lingkup *supply chain* melibatkan :

1. Penjadwalan produksi dan distribusi, yaitu penjadwalan proses produksi dan/atau distribusi;
2. Tingkat persediaan, penentuan jumlah dan lokasi dari setiap bahan baku, sub-*assembly*, dan gudang final *assembly;*
3. Jumlah tingkatan, yaitu penentuan jumlah tingkatan (*number of stages* atau eselon) yang membentuk supply chain. Disini terjadi peningkatan atau penurunan jumlah tingkatan (*chain's level*) secara vertikal dengan penggabungan (atau pengu-rangan) stages atau pemisahan (atau penambahan) *stages*;
4. Pusat distribusi (*distribution centre*)-*customer assignment*, yaitu penentuan distribution centre yang mana yang akan melayani pelanggan;
5. *Plant-product assignment*, yaitu penentuan pabrik yang mana yang akan memproduksi produk yang mana;
6. Hubungan pembeli-*supplier*, yaitu penentuan dan pengem-bangan aspek kritis pada hubungan pembeli-*supplier;*
7. Tahap spesifikasi pada diferensiasi produk, yaitu penentuan tahap proses produk manufaktur yang telah dilakukan diferensiasi (spesialis);
8. Jumlah tipe/jenis produk yang akan disimpan, yaitu pe-nentuan jumlah jenis produk yang akan disimpan sebagai *finished good inventory*.

2.2 *Reverse Logistics*

Menurut Rogers dan Tibben-Lembke (1999) *reverse logistics* merupakan aktivitas untuk merencanakan, mengaplikasikan, dan mengendalikan proses dari konsumen kembali ke manufaktur dengan tujuan untuk mendapatkan kembali manfaat dari produk atau untuk melakukan proses pembuangan yang tepat agar tercapai efisiensi terkait dengan arus material, persediaan, produk jadi, dan informasi. *Reverse logistics* saat ini menjadi salah satu alternatif terbaik yang dapat dipertimbangkan untuk mengurangi keterbatasan sumber daya bahan baku.

Selain itu, *reverse logistics* terbukti dapat memberikan nilai ekonomi tambahan bagi para pelakunya (Rivera dan Ertel, 2008). Dilain pihak, isu lingkungan menjadi salah satu motivasi terkuat untuk melakukan *reverse logistic*.

Aktivitas utama dari *reverse logistics* adalah mengumpulkan produk yang akan diperbaharui dan melakukan redistribusi material baru yang dihasilkan. Pada dasarnya tahapan aktivitas yang terjadi pada *reverse logistics* mirip dengan aktivitas yang terjadi pada *forward logistics*, namun dengan beberapa hal perbedaan. menjelaskan perbedaan tersebut meliputi beberapa hal yaitu :

1. Pada *reverse logistics* terdapat banyak titik *supply* dimana produk dapat diperoleh begitu juga dengan titik pengumpulan produk;
2. Dibutuhkan kerjasama yang baik dan suka rela dari supplier produk, dalam hal ini adalah konsumen, untuk menyerahkan barangnya ke titik pengumpulan produk;
3. Produk yang dikumpulkan biasanya memiliki nilai ekonomis rendah. Tidak seperti *forward logistics* yang lebih dahulu ada, jaringan logistik untuk *reverse logistics* dalam berbagai aspek perlu dikembangkan.

Lebih jauh, Bernon (2004) membagi jaringan logistik untuk *reverse logistics* menjadi empat. Pembedaan ini dibuat berdasarkan inisiator dari aktivitas *reverse logistics*. Jaringan logistik tersebut meliputi :

1. Jaringan logistik untuk produk *reusable*
2. Jaringan logistik untuk *remanufacturing*
3. Jaringan logistik untuk layanan masyarakat dan regulasi lingkungan oleh pemerintah
4. Jaringan logistik oleh swasta untuk pembaharuan produk.

2.2.1 *Recycling*

Recycling adalah daur ulang produk, yaitu merubah produk tidak terpakai kemudian diproses menjadi produk, komponen atau bahan baku. Setelah didaur ulang maka fungsi dan identitas asli dari bahan baku telah berubah. Berhasil tidaknya proses ini bergantung pada, ada tidaknya pasar untuk bahan baku hasil daur ulang dan kualitas dari bahan baku hasil daur ulang.

2.2.2 *Re-use*

Re-use merupakan proses pengumpulan bahan baku, produk atau komponen yang tidak terpakai lagi, yang masih bisa digunakan kembali sebagai komponen produk baru, seperti LPG, galon air mineral, botol minuman, dengan nilai ekonomis lebih rendah namun bermanfaat untuk kelangsungan bahan baku.

2.2.3 *Remanufakturing*

Remanufacturing adalah proses pengumpulan produk atau komponen dari produk yang telah digunakan, mengganti bagian yang rusak dengan komponen yang baru maupun bekas yang tidak rusak. Diproduksi menjadi produk baru dengan tetap memiliki identitas dan fungsi yang sama dengan produk awalnya, namun dengan standard kualitas yang berbeda menjadi *new brand product.* Pada *remanufacturing* dapat dihasilkan produk dengan kualitas yang lebih rendah.

2.3 *Closed Loop Supply Chain* (CLSC)

Closed Loop Supply Chain (CLSC) berfokus pada mengambil kembali produk dari pelanggan yang memberikan nilai tambah dengan menggunakan kembali seluruh produk atau beberapa bagian, modul, komponen (Schultmann dan Zumkeller, 2006)

CLSC tidak hanya sesuai dengan kebutuhan dalam membangun penghematan sumber daya dan ramah lingkungan pada masyarakat, tetapi juga dapat membantu perusahaan dalam meningkatkan keuntungan perusahaan dan memenangkan reputasi dalam kontribusi perlindungan lingkungan hijau. Penelitian tentang *Closed Loop Supply Chain* (CLSC) sampai saat ini masih kurang secara sistematik dan mendalam, terutama optimasi dan koordinasi pada *Closed Loop Supply Chain* (CLSC).

Salah satu model CLSC yang telah dikembangkan oleh Subramanian P., Ramkumar N., Narendran T.T., & Ganesh K. (2012) adalah sebagai berikut:

Notasi

Indeks

k = manufacturer (k=1, 2,...,K)

j = distributor (j=1, 2,...,J)

i = wholesaler (i=1, 2,...,I)

w = retailer (w=1, 2,...,W)

m = produk (m=1, 2,...,M)

z = periode (z=1, 2,...,Z)

c = collection centre (c=1, 2,...,C)

Biaya dari fungsi tujuan

$TPUC$ = total biaya pembelian (hanya biaya transportasi)

TPC = total biaya proses

$TCCPTC$ = total biaya transportasi dari collecting centre ke manufacturer

$TPDTC$ = total biaya transportasi dari manufacturer ke distributors

$TDWTC$ = total biaya transportasi dari distributors ke wholesalers

$TWRTC$ = total biaya transportasi dari wholesalers ke retailers

$TCCIC$ = biaya penyimpanan di collection centre

$TWIC$ = biaya penyimpanan di wholesaler

$TDIC$ = biaya penyimpanan di distributor

$TRIC$ = biaya penyimpanan di retailer

Input parameter

DEM_{wmz} = demand dari produk m di retailer w pada periode z

DS_j = kapasitas gudang distributor j

$ICCC_{cm}$ = biaya penyimpanan per waktu per produk m di collecting centre c

ICD_{jm} = biaya penyimpanan per waktu per produk m di distributor j

ICR_{wm} = biaya penyimpanan per waktu per produk m di retailer w

ICW_{im} = biaya penyimpanan per waktu per produk m di wholesaler i

PC_{km} = biaya produksi produk m di manufacturer k

P_k = jumlah waktu proses tersedia pada manufacturer k

PS_k = kapasitas gudang manufacturer k

PT_{km} = waktu proses untuk produk m di manufacturer k

RET_{cmz} = jumlah kembali produk m di collection centre c pada periode z

RS_w = kapasitas gudang retailer w

TC_k = biaya transportasi dari supplier terbaik ke manufacturer k

$TCCCP_{ck}$ = biaya transportasi dari collection centre c ke manufacturer k

$TCDW_{ji}$ = biaya transportasi dari distributor j ke warehouse i

$TCPD_{kj}$ = biaya transportasi dari manufacturer k ke distributor j

$TCWR_{iw}$ = biaya transportasi dari warehouse i ke retailer w

WS_i = kapasitas gudang di wholesaler i

Minimise Z = TPUC + TPC + TCCPTC + TPDTC + TDWTC + TWRTC + TCCIC + TWIC + TDIC + TRIC,

$$\text{TPUC} = \sum_k \sum_m \sum_z QRP_{kmz}.TC_k$$

$$\text{TPC} = \sum_k \sum_m \sum_z QP_{kmz}.PC_{km}$$

$$\text{TCCPTC} = \sum_c \sum_k \sum_m \sum_z QTCCP_{ckmz}.TCCCP_{ck}$$

$$\text{TPDTC} = \sum_k \sum_j \sum_m \sum_z QTPD_{kjmz}.TCPD_{kj}$$

$$\text{TDWTC} = \sum_j \sum_i \sum_m \sum_z QTDW_{jimz}.TCDW_{ji}$$

$$\text{TWRTC} = \sum_i \sum_w \sum_m \sum_z QTWR_{iwmz}.TCWR_{iw}$$

$$\text{TCCIC} = \sum_c \sum_m \sum_z INVC_{cmz}.ICCC_{cm}$$

$$\text{TWIC} = \sum_i \sum_m \sum_z INVW_{imz}.ICW_{im}$$

$$\text{TDIC} = \sum_j \sum_m \sum_z INVD_{jmz}.ICD_{jm}$$

$$\text{TRIC} = \sum_w \sum_m \sum_z INVR_{wmz}.ICR_{wm}$$

Konstrain

Manufacturer

Konstrain waktu proses $\sum_m (QP_{kmz}.PT_{km}) \le P_k \quad \forall k, \forall z.$

Konstrain kapasitas gudang $\sum_m QRP_{kmz} + \sum_m \sum_c QTCCP_{ckmz} \le PS_k \quad \forall k, \forall z.$

Kebutuhan bahan baku $QP_{kmz} - QRP_{kmz} = 0 \quad \forall k, \forall z, \forall m$

Konstrain aliran untuk manufacturer

$$QP_{kmz} + \sum_{c} QTCCP_{ckmz} - \sum_{j} QTPD_{kjmz} = 0 \quad \forall k, \forall z, \forall m$$

Distributor

Konstrain kapasitas gudang

$$\sum_{m} INVD_{jm(z-1)} - \sum_{k} \sum_{m} QTPD_{kjmz} \leq DS_{j}, \quad \forall j, \forall z.$$

Konstrain aliran untuk distributor

$$\sum_{m} INVD_{jm(z-1)} + \sum_{k} QTPD_{kjmz} - \left(INVD_{jmz} + \sum_{i} QTDW_{jimz} \right) = 0, \ \forall m, \forall j, \forall z.$$

Wholeseler

Konstrain kapasitas gudang

$$\sum_{m} INVW_{im(z-1)} - \sum_{j} \sum_{m} QTDW_{jimz} \leq WS_{i}, \quad \forall i, \forall z.$$

Konstrain aliran untuk wholesaler

$$INVW_{im(z-1)} + \sum QTDW_{jimz} - \left(INVW_{imz} + \sum_{i} QTWR_{iwmz} \right) = 0, \ \forall m, \forall i, \forall z.$$

Retailer

Konstrain kapasitas gudang

$$\sum_{m} INVR_{wm(z-1)} - \sum_{i} \sum_{m} QTWR_{iwmz} \leq RS_{w} \quad \forall m, \forall z.$$

Konstrain aliran demand

$$INVR_{wm(z-1)} + \sum QTWR_{iwmz} - INVR_{wmz} \geq DEM_{wmz}, \quad \forall m, \forall w, \forall z.$$

Collection Center

Kuantitas aliran kembali

$$\left(INVC_{cm(z-1)} + RET_{cmz}\right) + \left(\sum QTCCP_{ckmz} - INVC_{cmz} \right) = 0, \quad \forall m, \forall c, \forall z.$$

2.4 Optimasi

Menurut Santosa dan Willy (2011), optimasi merupakan sekumpulan formula matematis dan metoda numerik yang digunakan untuk mencari dan

menemukan kandidat terbaik dari alternatif-alternatif yang ada, optimasi digunakan untuk mendesain suatu sistem agar memperoleh biaya yang lebih murah atau keuntungan yang lebih banyak, menurunkan waktu proses, meratakan ultilitas, dan sebagainya. Untuk melakukan optimasi biasanya digunakan bantuan software untuk menyelesaikan permasalhan agar didapatkan waktu komputasi yang lebih cepat dengan hasil yang optimum. Syarat-syarat keberhasilan penerapan teknik optimasi adalah kemampuan membuat model matematis permasalahan, pengetahuan tentang teknik optimasi, dan kecakapan dalam program komputer.

2.5 Metaheuristik

Metaheuristik adalah metode untuk mencari solusi yang memadukan interaksi antara prosedur pencarian lokal dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik local optimum dan melakukan pencarian di ruang solusi untuk menemukan solusi global optimum. Metaheuristik biasanya berupa prosedur umum yang bisa diterapkan untuk berbagai problem dengan sedikit penyesuaian. Menurut Talbi (2009), metaheuristik dapat didefinisikan sebagai metode lanjut berbasis heuristik untuk menyelesaikan persoalan optimisasi secara efisien. Secara umum metehauristik dipakai untuk masalah-masalah yang komplek dan tidak bisa diselesaikan dengan mudah secara eksak. Metaheuristik tidak terlalu bermanfaat untuk persoalan yang dengan mudah dan cepat dapat diselesaikan secara eksak (penyelesaian eksak merupakan penyelesaian terbaik, tetapi seringkali metode ini tidak dapat diterapkan pada permasalahan optimasi, sehingga dipakailah metode pendekatan).

Karakteristik umum yang biasa dimiliki oleh pendekatan metaheuristik:

1. Biasanya stokhastik: menggunakan bilangan random yang nilainya stokhastik untuk menentukan keputusan dalam salah satu langkah dalam algoritma. Ini memungkinkan untuk mengatasi permasalahan banyaknya kemungkinan solusi dalam masalah kombinatorial.
2. Umumnya tidak mempunyai masalah dengan penghitungan gradient dari fungsi tujuan.

3. Biasanya diinspirasi oleh analogi fisik (*Simulated Annealing*), biologi (evolutionary algorithms) atau ethology (ant colony, particle swarm).
4. Mempunyai kelemahan umum: kesulitan mengatur nilai parameter dan komputasi yang lama, namun waktu komputasi ini juga kadang menjadi keunggulan dibanding optimasi eksak.

Santosa dan Willy (2011) menyatakan bahwa SA adalah algoritma yang meniru meniru perilaku proses peleburan baja yang dipanaskan sampai temperatur tertentu dan kemudian baja tersebut didinginkan secara perlahan-lahan.

