PENERAPAN *ANT LION OPTIMIZATION ALGORITHM* PADA PERMASALAHAN *KNAPSACK* 0-1

SKRIPSI

Oleh
Wangen Citro
NIM 151810101030

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

PENERAPAN *ANT LION OPTIMIZATION ALGORITHM* PADA PERMASALAHAN *KNAPSACK* 0-1

SKRIPSI

Diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Program Studi Matematika (S1) dan mencapai gelar Sarjana Sains

Oleh
Wangen Citro
NIM 151810101030

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

PERSEMBAHAN

Alhamdulillah wasyukurillah, dengan segalan puji bagi Allah SWT karena dengan limpahan rahmat dan hidayahnya saya dapat menyelesaikan skripsi ini dengan lancar. Skripsi ini saya persembahkan untuk:

1. Kedua orang tua saya, Ibunda Mualimah dan Ayahanda Mulyono tercinta yang telah memberikan dukungan, semangat, kasih sayang, perhatian, dan pengorbanan yang begitu besar serta doa yang tak pernah putus untuk anaknya;
2. Adik terkasih Jawot Kanigoro yang telah memeberikan semangat dan do'a;
3. Guru-guru sejak dari sekolah dasar sampai dengan perguruan tinggi yang telah mendidik dan membimbing selama ini;
4. Mohammad Rizky Adhe Nugraha yang senantiasa menemani, memberi dukungan, dan do'a;
5. Lailatul Febriana Nilasari sebagi patner riset yang selalu mendukung satu sama lain;
6. Almamater tercinta Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
7. Seluruh teman-teman "SIGMA" 2015 dan "UKM SPORA" yang telah memberikan motivasi serta dukungannya selama ini.

MOTTO

Barang siapa beriman kepada Allah dan hari akhir, maka hendaklah ia berkata baik atau diam. *)

Bagian terbaik dari hidup seseorang adalah perbuatan-perbuatan baiknya dan kasihnya yang tidak diketahui orang lain. **)

*) Nabi Muhammad SAW
**) William Wordsworth

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Wangen Citro

NIM : 151810101030

menyatakan dengan sesungguhnya bahwa skripsi yang berjudul ”Penerapan *Ant lion Optimization Algorithm* (ALO) Pada Permasalahan *Knapsack* 0-1” adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun, 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 mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 2020

Yang menyatakan,

Wangen Citro

NIM 151810101030

SKRIPSI

PENERAPAN *ANT LION OPTIMIZATION ALGORITHM* (ALO) PADA PERMASALAHAN *KNAPSACK* 0-1

Oleh

Wangen Citro
NIM 151810101030

Pembimbing

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

Dosen Pembimbing Anggota : Dr. Kiswara Agung Santoso, S.Si.,M.Kom.

PENGESAHAN

Skripsi berjudul " Penerapan *Ant lion Optimization Algorithm* (ALO) Pada Permasalahan *Knapsack* 0-1" telah diuji dan disahkan pada:

hari, tanggal :
tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Jember.

Tim Penguji:

Ketua,	Anggota 1,
Abduh Riski, S.Si.,M.Si.	Dr. Kiswara Agung Santoso,S.Si.,M.Kom.
NIP. 199004062015041001	NIP. 19720971998031003
Anggota II,	Anggota III,
Kosala Dwidja Purnomo, S.Si.,M.Si.	Prof. Drs. I Made Tirta, M.Sc., Ph.D.
NIP. 196908281998021001	NIP.195912201985031002

Mengesahkan,
Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Jember

Drs. Achmad Sjaifullah, M.Sc., Ph.D.
NIP. 195910091986021001

RINGKASAN

Penerapan *Ant lion Optimization Algorithm* (ALO) Pada Permasalahan *Knapsack* 0-1; Wangen Citro; 151810101030; 2020; 67 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Permasalahan *knapsack* merupakan permasalahan yang sering dihadapi oleh perusahaan dalam pengiriman dan pengelolaan barang. Kendala dalam pengiriman dan pengelolaan barang bisa saja terjadi ketika mencari solusi optimal untuk mengangkut/mengirim barang. Oleh karena itu dibutuhkan suatu metode untuk mencari solusi optimal dalam permasalahan *knapsack*. Salah satu permasalahan *knapsack* yaitu *knapsack* 0-1, dimana pada permasalahannya adalah penyimpanan barang akan dimasukkan secara utuh atau tidak sama sekali dengan kendala berat.

Masalah *knapsack* dapat diselesaikan dengan algoritma *metaheuristik*. Algoritma *Ant ion* merupakan algoritma *metaheuristik*. Sehingga, penelitian ini menyelesaikan tentang Penerapan *Ant lion Optimization Algorithm* (ALO) pada permasalahan *knapsack* 0-1. Data yang digunakan dalam penelitian ini merupakan data primer berupa data penjualan dari rumah produksi Kerajinan Bambu Hitam di Desa Pujerbaru, Kecamatan Maesan, Kabupaten Bondowoso. Rumah produksi Kerajinan Bambu Hitam ini memproduksi kerajinan dari bambu hitam. Data yang diambil yaitu berupa nama barang, jumlah barang, berat barang, modal, dan harga jual.

Algoritma *Ant lion* memiliki tiga parameter, yaitu parameter populasi (*Npop*), maksimal iterasi (*T*), dan ω. Parameter tersebut digunakan untuk mengetahui hasil dari algoritma *Ant lion* berupa solusi optimal dari beberapa percobaan yang dilakukan. Hasil dari algoritma *Ant lion* selanjutnya akan dibandingkan dengan hasil yang telah dihitung menggunakan metode *Simplex*.

Hasil simulasi akhir data dengan persentase deviasi yang menggunakan nilai dari masing-masing parameter. Nilai deviasi 0% menunjukkan bahwa tidak ada penyimpangan/hasil sama dengan nilai *simpleks*, begitupula ada beberapa nilai yang

mendekai hasil dari *Simpleks*/memiliki nilai penyimpangan yang kecil, yang artinya algoritma *Ant lion* baik digunakan untuk meyelesaikan permasalahan optimasi. Profit metode *Simpleks* Rp 12.610.000,-. Profit terbaik dari algoritma *Ant lion* terdapat pada nilai parameter *N*=30, ω=1, dan *T*=500 yang memiliki hasil sama dengan *Simpleks* yaitu Rp12.610.000,- dengan rata-rata persentase deviasi sebesar 0%.

Hasil dari *Ant lion Optimization Algorithm* (ALO) dibandingkan dengan hasil *Simpleks* dapat dikatakan bahwa algoritma *Ant lion* efektif untuk meyelesaikan permasalahaan optimasi. Kesimpulan ini berdasarkan dari hasil perhitungan yang dibandingkan dengan *Simpleks*, yang menunjukkan ada beberapa nilai yang sama dengan hasil perhitungan *Simpleks* dan ada beberapa perhitungan yang dibandingkan dengan *Simpleks* hasilnya mendekati dari hasil *Simpleks*. Solusi optimal yang didapatkan dari data kerajinan yaitu memiliki keuntungan Rp12.610.000,- dengan total berat 389,5 kg dan memiliki waktu komputasi 7,81228 detik. Nilai tersebut dijadikan sebagai hasil terbaik karena memiliki nilai iterasi yang cepat dengan profit yang optimum.

PRAKATA

Puji syukur ke hadirat Allah SWT, atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul " Penerapan *Ant lion Optimization Algorithm* (ALO) Pada Permasalahan *Knapsack* 0-1". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Bapak Abduh Riski, S.Si.,M.Si. selaku Dosen Pembimbing Utama dan Bapak Dr. Kiswara Agung Santoso, S.Si.,M.Kom. selaku Dosen Pembimbing Anggota yang telah membimbing dan memberikan arahan kepada penulis;
2. Bapak Kosala Dwidja Purnomo, S,Si.,M.Si. dan Bapak Prof. Drs. I Made Tirta, M.Sc., Ph.D. selaku Dosen Penguji yang telah memberikan masukan, saran dan kritik yang membangun dalam penyusunan skripsi ini;
3. Dian Anggraeni, S,Si., M.Si., selaku Dosen Pembimbing Akademik yang telah membimbing dalam pemilihan matakuliah;
4. Seluruh dosen dan karyawan Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember yang telah memberikan banyak ilmu dan pengalaman selama perkuliahan;
5. Seluruh kakak tingkat dan adik tingkat yang telah memberikan bantuan, dukungan dan pengalaman selama perkuliahan;
6. Almamater tercinta Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, SMAN 1 Purwoharjo, SMPN 2 Bangorejo, SDN 2 Kradenan, dan TK Pertiwi.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Persoalan transportasi merupakan kasus khusus dari optimisasi. Ada banyak masalah di bidang transportasi yang terkait dengan optimisasi. Salah satu permasalahan di bidang transportasi yang muncul adalah bagaimana suatu perusahaan mengatur produk apa yang harus di angkut agar memperoleh keuntungan yang maksimal, sementara perusahaan sendiri memiliki problematika/kendala yaitu kapasitas angkut dari kendaraan yang sangat terbatas.

Permasalahan *Knapsack* merupakan permasalahan yang sering dihadapi oleh perusahaan dalam pengiriman dan pengelolaan barang. Masalah pengiriman barang bisa memiliki karateristik yang berbeda-beda. Barang yang akan dimasukkan dalam media penyimpanan masing-masing memiliki berat dan profit atau nilai yang digunakan untuk menentukan prioritasnya dalam pemilihan tersebut. Kendala dalam pengiriman dan pengelolaan barang bisa saja terjadi ketika mencari solusi optimal, hal ini terjadi karena tidak adanya suatu metode untuk mencari solusi optimal. Oleh karena itu dibutuhkan suatu metode untuk mencari solusi optimal dalam permasalahan *Knapsack.*

Suyanto (2010) menjelaskan bahwa permasalahan *knapsack* dibagi menjadi tiga jenis, yaitu 0-1 *knapsack problem, bounded knapsack problem* dan *unbounded knapsack problem.* Pembagian tersebut didasarkan atas pola penyimpanan barang dengan nilai dan bobot yang bervariasi. Pada penelitian ini penulis ingin membahas tentang masalah *knapsack* 0-1, dimana pada permasalahan *knapsack* 0-1 adalah permasalahan penyimpanan barang dimana barang akan dimasukkan secara utuh atau tidak sama sekali.

