PENERAPAN ALGORITMA *CENTRIPETAL ACCELERATED PARTICLE SWARM OPTIMIZATION* PADA PERMASALAHAN *BOUNDED MULTIPLE KNAPSACK*

SKRIPSI

Oleh:

Riadhi Bahran Kasyfi
NIM 141810101028

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2019

PENERAPAN ALGORITMA *CENTRIPETAL ACCELERATED PARTICLE SWARM OPTIMIZATION* PADA PERMASALAHAN *BOUNDED MULTIPLE KNAPSACK*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan studi pada Program Studi Matematika (S1)
dan mencapai gelar sarjana Sains

Oleh:

Riadhi Bahran Kasyfi
NIM 141810101028

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2019

PERSEMBAHAN

Alhamdulillah, dengan segala puji bagi Allah yang dengan nikmat-Nya sempurnalah semua kebaikan, skripsi ini saya persembahkan untuk:

1. Ibu Dwi Abdi Amperawati dan Bapak Iwan Ebsilon Dewantoro Putro tercinta atas doa, kasih sayang tanpa batas, perhatian, dan segala kebaikan yang telah diberikan, semoga Allah selalu melindungi mereka dan memberikan umur panjang.
2. Saudara-saudaraku yang selalu memberikan dukungan serta doa
3. Guru-guru sejak taman kanak-kanak sampai perguruan tinggi yang telah memberikan ilmu dan bimbingan dengan penuh kesabaran serta keikhlasan
4. Extreme 2014 dan UKMS TITIK yang selalu memberikan dukungan, nasehat, keceriaan dan inspirasi.
5. Almamater Jurusan Matematika FMIPA Universitas Jember.

MOTO

"Kamu sungguh-sungguh akan diuji terhadap hartamu dan dirimu. Dan (juga) kamu benar-benar akan mendengar dari orang-orang yang diberi Al-Kitab sebelum kamu dan dari orang-orang yang mempersekutukan Allah, gangguan yang banyak yang menyakitkan hati. Jika kamu bersabar dan bertakwa, maka sesungguhnya yang demikian itu termasuk urusan yang patut diutamakan"
(QS. Ali' Imron: 186)

"Allah tidak membebani seseorang itu melainkan sesuai dengan kesanggupannya"
(QS. Al-Baqarah: 286)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Riadhi Bahran Kasyfi

NIM : 141810101028

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penerapan Algoritma *Centripetal Accelerated Particle Swarm Optimization* pada Permasalahan *Bounded Multiple Knapsack*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Januari 2019

Yang menyatakan,

Riadhi Bahran Kasyfi

NIM 141810101028

SKRIPSI

PENERAPAN ALGORITMA *CENTRIPETAL ACCELERATED PARTICLE SWARM OPTIMIZATION* PADA PERMASALAHAN *BOUNDED MULTIPLE KNAPSACK*

Oleh

Riadhi Bahran Kasyfi
NIM 141810101028

Pembimbing:

Dosen Pembimbing Utama : Ahmad Kamsyakawuni, S.Si., M.Kom.

Dosen Pembimbing Anggota : Abduh Riski, S.Si., M.Si.

PENGESAHAN

Skripsi berjudul "Penerapan Algoritma *Centripetal Accelerated Particle Swarm Optimization* pada Permasalahan *Bounded Multiple Knapsack*" telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Anggota I,
Ahmad Kamsyakawuni, S.Si., M.Kom. NIP. 197211291998021001	Abduh Riski, S.Si., M.Si. NIP. 199004062015041001
Anggota II,	Anggota III,
Dr. Kiswara Agung Santoso, S.Si., M.Kom. NIP. 197209071998031003	Ikhsanul Halikin, S.Pd., M.Si. NIP. 198610142014041001

Mengesahkan

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Jember

Drs. Sujito, Ph. D.

NIP. 196102041987111001

RINGKASAN

Penerapan Algoritma *Centripetal Accelerated Particle Swarm Optimization* pada Permasalahan *Bounded Multiple Knapsack*; Riadhi Bahran Kasyfi, 141810101028; 2019; 67 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Proses seleksi pada pengadaan barang suatu pabrik atau perusahaan sangat diperlukan agar perusahaan tidak mengalami kerugian. Perusahaan harus menentukan barang berserta jumlahnya yang akan dibeli dan dimasukkan pada sejumlah media penyimpanan yang dimiliki sedemikian sehingga tidak melebihi kapasitas media penyimpanan tersebut dan menghasilkan keuntungan yang maksimal. Dalam ilmu matematika permasalahan ini biasa disebut dengan permasalahan *bounded multiple knapsack*.

Penelitian ini akan menyelesaikan permasalahan *bounded multiple knapsack* menggunakan algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO). Tujuan dari penelitian ini adalah mendapatkan solusi optimum dan mengetahui keefektifan serta keefisiensian algoritma CAPSO dibandingkan dengan hasil dari metode *Simplex* dan hVEGA. Sebagai bahan simulasi, terdapat tiga data yang digunakan pada penelitian ini. Data pertama adalah data primer dari toko buah Barokah Jl. PB Soedirman, Patrang, Jember. Data kedua adalah data sekunder dari Wafirotullailiyah (2016). Data ketiga adalah data simulasi yang dibangkitkan secara acak. Data yang digunakan berisi nama barang, jumlah ketersediaan barang, berat setiap kemasan dan keuntungan. Terdapat dua *knapsack* yang digunakan sebagai media penyimpanan.

Berdasarkan hasil uji pengaruh parameter terhadap ketiga data yang digunakan diketahui bahwa parameter *pop* lebih memiliki pengaruh yang signifikan dibandingkan parameter maksimal iterasi. Semakin besar nilai parameter *pop*, hasil yang didapatkan semakin mendekati nilai optimalnya. Semakin besar maksimal iterasi, belum menjamin hasil yang didapatkan semakin baik.

Berdasarkan hasil percobaan akhir, algoritma CAPSO menemukan profit terbaik yang mampu ditemukan algoritma CAPSO pada data 1 sebesar Rp 3.555.000,- dan profit terburuknya adalah Rp 3.554.000,-. Pada data 2 profit terbaik yang mampu dihasilkan algoritma CAPSO sebesar Rp 7.538.650,- dan profit terburuknya adalah Rp 7.531.650,-. Pada data 3 profit terbaik yang didapatkan algoritma CAPSO sebesar Rp 8.512.000,- dan profit terburuknya adalah Rp 8.490.000,- Percobaan akhir tersebut menggunakan nilai *pop* = 1000 dan maksimal iterasi = 1000, dimana parameter *pop* lebih berpengaruh dibandingkan maksimal iterasi.

Hasil algoritma CAPSO dibandingkan hasil *Simplex* memiliki rata-rata persentase deviasi yang sangat kecil, yang artinya algoritma CAPSO efektif untuk menyelesaikan permasalahan *bounded multiple knapsack*. Algoritma CAPSO memiliki hasil yang lebih baik dan proses yang lebih cepat dibangdingkan dengan algoritma hVEGA, yang artinya algoritma CAPSO lebih efektif dan efisien.

PRAKATA

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala kuasa-Nya sehingga penulis dapat menyelesaiakan tugas akhir yang berjudul "Penerapan Algoritma *Centripetal Accelerated Particle Swarm Optimization* pada Permasalahan *Bounded Multiple Knapsack*". Penulisan tugas akhir ini dilakukan guna memenuhi salah satu syarat untuk menyelesaikan Program Studi Matematika (S1) dan mencapai gelar Sarjana Sains pada Jurusan Matematika FMIPA Universitas Jember.

Pada kesempatan ini, dengan segala hormat penulis mengucapkan terima kasih kepada:

1. Bapak Ahmad Kamsyakawuni, S.Si., M.Kom. selaku dosen pembimbing utama dan Bapak Abduh Riski, S.Si., M.Si. selaku dosen pembimbing anggota yang telah membimbing dalam penulisan tugas akhir ini.
2. Bapak Dr. Kiswara Agung Santoso, S.Si., M.Kom. selaku dosen penguji I dan Bapak Ikhsanul Halikin, S.Pd., M.Si. selaku dosen penguji II yang telah memberikan kritik dan saran.
3. Serta, beberapa pihak yang telah memberikan semangat dan doa.

Penulis menyadari bahwa penulisan ini masih jauh dari sempurna. Oleh karena itu, penulis mengharap kritik dan saran demi kesempurnaan penelitian selanjutnya. Semoga tugas akhir ini dapat bermanfaat bagi kita semua.

Jember, Januari 2019

Penulis

DAFTAR ISI

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Pengadaan barang merupakan suatu kegiatan yang menjadi implementasi suatu keputusan pabrik atau perusahaan untuk membeli barang yang dibutuhkan. Dalam proses pengadaan barang harus dilakukan beberapa tahap atau prosedur agar perusahaan tidak mengalami kerugian. Salah satu prosedur utama dalam pengadaan barang tersebut adalah seleksi. Proses seleksi sangat diperlukan untuk menentukan barang yang benar-benar dibutuhkan perusahaan dengan kondisi-kondisi yang harus dipertimbangkan, misalnya media penyimpanan. Barang-barang yang dibeli tidak boleh melebihi kapasitas media penyimpanan. Perusahaan dimungkinkan memiliki lebih dari satu media penyimpanan. Selain itu, barang yang akan dibeli dimungkinkan memiliki jumlah ketersediaan yang terbatas. Dengan demikian, perusahaan harus benar-benar memperhitungkan pemilihan barang agar memenuhi kondisi yang ada dan menghasilkan total keuntungan yang maksimal. Dalam ilmu matematika permasalahan ini biasa disebut dengan permasalahan *bounded multiple knapsack*.

