PENERAPAN ALGORITMA GENETIKA DAN ALGORITMA *HARMONY SEARCH* PADA PERMASALAHAN *KNAPSACK* 0-1

SKRIPSI

Oleh

Putri Rukmana Ariastuti
NIM 101810101044

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

PENERAPAN ALGORITMA GENETIKA DAN ALGORITMA *HARMONY SEARCH* 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

Putri Rukmana Ariastuti
NIM 101810101044

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Ibundaku Faridha Ariastuti, lentera kehidupanku. Malaikat yang memberikan seluruh hidupnya untukku, yang senantiasa mencurahkan kasih sayang, doa, dan semangat di setiap detik waktuku;
2. adik-adikku tercinta Intan Nur Wulandari, Nafisatul Muawanah, dan Bagaskara Rifa Dzulkarnain yang selalu memberikan semangat dan doa;
3. Kakek Amran Burhanudin (Alm.) tercinta, saya wujudkan cita-cita kakek sebagai sarjana pertama dalam keluarga;
4. para dosen dan guru saya sejak taman kanak-kanak hingga perguruan tinggi yang telah membimbing dan membagi ilmu dengan tulus dan ikhlas;
5. Almamater Jurusan Matematika FMIPA Universitas Jember, SMAN 5, SMPN 1, SD Al-Furqan, dan TK Al-Amien Jember.

MOTTO

"Allah akan meninggikan orang-orang yang beriman di antara kamu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat."
(terjemahan Q.S. Al-Mujadalah ayat 11)*)

"Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari suatu urusan), tetaplah bekerja keras untuk (urusan yang lain) dan hanya kepada Tuhanmulah engkau berharap."
(terjemahan Q.S. Al-Insyirah ayat 6-8)*)

"Anda tidak bisa mengubah orang lain, Anda yang harus menjadi perubahan yang Anda harapkan dari orang lain."
(Mahatma Gandhi)

*) Departemen Agama Republik Indonesia. 2004. *Al-Qur'an dan Terjemahannya*. Bandung: CV. Penerbit Diponegoro.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Putri Rukmana Ariastuti

NIM : 101810101044

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *Knapsack* 0-1" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan dalam 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, Juni 2015

Yang menyatakan,

Putri Rukmana Ariastuti
NIM 101810101044

SKRIPSI

PENERAPAN ALGORITMA GENETIKA DAN ALGORITMA *HARMONY SEARCH* PADA PERMASALAHAN *KNAPSACK* 0-1

Oleh
Putri Rukmana Ariastuti
NIM 101810101044

Pembimbing

Dosen Pembimbing Utama	**:**	**Ahmad Kamsyakawuni, S.Si., M.Kom.**
Dosen Pembimbing Anggota	**:**	**Ika Hesti Agustin, S.Si., M.Si.**

PENGESAHAN

Skripsi berjudul "Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *Knapsack* 0-1" telah diuji dan disahkan pada:

hari, tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji

Ketua,	Sekretaris,
Ahmad Kamsyakawuni, S.Si., M.Kom. NIP. 197211291998021001	Ika Hesti Agustin, S.Si., M.Si. NIP. 198408012008012006

Anggota Tim Penguji

Anggota I,	Anggota II,
Kusbudiono, S.Si., M.Si. NIP. 197704302005011001	M. Ziaul Arif, S.Si., M.Sc. NIP. 198501112008121002

Mengesahkan
Dekan,

Prof. Drs. Kusno, DEA, Ph.D.
NIP 196101081986021001

RINGKASAN

Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *Knapsack* 0-1; Putri Rukmana Ariastuti; 2015; 62 Halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Masalah *knapsack* adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang dimana setiap barang tersebut mempunyai berat dan keuntungan masing-masing yang berbeda, sehingga dari pemilihan barang tersebut didapatkan keuntungan yang maksimum. Masalah *knapsack* sering terjadi pada media transportasi ketika akan mengangkut banyak barang, dimana berat barang yang diangkut tidak boleh melebihi kapasitas media transportasi yang digunakan.

Penelitian ini membandingkan algoritma Genetika dan algoritma *Harmony Search*. Selain untuk mengetahui konsep dan hasil dari kedua algoritma ini juga bertujuan untuk mengetahui algoritma mana yang lebih baik untuk menyelesaikan permasalahan *knapsack* 0-1 dengan membandingkan hasil, *running time*, dan konvergensi dari kedua algoritma.

Hasil penelitian menunjukkan bahwa algoritma Genetika lebih baik daripada algoritma *Harmony Search* dalam menyelesaikan permasalahan *knapsack* 0-1 pada studi kasus UD. Permata Indah Situbondo. Hasil profit yang ditunjukkan oleh algoritma Genetika lebih stabil daripada algoritma *Harmony Search* yaitu pada nilai Rp 6.343.000,-, sedangkan hasil yang ditunjukkan algoritma *Harmony Search* mengalami perubahan yang fluktuatif sejak percobaan ketiga. Dilihat dari sisi yang lain, algoritma Genetika memiliki *running time* yang lebih lama daripada algoritma *Harmony Search*. Akan tetapi, algoritma Genetika mencapai konvergen lebih cepat daripada algoritma *Harmony Search*.

PRAKATA

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karuniaNya sehingga skripsi yang berjudul "Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *Knapsack* 0-1" dapat terselesaikan. Skripsi ini disusun untuk memenuhi salah satu syarat dalam menyelesaikan pendidikan strata 1 (S1) di Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember. Sholawat dan salam semoga tercurahkan kepada junjungan Nabi besar Muhammad SAW yang telah menjadi pembawa rahmatan lil'alamin.

Penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Bapak Ahmad Kamsyakawuni, S.Si., M.Kom. selaku Dosen Pembimbing Utama dan Ibu Ika Hesti Agustin, S.Si., M.Si. selaku Dosen Pembimbing Anggota yang telah memberikan bimbingan dan bantuan untuk penyempurnaan skripsi ini;
2. Bapak Kusbudiono, S.Si., M.Si. selaku Dosen Penguji I dan Bapak M. Ziaul Arif, S.Si., M.Sc. selaku Dosen Penguji II yang telah memberikan kritik dan saran yang membangun untuk penyempurnaan skripsi ini;
3. Bapak Ahmad Kamsyakawuni, S.Si., M.Kom. selaku Dosen Pembimbing Akademik yang telah memberikan banyak masukan selama menjalani perkuliahan;
4. Ibunda Faridha Ariastuti dan Ayahanda Nurul Huda tercinta;
5. Intan Nur Wulandari, Nafisatul Muawannah, dan Bagaskara Rifa Dzulkarnain, yang selalu menjaga semangat dan memotivasi untuk menjadi kakak yang baik dan selalu memberikan yang terbaik untuk keluarga;

6. sahabat sekaligus partner terbaik, Andina Ishmah Almira, Maghfirah, Anitha Indah Puspitasari, Annisa Luthfi S.R, Alif Mifta Mardiana, yang selalu mendukung serta berbagi saat suka dan duka;
7. Novan Adi Kusuma, S.T, yang selalu setia memberi perhatian, semangat dan kasih sayang, terima kasih telah tumbuh dewasa bersamaku;
8. murid-muridku tersayang Hendriana Wijaya, Vinca Natalia, Erik Hendrawan, Evangeline Carina, dan Erta Aprilia, terima kasih sudah mewarnai hariku selama beberapa tahun terakhir ini;
9. seluruh anggota Unit Kegiatan Mahasiswa Seni TITIK FMIPA Universitas Jember yang menanamkan rasa kekeluargaan, nilai-nilai positif, dan begitu banyak memberi pengalaman berharga;
10. seluruh keluarga besar Amran Burhanudin dan Sarpan, yang telah memberikan motivasi untuk selalu menjadi yang terbaik;
11. seluruh keluarga besar MATHGIC 2010 dan kakak angkatan matematika tersayang, terima kasih atas kekompakan dan persahabatannya;
12. semua pihak yang telah banyak membantu dan tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam menyusun skripsi ini masih terdapat kekurangan baik isi maupun susunannya. Oleh karena itu penulis mengharapkan kritik dan saran demi penyempurnaan skripsi ini. Akhirnya penulis berharap semoga skripsi ini dapat memberi manfaat dan sumbangan bagi ilmu pengetahuan.

Jember, Juni 2015 Putri Rukmana Ariastuti

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Knapsack merupakan suatu kantong atau tempat yang digunakan untuk memuat sesuatu objek. Kantong atau tempat tersebut hanya dapat menyimpan beberapa objek saja dengan ketentuan total ukuran objek tersebut lebih kecil atau sama dengan ukuran kapasitasnya (Paryati, 2009). Menurut Martello (2006), masalah *knapsack* adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang dimana setiap barang tersebut mempunyai berat dan keuntungan masing-masing yang berbeda, sehingga dari pemilihan barang tersebut didapatkan keuntungan yang maksimum. Masalah *knapsack* sering terjadi pada media transportasi ketika akan mengangkut banyak barang, dimana berat barang yang diangkut tidak boleh melebihi kapasitas media transportasi yang digunakan. Oleh karena itu dilakukan pemilihan barang dan diharapkan dari pemilihan tersebut didapatkan keuntungan yang maksimal. Keuntungan yang maksimal merupakan tujuan utama suatu perusahaan dalam mengelola modal secara efektif dan efisien.