Proses pendinginan baja setelah dipanaskan inilah kemudian diimitasi dalam metode metaheuristik *Simulated Annealing* dengan cara menentukan parameter temperatur kemudian mengontrol temperatur tersebut dengan memakain konsep dari distribusi Boltzmann. Distribusi ini menyatakan bahwa suatu energi sistem dalam keseimbangan panas pada temperatur *T* adalah terdistribusi secara probabilistik seperti diformulasikan dalam rumus sebagai berikut:

$$P(E) = e^{-E/kT}$$

Dimana:

P(E) = peluang mencapai tingkat energi *E*

T = temperatur

k = konstanta Boltzmann

Metode probabilitas Boltzmann dapat juga diterapkan dalam kasus minimasi fungsi. Jika nilai sekarang adalah *x* dan nilai fungsi adalah *f(x)*, energi E_i pada status x_i adalah

$$E_i = f_i = f(x_i)$$

Menurut kriteria Metropolis, probabilitas titik solusi selanjutnya adalah x_{i+1} yang bergantung pada perbedaan status energi atau fungsi tujuan di dua titik didapatkan dari:

$$P[E_{i+1}] = \min\{1, e^{-E/kT}\}$$

Dimana

$$\Delta E = E_{i+1} - E_i = \Delta f = f_{i+1} - f_i = f(x_{i+1}) - f(x_i)$$

Titik baru kemudian baru bisa ditemukan dengan menggunakan distribusi probabilitas Boltzmann. Nilai k merupakan faktor Boltzmann yang biasanya adalah bernilai 1. Jika $\Delta E \leq 0$, maka $P[E_{i+1}] = 1$ sehingga dalam kasus ini titik x_{i+1} selalu diterima. Tetapi, jika $\Delta E > 0$, maka x_{i+1} belum tentu ditolak. Nilai x_{i+1} masih dimugkinkan untuk diterima dengan probabilitas Boltzmann seperti formulasi di bawah ini:

$$P(E) = e^{-E/kT}$$

2.5.1 Algoritma *Simulated Annealing*

Dalam pembentukan algoritma dibutuhkan input parameter-parameter yang tepat untuk tercapainya keberhasilan pelaksanaan algoritma SA yaitu pemilihan nilai awal temperatur T, jumlah iterasi k sebelum melakukan penurunan temperatur, jumlah iterasi n, dan faktor penurun temperatur c. Pada pemilihan nilai awal temperatur T apabila terlalu tinggi maka akan membutuhkan penurunan temperatur yang banyak sampai konvergen, sebaliknya jika nilai awal temperatur T yang digunakan terlalu rendah maka kemungkinan melewatkan titik-titik potensial yang mungkin menjadi global optimum. Untuk faktor pereduksi c jika nilai yang digunakan terlalu besar (mendekati 1) akan memberikan pencarian yang lebih teliti namun dengan menambah waktu komputasi, sebaliknya jika nilai c terlalu kecil (mendekati 0) akan banyak titik-titik potensial untuk menjadi global optimum terlewati karena penurunan temperatur yang terlalu cepat. Begitu juga apabila jumlah iterasi k terlalu besar akan membantu mencapai keseimbangan termal pada satu siklus, tetapi dengan tambahan waktu komputasi. Untuk jumlah iterasi n sedikit, maka konvergensi akan terlalu cepat tercapai dan menyebabkan tidak mendapatkan global optimum.

Adapun algoritma *Simulated Annealing* sebagai berikut:

Input : T_0, c, n, itmax

Output : xopt, fbest

Bangkitkan x_0

fbest = $f(x_0)$

T = T_0

while kriteria kriteria penghentian belum tercapai **do** bangkitkan solusi baru x_n

```
    if (f(xn) < f(x0)) then
        fbest = f(xn)
        x0 = xn
    else
        Δf = f(xn) – f(x0)
        Bangkitkan bilangan random r
        if  r < exp(-Δf/kT) then
            x0 = xn
        end
    end
end
```

2.5.2 Algoritma *Tabu Search*

Tabu Search pertama kali diperkenalkan oleh Glover sekitar tahun 1986. Tabu search adalah sebuah metode optimasi yang berbasis pada local search atau proses pencarian yang bergerak dari satu solusi ke solusi berikutnya, dengan cara memilih solusi terbaik dari neighborhood solusi sekarang (current) yang tidak tergolong solusi terlarang (tabu). 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 lokal dapat terjebak pada daerah solusi optimum lokal tersebut pada iterasi-iterasi berikutnya.Perekaman solusi secara lengkap dalam sebuah forbidden list dan pengecekan apakah sebuah kandidat solusi tercatat dalam list tersebut merupakan cara yang mahal, baik dari sisi kebutuhan memori maupun kebutuhan waktu

komputasi. Jadi tabu list hanya menyimpan langkah transisi (move) yang merupakan lawan atau kebalikan dari 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 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 kriteria aspirasi, 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 kriteria aspirasi yang telah ditetapkan sebelumnya, maka move tersebut dapat digunakan untuk membentuk solusi berikutnya (status tabunya dibatalkan).

2.6 Posisi Penelitian

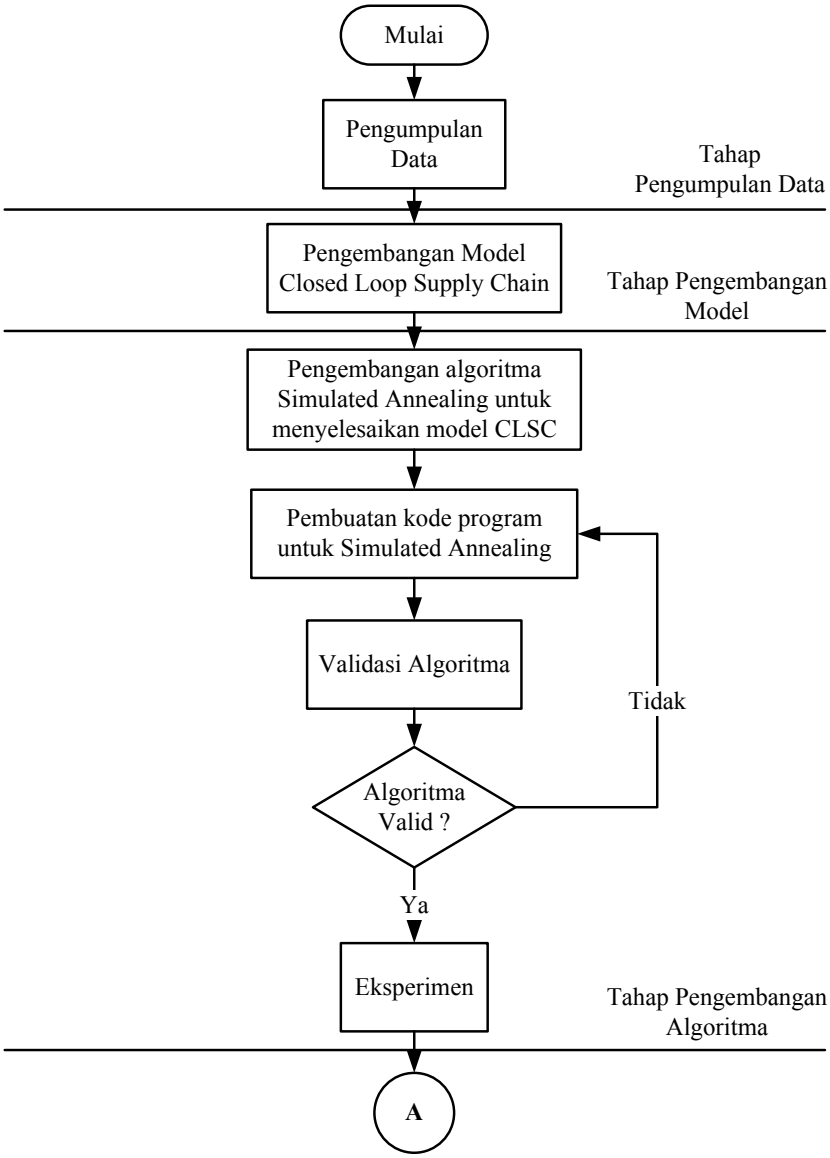
Penelitian yang menyangkut permasalahan *Closed Loop Supply Chain* (CLSC) telah banyak dikembangkan. Pada Tabel 2.1 teradapat posisi penelitian yang membandingkan penelitian ini dengan penelitian-penelitian sebelumnya yang memiliki beberapa kesamaan dan perbedaan. Posisi penelitian ini dibandingkan akan dengan penelitian yang dilakukan oleh Kannan, G., Sasikumar, P., & Devita, K., (2009) dan Subramanian P., Ramkumar N., Narendran T.T., dan Ganesh K., (2012), yang merupakan penelitian-penelitian yang menjadi rujukan dalam penelitian ini.

Tabel 2.1 Posisi Penelitian

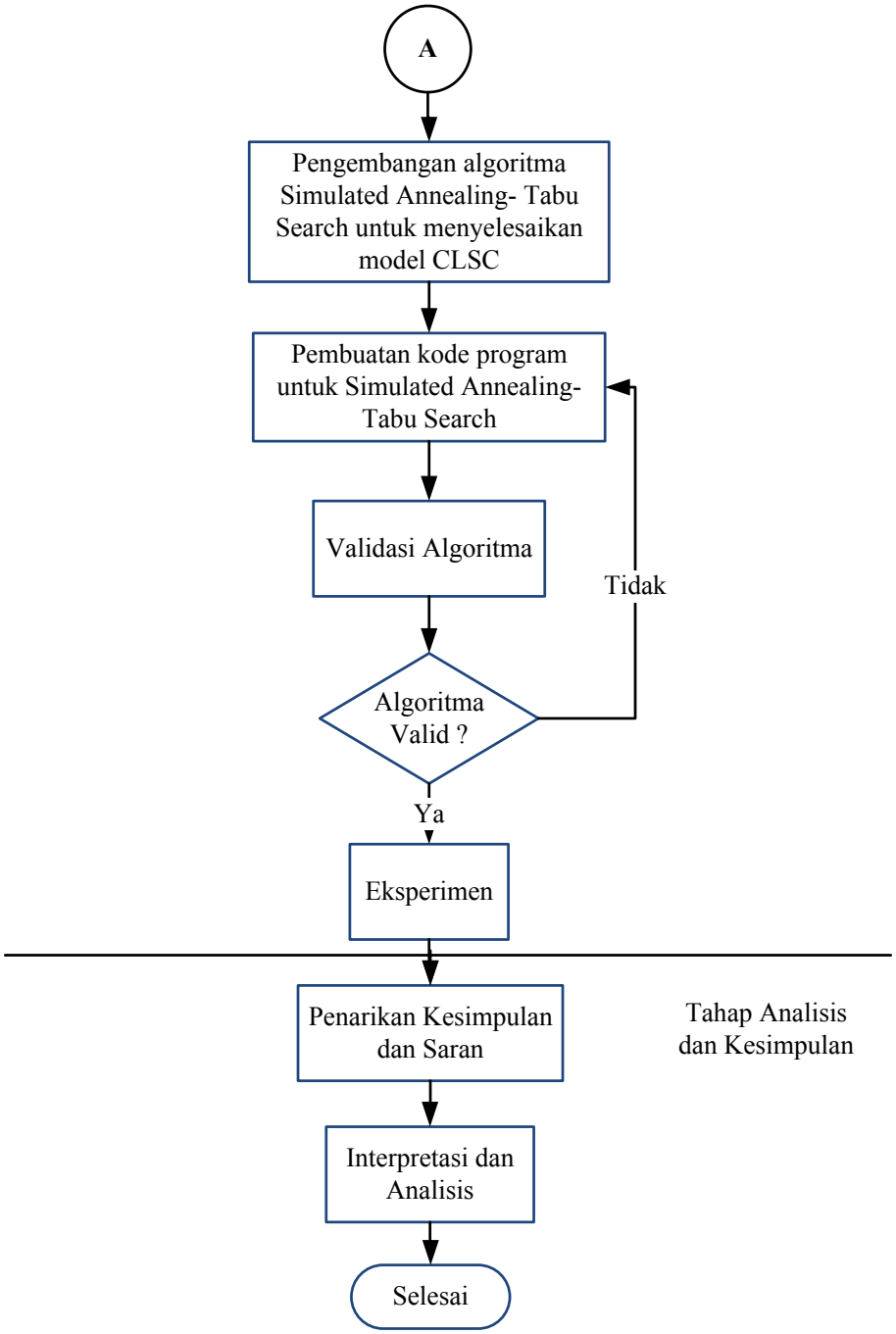
PENELITIAN	TEKNIK SOLUSI	DESKRIPSI MODEL
Kannan, G., Sasikumar, P., & Devita, K., (2009) : A Genetic Algorithm Approach for Solving A *Closed Loop Supply Chain* Model : A Case of Battery Recycling	Genetic Algorithm	1. Menggunakan node yang terdiri dari manufacturer, distributor, wholesaler, retailer, collection centre.
		2. Mempertimbangkan biaya dan waktu processing
Subramanian P., Ramkumar N., Narendran T.T., & Ganesh K., (2012) : A Technical Note on 'Analysis of *Closed Loop Supply Chain* Using Genetic Algorithm And Particle Swarm Optimization	Genetic Algorithm dan Particel Swarm Optimization	1. Menggunakan node yang terdiri dari supplier, manufacturing plants, distributors, wholeselers, retailers, initial collection points, disposal sites, recycling plants.
		2. Mempertimbangkan biaya dan waktu processing
Penelitian ini : Algoritma *Simulated Annealing* untuk Penyelesaian *Closed Loop Supply Chain* (CLSC) Studi Kasus PT. X	*Simulated Annealing* dan *Tabu Search*	1. Menggunakan node yang terdiri dari manufacturing plants, warehouse, distributor
		2. Tidak mempertimbangkan biaya dan waktu processing

BAB 3
METODOLOGI PENELITIAN

Pada bagian metodologi penelitian ini akan diuraikan langkah-langkah sistematis dan terarah yang akan dijadikan acuan sebagai kerangka penelitian. Metodologi penelitian dalam penelitian tugas akhir ini terdiri dari tahap pengumpulan data, pengembangan model, pengembangan algoritma, dan analisis dan kesimpulan.



