Optimasi *Multiple Can-Order Level* pada *Can-Order Policy* Menggunakan Algoritma *Simulated Annealing* (Studi Kasus: *Spare Part Inventory* PT X)

Faldy Maulana Yuantoro dan Budi Santosa
Departemen Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: budi_s@ie.its.ac.id

***Abstrak*—Persediaan merupakan stok dari barang-barang yang disimpan dan akan digunakan di masa yang akan datang. Salah satu permasalahan yang terjadi adalah apabila permintaan pada suatu *item* tergolong permintaan yang *intermittent.* Kebijakan persediaan pada jenis permintaan tersebut sulit untuk ditentukan karena pola pemintaan yang tidak pasti sehingga tingkat kesalahan prediksinya besar. Permasalahan jenis permintaan tersebut muncul pada kebutuhan *spare part* di PT X yang merupakan perusahaan peleburan aluminium. Untuk mengatasinya, model kebijakan *can order policy* menggunakan *multiple can-order level* (s_i, c_{ij}, S_{ij}) dapat menyelesaikan permasalahan dengan *joint replenishment* untuk mengkoordinasikan pesanan antar item. Model tersebut diketahui merupakan *integer non-linear programming* (INLP) yang dapat digolongkan ke dalam permasalahan *NP-Hard*. Pendekatan metaheuristik yaitu *simulated annealing* akan digunakan untuk mendapatkan solusi yang memuaskan dengan waktu penyelesaian yang cepat. Metode *Global Criterion* juga akan digunakan untuk mendapatkan fungsi *multi objective*, yaitu meminimasi total biaya persediaan dan meminimasi jumlah *carrier supplier*. Berdasarkan eksperimen dengan 3 *supplier* dam 157 *item* didapatkan hasil yang lebih baik dari kebijakan persediaan di PT X dengan adanya penghematan dari kondisi eksisting yang diterapkan perusahaan sebesar 2,6% pada jumlah *carrier* dan penghematan biaya sebesar 14,43% atau sebesar $81.570,79 (Rp 1.060.420.270).**

Kata Kunci—Can Order Policy,* Metaheuristik*, Multiple Can-Order Level, Simulated Annealing, Spare Part.

I. PENDAHULUAN

PENGENDALIAN persediaan yang dilakukan secara efektif dapat berperan sangat penting dalam hubungannya dengan *supply chain management*. Pengendalian *inventory* dilakukan dengan menentukan apa yang dipesan, berapa jumlah pesanan, dan kapan pemesanan dilakukan. Umumnya, pengendalian *inventory* digunakan untuk mengurangi adanya investasi, seperti biaya simpan, biaya pemesanan, dan biaya pembelian, dan juga untuk meningkatkan *customer service level* dengan menghindari adanya *shortage*.

Strategi penentuan pengendalian *inventory* ini salah satunya didasarkan pada klasifikasikan tipe permintaan yang berbeda. Permintaan dibagi berdasarkan 4 kategori, antara lain *smooth*, *erratic*, *intermittent*, dan *lumpy* [1]. Permasalahan yang terjadi pada jenis permintaan yang *intermittent* adalah sulitnya dalam menentukan strategi kebijakan persediaan karena pola pemintaan yang tidak pasti dan sulit untuk diprediksi.

Permasalahan jenis permintaan tersebut juga muncul pada kebutuhan *spare part* di PT X. PT X merupakan salah satu pabrik peleburan aluminium di Indonesia. Untuk menjaga keadalan dari peralatan produksi tersebut, perusahaan melakukan strategi *maintenance*. Untuk melakukan *preventive maintenance*, dibutuhkan adanya *spare part*, yang mana *spare part* tersebut merupakan komponen-komponen pendukung dari mesin-mesin produksi. Saat *spare part* tersebut tidak tersedia, proses *maintenance* yang seharusnya dilakukan, menjadi tidak dapat dilakukan sampai *spare part* dari mesin tersebut tersedia. Hal ini dapat mengakibatkan berhentinya proses produksi, yang mengakibatkan *demand* dari *customer* tidak bisa terpenuhi. Hal tersebut menunjukkan bahwa pentingnya pengendalian *inventory* pada *spare part*. PT X mengendalikan *spare part inventory* sebanyak kurang lebih 14.000 jenis *item*.

Dari *mechanical part* yang dibutuhkan pada 2018, 61% diantaranya tergolong ke dalam permintaan *intermittent.* Prediksi permintaan *spare part* yang *intermittent* tersebut yang selalu sulit untuk ditentukan berapa jumlah *part* di dalam proses *maintenance.* Selain itu, pemesanan *spare part* yang dilakukan secara independen pada PT X menyebabkan tingginya *ordering cost*. Padahal, proses tender pada *spare part* tidak terlalu signifikan berpengaruh, karena harus melakukan pembelian *item* yang banyak jenisnya dan penawaran dilihat berdasarkan kesesuaian spesifikasi *part*. Sehingga, perlu dilakukan kebijakan *inventory* yang efektif agar biaya *inventory* tetap rendah, dan jumlah *item* yang *stockout* juga rendah.