Masalah *knapsack* dapat diselesaikan dengan berbagai macam algoritma, salah satunya adalah algoritma Genetika seperti yang sudah dilakukan oleh Diah *et al* (2010) pada penelitiannya yang berjudul "*Penyelesaian Knapsack Problem Menggunakan Algoritma Genetika*". Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu, dimana masing-masing individu merepresentasikan sebuah solusi yang mungkin bagi persoalan yang ada untuk selanjutnya mengalami proses seleksi, pindah silang, dan mutasi sehingga didapatkan populasi baru yang memberikan solusi yang mendekati solusi optimal.

Pada penelitiannya yang berjudul "*Penyelesaian Persoalan Multidimensional Knapsack (PMK) dengan Algoritma Genetika*", Setiowati (2003) menggunakan

algoritma Genetika dan *Dynamic Programming* pada saat uji coba. Berdasarkan penelitiannya, disimpulkan bahwa algoritma Genetika cukup ampuh untuk menyelesaikan persoalan optimisasi kombinasi dan algoritma Genetika lebih efektif dan efisien dibandingkan dengan *Dynamic Programming*.

Masalah *knapsack* juga bisa diselesaikan dengan algoritma *Harmony Search* (HS) seperti yang sudah dilakukan oleh Zou *et al* (2011) dalam penelitiannya yang berjudul "*Solving 0-1 Knapsack Problem By a Novel Global Harmony Search Algorithm*". Algoritma ini didasarkan pada permainan musik dimana pemain musik akan mencari harmoni yang paling indah berdasarkan perkiraan estetika. Harmoni dalam musik tersebut merupakan representasi dari vektor solusi sedangkan proses improvisasinya merepresentasikan pencarian global atau lokal dalam teknik optimasi. *Harmony Search* memiliki struktur yang relatif mudah karena tidak perlu melibatkan kalkulasi matematika yang kompleks.

Berdasarkan penjelasan pada paragraf sebelumnya, penulis tertarik untuk mengangkat judul yang akan diteliti yaitu "Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *Knapsack* 0-1".

1.2 Rumusan Masalah

Dari latar belakang di atas, permasalahan yang akan dibahas dalam skripsi ini adalah sebagai berikut.

a. Bagaimana penyelesaian masalah *knapsack* 0-1 menggunakan algoritma Genetika dan algoritma *Harmony Search*?
b. Bagaimana perbandingan kedua algoritma tersebut berdasarkan hasil, lama waktu program dijalankan (*running time*), dan konvergensinya (iterasi yang menunjukkan kekonvergenan)?

1.3 Batasan Masalah

Adapun batasan-batasan masalah dalam hal ini adalah kelompok barang yang dipilih dimasukkan semua atau tidak sama sekali.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah sebagai berikut.

a. Menyelesaikan masalah *knapsack* 0-1 menggunakan algoritma Genetika dan algoritma *Harmony Search*.
b. Mengetahui perbandingan kedua algoritma tersebut berdasarkan hasil, lama waktu program dijalankan (*running time*), dan konvergensinya (iterasi yang menunjukkan kekonvergenan).

1.5 Manfaat

Manfaat yang diperoleh dari penelitian ini adalah memberikan pertimbangan tentang masalah efisiensi tempat dan waktu dalam pembelian (*reseller*) macam-macam barang yang harus dibeli dari toko lain. Selain itu, untuk mengetahui perbandingan antara algoritma Genetika dan algoritma *Harmony Search* dalam menyelesaikan masalah *knapsack* 0-1.

BAB 2. TINJAUAN PUSTAKA

2.1 Permasalahan *Knapsack* (*Knapsack Problems*)

Permasalahan *Knapsack* merupakan suatu permasalahan bagaimana memilih objek dari sekian banyak objek dan berapa besar objek tersebut akan disimpan sehingga diperoleh suatu penyimpanan yang optimal dengan memperhatikan objek yang terdiri dari n objek $(1, 2, 3, \dots, n)$ dimana setiap objek memiliki bobot (w_i) dan nilai profit (p_i) dengan memperhatikan juga kapasitas dari media penyimpanan 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 kombinatorial 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 dimensinya tidak terbatas.

Knapsack yang akan dibahas pada 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 bobot (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:

$$X = \{x_1, x_2, x_3, \dots, x_n\}$$

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 penyimpanan, 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 Maks/Min

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

dan fungsi kendala

$$z = \sum_{i=1}^{n} w_i x_i \leq M \tag{2.2}$$

dengan $x_i = 0$ atau $x_i = 1$, $i = 1, 2, 3, \ldots, n$,

dimana:

Z = nilai optimum dari fungsi tujuan,

z = kendala fungsi tujuan,

p_i = keuntungan barang, dengan $i = 1, 2, 3, \ldots, n$,

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

M = kapasitas media penyimpanan (*knapsack*)

x_i = banyaknya barang jenis ke-i.

2.2 Algoritma Genetika

Algoritma Genetika pertama kali diperkenalkan oleh John Holland pada tahun 1960. Algoritma Genetika adalah teknik pencarian dan optimasi yang berdasarkan pada mekanisme seleksi atau evolusi yang terjadi di alam. Algoritma Genetika mulai bekerja pada sekumpulan calon solusi yang dibangkitkan secara *random*. Sekumpulan calon solusi tersebut adalah populasi awal, masing-masing individu di dalam populasi awal disebut dengan kromosom. Gen adalah rangkaian yang membentuk suatu kromosom, dan kromosom menyatakan solusi suatu masalah.

Kromosom-kromosom pada setiap generasi akan dievaluasi oleh operator-operator algoritma. Operator-operator tersebut adalah seleksi, pindah silang (*crossover*), dan mutasi. Masing-masing kromosom pada setiap generasi dilihat nilai *fitness*-nya, kromosom yang lebih baik besar kemungkinan untuk dipilih pada generasi selanjutnya. Kromosom baru yang terbentuk pada generasi baru disebut kromosom anak (*offspring*). Setelah dihasilkan beberapa generasi, terbentuklah sekumpulan kromosom terbaik yang diharapkan mampu memberikan solusi optimum.

Menurut Goldberg (1989), algoritma genetika mempunyai karakteristik, yaitu sebagai berikut.

a. Algoritma Genetika diawali dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan.
b. Algoritma Genetika melakukan pencarian pada sebuah solusi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari individu.
c. Algoritma Genetika menggunakan informasi fungsi *fitness* sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik.
d. Algoritma Genetika menggunakan aturan-aturan transisi peluang, bukan aturan deterministik.

2.2.1 Parameter Algoritma Genetika

Parameter-parameter yang digunakan dalam algoritma Genetika sebagai berikut (Gen & Cheng, 1997).

a. Ukuran populasi

   Jumlah individu atau kromosom dinyatakan sebagai ukuran dari populasi.

b. Probabilitas *crossover* (Pc)

   Probabilitas *crossover* bernilai $0 \leq Pc \leq 1$, namun nilai probabilitas *crossover* yang baik menurut Gen & Cheng (1997) berkisar antara 0,6 sampai 1. Semakin besar nilai Pc maka semakin besar kemungkinan algoritma Genetika mengeksplorasi ruang pencarian, sekaligus mempercepat ditemukannya solusi optimum.

c. Probabilitas mutasi (Pm)

   Probabilitas mutasi didefinisikan sebagai presentase kromosom yang akan mengalami mutasi terhadap total kromosom di dalam populasi. Probabilitas mutasi bernilai $0 \leq Pm \leq 1$, namun nilai probabilitas mutasi yang baik menurut Gen & Cheng (1997) berkisar antara 0,001 sampai 0,2. Jika Pm terlalu kecil maka banyak kromosom yang memiliki potensi tidak akan pernah muncul, tetapi jika Pm terlalu besar maka akan banyak bermunculan kromosom yang kemungkinan tidak memiliki potensi dalam pencapaian solusi optimum. *Offspring* akan kehilangan kemiripan dengan kromosom induk (*parent*) yang memiliki potensi pada populasi

sebelumnya dan algoritma Genetika akan kehilangan kemampuan untuk belajar dari proses pencarian yang lalu.

d. Kriteria pemberhentian

Kriteria pemberhentian adalah *Number of Improvisation* (NI) yang merepresentasikan jumlah iterasi.

2.2.2 Prosedur Umum Algoritma Genetika

Prosedur umum algoritma Genetika adalah sebagai berikut.

a. Membangkitkan populasi awal

Membangkitkan populasi awal adalah proses membangkitkan sejumlah kromosom sebanyak ukuran populasi secara *random* sehingga didapatkan solusi awal.

b. Evaluasi solusi

Proses evaluasi dilakukan untuk menghitung nilai *fitness* dari masing-masing kromosom dan mengevaluasinya hingga kriteria pemberhentian terpenuhi. Nilai ini membedakan kualitas dari kromosom untuk mengetahui seberapa baik kromosom-kromosom dalam populasi.

c. Membentuk generasi baru

Proses pembentukan generasi baru menggunakan tiga operator yaitu seleksi, *crossover*, dan mutasi. Proses ini dilakukan berulang-ulang hingga kriteria pemberhentian terpenuhi.