Permasalahan *bounded multiple knapsack* atau *multidimensional bounded knapsack* merupakan salah satu variasi dari permasalahan *knapsack*, dimana terdapat beberapa barang yang harus dipilih dengan kendala kapasitas media dan jumlah barang yang terbatas. Kata *multiple* berarti media penyimpanan yang tersedia lebih dari satu. Tujuan dari *bounded multiple knapsack* adalah untuk memaksimalkan keuntungan. Permasalahan *bounded multiple knapsack* pernah diteliti Magazine dan Chern (1984) dengan menerapkan *fully polynomial approximation algorithm*. Penelitian lain oleh Akcay et al., (2007) menerapkan algoritma greedy. Pada penelitian tersebut disimpulkan bahwa algoritma heuristik greedy sangat kompetitif dibandingkan algoritma heuristik lain. Selain kedua penelitian tersebut, Wafirotullailiyah (2016) menyelesaikan permasalahan *bounded multiple knapsack* dengan menerapkan *Hybrid Virus Evolutionary Genetic Algorithm* (hVEGA). Algoritma hVEGA merupakan salah satu algoritma metaheuristik. Dari penelitian tersebut dapat disimpulkan bahwa algoritma hVEGA

mampu menghasilkan profit yang maksimum namun membutuhkan waktu komputasi yang cukup lama. Mengacu pada penelitian tersebut penulis tertarik untuk melakukan penelitian lebih lanjut tentang permasalahan *bounded multiple knapsack* dengan menggunakan algoritma metaheuristik lain.

Algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) merupakan salah satu algoritma metaheuristik yang dikenalkan oleh Beheshti dan Shamsuddin (2014) dalam penelitiannya untuk menyelesaikan persamaan nonlinier. Berdasarkan penelitian tersebut disimpulkan bahwa algoritma CAPSO memiliki beberapa kelebihan yaitu konsep sederhana, mudah diimplementasikan dan tidak membutuhkan parameter-parameter khusus. Selain itu, dari hasil percobaan pada penelitian tersebut menunjukkan bahwa algoritma CAPSO menghasilkan performa yang lebih baik dari beberapa algoritma metaheuristik lain, seperti *Gravitational Search Algorithm* (GSA) dan *Particle Swarm Optimization* (PSO). Algoritma CAPSO memiliki tingkat konvergensi yang cepat serta akurasi solusi yang baik.

Berdasarkan uraian di atas, penulis tertarik untuk meneliti lebih lanjut tentang permasalahan *bounded multiple knapsack.* Peneliti akan mengimplementasikan algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO). Dari penelitian ini diharapkan dapat ditemukan solusi yang optimal dengan waktu komputasi yang lebih cepat dibandingkan algoritma hVEGA.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat dirumuskan rumusan masalah sebagai berikut.

a. Bagaimana solusi optimum algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) pada permasalahan *bounded multiple knapsack*?
b. Bagaimana keefektifan dan keefisiensian algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) dibandingkan dengan metode *Simplex* dan hVEGA?

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian penyelesaian permasalahan *bounded multiple knapsack* ini adalah diasumsikan bahwa permintaan barang setiap konsumen sama.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut.

a. Mendapatkan solusi optimum pada permasalahan *bounded multiple knapsack* menggunakan algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO).
b. Mengetahui keefektifan dan keefisiensian algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) dibandingkan dengan hasil dari metode *Simplex* dan hVEGA.

1.5 Manfaat

Manfaat dari penelitian ini diharapkan dapat dijadikan referensi dalam penelitian selanjutnya yang berhubungan tentang permasalahan *bounded multiple knapsack,* algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO), serta dapat dijadikan acuan dalam pemilihan barang.

BAB 2. TINJAUAN PUSTAKA

2.1 Permasalahan *Knapsack*

Permasalahan *knapsack* merupakan masalah pemilihan barang yang akan disimpan pada suatu media penyimpanan dari sekian banyak barang dimana setiap barang memiliki berat dan keuntungan (*profit*). Penyelesaian permasalahan *knapsack* dilakukan dengan memilih barang-barang yang mempunyai keuntungan maksimum dan total berat tidak melebihi kapasitas media penyimpanan. Permasalahan *knapsack* telah diterapkan dalam berbagai bidang, termasuk *Resource Distribution*, *Investment Decision Making*, *Budget Controlling*, *Project Selection* dan sebagainya (Gherboudj, *et al*., 2012).

Knapsack memiliki beberapa jenis persoalan yaitu sebagai berikut (Pisinger, 1995).

a. *Knapsack* 0-1 (*binary knapsack*)

   Permasalahan *knapsack* dimana barang yang dimasukkan ke dalam media penyimpanan harus semuanya (1) atau tidak sama sekali (0).

b. *Knapsack* terbatas (*bounded knapsack*)

   Permasalahan *knapsack* dimana setiap barang tersedia sebanyak n unit dan jumlah barang yang dimasukkan ke dalam media penyimpanan terbatas.

c. *Knapsack* tak terbatas (*unbounded knapsack*)

   Permasalahan *knapsack* di mana setiap barang tersedia lebih dari satu unit dan jumlah barang yang dimasukkan ke dalam media penyimpanan tidak terbatas.

Terdapat banyak variasi dari masalah *knapsack* yang muncul dengan aplikasi jumlah besar dari masalah dasar. Variasi utama terjadi dengan mengubah jumlah parameter masalah seperti jumlah kendala, jumlah tujuan, atau bahkan jumlah medianya.

Beberapa variasi permasalahan *knapsack* di atas antara lain (Kellerer, *et al*., 2004):

a. Permasalahan *multiobjective knapsack*

   Permasalahan yang memiliki fungsi tujuan lebih dari satu untuk memaksimalkan keuntungannya.

b. Permasalahan *multiple constraints knapsack*

Permasalahan yang memiliki kendala lebih dari satu untuk memaksimalkan keuntungannya.

c. Permasalahan *multidimensional*/*multiple knapsack*

Permasalahan yang memiliki media penyimpanan lebih dari satu untuk memaksimalkan keuntungan

d. Permasalahan *quadratic knapsack*

Permasalahan yang tujuannya memaksimalkan fungsi objektif dalam bentuk kuadaratik untuk kendala kapasitas biner dan linier.

2.1.1 Permasalahan *Bounded Knapsack*

Misalkan, diberikan n buah objek untuk diangkut dalam sebuah media penyimpanan yang memiliki daya tampung c. Setiap objek j memiliki berat w_j, nilai keuntungan p_j dan batas ketersediaan barang b_j. Permasalahannya adalah bagaimana memilih barang dengan jumlah tertentu x_j, dimana jumlah barang tidak melebihi batas ketersediaannya.

Permasalahan *bounded knapsack* diformulasikan secara matematis sebagai berikut.

Fungsi tujuan:

$$\text{Maks } Z = \sum_{j=1}^{n} p_j x_j \tag{2.1}$$

Kendala

$$\sum_{j=1}^{n} w_j x_j \leq C \tag{2.2}$$

$$x_j \in \{0, 1, \dots, b_j\} \quad ; \; j = 1, 2, \dots, n \tag{2.3}$$

dimana

x_j : jumlah unit barang ke-j yang dimasukkan ke dalam *knapsack*

p_j : keuntungan/profit barang ke-j per unit

w_j : berat barang ke-j per unit

C : kapasitas maksimal *knapsack*

b_j : batas ketersediaan barang ke-j

n : banyak barang

2.1.2 Permasalahan *Bounded Multiple Knapsack*

Permasalahan *bounded multiple knapsack* merupakan suatu permasalahan *knapsack* dimana terdapat lebih dari satu *knapsack* yang digunakan untuk mengangkut semua atau sebagian item barang yang tersedia, dimana tiap-tiap item mempunyai batasan jumlah masing-masing. Tujuannya untuk memperoleh solusi optimal dengan adanya *knapsack* yang jumlahnya lebih dari satu dengan kapasitas masing-masing lebih kecil atau sama dengan kapasitas maksimum *knapsack* yang ada.

Permasalahan *bounded multiple knapsack* diformulasikan secara matematis sebagai berikut:

Fungsi tujuan:

$$\text{Maks } Z = \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{2.4}$$

Kendala

$$\sum_{j=1}^{n} w_j x_{ij} \leq C_i \quad ; \; i = 1,2,\dots,m \tag{2.5}$$

$$\sum_{i}^{m} x_{ij} \leq b_j \quad ; \; j = 1,2,\dots,n \tag{2.6}$$

$$x_{ij} \in \{0,1,\dots,b_j\} \quad ; \; i = 1,2,\dots,m; \; j = 1,2,\dots,n \tag{2.7}$$

dimana

x_{ij} : jumlah unit barang ke-j yang dimasukkan ke dalam *knapsack* ke-i

p_j : keuntungan/profit barang ke-j per unit

w_j : berat barang ke-j per unit

C_i : kapasitas maksimal *knapsack* ke-i

b_j : batas ketersediaan barang ke-j

m : banyak *knapsack*

n : banyak barang

2.2 Algoritma Metaheuristik

Algoritma merupakan suatu metode atau tahapan sistematis yang digunakan untuk memecahkan suatu permasalahan. Algoritma juga dapat dikatakan sebagai langkah secara sistematis dan logis dalam penyelesaian masalah. Penulisan algoritma ditulis dengan notasi khusus yang mudah dimengerti dan dapat

diterjemahkan kedalam suatu bahasa pemrograman. Suatu algoritma memerlukan masukan (*input*) tertentu untuk memulai sehingga menghasilkan keluaran (*output*) tertentu.

Sebuah algoritma tidak hanya harus benar (efektif) tetapi juga harus efisien. Algoritma yang baik adalah algoritma yang dapat meminimumkan kebutuhan ruang dan waktu. Ruang memori yang dibutuhkan dan lamanya waktu proses untuk menjalankan suatu algoritma menjadi hal yang perlu diperhatikan. Hal ini di karenakan meskipun suatu algoritma memberikan hasil yang mendekati optimal tetapi waktu yang dibutuhkan sangat lama, maka algoritma tersebut jarang digunakan dalam penyelesaian masalah optimasi (Munir, 2005).

Metaheuristik merupakan algoritma yang didesain untuk menyelesaikan permasalahan optimasi melalui proses pendekatan. Algoritma metaheuristik secara umum diterapkan pada permasalahan-permasalahan yang tidak ada penyelesaian secara spesifik yang memenuhi. Algoritma metaheuristik secara luas digunakan untuk menyelesaikan permasalahan kompleks dalam industri dan jasa. Hampir semua algoritma metaheuristik terinspirasi dari alam, seperti prinsip biologi, fisika, atau etologi. Algoritma metaheuristik menggunakan komponen stokastik yang artinya mengandung variabel random. Algoritma metaheuristik juga memiliki parameter yang dibutuhkan untuk menyesuaikan pada permasalahan (Boussaid, *et al*.,2013).