Masalah *knapsack* dapat diselesaikan menggunakan algoritma *metaheuristik*. *Metaheuristik* adalah metode optimasi dengan pendekatan dimana metode ini didasarankan dari kecerdasan buatan dengan pembangkitan bilangan *random* pada solusi awalnya. *Metaheuristik* sebagian besar berasal dari tingkah laku hewan di alam yang kemudian dijadikan metode yang mampu menyelesaikan

permasalahan optimasi dengan cepat dan efisien. Metode *Metaheuristik* dapat menyelesaikan permasalahan kompleks dalam penerapannya. Metode-metode metaheuristik yang dapat digunakan diantaranya, algoritma *Particle Swarm Optimization* (PSO), *Genetic Algorithm* (GA), algoritma *Ant Colony Optimization* (ACO), dan *Firefly Algorihm* (FA). Beberapa algoritma metaheuristikk baru yang terispirasi dengan tingah laku hewan lainnya, diantaranya algoritma *Ant lion Optimization* (ALO), dan *Whale Optiization Algorithm* (WOA).

Mirjalili (2015) mengungkapkan bahwa penelitiannya ini mengusulkan algoritma baru yang diilhami perilaku berburu hewan di alam yang disebut *Ant Lion Optimizer (ALO).* Algoritma ALO meniru mekanisme perburuan *Ant lion* di alam. Lima langkah utama mangsa berburu seperti berjalan semut secara acak, membangun perangkap, menjebak semut dalam perangkap, menangkap mangsa, dan membangun kembali perangkap. Hasil dari penelitian membuktikan bahwa algoritma yang diusulkan mampu memberikan hasil yang sangat kompetitif dalam hal peningkatan eksplorasi, penghindaran optimal lokal, eksploitasi, dan konvergensi. Algoritma ALO juga menunjukkan bahwa algoritma ini memiliki manfaat dalam memecahkan masalah yang dibatasi dengan ruang pencarian yang beragam.

Rukmana (2015) menggunakan dua algoritma yaitu algoritma genetika dan algoritma *harmony search*. Berdasarkan penelitiannya mengenai *knapsack* 0-1, algoritma genetika mencapai keuntungan lebih besar, sehingga dapat disimpulkan bahwa algoritma genetika lebih baik daripada algoritma *harmony search.* Berdasarkan penelitian terdahulu dapat disimpulkan bahwa algoritma *metaheuristik* baik digunakan untuk menyelesaikan permasalahan *knapsack.*

Penelitian ini dilakukan untuk memperoleh nilai maksimum yang dipengaruhi oleh kapasitas muat barang. Berdasarkan penjelasan diatas mengenai permasalahan *knapsack* 0-1, penulis tertarik untuk menggunakan algoritma *Ant lion* yang bersifat *metaheuristik*, karena pada aplikasinya algoritma *metaheuristik* biasanya digunakan untuk pergerakan acak dan pencarian lokal. Pergerakan acak memberikan jalan yang baik untuk beralih dari pencarian lokal dengan pencarian global. Sehingga algoritma *Ant lion* akan diterapkan ke dalam permasalahan *knapsack* 0-1, dengan

harapan algoritma tersebut menghasilkan solusi yang lebih optimal dari algoritma sebelumnya.

1.1 Rumusan Masalah

Permasalahan yang akan dibahas pada penelitian ini berdasarkan latar belakang adalah bagaimana mencari solusi optimal dengan memaksimalkan keuntungan dengan berat barang yang terbatas pada permasalahan *knapsack* 0-1 dengan menerapkan algoritma *Ant lion.*

1.2 Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah mencari solusi optimal yang memaksimalkan keuntungan dengan berat barang yang terbatas pada permasalahan *knapsack* 0-1 dengan menerapkan algoritma *Ant lion.*

1.3 Manfaat

a. Manfaat dari penelitian ini diharapkan dapat memberikan solusi optimal ketika memaksimalkan keuntungan dengan berat barang yang terbatas pada algoritma *Ant lion* dalam permasalahan *knapsack* 0-1.
b. Dapat dijadikan referensi dalam penelitian selanjutnya mengenai algoritma *Ant lion optimization* (ALO).

BAB 2. TINJAUAN PUSTAKA

2.1 Optimasi

Optimasi adalah salah satu ilmu matematika yang fokus untuk mendapatkan nilai minimum atau maksimum secara sistematis dari suatu fungsi, peluang, maupun pencarian nilai lainnya dalam berbagai kasus. Untuk mendapatkan nilai yang optimal baik maksimum atau minimum, secara sistematis dilakukan pemilihan nilai variabel *integer* atau nyata yang akan memberikan solusi optimal (Hayyu, 2016).

Berikut ini adalah beberapa persoalan yang memerlukan optimasi yang sering muncul (Munir, 2005):

a. penentuan pemilihan barang pada masalah *knapsack*;
b. menentukan lintasan terpendek dari suatu tempat ke tempat yang lain;
c. menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimumkan dan hasil produksi tetap maksimal;
d. mengatur jalur kendaraan umum agar semua lokasi dapat dijangkau.

2.2 Permasalahan *Knapsack (Knapsack Problem)*

Permasalahan *Knapsack* merupakan suatu permasalahan bagaimana memilih objek dari sekian banyak objek dan beberapa besar objek tersebut akan disimpan sehingga diperoleh suatu penyimpanan yang optimal dengan memperhatikan objek yang terdiri dari n objek (1,2,3,....,n) dimana setiap objek memiliki bobot (w_i) dan nilai profit (p_i) dengan memperhatikan juga kapasitas dari media penyipanan sebesar (M).

Permasalahan *knapsack* ini dapat digunakan pada bidang jasa pengangkutan barang seperti pengangkutan peti kemas dalam sebuah media pengangkut. Dalam usaha tersebut, diinginkan keuntungan yang maksimal untuk mengangkut barang yang ada dengan tidak melebihi kapasitas yang ada. Berdasarkan persoalan tersebut, diharapkan ada suatu solusi yang secara otomatis dapat mengatasi

persoalan itu. *Knapsack* adalah permasalahan mengenai optimalisasi kombinasional dimana harus mencari solusi terbaik dari banyak kemungkinan yang dihasilkan (Dimyati, 2004).

Knapsack terdiri dari beberapa persoalan yaitu sebagai berikut (Martello, 2006).

a. *Knapsack* 0-1 *(Integer Knapsack)*

   Objek yang dimasukkan ke dalam media penyimpanan dimensinya harus dimasukkan semua atau tidak sama sekali.

b. *Knapsack* terbatas *(Bounded Knapsack)*

   Objek yang dimasukkan ke dalam media penyimpanan dimensinya bisa dimasukkan sebagian atau seluruhnya.

c. *Knapsack* tak terbatas (*Unbounded Knapsack*)

   Jumlah objek yang dimasukkan ke dalam media penyimpanan jumlahnya tidak terbatas.

Knapsack yang akan dibahas dalam skripsi ini adalah jenis *knapsack* 0-1 (*integer knapsack)*. Variabel keputusan yang diperoleh yaitu x_i bernilai 1 jika objek dipilih dan x_i bernilai 0 jika objek tidak dipilih.

Permasalahan *knapsack* bilangan bulat merupakan permasalahan program bilangan bulat (*integer)* yang memiliki satu kendala tunggal, sehingga pada modelnya ditambahkan batasan untuk variabel keputusan yang dihasilkan harus bernilai bulat (*integer*). Dalam masalah ini, diberikan n buah objek dan sebuah media penyimpanan yang memiliki daya tampung maksimal senilai *M*. Setiap benda memiliki berat (w_i) dengan nilai keuntungan/profit (p_i). Objektif dari permasalahan ini adalah bagaimana memilih objek-objek yang dimasukkan ke dalam media penyimpanan sehingga tidak melebihi kapasitas dari media penyimpanan namun memaksimalkan total keuntungan yang diperoleh. Pada persoalan *integer knapsack* barang yang diangkut dimensinya harus diangkut seluruhnya atau tidak sama sekali.

Permasalahan *integer knapsack* mempunyai solusi persoalan yang dinyatakan sebagai himpunan:

$$X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$$

Dalam hal ini, $x_i = 1$ jika benda ke-i dimasukkan ke dalam media penyimpanan, atau $x_i = 0$ jika benda ke-i tidak dimasukkan ke dalam media penyimpanan. Karena itulah persoalan ini dinamakan *knapsack* 0-1. Sebagai contoh, $X = \{1,0,0,1\}$ adalah sebuah solusi yang ditemukan, maka benda ke-1 dan ke-4 dimasukkan ke dalam media penyipanan, dan benda ke-2 dan ke-3 tidak dimasukkan ke dalam media penyimpanan.

Secara matematis, persoalan *knapsack* 0-1 dapat dirumuskan sebagai berikut:
Fungsi tujuan Maksimum

$$Z = \sum_{i=1}^{n} p_i \, x_i \qquad (2.1)$$

Dan fungsi kendala

$$\sum_{i=1}^{n} w_i \, x_i \leq M \qquad (2.2)$$

Dengan $x_i \in \{0,1\}$, untuk $i = 1,2,3,...,n$,

Dimana:

Z : nilai optimum dari fungsi tujuan,

n : banyak barang,

p_i : keuntungan barang, dengan $i = 1,2,3,...,n$,

w_i : berat (*weight*) barang, dengan $i = 1,2,3,...,n$,

M : kapasitas media penyimpanan (*knapsack*)

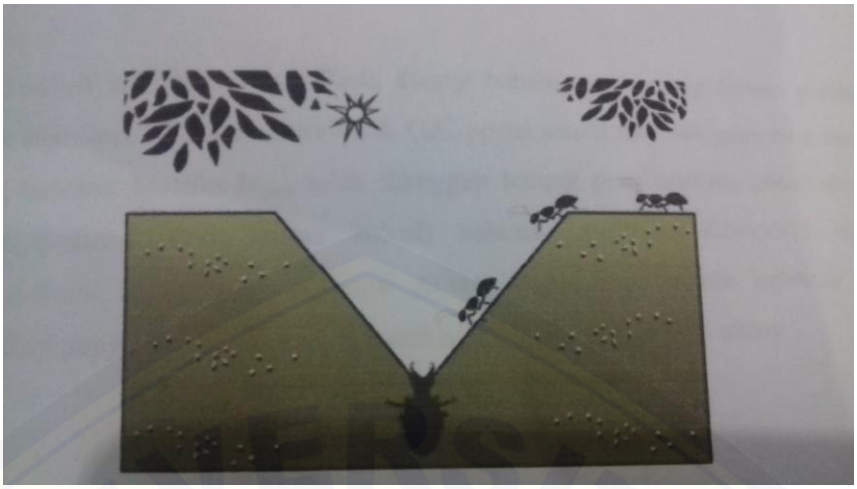
x_i : banyaknya barang jenis ke-i

2.3 *Ant Lion Optimization Algorithm* (ALO)