2.2.3 Komponen Algoritma Genetika

Komponen algoritma Genetika adalah sebagai berikut.

a. Representasi Kromosom

Merepresentasikan kromosom sebagai calon solusi suatu masalah dalam populasi awal dengan teknik pengkodean, teknik ini meliputi pengkodean gen dari kromosom. Gen merupakan bagian dari kromosom, dimana satu gen biasanya akan mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk nilai bit, pohon, *array* bilangan real, elemen permutasi, dan lain-lain.

Ada beberapa cara pengkodean, antara lain:

1) Pengkodean Biner

Pengkodean biner merupakan pengkodean yang sering digunakan. Setiap kromosom dalam pengkodean biner direpresentasikan dalam barisan bit 0 atau 1, seperti ditunjukkan pada Tabel 2.1 berikut.

Tabel 2.1 Contoh Pengkodean Biner

Kromosom	Pengkodean					
Kromosom 1	1	0	0	1	1	0
Kromosom 2	1	1	0	0	1	1

2) Pengkodean Permutasi

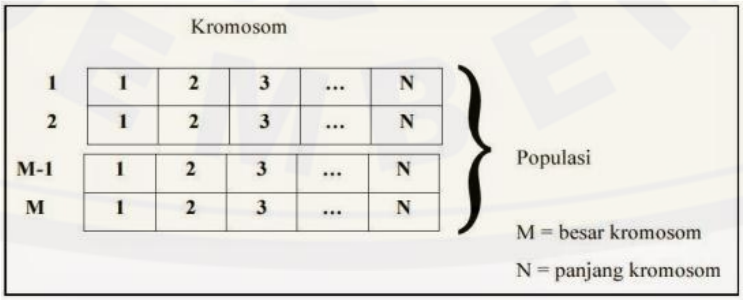
Pengkodean permutasi adalah pengkodean yang digunakan dalam masalah pengurutan data, *Travelling Salesman Problem* (TSP) atau masalah pengurutan tugas. Setiap kromosom dalam pengkodean permutasi merupakan barisan angka yang merepresentasikan angka dalam urutan. Sebagai contoh, dapat dilihat pada Tabel 2.2 berikut.

Tabel 2.2 Contoh Pengkodean Permutasi

Kromosom	Pengkodean					
Kromosom 1	4	6	2	1	5	3
Kromosom 2	2	3	1	4	6	5

b. Pembentukan Populasi Awal

Proses pada algoritma Genetika diawali dengan pembentukan populasi awal dimana populasi awal ini merupakan generasi awal yang akan digunakan untuk membangkitkan generasi-generasi selanjutnya. Contoh populasi dapat dilihat pada Gambar 2.1.



Gambar 2.1 Populasi

c. Fungsi *Fitness*

Fungsi *fitness* adalah fungsi yang akan digunakan untuk mengukur tingkat kebugaran suatu kromosom dalam populasi. Oleh sebab itu, fungsi *fitness* digunakan untuk mengevaluasi kromosom-kromosom pada setiap generasi. Kualitas sebuah kromosom di dalam suatu populasi ditunjukkan oleh besarnya nilai *fitness*. Semakin besar nilai *fitness* maka semakin bugar kromosom dalam populasi sehingga semakin besar kemungkinan kromosom tersebut dapat bertahan pada generasi berikutnya (Chen *et al*, 2003).

Jika masalahnya adalah meminimalkan fungsi *h(x)* (masalah minimasi), maka fungsi $h(x)$ tidak bisa digunakan secara langsung. Oleh sebab itu, nilai *fitness* dapat menggunakan persamaan $f(x) = \frac{1}{h(x)}$ yang artinya semakin kecil nilai *h(x)* maka semakin besar nilai *f(x)* nya. Tetapi jika solusi yang dicari dalam masalah optimasi adalah memaksimalkan sebuah fungsi *h(x)* (masalah maksimasi), maka nilai *fitness* yang digunakan adalah nilai dari fungsi *h(x)* tersebut, seperti ditunjukkan pada persamaan (2.3) berikut ini

$$f(x) = h(x) \tag{2.3}$$

dimana,

$f(x)$ = nilai *fitness*;

$h(x)$ = nilai optimasi;

x = kromosom.

d. Seleksi

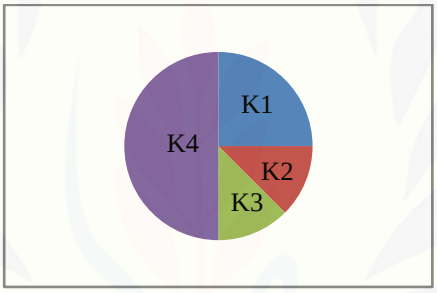
Salah satu teknik seleksi dalam algoritma Genetika adalah *roullete wheel* (roda *roullete*). Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang memiliki nilai *fitness* yang tinggi untuk bereproduksi.

Roda *roullete* ini dibagi menjadi bagian-bagian yang sama dengan jumlah populasi. Tiap bagian merupakan kromosom dengan ukuran tertentu yang sebanding dengan nilai *fitness*nya. Pemilihan kromosom untuk bereproduksi

dilakukan dengan memutar roda *roullete* sebanyak *N* kali, dimana *N* adalah ukuran populasi. Proses pemilihan dengan roda *roullete* ditunjukkan dengan contoh, seperti yang terlihat pada Gambar 2.2.

Tabel 2.3 Contoh Nilai *Fitness*

Kromosom	*Fitness*
K1	2
K2	1
K3	1
K4	4
Jumlah	8



Gambar 2.2 Roda *Roullete* dari Tabel 2.3

Langkah-langkah seleksi yang dilakukan dengan metode roda *roullete* adalah sebagai berikut:

1) Menghitung total *fitness* dari semua kromosom;

$$total\ fitness = \sum_{i=1}^{n} h\,(x) \tag{2.4}$$

2) Menghitung probabilitas relatif (p_k) setiap kromosom dengan

$$p_k = \frac{fitness\ (k)}{total\ fitness} \tag{2.5}$$

3) Menghitung probabilitas kumulatif (q_k) dengan

$$q_k = q_{k-1} + p_k \text{ dan } q_0 = 0 \tag{2.6}$$

4) Menciptakan nilai *random* (r_i), $0 \leq r_i \leq 1$, sebanyak ukuran populasi;

5) Memilih kromosom ke-*k* untuk dijadikan *parent* pada proses *crossover* dengan syarat $q_{k-1} < r_i < q_k$ dengan $k, i = 1,2,3, \dots, N$.

Kromosom yang terpilih dari proses seleksi akan melewati proses genetika yaitu *crossover* dan mutasi.

e. Pindah Silang (*Crossover*)

Menurut Chen *et al* (2003), *crossover* akan menukar informasi genetik antara dua kromosom *parent* yang terpilih dari proses seleksi untuk membentuk dua *offspring*. Operator *crossover* bekerja pada sepasang kromosom *parent* untuk menghasilkan dua kromosom *offspring* dengan menukarkan beberapa gen yang dimiliki masing-masing kromosom *parent*.

Proses *crossover* akan terjadi pada sepasang kromosom jika suatu bilangan yang dibangkitkan secara *random* (r), $0 \leq r \leq 1$, nilainya kurang dari atau sama dengan Pc. Bilangan *random* tersebut dibangkitkan setiap kali akan melakukan *crossover* pada kromosom.

Metode *crossover* yang digunakan yaitu *crossover* satu titik atau *crossover* banyak titik. *Crossover* satu titik dan banyak titik biasanya digunakan untuk representasi kromosom dalam biner. Pada *crossover* satu titik, posisi *crossover* k $(k = 1,2, \dots, N-1)$ dengan *N*=panjang kromosom diseleksi secara random. Pada *crossover* banyak titik, *m* posisi penyilangan k_i $(k = 1,2, \dots, N-1, i = 1,2, \dots, m)$ dengan N=panjang kromosom diseleksi secara random. Gen-gen ditukar antar kromosom pada titik tersebut untuk menghasilkan anak.

Tabel 2.4 Contoh *Parent* 1 dan 2 dengan *Crossover* Satu Titik

Kromosom	Pengkodean				
Parent 1	1	0	**1**	**0**	**1**
Parent 2	1	1	**0**	**1**	**0**

Tabel 2.5 Contoh *Offspring* 1 dan 2 Hasil *Crossover* Satu Titik

Kromosom	Pengkodean				
Offspring 1	1	0	**0**	**1**	**0**
Offspring 2	1	1	**1**	**0**	**1**

Tabel 2.6 Contoh *Parent* 1 dan 2 dengan *Crossover* Dua Titik

Kromosom	Pengkodean				
Parent 1	1	0	**1**	**0**	1
Parent 2	1	1	**0**	**1**	0

Tabel 2.7 Contoh *Offspring* 1 dan 2 Hasil *Crossover* Dua Titik

Kromosom	Pengkodean				
Offspring 1	1	0	**0**	**1**	1
Offspring 2	1	1	**1**	**0**	0

f. Mutasi

Mutasi berperan untuk menempatkan kembali kromosom-kromosom dari populasi akibat proses seleksi dan memunculkan kromosom-kromosom yang belum pernah ada pada populasi sebelumnya (Gen & Cheng, 1997).