Seperti yang dikemukakan Ghorbel [2], untuk mengatasi adanya variasi pada suplai kuantitas dan periode yang bervariasi, maka digunakan *replenishment policies* jenis (s, S). Pada *multi-item inventory control*, *joint replenishment problem* (JRP) dapat mengoordinasikan jumlah kuantitas pesanan pada sejumlah *item* dalam satu gudang [3]. Ketika ada beberapa *item* yang dipesan pada *supplier* yang sama, biaya pemesanan akan bergantung pada jumlah pesanan dan jumlah *item* yang dipesan. Walaupun begitu, jumlah pesanan dari *item* yang berbeda adalah independen walaupun memiliki *supplier* yang sama. Untuk mengatasinya, *Can-order Policy*, yang merupakan pengembangan dari kebijakan (s, S), dapat mengendalikan koordinasi pesanan yang mengakomodasi keterkaitan antar *item* [4] dan memiliki hasil yang lebih baik dari *uncoordinated policy* dengan menghasilkan 20% *saving* [5]. *Can-order policy* dilakukan dengan menentukan 3 parameter (s, c, S), yaitu *re-order level (*s), *can-order level* (c), dan *order-up-to level* (S).

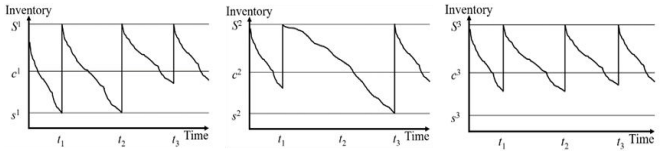
Pada formulasi model yang dibangun oleh Nagasawa & Irohara [3], model kebijakan (s, c, S) diketahui merupakan *integer non-linear programming* (INLP). Sehingga, permasalahan seperti ini digolongkan pada permasalahan *NP-Hard*. Sehingga, jika diselesaikan dengan metode eksak, akan membutuhkan waktu penyelesaian yang sangat lama karena *item* yang ditangani pada *spare part inventory* mencapai ratusan sampai ribuan *item*. Pendekatan metaheuristik akan digunakan untuk mendapatkan solusi yang memuaskan dengan waktu penyelesaian yang cepat. Nagasawa & Irohara [3] menyelesaikan permasalahan tersebut menggunakan *Genetic Algorithm*. Jika GA digunakan pada *spare part inventory* yang memiliki ribuan jenis *item*, waktu penyelesaian setiap iterasi dapat menjadi lama karena solusi dibangun berdasarkan populasi.

Oleh karena itu, penulis membangun model *multi objective multiple can order policy* menggunakan *Simulated Annealling*, agar diharapkan waktu penyelesaiannya lebih. Metode *Global Criterion* juga akan digunakan untuk mendapatkan fungsi *multi objective* yang mempertimbangkan fungsi tujuan berdasarkan selisihnya dengan solusi optimal. Fungsi tujuan dari model yang dikembangkan adalah untuk meminimasi total biaya persediaan dan meminimasi jumlah *carrier supplier* dengan memperhatikan kapasitas penyimpanan dan *multiple can-order level* untuk mengkoordinasikan *item* yang memiliki *supplier* yang sama.

II. LANDASAN TEORI

A. Can Order Policy

Can-order policy atau (s, c, S) *policy* adalah salah satu tipe dari *joint replenishment policy*. Balintfy [5] menunjukkan bahwa *can-order policy* setidaknya lebih baik dari pada *replenishment policy* terkoordinasi lainnya. *Can order policy* (sistem s, c, S) digunakan untuk mengendalikan *item* - *item* yang terkoordinasi dengan menggunakan 3 parameter, yaitu *order-up-to level* (S), *can-order level* (c), dan *must-order level* atau *re-order level* (s). Ketika level persediaan dari suatu *item* berada dibawah *s*, pesanan d*item*patkan pada level persediaan *item* tersebut mencapai *S*. Untuk *item* lain yang level persediannya dibawah *c*, maka pesanan akan d*item*patkan pada level persediaan mencapai *S*. Berikut merupakan gambaran dari persediaan menggunakan *can order policy*.

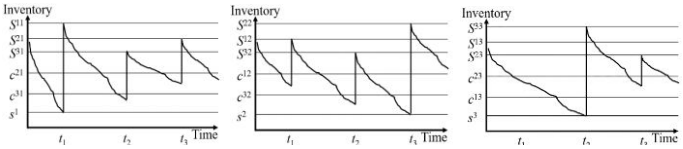


Gambar 1. Sistem *Can-Order Policy* pada 3 item.

B. Multiple Can-order Level pada Can-order Policy

Multiple can-order level pada kebijakan *can-order* diusulkan pertama kali oleh Nagasawa dan Irohara pada tahun 2016. Nagasawa [3] mengadopsi model pada *ordering policy* yang mempertimbangkan adanya korelasi *item. Correlated* demand merupakan adanya pengaruh dari permintaan dari 2 atau lebih *item* pada sekali pembelian dengan probabilitas tinggi. Dari model *can-order policy,* perubahan dilakukan pada *can-order level* dan *order-up-to level* bergantung pada *ordering trigger item*.