Ant Lion Optimization Algorithm (ALO) merupakan salah satu metode metaheuristik baru yang termasuk kedalam *Swarm Intelligent* yang termotivasi dari tingkah laku hewan *Ant lion* di alam. *Ant Lion Optimization Algorithm* (ALO) dikembangkan oleh Seyedali Mirjalili yang telah diterima pada tahun 2015 di Universitas Griffith, Natahan, Brisbane, Australia. *Ant lion* (*doodlebugs*) nama hewan ini berasal dari perilaku berburu yang unik dan mangsa favorit. Seekor larva

Ant lion menggali lubang berbentuk kerucut dipasir dengan memindahkan sepanjang jalan melingkar dan membuang pasir dengan rahangnya yang sangat besar. Setelah menggali perangkap, larva akan menyembunyikan dirinya pada bagian bawah kerucut, *Ant lion* sendiri termasuk ke dalam hewan predator duduk-dan-tunggu, dimana dirinya akan menunggu semut untuk terjebak di dalam lubang seperti gambar yang diilutrasikan pada Gambar 2.1 Tepi pada lubang kerucut cukup terjal, ini berguna untuk mangsa jatuh ke dalam bagian bawah perangkap dengan mudah. Setelah *Ant lion* menyadari bahwa mangsanya jatuh kebagian bawah perangkap, maka dirinya akan mencoba menangkap mangsanya. Namun pada dasarnya mangsa tidak akan semudah itu tertangkap dengan segera dan berusaha untuk melarikan diri dari perangkap. *Ant lion* cerdas akan membuang pasir keluar tepi lubang untuk meluncurkan mangsa ke bagian bawah lubang. Ketika mangsa tertangkap ke rahang *Ant lion*, kemudian mangsanya akan ditarik ke bawah tanah untuk dikonsumsi. Setelah mengkonsumsi mangsa, *Ant lion* membuang sisa-sisa makanan keluar sisi lubang dan mengubah lubang untuk berburu berikutnya.

Perilaku lain yang menarik tentang *Ant lion* yang telah diamati dalam hidupnya adalah relevansi dari ukuran perangkap dan dua hal diantaranya yaitu, tingkat kelaparan dan keadaan bulan. *Ant lion* akan cenderung menggali lubang perangkap yang lebih besar ketika sangat lapar atau pada saat bulan penuh (*Full Moon*). *Ant lion* telah berevolusi dan beradaptasi dengan cara ini untuk meningkatkan kesempatan untuk bertahan hidup. Hal ini juga telah ditemukan bahwa *Ant lion* tidak langsung mengamati bentuk bulan untuk memutuskan tentang ukuran perangkap, tetapi memiliki jam lunar internal untuk membuat keputusan kapan bulan penuh (*Full Moon*). Inspirasi utama *Ant Lion Optimization Algorithm* (ALO) berasal dari perilaku penuaan untuk larva *Ant lion* ini. *Ant Lion Optimization Algorithm* (ALO) merupakan algoritma optimasi yang dikembangkan untuk menyelesaikan masalah optimasi dengan menerapkan pembuatan perangkap dalam menjebak mangsa (semut). Lima langkah utama yang diterapkan dalam berburu mangsa seperti berjalan acak semut, membangun perangkap, jebakan semut di perangkap, menangkap mangsa, dan membangun kembali perangkap.



Gambar 2.1 Ilustrasi *Ant lion* sedang menunggu Semut
(Sumber : Mirjalili, 2015)

Ant Lion Optimization Algorithm (ALO) meniru interaksi antara *Ant lion* dan semut dalam perangkap. Model interaksinya, semut diminta untuk pidah ruang pencarian, dan *Ant lion* diperbolehkan untuk memburu semut dan menjadi lebih mudah menggunakan perangkap. Semut bergerak secara stokastik di alam ketika mencari makanan, berjalan-jalan secara acak dipilih untuk model semut sebagai berikut:

$$X(t) = [\,0\,, cumsum(2r(t_1) - 1), c\ \ldots, cumsum(2r(t_n) - 1)] \quad (2.3)$$

dimana $cumsum$ menghitung jumlah kumulatif, n adalah jumlah maksimum iterasi, t menunjukkan langkah dari jalan acak, dan $r(t)$ adalah fungsi stokastik didefinisikan sebagai berikut :

$$r(t) = \begin{cases} 1 & if\ rand > 0.5 \\ 0 & if\ rand \leqslant 0.5 \end{cases} \quad (2.4)$$

dimana t menunjukkan langkah dari jalan acak dan $rand$ adalah angka acak yang dihasilkan dengan distribusi seragam dalam interval $[0,1]$.

Posisi semut disimpan dan dimanfaatkan selama optimasi dalam matriks berikut :

$$M_{Ant=}\begin{bmatrix} A_{1,1} & A_{1,2} & \dots & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & \dots & A_{2,d} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & \dots & A_{n,d} \end{bmatrix} \quad (2.5)$$

dimana M_{ant} adalah matriks untuk menyimpan posisi setiap semut, $A_{i,j}$ menunjukkan nilai variabel ke-j (dimensi) dari-i semut, n adalah jumlah semut, dan d adalah jumlah variabel. Perlu dicatat bahwa semut mirip dengan partikel dalam PSO atau individu di GA, posisi semut merujuk parameter untuk solusi tertentu. Matriks M_{ant} telah dianggap untuk menyimpan posisi semua semut (variabel dari semua solusi) selama optimasi. Kemudian untuk mengevaluasi setiap semut, *fitness* (tujuan) fungsi digunakan selama optimasi dan mengikuti penyimpanan matriks berserta nilai *fitness* dari semua semut:

$$M_{OA=}\begin{bmatrix} f([A_{1,1} & A_{1,2} & \dots & \dots & A_{1,d}]) \\ f([A_{2,1} & A_{2,2} & \dots & \dots & A_{2,d}]) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ f([A_{n,1} & A_{n,2} & \dots & \dots & A_{n,d}]) \end{bmatrix} \quad (2.6)$$

dimana M_{OA} adalah matriks untuk menyimpan *fitness* setiap semut, $A_{i,j}$ menunjukkan nilai variabel dimensi ke-j dari-i semut, n adalah jumlah semut, dan f adalah fungsi tujuan. Selain untuk berburu semut, kita mengasumsikan *Ant lion* juga bersembunyi disuatu tempat dalam ruang pencarian, dalam rangka menyimpan posisi mereka dan nilia-nilai *fitness*, maka kemudian memanfaatkan matriks berikut:

$$M_{antlion=}\begin{bmatrix} AL_{1,1} & AL_{1,2} & \dots & \dots & AL_{1,d} \\ AL_{2,1} & AL_{2,2} & \dots & \dots & AL_{2,d} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ AL_{n,1} & AL_{n,2} & \dots & \dots & AL_{n,d} \end{bmatrix} \quad (2.7)$$

dimana $M_{antlion}$ adalah matriks untuk menyimpan posisi dari setiap *Ant lion*, $AL_{i,j}$ menunjukkan nilai variabel ke-j (dimensi) dari-i semut, n adalah jumlah semut, dan d adalah jumlah variabel (dimensi).

$$M_{OAL}=\begin{bmatrix} f([AL_{1,1} & AL_{1,2} & \dots & \dots & AL_{1,d}]) \\ f([AL_{2,1} & AL_{2,2} & \dots & \dots & AL_{2,d}]) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ f([AL_{n,1} & AL_{n,2} & \dots & \dots & AL_{n,d}]) \end{bmatrix} \quad (2.8)$$

dimana M_{OAL} adalah matriks untuk menyimpan *fitness* setiap *Ant lion*, $AL_{i,j}$ menunjukkan nilai variabel dimensi ke-j dari i-*Ant lion*, n adalah jumlah *Ant lion*, dan f adalah fungsi tujuan.

Beberapa kondisi berikut akan diterapkan selama proses optimisasi, diantara lain :

a. Semut bergerak disekitar ruang pencarian menggunakan jalan acak semut yang berbeda.
b. Jalan acak diterapkan untuk semua dimensi dari semut.
c. Jalan acak dipengaruhi oleh perangkap dari *Ant lion*.
d. *Ant lion* dapat membangun lubang perangkap sebanding dengan *fitness* mereka (*fitness* tinggi, maka lubang besar).
e. *Ant lion* dengan lubang-lubang yang besar memiliki probabilitas yang lebih tinggi untuk menangkap semut.
f. Setiap semut dapat ditangkap oleh *Ant lion* dalam setiap iterasi dan *Elite* (*Ant lion* yang memiliki keuntungan paling tinggi).
g. Jangkauan dari jalan acak menurun secara adaptif untuk mensimulasikan pergeseran semut menuju *Ant lion*.
h. Jika seekor semut menjadi lebih *fitness* (pantas) daripada *Ant lion*, ini berarti bahwa penangkapan dan penarikan di bawah tanah oleh *Ant lion*.
i. Seekor *Ant lion* mengatur ulang posisi dirinya untuk menangkap mangsa baru lagi dan membangun lubang untuk meningkatkan perubahan menangkap mangsa lain setelah masing – masing pemburuan.

2.3.1 Jalan acak dari semut (*Random walks of ants*)

Jalan acak atau *random walk* semua didasarkan pada (2.3) Semut memperbarui posisi mereka dengan berjalan acak disetiap langkah optimasi, karena setiap ruang pencarian memiliki batas (berbagai variabel). Namun, pada persamaan (2.2) tidak

dapat langsung digunakan untuk memperbarui posisi semut, dalam rangka untuk menjaga jalan acak tetap berada dalam ruang pencarian, mereka dinormalisasi menggunakan persamaan berikut (normalisasi min-maks) :

$$X_i^t = \frac{(X_i^t - a_i) \times (a_i - c_i^t)}{(b_i - a_i)} + c_i \quad (2.9)$$

dimana a_i adalah minimum jalan acak semut dari variabel ke$-i$, b_i adalah maksimum jalan acak semut pada variabel ke$-i$, c_i^t adalah minimum variabel ke$-i$ pada iterasi ke$-t$, d_i^t adalah maksimum variabel ke$-i$ pada iterasi ke$-t$. Persamaan (2.9) harus diterapkan pada setiap iterasi untuk menjamin terjadinya jalan acak (*random walk*) di dalam ruang pencarian.