Proses mutasi akan terjadi pada suatu gen, jika suatu bilangan yang dibangkitkan secara *random* (r), $0 \leq r \leq 1$, nilainya kurang dari atau sama dengan Pm. Metode mutasi diantaranya sebagai berikut.

1) Mutasi dalam pengkodean biner

Mutasi pada pengkodean biner dilakukan dengan mengubah nilai bit pada posisi tertentu yang dipilih secara *random* atau menggunakan skema tertentu pada kromosom, yang disebut dengan *inverse* bit, seperti pada Tabel 2.8 berikut ini.

Tabel 2.8 Contoh Mutasi pada Pengkodean Biner

Kromosom	Pengkodean					
Parent	1	0	0	1	1	0
Offspring	1	1	0	0	1	1

2) Mutasi dalam pengkodean permutasi

Mutasi pada pengkodean permutasi dilakukan dengan memilih dua posisi gen dari kromosom dan kemudian nilainya saling dipertukarkan.

Tabel 2.9 Contoh Mutasi pada Pengkodean Permutasi

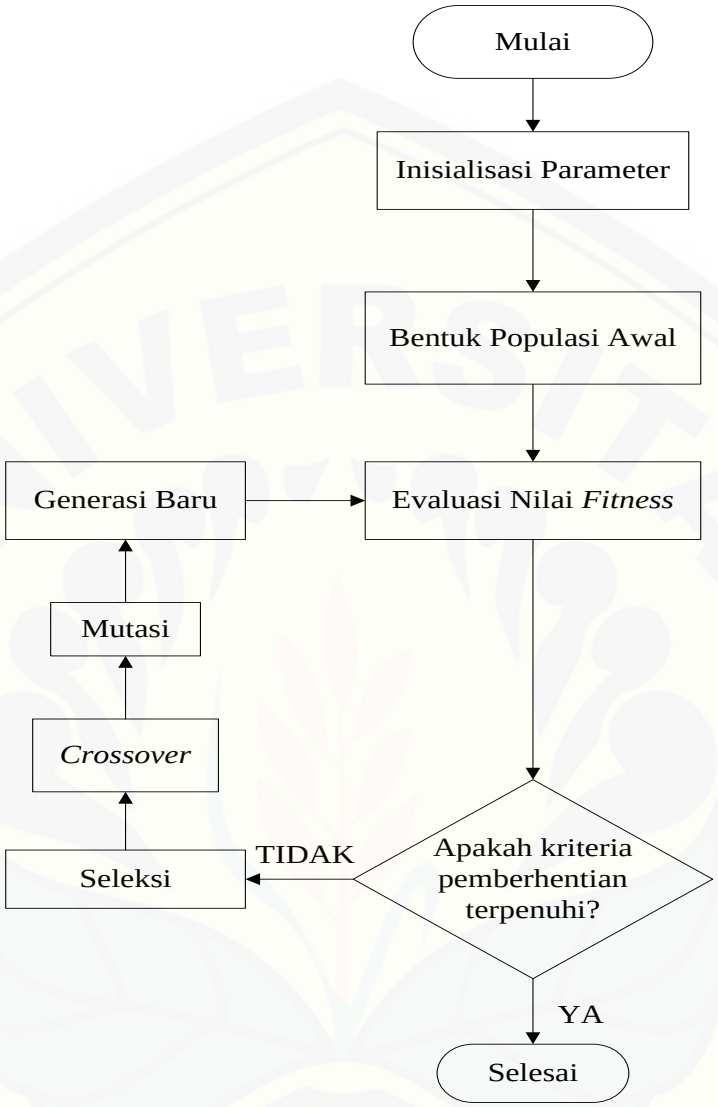
Kromosom	Pengkodean					
Parent	4	6	2	1	5	5
Offspring	2	3	2	4	6	5

Proses mutasi yang dilakukan dalam pengkodean biner dengan mengubah langsung gen-gen pada kromosom tidak dapat dilakukan pada pengkodean permutasi, karena konsistensi urutan permutasi harus diperhatikan.

g. Kriteria Pemberhentian

Proses algoritma Genetika akan terus menerus berlangsung hingga kriteria pemberhentian terpenuhi. Kriteria pemberhentian dalam algoritma Genetika adalah jumlah generasi (iterasi) yang diinginkan sudah tercapai.

Skema proses algoritma Genetika dapat digambarkan seperti pada Gambar 2.3.



Gambar 2.3 *Flowchart* Algoritma Genetika

2.3 Algoritma *Harmony Search* (HS)

Harmony Search (HS) pertama kali diperkenalkan oleh Geem pada tahun 2001. Ide dasar algoritma HS adalah meniru proses perbaikan harmoni musik yang dilakukan oleh kelompok pemain musik (Santosa dan Willy, 2011). Ketika kelompok pemain musik melakukan perbaikan pada harmoni musik yang dimainkan, maka ada tiga kemungkinan pilihan, antara lain memainkan musik yang terkenal sesuai ingatan

mereka, memainkan harmoni musik yang serupa dengan memainkan harmoni musik yang terkenal namun ada sedikit penyesuaian, atau membuat harmoni musik yang baru. Teknik ini menggunakan proses pencarian seperti proses improvisasi musik yang bertujuan untuk mendapatkan keadaan terbaik berdasarkan perkiraan estetika.

Dengan analogi tersebut, HS melakukan proses optimasi untuk mendapatkan keadaan terbaik dengan cara mengevaluasi fungsi objektif. Seperti perkiraan estetika yang ditentukan menggunakan himpunan pola-pola nada (*pitch*) yang dikeluarkan oleh alat-alat musik, fungsi objektif pada HS dihitung menggunakan himpunan nilai-nilai yang diberikan untuk variabel-variabel keputusan. Jika kualitas suara estetika dapat diperbaiki melalui latihan demi latihan, maka nilai fungsi objektif juga dapat terus ditingkatkan dari iterasi ke iterasi.

Ada beberapa strategi yang bisa digunakan untuk mendapatkan harmonisasi yang baik. Pada pertunjukan musik yang dilakukan oleh para musisi jazz, biasanya mereka memiliki tiga pilihan untuk melakukan improvisasi.

a. Memainkan nada yang sering dimainkan, yang merupakan karakteristik dari bagian musik tersebut. Dalam musik, nada ini dikenal dengan sebutan nada dasar. Pada dasarnya semua musisi mengerti nada dasar tersebut dan dapat memainkan nada tersebut dengan sendirinya, dengan kata lain nada-nada dasar ini sudah tersimpan dalam memori tersebut.
b. Memainkan nada yang serupa nada dasar. Biasanya musisi memperkaya musik dengan menaikkan atau menurunkan nada dasar sesuai dengan eksplorasi mereka. Hal ini akan menyebabkan permainan musik serupa dengan variasi nada yang berbeda.
c. Memainkan nada sembarang, yang memberikan musisi kebebasan untuk berimprovisasi. Dengan cara seperti ini, ada kemungkinan not yang dihasilkan sedikit atau tidak memiliki hubungan dengan lagu yang ditampilkan. Hal ini membutuhkan bakat dan imajinasi dari musisi, yang membuatnya menjelajahi dunia musik dan memberikan nuansa baru pada musik dengan tema-tema segar dan baru.

Dari tiga pilihan berimprovisasi tersebut, para peneliti memiliki strategi untuk mendapatkan harmonisasi yang baik. Hal ini diadaptasi, sehingga setiap variabel keputusan dalam mengambil sebuah nilai dalam algoritma HS harus berpedoman pada tiga aturan ini (Geem *et al*, 2005).

a. Mengambil nilai yang sudah terdapat pada memori harmoni, biasanya didefinisikan sebagai *memory considerations*.
b. Mengambil nilai yang berdekatan dari satu nilai pada harmoni, biasanya didefinisikan sebagai *pitch adjustments*.
c. Mengambil nilai acak dari rentang nilai yang mungkin, biasanya didefinisikan sebagai *randomization*.

Tiga aturan ini nantinya akan dibutuhkan dalam menyusun algoritma HS yang diwakilkan oleh beberapa parameter. Secara umum, parameter-parameter ini akan sangat mempengaruhi kinerja dari HS dalam mendapatkan solusi yang optimal.