Pada model Nagasawa [3], untuk setiap *item i*, *ordering policy* yang dilakukan berupa (s_i, c_{ij}, S_{ij}). Ketika *inventory level* dari *item i* berada pada *re-order level* s_i, hal tersebut akan memicu *replenishment order* yang menaikkan level persediaan *item* i mencapai *order-up-to level* dari *item* tersebut S_{ii}. Pada waktu yang sama, jika level persediaan dari *item k* lainnya *berada* dibawah atau tepat pada *can-order level* ketika *item i* berada pada *re-order level*, lalu *can-order level*-nya disebut sebagai c_{ik}. *Item k* dimasukkan kedalam tambahan pesanan mencapai *order-up-to level* untuk *item I*, S_{ik}. Berikut merupakan gambaran *can-order policy* dengan *multiple can-order level*.



Gambar 2. *Multiple Can-order Level* pada Sistem (s, c, S) untuk 3 item yang berkorelasi

C. Algoritma Simulated Annealing

Metaheuristik merupakan metoda untuk mencari solusi yang memadukan interaksi antara prosedur pencarian lokal dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari lokal optimal [9]. Salah satu metode pada metaheuristik yang meniru peroses fisik adalah *Simulated Annealing* (SA). Algoritma SA pertama kali d*item*ukan oleh Metropolis, Rosenbluth dan Teller pada tahun 1953 untuk menyimulasikan proses pendinginan material dalam *heat bath*. *Annealing* adalah satu teknik yang dikenal dalam bidang metalurgi, digunakan dalam mempelajari proses pembentukan kristal dalam suatu materi [9]. Distribusi probabilitas Boltzmann ini digunakan sebagai acuan untuk menerima terlebih dahulu solusi yang lebih buruk atau tidak sehingga dapat menghindari jebakan *local optima*.

D. Metode Global Kriteria

Optimasi multi-tujuan merupakan persoalan optimasi yang memiliki lebih dari satu fungsi tujuan, yang memungkinkan terjadi konflik antar fungsi tujuan tersebut [9]. Global kriteria merupakan salah satu dari pendekatan *a priori* yang menjelaskan teknik untuk permasalahan fungsi tujuan dengan menjadikannya sebagai fungsi tunggal. Tujuannya adalah untuk mengukur seberapa dekat pengambil keputusan dapat menentukan solusi optimal. Fungsi *multi objective* yang digunakan pada global kriteria, adalah sebagai berikut [10].

$$Min \quad F = \sum_{l=1}^{K} \left[\frac{f_l(x^*) - f_l(x)}{f_l(x^*)} \right]^p \qquad (1)$$

dimana,

$f_l(x)$ = fungsi tujuan *l*

$f_l(x^*)$ = Nilai optimal fungsi objektif *l* (solusi fungsi ideal)

p = Bobot terhadap nilai penyimpangan

III. DESKRIPSI PERMASALAHAN

PT X merupakan salah satu pabrik peleburan aluminium terbesar di Asia Tenggara. Untuk melakukan proses produksi dari bahan baku sampai produk jadi, perusahaan membutuhkan kinerja dari mesin-mesin produksi. Sehingga,

keandalan dari mesin maupun *supply tools* dalam proses produksi merupakan salah satu faktor penting agar *demand* dari produk jadi tetap bisa dipenuhi seutuhnya.

Saat *spare part* tersebut tidak tersedia, maka proses *maintenance* yang seharusnya dilakukan menjadi tidak dapat dilakukan sampai *spare part* dari mesin tersebut tersedia. Hal ini dapat mengakibatkan berhentinya proses produksi, yang mengakibatkan *demand* dari *customer* tidak bisa terpenuhi. Hal tersebut menunjukkan bahwa pentingnya pengendalian *inventory* pada *spare part*.

Banyaknya proporsi permintaan yang tergolong *intermittent, lumpy*, maupun *erratic* menyebabkan kebijakan persediaan tidak bisa dilakukan dengan EOQ, maupun perhitungan *safety stock* pada umumnya karena tidak berdistribusi normal. Jika dilakukan kebijakan tersebut, biaya persediaannya termasuk biaya *safety stock* akan naik, karena adanya variabilitas yang sangat tinggi diakibatkan karena adanya 0 permintaan pada beberapa periode.

Pemesanan *spare part* yang dilakukan secara independen pada PT X menyebabkan tingginya *ordering cost*. Padahal, proses tender pada *spare part* tidak terlalu signifikan berpengaruh, karena harus melakukan pembelian *item* yang banyak jenisnya dan penawaran dilihat berdasarkan kesesuaian spesifikasi *part*. Jadi, saat ada *item* dengan *supplier* yang sama, sementara kebutuhannya pada periode yang berbeda, *item – item* tersebut akan dipesan secara independen pada periode dimana *item* tersebut dibutuhkan. Berikut merupakan contoh data yang akan digunakan.