Gambar 3.1 Tahapan Metodologi Penelitian



Gambar 3.2 Tahapan Metodologi Peneletian (Lanjutan)

3.1 Tahap Pengumpulan Data

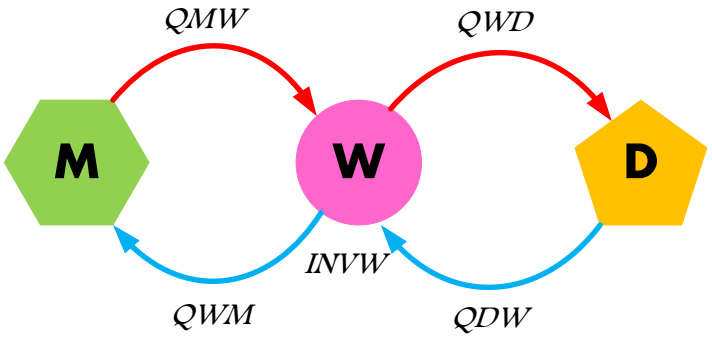
Metodologi penelitian ini dimulai dengan generate data sebagai data yang nantinya digunakan sebagai input dalam pengerjaan Tugas Akhir ini. Data yang dibutuhkan adalah data untuk proses *Closed Loop Supply Chain* meliputi :

1. Data jaringan distribusi, jarak distribusi
2. Data kapasitas produksi pabrik, kapasitas *warehouse, demand* disetiap *distributor*
3. Data *inventory cost* disetiap fasilitas
4. Data penggunaan barang *reuse*

3.2 Tahap Pengembangan Model Matemastis

Dalam tahap ini dilakukan pembuatan Model untuk permasalahan *Closed Loop Supply Chain* dari model CLSC yang telah dikembangkan oleh Subramanian P., Ramkumar N., Narendran T.T., & Ganesh K. (2012) yang kemudian disesuaikan dengan model CLSC.

Dilakukan pengembangan model dengan disesuaikan objek amatan dengan jalur distribusi pada gambar 3.3



Gambar 3.3 Alur Distribusi

Jaringan distribusi dimulai dari supplier yang melakukan supply terhadap kebutuhan komponen baru yang nantinya dapat di reuse kepada manufaktur, manufaktur memiliki kapasitas produksi yang komponennya didapatkan dari supplier terkait barang baru dan warehouse terkait barang reuse, manufaktur akan mengalokasikan hasil produksi pada warehouse yang memiliki kapasitas gudang, warehouse nantinya akan mendistribusikan kepada distributor yang ada sesuai demand yang dibutuhkan.

3.3 Tahap Pengembangan Algoritma

Dalam tahapan ini dilakukan pengembangan algoritma SA yang akan digunakan untuk menyelesaikan permasalahan *Closed Loop Supply Chain* ini.

3.3.1 Pembuatan Kode Program untuk SA

Pembuatan kode program untuk simulated annealing dari model *closed loop supply chain* yang sudah fix dari tahap sebelumnya.

1. Tentukan parameter : Temperatur awal (T_0), Faktor pereduksisi temperaur (c), siklus penurunan suhu (n) dan stopping criteria
2. Bangkitkan solusi awal (x_0) secara random
3. Hitung nilai fungsi tujuan
4. Ulangi langkah berikut hingga *stopping criteria* tercapai
   a. Update iterasi = i+1, dan iterasi siklusi = p+1
   b. Jika p = n, update temparature T=T*c, p=0

      Jika tidak , T=T
   c. Bangkitkan solusi baru berdasarkan solusi sebelumnya

      swap*, *slide*,** atau ***flip
   d. Hitung nilai fungsi tujuan F(x)

      Jika solusi baru lebih baru lebih baik dari solusi awal

      set x = x baru

      Jika tidak

      cek kriteria metropolis, $e^{-E/kT}$

      Bangkitkan bilangan random (r)

      Jika r > $e^{-E/kT}$

      set x = x baru

      Jika tidak, x = x lama

   Cek stopping criteria, Jika dipenuhi berhenti , jika tidak kembali ke a

Berikut ini adalah penjelasan yang dari pengembangan algoritma yang digunakan untuk menyelesaiakan permasalahan *Closed Loop Supply Chain* :

1. Menentukan parameter awal

   Dalam *simulated annealing* terdapat beberapa parameter yang digunakan yang akan mempengaruhi proses pencarian solusi. Parameter-parameter tersebut adalah temperatur awal, faktor pereduksi temparaturdan siklus penurunan temperatur. Temperatur awal dimulai dari temperatur yang tinggi yang akan turun seiring dengan

bertambahnya iterasi. Faktor pereduksi suhu merupakan nilai yang menjadi faktor pengurang temperatur setelah beberapa kali iterasi sesuai dengan nilai siklus penurunan temperatur (c). Faktor pereduksi suhu ini bernilai antara 0 dan 1 sehingga setelah n kali iterasi siklus (p), temperatur akan semakin turun yang mengakibatkan probabilitas penerimaan solusi yang lebih buruk akan semakin kecil.

2. Membangkitkan solusi awal secara random

   Pembangkitan solusi awal untuk permasalahan *Closed Loop Supply Chain* dengan simulated annealing dengan cara membangkitkan random permutasi untuk urutan distributor yang dituju, random permutasi untuk urutan warehouse, random permutasi untuk urutan jenis produk, dan random permutasi untuk urutan barang jadi atau botol kosong untuk stage pertama. Kemudian untuk stage kedua berhubungan dengan pengaturan gudang untuk produk sisa produksi yang tidak dikirimkan ke distributor dilakukan pembangkitan random permutasi untuk urutan *warehouse*, random permutasi untuk urutan produk.

3. Menghitung nilai fungsi tujuan

   Perhitungan fungsi tujuan dilakukan dengan memperhitungkan tujuan yang ingin dicapai pada tahapan yang dilakukan dengan mempertimbangkan konstrain-konstrain yang membatasi.

4. Menperbaharui iterasi dan siklus

   Update nilai iterasi dilakukan untuk mengetahui sampai kapan pengulangan metode ini dilakukan dan bisa digunakan sebagai salah satu kriteria pemberhentian. *Update* siklus ini dilakukan untuk memberikan jadwal kapan dilakukan penurunan temperatur. Jika siklus telah bernilai sama dengan n maka akan dilakukan penurunan temperatur sebesar faktor pereduksi yang telah ditentukan di awal.

5. Membangkitkan solusi baru

   Solusi baru dibangkitkan berdasarkan nilai dari solusi sebelumnya dengan melakukan dengan cara melakukan *swap*, *slide*, atau *flip*.

6. Membandingkan solusi baru dengan solusi sebelumnya

   Beradasarkan solusi baru yang didapatkan kemudian dilakukan perhitungan *fitness function* berdasarkan nilai fungsi tujuannya. Jika solusi yang baru lebih baik daripada solusi yang lama, maka secara langsung solusi yang baru dinyatakan sebagai solusi terbaik untuk saat ini. Jika solusi yang baru tidak lebih baik daripada solusi yang lama, maka akan dilakukan kriteria metropolis yang ditentukan berdasarkan probabilitas Boltzman. Perhitungan probabilitas boltzman ini kemudian dibandingkan dengan bilangan random yang dibangkitkan untuk menentukan apakah solusi yang baru yang tidak lebih baik tersebut diterima atau tidak. Ketika bilangan random ≤ probabilitas boltzman maka nilai x baru akan diterima untuk sementara sedangkan jika bilangan random > probabilitas Boltzman maka solusi baru akan ditolak, dan solusi lama dinyatakan sebagai solusi baru.

7. Kriteria pemberhentian

   Terdapat beberapa parameter kriteria pemberhentian yang dapat digunakan, diantaranya adalah nilai dari temperatur yang sudah sangat kecil, nilai dari solusi yang sudah tidak berubah secara signifikan dengan variansi yang sudah sangat kecil, jumlah iterasi yang sudah sangat banyak, dan sebagainya. Dalam penelitian ini, kriteria pemberhentian yang digunakan adalah nilai dari solusi yang sudah tidak berubah secara signifikan dengan variansi yang sudah sangat kecil, jumlah iterasi maksimum hingga nilai temperatur yang sangat kecil. Jika kriteria pemberhentian belum tercapai, maka iterasi akan diulang dengan meng*update* lagi jumlah iterasi dan siklus temperatur dan menggunakan solusi sementara yang didapatkan dari iterasi sebelumnya. Tetapi, jika parameter kriteria pemberhentian telah tercapai, maka selesai langkah-langkah dalam *simulated annealing* dengan *output* yang didapatkan adalah nilai dari *decision variabel* serta waku komputasi yang diperlukan.

3.3.2 Pembuatan Kode Program untuk SA-TS

Pembuatan kode program *Simulated Annealing-Tabu Search* diambil dari algoritma *Simulated Annealing* untuk inisiasi solusi awal, pembangkitan solusi baru, dan update solusi model. Algoritma *Tabu Search* untuk penyelesaikan *Closed Loop Supply Chain* ini terletak pada penggunaan *Tabu List.*Tabu List sendiri berfungsi untuk menyimpan solusi buruk yang sudah pernah dilakukan. Kemudian ketika dilakukan pembangkitan solusi baru, solusi tersebut akan dibandingkan dengan Tabu List apabila solusi baru sudah masuk tabu list maka tidak perlu dihitung nilai fungsi tujuannya, kemudian kembali dilakukan pembangkitan solusi baru, begitu seterusnya. Berikut ini merupakan tahapan pada SA-TS untuk penyelesesain permasalahan CLSC.

1. Tentukan parameter : Temperatur awal (T_0), Faktor pereduksisi temperaur (c), siklus penurunan suhu (n) dan stopping criteria
2. Bangkitkan solusi awal (x_0) secara random
3. Hitung nilai fungsi tujuan
4. Ulangi langkah berikut hingga *stopping criteria* tercapai
   a. Update iterasi = i+1, dan iterasi siklusi = p+1
   b. Jika p = n, update temparature T=T*c, p=0
      Jika tidak , T=T
   c. Bangkitkan solusi baru berdasarkan solusi sebelumnya
      swap*, *slide*,** atau ***flip
   d. Bandingkan solusi baru dengan *Tabu List*
      Jika solusi baru = tabu list
         Bangkitkan solusi baru
      Jika tidak, lanjutkan
   e. Hitung nilai fungsi tujuan F(x)
      Jika solusi baru lebih baik dari solusi awal
         set solusi lama = solusi baru
         set solusi lama dimasukkan tabu list
      Jika tidak
         cek kriteria metropolis, $e^{-E/kT}$

Bangkitkan bilangan random (r)

Jika r > $e^{-E/kT}$

set solusi lama = solusi baru

set solusi lama dimasukkan tabu list

Jika tidak, solusi baru = solusi lama

set solusi baru dimasukkan tabu list

Cek stopping criteria, Jika dipenuhi berhenti , jika tidak kembali ke a.

3.3.3 Verifikasi dan Validasi Algoritma SA-TS

Tahap selanjutnya dalam penelitian ini adalah melakukan verifikasi dan validasi algoritma. Verifikasi model dilakukan untuk memastikan bahwa algoritma yang ada sudah dapat menyelesaikan permasalahan dari model matematis CLSC atau tidak. Pada penelitian ini, validasi dilakukan dengan cara memeriksa hasil eksperimen untuk permasalahan sederhana dengan metode *enumerasi.* Jika hasil eksperimen kecil sama dengan hasil optimal dari enumerasi maka model yang telah dibuat dikatakan valid

3.3.4 Eksperimen

Eksperimen dilakukan untuk mengetahui sensitivitas dengan merubah beberapa parameter-parameter yang digunakan dalam algoritma ini, yang yang dilakukan dengan menerjemahkan model sesuai dengan algoritma *Simulated Annelaing* kedalam basa kode *software.* Eksperimen dilakukan terhadap data set *Closed Loop Supply Chain* yang telah didapatkan.

3.4 Tahap Analisis dan Kesimpulan

Pada tahap analisis dan kesimpulan ini dilakukan tahapan sebagai berikut.

3.4.1 Analisis dan Interpretasi

Pada tahap analisis dan interpretasi ini akan dilakukan analisis terkait hasil dari eksperimen yang dilakukan. Analisis dilakukan untuk mengetahui performansi dari algoritma yang dikembangkan untuk menyelesaikan suatu data. Selain itu juga akan dilakukan analisa terkait pengaruh penetapan kombinasi parameter-parameter awal SA, sehingga diperoleh rekomendasi untuk penggunaan parameter awal yang baik.

3.4.2 Penarikan Kesimpulan dan Saran

Setelah selesai tahap analisis, akan dilakukan penarikan kesimpulan terkait hasil eksperimen yang dilakukan. Kesimpulan didapatkan berdasarkan hasil analisis dan interpretasi hasil dari eksperimen yang telah dilakukan. Kesimpulan juga menjawab tujuan yang ingin dicapai dalam melakukan penelitian. Setelah itu akan diberikan saran-saran yang dapat dijadikan sebagai rekomendasi untuk acuan penelitian selanjutnya.

BAB 4
PENGEMBANGAN MODEL DAN ALGORITMA

Pada bab pengembangan model dan algoritma ini akan dijelaskan tentang tahap-tahap yang dilakukan dalam mengembangkan model *Closed Loop Supply Chain* dan algoritma *Simulated Annealing* yang digunakan dalam penyelesaian permasalahan alokasi.

4.1 Pengembangan Model *Closed Loop Supply Chain*

Pengembangan model *Closed Loop Supply Chain* pada bab ini didasarkan pada model konseptual yang telah dibuat dalam Gambar 3.3 yang merupakan gambaran distribusi *Closed Loop Supply Chain* pada perusahaan X. Model ini dikembangkan dari Subramanian P., Ramkumar N., Narendran T.T., & Ganesh K. (2012). Dengan fungsi tujuan minimasi total biaya *Closed Loop Supply Chain* karena pada dasarnya biaya merupakan salah satu parameter yang diperhitungkan dalam penentuan kebijakan alokasi dan distribusi.

Model *Closed Loop Supply Chain* digambarkan dalam model matematis yang terdiri dari beberapa bagian yang dipertimbangkan serta konstrain atau batasan dalam model.