Langkah-langkah yang digunakan dalam melakukan algoritma HS menurut Geem dan Williams (2007) adalah sebagai berikut ini.

a. Langkah pertama dari algoritma HS adalah inisialisasi masalah dan parameter algoritma. Sebelum membahas parameter HS, perlu diformulasikan masalah optimasi sebagai berikut:

$$\min\{f(x)|x \in X\} \text{ atau } \max\{f(x)|x \in X\} \quad (2.7)$$

dengan $x_i \in X_i, i = 1,2,\dots,N; BB(i) \leq X_i \leq BA(i)$

dimana:

$f(x)$ = fungsi objektif;
x_i = himpunan dari setiap variabel keputusan ke-*i*;
X_i = himpunan dari rentang nilai yang mungkin untuk setiap variabel keputusan;
N = jumlah variabel keputusan;
$BB(i)$ = batas bawah variabel keputusan ke-*i*;
$BA(i)$ = batas atas variabel keputusan ke-*i*.

Berikut adalah parameter yang digunakan secara umum dalam HS (Geem dan Williams, 2007).

1) *Harmony Memory Size* (HMS) adalah jumlah dari kumpulan *harmony memory*, bisa disebut sebagai jumlah populasi;
2) *Harmony Memory Consideration Rate* (HMCR) adalah probabilitas dari *harmony memory* untuk digunakan kembali sebagai hasil dari vektor solusi. Parameter ini memiliki rentang nilai dari 0 sampai 0,99. Parameter ini tidak menggunakan nilai 1, untuk mencegah terjadinya stagnansi nilai (range nilai tidak berubah dan perbaikan minimal). Menurut Setiawan (2010) parameter yang sering digunakan biasanya berkisar antara 0,7 sampai 0,95 karena jika nilai HMCR terlalu kecil maka hanya sedikit harmoni bagus yang terpilih dan juga dapat menyebabkan proses konvergensi terlalu lambat. Jika nilai HMCR terlalu besar maka akan menyebabkan vektor solusi pada *harmony memory* banyak terpakai dan tidak sempat mengeksplorasi yang lain, dimana pada akhirnya sulit mencapai solusi yang bagus.
3) *Pitch Adjustment Rate* (PAR) adalah parameter yang mempunyai peran signifikan dalam menentukan jumlah nilai yang harus diubah, disesuaikan, atau ditukar dengan nilai yang lain. Parameter ini memiliki rentang dari nol sampai satu.
4) *Bandwidth* (bw) adalah jumlah perubahan maksimal dalam penyesuaian, biasanya yang sering digunakan adalah 0,1 dan 0,2 (Setiawan, 2010).
5) *Number of Improvisations* (NI) adalah parameter yang digunakan sebagai kriteria pemberhentian pengulangan, biasanya berapa jumlah iterasi untuk mendapatkan nilai yang optimal.

b. Langkah kedua dari algoritma HS adalah inisialisasi *Harmony Memory* (HM). Pada tahap ini dibangkitkan matriks *harmony memory* secara *random* yang berisi vektor-vektor solusi sebanyak HMS. Bentuk HM bila digambarkan dalam matriks ditunjukkan pada Gambar 2.4.

$$HM = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \cdots & \vdots \\ x_{HMS,1} & x_{HMS,2} & \cdots & x_{HMS,N} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{HMS}) \end{bmatrix}$$

Gambar 2.4 *Harmony Memory*

Untuk setiap x_t dengan $t \in \{1 \dots HMS\}$, perlu ditentukan nilai fungsi objektifnya $f(x_t)$, agar bisa diketahui mana himpunan yang memiliki nilai terbaik dan nilai terburuk. Dalam prosesnya, himpunan dengan nilai terbaik akan menjadi solusi yang digunakan, sedangkan himpunan dengan nilai terburuk akan tereliminasi dan digantikan dengan himpunan yang nilainya lebih baik selama perulangan berlangsung.

c. Langkah ketiga dari algoritma HS adalah improvisasi harmoni baru dengan membangkitkan harmoni baru x_i sehingga membentuk vektor solusi baru $x' = x_1', x_2', \dots, x_N'$. Berikut adalah proses pembangkitan vektor solusi yang baru (Setiawan, 2010).

   1) Apabila bilangan *random* yang dibangkitkan di atas HMCR maka akan dibangkitkan variabel keputusan baru secara *random* dengan formulasi sebagai berikut:

$$x_i' = rand[BB(i), BA(i)] \quad (2.8)$$

   dimana,

   x_i' = variabel keputusan ke-*i*;

   $BB(i)$ = batas bawah variabel keputusan ke-*i*;

   $BA(i)$ = batas atas variabel keputusan ke-*i*.

   2) Apabila bilangan *random* yang dibangkitkan kedua kalinya di atas PAR, maka variabel keputusan ke-*i* akan diambil dari *harmony memory* dengan formulasi sebagai berikut:

$$d_1 = int[1 + (HMS - 1)rand] \quad (2.9)$$

$$d_2 = HM(d_1, i) \quad (2.10)$$

$$x_i' = d_2 \quad (2.11)$$

dimana,

d_1 = nilai yang menyatakan pemilihan lokasi pada *harmony memory* secara *random*;

int = *integer*;

d_2 = nilai variabel keputusan yang diambil dari *harmony memory*.

3) Apabila bilangan *random* yang dibangkitkan di bawah HMCR dan bilangan *random* yang dibangkitkan kedua kalinya di bawah PAR, maka terdapat penyesuaian terhadap variabel keputusan ke-*i* yang telah diambil dari *harmony memory* dengan formulasi sebagai berikut:

$$d_3 = d_2 + bw \times \varepsilon \quad (2.12)$$

$$x_i^{'} = d_3 \quad (2.13)$$

dimana,

d_3 = nilai variabel keputusan setelah dilakukan penyesuaian;

bw = jumlah perubahan maksimal dalam penyesuaian;

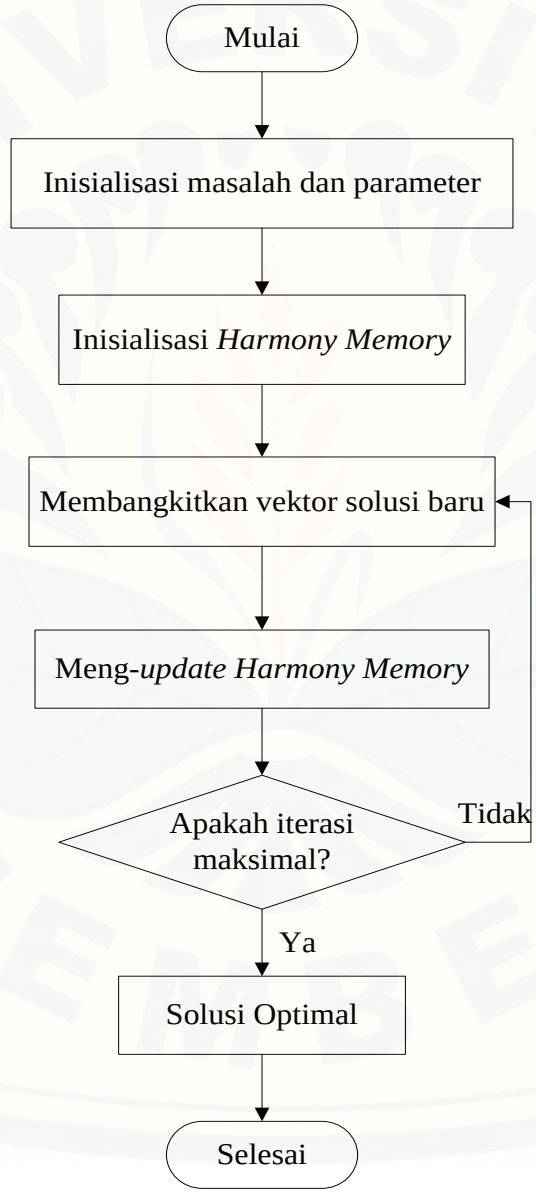
ε = bilangan *random* antara [-1,1].

Kemudian setelah dilakukan proses penyesuaian maka akan dilakukan pengecekan terhadap batas atas dan batas bawah dari variabel keputusan. Apabila $d_3 < BB$ maka $d_3 = BB$, lalu apabila $d_3 > BA$ maka $d_3 = BA$. Batas bawah dari semua variabel keputusan adalah 0 dan batas atasnya adalah 1.

d. Langkah keempat dari algoritma HS adalah meng-*update harmony memory*. Apabila nilai vektor solusi yang baru mempunyai nilai maksimal daripada nilai vektor solusi minimal yang sudah ada pada *harmony memory* dilihat dari sudut pandang nilai fungsi tujuan, maka vektor solusi yang baru akan dimasukkan ke dalam *harmony memory* dan vektor solusi minimal pada *harmony memory* akan dikeluarkan. Apabila nilai vektor solusi yang baru lebih kecil daripada nilai vektor solusi minimal pada *harmony memory*, maka tidak akan terjadi perubahan pada *harmony memory*.

e. Langkah kelima dari algoritma HS adalah mengecek kriteria pemberhentian. Apabila kriteria pemberhentian yang dalam hal ini adalah nilai maksimal telah tercapai maka iterasi dihentikan, apabila belum tercapai maka kembali ke langkah tiga. Kriteria pemberhentian yang digunakan adalah jumlah iterasi maksimal.