Tabel 1.
Data Permintaan, Volume, dan Persediaan Awal Data Uji 1

Material Number	*Vendor*	*Initial Inventory*	*Volume* (cm^3)	*Demand*		
				t1	*t2*	*t3*
A822	AD2	0	443.7	0	1	0
A823	AD2	1	1841.508	1	0	3
A075	AD2	6	116.28	8	4	4

Tabel 2.
Data Biaya Persediaan Data Uji 1

Material Number	Harga	*Holding Cost*	*Ordering Cost*	*Backordering Cost*
A822	$ 323.42	$ 33.96	$ 25.70	$ 371.93
A823	$ 272.13	$ 28.57	$ 24.67	$ 312.95
A075	$ 516.24	$ 54.21	$ 29.56	$ 774.36

IV. PENGEMBANGAN MODEL DAN ALGORITMA

A. *Model Matematis*

Berikut merupakan model matematis yang dikembangkan dari model yang telah dibangun oleh Nagasawa [3] pada permasalahan *multiple can-order level*:

- Parameter:

i = jenis *item* (i ϵ I)
j = jenis *correlated item* (j ϵ J)
t = waktu perencanaan (t ϵ T)
M = big-M
d_{it} = *demand item* i pada periode ke t
v_i = volume *item* i
p_i = harga *item* i
Ch_i = *holding cost item* i
Co_i = *orderring cost item* i
Cb_i = *back-orderring cost item* i
b = kapasitas angkut
G = *storage capacity*

- Variabel Keputusan:

y_t = jumlah angkutan pada periode ke t
s_i = *re-order level item* i
c_{ij} = *can-order level item* j ketika *item* i dipesan
S_{ij} = *order-up-to level item* j ketika *item* i dipesan
o_{it} = jumlah *item* i yang *stockout* pada periode t
I_{it} = *inventory level item* i pada periode t
x_{it} = jumlah *order quantity item* i pada periode t
r_{it} = $\begin{cases}1, \text{ jika } I_{it} \text{ berada di bawah } s_i \text{ pada periode } t\\0, \text{ otherwise}\end{cases}$
k_{ijt} = $\begin{cases}1, \text{ jika } I_{it} \text{ berada di bawah } c_{ij} \text{ pada periode } t\\0, \text{ otherwise}\end{cases}$
q_{ijt} = $\begin{cases}1, \text{ jika } I_{it} \text{ berada di bawah } c_{ij}, \text{ dan setidaknya 1 item dipesan pada periode } t \text{ dan tambahan pesanan ditempatkan di } S_{ij}\\0, \text{ otherwise}\end{cases}$

- Fungsi Tujuan:

Minimasi Total *Carrier*:

$$Min \quad Z_1 = \sum_{t=1}^{T} y_t \tag{2}$$

Minimasi Total Biaya:

Total Biaya = Total *Purchasing Cost* + Total *Holding Cost* + Total *Ordering Cost* + Total *Backorderring Cost*

$$Min \quad Z_2 = \sum_{t=1}^{T}\sum_{i=1}^{I} x_{it} p_i + \sum_{t=1}^{T}\sum_{i=1}^{I} I_{it} Ch_i + \sum_{t=1}^{T}\sum_{i=1}^{I} r_{it} Co_i + \sum_{t=1}^{T}\sum_{i=1}^{I} o_{it} Cb_i \tag{3}$$

- Konstrain:

Keseimbangan persediaan untuk semua *item*:

$$I_{i,t-1} + x_{it} - I_{it} + o_{it} = d_{it}; \forall i, \forall t \tag{4}$$

Jumlah pesanan *item* selama periode waktu dibawah kapasitas angkut:

$$\sum_{i=1}^{I} v_i x_{it} \le b y_t; \forall t \tag{5}$$

Jika level persediaan jatuh pada atau di bawah *re-order level*, maka pesanan akan ditempatkan:

$$I_{i,t-1} - d_{it} + o_{it} + M r_{it} \ge s_i; \forall i, \forall t \tag{6}$$

$$I_{i,t-1} - d_{it} + o_{it} + M(1 - r_{it}) \le s_i; \forall i, \forall t \tag{7}$$

Kuantitas pesanan ditentukan dari selisih *order-up-to level* dan *inventory level:*

$$S_{ii} \le I_{i,t-1} - d_{it} + o_{it} + x_{it} + M(1 - r_{it}); \forall i, \forall t \tag{8}$$

$$S_{ii} \ge I_{i,t-1} - d_{it} + o_{it} + x_{it} - M(1 - r_{it}); \forall i, \forall t \tag{9}$$

Can-order level tidak lebih dari nilai *order-up-to level:*

$$S_{ij} \ge c_{ij}; \forall i, \forall j \tag{10}$$

Re-order level tidak lebih dari *can-order level*:

$$c_{ij} \ge s_j; \forall i, \forall j \tag{11}$$

Level persediaan dari suatu *item* apakah berada pada atau di bawah *can-order level* dari *item*:

$$I_{j,t-1} - d_{jt} + o_{jt} + M k_{ijt} \ge c_{ij}; \forall i, \forall j, \forall t \tag{12}$$

$$I_{j,t-1} - d_{jt} + o_{jt} - M(1 - k_{ijt}) \le c_{ij}; \forall i, \forall j, \forall t \tag{13}$$

Ketika level persediaan dari setidaknya satu *item* berada pada atau di bawah *re-order level*, pesanan ditempatkan termasuk *item* yang level persediaannya berada dibawah atau pada *can-order level*:

$$q_{ijt} \le k_{ijt}; \forall i, \forall j, \forall t \tag{14}$$

$$q_{ijt} \le r_{it}; \forall i, \forall j, \forall t \tag{15}$$

$$\sum_{i=1}^{I} q_{ijt} \le 1; \forall i, \forall t \tag{16}$$

$$c_{ij}k_{ijt} \le \sum_{i=1}^{I} c_{ij}q_{ijt}; \forall i, \forall j, \forall t \quad (17)$$