4.1.1 Model Matematis *Closed Loop Supply Chain*

Berikut ini adalah model yang sudah dikembangkan dan disesuaikan dengan objek amatan di Indonesia yang akan dipakai untuk penyelesaian masalah CLSC :

Notasi

Indeks

m = manufacturer (m=1, 2,...,M)

w = warehouse (w=1, 2,...,W)

d = distributor (d=1, 2,...,D)

p = produk (p=1, 2,...,P)

t = periode (t=1, 2,...,T)

Biaya Dari Fungsi Tujuan

$TCMW$ = total biaya transportasi dari manufactur ke warehouse

$TCWD$ = total biaya transportasi dari warehouse ke distributor

$TCDW$ = total biaya transportasi dari distributor ke warehouse

$TCWM$ = total biaya transportasi dari warehouse ke manufactur

$TCIM$ = biaya penyimpanan di manufacturer

$TCIW$ = biaya penyimpanan di warehouse

Input Parameter

DEM_{dpt} = demand dari produk p di distributor d pada periode t

ICW_{pm} = biaya penyimpanan per waktu per produk p di warehose w

CP_m = kapasitas produksi manufacturer m

REW_{pwt} = jumlah kembali produk p di warehose w pada periode t

REM_{pwt} = jumlah kembali produk p di manufacture m pada periode t

CMW_{mw} = biaya transportasi dari manufacturer m ke warehouse w

CWD_{wd} = biaya transportasi dari warehouse w ke distributor d

CDW_{dw} = biaya transportasi dari distributor d ke warehouse w

CWM_{wm} = biaya transportasi dari warehouse w ke manufacturer m

CW_w = kapasitas pada setiap warehouse w

Decision Parameter

QMW_{mwpt} = Kuantitas yang berpindah dari produk p pada periode t dari manufacture m ke warehouse w

QWD_{wdpt} = Kuantitas yang berpindah dari produk p pada periode t dari warehouse w ke distributor d

QDW_{dwpt} = Kuantitas yang berpindah dari produk p pada periode t dari distributor d ke warehouse w

QWM_{wmpt} = Kuantitas yang berpindah dari produk p pada periode t dari warehouse w ke manufacture m

$INVW_{wpt}$ = *Inventory* dari produk p pada periode t di warehouse w

Minimise Z = TCMW + TCWD + TCDW + TCWM + TCIM + TCIW,

$$\text{TCMW} = \sum_{m}\sum_{w}\sum_{p}\sum_{t} QMW_{mwpt}.CMW_{mw}$$

$$\text{TCWD} = \sum_{w}\sum_{d}\sum_{p}\sum_{t} QWD_{wdpt}.CWD_{wd}$$

$$\text{TCDW} = \sum_{d}\sum_{w}\sum_{p}\sum_{t} QDW_{dwpt}.CDW_{dw}$$

$$\text{TCWM} = \sum_{w}\sum_{m}\sum_{p}\sum_{t} QWM_{wmpt}.CWM_{wm}$$

$$\text{TCIW} = \sum_{w}\sum_{p}\sum_{t} INVW_{wpt}.ICW_{wp}$$

Konstrain

Manufacturer

Konstrain distribusi keluar manufaktur

Jumlah produk yang keluar dari manufaktur =jumlah bahan baku yang masuk ke manufaktur dari supplier dan product return dari warehouse

$$\sum_{p}\sum_{w} QMW_{smpt} = \sum_{p}\sum_{s} QSM_{smpt} + \sum_{p}\sum_{w} QWM_{wmpt}.\alpha \quad \forall m, \forall t.$$

Konstrain kapasitas produksi

Jumlah produk yang keluar dari manufaktur ke warehouse harus kurang dari atau sama dengan kapasitas produksi

$$\sum_{w} QMW_{smpt} \le CP_{m} \quad \forall m, \forall p, \forall t.$$

Warehouse

Konstrain kapasitas gudang

Jumlah inventory periode sebelumnya + jumlah produk masuk dari manufaktur + jumlah produk reuse dari distributor harus kurang dari atau sama dengan kapasitas gudang

$$\sum_{p} INVW_{wp(t-1)} + \sum_{p}\sum_{m} QMW_{mwpt} + \sum_{p}\sum_{d} REW_{dwpt} \le CW_{w}, \ \forall w, \forall t.$$

Konstrain aliran untuk gudang

Jumlah inventory periode sebelumnya + jumlah produk masuk dari manufaktur – jumlah produk keluar ke distributor + jumlah produk reuse dari distributor – jumlah produk reuse yang dibuang – jumlah produk resuse ke luar ke manufaktur – jumlah inventory saat ini = 0

$$INVW_{wp(t-1)} + \sum_{m} QMW_{mwpt} - \sum_{d} QWD_{wdpt} + \sum_{d} REW_{dwpt} - DIW_{wpt} - \sum_{m} REM_{mwpt} - INVW_{wpt} = 0,$$

$$\forall w, \forall p, \forall t.$$

Distributor

Konstrain aliran untuk distributor

Jumlah inventory periode sebelumnya + jumlah produk masuk dari warehose – (jumlah inventory saat ini + jumlah produk reuse ke warehose + jumlah produk yang tidak kembali ke distributor

$$INVD_{dp(t-1)} + \sum_{w} QWD_{wdpt} - \left(INVD_{dpt} + \sum_{w} REW_{dwpt} + DID_{dpt} \right) = 0,$$

$$\forall d, \forall p, \forall t.$$

4.2 Pengembangan Algoritma *Simulated Annealing* untuk *Closed Loop Supply Chain*

Pada penelitian ini dikembangkan algoritma *SA* untuk *Closed Loop Supply Chain*. Algoritma *SA* yang dgunakan untuk menyelesaikan permasalahan *Closed Loop Supply Chain* dapat dilihat pada Lampiran. Pada tabel 4.1, tabel 4.2, dan tabel 4.3 merupakan data kecil yang digunakan untuk menguji algoritma dan model matematis adalah data dari Kannan, G., Sasikumar, P., & Devita, K., (2009) :

Tabel 4.1 Data *Demand,* Botol Balik ke *Warehouse,* Botol Balik ke *Manufacture.*

Distibutor		*Demand*	Botol Balik ke *Warehouse*	Botol Balik ke *Manufacture*
	D1	7500	6700	6000
	D2	8000	7500	7000

Tabel 4.2 Jarak dari *Manufacture* ke *Warehouse, Warehouse* ke *Manufacture, Warehouse* ke *Distributor, Distributor* ke *Warehouse.*

		Manufacture	***Warehouse***		***Distributor***	
		1	W1	W2	D1	D2
Manufacture	1	0	25	18	-	-
Warehouse	W1	23	0	-	200	150
	W2	20	-	0	180	175
Distibutor	D1	-	190	140	0	-
	D2	-	185	165	-	0

Tabel 4.3 Data Kapasitas Produksi dan *Warehouse*

		Kapasitas
		C
Produksi	1	16500
Warehouse	W1	15000
	W2	18000

Dalam kasus kecil ini terdapat satu jenis produk, dua *warehouse* dan dua *distributor*. Berikut ini penjelasan langkah-langkah yang terdapat dalam algoritma SA yang digunakan :

Langkah 1 : Inisialisasi Parameter

Parameter algoritma SA CLSC untuk menyelesaikan kasus sederhana ini adalah sebagai berikut:

- Faktor pereduksi temperatur (c)=0.85
- Siklus penurunan temperature=10
- Temperature awal=500
- Jumlah iterasi=10

Langkah 2: Pembangkitan Solusi Awal

Solusi awal dalam algoritma *Simulated Annealing* dibangkitkan secara *random.* Solusi awal yang digunakan terdiri dari dua *stage*, *stage* pertama terdiri dari empat komponen yaitu urutan *distributor*, urutan *warehouse*, urutan produk, urutan jenis produk, dan cara pengisian *warehouse*. *Stage* kedua terdiri dari tiga

komponen yaitu urutan *warehouse*, urutan produk, dan cara pengisian *warehouse*. Berikut ini adalah masing-masing cara yang digunakan untuk membangkitkan sampel awal:

- Solusi awal *stage* 1, urutan *distributor, warehouse,* produk, jenis produk, dan cara pengisian *warehouse*

  Urutan dibentuk dengan membangkitkan random permutasi sejumlah komponen.

  i = urutan produk

  j = urutan jenis produk

  k = urutan *distributor*

  l = urutan *warehouse*

  rn = cara pengisian *warehouse*

```
i =
     1
j =
     2     1

k =
     1     2

l =
     2     1

rn =
     1     2
     2     1
```

  Matriks solusi ini menyatakan bahwa komponen (i) hanya terdapat satu produk maka tidak ada urutan yang berarti, untuk komponen (j) terdapat dua jenis dengan urutan produk balik terlebih dahulu dahulu kemudian produk baru, untuk komponen (k) terdapat dua *distributor* dengan urutan pengambilan dan pengiriman *distributor* satu terlebih dahulu kemudian *distributor* dua, untuk komponen (l) terdapat dua *warehouse* dengan urutan pengambilan dan pengiriman *warehouse* dua terlebih dahulu kemudian distributor satu, dan yang terakhir adalah komponen (rn) yang

menunjukkan cara pengisian setiap *warehouse* untuk *warehouse* 1 ditunjukkan oleh baris pertama yang menyatakan bahwa diperbolehkan sebagian *demand* untuk mengisi sisa tempat di *warehouse* 1, untuk *warehouse* 2 ditunjukkan oleh baris kedua yang menyatakan bahwa tidak diperbolehkan sebagian *demand* untuk mengisi sisa tempat di *warehouse* 2.

- Solusi awal *stage* 2, urutan *warehouse,* produk, dan cara pengisian *warehouse*

  Urutan dibentuk dengan membangkitkan random permutasi sejumlah komponen.

  s = urutan produk

  l2 = urutan *warehouse*

  rn = cara pengisian *warehouse*

```
s =
     1


l2 =
     2     1


rn =
     1     2
     2     1
```

Matriks solusi ini menyatakan bahwa komponen (s) hanya terdapat satu produk maka tidak ada urutan yang berarti, untuk komponen (l2) terdapat dua *warehouse* dengan urutan pengambilan dan pengiriman *warehouse* dua terlebih dahulu kemudian distributor satu, dan yang terakhir adalah komponen (rn) yang menunjukkan cara pengisian setiap *warehouse* untuk *warehouse* 1 ditunjukkan oleh baris pertama yang menyatakan bahwa diperbolehkan sebagian *demand* untuk mengisi sisa tempat di *warehouse* 1, untuk *warehouse* 2 ditunjukkan oleh baris kedua yang menyatakan bahwa tidak diperbolehkan sebagian *demand* untuk mengisi sisa tempat di *warehouse* 2.

Langkah 3: Perhitungan Fungsi Tujuan

Perhitungan fungsi tujuan dilakukan setelah terlebih dahulu memastikan bahwa solusi yang digenerate tidak melanggar konstrain dari model *Closed Loop Supply Chain*. Tahapan perhitungan fungsi tujuan secara lebih lengkap dapat dilihat pada Lampiran A bagian *pseudo-code* algoritma. Dengan nilai total *cost* CLSC sebagai berikut :

```
TOT1 =
    19225125
```

Langkah 4: Pembangkitan Solusi Baru

Langkah dalam membangkitkan solusi baru *stage* pertama yang terdiri dari empat komponen yaitu urutan *distributor*, urutan *warehouse*, urutan produk, urutan jenis produk, dan cara pengisian *warehouse* serta s*tage* kedua terdiri dari tiga komponen yaitu urutan *warehouse*, urutan produk, dan cara pengisian *warehouse* adalah dengan cara melakukan *swap*, *slide*, atau *flip*. Uuntuk menentukan apakah metode *swap*, *slide*, atau *flip* yang akan digunakan, maka dibangkitkan bilangan *random* dan dilakukan penentuan dengan kriteria sebagai berikut:

- Jika bilangan *random* yang dibangkitkan adalah antara 0 – 0.33, maka metode *flip* akan digunakan.
- Jika bilangan *random* yang dibangkitkan adalah antara 0.34 – 0.67, maka metode *swap* akan digunakan.
- Jika bilangan *random* yang dibangkitkan adalah antara 0.68 – 1, maka metode *slide* akan digunakan.

Setelah mendapatkan solusi baru maka dilakukan perhitungan fungsi tujuan yang dihasilkan dengan menggunakan solusi baru sebagai berikut :

```
i =
     1

j =
     1     2

k =
```

```
     1     2

l =
     1     2

rn =
     1     2
     2     1

s =
     1

l2 =
     1     2

rn =
     1     2
     2     1
```

Berdasarkan solusi baru yang telah didapatkan maka dilakukan perhitungan total cost CLSC yang didapatkan sebesar:

```
TOT2 =
    20831100
```

Langkah 5: Membandingkan Solusi Lama dengan Solusi Baru

Setelah mendapatkan solusi baru kemudian dilakukan perhitungan selisih antara fungsi tujuan yang didapatkan dengan solusi baru dan solusi yang lama. Jika nilai solusi yang baru lebih baik dibandingkan dengan solusi yang sebelumnya maka solusi baru akan diterima. Tetapi, jika solusi baru tidak lebih baik dibandingkan dengan solusi yang lama, maka akan dilakukan perhitungan kriteria metropolis untuk menentukan apakah solusi yang baru dapat diterima atau tidak. Perhitungan kriteria metropolis dilakukan dengan membandingkan antara probabilitas Boltzman dengan bilangan random, jika bilangan random lebih kecil dari kriteria metropolis maka solusi yang baru akan diterima jika tidak maka solusi yang baru aka ditolak.

Dalam perhitungan total *cost* CLSC baru lebih buruk dari cost CLSC, maka dapat dilakukan perhitungan kriteria metropolis untuk mengetahui probabilitas penerimaan:

Temperatur=500

$P(E) = e^{-\Delta E/kT}$

$P(E) = e^{-1605975/500}$

$P(E) = 0$

Selanjutnya digenerate bilangan random yang didapatkan sebesar 0.254, karena nilai bilangan random lebih dari probabilitas Boltzman maka dapat disimpulkan solusi yang baru tidak dapat diterima sehingga dapat ditetapkan nilai solusi saat ini sama dengan solusi lama kemudian komputasi dilakukan hingga mencapai *stopping criteria*.

Langkah 6: *Update* Iterasi, Siklus dan Temperatur

Update iterasi dan siklus dilakukan setelah didapatkan solusi yang baru. Sedangkan *update* nilai temperatur dilakukan setelah tercapai jumlah siklus sebesar n yang merupakan nilai maksimal jumlah siklus. Setelah mencapai jumlah siklus n, temperatur akan direduksi dengan menggunakan faktor pereduksi temperatur (c). Selain itu, setelah mencapai jumlah siklus n, nilai siklus kembali ditetapkan menjadi sama dengan satu.

Langkah 7: *Stopping Criteria*

Penetapan *stopping criteria* akan menentukan kapan algoritma berhenti melakukan komputasi. Dalam penelitian ini *stopping criteria* yang digunakan adalah jumlah iterasi maksimum. Jika *stopping criteria* belum tercapai, maka iterasi akan diulang dengan meng-*update* iterasi dan siklus serta tempaeratur. Sementara itu, jika *stopping criteria* telah tercapai maka akan ditampilkan *output*an dari algoritma *simuated annealing* yang telah dibuat yakni menampilkan nilai fungsi tujuan, nilai variabel keputusan, serta waktu komputasi.

4.3 Verifikasi dan Validasi

Verifikasi model dilakukan untuk memastikan bahwa algoritma yang ada sudah dapat menyelesaikan permasalahan dari model matematis CLSC atau tidak.