2.3 Algoritma *Centripetal Accelerated Particle Swarm Optimization*

Algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) merupakan algoritma optimasi metaheuristik yang didasarkan pada hukum pergerakan Newton. Algoritma CAPSO diterapkan pada topologi lokal dan global (Beheshti, *et al*., 2014).

Dalam hukum pergerakan Newton, kecepatan dan posisi akhir dari setiap objek dengan percepatan konstan dihitung berdasarkan Persamaan (2.8) dan (2.9) berikut:

$$V_2 = V_1 + a \times \Delta t \tag{2.8}$$

$$X_2 = X_1 + \frac{1}{2} \times a \times \Delta t^2 + V_1 \times \Delta t \tag{2.9}$$

dimana V_1 dan V_2 merupakan kecepatan awal dan kecepatan akhir. a mereprestasikan percepatan objek. X_1 dan X_2 adalah posisi awal dan posisi akhir. Δt didefinisikan sebagai selisih antara pergerakan waktu akhir dan waktu awal:

$$\Delta t = t_2 - t_1 \tag{2.10}$$

Dalam algoritma CAPSO setiap kandidat solusi dianggap sebagai partikel dalam ruang pencarian multidimensi dan memiliki empat spesifikasi: posisi (X), kecepatan (V), percepatan (a) dan percepatan sentripetal (A). Setiap partikel mengubah posisinya berdasarkan pengalamannya sendiri dan tetangganya.

Untuk ruang pencarian d-dimensi dengan N partikel, posisi dan kecepatan dari partikel ke-i direpresentasikan pada Persamaan (2.11) dan (2.12)

$$\overrightarrow{X_i} = (x_{i1}, x_{i2}, \ldots, x_{id}) \quad ; i = 1, 2, \ldots, N \tag{2.11}$$

$$\overrightarrow{V_i} = (v_{i1}, v_{i2}, \ldots, v_{id}) \quad ; i = 1, 2, \ldots, N \tag{2.12}$$

dimana x_{id} dan v_{id} menunjukkan posisi dan kecepatan dari partikel ke-i di dimensi ke-d.

Posisi terbaik setiap partikel (P_i) dan posisi terbaik global (P_g) dituliskan dalam Persamaan (2.13) dan (2.14):

$$\overrightarrow{P_i} = (p_{i1}, p_{i2}, \ldots, p_{id}) \quad ; i = 1, 2, \ldots, N \tag{2.13}$$

$$\overrightarrow{P_g} = (p_{g1}, p_{g2}, \ldots, p_{gd}) \tag{2.14}$$

Ketika partikel bergerak dalam ruang pencarian, partikel tersebut harus dipercepat menuju optimum global. Oleh karena itu, dua jenis percepatan dapat dianggap sebagai berikut:

$$a_{id}(t) = rand \times \left(P_{id}(t) - X_{id}(t)\right) + rand \times \left(P_{gd}(t) - X_{id}(t)\right) \tag{2.15}$$

$$A_{id}(t) = E_i(t) \times rand \times \left(P_{id}(t) - P_{md}(t) - X_{id}(t)\right) \tag{2.16}$$

dimana:

$a_{id}(t)$: percepatan partikel ke-i dalam dimensi ke-d

$A_{id}(t)$: percepatan sentripetal partikel ke-i dalam dimensi ke-d

$rand$: variabel random dengan distribusi seragam dalam interval [0, 1]

$x_{id}(t)$: posisi saat ini

$p_{id}(t)$: posisi terbaik partikel ke-i dalam dimensi ke-d

$p_{gd}(t)$: posisi terbaik global dalam dimensi ke-d

$p_{md}(t)$: posisi tengah dalam dimensi ke-d

$E_i(t)$: koefisien percepatan yang dihitung dengan Persamaan (2.17) dan (2.18)

$$e_i(t) = \frac{fit_i(t) - GWfit(t)}{Avgfit(t) - GWfit(t)} \tag{2.17}$$

$$E_i(t) = \frac{e_i(t)}{\sum_{j=1}^{N} e_j(t)} \tag{2.18}$$

dimana:

$fit_i(t)$: nilai *fitness* dari partikel ke-i

$GWfit(t)$: nilai *fitness* terburuk

$Avgfit(t)$: rata-rata *fitness*

Nilai *fitness* dari partikel ke-i pada umumnya dihutung berdasarkan Persamaan (2.19).

$$fit_i = \begin{cases} Z & \text{, jika permasalahan maksimasi} \\ \frac{1}{Z} & \text{, jika permasalahan minimasi} \end{cases} \tag{2.19}$$

Dengan demikian, kecepatan dari partikel ke-i dalam dimensi ke-d diperbarui berdasarkan Persamaan (2.20) berikut:

$$v_{id}(t+1) = v_{id}(t) + a_{id}(t) + A_{id}(t) \tag{2.20}$$

dimana $v_{id}(t)$ adalah kecepatan saat ini dan $v_{id}(t+1)$ adalah kecepatan baru.

Selanjutnya posisi partikel diperbarui berdasarkan Persamaan (2.21).

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2.21}$$

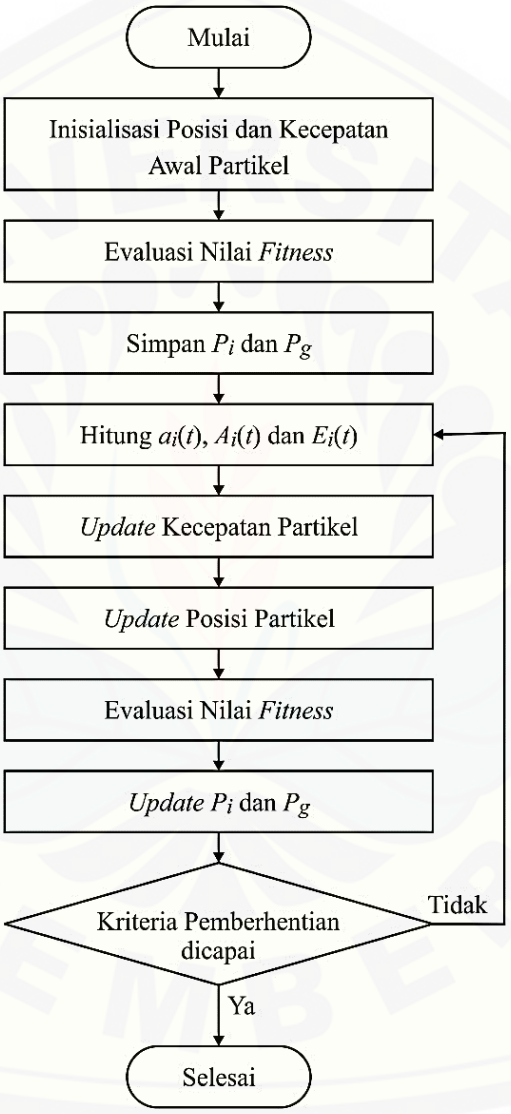
dimana $x_{id}(t)$ adalah posisi saat ini dan $x_{id}(t+1)$ adalah posisi baru.

Berikut langkah-langkah algoritma CAPSO:

a. Bangkitkan posisi dari populasi partikel secara acak atau random, dan kecepatan awal diawali dengan nilai yang sama yaitu nol (0).
b. Evaluasi nilai *fitness* dari setiap partikel(fit_i).
c. Simpan posisi terbaik setiap partikel (P_i) dan posisi terbaik global (P_g).
d. Hitung $a_i(t)$, $A_i(t)$ and $E_i(t)$ untuk $i = 1, 2, \ldots, N$.
e. Perbarui kecepatan partikel.
f. Perbarui posisi partikel.
g. Evaluasi nilai *fitness* dari setiap partikel(fit_i).

h. Perbarui posisi terbaik setiap partikel (P_i) dan posisi terbaik global (P_g). Jika nilai *fitness* dari setiap partikel (fit_i) lebih baik dari nilai *fitness* terbaik partikel tersebut (P_i) maka perbarui (P_i). Jika nilai *fitness* terbaik lebih baik dari pada nilai *fitness* terbaik global (P_g), maka perbarui P_g.

i. Ulangi langkah d – h hingga kriteria pemberhentian terpenuhi



Gambar 2.1 Skema algoritma CAPSO

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Sebagai bahan simulasi, terdapat tiga data yang digunakan pada penelitian ini. Data pertama adalah data primer dari toko buah Barokah Jl. PB Soedirman, Patrang, Jember. Data kedua adalah data sekunder dari Wafirotullailiyah (2016). Data ketiga adalah data simulasi yang dibangkitkan secara acak. Data yang digunakan berisi nama barang, jumlah ketersediaan barang, berat setiap kemasan dan keuntungan. Terdapat dua *knapsack* yang digunakan sebagai media penyimpanan. Ketiga data tersebut dapat dilihat pada Lampiran A.