2.3.2 Penangkapan pada lubang *Ant lion*

Sebagaimana dibahas di atas, jalan acak semut dipengaruhi oleh perangkap *Ant lion*, untuk Model matematis asumsi ini, persamaan berikut diberikan:

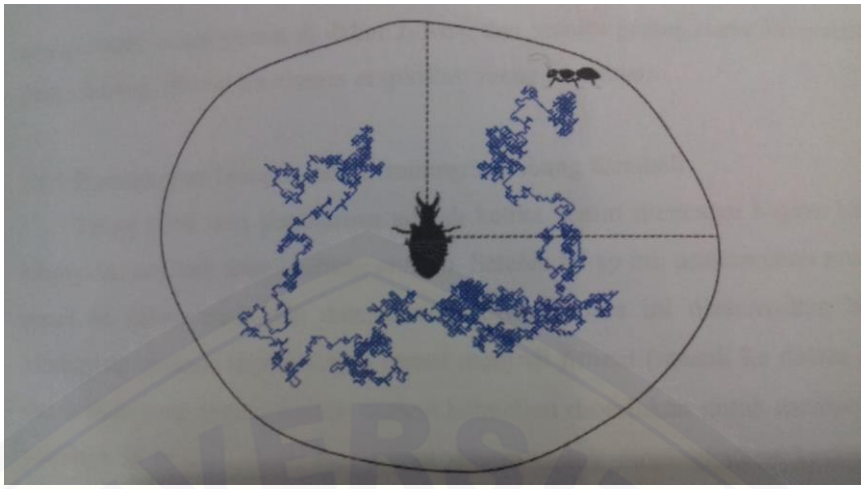
$$c_i^t = Antlion_j^t + c^t \quad (2.10)$$

$$d_i^t = Antlion_j^t + d^t \quad (2.11)$$

dimana c^t adalah minimum semua variabel pada iterasi dari$-t$, d^t mengindikasi sebuah vektor yang termasuk dalam maksimum semua variabel pada iterasi dari$-t$, c_i^t adalah minimum semua variabel ke$-i$ semut, d_i^t adalah maksimum semua variabel ke$-i$ semut, dan $Antlion_j^t$ menunjukkan posisi *Ant lion* dari$-j$ pada iterasi ke$-t$. Persamaan (2.10) dan (2.11) menunjukkan jalan semut secara acak pada hiperbola didefinisikan dengan vektor c dan d disekitar *Ant lion* yang dipilih.

2.3.3 Pembangunan perangkap

Roda rolet digunakan untuk memodelkan kemampuan berburu *Ant lion*. Seperti pada Gambar 2.2 menunjukkan semut diasumsikan terjebak dalam satu *Ant lion* yang dipilih. *Ant Lion Optimization Algorithm* (ALO) diperlukan untuk memanfaatkan operator roda rolet untuk memilih *Ant lion* berdasarkan pada *fitness* mereka selama optimasi. Mekanisme ini memberikan peluang yang tinggi terhadap *fitness Ant lion* untuk menangkap semut.



Gambar 2.2 Jalan Acak Semut di dalam Perangkap *Ant lion*

(Sumber : Mirjalili, 2015)

2.3.4 Pergeseran semut menuju *Ant lion*

Mekanisme yang diusulkan sejauh ini, *Ant lion* mampu membangun perangkap sebanding dengan *fitness* mereka dan semut yang diperlukan untuk bergerak secara acak. Namun, *Ant lion* membuang pasir keluar tepi lubang ketika mereka menyadari bahwa semut dalam perangkap. Perilaku ini untuk meluncurkan semut ke bagian bawah lubang agar terjebak dan tidak bisa melarikan diri. Pemodelan untuk perilaku ini secara matematis, jari-jari jalan acak hiperbola menurun secara adaptif. Persamaan berikut diberikan :

$$c^t = \frac{c^t}{I} \tag{2.12}$$

$$d^t = \frac{d^t}{I} \tag{2.13}$$

dimana I adalah rasio, c^t adalah minimum semua variabel pada iterasi ke$-t$, dan d^t mengindikasi vektor yang termasuk maksimum semua variabel pada iterasi ke$-t$. Pada persamaan (2.12) dan (2.13), $I = 10^{\omega}\frac{t}{T}$ dimana t adalah iterasi saat ini, T adalah jumlah maksimum iterasi, dan ω adalah sebuah konstanta didefinisikan berdasarkan pada iterasi saat ini. Pada dasarnya, ω konstan dapat menyesuaikan tingkat akurasi eksploitasi. Persamaan ini mengecilkan jari-jari memperbarui posisi semut di dalam lubang dan meniru proses semut bergeser ke dalam lubang. Hal ini menjamin eksploitasi ruang pencarian.

2.3.5 Penangkapan mangsa dan membangun lubang kembali

Tahap akhir dari pemburuan adalah ketika semut mencapai bagian bawah lubang dan terjebak dalam rahang *Ant lion*. Setelah tahap ini, *Ant lion* yang menarik semut ke dalam pasir dan mengkonsumsi mangsanya. Proses ini diasumsikan bahwa menangkap mangsa terjadi ketika semut menjadi *fitness* (masuk ke dalam pasir) dari *Ant lion* yang sesuai. Seekor *Ant lion* kemudian diperlukan untuk memperbarui posisinya ke posisi terbaru dari semut diburu untuk meningkatkan kesempatan menangkap mangsa baru. Persamaan berikut diberikan :

$$Antlion_j^t = Ant_i^t \quad if\ f(\mathrm{Ant}_i^t) > f\left(Antlion_j^t\right) \tag{2.14}$$

dimana t menunjukkan iterasi saat ini, $Antlion_j^t$ menunjukkan posisi seleksi *Ant lion* ke$-j$ pada iterasi t, dan Ant_i^t mengindikasi posisi semut ke$-i$ pada iterasi t.

2.3.6 *Elite*

Elite merupakan ciri penting dari evolusioner algoritma yang memungkinkan mereka mempertahankan solusi terbaik (s) yang diperoleh pada setiap tahap proses optimasi. *Ant lion* terbaik diperoleh sejauh ini di setiap iterasi disimpan dan dianggap sebagai *Elite*. *Elite* adalah *Ant lion* yang paling pantas itu harus dapat mempengaruhi pergerakan semua semut selama iterasi. Oleh karena itu, diasumsikan bahwa setiap jalan semut secara acak sekitar *Ant lion* yang dipilih oleh roda rolet dan *Elite* secara bersamaan sebagai berikut :

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{2.15}$$

dimana R_A^t adalah jalan acak disekitar *Ant lion* yang dipilih dengan roda rolet pada iterasi t, R_E^t adalah jalan acak disekitar *Elite* pada iterasi t, Ant_i^t mengindikasi posisi semut dari$-i$ pada iterasi t.

2.4 Metode *Simpleks*

Metode ini dikembangkan oleh George Dantzig pada 1946 dan sepertinya cocok untuk komputerisasi masa kini. Metode simpleks merupakan prosedur algoritma yang digunakan untuk pengambilan keputusan pada iterasi berikutnya.

Metode simpleks merupakan suatu penyelesaian masalah-masalah program linier yang meliputi banyak pertidaksamaan dan banyak variabel. Dalam menggunakan metode simpleks untuk menyelesaikan masalah-masalah program linier, model program linier harus diubah ke dalam suatu bentuk umum yang dinamakan "bentuk baku". Ciri-ciri bentuk baku model program linier adalah semua kendala berupa persamaan dengan sisi kanan nonnegatif, fungsi tujuan dapat memaksimumkan atau meminimumkan (Dantzig, 2002).

Ada beberapa hal yang harus diperhatikan dalam membuat bentuk baku, yaitu:

a. Fungsi kendala dengan pertidaksamaan ≤ dalam bentuk umum, diubah menjadi persamaan (=) dengan menambahkan satu variabel slack;
b. Fungsi kendala dengan pertidaksamaan ≥ dalam bentuk umum, diubah menjadi persamaan (=) dengan menggunakan satu variabel surplus;
c. Fungsi kendala dengan persamaan dalam bentuk umum, ditambahkan satu variabel artifisial (variabel buatan).

2.5 *Simpleks* pada *Solver Add-In* Microsoft Excel

Solver adalah program *add- in* yang berada di bawah program Excel. Program *solver* ini berisi banyak perintah yang berfungsi untuk melakukan analisis terhadap masalah optimasi. Jika kita menginstal Microsoft Excel, maka tidak secara otomatis *solver* ini terinstal, sehingga harus di instal secara khusus setelah program Excel terinstal pada komputer. Program *solver* ini cukup baik untuk menyelesaikan masalah optimasi (Dwijanto, 2008).

Menurut Ragsdale (2007) *Solver Parameters* adalah salah satu aplikasi MS *Excel* yang memfasilitasi penyelesaian masalah optimasi. Terdapat 3 jenis algoritma untuk menyelesaikan masalah optimasi pasa *Solver Parameter*, yaitu : *Standart GRG Nonlinear*, *Standard Simplex LP*, dan *Standard Evolutionary*.