Kuantitas *item* yang dipesan sebesar selisih dari level persediaan dan *order-up-to level* yang memiliki hubungan dengan *item* lain terkorelasi:

$$S_{ij} \le I_{j,t-1} - d_{jt} + o_{jt} + x_{jt} + M(1-q_{ijt}); \forall i, \forall j, \forall t \quad (18)$$

$$S_{ij} \ge I_{j,t-1} - d_{jt} + o_{jt} + x_{jt} - M(1-q_{ijt}); \forall i, \forall j, \forall t \quad (19)$$

Item dengan level persediaan berada pada atau di bawah *can-order* dan *re-order level* bisa dilakukan pemesanan:

$$x_{jt} \le M(r_{jt} + \sum_{i=1}^{I} q_{ijt}); \forall j, \forall t \quad (20)$$

Jumlah *item* yang *shortage* pada setiap *item* tidak lebih dari permintaan:

$$o_{it} \le d_{it}; \forall i, \forall t \quad (21)$$

Jumlah *item* yang dipesan pada suatu periode tidak melebihi kapasitas penyimpanan*:*

$$\sum_{i=1}^{I} x_{it} \le G; \forall t \quad (22)$$

Konstrain variabel biner dan *non-negativity*

$$r_{it}, k_{ijt}, q_{ijt} = \{0,1\}; \forall i, \forall j, \forall t \quad (23)$$

$$y_t, I_{it}, x_{it}, o_{it}, s_i, c_{ij}, S_{ij} \ge 0; \forall i, \forall t \quad (24)$$

B. Pengembangan Algoritma Simulated Annealing

Pengembangan algoritma didasarkan pada model konseptual yang telah dibuat. Berikut ini adalah penjelasan langkah-langkah yang terdapat dalam algoritma SA yang digunakan:

Langkah 1: Inisialisasi

Parameter algoritma SA yang digunakan untuk menyelesaikan contoh kasus ini adalah faktor pereduksi temperatur (cr) dan jumlah siklus (n).

Langkah 2: Pembangkitan Solusi

Di dalam algoritma *simulated annealing*, solusi dibangkitkan secara *random.* Solusi awal yang digunakan terdiri dari beberapa variabel seperti, *re-order level* (s_i), *can-order level* (c_{ij}), *order-up-to level* (S_{ij}), dan jumlah item yang shortage (o_{it}).

Langkah 3: Perhitungan Fungsi Tujuan

Pada penyelesaian permasalahan multi tujuan, model akan menyelesaiakan kedua *problem* fungsi tujuan 1 dan fungsi tujuan 2 terlebih dahulu. Kemudian, model dilakukan *running* kembali dengan fungsi multi tujuan menggunakan metode global kriteria seperti pada Persamaan (1).

Langkah 4: Pembangkitan Solusi Baru

Initial solution yang telah dibangkitkan akan diubah menjadi nilai terdekatnya atau biasa disebut *neighborhood search* sebagai solusi baru. Perubahan solusi yang telah dibangkitkan dilakukan dengan cara melakukan kombinasi perubahan penambahan 1 satuan, pengurangan 1 satuan, atau tidak dilakukan perubahan (bertambah 0 satuan) pada variabel *s, c, S.*

Langkah 5: Membandingkan Solusi Lama dengan Solusi Baru

Setelah mendapatkan solusi baru kemudian dilakukan perhitungan Δf yang merupakan selisih antara fungsi tujuan yang didapatkan dengan solusi baru dan solusi yang lama. Jika nilai Δf *negative* maka solusi yang baru lebih baik dibandingkan dengan solusi yang sebelumnya sehingga 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 beru diterima atau tidak.

Langkah 6: *Update* Iterasi, Siklus dan Temperatur

Setelah mencapai jumlah siklus n, temperatur akan direduksi dengan menggunakan faktor pereduksi temperatur (cr). Selain itu, setelah mencapai jumlah siklus n, nilai siklus kembali ditetapkan menjadi sama dengan satu.

Langkah 7: *Stopping Criteria*

Dalam penelitian ini *stopping criteria* yang digunakan adalah nilai temperatur yang mendekati 0.

C. Verifikasi dan Validasi

Verifikasi model eksak dilakukan dengan melakuakan evaluasi struktur model yang digenerate dalam *software* LINGO. Evaluasi model didasarkan pada apakah model yang digenerate telah memiliki struktur yang sesuai dengan model matematisnya.

Dari hasil perhitungan model multi obyektif, hasilnya telah sama dengan perhitungan eksak. Tetapi, untuk mencapai hasil yang sama dibutuhkan replikasi dengan jumlah yang besar. Berikut merupakan rekapitulasi hasil verifikasi dan validasi menggunakan data 3 item dan 3 periode.