Validasi merupakan tahapan yang dilakukan untuk mengetahui apakah model yang dibuat telah mampu merepresentasikan permasalahan yang diselesaikan. Validasi dalam penelitian dilakukan untuk juga untuk algortma *Simulated Annealing* dan *Simulated Annealing- Tabu Search.*

4.3.1 Verifikasi dan Validasi Algoritma *Simulated Annealing*

Verifikasi algoritma dilakukan dengan membandingkan apakah logika perhitungan dalam algoritma sudah sama dengan logika perhitungan manual. Selain itu dilakukan dengan menunjukkan bahwa tidak terdapat error dalam melakukan komputasi dengan *software* MATLAB. Dalam algoritma SA untuk CLSC ini sudah sama antara perhitungan algoritma SA dengan algoritma perhitungan manual yang dilakukan dengan enumerasi.

Validasi Algoritma dilakukan dengan cara membandingkan hasil *output* komputasi algoritma SA dengan hasil *output* enumerasi untuk permasalahan kecil. Berikut ini adalah hasil perhitungan dari algoritma SA:

```
W_New =
        7500
        9000

QWD =
           0        7500
        7500         500

REM(:,:,1) =
        6000        7000

REM(:,:,2) =
        6375        5695

QDW =
           0        6700
        7500           0
INVW(:,:,1) =
     0     0

INVW(:,:,2) =
```

```
     0     1000

TCMW =
      646875
      558900

TCWM =
  1.0e+005 *
    4.7610
    4.8300

TCWD =
           0     4657500
     3881250      301875

TCDW =
           0     4276275
     3622500           0

TCIW =
     0     0

total_SA =
    18904275

waktu =
    0.6240
```

Berdasarkan *output* yang dihasilkan oleh komputasi dapat disimpulkan bahwa pada permasalahan yang sama algoritma SA dapat menghasilkan solusi yang sama dengan yang dihasilkan pada enumerasi sehingga dapat disimpulkan bahwa algoritma SA untuk CLSC ini adalah valid.

4.3.2 Verifikasi dan Validasi Algoritma *Simulated Annealing - Tabu Search*

Verifikasi algoritma dilakukan dengan membandingkan apakah logika perhitungan dalam algoritma sudah sama dengan logika perhitungan manual. Selain itu dilakukan dengan menunjukkan bahwa tidak terdapat error dalam

melakukan komputasi dengan *software* MATLAB. Dalam algoritma SA-TS untuk CLSC ini sudah sama antara perhitungan algoritma SA-TS dengan algoritma perhitungan manual yang dilakukan dengan enumerasi.

Validasi Algoritma dilakukan dengan cara membandingkan hasil *output* komputasi algoritma SA-TS dengan hasil *output* enumerasi untuk permasalahan kecil. Berikut ini adalah hasil perhitungan dari algoritma SA-TS:

```
W_New =
        7500
        9000

QWD =
           0        7500
        7500         500
W_Ret =
        7500
        6700

QDW =
           0        6700
        7500           0

INVW(:,:,1) =
     0     0

INVW(:,:,2) =
     0     1000

total_SA_TS =
    18904275

waktu =
    0.2964
```

Berdasarkan *output* yang dihasilkan oleh komputasi dapat disimpulkan bahwa pada permasalahan yang sama algoritma SA-TS dapat menghasilkan solusi yang sama dengan yang dihasilkan pada enumerasi sehingga dapat disimpulkan bahwa algoritma SA-TS untuk CLSC ini adalah valid.

BAB 5
EKSPERIMEN DAN ANALISIS

Pada bab eksperimen dan analisis ini akan dijabarkan mengenai analisis hasil eksperimen dari bab pengujian algoritma. Di samping itu juga dilakukan analisis perbandingan dari algoritma SA-TS dengan algoritma SA yang dikembangkan.

5.1 Deskripsi Data Uji

Data yang digunakan untuk melakukan pengujian pada algoritma SA dan algoritma SA-TS. Tediri dari beberapa jenis data yang dibedakan berdasarkan ukuran datanya. Ukuran data bergantung pada jumlah produk, periode perencanaan, jumlah *warehouse* dan jumlah *distributor* yang diperhitungkan. Pada Tabel 5.1 ini adalah data-data yang digunakan dalam melakukan eksperimen:

Tabel 5.1 Data Uji

Data ke-	Ukuran data uji				Sumber data
	Produk	Periode	*Warehouse*	*Distributor*	
1	1	1	2	2	Data *Generate*
2	2	3	3	5	Data *Generate*
3	4	7	8	31	Data Demand dan *Generate*

Data ke-1 merupakan data yang digunakan untuk melakukan validasi, sementara itu data uji lainnya yang digunakan pada penelitian ini secara lengkap dapat dilihat pada Lampiran.

5.2 Eksperimen

Eksperimen dilakukan dengan menggunakan *software* MATLAB 7.10.0 (R2010a) untuk menyelesaikan permasalahan CLSC dengan menggunakan algoritma SA dan algoritma SA-TS. Spesifikasi komputer yang digunakan adalah Intel ® Core™ i5 2.53 G.Hz (CPU), RAM 2.00 GB.

5.2.1 Eksperimen dengan Algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search*

Pada bagian ini akan dijelaskan tentang eksperimen yang dilakukan dengan menggunakan algoritma SA dan algoritma SA-TS untuk menyelesaiakan problem *Closed Loop Supply Chain.*

5.2.1.1 Eksperimen Uji Parameter

Dalam algoritma SA terdapat beberapa parameter yang digunakan untuk input yaitu temperetur awal (To), faktor pereduksi temperatur (c) dan banyaknya siklus (n). Nilai parameter-parameter ini akan mempengaruhi kualitas dan kecepatan dalam menghasilkan solusi sehingga perlu dilakukan uji parameter untuk menentukan nilai parameter yang tepat untuk mendapatkan solusi yang terbaik. Nilai faktor pereduksi temperatur (c) yang diuji adalah 0.3, 0.6, dan 0,9. Nilai temperetur awal (To) yang diuji adalah 300, 500, dan 700. Berikut ini adalah hasil eksperimen uji parameter untuk nilai factor pereduksi temperature (c) yang dilakukan dengan menggunakan Data ke-2:

Tabel 5.2 Uji Parameter Faktor Pereduksi Temperatur

c=0.3, n=10, maxiter=2000, To=600			
No Replikasi	Solusi	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	2 1 1 2 4 3 2 1 5 2 3 1 1 3 2 1 2 2 1 2 1 2 1	1.2048	16.8169
2	2 1 1 2 4 3 2 1 5 2 3 1 1 3 2 1 2 2 1 2 1 2 1	1.2048	16.8013
3	2 1 2 1 1 4 3 2 5 1 2 3 1 3 2 1 2 1 2 1 2 2 1	1.2085	17.4409
Rata-rata		1.2060	17.0197

Tabel 5.3 Uji Parameter Faktor Pereduksi Temperatur (Lanjutan)

c=0.6, n=10, maxiter=2000, To=600			
No Replikasi	Solusi	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	2 1 1 2 4 3 2 1 5 2 3 1 1 3 2 1 2 2 1 2 1 2 1	1.2048	16.9573
2	2 1 1 2 4 3 2 1 5 2 3 1 1 3 2 1 2 2 1 2 1 2 1	1.2048	16.7857
3	2 1 1 2 4 3 2 1 5 2 3 1 1 3 2 1 2 2 1 2 1 2 1	1.2048	16.8793
Rata-rata		1.2048	16,8741
c=0.9, n=10, maxiter=2000, To=500			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	18.0805
2	1 2 2 1 3 2 5 4 1 1 3 2 2 3 1 2 1 1 2 2 1 2 1	1.2081	16.8735
3	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	17.3785
Rata-rata		1.2084	17.4442

Berdasarkan Tabel 5.2 dan 5.3 uji parameter dapat diketahui pada faktor pereduksi temperatur yang mana dapat dihasilkan kualitas solusi dengan waktu komputasi yang lebih cepat. Penilaian kualitas solusi didasarkan pada nilai rata-rata total biaya CLSC, dengan nilai rata-rata terbaik sebesar 1.2048 x 1e+004 pada temperatur produksi c=0.6. Jika dilihat dari rata-rata waktu komputasi, skenario dengan faktor pereduksi temperatur sebesar 0.6 memiliki waktu komputasi baik dan kualitas solusi yang dihasilkan paling baik. Karena itu dalam penelitian ini akan digunakan faktor pereduksi temperatur sebesar 0.6.

Setelah membandingkan perubahan parameter faktor pereduksi temperatur, maka berikutnya dilakukan uji perubahan parameter temperetur awal (To) yang

diuji adalah 300, 500, dan 700 dan jumlah iterasi maksimum sebesar 2000. Dapat dilihat pada Tabel 5.4 dan Tabel 5.5 merupakan hasil dari uji parameter perubahan jumlah siklus.

Tabel 5.4 Uji Parameter Temperatur Awal

c=0.6, n=10, maxiter=2000, To=300			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8793
2	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	16.8637
3	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	16.7701
Rata-rata		1.2073	16.8377
c=0.6, n=10, maxiter=2000, To=600			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8013
2	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8637
3	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.9573
Rata-rata		1.2048	16.8741

Tabel 5.5 Uji Parameter Temperatur Awal (Lanjuan)

c=0.6, n=10, maxiter=2000, To=900			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	16.9885
2	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8481
3	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.9105
Rata-rata		1.2060	16.9157

Dengan membandingkan hasil uji nilai temperature awal (t) dengan skenario temperature awal sebesar 300, 600, dan 900 dapat diambil kesimpulan bahwa parameter nilai temperature awal yang terbaik adalah pada temperature 600 karena menghasilkan rata-rata kualitas solusi yang lebih baik dibandingkan skenario lain meskipun waktu komputasinya lebih lambat. Berdasarkan hasil uji parameter yang telah dilakukan maka pada eksperimen selanjutnya akan digunakan nilai parameter faktor pereduksi temperatur sebesar 0.6 dan temperature awal sebesar 600 dengan *stopping criteria* yang digunakan adalah jumlah iterasi maksimum.

5.2.1.2 Eksperimen SA dan SA-TS pada Data Uji Ke-2

Pada eksperimen Data Uji Ke-2 menggunakan data sedang yang terdiri dari dua produk, tiga periode, tiga warehouse dan lima distributor, akan dilakukan eksperimen menggunakan algoritma *Simulated Annealing* dan algoritma *Simulated Annealing-Tabu Search.* Berikut adalah ekperimen sebanyak sepuluh replikasi untuk setiap algoritma :

Tabel 5.6 Hasil Eksperimen Data Uji 2 dengan *Simulated Annealing*

c=0.6, n=10, maxiter=2000, To=600			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	10.4365
2	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.6081
3	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.5925
4	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.6549
5	1 2 2 1 3 2 5 4 1 1 3 2 2 3 1 2 1 1 2 2 1 2 1	1.2081	10.5457
6	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.5925
7	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	10.4521
8	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.5145
9	1 2 2 1 1 4 3 2 5 3 1 2 1 2 3 2 1 2 1 1 2 1 2	1.2134	10.5145
10	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	10.6705
Rata-rata		1.2082	10.5582

Setelah dilakukan replikasi sebanyak sepuluh kali untuk algoritma *Simulated Annealing* maka dilakukan rekapitulasi pada Tabel 5.6. Selanjutnya dilakukan replikasi sebanyak sepuluh kali untuk algoritma *Simulated Annealing-Tabu Search* pada Tabel 5.7

Tabel 5.7 Hasil Eksperimen Data Uji 2 dengan *Simulated Annealing-Tabu Search*

c=0.6, n=10, maxiter=2000, To=600			
No Replikasi	Solusi Urutan	Total Biaya (1e+004)	Waktu Komputasi (detik)
1	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8013
2	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8637
3	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.9573
4	2 1 2 1 4 1 2 3 5 3 1 2 1 2 3 1 2 2 1 1 2 1 2	1.2085	16.8949
5	1 2 2 1 3 2 5 4 1 1 3 2 2 3 1 2 1 1 2 2 1 2 1	1.2081	16.8481
6	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8949
7	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8169
8	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.8481
9	1 2 2 1 3 2 5 4 1 1 3 2 2 3 1 2 1 1 2 2 1 2 1	1.2081	16.8637
10	1 2 2 1 4 1 2 5 3 3 1 2 3 1 2 1 2 2 1 2 1 2 1	1.2048	16.7233
Rata-rata		1.2058	16.8512

Setelah mendapatkan solusi untuk permasalahan ini dengan menggunakan *Simulated Annealing* yaitu pada Tabel 5.6 dan *Simulated Annealing-Tabu Search* yaitu Tabel 5.7, maka dilakukan perhitungan GAP atau selisih antara hasil yang didapat dari *Simulated Annealing* dan *Simulated Annealing-Tabu Search* dengan menggunakan perhitungan. Rumus GAP untuk fungsi tujuan minimasi total biaya adalah sebagai berikut :

$$GAP = \frac{(Hasil\ Perhitungan\ SA\_TS - Hasil\ Perhitungan\ SA)}{Hasil\ perhitungan\ SA\_TS}\ X\ 100\%$$

Tabel 5.8 Perbandingan Nilai Perubahan Total Biaya pada Data Uji 2

Total Biaya			
No Replikasi	SA-TS (1e+004)	SA (1e+004)	GAP
1	1.2048	1.2048	0.0000%
2	1.2048	1.2085	-0.3071%
3	1.2048	1.2085	-0.3071%
4	1.2085	1.2085	0.0000%
5	1.2081	1.2081	0.0000%
6	1.2048	1.2085	-0.3071%
7	1.2048	1.2048	0.0000%
8	1.2048	1.2085	-0.3071%
9	1.2081	1.2134	-0.4387%
10	1.2048	1.2085	-0.3071%
Rata-rata			-0,197%

Berikut ini adalah rumus yang digunakan untuk menghitung GAP waktu komputasi :

$$Perbandingan = \frac{waktu\ komputasi\ SA\_TS}{waktu\ komputasi\ SA}$$

Tabel 5.9 Perbandingan Waktu Komputasi pada Data Uji 2

Waktu Komputasi			
No Replikasi	SA-TS (detik)	SA (detik)	Perbandingan
1	16.8013	10.4365	0,6212
2	16.8637	10.6081	0,6290
3	16.9573	10.5925	0,6247
4	16.8949	10.6549	0,6307
5	16.8481	10.5457	0,6259
6	16.8949	10.5925	0,6270
7	16.8169	10.4521	0,6215
8	16.8481	10.5145	0,6241
9	16.8637	10.5145	0,6235
10	16.7233	10.6705	0,6381
Rata-rata			0,6266

5.2.1.3 Eksperimen SA dan SA-TS pada Data Uji Ke-3

Berikut ini adalah hasil dari eksperimen yang dilakukan untuk menyelesaikan permasalahan *CLSC* pada data uji ketiga dengan menggunakan algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search*. Data Uji Ke-3 ini merupakan data besar dengan empat produk, tujuh periode, 8 warehouse dan 31 distributor. Pada Tabel 5.10 merupakan rangkuman sepuluh replikasi yang meghasilkan solusi terbaik menggunakan algoritma SA, dan pada Tabel 5.11 merupakan rangkuman sepuluh replikasi terbaik dengan menggunakan algoritma SA-TS untuk permasalahan CLSC.