Secara umum, prosedur algoritma *Harmony Search* dapat dilihat pada Gambar 2.5.



Gambar 2.5 *Flowchart* Algoritma *Harmony Search*

2.4 Kriteria Kekonvergenan

Menurut Sivanandam & Deepa (2008) pemberhentian atau kriteria kekonvergenan suatu algoritma dapat dituliskan sebagai berikut.

a. Iterasi maksimum

   Algoritma akan berhenti ketika iterasi maksimum sudah tercapai.

b. Batas waktu maksimum

   Proses algoritma akan berhenti ketika batas waktu maksimum telah terlampaui. Jika iterasi maksimum sudah tercapai sebelum batas waktunya terpenuhi maka prosesnya akan berhenti.

c. Nilai *fitness*/fungsi objektif tidak terjadi perubahan

   Proses algoritma akan berhenti jika tidak ada perubahan pada nilai *fitness* atau nilai fungsi objektif terbaiknya.

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Sebagai bahan simulasi, peneliti menggunakan data pada skripsi Ridho (2014), yaitu data sejumlah barang yang akan dibeli oleh UD. Permata Indah, yaitu usaha dagang dibidang obat-obat pertanian yang terletak di Jalan Raya Mlandingan, Kecamatan Mlandingan, Kabupaten Situbondo. Data yang diambil adalah data yang berupa harga beli barang, harga jual barang, banyaknya barang yang dibeli, serta kapasitas angkut maksimumnya, seperti disajikan pada Lampiran A. Data tersebut digunakan karena memenuhi persoalan *knapsack* 0-1.

3.2 Langkah-langkah Penelitian

Langkah-langkah yang dilakukan dalam menyelesaikan masalah *knapsack* 0-1 dengan menggunakan algoritma Genetika dan algoritma *Harmony Search* dapat dilihat pada Gambar 3.1.

Adapun penjelasan dari Gambar 3.1 adalah sebagai berikut.

a. Studi Literatur

   Langkah awal yang dilakukan adalah mengumpulkan berbagai macam teori dari sejumlah literatur yang berkaitan dengan algoritma Genetika dan algoritma *Harmony Search*.

b. Pengambilan Data

   Pada langkah ini dilakukan pengambilan data yaitu pada skripsi Ridho (2014).

c. Mengidentifikasi Data

   Pada langkah ini data diidentifikasi untuk mencari keuntungan (p_i) dan berat (w_i) pada masing-masing barang.

d. Menerapkan algoritma Genetika dan algoritma *Harmony Search* untuk menyelesaikan masalah *knapsack* 0-1

1) Algoritma Genetika

a) Inisialisasi algoritma Genetika dan parameter-parameter permasalahan optimasi, dalam langkah ini permasalahan *knapsack* 0-1 direpresentasikan sebagai suatu kromosom dimana di dalam suatu kromosom terdapat sejumlah gen yang mengkodekan informasi yang disimpan dalam kromosom. Pada penelitian ini menggunakan teknik pengkodean dalam bentuk *string bit*/*varchar* yang digunakan dalam pemrograman genetika. Fungsi tujuan dari permasalahan *knapsack* 0-1 ini adalah memaksimalkan keuntungan dengan meminimasi berat agar dapat disimpan dalam media penyimpanan, seperti yang ditunjukkan pada persamaan (2.1) dan (2.2). Selain itu parameter-parameter yang akan digunakan dalam algoritma Genetika diantaranya ukuran populasi, probabilitas *crossover* (Pc), probabilitas mutasi (Pm), dan NI. Dimana parameter-parameter ini ditentukan pada langkah pertama.

b) Membentuk populasi awal adalah proses membangkitkan sejumlah kromosom secara acak (*random*) dimana setiap kromosom menyatakan alternatif solusi yang mungkin.

c) Menghitung nilai fungsi *fitness* pada permasalahan *knapsack* 0-1, bisa dimulai dengan tujuan yang ingin dicapai, yaitu menemukan sejumlah barang yang total nilainya paling besar (maksimasi) dan total beratnya kurang atau sama dengan total berat yang diijinkan. Perhitungan nilai fungsi fitness menggunakan persamaan (2.1) dan (2.3).

d) Langkah selanjutnya adalah melakukan pengecekan kriteria pemberhentian. Apabila kriteria pemberhentian telah tercapai maka iterasi diberhentikan, apabila belum tercapai akan dibentuk generasi baru dengan menggunakan operator-operator algoritma Genetika yaitu seleksi, *crossover*, dan mutasi. Kriteria pemberhentian yang digunakan adalah jumlah iterasi maksimal.

e) Apabila kriteria pemberhentian belum tercapai, maka dilanjutkan dengan proses seleksi menggunakan teknik seleksi roda *roullete*. Langkah-langkah seleksi yang dilakukan dengan metode roda *roullete* adalah sebagai berikut.
   (1) Menghitung total *fitness* dari semua kromosom menggunakan persamaan (2.4).
   (2) Menghitung probabilitas relatif setiap kromosom dengan menggunakan persamaan (2.5) dan menghitung probabilitas kumulatif (q_k) dengan menggunakan persamaan (2.6).
   (3) Membangkitkan bilangan *random* (r_i) sebanyak jumlah kromosom, lalu memeriksa setiap nilai *random* yang dibangkitkan.

f) Langkah selanjutnya adalah proses *crossover*. Untuk melakukan proses *crossover*, langkah-langkahnya sebagai berikut.
   (1) Memilih kromosom yang akan dijadikan sebagai *parent* dengan memilih kromosom yang memiliki nilai *random* $(r_i) \leq Pc$
   (2) Memilih gen pada kromosom *parent* dengan aturan *random*, kemudian menukar gen-gen antar *parent*.
   (3) Membentuk kromosom *offspring* hasil pertukaran gen-gen antar *parent*.

g) Langkah selanjutnya adalah proses mutasi. Langkah-langkah proses mutasi adalah sebagai berikut.
   (1) Memilih kromosom yang akan dijadikan *parent* dengan memilih kromosom yang memiliki nilai *random* $(r_i) \leq Pm$.
   (2) Memilih gen pada kromosom *parent* dengan aturan *random* kemudian menukar gen hingga terbentuk *offspring*.

   Setelah generasi baru terbentuk, maka kembali ke langkah c, untuk mengevaluasi nilai fungsi fitness. Setelah itu untuk iterasi selanjutnya adalah membuat generasi baru lagi hingga kriteria pemberhentian terpenuhi yaitu jumlah iterasi maksimal sudah tercapai.

2) Algoritma *Harmony Search*

a) Inisialisasi masalah dan parameter algoritma HS, pada tahap ini semua variabel pada data diinisialisasikan seperti persamaan 2.7. Selain itu, parameter-parameter yang akan digunakan dalam algoritma HS juga ditentukan dalam langkah pertama. Parameter-parameter HS diantaranya adalah HMS, HMCR, PAR, bw, dan NI.

b) Inisialisasi *Harmony Memory* (HM), pada tahap ini dibangkitkan matriks HM secara *random* yang berisi vektor-vektor solusi sebanyak HMS. Vektor-vektor yang terdiri dari bilangan *random* antara 0 sampai 1, yang didefinisikan dengan x_i dalam hal ini adalah urutan barang pada data. Kemudian dilakukan pemilihan untuk mencari nilai maksimum sesuai dengan prosedur pada algoritma HS. Pemilihan ini yaitu berdasarkan urutan bilangan acak, dimana bilangan yang bernilai besar akan dipilih sedangkan yang bernilai paling kecil tidak dipilih. Hasil dari pemilihan ini kemudian diimplementasikan dalam bentuk matriks (0,1).