3.2 Langkah-langkah Penelitian

Adapun langkah-langkah yang akan dilakukan pada penelitian ini untuk menyelesaikan permasalahan *bounded multiple knapsack* sebagai berikut:

a. Studi Literatur

   Langkah pertama dalam penelitian ini adalah pengumpulan literatur tentang algoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) dan permasalahan *bounded multiple knapsack.* Kemudian dilakukan studi terhadap literatur tersebut.

b. Pengumpulan dan Pengidentifikasian Data

   Pada langkah ini dilakukan pengumpulan data dari toko buah Barokah, pengambilan data pada Wafirotullailiyah (2016), serta pembangkitan data simulasi. Data yang dikumpulkan disesuaikan dengan yang dibutuhkan, antara lain: nama barang, jumlah ketersediaan, berat satuan, keuntungan.

c. Penerapan algoritma *Centripetal Accelerated Particle Swarm Optimization*

   1) Inisialisasi partikel awal

      a) Pembangkitan posisi awal

$$\mathbf{X} = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^m \end{bmatrix} \quad ; \quad X^k = \begin{bmatrix} x_{1,1}^k & x_{1,2}^k & \dots & x_{1,d}^k \\ x_{2,1}^k & x_{2,2}^k & \dots & x_{2,d}^k \\ \vdots & \vdots & \ddots & \vdots \\ x_{pop,1}^k & x_{pop,2}^k & \dots & x_{pop,d}^k \end{bmatrix}, k = 1, 2, \dots, m \quad (3.1)$$

$$x_{i,j}^{k} \in [0,1]\ ;\ i = 1,2,\dots,pop\ ;\ j = 1,2,\dots,d\ ;\ k = 1,2,\dots,m$$

dimana:

x_{ij}^{k} : posisi partikel ke-i dimensi ke-j *knapsack* ke-k

m : banyaknya *knapsack*

pop : banyaknya partikel

d : dimensi atau banyaknya barang

b) Diskritisasi solusi

$$\mathbf{Y} = \begin{bmatrix} Y^1 \\ Y^2 \\ \vdots \\ Y^m \end{bmatrix} \quad ; \quad Y^k = \begin{bmatrix} y_{1,1}^k & y_{1,2}^k & \cdots & y_{1,d}^k \\ y_{2,1}^k & y_{2,2}^k & \cdots & y_{2,d}^k \\ \vdots & \vdots & \ddots & \vdots \\ y_{pop,1}^k & y_{pop,2}^k & \cdots & y_{pop,d}^k \end{bmatrix}, k = 1,2,\dots,m \quad (3.2)$$

$$y_{i,j}^{k} = round\left(x_{i,j}^{k} \times b_j\right)\ ;\ i = 1,2,\dots,pop\ ;\ j = 1,2,\dots,d\ ;\ k = 1,2,\dots,m$$

dimana:

y_{ij}^{k} : solusi ke-i dimensi ke-j *knapsack* ke-k

b_j : jumlah ketersediaan barang ke-j

c) Pembangkitan kecepatan awal

$$\mathbf{V} = \begin{bmatrix} V^1 \\ V^2 \\ \vdots \\ V^m \end{bmatrix} \quad ; \quad V^k = \begin{bmatrix} v_{1,1}^k & v_{1,2}^k & \cdots & v_{1,d}^k \\ v_{2,1}^k & v_{2,2}^k & \cdots & v_{2,d}^k \\ \vdots & \vdots & \ddots & \vdots \\ v_{pop,1}^k & v_{pop,2}^k & \cdots & v_{pop,d}^k \end{bmatrix}, k = 1,2,\dots,m$$

dimana:

v_{ij}^{k} : kecepatan partikel ke-i dimensi ke-j *knapsack* ke-k

2) Pengecekan Solusi

Setiap solusi harus memenuhi kendala berikut

$$\sum_{k=1}^{m} y_{i,j}^{k} \le b_j \quad ;\ i = 1,2,\dots,pop\ ;\ j = 1,2,\dots,d \quad (3.3)$$

$$\sum_{j=1}^{d} y_{i,j}^{k} \times w_j \le C_k\ ;\ i = 1,2,\dots,pop\ ;\ k = 1,2,\dots,m \quad (3.4)$$

dimana:

w_j : berat barang ke-j

C_k : kapasitas *knapsack* ke-k

Jika solusi yang tidak memenuhi kendala, maka perlu dilakukan pinalti.

3) Hitung *Fitness*

Nilai *fitness* dari setiap solusi dihitung berdasarkan total profitnya.

$$Fitness_i = \sum_{k=1}^{m} \sum_{j=i}^{d} y_{i,j}^{k} \times p_j \quad ; \quad i = 1, 2, \dots, pop \qquad (3.5)$$

dimana:

p_j : profit barang ke-j

4) Simpan posisi terbaik setiap partikel (P_i) dan posisi terbaik global (P_g).
5) Hitung $a_i(t)$, $A_i(t)$ and $E_i(t)$ untuk $i = 1, 2, \dots, pop$.
6) Perbarui kecepatan partikel.
7) Perbarui posisi partikel.
8) Pengecekan solusi.
9) Hitung *fitness* dari posisi baru setiap partikel.
10) Perbarui posisi terbaik setiap partikel (P_i) dan posisi terbaik global (P_g). Jika nilai *fitness* dari setiap partikel (fit_i) lebih baik dari nilai *fitness* terbaik partikel tersebut (P_i) maka perbarui (P_i). Jika nilai *fitness* terbaik lebih baik daripada nilai *fitness* terbaik global (P_g), maka perbarui P_g.
11) Ulangi langkah 5) – 10) hingga iterasi maksimal dicapai.

d. Pembuatan Program

Pada langkah ini yaitu membuat program penerapan algoritma *Centripetal Accelerated Particle Swarm Optimization* untuk menyelesaikan permasalahan *bounded multiple knapsack* menggunakan *software* MATLAB. Program dibuat dalam bentuk *Graphical User Interface* (GUI).

e. Simulasi Program

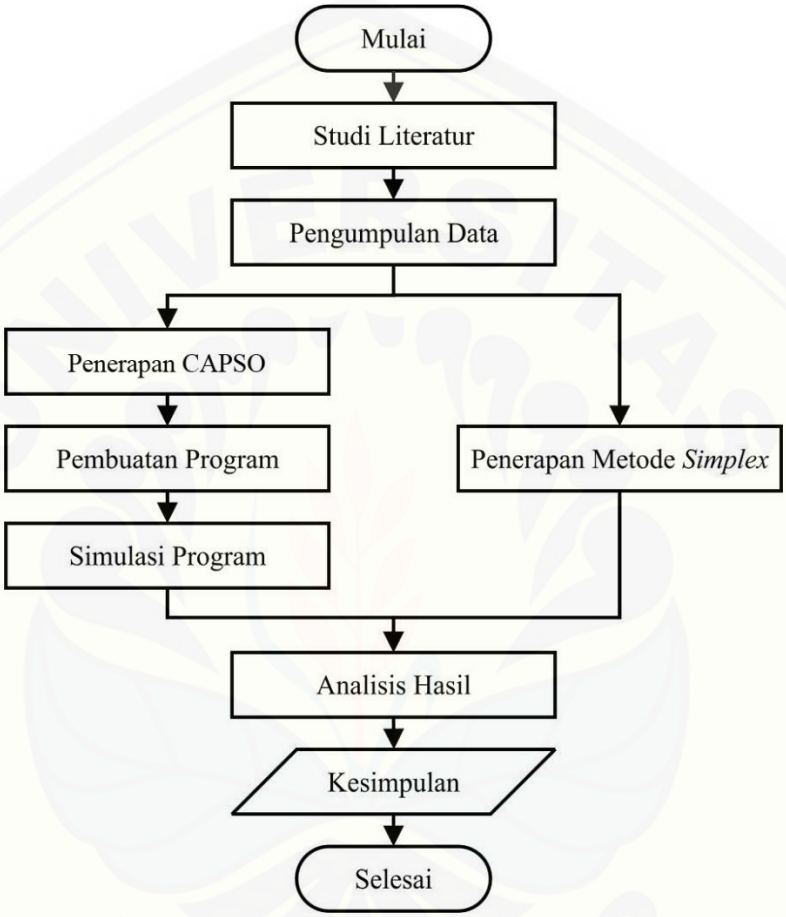
Program yang telah dibuat akan disimulasikan untuk menyelesaikan permasalahan dengan menggunakan data yang dikumpulkan.

f. Analisis Hasil

Hasil yang diperoleh dari algoritma *Centripetal Accelerated Particle Swarm Optimization* kemudian akan dibandingkan dengan hasil metode *Simplex* dan hVEGA. Metode *Simplex* dijalankan melalui Solver Add-in pada Microsoft Excel. Hasil dari hVEGA diambil dari penelitian Wafirotullailiyah (2016).

g. Kesimpulan

Setelah hasil dianalisis, selanjutnya akan ditarik kesimpulan mengenai solusi terbaik, tingkat keefektifan dan keefisiensian algoritma *Centripetal Accelerated Particle Swarm Optimization* pada permasalahan *bounded multiple knapsack*.



Gambar 3.1 Skema langkah-langkah penelitian

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan, diambil kesimpulan sebagai berikut.

a. Solusi terbaik yang dihasilkan oleh algoritma CAPSO pada ketiga data yang digunakan adalah Rp 3.555.000,- (data 1), Rp 7.538.650,- (data 2) dan Rp 8.512.000,- (data 3). Solusi terbaik tersebut diperoleh dengan menggunakan nilai parameter *pop* = 1000 dan maksimal iterasi = 1000, dimana parameter *pop* lebih berpengaruh dibandingkan maksimal iterasi.
b. Hasil algoritma CAPSO dibandingkan hasil *Simplex* memiliki rata-rata persentase deviasi yang kecil, yang artinya algoritma CAPSO efektif untuk menyelesaikan permasalahan *bounded multiple knapsack*. Algoritma CAPSO memiliki hasil yang lebih baik dan proses yang lebih cepat dibangdingkan dengan algoritma hVEGA, yang artinya algoritma CAPSO lebih efektif dan efisien.

5.2 Saran

Saran untuk penelitian berikutnya dapat mengembangkan permasalahan *bounded knapsack* dimana kendala yang digunakan tidak hanya berat barang, akan tetapi juga volume serta modal. Penelitian berikutnya dapat menggunakan *single knapsack* maupun *multiple knapsack*. Selain itu, disarankan untuk menerapkan alsoritma *Centripetal Accelerated Particle Swarm Optimization* (CAPSO) pada permasalahan optimasi lain.

DAFTAR PUSTAKA

Akcay, Y., H. Li dan S.H. Xu. 2007. Greedy Algorithm for the General Multidimensional Knapsack Problem. *Ann Oper Res*, 150:17-29.

Beheshti, Z. dan S.M. Shamsuddin. 2014. CAPSO: Centripetal Accelerated Particle Swarm Optimization. *Information Sciences*, 258: 54-79.

Boussaid, I., J. Lepagnot dan P. Siarry. 2013. A survey on optimization metaheuristics. *Information Sciences*, 237: 82-117.

Gherboudj, A., A. Layeb dan S. Chikhi. 2012. Solving 0-1 Knapsack Problem by a Discrete Binary Version of Cuckoo Search Algorithm. *International Journal Bio-Inspired Computation*, 4(4): 229-236.

Kellerer, H., D. Pisinger dan U. Pferschy. 2004. *Knapsack Problem*. Berlin: Springer.