Ada 3 komponen utama dari model *spredsheet* pada *Solver Parameters*, yaitu:

1. *Set* atau *Target Cell*

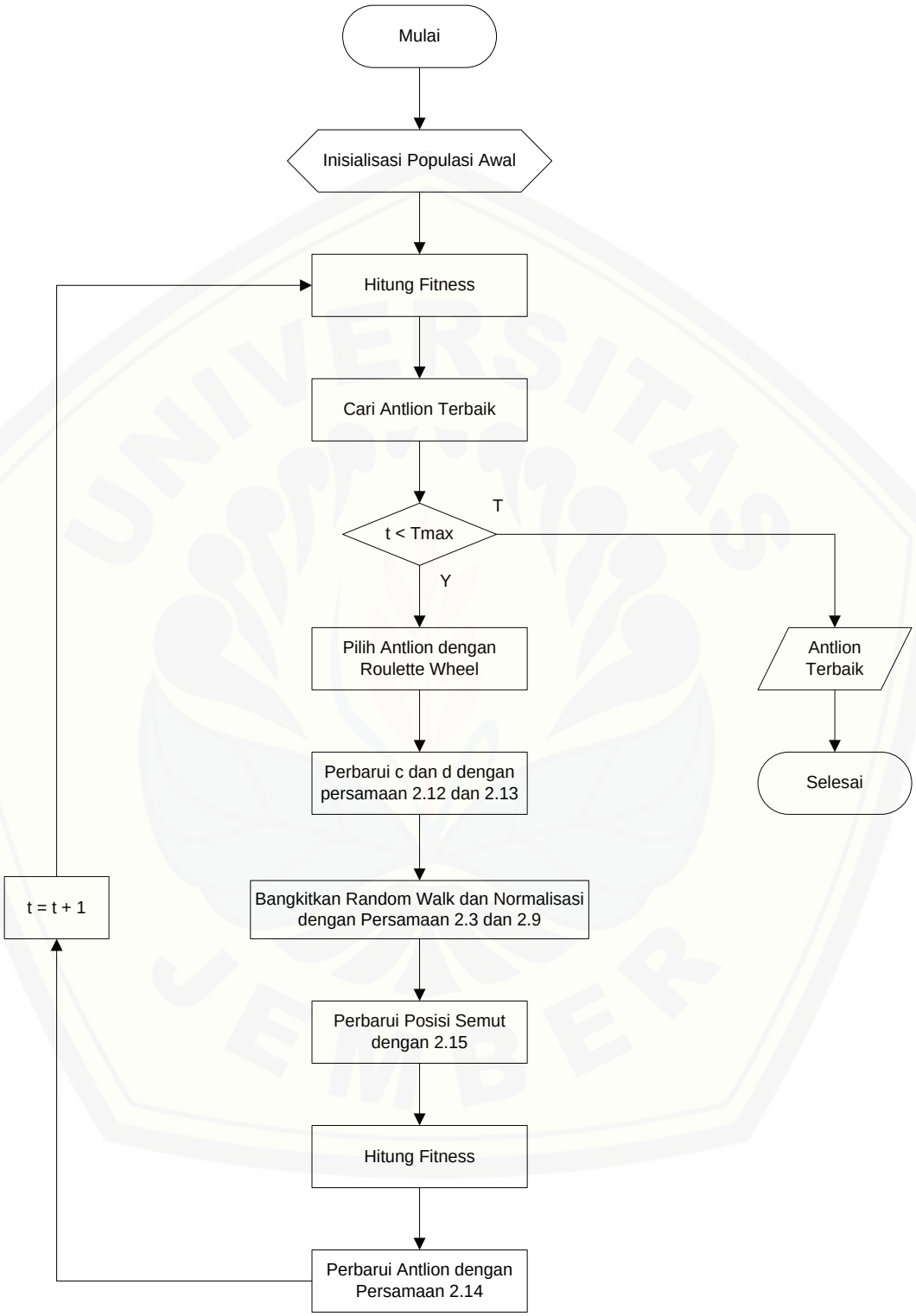
   Cell ini mewakili fungsi objektif pada model yang dapat berupa memaksimumkan atau meminimumkan.
2. *Variable* atau *Changing Cells*

Cells ini mewakili variabel-variabel keputusan pada model.

3. *Constraint Cells*

   Cells ini mewakili formulasi ruas kiri dari fungsi-fungsi kendala pada model beserta nilai limit bawah dan atasnya.

Langkah-langkah dari Algoritma *Ant lion Optimization* (ALO)



(Sumber : Mirjalili, 2015)

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang akan digunakan dalam penelitian ini adalah data penjualan dari rumah produksi kerajinan Bambu Hitam di Desa Pujerbaru, Kecamatan Maesan, Kabupaten Bondowoso dan data simulasi. Kerajinan Bambu Hitam memproduksi berbagai macam kerajinan, jenis kerajinan yang berasal dari bambu hitam yang kemudian dibentuk dan diukir sesuai yang di inginkan. Data yang diambil berupa modal, nama barang, berat benda, jumlah kerajinan, harga jual dan keuntungan.

3.2 Langkah-langkah Penelitian

Penelitian penerapan algoritma *Ant lion Optimization* (ALO) untuk menyelesaikan permasalahan *knapsack* 0-1 akan dilakukan dengan langkah-langkah sebagai berikut.

a. Studi Literatur

   Langkah pertama yang dilakukan yaitu dengan mencari dan mempelajari referensi yang berkaitan dengan permasalahan *knapsack,* algoritma *Ant lion Optimization* (ALO) pada jurnal internasional dari buku yang berkaitan dengan hal tersebut.

b. Mengambil Data

   Langkah kedua yaitu mengambil data dilakukan dengan mengambil data penjualan kerajinan Bambu Hitam di Desa Pujerbaru, Kecamatan Maesan, Kabupaten Bondowoso. Data yang diambil berupa data modal (Rp), berat barang (kg), nama barang, jumlah barang, dan harga jual (Rp).

c. Mengidentifikasi Data

   Langkah ketiga yaitu mengidentifikasi data dilakukan dengan memilih data penjualan kerajinan bambu hitam dengan permasalahan *knapsack* untuk mencari berat dan keuntungan dari masing-masing kerajinan.

d. Penerapan *Ant-Lion Optimization Algorithm* (ALO)

Menerapkan algoritma *Ant lion Optimization Algorithm* (ALO) pada data yang telah teridentifikasi dengan langkah-langkah berikut.

1) Inisialisasi populasi awal
2) Menghitung *fitness* dari *Ant lion*
3) Mencari *Ant lion* terbaik
4) Memilih *Ant lion* dengan *Roulatte Wheel*
5) Perbarui *Ant lion* dengan persamaan (2.12) dan (2.13)
6) Membangkitkan *Random Walk* dan Normalisasi dengan persamaan (2.3) dan (2.9)
7) Memperbarui posisi semut dengan persamaan (2.15)
8) Menghitung *fitness*
9) Memperbarui *Ant lion* dengan persamaan (2.14)
10) Cari *ant-lion* terbaik

e. Membuat Program

Membuat program sesuai dengan algoritma yang telah di tentukan pada langkah (d) menggunakan *software* Matlab R2015b lalu di simulasikan menggunakan menu *guide* pada Matlab R2015b.

f. Simulasi Program

Langkah keenam yaitu dengan melakukan simulasi program dari program yang telah dibuat pada langkah (e).

g. Analisis Hasil

Langkah ketujuh yaitu melakukan analisis hasil dari data yang telah dimasukkan ke dalam program tersebut. Kemudian untuk melihat keoptimalan hasil algoritma ALO, hasil algoritma juga akan dibandingkan dengan hasil

metode *simpleks* dengan menghitung persentase deviasi (*PD*) menggunakan persamaan (3.1). Perhitungan persentase deviasi dilakukan untuk memperoleh keyakinan bahwa nilai-nilai hasil prediksi cukup mewakili nilai pengisian data yang hilang, maka dihitung persentase deviasi dari nilai hasil prediksi. Jika nilai kesalahan 0 maka nilai prediksi sama dengan nilai *simpleks*, jika nilai kesalahan sangat kecil berarti nilai prediksi hampir sama dengan nilai *simpleks*. Sebaliknya jika nilai kesalahan sangat besar maka terjadi penyimpangan dari hasil sebenarnya (Vallada, 2011).

$$PD = \frac{Simpleks - Z}{Simpleks} x\ 100\% \qquad (3.1)$$

Dengan

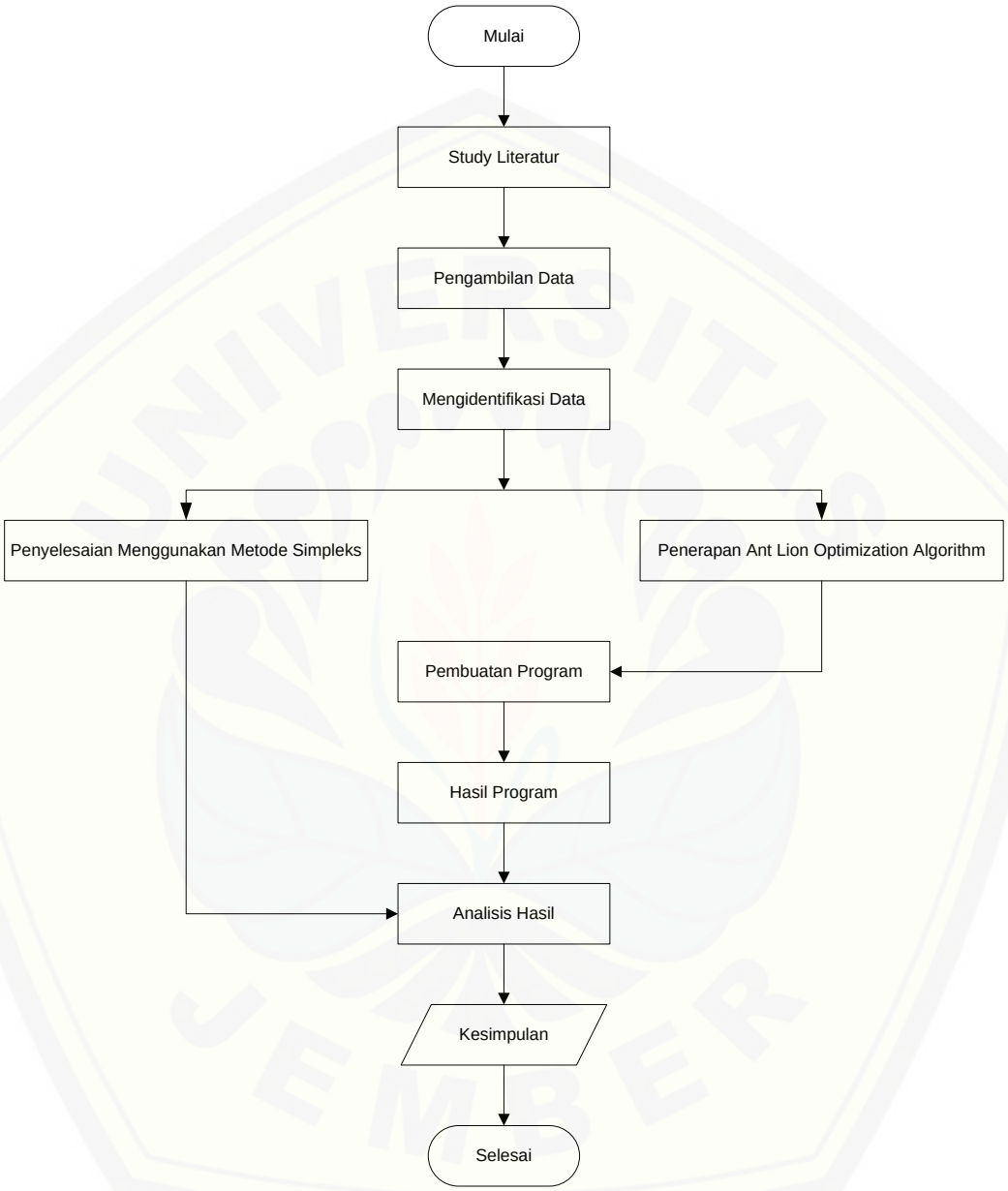
Simpleks : hasil metode *simpleks*

Z : hasil algoritma ALO

h. Membuat Kesimpulan

Membuat kesimpulan berdasarkan hasil simulasi menggunakan program yang nantinya akan memberikan jawaban yang mengacu dari tujuan-tujuan peneliti. Saran-saran yang bersifat membangun dan dirasa perlu untuk perkembangan penelitian selanjutnya juga dapat dipaparkan pada bagian ini.