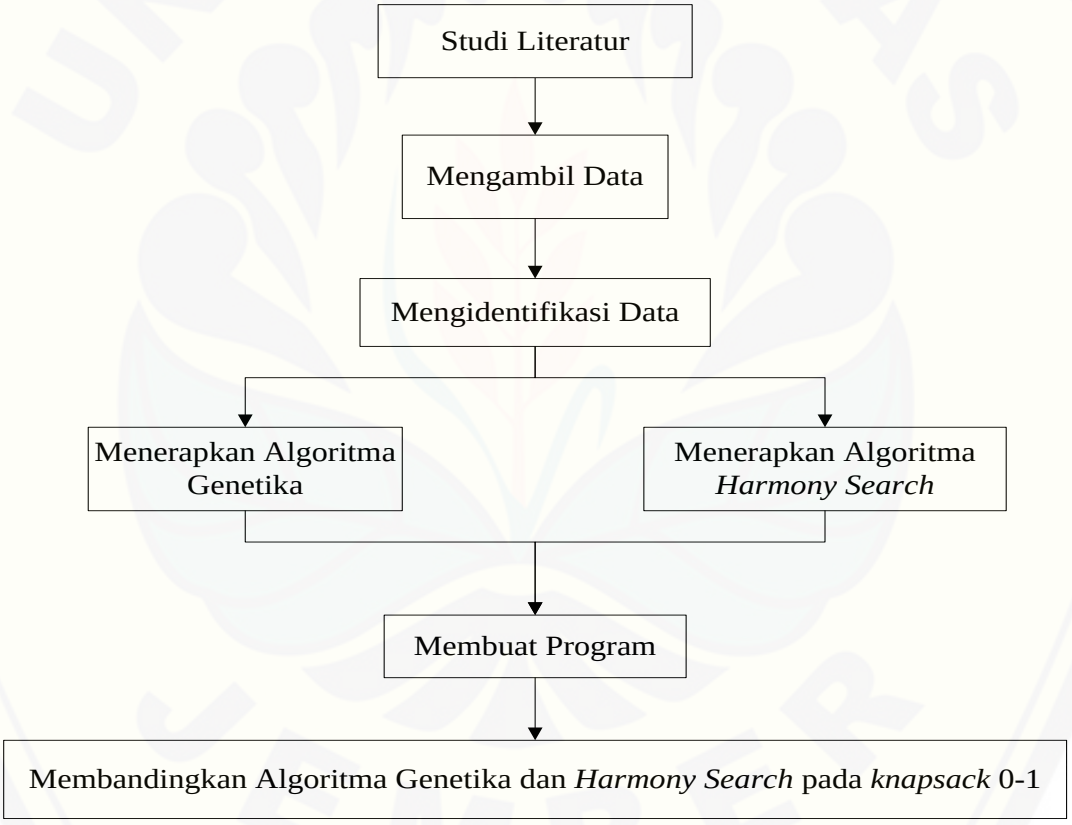
c) Setelah dilakukan pemilihan barang, tahap berikutnya adalah improvisasi harmoni baru. Pada tahap ini kita pilih vektor solusi yang nilainya paling minimum untuk dicari vektor solusi yang baru dengan tujuan mengubah pemilihan barang dengan membandingkan nilai yang dibangkitkan secara *random* dengan HMCR dan PAR.

d) Setelah didapat vektor solusi yang baru, langkah selanjutnya adalah memperbarui HM. Vektor solusi yang baru kemudian dibandingkan dengan vektor solusi yang lama, apabila vektor solusi yang baru lebih baik maka vektor solusi yang lama digantikan dengan vektor solusi yang baru. Apabila vektor solusi yang baru lebih buruk, maka tidak ada pergantian, kemudian ulangi langkah b dan c.

e) Langkah selanjutnya adalah dilakukan pengecekan kriteria pemberhentian. Apabila kriteria pemberhentian telah tercapai, maka iterasi diberhentikan,

apabila belum tercapai maka kembali ke langkah c. Kriteria pemberhentian yang digunakan adalah jumlah iterasi maksimal.

e. Langkah penelitian selanjutnya adalah pembuatan program dengan menggunakan *software* Matlab. Pada langkah ini, penulis akan membuat *script* dan desain program berdasarkan metode yang digunakan.
f. Langkah yang terakhir adalah membandingkan kedua algoritma berdasarkan nilai optimal yang dicapai, *running time* yang diperlukan dalam perhitungan, dan konvergensi dari setiap algoritma. Sehingga dapat dipilih algoritma mana yang lebih baik penggunaannya pada permasalahan *knapsack* 0-1.



Gambar 3.1 Skema Langkah-langkah Penelitian

BAB 4. HASIL DAN PEMBAHASAN

Pada bab ini dibahas hasil dari penerapan algoritma Genetika dan algoritma *Harmony Search* pada permasalahan *knapsack* 0-1 dengan menggunakan program yang telah dibuat dengan pemrograman Matlab.

4.1 Hasil

Data pada Lampiran A merupakan data harga beli barang, harga jual barang, dan jumlah barang yang dibeli oleh UD. Permata Indah. Kapasitas angkut maksimum (M) sebesar 1000 kg atau 1 ton. Berdasarkan data tersebut, peneliti menghitung berat barang (w_i) dan profit (p_i) dari masing-masing barang pada Lampiran. Untuk mencari nilai profit (p_i), diperoleh dari perhitungan selisih harga beli dan harga jual yang ditetapkan. Untuk mencari berat total barang (w_i), diperoleh dari perhitungan perkalian berat dari tiap barang dengan jumlah barang yang dibeli. Berikut ini merupakan data hasil perhitungan berat barang (w_i) dan profit (p_i) dari Lampiran.

Tabel 4.1 Data Berat dan Keuntungan

No.	Nama Barang (i)	Jumlah Barang	Berat Barang (kg) (w_i)	Profit (Rp) (p_i)
1.	CIHERANG 10kg	5	50	20000
2.	PADI 64 10kg	5	50	25000
3.	WAY APU 10kg	5	50	25000
4.	TOWATI 10kg	5	50	25000
5.	SITUBAGEDID 10kg	5	50	20000
6.	BISI 2 20kg	3	60	135000
7.	P21 20kg	3	60	180000
8.	P27 20kg	3	60	120000
9.	PAC 224 20kg	3	60	120000
10.	TM BA 1kg	2	2	50000
11.	TM JAWARA 1kg	2	2	48000
12.	TM PANAMAS 1kg	2	2	40000
13.	PURADON 40kg	3	120	165000
14.	REAGENT 10kg	4	40	80000
15.	REAGENT CAIR 1kg	5	5	175000

No.	Nama Barang (i)	Jumlah Barang	Berat Barang (kg) (w_i)	Profit (Rp) (p_i)
16.	SEDRIK 5kg	5	25	475000
17.	SEDRIK 8kg	5	40	400000
18.	DURSBAN 1kg	5	5	125000
19.	DURSBAN 2.5kg	5	12.5	150000
20.	DURSBAN 5kg	5	25	175000
21.	DURSBAN 10kg	1	10	50000
22.	URACRON 1kg	5	5	175000
23.	URACRON 2.5kg	5	12.5	200000
24.	FASTACT 1kg	5	5	75000
25.	FASTACT 2.5kg	5	12.5	110000
26.	FASTACT 4kg	5	20	150000
27.	POLIDOR 1kg	5	5	100000
28.	POLIDOR 2kg	5	10	125000
29.	POLIDOR 4kg	5	20	175000
30.	GAMATOC 1kg	5	5	150000
31.	GAMATOC 2.5kg	5	12.5	200000
32.	AKODAN 1kg	5	5	150000
33.	DARMAIBAS 1kg	5	5	75000
34.	ASUDRIN 1kg	5	5	125000
35.	ASUDRIN 5kg	5	25	200000
36.	DIPUSTAR 2kg	5	10	100000
37.	DIPUSTAR 4kg	5	20	200000
38.	SPONTAN 2kg	5	10	150000
39.	SPONSOR 2kg	5	10	175000
40.	SEPIN 2kg	5	10	125000
41.	CONDIFOR 2kg	5	10	250000
42.	GRANTONIC 10kg	1	10	80000
43.	GRANTONIC 20kg	1	20	180000
44.	SEPLASH 10kg	1	10	80000
45.	SEPLASH 20kg	1	20	180000
46.	PEREKAT 10kg	1	10	120000
47.	PEREKAT 20kg	1	20	130000

4.1.1 Perhitungan

Untuk memudahkan perhitungan, penulis menggunakan sampel data, yaitu 5 barang yang mewakili keseluruhan barang seperti ditunjukkan pada Tabel 4.2. Kapasitas maksimum untuk data sampel data ini adalah 130 kg.

Tabel 4.2 Sampel Data Barang

No.	Nama Barang (i)	Jumlah Barang	Berat Barang (kg) (w_i)	Profit (Rp) (p_i)
1.	Ciherang 10 kg	5	50	20000
2.	BISI 2 20 kg	3	60	135000
3.	Sedrik 5 kg	5	25	475000
4.	Polidor 4 kg	5	20	175000
5.	Asudrin 1 kg	5	5	125000

a. Penyelesaian Masalah *Knapsack* 0-1 dengan Algoritma Genetika

Langkah-langkah penyelesaian masalah *knapsack* 0-1 untuk memaksimalkan keuntungan serta meminimasi berat agar dapat disimpan dalam media penyimpanan dengan Algoritma Genetika adalah sebagai berikut.

1) Inisialisasi parameter algoritma. Pada langkah pertama ini, semua parameter harus ditentukan, penentuan ini berdasarkan penjelasan pada bab 2. Parameter-parameter tersebut diantaranya sebagai berikut.
   a) Ukuran populasi (*n*) yang digunakan adalah 5;
   b) Probabilitas *crossover* (*Pc*) bernilai 0,6;
   c) Probabilitas mutasi (*Pm*) bernilai 0,02;
   d) Jumlah generasi/jumlah iterasi maksimum yang digunakan adalah 100.
2) Langkah kedua dari algoritma Genetika yaitu membentuk populasi awal. Pada tahap ini dibangkitkan kromosom sebanyak *n* populasi secara *random* seperti ditunjukkan pada Tabel 4.3.