Magazine, M.J. dan M.S. Chern. 1984. A Note on Approximation Schemes for Multidimensional Knapsack Problem. Mathematics of Operations Reasearch, 9(2): 244-247.

Munir, R. 2005. *Matematika Diskrit. Revisi 5*. Bandung: Informatika.

Pisinger, D. 1995. A Minimal Algorithm for the Multiple-Choice Knapsack Problem. *European Journal of Operational Research*, 83(2): 394-410.

Wafirotullailiyah. 2016. Penerapan Hybrid Virus Evolutionary Genetic Algorithm (hVEGA) pada Bounded Multiple Knapsack. *Skripsi*. Jember: Jurusan Matematika FMIPA Universitas Jember.

LAMPIRAN

Lampiran A. Data Penelitian

A.1 Data 1 (Kapasitas *Knapsack* = 600kg)

No	Nama Barang	Jumlah	Berat Satuan (kg)	Harga Beli (Rp)	Harga Jual (Rp)
1	Semangka	10	20	80.000	100.000
2	Salak	12	5	35.000	45.000
3	Nanas	3	7	28.000	42.000
4	Pepaya	5	15	45.000	75.000
5	Manggis	12	3	45.000	60.000
6	Pisang Susu	10	2.5	20.000	25.000
7	Pisang Saji	10	1.5	7.500	10.000
8	Pisang Ambon	12	2.5	10.000	15.000
9	Pisang Berlin	9	1	6.000	8.000
10	Pisang Kepok	6	1.5	20.000	25.000
11	Kelengkeng	5	3	90.000	105.000
12	Mangga Manalagi	8	10	70.000	100.000
13	Mangga Golek	5	10	70.000	100.000
14	Mangga Gadung	8	10	80.000	120.000
15	Mangga Madu	7	10	60.000	100.000
16	Anggur	8	1	30.000	35.000
17	Jeruk	4	10	80.000	120.000
18	Apel Merah	5	5	125.000	150.000
19	Apel Malang	8	5	65.000	90.000
20	Pir	5	5	65.000	90.000
21	Buah Naga	5	5	40.000	65.000
22	Alpukat	20	5	50.000	75.000
23	Belimbing	4	10	20.000	30.000
24	Sawo	5	4	16.000	20.000
25	Melon	5	20	80.000	100.000
26	Kesemek	5	4	20.000	26.000
27	Jambu Merah	8	5	20.000	25.000
28	Jambu Kristal	8	5	75.000	100.000
29	Strawberri	8	1	6.000	8.000

A.2 Data 2 (Kapasitas *Knapsack* = 750kg)

No	Nama Barang	Jumlah Ketersediaan (kemasan)	Berat Satuan (kg)	Profit (Rp)
1	Anggur	10	0,5	2.300
2	Alpukat	7	5	20.000
3	Apel	12	15	47.000
4	Belimbing	5	10	12.000
5	Buah Naga	11	20	52.000
6	Duku	8	1	2.200
7	Durian	0	1	3.000
8	Jambu Merah	4	10	14.000
9	Jambu Kristal	8	10	18.500
10	Jeruk	9	20	75.000
11	Kesemek	2	3	7.500
12	Mangga	13	15	255.000
13	Manggis	3	5	9.700
14	Melon	5	10	22.000
15	Nanas	7	12	23.500
16	Nangka	14	1	1.850
17	Pepaya	6	20	41.000
18	Salak	12	20	112.000
19	Sawo	3	3	6.500
20	Semangka	2	4,5	8.500
21	Strowberi	16	0,5	1.800
22	Anggur Merah	11	0,5	2.500
23	Langsep	4	1	1.250
24	Kelengkeng	6	2,5	5.000
25	Rambutan	12	5	5.750

A.3 Data 3 (Kapasitas *Knapsack* = 1500kg)

No	Nama Barang	Jumlah Ketersediaan (Kemasan)	Berat per Kemasan (kg)	Profit per Kemasan (Rp)
1	Barang 1	30	5	10000
2	Barang 2	12	10	10000
3	Barang 3	25	4	8000
4	Barang 4	15	5	10000
5	Barang 5	12	1	2000
6	Barang 6	40	2	10000
7	Barang 7	18	8	32000
8	Barang 8	20	3	3000
9	Barang 9	5	10	50000
10	Barang 10	18	1	3000
11	Barang 11	16	10	30000
12	Barang 12	20	5	5000

No	Nama Barang	Jumlah Ketersediaan (Kemasan)	Berat per Kemasan (kg)	Profit per Kemasan (Rp)
13	Barang 13	11	8	24000
14	Barang 14	10	10	50000
15	Barang 15	17	3	15000
16	Barang 16	5	4	4000
17	Barang 17	10	5	10000
18	Barang 18	20	8	24000
19	Barang 19	12	8	8000
20	Barang 20	18	1	4000
21	Barang 21	15	2	2000
22	Barang 22	12	4	8000
23	Barang 23	40	0,5	1000
24	Barang 24	14	5	5000
25	Barang 25	2	3	9000
26	Barang 26	15	2	10000
27	Barang 27	14	2	10000
28	Barang 28	15	10	40000
29	Barang 29	18	8	24000
30	Barang 30	20	5	10000
31	Barang 31	16	10	20000
32	Barang 32	25	1	1000
33	Barang 33	20	8	24000
34	Barang 34	12	8	24000
35	Barang 35	1	5	15000
36	Barang 36	5	2	6000
37	Barang 37	18	5	10000
38	Barang 38	10	3	12000
39	Barang 39	30	1	4000
40	Barang 40	5	2	8000
41	Barang 41	12	10	10000
42	Barang 42	15	1	2000
43	Barang 43	4	3	3000
44	Barang 44	15	1	4000
45	Barang 45	8	5	20000
46	Barang 46	30	0,5	1000
47	Barang 47	10	10	30000
48	Barang 48	4	2	8000
49	Barang 49	8	1	5000
50	Barang 50	5	3	15000

Lampiran B. Hasil Metode *Simplex* dengan *Solver* Add-In Ms. Excel

B.1 Hasil *Simplex* Data 1

Microsoft Excel 16.0 Answer Report
Worksheet: [Data Barokah.xlsx]Sheet3
Report Created: 05/12/2018 13.46.27
Result: Solver found an integer solution within tolerance. All Constraints are satisfied.
Solver Engine
Engine: Simplex LP
Solution Time: 350,516 Seconds.
Iterations: 6 Subproblems: 112360
Solver Options
Max Time 600 sec, Iterations Unlimited, Precision 0,000001, Use Automatic Scaling
Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 0%, Assume NonNegative

Objective Cell (Max)

Cell	Name	Original Value	Final Value
R33	P2	3551000	3555000

Variable Cells

Cell	Name	Original Value	Final Value	Integer
L2	Semangka K1	0	0	Integer
M2	Semangka K2	7	7	Integer
L3	Salak K1	0	0	Integer
M3	Salak K2	12	12	Integer
L4	Nanas K1	0	0	Integer
M4	Nanas K2	3	3	Integer
L5	Pepaya K1	0	0	Integer
M5	Pepaya K2	5	5	Integer
L6	Manggis K1	0	0	Integer
M6	Manggis K2	12	12	Integer
L7	Pisang Susu K1	1	1	Integer
M7	Pisang Susu K2	9	9	Integer
L8	Pisang Saji K1	0	0	Integer
M8	Pisang Saji K2	10	10	Integer
L9	Pisang Ambon K1	0	0	Integer
M9	Pisang Ambon K2	12	12	Integer
L10	Pisang Berlin K1	0	0	Integer
M10	Pisang Berlin K2	9	9	Integer
L11	Pisang Kepok K1	0	5	Integer
M11	Pisang Kepok K2	6	1	Integer
L12	Kelengkeng K1	5	5	Integer
M12	Kelengkeng K2	0	0	Integer
L13	Mangga Manalagi K1	5	6	Integer
M13	Mangga Manalagi K2	3	2	Integer
L14	Mangga Golek K1	5	5	Integer
M14	Mangga Golek K2	0	0	Integer
L15	Mangga Gadung K1	8	8	Integer

M15	Mangga Gadung K2	0	0	Integer
L16	Mangga Madu K1	7	7	Integer
M16	Mangga Madu K2	0	0	Integer
L17	Anggur K1	7	8	Integer
M17	Anggur K2	1	0	Integer
L18	Jeruk K1	4	4	Integer
M18	Jeruk K2	0	0	Integer
L19	Apel Merah K1	5	5	Integer
M19	Apel Merah K2	0	0	Integer
L20	Apel Malang K1	8	8	Integer
M20	Apel Malang K2	0	0	Integer
L21	Pir K1	5	5	Integer
M21	Pir K2	0	0	Integer
L22	Buah Naga K1	5	5	Integer
M22	Buah Naga K2	0	0	Integer
L23	Alpukat K1	20	20	Integer
M23	Alpukat K2	0	0	Integer
L24	Belimbing K1	0	0	Integer
M24	Belimbing K2	1	1	Integer
L25	Sawo K1	5	1	Integer
M25	Sawo K2	0	0	Integer
L26	Melon K1	0	0	Integer
M26	Melon K2	5	5	Integer
L27	Kesemek K1	0	0	Integer
M27	Kesemek K2	5	5	Integer
L28	Jambu Merah K1	2	1	Integer
M28	Jambu Merah K2	2	7	Integer
L29	Jambu Kristal K1	8	8	Integer
M29	Jambu Kristal K2	0	0	Integer
L30	Strawberri K1	0	3	Integer
M30	Strawberri K2	8	5	Integer

B.2 Hasil *Simplex* Data 2

Microsoft Excel 16.0 Answer Report
Worksheet: [Simpleks.xlsx]Diskrit
Report Created: 09/11/2018 10.33.10
Result: Solver found an integer solution within tolerance. All Constraints are satisfied.
Solver Engine
Engine: Simplex
LP
Solution Time: 2,032 Seconds.
Iterations: 10 Subproblems: 1042
Solver Options
Max Time Unlimited, Iterations Unlimited, Precision 0,000001, Use Automatic Scaling
Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%, Assume NonNegative

Objective Cell
(Max)