Tabel 5.10 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
1	1	4	3	2	1	5.4395	51.2307
	2	10	19	7	14		
	21	15	24	23	26		
	31	5	3	6	1		
	22	18	30	2	16		
	11	4	9	29	13		
	12	25	20	8	27		
	17	28	1	4	2		
	3	6	8	5	7		
	8	5	2	6	3		
	1	4	7	3	4		
	2	1	2	1	1		
	2	2	1	2	1		
	2	1	2	1	1		
	2	1	2				
2	2	3	4	1	2	5.3768	53.0091
	1	31	6	15	30		
	4	26	24	19	21		
	27	29	17	18	9		
	13	3	20	8	16		
	7	14	5	25	11		
	1	2	22	12	10		
	23	28	4	1	3		
	7	2	8	6	5		
	3	7	1	8	2		
	4	5	6	4	1		
	2	3	2	1	2		
	1	2	1	1	2		
	2	1	1	2	2		
	1	1	2				

Tabel 5.11 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
3	2 1 7 5 2 26 28 29 7 6 7 1 2 1 2	4 17 10 22 19 15 23 1 8 4 5 4 1 2 2	3 25 11 20 3 18 4 1 5 3 2 1 2 2 1	1 31 24 30 13 21 12 4 6 1 2 2 1 1	2 16 27 14 6 8 9 3 2 8 3 1 2 1	5.2949	50.7783
4	3 2 31 23 26 12 15 3 6 6 1 3 1 1 1	2 13 2 17 24 4 21 27 8 8 2 1 2 2 1	1 10 1 25 16 6 28 1 2 3 4 2 1 2 2	4 30 20 14 19 8 9 4 7 5 4 1 2 1	1 7 5 22 11 29 18 3 5 7 2 2 1 2	5.2162	50.1699
5	4 1 13 30 9 12 15 6 5 1 4 1 2 2 1	1 24 31 19 21 1 8 2 7 8 7 3 1 1 2	2 23 20 14 10 5 16 1 8 6 5 1 2 1 1	3 7 25 3 18 17 28 4 6 3 2 2 2 2	2 22 26 11 27 29 4 2 3 2 4 1 1 2	5.2572	50.5755

Tabel 5.12 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan 2)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
6	3 2 7 11 24 8 23 10 3 6 2 3 2 2 1	1 26 14 2 31 12 9 16 8 1 7 2 2 1 1	2 30 6 13 15 28 21 1 5 5 8 1 1 2 2	4 27 22 25 17 29 18 4 6 3 4 2 1 1	1 3 19 5 20 4 1 2 7 4 1 1 2 2	5.3150	50.7315
7	4 2 15 12 17 10 23 7 8 1 2 2 1 2 1	3 30 29 11 25 6 3 21 7 4 7 4 1 1 2	1 5 13 16 14 1 31 1 6 5 3 1 2 2 1	2 22 27 20 24 26 4 2 4 8 3 2 1 1	1 18 2 8 9 19 28 3 5 6 1 2 2 2	5.4623	50.3259
8	1 2 5 22 3 6 4 29 3 3 1 3 2 2 2	3 7 19 24 15 12 17 9 6 2 7 2 1 1 1	2 18 30 2 11 21 27 1 7 4 8 1 2 2 2	4 14 25 1 23 13 28 4 2 6 1 2 2 1	1 20 10 31 26 8 16 5 8 5 4 1 1 1	5.1719	50.6067

Tabel 5.13 Hasil Eksperimen *Simulated Annealing* pada Data Uji 3 (Lanjutan 3)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
9	1 2 27 10 26 6 7 20 6 6 8 2 1 1 1	3 12 1 21 30 14 23 24 5 3 5 4 1 2 1	2 5 4 9 19 17 13 4 8 2 1 2 2 1 2	4 29 16 28 2 8 31 1 7 7 3 1 1 2	1 18 15 3 22 11 25 3 2 4 1 2 2 2	5.2916	51.7143
10	2 2 17 13 27 29 15 2 6 8 5 1 1 2 1	4 11 24 16 14 1 25 22 5 6 2 4 2 1 1	3 30 9 8 18 7 21 4 2 3 4 1 1 2 2	1 3 10 28 31 26 5 1 7 1 3 2 1 1	1 6 12 4 23 19 20 3 8 7 2 2 2 2	5.3818	51.3555
Rata-rata						5.3207	51.0497

Setelah dilakukan eksperimen sebanyak sepuluh kali replikasi yang direkap pada Tabel 5.11, 5.12, dan 5.13 untuk algoritma *Simulated Annealing* selanjutnya dilakukan eksperimen menggunakan algoritma *Simulated Annealing-Tabu Search* untuk Data Uji 3.

Tabel 5.14 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
1	2 2 17 13 27 29 15 2 6 8 5 1 1 2 1	4 11 24 16 14 1 25 22 5 6 2 4 2 1 1	3 30 9 8 18 7 21 4 2 3 4 1 1 2 2	1 3 10 28 31 26 5 1 7 1 3 2 1 1	1 6 12 4 23 19 20 3 8 7 2 2 2 2	5.2337	58.2820
2	2 1 4 27 13 7 1 23 7 3 4 2 1 2 1	3 31 26 29 3 14 2 28 2 7 5 3 2 1 1	4 6 24 17 20 5 22 4 8 1 6 2 1 1 2	1 15 19 18 8 25 12 1 6 8 4 1 1 2	2 30 21 9 16 11 10 3 5 2 1 2 2 2	5.3120	53.0091
3	2 1 7 5 2 26 28 29 7 6 7 1 2 1 2	4 17 10 22 19 15 23 1 8 4 5 4 1 2 2	3 25 11 20 3 18 4 1 5 3 2 1 2 2 1	1 31 24 30 13 21 12 4 6 1 2 2 1 1	2 16 27 14 6 8 9 3 2 8 3 1 2 1	5.3120	55.9108

Tabel 5.15 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 1)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
4	1 2 14 20 19 29 15 18 8 6 3 1 2 2 1	3 11 12 5 10 17 16 27 6 2 7 3 1 1 2	2 7 3 8 1 9 23 1 5 5 1 1 2 1	4 25 31 6 22 21 28 3 4 8 4 2 1 2	1 30 26 2 24 13 4 7 2 4 2 1 2 2	5.1640	59.6860
5	2 2 17 13 27 29 15 2 6 8 5 1 1 2 1	4 11 24 16 14 1 25 22 5 6 2 4 2 1 1	3 30 9 8 18 7 21 4 2 3 4 1 1 2 2	1 3 10 28 31 26 5 1 7 1 3 2 1 1	1 6 12 4 23 19 20 3 8 7 2 2 2 2	5.2453	50.5755
6	1 2 14 20 19 29 15 18 8 6 3 1 2 2 1	3 11 12 5 10 17 16 27 6 2 7 3 1 1 2	2 7 3 8 1 9 23 1 5 5 1 1 2 1	4 25 31 6 22 21 28 3 4 8 4 2 1 2	1 30 26 2 24 13 4 7 2 4 2 1 2 2	5.2279	58.3444

Tabel 5.16 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 2)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
7	2 2 3 26 12 8 9 25 7 2 8 2 2 2 2	3 31 6 16 29 13 23 21 8 1 6 3 1 1 2	4 5 7 11 2 15 17 1 2 4 3 2 2 1 1	1 30 28 1 20 18 19 4 3 5 4 1 2 2	1 14 22 10 24 27 4 6 5 7 1 1 1 1	5.2799	59.2100
8	4 1 22 2 6 7 1 9 3 3 1 1 2 1 1	1 26 20 17 19 27 21 16 2 7 6 2 1 2 1	2 5 14 30 18 29 4 1 8 5 8 1 2 1 2	3 23 25 24 3 28 11 4 6 2 3 2 2 2	2 10 8 31 13 15 12 5 7 4 4 1 1 2	5.2286	58.3676
9	3 1 24 31 2 9 10 12 5 8 1 4 1 1 2	1 7 20 22 14 4 8 28 7 5 6 1 1 2 2	2 26 17 19 18 1 29 1 8 2 3 2 2 2 1	4 23 16 25 3 15 13 4 2 4 2 1 2 1	2 5 21 11 30 6 27 3 6 7 3 2 1 1	5.1973	58.9996

Tabel 5.17 Hasil Eksperimen *Simulated Annealing-Tabu Search* pada Data Uji 3 (Lanjutan 3)

c=0.6, n=10, maxiter=2000, To=600							
No Replikasi	Solusi Urutan					Total Biaya (1e+008)	Waktu Komputasi (detik)
10	3	2	1	4	1	5.2106	57.9232
	2	27	24	30	11		
	2	12	14	29	25		
	6	10	19	22	5		
	7	23	26	13	20		
	15	18	31	21	8		
	9	3	4	17	1		
	16	28	1	4	2		
	5	7	3	6	8		
	7	8	5	6	4		
	1	2	3	3	2		
	1	4	2	1	1		
	2	2	1	1	2		
	1	2	2	1	1		
	2	2	1				
Rata-rata						5.2411	57.0308

Setelah mendapatkan solusi untuk permasalahan ini dengan menggunakan *Simulated Annealing* dan *Simulated Annealing-Tabu Search,* maka dilakukan perhitungan GAP atau selisih antara hasil yang didapat dari *Simulated Annealing* dan *Simulated Annealing-Tabu Search* dengan yang didapat dari perhitungan. Rumus yang digunakan untuk melakukan perhitungan GAP dari total biaya CLSC adalah sebagai berikut :

$$GAP = \frac{(Hasil\ Perhitungan\ SA\_TS - Hasil\ Perhitungan\ SA)}{Hasil\ perhitungan\ SA\_TS}\ X\ 100\%$$

Rumus GAP dilakukan untuk setiap replikasi pada eksperimen kemudian direkap pada Tabel 5.16 yang menyatakan persentase GAP antara *Simulated Annealing* dan *Simulated Annealing-Tabu Search*

Tabel 5.18 Perbandingan Nilai Perubahan Total Biaya pada Data Uji 3

Total Biaya			
No Replikasi	SA-TS (1e+004)	SA (1e+008)	GAP
1	5.2337	5.4395	-3.9322%
2	5.3120	5.3768	-1.2199%
3	5.3120	5.2949	-1.3140%
4	5.1640	5.2162	-0.1530%
5	5.2453	5.2572	-0.2269%
6	5.2279	5.3150	-1.6661%
7	5.2799	5.4623	-3.4546%
8	5.2286	5.1719	-1.2049%
9	5.1973	5.2916	-0.3637%
10	5.2106	5.3818	-1.6179%
Rata-rata			-1.515%

Selain menghitung GAP antar nilai total biaya masing-masing algoritma dapat dilakukan perhitungan sesilsih untuk waktu komputasi. Berikut ini adalah rumus yang digunakan untuk menghitung GAP waktu komputasi :

$$Perbandingan = \frac{waktu\ komputasi\ SA\_TS}{waktu\ komputasi\ SA}$$

Tabel 5.19 Perbandingan Waktu Komputasi pada Data Uji 3

Waktu Komputasi			
No Replikasi	SA-TS (detik)	SA (detik)	Perbandingan
1	58.2820	51.2307	0.8790
2	53.0091	53.0091	1.0000
3	55.9108	50.7783	0.9185
4	59.6860	50.1699	0.8479
5	50.5755	50.5755	1.0000
6	58.3444	50.7315	0.8695
7	59.2100	50.3259	0.8500
8	58.3676	50.6067	0.8860
9	58.9996	51.7143	0.8503
10	57.9232	51.3555	0.8766
Rata-rata			0.8978

5.1 Analisis Hasil Eksperimen

Pada bagian ini akan diberikan analisis terkait hasil yang didapatkan dalam eksperimen yang dilakukan dengan menggunakan algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search* untuk menyelesaikan permasalahan *Closed Loop Supply Chain*.

5.1.1 Analisis Uji Parameter Algoritma *Simulated Annealing* dan *Simulated Annealing-Tabu Search*

Tahapan pertama yang dilakukan sebelum melakukan eksperimen algoritma SA untuk menyelesaikan permasalahan CLSC adalah menentukan nilai parameter yang tepat untuk mendapatkan solusi yang terbaik dengan waktu komputasi yang cepat. Dalam uji parameter, nilai parameter yang diubah adalah faktor pereduksi temperature (c) dan temperature awal (t). Pada uji parameter faktor pereduksi temperatur, skenario pereduksi temperatur yang digunakan adalah 0.3, 0.6 dan 0.9, skenario ini diuji dengan menggunakan jumlah siklus=10 dan *stopping criteria* yang digunakan adalah iterasi maksimum sebesar 2000 dengan banyak replikasi adalah sejumlah tiga kali replikasi untuk masing masing factor pereduksi temperatur.

Pada faktor pereduksi temperatur sebesar 0.9, penurunan temperatur dilakukan sedikit demi sedikit yang menyebabkan temperature turun secara perlahan, semakin lamban turunnya temperature maka semakin lama waktu untuk setiap n kali siklus. Nilai temperatur yang masih terlalu besar pada tiap iterasi akan mengakibatkan peluang untuk menerima solusi yang lebih buruk lebih besar sehingga akan membutuhkan waktu yang cukup lama untuk mencapai konvergensi. Sementara itu pada faktor pereduksi sebesar 0.6 akan mengakibatkan penurunan temperatur yang lebih cepat sehingga akan memungkinkan solusi mencapai konvergensi dengan lebih cepat namun akan mudah untuk terjebak pada solusi local optimal. Hal inilah yang menjadi alasan mengapa pada faktor pereduksi sebesar 0.3 rata-rata solusi yang dihasilkan lebih buruk dibandingkan pada skenario faktor pereduksi bernilai sama dengan 0.6 dan 0.9. Nilai faktor pereduksi yang paling baik pada kasus ini adalah sebesar 0.6 karena menghasilkan rata-rata solusi yang lebih baik dibandingkan dengan faktor pereduksi lainnya.

Dalam skenario perubahan faktor pereduksi temperatur, lama waktu komputasi tidak dapat digunakan sebagai pengambilan keputusan skenario yang baik karena *stopping criteria* yang digunakan adalah jumlah iterasi maksimum. Waktu komputasi pada dasarnya bergantung pada jenis *stopping criteria* yang digunakan.

Uji parameter yang kedua adalah temperature awal (t). Besarnya temperature awal mempengaruhi waktu komputasi dan jumlah pencarian, semakin besar temperature maka semakin lama waktu untuk setiap n kali siklus. Nilai temperatur yang masih terlalu besar pada tiap iterasi akan mengakibatkan peluang untuk menerima solusi yang lebih buruk lebih besar sehingga akan membutuhkan waktu yang cukup lama untuk mencapai konvergensi.