Tabel 3.
Perbandingan Hasil Verifikasi SA dengan Perhitungan Eksak pada Data Uji 1

Fungsi Tujuan		Nilai Solusi			Waktu Komputasi (detik)		
		Eksak	SA	Gap %	Eksak	SA	Gap%
Single Objektif	Z1	1	1	0.0%	3	19.8	560.00 %
	Z2	6441.25	6441.25	0.0%	18	23.6	31.11%
Multi Objektif	Z1	1	1	0.0%	870	40.8	-95.31%
	Z2	7187.63	7187.63	0.0%	870	40.8	-95.31%

Jika dilihat pada nilai solusi, perhitungan SA tidak memiliki perbedaan yang signifikan pada perhitungan eksak. Perbedaan terjadi saat dilakukan *running* model multi tujuan. Hal ini terjadi karena pada SA dibangkitkan bilangan random yang kemudian akan diperbaiki dengan mencari variabel keputusan berdasarkan *Neighborhood Search*. Sedangkan, jika dilihat pada waktu komputasi, Algoritma SA pada model single obyektif memiliki waktu yang lebih lama dari pada perhitungan eksak. Tetapi, bila dibandingkan dengan model multi obyektif, Algoritma SA lebih cepat dari pada perhitungan eksak. Hal ini dikarenakan terdapat fungsi *non-linear* pada fungsi *multi objective* multi tujuan.

Setelah model dengan 1 *supplier* dengan 3 item dan 3 periode telah valid, model akan dicoba *running* kembali menggunakan permasalahan *multi supplier* dengan ukuran 2 *supplier* dengan 3 item dari *supplier 1* dan 2 item dari *supplier 2* dengan 3 periode. Berikut merupakan rekapan hasil pada permasalahan kedua.

Tabel 4.
Perbandingan Hasil Verifikasi SA dengan Perhitungan Eksak pada Data Uji 2

Fungsi Tujuan		Nilai Solusi			Waktu Komputasi (detik)		
		Eksak	SA	GAP %	Eksak	SA	GAP%
Single Objektif	Z	1	1	0.00%	4	36.14	803.5 %
	Z2	8788.61	8788.61	0.00%	49	38.15	-22.1%

Multi Objektif	Z 1	1	1	0.00%	445	273	-38.7%
	Z 2	9644.32	10371	7.53%	445	273	-38.7%

Pada permasalahan multi supplier, didapatkan hasil solusi dengan perbedaan terbesar pada model multi obyektif fungsi tujuan biaya, yaitu sebesar 7.53%. Dari hasil tersebut, algoritma *Simulated Annealing* dapat dikatakan telah valid, mengacu pada hasil solusinya. Dengan demikian, model yang telah dibangun dapat dilanjutkan untuk permasalahan dengan ukuran yang lebih besar atau pada permasalahan nyata

V. EKSPERIMEN DAN ANALISA

Eksperimen dilakukan dengan menggunakan *software* MATLAB dan LINGO. *Software* MATLAB digunakan untuk menyelesaikan permasalahan dengan menggunakan algoritma *simulated annealing*. Spesifikasi komputer yang digunakan adalah Intel ® Core™ i5-4570 3.20 G.Hz (CPU), RAM 8 GB.

A. *Eksperimen Uji Parameter*

Dalam algoritma SA terdapat beberapa parameter yang digunakan untuk mengenerate solusi diantaranya adalah cr (faktor pereduksi temperatur) dan n (banyaknya siklus). Nilai parameter-parameter ini akan mempengaruhi kualitas dan kecepatan dalam men-*generate* solusi sehingga perlu dilakukan uji parameter untuk menentukan nilai parameter yang tepat untuk mendapatkan solusi yang terbaik. Nilai cr yang diuji adalah 0.9 dan 0,6. Sedangkan nilai maksimum iterasi tiap siklus yang diuji adalah 100 dan 1000.

Untuk melakukan pengujian parameter algoritma *simulated annealing*, data yang digunakan setiap data uji akan digunakan dengan jumlah *item*, *periode*, dan *supplier* yang berbeda. Berikut merupakan ukuran data yang akan digunakan dalam eksperiman uji parameter.

Tabel 5.
Ukuran Data Uji

Data	Produk	Periode	Supplier	Variabel	Konstrain
Data Uji 1	3	3	1	163	304
Data Uji 2	5	3	2	363	452
Data Uji 3	10	12	3	3482	4046
Data Studi Kasus	157	12	3	49.455++	57.000++

Data - data tersebut diambil dari data studi kasus *spare part inventory* PT X melalui sampel dengan memperhitungkan item dan *supplier* yang paling berkontribusi atau memiliki nilai yang tertinggi dari keseluruhan proses pengendalian persediaan. Berikut merupakan hasil perhitungan eksak dari data uji 1, 2, dan 3.

Tabel 6.
Hasil Perhitungan Eksak Setiap Data Uji

Hasil	Data Uji 1	Data Uji 2	Data Uji 3*
Jumlah Carrier (unit)	1	2	37
Total Biaya ($)	7187.63	9644.32	276464.2
Fungsi Multi objective	0.013	0.009	-
Waktu Komputasi (detik)	870	445	68965
Iterasi	3578609	978242	15100856

*Hasil Lokal Optimal

Dari uji parameter dengan data uji 1,2, dan 3, didapatkan input parameter yang optimal dengan nilai pereduksi temperatur sebesar 0.6 dan jumlah iterasi tiap siklus sebanyak 1000 iterasi. Setelah didapatkan parameter yang mendapatkan solusi paling baik, setiap data uji akan dilakukan eksperimen kembali dengan melihat perbedaan solusi yang terjadi jika dibandingkan solusi optimalnya dari 10 replikasi. Berikut merupakan hasil 10 replikasi pada data uji 2.