Cell	Name	Original Value	Final Value
M28	P1	7541150	7541150

Variable Cells

Cell	Name	Original Value	Final Value	Integer
H2	Anggur Knapsack 1	2	2	Integer
I2	Anggur Knapsack 2	8	8	Integer
H3	Alpukat Knapsack 1	5	5	Integer
I3	Alpukat Knapsack 2	2	2	Integer
H4	Apel Knapsack 1	11	11	Integer
I4	Apel Knapsack 2	1	1	Integer
H5	Belimbing Knapsack 1	0	0	Integer
I5	Belimbing Knapsack 2	0	0	Integer
H6	Buah Naga Knapsack 1	0	0	Integer
I6	Buah Naga Knapsack 2	11	11	Integer
H7	Duku Knapsack 1	0	0	Integer
I7	Duku Knapsack 2	8	8	Integer
H8	Durian Knapsack 1	0	0	Integer
I8	Durian Knapsack 2	0	0	Integer
H9	Jambu Merah Knapsack 1	2	2	Integer
I9	Jambu Merah Knapsack 2	0	0	Integer
H10	Jambu Kristal Knapsack 1	0	0	Integer
I10	Jambu Kristal Knapsack 2	8	8	Integer
H11	Jeruk Knapsack 1	0	0	Integer
I11	Jeruk Knapsack 2	9	9	Integer
H12	Kesemek Knapsack 1	0	0	Integer
I12	Kesemek Knapsack 2	2	2	Integer
H13	Mangga Knapsack 1	13	13	Integer
I13	Mangga Knapsack 2	0	0	Integer
H14	Manggis Knapsack 1	0	0	Integer
I14	Manggis Knapsack 2	3	3	Integer
H15	Melon Knapsack 1	0	0	Integer
I15	Melon Knapsack 2	5	5	Integer
H16	Nanas Knapsack 1	0	0	Integer
I16	Nanas Knapsack 2	7	7	Integer
H17	Nangka Knapsack 1	0	0	Integer
I17	Nangka Knapsack 2	14	14	Integer
H18	Pepaya Knapsack 1	6	6	Integer
I18	Pepaya Knapsack 2	0	0	Integer
H19	Salak Knapsack 1	10	10	Integer
I19	Salak Knapsack 2	2	2	Integer
H20	Sawo Knapsack 1	2	2	Integer
I20	Sawo Knapsack 2	1	1	Integer
H21	Semangka Knapsack 1	1	1	Integer
I21	Semangka Knapsack 2	1	1	Integer
H22	Strowberi Knapsack 1	16	16	Integer
I22	Strowberi Knapsack 2	0	0	Integer

H23	Anggur Merah Knapsack 1	11	11	Integer
I23	Anggur Merah Knapsack 2	0	0	Integer
H24	Langsep Knapsack 1	0	0	Integer
I24	Langsep Knapsack 2	1	1	Integer
H25	Kelengkeng Knapsack 1	0	0	Integer
I25	Kelengkeng Knapsack 2	6	6	Integer
H26	Rambutan Knapsack 1	0	0	Integer
I26	Rambutan Knapsack 2	0	0	Integer

B.3 Hasil *Simplex* Data 3

Microsoft Excel 16.0 Answer Report
Worksheet: [Simplex data 50.xlsx]Diskrit
Report Created: 09/11/2018 10.37.06
Result: Solver found an integer solution within tolerance. All Constraints are satisfied.
Solver Engine
Engine: Simplex
LP
Solution Time: 0,546 Seconds.
Iterations: 3 Subproblems: 10
Solver Options
Max Time Unlimited, Iterations Unlimited, Precision 0,000001, Use Automatic Scaling
Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%, Assume NonNegative

Objective Cell (Max)

Cell	Name	Original Value	Final Value
N53	P2	8512000	8512000

Variable Cells

Cell	Name	Original Value	Final Value	Integer
H2	Barang 1 Knapsack 1	0	0	Integer
I2	Barang 1 Knapsack 2	30	30	Integer
H3	Barang 2 Knapsack 1	0	0	Integer
I3	Barang 2 Knapsack 2	0	0	Integer
H4	Barang 3 Knapsack 1	0	0	Integer
I4	Barang 3 Knapsack 2	25	25	Integer
H5	Barang 4 Knapsack 1	0	0	Integer
I5	Barang 4 Knapsack 2	15	15	Integer
H6	Barang 5 Knapsack 1	0	0	Integer
I6	Barang 5 Knapsack 2	12	12	Integer
H7	Barang 6 Knapsack 1	38	38	Integer
I7	Barang 6 Knapsack 2	2	2	Integer
H8	Barang 7 Knapsack 1	18	18	Integer
I8	Barang 7 Knapsack 2	0	0	Integer
H9	Barang 8 Knapsack 1	1	1	Integer
I9	Barang 8 Knapsack 2	0	0	Integer

H10	Barang 9 Knapsack 1	5	5	Integer
I10	Barang 9 Knapsack 2	0	0	Integer
H11	Barang 10 Knapsack 1	0	0	Integer
I11	Barang 10 Knapsack 2	18	18	Integer
H12	Barang 11 Knapsack 1	2	2	Integer
I12	Barang 11 Knapsack 2	14	14	Integer
H13	Barang 12 Knapsack 1	0	0	Integer
I13	Barang 12 Knapsack 2	20	20	Integer
H14	Barang 13 Knapsack 1	11	11	Integer
I14	Barang 13 Knapsack 2	0	0	Integer
H15	Barang 14 Knapsack 1	10	10	Integer
I15	Barang 14 Knapsack 2	0	0	Integer
H16	Barang 15 Knapsack 1	17	17	Integer
I16	Barang 15 Knapsack 2	0	0	Integer
H17	Barang 16 Knapsack 1	0	0	Integer
I17	Barang 16 Knapsack 2	5	5	Integer
H18	Barang 17 Knapsack 1	0	0	Integer
I18	Barang 17 Knapsack 2	10	10	Integer
H19	Barang 18 Knapsack 1	20	20	Integer
I19	Barang 18 Knapsack 2	0	0	Integer
H20	Barang 19 Knapsack 1	0	0	Integer
I20	Barang 19 Knapsack 2	12	12	Integer
H21	Barang 20 Knapsack 1	0	0	Integer
I21	Barang 20 Knapsack 2	18	18	Integer
H22	Barang 21 Knapsack 1	1	1	Integer
I22	Barang 21 Knapsack 2	0	0	Integer
H23	Barang 22 Knapsack 1	0	0	Integer
I23	Barang 22 Knapsack 2	12	12	Integer
H24	Barang 23 Knapsack 1	0	0	Integer
I24	Barang 23 Knapsack 2	40	40	Integer
H25	Barang 24 Knapsack 1	0	0	Integer
I25	Barang 24 Knapsack 2	14	14	Integer
H26	Barang 25 Knapsack 1	0	0	Integer
I26	Barang 25 Knapsack 2	2	2	Integer
H27	Barang 26 Knapsack 1	15	15	Integer
I27	Barang 26 Knapsack 2	0	0	Integer
H28	Barang 27 Knapsack 1	14	14	Integer
I28	Barang 27 Knapsack 2	0	0	Integer
H29	Barang 28 Knapsack 1	15	15	Integer
I29	Barang 28 Knapsack 2	0	0	Integer
H30	Barang 29 Knapsack 1	18	18	Integer
I30	Barang 29 Knapsack 2	0	0	Integer
H31	Barang 30 Knapsack 1	0	0	Integer
I31	Barang 30 Knapsack 2	20	20	Integer
H32	Barang 31 Knapsack 1	0	0	Integer
I32	Barang 31 Knapsack 2	16	16	Integer
H33	Barang 32 Knapsack 1	0	0	Integer
I33	Barang 32 Knapsack 2	0	0	Integer
H34	Barang 33 Knapsack 1	20	20	Integer
I34	Barang 33 Knapsack 2	0	0	Integer
H35	Barang 34 Knapsack 1	12	12	Integer
I35	Barang 34 Knapsack 2	0	0	Integer
H36	Barang 35 Knapsack 1	1	1	Integer

I36	Barang 35 Knapsack 2	0	0	Integer
H37	Barang 36 Knapsack 1	0	0	Integer
I37	Barang 36 Knapsack 2	5	5	Integer
H38	Barang 37 Knapsack 1	0	0	Integer
I38	Barang 37 Knapsack 2	18	18	Integer
H39	Barang 38 Knapsack 1	10	10	Integer
I39	Barang 38 Knapsack 2	0	0	Integer
H40	Barang 39 Knapsack 1	0	0	Integer
I40	Barang 39 Knapsack 2	30	30	Integer
H41	Barang 40 Knapsack 1	0	0	Integer
I41	Barang 40 Knapsack 2	5	5	Integer
H42	Barang 41 Knapsack 1	0	0	Integer
I42	Barang 41 Knapsack 2	12	12	Integer
H43	Barang 42 Knapsack 1	0	0	Integer
I43	Barang 42 Knapsack 2	15	15	Integer
H44	Barang 43 Knapsack 1	0	0	Integer
I44	Barang 43 Knapsack 2	0	0	Integer
H45	Barang 44 Knapsack 1	0	0	Integer
I45	Barang 44 Knapsack 2	15	15	Integer
H46	Barang 45 Knapsack 1	8	8	Integer
I46	Barang 45 Knapsack 2	0	0	Integer
H47	Barang 46 Knapsack 1	0	0	Integer
I47	Barang 46 Knapsack 2	30	30	Integer
H48	Barang 47 Knapsack 1	10	10	Integer
I48	Barang 47 Knapsack 2	0	0	Integer
H49	Barang 48 Knapsack 1	0	0	Integer
I49	Barang 48 Knapsack 2	4	4	Integer
H50	Barang 49 Knapsack 1	8	8	Integer
I50	Barang 49 Knapsack 2	0	0	Integer
H51	Barang 50 Knapsack 1	5	5	Integer
I51	Barang 50 Knapsack 2	0	0	Integer

Lampiran C. Hasil Percobaan Uji Parameter

C.1 Uji Parameter Populasi Data 1

Pop = 25

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 600	3554000	430	46,4933
2	599,5; 599,5	3552000	9	46,2578
3	600; 600	3547000	13	46,035
4	598,5; 600	3554000	40	43,707
5	599; 600	3552000	303	42,8389
6	596,5; 599,5	3551000	9	43,5299
7	599; 599,5	3549000	403	43,675
8	597,5; 599	3548000	46	43,135
9	599; 600	3550000	330	44,0552
10	599,5; 599	3535000	700	41,8485
Rata-rata		3549200	228,3	44,15756