5.1.2 Analisis Hasil SA dan SA-TS untuk Data Uji 2

Berdasarkan hasil replikasi eksperimen yang dilakukan untuk menyelesaikan permasalahan CLSC pada Data Uji 2 untuk algoritma SA nilai terbaik yang didapatkan adalah pada replikasi pertama dan ketujuh dengan nilai total biaya CLSC sebesar 1.2048*1e+4. Untuk Data Uji 2 setelah dilakukan replikasi maka didapatkan rata-rata untuk total biaya CLSC sebesar 1.2082*1e+4 dengan rata-rata waktu komputasi sebesar 10,5582.

Untuk algoritma SA-TS nilai terbaik yang didapatkan adalah pada replikasi pertama, ke-2, ke-3, ke-6, ke-7, ke-8, dan ke-10 dengan nilai total biaya CLSC sebesar 1.2048*1e+4. Untuk Data Uji 2 setelah dilakukan replikasi maka didapatkan rata-rata untuk total biaya CLSC sebesar 1.2058*1e+4 dengan rata-rata waktu komputasi sebesar 16,8512.

Setelah dilakukan ekperimen dengan replikasi sebanyak sepuluh kali tersebut maka dapat dibandingkan total biaya CLSC antar algoritma dengan rekap data pada Tabel 5.8, selain total biaya, dibandingkan pula waktu komputasi untuk kedua algoritma dengan rekap data pada Tabel 5.9. Performansi algoritma SA-TS dapat dilihat memeliki kualitas hasil yang lebih baik dibandingkan menggunakan algoritma SA biasa namun waktu komputasi pada SA-TS lebih lama dikarenakan adanya checking pada tabu list terkait solusi baru agar solusi baru yang sudah pernah dilakukan perhitungan dan buruk dapat dihindari. Hal ini ditunjukkan agar tidak perlu melakukan perhitungan, namun membandingkan solusi baru dan tabu

list ternyata memberikan dampak yang lebih lama. Namun keakuratan dalam mencapai minimasi biaya CLSC dapat lebih terpercaya.

5.1.3 Analisis Hasil SA dan SA-TS untuk Data Uji 3

Pada eksperimen yang dilakukan pada Data Uji 3 yang terdiri dari 4 produk, 7 periode, 8 *warehouse* dan 31 distributor digunakan parameter yang sama degan Data Uji 2. nilai terbaik yang didapatkan adalah pada replikasi ke-9 dengan nilai total biaya CLSC sebesar 5.1719*1e+8. Untuk Data Uji 3 setelah dilakukan replikasi maka didapatkan rata-rata untuk total biaya CLSC sebesar 5.3207*1e+8 dengan rata-rata waktu komputasi sebesar 51.0497.

Untuk algoritma SA-TS nilai terbaik yang didapatkan adalah pada replikasi dengan nilai total biaya CLSC sebesar 5.1604*1e+8. Untuk Data Uji 3 setelah dilakukan replikasi maka didapatkan rata-rata untuk total biaya CLSC sebesar 5.2411*1e+4 dengan rata-rata waktu komputasi sebesar 57,0308..

Setelah dilakukan ekperimen dengan replikasi sebanyak sepuluh kali tersebut maka dapat dibandingkan total biaya CLSC antar algoritma dengan rekap data pada Tabel 5.8, selain total biaya, dibandingkan pula waktu komputasi untuk kedua algoritma dengan rekap data pada Tabel 5.9. Performansi algoritma SA-TS dapat dilihat memeliki kualitas hasil yang lebih baik dibandingkan menggunakan algoritma SA biasa namun waktu komputasi pada SA-TS lebih lama dikarenakan adanya checking pada tabu list terkait solusi baru agar solusi baru yang sudah pernah dilakukan perhitungan dan buruk dapat dihindari. Hal ini ditunjukkan agar tidak perlu melakukan perhitungan, namun membandingkan solusi baru dan tabu list ternyata memberikan dampak yang lebih lama. Namun dalam permasalahan besar pada Data Uji 3 ini tidak didapatkan hasil yang konvergensi. Setiap replikasi selalu menghasilkan total biaya yang berbeda hal ini dipengaruhi oleh *stopping criteria* yaitu maksimum iterasi, apabila maksimum itersi diperbesar, kemungkinan mencapai hasil yang konvergen semakin banyak.

BAB 6
KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan hasil penelitian dan saran-saran yang dapat digunakan untuk penelitian berikutnya

6.1 Kesimpulan

Berdasarkan hasil eksperimen dan analisis yang telah dilakukan pada bab sebelumnya, berikut ini adalah kesimpulan yang dapat diambil dalam penelitian ini :

1. Dalam penelitian ini dihasilkan model *Closed Loop Supply Chain* yang dengan tujuan minimasi total biaya CLSC yang terdiri dari biaya inventory dan biaya distribusi dalam model menggunakan node yang terdiri dari manufacturing plants, warehouse, dan distributor, serta memiliki fariansi produk.
2. Algoritma SA yang dikembangkan dapat menyelesaikan setiap permasalahan pada Data Uji. Begitu pula untuk algoritma SA-TS yang dikembangkan dapat menyelesaikan setiap permasalahan pada Data Uji. Pada data skala kecil kualitas solusi yang dihasilkan SA dan SA-TS memiliki solusi yang mendekati optimalnya, sedangkan pada kasus besar waktu komputasi yang dibutuhkan SA lebih cepat jika dibandingkan dengan SA-TS namun untuk hasil minimasi total biaya yang dihasilkan lebih baik ketika menggunakan algoritma SA-TS disbanding dengan SA.

6.2 Saran

Berikut ini adalah beberapa saran yang dapat diberikan dalam penelitian ini untuk penelitian selanjutnya adalah :

1. Menerapkan teknik penyelesaian CLSC pada perusahaan secara langsung.

2. Menggunakan algoritma metaheuristik lain yang termasuk dalam kelompok *population based* seperti PSO, ACO dan beberapa teknik lainnya untuk mengetahui perbandingan performansi algoritma SA dengan algoritma lain pada kasus CLSC ini.
3. Dalam pengembangan model CLSC selanjutnya, dapat dipertimbangkan kapasitas produksi yang statis, memperhitungkan *backorder,* dan menambahkan bagaian pembuangan dan bagian reuse serta biaya yang terkait.

DAFTAR PUSTAKA

Beamon, B.M. 1998. Supply chain design and analysis : Models and methods. *International Journal of Production Economics, 55*, 281-294.

Bernon, M., Cullen, J., & Rowat, C. 2004. *The Efficiency of Reverse Logistics*. Cranfield University, UK.

Kannan, G., Sasikumar, P., & Devita, K. 2009. A Genetic Algorithm Approach for Solving A Closed Loop Supply Chain Model : A Case of Battery Recycling. *Applied Mathematical Modelling*, 655-670.

Pujawan, I.N. 2005. *Supply Chain Management*. Suarabaya: Penerbit Guna Widya.

Rivera, R. & Ertel, J. 2008 Reverse logistics network design for the collection of End-of Life Vehicles in Mexico. *European Journal of Operational Research 196 :* 930–939

Rogers, D.S. & Tibben-Lembke, R., 1999. *Going Backwards: Reverse Logistics Trends and Practices,* Reverse Logistics Executive Council, University of Nevada, Reno Center for Logistics Management.

Santosa, B. & Willy, P., 2011. *Metoda Metaheuristik Konsep dan Implementasi*. 1st ed. Surabaya: Prima Printing.

Subramanian P., Ramkumar N., Narendran T.T., & Ganesh K. 2012. A Technical Note on 'Analysis of Closed Loop Supply Chain Using Genetic Algorithm And Particle Swarm Optimization. *International Journal of Production Research Vol. 50, No. 2,* 593–602.

Schultmann, F., Zumkeller, M., & Rentz, O. 2006. Modeling reverse logistic tasks within closedloop supply chains: An example from the automotive industry. *European Journal of Operational Research 171:* 1033–1050.

Wang, X., Golden, B.L., & Wasil, E.A. 2008. Using a Genetic Algorithm to Solve the Generalized Orienteering Problem. *The Vehicle Routing Problem: Latest Advances and New Challenges***,** 263-274.

LAMPIRAN

Kode Program Algoritma *Simulated Annealing* untuk Alokasi pada *Closed Loop Supply Chain*

```
function
[sol_SA,W_New,QWD,REM,QDW,INVW,total_SA,waktu]=SA__CLSC(nd,nw,np,n
b,nt,nr,prod,demand,returw,returm,gudang,JAR_WD,JAR_DW,JAR_MW,JAR_
WM,ratedist,ratebalikx,ratebaliky,ICW,tempawal,c,maxiter)
start = cputime;

temperatur=tempawal;
iterasi=1;
siklus=0;
Biaya=0;

%r = index retailer
%w = index warehouse
%p = index produk
%b = index tipe barang
%t = index periode

CP=prod;
CW=gudang;

DEM=zeros(np,nd,nt);

REW=zeros(np,nd,nt);

REM =zeros(np,nw,nt);

TREM=zeros(np,nt);

INVBM = zeros(np,nt);

TCMW = zeros(nw,nt);
TCWM = zeros(nw,nt);
TCWD  = zeros(nd,nw,nt);
TCDW  = zeros(nd,nw,nt);

INVW = zeros(np,nw,nt+1);

TOT=zeros(1,nt);

QWD=zeros(nd,nw,nt);
QDW=zeros(nd,nw,nt);
W_New=zeros(nw,np,nt);
W_Ret=zeros(nw,np,nt);

i=randperm(np);
j=randperm(nb);
k=randperm(nd);
```

```
l=randperm(nw);
l2=randperm(nw);
s=randperm(np);
rn=randperm(nr);

sol1=[i j k l l2 s rn];
t=1;

while t <= nt

d = 1;
w = 1;
p = 1;
b = 1;
r = 1;

REW(:,:,t+1)=floor(DEM(:,:,t).*ratebalikx);
tcek=t  ;
INVWS=INVW(:,:,t);

REM(:,:,t+1)=floor(W_Ret(:,:,t)'.*ratebaliky);

    w=1;
    p=1;
    r=1;
if sum(CP)-sum(sum((W_New(:,:,t))))+sum(sum(INVW(:,:,t)))> 0
    sisa=CP-sum(W_New(:,:,t));
else
    sisa =[0 0 0 0];
end

INVW(:,:,t+1)=INVWS;

CW=gudang-sum(INVW(:,:,t+1));

TCMW=sum(W_New(:,:,t),2).*(ratedist*JAR_MW); % TCMW = total produk
dari manufaktur ke warehouse per periode
transpose=REM(:,:,t)'; % transpose=
TCWM=sum(transpose,2).*(ratedist*JAR_WM); % TCWM = total botol
dari warehouse ke manufaktur per periode
TCWD=(QWD(:,:,t)).*(ratedist*JAR_WD); % TCWD = total produk dari
warehouse ke retailer per periode
TCDW=(QDW(:,:,t)).*(ratedist*JAR_DW); % TCDW = total botol dari
retailer ke warehouse per periode
TCIW = INVW(:,:,t+1).*ICW; % TCIW = total inventory (produk masuk-
produk keluar) di warehouse per periode
totTCMW = sum(TCMW); % total biaya distribusi produk dari
manufaktur ke warehouse per periode
totTCWM = sum(TCWM); % total biaya distribusi botol dari warehouse
ke manufaktur per periode
totTCWD = sum(sum(TCWD)); % total biaya distribusi produk dari
warehouse ke retailer per periode
totTCDW = sum(sum(TCDW)); % total biaya distribusi botol dari
retailer ke warehouse per periode
totTCIW = sum(sum(TCIW)); % total biaya inventory setiap periode

TOT(:,t)=totTCMW+totTCWM+totTCWD+totTCDW+totTCIW; %total biaya
closed loop supply chain
```

```
t=t+1;
end

total=sum(TOT);
total1=total;

while iterasi<maxiter

    i0=i;
    j0=j;
    k0=k;
    l0=l;
    l20=l2;
    s0=s;
    rn0=rn;

    batas1=sort(ceil(np*rand(1,2)));
    I=batas1(1);
    J=batas1(2);

    r1=rand;
    if r1<0.333
        i(:,I:J)=fliplr(i0(:,I:J));
    elseif r1<0.667 && r1>0.333
        i(:,[I J])=i0(:,[J I]);
    else
        i(:,I:J)=i0(:,[I+1:J I]);
    end

    batas2=sort(ceil(nb*rand(1,2)));
    K=batas2(1);
    L=batas2(2);

    r2=rand;
    if r2<0.333
        j(:,K:L)=fliplr(j0(:,K:L));
    elseif r2<0.667 && r2>0.333
        j(:,[K L])=j0(:,[L K]);
    else
        j(:,K:L)=j0(:,[K+1:L K]);
    end

    batas3=sort(ceil(nd*rand(1,2)));
    M=batas3(1);
    N=batas3(2);

    r3=rand;
    if r3<0.333
        k(:,M:N)=fliplr(k0(:,M:N));
    elseif r3<0.667 && r3>0.333
        k(:,[M N])=k0(:,[N M]);
    else
        k(:,M:N)=k0(:,[M+1:N M]);
    end

    batas4=sort(ceil(nw*rand(1,2)));
    O=batas4(1);
    P=batas4(2);
```

```
    r4=rand;
    if r4<0.333
        l(:,O:P)=fliplr(l0(:,O:P));
    elseif r4<0.667 && r4>0.333
        l(:,[O P])=l0(:,[P O]);
    else
        l(:,O:P)=l0(:,[O+1:P O]);
    end

    batas5=sort(ceil(nb*rand(1,2)));
    Q=batas5(1);
    R=batas5(2);

    r5=rand;
    if r5<0.333
        s(:,Q:R)=fliplr(s0(:,Q:R));
    elseif r5<0.667 && r5>0.333
        s(:,[Q R])=s0(:,[R Q]);
    else
        s(:,Q:R)=s0(:,[Q+1:R Q]);
    end

    batas6=sort(ceil(nr*rand(1,2)));
    S=batas6(1);
    T=batas6(2);

    r6=rand;
    if r6<0.333
        rn1(:,S:T)=fliplr(rn10(:,S:T));
    elseif r6<0.667 && r6>0.333
        rn1(:,[S T])=rn10(:,[T S]);
    else
        rn1(:,S:T)=rn10(:,[S+1:T S]);
    end

    batas14=sort(ceil(nw*rand(1,2)));
    YA=batas14(1);
    ZA=batas14(2);

    r14=rand;
    if r14<0.333
        l2(:,YA:ZA)=fliplr(l20(:,YA:ZA));
    elseif r14<0.667 && r14>0.333
        l2(:,[YA ZA])=l20(:,[ZA YA]);
    else
        l2(:,YA:ZA)=l20(:,[YA+1:ZA YA]);
    end

sol2=[i j k l l2 s rn] ;

CP=prod;
CW=gudang;

DEM=zeros(np,nd,nt);

REW=zeros(np,nd,nt);

REM =zeros(np,nw,nt);
```

```
TREM=zeros(np,nt);

INVBM = zeros(np,nt);

TCMW = zeros(nw,nt);
TCWM = zeros(nw,nt);
TCWD  = zeros(nd,nw,nt);
TCDW  = zeros(nd,nw,nt);

INVW = zeros(np,nw,nt+1);

TOT=zeros(1,nt);

QWD=zeros(nd,nw,nt);
QDW=zeros(nd,nw,nt);
W_New=zeros(nw,np,nt);
W_Ret=zeros(nw,np,nt);

t=1;

while t <= nt

d = 1;
w = 1;
p = 1;
b = 1;
r = 1;

REW(:,:,t+1)=floor(DEM(:,:,t).*ratebalikx);
tcek=t  ;
INVWS=INVW(:,:,t);

REM(:,:,t+1)=floor(W_Ret(:,:,t)'.*ratebaliky);

    w=1;
    p=1;
    r=1;
if sum(CP)-sum(sum((W_New(:,:,t))))+sum(sum(INVW(:,:,t)))> 0
    sisa=CP-sum(W_New(:,:,t));
else
    sisa =[0 0 0 0];
end

INVW(:,:,t+1)=INVWS;

CW=gudang-sum(INVW(:,:,t+1));

TCMW=sum(W_New(:,:,t),2).*(ratedist*JAR_MW);
transpose=REM(:,:,t)';
TCWM=sum(transpose,2).*(ratedist*JAR_WM);
TCWD=(QWD(:,:,t)).*(ratedist*JAR_WD);
TCDW=(QDW(:,:,t)).*(ratedist*JAR_DW);
TCIW = INVW(:,:,t+1).*ICW;
totTCMW = sum(TCMW); totTCWM = sum(TCWM);
totTCWD = sum(sum(TCWD));
totTCDW = sum(sum(TCDW));
totTCIW = sum(sum(TCIW));
```

```
TOT(:,t)=totTCMW+totTCWM+totTCWD+totTCDW+totTCIW;
t=t+1;
end

total=sum(TOT);
total2=total;

    band=abs(total2-total1);

    if total2<total1
        i;
        j;
        k;
        l;
        s;
        rn;

        sol=[i j k l l2 s rn1 rn2 rn3 rn4 rn5 rn6 rn7 rn8];
        if siklus>=10
            temperatur=c*temperatur;
            siklus=0;
        end
        total1=total2;
     else
        if rand(1)<exp(-band/temperatur) %/temperatur
            i;
            j;
            k;
            l;
            s;
            rn;

            sol=[i j k l l2 s rn];
            total1=total2;
         end
        i=i0;
        j=j0;
        k=k0;
        l=l0;
        s=s0;
        rn1=rn0;
        sol=[i j k l l2 s rn];
    end

    siklus=siklus+1;
    iterasi=iterasi+1;
    Biaya=[Biaya total1];
plot(Biaya);
total=sum(TOT);
finish = cputime;
waktu=finish-start;
sol_SA=sol;
total_SA = total;
```