Berikut skema untuk menyelesaikan permasalahan *knapsack 0-1* menggunakan algoritma *Ant lion Optimization* (ALO).



Gambar 3.1 Skema Metode Penelitian

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang telah diuraikan dapat diambil kesimpulan sebagai berikut.

a. Solusi terbaik yang dihasilkan oleh algoritma *Ant lion* adalah Rp 12.610.000. Solusi terbaik tersebut diperoleh dengan menggunakan nilai parameter *pop* 30 dan maksimal iterasi 500, dimana parameter maksimal iterasi lebih berpengaruh dibandingkan parameter *pop*.
b. Hasil algoritma *Ant lion* dibandingkan hasil *simplex* memiliki rata-rata persentase deviasi 0,000420375, hasil tersebut memiliki penyimpangan yang kecil sehingga algoritma *Ant lion* efektif untuk menyelesaikan permasalahan *knapsack 0-1*.

5.2 Saran

Saran untuk penelitian selanjutnya dapat mengembangkan atau memodifikasi algoritma *Ant lion Optimization* (*ALO*) pada permasalahan lain, dimana kendala yang digunakan tidak hanya berat barang, akan tetapi juga volume. Penelitian berikutnya bias menggunakan *multiple knapsack*. Selain itu, disarankan untuk dapat menerapkan algoritma *Ant lion* pada permasalahan optimasi lain.

DAFTAR PUSTAKA

Ariastuti, P. R. 2015. *Penerapan Algoritma Genetika dan Algoritma Harmony Search pada Permasalahan Knapsack 0-1*. Tidak diterbitkkan. *Jurnal*. Jember: Universitas Jember.

Dantzig, G. B. (2002). Linear Programming. *Operation Research*, 50(1).42-47

Dimyati, T.T. dan Dimyati, A. 2004. *Operations Research Model-model Pengambilan Keputusan.* Bandung: Sinar Baru Algesindo.

Dwijanto. 2008. *Program Linear Berbantu Komputer: Lindo, Lingo*, dan *Solver*. Semarang: UNNES Press.

Hayyu, A.N. 2016. *Penerapan Algoritma Artificial Bee Colony Perasalahan Multiple Constraints Bounded knapsack.* Tidak diterbitkan. *Jurnal*. Jember: Universitas Jember.

Martello, S. 2006. *New Trends in Exact Algorithms for the 0-1 Knapsack Problem.* http://www.cise.ufl.edu/~sahni/papers/knap.pdf. [Diakses pada 7 Januari 2019]

Metha, M. dan Nischal, M.M. 2015. *Ant lion optimization for optimum power generation with velve point effect.* Int. J. Res. Appl. Sci. Eng. Technol.2:1-6.

Mhand, H. Slim, A dan Abdelkader, S. *An Efficient Algorithm for the Knapsack Sharing Problem*. Computational Optimization and Applications, Spinger Verlag, 2002, 23 (1), pp.27-45. 10.1023/A:1019920507008.hal-000231189

Mirjalili, S. 2015. *The Ant Lion Optimizer.* School of Information and Communication Technology, Griffith University. Australia.

Munir, R. 2005. *Strategi Algoritmik.* Yogyakarta: Graha Ilmu.

Petovic, M. Petronijevic, J. Mitic, M. Vukovic, N. Plemic, A. Miljikkovic dan Z. Babic, B.2015. The Ant Lion Optimization Algorithm for Fexible Process Planning. Serbia: *Journal of Production Engineering*. Vol.18 No.2.

Ragsdale, C.T., 2007. *Spreadsheet Modelling and Decision Analysis*: *A Practical Introduction to Management Science*, 5e. Spinger, Verlag, New York.

Suyanto. 2010. *Algoritma Optimasi Deterministik atau Probalisik*. Yogyakarta: Graha Ilmu.

Vallada Regalado, E.;Ruiz Garcia, R.(2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*. 211:6112-622. doi:10.1016/j.ejor.2011.01.011.

LAMPIRAN

Lampiran A. Data Penelitian

A.1 Data 1 (Kapasitas Knapsack = 400kg)

No	Nama Barang	Jumlah	Berat Satuan(Kg)	Biaya Produksi/item	Harga Jual/item
1	Meja Biasa	8	8	165500	200000
2	Meja Kaca	4	10	196000	250000
3	Meja Sudut	4	10	248000	300000
4	Nampan 1	20	0,5	28000	35000
5	Nampan 2	18	1	43000	55000
6	Nampan 3	16	1	48500	65000
7	Nampan 4	15	1	46000	75000
8	Peti Harta Besar	20	2	210000	350000
9	Peti Harta Kecil	20	1	163000	250000
10	Vas Bunga Duduk	18	0,5	32500	70000
11	Vas Bunga Lantai	18	0,75	36000	100000
12	Lampu Gantung	6	3,5	245000	300000
13	Lampu Dinding 1	9	2,5	70000	100000
14	Lampu Dinding 2	9	2,5	98000	150000
15	Hiasan Dinding	20	0,5	20000	25000
16	Pigura 20R	12	1	45500	50000
17	Pigura 85x65	12	1,5	88000	200000
18	Kere Gambar 1	15	1,5	46000	75000
19	Kere Gambar 2	10	1,75	57500	100000
20	Kere Gambar 3	8	2	80000	150000
21	Kere Gambar 4	8	2,25	96000	200000

A.1 Data 2 (Kapasitas Knapsack = 1000kg)

No	Nama Barang	Jumlah	Berat Satuan(kg)	Biaya Produksi/item	Harga Jual/item
1	Barang 1	4	1,6	196500	214000
2	Barang 2	5	2	39000	41500
3	Barang 3	3	8,7	14000	16500
4	Barang 4	9	5	14500	15500
5	Barang 5	10	7,2	161000	167500
6	Barang 6	7	3,8	19500	20500
7	Barang 7	5	7	21500	22000
8	Barang 8	3	8	190000	207000
9	Barang 9	5	7	53000	61500

No	Nama Barang	Jumlah	Berat Satuan (kg)	Biaya Produksi/item	Harga Jual/item
10	Barang 10	12	5,2	41500	48000
11	Barang 11	1	4,3	144000	162500
12	Barang 12	1	0,6	118500	125500
13	Barang 13	8	9,2	31500	35000
14	Barang 14	7	8,8	58000	66000
15	Barang 15	8	2	151500	157500
16	Barang 16	1	2	87000	88500
17	Barang 17	4	3	49500	50500
18	Barang 18	9	3,5	146000	172500
19	Barang 19	3	2,4	97000	108500
20	Barang 20	5	0,8	106500	114000
21	Barang 21	3	8,5	143500	172000
22	Barang 22	9	2,2	105500	115000
23	Barang 23	1	4,5	72000	82000
24	Barang 24	10	1	102500	107500
25	Barang 25	1	5,6	95000	110000
26	Barang 26	10	1,5	200000	218000
27	Barang 27	7	7,5	151500	154500
28	Barang 28	3	6	132500	143000
29	Barang 29	6	6	134000	151500
30	Barang 30	5	5	150000	166500
31	Barang 31	2	4,7	77500	82000
32	Barang 32	6	5,1	97500	113000
33	Barang 33	4	1,8	151500	168000
34	Barang 34	12	2	113000	130000
35	Barang 35	7	9,6	189000	196500
36	Barang 36	10	2,9	132000	158500
37	Barang 37	6	3,6	136500	142000
38	Barang 38	4	2,4	18500	21500
39	Barang 39	6	2,4	149000	158000
40	Barang 40	5	9	155000	170500
41	Barang 41	4	3,6	17500	20500
42	Barang 42	3	10	15500	17500
43	Barang 43	12	9	85000	89500
44	Barang 44	9	0,8	139000	164000
45	Barang 45	3	5	197500	225000
46	Barang 46	4	4,6	99000	106000
47	Barang 47	5	8	163500	185000
48	Barang 48	8	7,2	54500	63000
49	Barang 49	3	8	188000	210500
50	Barang 50	10	3	103500	123000

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

B.1 Hasil *Simpex* Data Kerajinan

Microsoft Excel 14.0 Answer Report
Worksheet: [Data Skripsi - Simplex.xlsx]Sheet1 (2)
Report Created: 01/12/2019 13:56:03
Result: Solver found a solution. All Constraints and optimality conditions are satisfied. Solver Engine
Engine: Simplex LP
Solution Time:0,328 Seconds. Iterations: 1 Subproblems: 34
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
$J$24	Profit	0	12610000

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$H$2	Meja Biasa Solusi	0	0	Binary
$H$3	Meja Kaca Solusi	0	1	Binary
$H$4	Meja Sudut Solusi	0	1	Binary
$H$5	Nampan 1 Solusi	0	1	Binary
$H$6	Nampan 2 Solusi	0	1	Binary
$H$7	Nampan 3 Solusi	0	1	Binary
$H$8	Nampan 4 Solusi	0	1	Binary
$H$9	Peti Harta Besar Solusi	0	1	Binary
$H$10	Peti Harta Kecil Solusi	0	1	Binary
$H$11	Vas Bunga Duduk Solusi	0	1	Binary
$H$12	Vas Bunga Lantai Solusi	0	1	Binary
$H$13	Lampu Gantung Solusi	0	1	Binary
$H$14	Lampu Dinding 1 Solusi	0	1	Binary
$H$15	Lampu Dinding 2 Solusi	0	1	Binary
$H$16	Hiasan Dinding Solusi	0	1	Binary
$H$17	Pigura 20R Solusi	0	0	Binary
$H$18	Pigura 85x65 Solusi	0	1	Binary
$H$19	Kere Gambar 1 Solusi	0	1	Binary
$H$20	Kere Gambar 2 Solusi	0	1	Binary
$H$21	Kere Gambar 3 Solusi	0	1	Binary

$H$22	Kere Gambar 4 Solusi	0	1	Binary

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$I$24	Berat	389,5	$I$24<=$I$26	Not Binding	10,5
$H$2:$H$22=Binary					

B.2 Hasil *Simplex* Data Simulasi

Microsoft Excel 14.0 Answer Report

Worksheet: [Simplex Data Simulasi 1.xlsx]Sheet1

Report Created: 01/12/2019 13:58:52

Result: Solver found a solution. All Constraints and optimality conditions are satisfied.