Tabel 7.
Hasil Eksperimen Data Uji 2

Replikasi	Jumlah Carrier (unit)		Total Biaya ($)		Multi objective		Waktu Komputasi (detik)	
	Nilai	Gap%	Nilai	Gap%	Nilai	Gap%	Nilai	Gap%
1	2	0.00%	9644.32	0.00%	0.009	0.00%	40.7	-90.90%
2	2	0.00%	9644.32	0.00%	0.009	0.00%	40.1	-91.00%
3	3	50.00%	8838.4	-8.4%	0.25	2537.40%	41.7	-90.60%
4	3	50.00%	8838.4	-8.4%	0.25	2537.40%	40.7	-90.90%
5	2	0.00%	9644.32	0.00%	0.009	0.00%	40.8	-90.80%
6	2	0.00%	10542	9.30%	0.04	319.90%	40.9	-90.80%
7	2	0.00%	9644.32	0.00%	0.009	0.00%	41	-90.80%
8	2	0.00%	9644.32	0.00%	0.009	0.00%	41.1	-90.80%
9	3	50.00%	8838.4	-8.4%	0.25	2537.40%	41.2	-90.70%
10	2	0.00%	9644.32	0.00%	0.009	0.00%	40.9	-90.80%

Berikut hasil rata-rata *gap* atau perbedaan hasil terhadap perhitungan eksak dari setiap data uji.

Tabel 8.
Rata-Rata Perbedaan Hasil Algoritma SA dengan Perhitungan Eksak

Data	Rata- Rata Gap %		
	Carrier	Biaya	Waktu
Data Uji 1	0.00%	2.04%	-95.30%
Data Uji 2	15.00%	-1.59%	-90.81%
Data Uji 3	-14.60%	-61.20%	-93.70%

B. *Eksperimen Penyelesaian Studi Kasus*

PT X saat ini menerapkan kebijakan persediaan (*s*, *T*) dengan *s* merupakan *re-order level* dan *T* adalah *replenishment period*. Saat persediaan menyentuh atau dibawah *re-order level*, yang ditetapkan perusahaan sebesar *safety stock*, perusahaan akan melakukan pemesanan item sebanyak permintaan selama 3 bulan kedepan dari periode dimana *level* persediaan dibawah *re-order level*. Dari 157 *item* yang disuplai 3 *supplier*, total biaya yang didapat sebesar $586.185,59 dan total *carrier supplier* sebanyak 76 unit.

Setelah didapatkan parameter yang menghasilkan solusi terbaik atau lebih cepat konvergen, algoritma SA dicoba untuk menyelesaikan permasalahan riil pada data studi kasus *spare part inventory* PT X. Berikut merupakan hasil yang didapatkan dari algoritma SA dengan parameter pereduksi temperatur sebesar 0.6 dan maksimum iterasi siklus sebesar 1000.

Tabel 9.
Hasil *Running* Algoritma SA pada Penyelesaian Studi Kasus

Replikasi	Jumlah *Carrier (unit)*	Total Biaya ($)	Waktu Komputasi (detik)
1	74	512,920	37,031.3
2	76	545,650	37,031.3
3	74	501,610	19,345.7
4	81	705,110	16,634.0
5	77	739,400	21,635.0
6	79	596,350	32,117.0
7	72	645,290	36,708.0
8	74	512,920	37,031.3
9	76	545,650	37,031.3
10	72	645,290	36,708.0
Rata-	75.5	595019	31,127.3

rata

Dari hasil tersebut dapat diketahui dari 74 *carrier*, 44 diantaranya merupakan *carrier* dari *supplier* CI10, serta

C. Analisa dan Hasil

Pada kondisi eksisting perusahaan menerapkan kebijakan *re-order level* sebesar *safety stock* dengan jumlah pemesanan sebesar kumulatif permintaan selama 3 bulan kedepan. Pemesanan *spare part* yang dilakukan secara independen pada PT X menyebabkan tingginya *ordering cost*.

Hal ini menyebabkan jumlah *carrier* yang diantarkan *supplier* menjadi tinggi karena bila pemesanan yang dilakukan berbeda periodenya. Dari hasil perhitungan, total biaya yang dikeluarkan perusahaan sebesar $586,185.59 dan total jumlah *carrier supplier* sebesar 76 unit.

Eksperimen yang dilakukan pada algoritma SA, didapatkan parameter terbaik sebesar 1000 maksimum siklus dan 0.6 pereduksi temperatur. Jika dilihat pada hasil tiap parameter, semakin lama waktu komputasi atau semakin besar jumlah iterasi, solusi yang didapatkan bisa jauh lebih optimal. Pada data uji 1 dan 2, solusi yang dihasilkan lebih mudah mencapai nilai optimalnya karena ukuran data yang kecil. Tetapi saat dicoba dengan ukuran data yang lebih besar pada data uji 3, hasil setiap replikasi bisa jauh berbeda. Bahkan, saat dilakukan eksperiman menggunakan metode eksak, hasilnya tidak bisa mencapai nilai optimalnya karena keterbatasan *memory* dan waktu. Jika bisa dilakukan lebih banyak iterasi lagi, bisa jadi model yang dibangun dapat mendapatkan solusi terbaiknya.