Pop = 50

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	599,5; 598,5	3553000	175	46,8054
2	600; 600	3554000	229	48,446
3	599; 598,5	3544500	39	49,4146
4	598,5; 599,5	3553000	175	47,1014
5	600; 598	3551000	5	47,3819
6	596; 600	3551000	18	47,8465
7	600; 599,5	3553500	31	54,1539
8	599,5; 599,5	3552000	40	49,7252
9	600; 598,5	3549000	556	49,8489
10	600; 600	3553000	477	49,5307
Rata-rata		3551400	174,5	49,02545

Pop = 100

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	599,5; 599,5	3554000	545	56,7313
2	598; 600	3553000	547	57,5419
3	600; 596	3551000	39	62,3008
4	600; 600	3554000	209	57,2011
5	596; 600	3551000	34	62,2202
6	598,5; 599,5	3553000	27	54,0835
7	600; 598	3553000	82	58,8451
8	599,5; 599,5	3549000	19	51,446
9	600; 600	3554000	311	56,3588
10	600; 600	3554000	159	53,9336
Rata-rata		3552600	197,2	57,06623

Pop = 200

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 600	3554000	115	79,6312
2	598; 600	3553000	28	75,6631
3	600; 599	3554000	189	76,235
4	600; 600	3554000	255	77,0342
5	600; 599,5	3553500	27	74,5431
6	600; 600	3554000	387	77,8761
7	600; 600	3554000	189	72,763
8	599; 600	3547000	23	71,3736
9	599; 600	3554000	772	72,3416
10	600; 600	3554000	55	70,8608
Rata-rata		3553150	204	74,83217

Pop = 500

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 599,5	3553500	22	131,6055
2	600; 600	3553000	24	122,0115
3	600; 600	3555000	920	117,2992
4	600; 600	3554000	19	140,366
5	600; 600	3554000	80	130,9253
6	600; 600	3554000	167	122,5864
7	600; 600	3554000	361	122,3286
8	600; 600	3554000	11	126,1808
9	600; 600	3554000	203	130,8874
10	600; 600	3554000	262	117,1137
Rata-rata		3553950	206,9	126,13044

C.2 Uji Parameter Populasi Data 2

Pop = 25

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	748,5; 750	7505800	453	41,943
2	750; 749,5	7522000	818	43,7399
3	750; 750	7507950	943	42,3393
4	749; 750	7520150	967	43,5855
5	750; 750	7505150	937	42,7732
6	750; 750	7519100	844	44,2477
7	749,5; 750	7502150	446	43,1668
8	750; 749,5	7515400	690	44,6909
9	750; 750	7513550	961	44,5124
10	748; 749,5	7495300	768	42,5724
Rata-rata		7510655	782,7	43,35711

Pop = 50

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	748; 750	7513400	939	47,3746
2	749; 750	7519900	920	48,3397
3	749,5; 750	7505600	908	48,5968
4	750; 749,5	7513900	542	46,795
5	748; 750	7514900	265	48,8131
6	750; 750	7496350	538	49,6596
7	750; 750	7499200	434	46,174
8	748; 750	7502550	316	47,9823
9	750; 749,5	7524300	935	48,1759
10	750; 750	7526850	994	47,2227
Rata-rata		7511695	679,1	47,91337

Pop = 100

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 748,5	7512450	457	53,1506
2	749; 749	7531400	328	56,2652
3	750; 749,5	7505250	878	54,9986
4	750; 749	7519450	994	56,9429
5	750; 749,5	7522900	931	54,547
6	750; 750	7514950	580	55,3676
7	750; 750	7527550	638	53,9492
8	750; 750	7507850	317	54,7328
9	750; 748,5	7522650	995	55,5227
10	749,5; 749,5	7512400	908	59,671
Rata-rata		7517685	702,6	55,51476

Pop = 200

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 750	7527300	630	70,5771
2	750; 750	7520250	795	67,1549
3	750; 750	7522200	760	70,9146
4	750; 750	7532250	780	78,4432
5	750; 750	7526250	856	74,4169
6	749,5; 750	7536150	387	75,1218
7	750; 750	7529750	415	70,2816
8	750; 750	7535600	646	74,7897
9	750; 750	7525850	768	70,8536
10	750; 750	7516300	956	68,2452
Rata-rata		7527190	699,3	72,07986

Pop = 500

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 750	7524300	501	124,9798
2	750; 750	7530950	900	127,1237
3	750; 750	7535300	958	120,888
4	750; 749,5	7529850	806	129,8563
5	750; 749,5	7535900	486	128,2009
6	750; 749,5	7529950	271	115,4439
7	750; 750	7527850	934	129,6277
8	750; 749,5	7533400	276	123,1218
9	750; 748,5	7531150	683	140,6484
10	750; 749,5	7538050	475	137,3704
Rata-rata		7531670	629	127,72609

C.3 Uji Parameter Populasi Data 3

Pop = 25

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1499; 1500	8485000	290	44,9677
2	1497,5; 1499,5	8495000	363	44,8945
3	1497,5; 1498,5	8467000	379	43,9735
4	1498; 1500	8495000	427	46,1092
5	1499; 1500	8457000	145	45,6187
6	1500; 1500	8499000	641	45,0914
7	1498; 1500	8479000	807	46,0849
8	1499,5; 1498,5	8484000	723	46,3895
9	1500; 1500	8486000	663	47,9745
10	1500; 1500	8418000	748	46,94
Rata-rata		8476500	518,6	45,80439

Pop = 50

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1500; 1498	8485000	958	53,861
2	1500; 1500	8468000	607	50,3883
3	1500; 1500	8464000	799	50,1037
4	1500; 1500	8462000	241	49,9623
5	1498; 1499	8473000	552	52,5798
6	1500; 1499	8501000	132	52,5979
7	1499,5; 1500	8487000	984	55,9378
8	1498,5; 1499,5	8446000	855	51,6626
9	1499,5; 1497,5	8509000	717	52,7623
10	1500; 1499,5	8483000	183	51,4401
Rata-rata		8477800	602,8	52,12958

Pop = 100

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1500; 1499	8499000	187	65,5203
2	1499; 1499	8498000	164	79,6139
3	1500; 1500	8502000	293	69,1466
4	1500; 1499	8491000	370	68,0386
5	1500; 1500	8472000	307	69,6045
6	1500; 1500	8470000	792	57,8609
7	1500; 1500	8465000	536	57,1125
8	1499,5; 1500	8465000	562	56,9796
9	1500; 1499,5	8498000	733	61,7836
10	1500; 1499	8511000	486	71,7915
Rata-rata		8487100	443	65,7452

Pop = 200

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1498; 1498,5	8503000	733	84,6384
2	1498,5; 1500	8509000	217	91,1741
3	1500; 1500	8487000	822	91,8622
4	1499,5; 1500	8491000	359	101,8312
5	1500; 1500	8493000	416	85,6746
6	1500; 1499,5	8504000	909	106,8593
7	1500; 1500	8512000	331	93,9763
8	1500; 1500	8496000	329	92,2128
9	1500; 1500	8502000	118	94,057
10	1499,5; 1500	8490000	837	91,4029
Rata-rata		8498700	507,1	93,36888

Pop = 500

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1499,5; 1500	8505000	218	176,2898
2	1499,5; 1499,5	8508000	282	211,4839
3	1499; 1500	8511000	439	191,973
4	1500; 1500	8496000	336	148,363
5	1500; 1500	8512000	345	165,271
6	1500; 1500	8510000	123	172,9395
7	1500; 1500	8480000	364	141,1383
8	1500; 1500	8482000	695	204,9416
9	1500; 1500	8506000	190	185,1276
10	1500; 1499,5	8510000	48	167,3788
Rata-rata		8502000	304	176,49065

C.4 Uji Parameter Maks. Iterasi Data 1

Iter = 1000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 599,5	3553500	22	131,6055
2	600; 600	3553000	24	122,0115
3	600; 600	3555000	920	117,2992
4	600; 600	3554000	19	140,366
5	600; 600	3554000	80	130,9253
6	600; 600	3554000	167	122,5864
7	600; 600	3554000	361	122,3286
8	600; 600	3554000	11	126,1808
9	600; 600	3554000	203	130,8874
10	600; 600	3554000	262	117,1137
Rata-rata		3553950	206,9	126,13044

Iter = 2000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 600	3554000	29	226,3736
2	600; 600	3554000	50	251,832
3	600; 600	3554000	66	259,4104
4	600; 600	3554000	1566	213,139
5	600; 600	3554000	59	252,211
6	600; 600	3555000	1156	233,5015
7	600; 600	3554000	194	268,81
8	600; 600	3554000	242	260,0963
9	600; 600	3554000	47	211,1638
10	600; 600	3555000	608	266,3056
Rata-rata		3554200	401,7	244,28432

Iter = 5000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	600; 600	3554000	12	680,4361
2	600; 600	3554000	29	556,7375
3	599,6; 598,5	3553000	14	485,1591
4	600; 600	3554000	15	521,1517
5	600; 599,5	3553500	43	530,1425
6	600; 600	3555000	258	566,1789
7	600; 600	3554000	36	495,4448
8	598,5; 599,5	3553000	106	514,0701
9	600; 600	3554000	62	548,229
10	600; 600	3554000	202	537,92
Rata-rata		3553850	77,7	543,54697

C.5 Uji Parameter Maks. Iterasi Data 2

Iter = 1000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 750	7524300	501	124,9798
2	750; 750	7530950	900	127,1237
3	750; 750	7535300	958	120,888
4	750; 749,5	7529850	806	129,8563
5	750; 749,5	7535900	486	128,2009
6	750; 749,5	7529950	271	115,4439
7	750; 750	7527850	934	129,6277
8	750; 749,5	7533400	276	123,1218
9	750; 748,5	7531150	683	140,6484
10	750; 749,5	7538050	475	137,3704
Rata-rata		7531670	629	127,72609