Solver Engine

Engine: Simplex LP

Solution Time: 0,203 Seconds.

Iterations: 3 Subproblems: 14

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
$K$53	Profit	0	3068500

Variabe Cells

Cell	Name	Original Value	Final Value	Integer
$I$2	Barang 1 Solusi	0	1	Binary
$I$3	Barang 2 Solusi	0	1	Binary
$I$4	Barang 3 Solusi	0	1	Binary
$I$5	Barang 4 Solusi	0	0	Binary
$I$6	Barang 5 Solusi	0	1	Binary
$I$7	Barang 6 Solusi	0	0	Binary
$I$8	Barang 7 Solusi	0	0	Binary
$I$9	Barang 8 Solusi	0	1	Binary
$I$10	Barang 9 Solusi	0	1	Binary
$I$11	Barang 10 Solusi	0	1	Binary
$I$12	Barang 11 Solusi	0	1	Binary
$I$13	Barang 12 Solusi	0	1	Binary
$I$14	Barang 13 Solusi	0	0	Binary
$I$15	Barang 14 Solusi	0	1	Binary

$I$16	Barang 15 Solusi	0	1	Binary
$I$17	Barang 16 Solusi	0	1	Binary
$I$18	Barang 17 Solusi	0	1	Binary
$I$19	Barang 18 Solusi	0	1	Binary
$I$20	Barang 19 Solusi	0	1	Binary
$I$21	Barang 20 Solusi	0	1	Binary
$I$22	Barang 21 Solusi	0	1	Binary
$I$23	Barang 22 Solusi	0	1	Binary
$I$24	Barang 23 Solusi	0	1	Binary
$I$25	Barang 24 Solusi	0	1	Binary
$I$26	Barang 25 Solusi	0	1	Binary
$I$27	Barang 26 Solusi	0	1	Binary
$I$28	Barang 27 Solusi	0	0	Binary
$I$29	Barang 28 Solusi	0	1	Binary
$I$30	Barang 29 Solusi	0	1	Binary
$I$31	Barang 30 Solusi	0	1	Binary
$I$32	Barang 31 Solusi	0	1	Binary
$I$33	Barang 32 Solusi	0	1	Binary
$I$34	Barang 33 Solusi	0	1	Binary
$I$35	Barang 34 Solusi	0	1	Binary
$I$36	Barang 35 Solusi	0	1	Binary
$I$37	Barang 36 Solusi	0	1	Binary
$I$38	Barang 37 Solusi	0	1	Binary
$I$39	Barang 38 Solusi	0	1	Binary
$I$40	Barang 39 Solusi	0	1	Binary
$I$41	Barang 40 Solusi	0	1	Binary
$I$42	Barang 41 Solusi	0	1	Binary
$I$43	Barang 42 Solusi	0	0	Binary
$I$44	Barang 43 Solusi	0	0	Binary
$I$45	Barang 44 Solusi	0	1	Binary
$I$46	Barang 45 Solusi	0	1	Binary
$I$47	Barang 46 Solusi	0	1	Binary
$I$48	Barang 47 Solusi	0	1	Binary
$I$49	Barang 48 Solusi	0	1	Binary
$I$50	Barang 49 Solusi	0	1	Binary
$I$51	Barang 50 Solusi	0	1	Binary

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$J$53	Berat	999,1	$J$53<=$J$55	Not Binding	0,9
$I$2:$I$51=Binary					

Lampiran C. Hasil Percobaan Uji Parameter

C.1 Uji Parameter Populasi data Kerajinan

Pop = 25; Itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	10	1,5720	12610000
2	7	1,5565	12610000
3	9	1,5655	12610000
4	7	1,5579	12564000
5	4	1,5631	12610000
6	8	1,5672	12610000
7	4	1,6391	12610000
8	3	1,5700	12524000
9	7	1,5655	12610000
10	8	1,5581	12610000
Rata-rata	6,7	1,57149	12596800

Pop = 30; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	6	1,5580	12610000
2	4	1,5572	12610000
3	4	1,5689	12610000
4	6	1,5599	12610000
5	6	1,5690	12610000
6	11	1,5703	12610000
7	10	1,5657	12610000
8	14	1,5587	12610000
9	7	1,5653	12610000
10	4	1,6506	12610000
Rata-rata	7,2	1,57236	12610000

Pop = 35; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	12	1,6342	12610000
2	10	1,5605	12610000
3	10	1,5571	12610000
4	6	1,5669	12610000
5	3	1,5566	12464000
6	4	1,7485	12610000
7	7	1,5613	12610000
8	4	1,5558	12610000
9	3	1,5696	12610000
10	10	1,5710	12610000
Rata-rata	6,9	1,58815	12595400

Pop = 40; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	4	1,5661	12516000
2	7	1,5567	12462000
3	12	1,5644	12610000
4	6	1,5648	12610000
5	6	1,5735	12610000
6	7	1,5618	12516000
7	4	1,5687	12610000
8	5	1,5682	12610000
9	6	1,5896	12610000
10	9	1,6690	12610000
Rata-rata	6,6	1,57828	12576400

Pop = 25; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	13	3,1191	12610000
2	11	3,1187	12610000
3	20	3,1324	12610000
4	7	3,1311	12610000
5	10	3,1312	12610000
6	6	3,1296	12610000
7	8	3,1391	12516000
8	5	3,1193	12610000
9	9	3,1196	12516000
10	12	3,1215	12610000
Rata-rata	10,1	3,12616	12591200

Pop 30; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	15	3,2093	12610000
2	5	3,1229	12564000
3	9	3,1219	12610000
4	11	3,1306	12610000
5	5	3,1272	12610000
6	23	3,1302	12610000
7	21	3,1274	12610000
8	6	3,1197	12610000
9	19	3,1206	12610000
10	9	3,1214	12610000
Rata-rata	12,3	3,13312	12605400

Pop = 35; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	8	3,1302	12610000
2	8	3,1630	12610000
3	5	3,1262	12610000
4	8	3,1330	12610000
5	10	3,1297	12610000
6	14	3,1184	12610000
7	10	3,1316	12610000
8	13	3,2153	12610000
9	4	3,1200	12610000
10	10	3,1207	12610000
Rata-rata	9	3,13881	12610000

Pop = 40; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	5	3,1206	12610000
2	3	3,1309	12610000
3	6	3,1282	12610000
4	6	3,1276	12610000
5	7	3,1205	12610000
6	4	3,1253	12610000
7	7	3,1302	12610000
8	10	3,1192	12610000
9	13	3,1319	12610000
10	5	3,1252	12610000
Rata-rata	6,6	3,12596	12610000

Pop = 25; itermax = 500

No	Iterasi Konvergen	Waktu Koomputasi (s)	Profit
1	14	7,8121	12610000
2	22	7,8140	12610000
3	17	7,8190	12610000
4	24	7,8097	12610000
5	9	7,8150	12610000
6	13	7,8480	12610000
7	5	7,8097	12610000
8	22	7,8072	12610000
9	27	7,8099	12610000
10	20	7,8168	12610000
Rata-rata	17,3	7,81614	12610000

Pop = 30; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	9	7,8143	12610000
2	7	7,8086	12610000
3	12	7,8186	12610000
4	10	7,8117	12610000
5	10	7,8050	12610000
6	9	7,8103	12610000
7	12	7,8077	12610000
8	16	7,8161	12610000
9	16	7,8174	12610000
10	18	7,8131	12610000
Rata-rata	11,9	7,81228	12610000

Pop = 35; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	9	7,8059	12610000
2	5	7,8180	12610000
3	9	7,8107	12610000
4	5	7,8108	12610000
5	5	7,8085	12610000
6	20	7,8904	12610000
7	7	7,8064	12610000
8	8	7,8146	12610000
9	30	7,8139	12610000
10	16	7,8142	12610000
Rata-rata	11,4	7,81934	12610000

Pop = 40; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	4	7,9076	12610000
2	5	7,8199	12610000
3	7	7,8123	12610000
4	5	7,8047	12610000
5	8	7,8111	12610000
6	6	7,8127	12610000
7	11	7,8045	12610000
8	9	7,8054	12610000
9	12	7,8109	12610000
10	12	7,8148	12610000
Rata-rata	7,9	7,82039	12610000

Pop = 25; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	11	15,6173	12610000
2	8	15,6199	12610000
3	28	15,6278	12610000
4	13	15,6240	12610000
5	28	15,6355	12610000
6	15	15,6475	12610000
7	4	15,6259	12610000
8	6	15,6143	12610000
9	11	15,7005	12610000
10	20	15,6232	12610000
Rata-rata	14,4	15,63359	12610000

Pop = 30; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	14	15,6191	12610000
2	20	15,6173	12610000
3	20	15,7430	12610000
4	15	15,6221	12610000
5	32	15,6241	12610000
6	13	15,6196	12610000
7	12	15,6187	12610000
8	7	15,7849	12610000
9	21	15,6143	12610000
10	9	15,6218	12610000
Rata-rata	16,3	15,64849	12610000

Pop = 35; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	16	15,6371	12610000
2	10	15,6892	12610000
3	8	15,6148	12610000
4	14	15,7054	12610000
5	9	15,6296	12610000
6	9	15,6240	12610000
7	16	15,7315	12610000
8	10	15,6291	12610000
9	15	15,7219	12610000
10	21	15,6146	12610000
Rata-rata	12,8	15,65972	12610000