Setelah dilakukan perhitungan kondisi eksisting dan model optimasi, didapatkan hasil perbaikan seperti pada tabel berikut ini.

Tabel 10.
Perbandingan Jumlah *Carrier* Kondisi Eksisting dan Hasil Optimasi

Supplier	Carrier (unit)		*Saving %*
	Kondisi Eksisting	Hasil Optimasi	
AD2	13	14	-0.08%
CI10	41	44	-0.07%
GE16	22	16	27.27%
Total	**76**	**74**	**2.63%**

Perbedaan ini terjadi karena adanya perbedaan kebijakan mengenai *re-order level* dan jumlah pemesanan atau *order-up-to level*. Pada kondisi eksisting, terdapat beberapa item yang *re-order level*-nya lebih besar jika dibandingkan dengan hasil perbaikan.

Tabel 11.
Perbandingan Biaya Kondisi Eksisting dan Hasil Optimasi

Komponen Biaya	Biaya ($)		*Saving %*
	Kondisi Eksisting	Hasil Optimasi	
Biaya Pemesanan	$ 20,756.84	$ 4,994.80	75.94%
Biaya Pembelian	$ 406,771.11	$ 358,750.00	11.81%
Biaya Penyimpanan	$ 158,657.64	$ 140,870.00	11.21%
Total Biaya	$ 586,185.59	$ 504,614.80	14.43%

biaya terendah terletak pada biaya pemesanan. Sedangkan pada biaya *backorder* nilainya sebesar $0, karena tidak terjadi *shortage.* Hasil ini kemudian akan dibandingkan dengan kondisi eksisting yang diterapkan perusahaan.

Jika dilihat dari komponen setiap biaya, perubahan yang signifikan terjadi pada biaya pemesanan yaitu menurun sebesar 75.94%, hal ini terjadi karena pada model *multiple can-order level* yang dibangun memungkinkan terjadinya penyatuan pesanan pada setiap kelompok item yang memiliki kesamaan *supplier*.

Sedangkan pada total harga *item* dan biaya penyimpanan menurun sebesar 11.81% dan 11.21%. Penurunan biaya pembelian terjadi karena adanya perbedaan jumlah pesanan antara kondisi eksisting dan hasil perbaikan. Perbedaan kebijakan *re-order level* dan *order quantity* menyebabkan jumlah item yang dipesan juga menjadi berbeda. Hal tersebut juga mempengaruhi biaya penyimpanan yang juga menurun. Pada biaya *backorder,* nilai kondisi eksisting dan optimasi sebesar $0.

VI. KESIMPULAN

Berdasarkan hasil eksperimen yang telah dilakukan, dapat diketahui bahwa algoritma SA yang dibangun dapat menemukan nilai solusi yang mendekati optimum, tetapi sulit mencapai nilai global optimal bila dilakukan *running* model dengan jumlah data yang besar. Pada eksperimen yang dilakukan, *gap* yang terjadi antara hasil algortima SA dengan perhitungan eksak masih berselisih jauh. Hal ini terjadi karena saat dilakukan dalam jumlah yang besar, variabel yang dibangkitkan nilainya juga semakin besar dan meningkat secara eksponensial. Sehingga waktu komputasi yang dihasilkan juga menjadi sangat lama.

Berdasarkan eksperimen dengan 3 *supplier* dam 157 *item* menggunakan *multiple can-order level* dengan *can-order policy* yang dihasilkan algoritma *Simulated Annealing*, didapatkan hasil yang lebih baik dari kebijakan persediaan di PT X dengan biaya persediaan sebesar $501,610.00 dan jumlah *carrier* sebanyak 74 unit. Hasil tersebut menunjukkan adanya penghematan dari kondisi eksisting yang diterapkan perusahaan sebesar 2 unit pada jumlah *carrier*, dan penghematan biaya 14.43% atau sebesar $81,570.79 (Rp 1,060,420,270).

DAFTAR PUSTAKA

[1] A. Syntetos and et al, "On the Categorization of Demand Patterns," *J Opl Res Soc*, vol. 56, pp. 495–503, 2005.

[2] N. Ghorbel, S.-A. Addouche, and A. El-Mhamedi, "Replenishment Policies in Static and Dynamic Spare Part Inventory Control: A Survey," *IJEDR*, vol. 2, no. 4, 2014.

[3] K. Nagasawa and T. Irohara, "Multiple Can-order Level for Can-order Policies under Carrier Capacity and Correlated Demands," *J Jpn Ind Manag. Assoc*, vol. 67, pp. 114–123, 2016.

[4] I. A. Putra and I. N. Pujawan, "Pengendalian Persediaan Spare Part dengan Menggunakan Can-Ordering Policy Studi Kasus: PT. PJB Unit Pembangkitan Gresik," Surabaya, 2011.

[5] J. L. Balintfy, "On A Basic Class of Multi-Item Inventory Problems," *Manag. Sci*, vol. 10, pp. 287–297, 1964.