Iter = 2000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 749,5	7538050	117	230,188
2	749,5; 750	7536650	743	255,2630
3	750; 750	7532700	1730	249,257
4	749,5; 750	7533550	1144	242,757
5	750; 749,5	7535550	450	230,9508
6	749,5; 749,5	7529550	1474	251,647
7	750; 750	7537400	753	249,5796
8	750; 750	7527900	103	229,8209
9	749,5; 750	7528150	269	253,9033
10	750; 749,5	7522500	1393	238,5118
Rata-rata		7532200	817,6	243,18784

Iter = 5000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	750; 750	7537400	1112	562,8714
2	750; 750	7533900	693	567,2149
3	749,5; 750	7537200	818	594,7331
4	750; 750	7533450	547	575,2325
5	749,5; 750	7534900	349	628,0032
6	750; 750	7528900	527	623,4069
7	749,5; 749,5	7529700	406	604,5437
8	749,5; 750	7535550	738	560,2899
9	749,5; 750	7538050	867	584,3988
10	750; 750	7526400	1236	546,1917
Rata-rata		7533545	729,3	584,68861

C.6 Uji Parameter Maks. Iterasi Data 3

Iter = 1000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1499,5; 1500	8505000	218	176,2898
2	1499,5; 1499,5	8508000	282	211,4839
3	1499; 1500	8511000	439	191,973
4	1500; 1500	8496000	336	148,363
5	1500; 1500	8512000	345	165,271
6	1500; 1500	8510000	123	172,9395
7	1500; 1500	8480000	364	141,1383
8	1500; 1500	8482000	695	204,9416
9	1500; 1500	8506000	190	185,1276
10	1500; 1499,5	8510000	48	167,3788
Rata-rata		8502000	304	176,49065

Iter = 2000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1500; 1500	8500000	187	305,9164
2	1500; 1500	8512000	153	339,0552
3	1500; 1500	8479000	472	262,5075
4	1500; 1500	8494000	191	253,0311
5	1500; 1500	8501000	996	357,2566
6	1500; 1500	8505000	87	256,61
7	1500; 1500	8512000	979	313,0238
8	1500; 1500	8497000	165	249,9907
9	1500; 1500	8507000	328	272,8505
10	1500; 1500	8500000	1757	301,8189
Rata-rata		8500700	531,5	291,20607

Iter = 5000

No	Berat	Profit	Iterasi Konvergen	Waktu Komputasi
1	1500; 1500	8500000	393	629,5346
2	1500; 1500	8512000	1126	812,7586
3	1499,5; 1500	8502000	439	747,2052
4	1500; 1500	8507000	2272	742,6542
5	1500; 1499,5	8510000	43	697,5153
6	1499,5; 1500	8502000	320	757,4251
7	1500; 1500	8499000	608	706,3443
8	1500; 1500	8503000	126	911,3284
9	1500; 1500	8508000	459	821,9901
10	1500; 1500	8483000	1453	800,4602
Rata-rata		8502600	723,9	762,7216

Lampiran D. Skrip Program CAPSO

D.1 Skrip Utama

```
tic;
%get parameter
Npop=str2num(get(handles.editNpop,'string'));
Imax=str2num(get(handles.editImax,'string'));
Cap=str2num(get(handles.editCap,'string'));
Nknapsack=length(Cap);
%get data
data001 = get(handles.uitable1,'UserData');
data=data001{1};
NamaBarang=data001{2};
Nbar=size(data,1);
dim=Nknapsack*Nbar;
%inisialisasi
for i=1:Npop
    Xswarm(i,:)=rand(1,dim);
    [Xswarm(i,:),Yswarm(i,:),Berat(i,:),Profit(i,:)]=
        discreteswarm(Xswarm(i,:),data,Nknapsack,Cap);
    Fitness(i)=sum(Profit(i,:));
end
Vswarm=ones(Npop,dim);
%Pi & Pg awal
Piswarm=Xswarm;
Pidis=Yswarm;
PiW=Berat;
PiP=Profit;
PiFit=Fitness;
best=find(PiFit==max(PiFit));
Pgswarm=Piswarm(best(1),:);
Pgdis=Pidis(best(1),:);
PgW=PiW(best(1),:);
PgP=PiP(best(1),:);
PgFit=max(PiFit);
grafik(1)=PgFit;
inon=0;
%iterasi
for iter=1:Imax
    %Hitung ei(t)
    if max(Fitness)~=min(Fitness)
        ei=(Fitness-min(Fitness))/(sum(Fitness)/Npop-
            min(Fitness));
    else
        ei=zeros(1,Npop);
    end
    %Hitung ai(t), Ai(t) &Ei(t)
    if max(Fitness)~=min(Fitness)
        Ei=ei/sum(ei);
    else
        Ei=ei;
    end
    ai=rand(Npop,dim).*(Piswarm-
        Xswarm)+rand(Npop,dim).*(repmat(Pgswarm,Npop,1)-Xswarm);
```

```
    Ai=repmat(Ei',1,dim).*rand(Npop,dim).*(Piswarm-
        repmat(((max(Xswarm)-min(Xswarm))/2+min(Xswarm)),Npop,1)-
        Xswarm);%median position -> titik tengah
    %Update Kecepatan
    Vswarm=Vswarm+ai+Ai;
    %Update Posisi
    Xswarm=Xswarm+Vswarm;
    for i=1:Npop
        if min(Xswarm(i,:))<0
            Xswarm(i,:)=(Xswarm(i,:)-min(Xswarm(i,:)))/
               (max(Xswarm(i,:))-min(Xswarm(i,:)));
        elseif max(Xswarm(i,:))>1
            Xswarm(i,:)=Xswarm(i,:)/max(Xswarm(i,:));
        end
        [Xswarm(i,:),Yswarm(i,:),Berat(i,:),Profit(i,:)]=
            discreteswarm(Xswarm(i,:),data,Nknapsack,Cap);
        Fitness(i)=sum(Profit(i,:));
    end
    %Update Pi
    dfFit=Fitness-PiFit;
    imP=find(dfFit>0);
    if ~isempty(imP)
        Piswarm(imP,:)=Xswarm(imP,:);
        Pidis(imP,:)=Yswarm(imP,:);
        PiW(imP,:)=Berat(imP,:);
        PiP(imP,:)=Profit(imP,:);
        PiFit(imP)=Fitness(imP);
    end
    %Update Pg
    best=find(PiFit==max(PiFit));
    Pgswarm=Piswarm(best(1),:);
    Pgdis=Pidis(best(1),:);
    PgW=PiW(best(1),:);
    PgP=PiP(best(1),:);
    PgFit=max(PiFit);
    %plot
    grafik(iter+1)=PgFit;
    if grafik(iter+1)~=grafik(iter)
        inon=iter;
    end
    plot(0:length(grafik)-1,grafik,'LineWidth',2);
    line(inon,grafik(inon+1),'Marker','s','MarkerEdgeColor',
       'k','MarkerFaceColor','y','MarkerSize',5);
    if inon<iter*0.75
        text(inon,grafik(inon+1),sprintf(['\n\n Iter: '
           num2str(inon) '\n Fitness: ' sprintf('%12d',
           grafik(inon+1))]));
    else
        text(inon,grafik(inon+1),sprintf([' Iter: ' num2str(inon)
          '\n Fitness: ' sprintf('%12d',grafik(inon+1))
          '\n\n']),'HorizontalAlignment','Right');
    end
    ylim([min(grafik) max(grafik)*1.005]);
    xlabel('Iterasi'); ylabel('Total Profit (Fitness)');
    pause(0.00001);
end
```

```
for i=1:Nknapsack
    rname{i}=['Knapsack ' num2str(i)];
end
set(handles.uitable2,'data',reshape(Pgdis,Nbar,Nknapsack)',
    'UserData',reshape(Pgdis,Nbar,Nknapsack)',...
    'columnname',NamaBarang,'rowname',char(rname));
set(handles.uitable3,'data',[PgW',PgP'],'UserData',[PgW',PgP'],...
    'columnname',{'Berat','Profit'},'rowname',char(rname));
set(handles.textWK,'string',[num2str(toc) ' detik']);
set(handles.axes1,'UserData',inon);
```

D.2 Skrip Diskritisasi

```
function
[Xswarm,Yswarm,Wi,Pi]=discreteswarm(Xswarm,data,Nknapsack,Cap)
dim=length(Xswarm);
Nbar=size(data,1);
Knapsack=reshape(Xswarm,Nbar,Nknapsack)';
Knapdis=round(Knapsack.*repmat(data(:,1)',Nknapsack,1));
sumk=sum(Knapdis);
for i=1:Nbar
    while sumk(i)>data(i,1)
        i1=find(Knapdis(:,i)>0);
        ps=ceil(rand*length(i1));
        Knapsack(i1(ps),i)=Knapsack(i1(ps),i)-1/data(i,1);
        if Knapsack(i1(ps),i)<0
            Knapsack(i1(ps),i)=0.5*rand/data(i,1);
        end
        Knapdis(i1(ps),i)=round(Knapsack(i1(ps),i)*data(i,1));
        sumk(i)=sum(Knapdis(:,i));
    end
end
%cek berat
for i=1:Nknapsack
    Wi(i)=hitungberat(Knapdis(i,:),data);
    while Wi(i)>Cap(i)
        d1=find(Knapdis(i,:)>0);
        d=ceil(rand*length(d1));
        Knapsack(i,d1(d))=Knapsack(i,d1(d))-1/data(d1(d),1);
        if Knapsack(i,d1(d))<0
            Knapsack(i,d1(d))=0.5*rand/data(d1(d),1);
        end
        Knapdis(i,d1(d))=round(Knapsack(i,d1(d))*data(d1(d),1));
        Wi(i)=hitungberat(Knapdis(i,:),data);
    end
    Pi(i)=hitungprofit(Knapdis(i,:),data);
end
Xswarm=reshape(Knapsack',1,dim);
Yswarm=reshape(Knapdis',1,dim);
```

D.3 Skrip Hitung Berat

```
function Wi=hitungberat(Knapdis,data)
Wi=sum(Knapdis.*data(:,2)');
```

D.4 Skrip Hitung Profit

```
function Pi=hitungprofit(Knapdis,data)
Pi=sum(Knapdis.*data(:,3)');
```