Pop = 40; itermax = 1000

No	Iterasi konvergen	Waktu Komputasi (s)	Profit
1	11	15,6169	12610000
2	22	15,7565	12610000
3	20	15,6270	12610000
4	12	15,6161	12610000
5	17	15,6501	12610000
6	12	15,6254	12610000
7	7	15,6161	12610000
8	14	15,6238	12610000
9	13	15,6259	12610000
10	10	15,7017	12610000
Rata-rata	13,8	15,64595	12610000

C.1 Uji ParameterPopulasi data Pembanding

Pop = 25; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	20	1,57	3015000
2	46	1,5667	3036500
3	35	1,5674	3019000
4	31	1,5593	3019500
5	66	1,5919	3068500
6	32	1,5581	3028500
7	60	1,5604	3053500
8	35	1,559	3055500
9	44	1,5699	3068500
10	21	1,5553	2948500
Rata-rata	39	1,5658	3031300

Pop 30; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	42	1,5608	3049000
2	24	1,5661	3019500
3	51	1,5699	3025500
4	82	1,5665	3036500
5	56	1,558	3026500
6	67	1,5701	3016000
7	40	1,5628	3023000
8	27	1,5642	3015000
9	33	1,5636	3029000
10	36	1,5629	2996000
Rata-rata	45,8	1,56449	3023600

Pop 35; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	21	1,5657	3068500
2	32	1,5697	3020000
3	23	1,5635	3024500
4	27	1,5670	3068500
5	34	1,5593	3029000
6	44	1,5709	3046500
7	22	1,5669	3037000
8	81	1,5694	3040000
9	55	1,6594	3068500
10	50	1,5687	3068500
Rata-rata	38,9	1,57605	3047100

Pop = 40; itermax = 100

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	38	1,5649	3068000
2	67	1,5690	3066000
3	31	1,5559	3016500
4	34	1,5639	3024500
5	99	1,5695	3050000
6	48	1,5628	3029000
7	68	1,5671	3068500
8	40	1,5572	3068500
9	39	1,5624	3068500
10	39	1,5577	3029000
Rata-rata	50,3	1,56304	3048850

Pop = 25; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	98	3,1189	3068500
2	164	3,1215	3023500
3	99	3,12	3068000
4	47	3,1325	3032000
5	45	3,1214	3029000
6	48	3,2259	3029000
7	32	3,1247	3068500
8	31	3,1286	3054500
9	40	3,1177	3012500
10	90	3,1276	3035500
Rata-rata	69,4	3,13388	3042100

Pop = 30; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	89	3,1194	3068000
2	66	3,1192	3068500
3	39	3,1181	3025500
4	47	3,1291	3029000
5	51	3,1209	3029000
6	56	3,1324	3068500
7	30	3,1325	3068500
8	52	3,1272	3028000
9	54	3,1249	3033000
10	73	3,1246	3057000
Rata-rata	55,7	3,12483	3047500

Pop = 35; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	56	3,1296	3068000
2	46	3,1245	3040000
3	67	3,1186	3049000
4	87	3,1216	3047500
5	55	3,1260	2990500
6	56	3,1337	3068500
7	64	3,1306	3068500
8	50	3,1325	3068500
9	139	3,1214	3068500
10	23	3,1177	3029000
Rata-rata	64,5	3,12562	3049800

Pop = 40; itermax = 200

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	60	3,1322	3036000
2	144	3,1237	3066000
3	58	3,1290	3068500
4	44	3,1953	3068500
5	89	3,1186	3068500
6	77	3,1317	3068500
7	121	3,1208	3068500
8	59	3,1281	3064000
9	79	3,1190	3068500
10	39	3,1260	3023000
Rata-rata	77	3,13244	3060000

Pop = 25;itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	109	7,8194	3040000
2	108	7,8032	3068000
3	97	7,8190	3068500
4	100	7,8111	3056500
5	45	7,8099	3024500
6	265	7,8183	3068500
7	85	7,8210	3068000
8	65	7,8129	2993000
9	87	7,8119	3040000
10	46	7,8166	3031500
Rata-rata	100,7	7,81433	3045850

Pop = 30; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	114	7,8194	3038500
2	105	7,8059	3068500
3	143	7,8097	3056000
4	124	7,8144	3068500
5	251	7,8131	3059000
6	99	7,8065	3068500
7	74	7,8164	3012500
8	157	7,8054	3057000
9	102	7,8174	3068500
10	143	7,8177	3068500
Rata-rata	131,2	7,03082	3056550

Pop = 35; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	489	7,8040	3039500
2	210	7,8123	3068000
3	68	7,8141	3036500
4	127	7,8180	3068500
5	133	7,8085	3039000
6	97	7,8124	3068500
7	138	7,8050	3068500
8	74	7,8120	3068500
9	97	7,8143	3033000
10	56	7,8054	3068500
Rata-rata	148,9	7,8106	3055850

Pop = 40; itermax = 500

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	233	7,8082	3068500
2	199	7,8108	3068500
3	107	7,8265	3068500
4	89	7,8169	3068500
5	56	7,8079	3068000
6	171	7,8188	3036500
7	103	7,8051	3040000
8	56	7,8070	3068500
9	170	7,8121	3068500
10	100	7,8150	3068000
Rata-rata	128,4	7,81283	3062350

Pop = 25; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	123	15,6249	3032000
2	172	15,6205	3064500
3	208	15,6142	3063000
4	460	15,6275	3068500
5	155	15,6193	3068500
6	157	15,7379	3068500
7	321	15,6205	3036000
8	59	15,6148	3027000
9	68	15,6292	3028500
10	245	15,6209	3068500
Rata-rata	196,8	15,63297	3052500

Pop = 30; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	116	15,7549	3068500
2	216	5,61570	3052000
3	134	15,6151	3024000
4	221	15,6199	3068000
5	245	15,7387	3068000
6	220	15,6179	3028000
7	401	15,6289	3068500
8	147	15,7286	3058000
9	175	15,6267	3068500
10	172	15,6222	3061000
Rata-rata	204,7	1,57549	3056450

Pop = 35; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	99	15,6154	3068500
2	174	15,6238	3068000
3	210	15,6243	3068000
4	295	15,6802	3068500
5	132	15,7197	3068500
6	118	15,6183	3068500
7	175	15,6295	3068500
8	127	5,62920	3068500
9	125	15,6287	3068500
10	188	15,6268	3068500
Rata-rata	164,3	14,63959	3068400

Pop = 40; itermax = 1000

No	Iterasi Konvergen	Waktu Komputasi (s)	Profit
1	127	15,7343	3068500
2	100	15,6257	3030000
3	243	15,6230	3068500
4	273	15,6216	3068500
5	262	15,6597	3068500
6	288	15,6175	3068500
7	183	15,6238	3068500
8	122	15,6178	3068500
9	142	15,6298	3068500
10	242	15,7246	3039500
Rata-rata	198,2	15,64778	3061750

D.1 Pseudocode Algoritma Antlion Optimizer

Pseudocode Algoritma Ant Lion Optimizer

```
Input:
Banyak jenis barang (D)
Jumlah barang (m)
Berat barang (w)
Harga produksi (b)
Harga jual (p)
Kapasitas berat kendaraan(Cap)
Kapasitas volume kendaraan(Space) Modal (M)
a = 1
```

```
a max = -1
Populasi (N); w; Maksimal iterasi (T)

{Pembangkitan Solusi dan Random Walk}
For1 i = 1 : N
   For2 d = 1 : D
         M1[i,d] = rand(0,1)
         Y1[i,d] = round(M1[i,d])
         M2[i,d] = rand(0,1)
         Y2[i,d] = round(M2[i,d])
     EndFor2
EndFor1


{ Inisialisasi Populasi }
For1 I = 1 : N
      For2 d = 1 : D
            M1 [ i , d ] = rand ( 0 , 1 )
            Y1 [ i , d ] = round (M1 [ i , d ])
            M2 [ i , d ] = rand ( 0 , 1 )
            Y2 [ i , d ] = round(M2 [ i , d ])
      EndFor2
EndFor1

{Periksa kendala}
For i = 1 : N
For j = 1 : D
      Berat1 = Berat1 + W [j]*Y1[i,j]
      Berat2 = Berat2 + W [j]*Y2[i,j]
         End
    While (Berat1>berat)
      For j = 1 : D
        Y1[i,j] = round(rand[0,1])
        Berat1 = berat1 + W(j)*Y1(i,j)
        End;End;
While (Berat2>berat)
      For j = 1 : D
        Y2[i,j] = round(rond[0,1])
        Berat2 = berat2 + W(j)*Y2(i,j)
```

```
      End;End;
        End;
Protot1 = 0; Protot2 = 0

      For i = 1 : N
        Profit1[i] = 0; Profit2[i] = 0;
        For j = 1 : D

          Profit1[i] = Profit1[i]+Y1[I,j][P(j)-b(j)]
          Profit2[i] = Profit2[i]+Y2[I,j][P()-b(j)]
          Protot1 = Protot1 + Profit1[i]
          Protot2 = Protot2 + Profit2[i]
           End
         EndFor
       EndFor
     Mak = profit2 [I]; Elite = 1
     For i = 1 : N
     If mak < profit2 (i)
       Mak = profit2 (i)
       Elite = i
     Endif
Endfor
     For i = 1 : N
         Prob[i] = Profit2[i]/Protot2
     For i = 1 : N
       Pk[i] = 0
     For j = 1 : i
       Pk[i] = Pk[i] + Pk[j]
       For i = 1 : D
        c[i] = x[1,i]
        d[i] = x[1,i]
        For j = 1 : n
        If c [i] > x [j,i]
          c[i] = x[j,i]
        If d [i] <x [j,i]
          d[i] = x[j,i]
         End
       End
```

```
 End
For k = 1 : N
For I = 1 : D

   e(i) = c(i) + x(k,i)
   f(i) = d(i) + x(k,i)
 Endfor
 for i = 1 : D
   r[i] = round (rand(0..1));
    if r[i] = 0
      x[i] = -1
    if r[i] = 1
      x[i] = 1
    Endif
  Endfor
  For t = 1 : D
   RA(t) = c(t)+((x(t)-a)*(f(t)-e(t))/(amax-a))
 For i = 1 : D
   e(i) = c(i) + Elite
   f(i) = d(i) + Elite
   s(i) = round(rand(0..1))
   if s(i) = 0 x(i) = -1
   if s(i) = 1 x(i) = 1
   RE(i) = c(i) + (((x(i)-a)*(f(i)-e(i)))/(amax-a))
 Endif; Endfor
 For i = 1 : D
     Ant(k,i) = round((RA(i)+RE(i))/2)
  Endfor
Endfor
```