Kode Program Algoritma *Simulated Annealing-Tabu Search* untuk Alokasi pada *Closed Loop Supply Chain*

```
function
[sol_SA_TS,W_New,QWD,W_Ret,QDW,INVW,total_SA_TS,waktu]=SA_TS__CLSC
(nd,nw,np,nb,nt,nr,prod,demand,returw,returm,gudang,JAR_WD,JAR_DW,
JAR_MW,JAR_WM,ratedist,ratebalikx,ratebaliky,ICW,tempawal,c,maxite
r,n)

op=1;
start = cputime;
temperatur=tempawal;
iterasi=1;
siklus=0;
Biaya=0;
tabu_list=zeros(maxiter,73);

%r = index retailer
%w = index warehouse
%p = index produk
%b = index tipe barang
%t = index periode

CP=prod;
CW=gudang;
DEM=zeros(np,nd,nt);
REW=zeros(np,nd,nt);
REM =zeros(np,nw,nt);
TREM=zeros(np,nt);
TCMW = zeros(nw,nt);
TCWM = zeros(nw,nt);
TCWD  = zeros(nd,nw,nt);
TCDW  = zeros(nd,nw,nt);
INVW = zeros(np,nw,nt+1);
TOT=zeros(1,nt);

QWD=zeros(nd,nw,nt);
QDW=zeros(nd,nw,nt);
W_New=zeros(nw,np,nt);
W_Ret=zeros(nw,np,nt);

i=randperm(np);
j=randperm(nb);
k=randperm(nd);
l=randperm(nw);
l2=randperm(nw);
s=randperm(np);
rn=randperm(nr);
sol1=[i j k l l2 s rn];
sol=zeros(1,73);
t=1;

while t <= nt
d = 1;
```

```
w = 1;
p = 1;
b = 1;
r = 1;
REW(:,:,t+1)=floor(DEM(:,:,t).*ratebalikx);
tcek=t  ;
INVWS=INVW(:,:,t);
REM(:,:,t+1)=floor(W_Ret(:,:,t)'.*ratebaliky);
    w=1;
    p=1;
    r=1;
if sum(CP)-sum(sum((W_New(:,:,t))))+sum(sum(INVW(:,:,t)))> 0
    sisa=CP-sum(W_New(:,:,t));
else
    sisa =[0 0 0 0];
end
INVW(:,:,t+1)=INVWS;
CW=gudang-sum(INVW(:,:,t+1));

TCMW=sum(W_New(:,:,t),2).*(ratedist*JAR_MW); % TCMW = total produk
dari manufaktur ke warehouse per periode
transpose=REM(:,:,t)'; % transpose=
TCWM=sum(transpose,2).*(ratedist*JAR_WM); % TCWM = total botol
dari warehouse ke manufaktur per periode
TCWD=(QWD(:,:,t)).*(ratedist*JAR_WD); % TCWD = total produk dari
warehouse ke retailer per periode
TCDW=(QDW(:,:,t)).*(ratedist*JAR_DW); % TCDW = total botol dari
retailer ke warehouse per periode
TCIW = INVW(:,:,t+1).*ICW; % TCIW = total inventory (produk masuk-
produk keluar) di warehouse per periode
totTCMW = sum(TCMW); % total biaya distribusi produk dari
manufaktur ke warehouse per periode
totTCWM = sum(TCWM); % total biaya distribusi botol dari warehouse
ke manufaktur per periode
totTCWD = sum(sum(TCWD)); % total biaya distribusi produk dari
warehouse ke retailer per periode
totTCDW = sum(sum(TCDW)); % total biaya distribusi botol dari
retailer ke warehouse per periode
totTCIW = sum(sum(TCIW)); % total biaya inventory setiap periode

TOT(:,t)=totTCMW+totTCWM+totTCWD+totTCDW+totTCIW; %total biaya
closed loop supply chain

t=t+1;
end

total=sum(TOT);
total1=total;
sol1=[i j k l l2 s rn];

while iterasi<maxiter
sama=1;
    while sama==1
    i0=i;
    j0=j;
    k0=k;
    l0=l;
```

```
l20=l2;
s0=s;
rn0=rn;

batas1=sort(ceil(np*rand(1,2)));
I=batas1(1);
J=batas1(2);

r1=rand;
if r1<0.333
    i(:,I:J)=fliplr(i0(:,I:J));
elseif r1<0.667 && r1>0.333
    i(:,[I J])=i0(:,[J I]);
else
    i(:,I:J)=i0(:,[I+1:J I]);
end

batas2=sort(ceil(nb*rand(1,2)));
K=batas2(1);
L=batas2(2);

r2=rand;
if r2<0.333
    j(:,K:L)=fliplr(j0(:,K:L));
elseif r2<0.667 && r2>0.333
    j(:,[K L])=j0(:,[L K]);
else
    j(:,K:L)=j0(:,[K+1:L K]);
end

batas3=sort(ceil(nd*rand(1,2)));
M=batas3(1);
N=batas3(2);

r3=rand;
if r3<0.333
    k(:,M:N)=fliplr(k0(:,M:N));
elseif r3<0.667 && r3>0.333
    k(:,[M N])=k0(:,[N M]);
else
    k(:,M:N)=k0(:,[M+1:N M]);
end

batas4=sort(ceil(nw*rand(1,2)));
O=batas4(1);
P=batas4(2);

r4=rand;
if r4<0.333
    l(:,O:P)=fliplr(l0(:,O:P));
elseif r4<0.667 && r4>0.333
    l(:,[O P])=l0(:,[P O]);
else
    l(:,O:P)=l0(:,[O+1:P O]);
end
```

```
    batas5=sort(ceil(nb*rand(1,2)));
    Q=batas5(1);
    R=batas5(2);

    r5=rand;
    if r5<0.333
        s(:,Q:R)=fliplr(s0(:,Q:R));
    elseif r5<0.667 && r5>0.333
        s(:,[Q R])=s0(:,[R Q]);
    else
        s(:,Q:R)=s0(:,[Q+1:R Q]);
    end

    batas6=sort(ceil(nr*rand(1,2)));
    S=batas6(1);
    T=batas6(2);

    r6=rand;
    if r6<0.333
        rn1(:,S:T)=fliplr(rn10(:,S:T));
    elseif r6<0.667 && r6>0.333
        rn1(:,[S T])=rn10(:,[T S]);
    else
        rn1(:,S:T)=rn10(:,[S+1:T S]);
    end

    batas7=sort(ceil(nr*rand(1,2)));
    U=batas7(1);
    V=batas7(2);

    r14=rand;
    if r14<0.333
        l2(:,YA:ZA)=fliplr(l20(:,YA:ZA));
    elseif r14<0.667 && r14>0.333
        l2(:,[YA ZA])=l20(:,[ZA YA]);
    else
        l2(:,YA:ZA)=l20(:,[YA+1:ZA YA]);
    end

    sol=[i j k l l2 s rn];

        for tl=1:op;
            cari=find(sol==tabu_list(tl,:));
            if length(cari)==73
                iterasi=iterasi+1;
                sama = 1;
                break
            else
                sama=0;
            end
        end
    end
CP=prod;
CW=gudang;
DEM=zeros(np,nd,nt);
REW=zeros(np,nd,nt);
```

```
REM =zeros(np,nw,nt);
TREM=zeros(np,nt);
TCMW = zeros(nw,nt);
TCWM = zeros(nw,nt);
TCWD  = zeros(nd,nw,nt);
TCDW  = zeros(nd,nw,nt);
INVW = zeros(np,nw,nt+1);
TOT=zeros(1,nt);

QWD=zeros(nd,nw,nt);
QDW=zeros(nd,nw,nt);
W_New=zeros(nw,np,nt);
W_Ret=zeros(nw,np,nt);

t=1;
while t <= nt
d = 1;
w = 1;
p = 1;
b = 1;
r = 1;
REW(:,:,t+1)=floor(DEM(:,:,t).*ratebalikx);
tcek=t  ;
INVWS=INVW(:,:,t);
REM(:,:,t+1)=floor(W_Ret(:,:,t)'.*ratebaliky);
    w=1;
    p=1;
    r=1;
if sum(CP)-sum(sum((W_New(:,:,t))))+sum(sum(INVW(:,:,t)))> 0
    sisa=CP-sum(W_New(:,:,t));
else
    sisa =[0 0 0 0];
end
INVW(:,:,t+1)=INVWS;
CW=gudang-sum(INVW(:,:,t+1));

TCMW=sum(W_New(:,:,t),2).*(ratedist*JAR_MW);
transpose=REM(:,:,t)';
TCWM=sum(transpose,2).*(ratedist*JAR_WM);
TCWD=(QWD(:,:,t)).*(ratedist*JAR_WD);
TCDW=(QDW(:,:,t)).*(ratedist*JAR_DW);
TCIW = INVW(:,:,t+1).*ICW;
totTCMW = sum(TCMW);
totTCWM = sum(TCWM);
totTCWD = sum(sum(TCWD));
totTCDW = sum(sum(TCDW));
totTCIW = sum(sum(TCIW));

TOT(:,t)=totTCMW+totTCWM+totTCWD+totTCDW+totTCIW;

t=t+1;
end

total=sum(TOT);

total2=total;
```

```
    band=abs(total2-total1);

    if total2<total1
        sol=sol1;
        tabu_list(op,:)=sol;
        op=op+1;
        i;
        j;
        k;
        l;
        s;
        rn;
        sol1=[i j k l l2 s rn];
        if siklus>=10
            temperatur=c*temperatur;
            siklus=0;
        end
        total1=total2;

    else
        if rand(1)<exp(-band/temperatur) %/temperatur
            sol=[i j k l l2 s rn];
            tabu_list(op,:)=sol;
            op=op+1;
            i;
            j;
            k;
            l;
            s;
            rn;
            sol1=[i j k l l2 s rn];
            total1=total2;
        else
            sol=[i j k l l2 s rn];
            tabu_list(op,:)=sol;
            op=op+1;
            i=i0;
            j=j0;
            k=k0;
            l=l0;
            s=s0;
            rn=rn0;
        end
    end
    siklus=siklus+1;
    iterasi=iterasi+1;
    Biaya=[Biaya total1];
    plot(Biaya);
end

finish = cputime;
waktu=finish-start;
sol_SA_TS=sol;
total_SA_TS = total1;
```

BIOGRAFI PENULIS

Penulis memiliki nama lengkap Risal Arsyad Muhaddad. Penulis dilahirkan di kota Surabaya pada tanggal 27 September tahun 1991 dari pasangan Bapak Surama dan Ibu Mustlah. Penulis merupakan anak pertama dari dua bersaudara. Penulis menempu pendidikan di TK ABA III Tuban (1996-1998), SDN Latsari 1 Tuban (1998-2004), SMPN 1 Tuban (2004-2007) dan SMAN 1 Tuban (2007-2010). Setelah lulus dari SMA penulis memutuskan untuk melanjutkan kuliah di jenjang, dan Alhamdulillah diterima di Jurusan Teknik Industri ITS. Penulis memiliki beberapa hobi antara lain menggambar, membaca, menonton film dan anime serta jogging, namun hobi yang masih berjalan hanya menonton film dan anime serta jogging. Film yang paling disukai oleh penulis adalah anime tentang sekolah, dan Studio Gibli, selain itu penulis menggemari serial Pretty Little Liar, Sherlock, Devious Maid, dan Game of Thrones. Pada tahun kedua perkuliahan, penulis mengikuti Organisasi Mahasiswa di BEM FTI ITS sebagai staff departemen Kepofesian dan Kesejahteraan Mahasiswa. Selama perkuliahan di Jurusan Teknik Indutri ITS penulis memiliki ketertarikan pada bidang perkuliahan laboratorium KOI yang berhubungan dengan optimasi seperti modeling dan metaheuristik, selain itu penulis tertarik pada bidang Logistik dan *Supply Chain Management*, manajemen distribusi, serta keuangan. Penulis dapat dihubungi melalui email risal.arsyad@gmail.com