Tabel 4.3 Populasi Awal

Kromosom (k)	Pengkodean				
K_1	1	1	0	1	0
K_2	1	0	1	1	1
K_3	0	1	1	1	1

Kromosom (k)	Pengkodean				
K_4	0	1	0	1	1
K_5	1	0	1	0	1

3) Langkah ketiga dari algoritma Genetika yaitu menghitung nilai fungsi *fitness*, yaitu dengan menggunakan persamaan (2.1) dan (2.3). Tabel 4.4 merupakan hasil perhitungan nilai fungsi *fitness* masing-masing kromosom dan berat totalnya.

Tabel 4.4 Nilai Fungsi *Fitness* dan Berat Total

Kromosom (k)	Nilai *Fitness* (Rp)	Berat Total (kg)
K_1	330000	130
K_2	795000	100
K_3	910000	110
K_4	435000	85
K_5	620000	80

4) Langkah selanjutnya adalah melakukan pengecekan kriteria pemberhentian. Apabila kriteria pemberhentian yaitu jumlah iterasi maksimum telah tercapai, maka iterasi dihentikan, namun apabila belum tercapai akan dibentuk generasi baru menggunakan operator-operator algoritma Genetika yaitu seleksi, *crossover*, dan mutasi.
5) Apabila kriteria pemberhentian belum tercapai, maka dilanjutkan dengan proses seleksi menggunakan teknik seleksi roda *roullete*. Langkah-langkah seleksi yang dilakukan dengan metode roda *roullete* adalah sebagai berikut.
   a) Menghitung total *fitness* dari semua kromosom menggunakan persamaan (2.4). Total *fitness* pada Tabel 4.4 sebesar Rp 3.090.000.
   b) Menghitung probabilitas relatif setiap kromosom dengan menggunakan persamaan (2.5) dan menghitung probabilitas kumulatif (q_k) dengan

menggunakan persamaan (2.6). Hasil perhitungan tersebut disajikan pada Tabel 4.5.

Tabel 4.5 Hasil Perhitungan Probabilitas Relatif dan Kumulatif

Kromosom (k)	Probabilitas Relatif (p_k)	Probabilitas Kumulatif (q_k)	Interval
K_1	0.107	0.107	0 – 0.107
K_2	0.257	0.364	0.107 – 0.364
K_3	0.294	0.658	0.364 – 0.658
K_4	0.141	0.799	0.658 – 0.799
K_5	0.201	1	0.799 – 1

c) Membangkitkan bilangan *random* (r_i) sebanyak jumlah kromosom yaitu lima, lalu memeriksa setiap nilai *random* yang dibangkitkan, seperti ditunjukkan pada Tabel 4.6.

Tabel 4.6 Kromosom Terpilih dari Proses Seleksi

r_i	Kromosom yang Terpilih
0.6324	K_3
0.0975	K_1
0.2785	K_2
0.5469	K_3
0.9575	K_5

6) Langkah selanjutnya adalah proses *crossover*. Untuk melakukan proses *crossover*, langkah-langkahnya sebagai berikut.

a) Memilih kromosom yang akan dijadikan sebagai *parent* dengan memilih kromosom yang memiliki nilai *random* (r_i) $\leq Pc$ seperti pada Tabel 4.7.

Tabel 4.7 Kromosom Terpilih untuk *Crossover*

r_i	Kromosom Terpilih pada Seleksi	Kromosom Terpilih untuk *Crossover*
0.8003	K_3	TIDAK
0.1419	K_1	*Parent* 1
0.4218	K_2	*Parent* 2
0.9157	K_3	TIDAK
0.7922	K_5	TIDAK

b) Memilih posisi gen pada kromosom *parent* dengan aturan *random*, kemudian menukar gen-gen antar *parent* yang terpilih.

Parent 1

1	1	0	1	0

Parent 2

1	0	1	1	1

c) Membentuk kromosom *offspring* hasil pertukaran gen-gen antar *parent*.

Offspring 1

1	0	1	1	0

Offspring 2

1	1	0	1	1

7) Langkah selanjutnya adalah proses mutasi. Langkah-langkah proses mutasi adalah sebagai berikut.

a) Memilih kromosom yang akan dijadikan *parent* dengan memilih kromosom yang memiliki nilai *random* $(r_i) \leq Pm$.

b) Memilih gen pada kromosom *parent* dengan aturan *random* kemudian menukar gen hingga terbentuk *offspring*.

Parent

1	1	0	1	1

Offspring

1	1	1	1	0

Setelah melakukan proses seleksi, *crossover*, dan mutasi maka dihasilkan generasi baru seperti pada Tabel 4.8. Dari Tabel 4.8 dapat diketahui bahwa nilai *fitness* terbaik adalah pada kromosom 3 sebesar 910000.

Tabel 4.8 Generasi Baru

Kromosom (k)	Pengkodean					Nilai *Fitness*
K_1	1	0	1	1	0	670000
K_2	1	1	1	1	0	805000
K_3	0	1	1	1	1	910000
K_4	0	1	0	1	1	435000
K_5	1	0	1	0	1	620000

Setelah itu untuk iterasi selanjutnya adalah mengevaluasi nilai *fitness* dan membuat generasi baru lagi hingga kriteria pemberhentian terpenuhi yaitu jumlah iterasi maksimal sudah tercapai.

b. Penyelesaian Masalah *Knapsack* 0-1 dengan Algoritma *Harmony Search* (HS)

Langkah-langkah penyelesaian masalah *knapsack* 0-1 dengan Algoritma *Harmony Search* adalah sebagai berikut.

1) Inisialisasi masalah dan parameter algoritma. Dalam hal ini, fungsi objektifnya adalah maksimasi $f(x)$ yaitu untuk memilih barang dan mencari keuntungan maksimum.

   Pada langkah pertama ini, semua parameter harus ditentukan. Penentuan ini berdasarkan penjelasan pada bab 2, yaitu sebagai berikut.

   a) *Harmony memory Size* (HMS) bernilai 5;

b) *Harmony memory Consideration Rate* (HMCR) bernilai 0,8;

c) *Pitch Adjustment Rate* (PAR) bernilai 0,4;

d) *Bandwidth* (*bw*) bernilai 0,2;

e) Kriteria pemberhentian yaitu jumlah iterasi maksimum sebanyak 100.

2) Langkah kedua dari algoritma HS adalah inisialisasi *Harmony memory* (HM). Pada tahap ini, dibangkitkan matriks HM secara *random* yang berisi vektor-vektor solusi sebanyak HMS seperti ditunjukkan pada Gambar 4.1 berikut ini.

$$\begin{bmatrix} 0.21 & 0.34 & 0.95 & 0.02 & 0.31 \\ 0.72 & 0.93 & 0.04 & 0.32 & 0.44 \\ 0.81 & 0.11 & 0.06 & 0.93 & 0.12 \\ 0.64 & 0.33 & 0.65 & 0.52 & 0.31 \\ 0.72 & 0.51 & 0.16 & 0.31 & 0.47 \end{bmatrix}$$

Gambar 4.1 Matriks *Harmony Memory*

Kemudian melakukan proses pemilihan masing-masing vektor solusi dengan matriks [0,1] seperti ditunjukkan pada Gambar 4.2 berikut ini.

$$\begin{bmatrix} VS1 \\ VS2 \\ VS3 \\ VS4 \\ VS5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Gambar 4.2 Representasi Vektor Solusi

Setiap vektor solusi dihitung nilai fungsi objektifnya, agar dapat diketahui himpunan vektor solusi yang memiliki nilai terbaik dan terburuk. Nilai fungsi objektif setiap vektor solusi ditunjukkan pada Gambar 4.3 berikut ini.

$$\begin{bmatrix} VS1 \\ VS2 \\ VS3 \\ VS4 \\ VS5 \end{bmatrix} = \begin{bmatrix} 795000 \\ 795000 \\ 735000 \\ 910000 \\ 910000 \end{bmatrix}$$

Gambar 4.3 Nilai Fungsi Objektif

Dari Gambar 4.3 tersebut, diketahui bahwa vektor solusi 4 dan 5 merupakan vektor solusi terbaik, sedangkan vektor solusi 3 merupakan vektor solusi terburuk.

Karena vektor solusi 3 merupakan vektor solusi terburuk, maka vektor solusi tersebut yang akan diproses pada langkah ketiga.

3) Langkah ketiga dari algoritma HS adalah improvisasi harmoni baru.

Pada tahap ini, vektor solusi terburuk akan digantikan dengan membentuk vektor solusi baru. Prosedur pembangkitan vektor solusi yang baru merujuk pada langkah-langkah yang yang ditulis pada bab 2 bagian pembangkitan vektor solusi baru (VSB). Berikut adalah langkah-langkah dalam pembangkitan vektor solusi yang baru.

a) Pembangkitan variabel keputusan baru yang pertama $i = 1$

Bilangan *random* yang dibangkitkan bernilai 0.91 sedangkan HMCR bernilai 0.8, sehingga bilangan $rand > HMCR$, maka menggunakan persamaan (2.8) diperoleh $x_1^{'} = 0.82$.

b) Pembangkitan variabel keputusan baru yang kedua $i = 2$

Bilangan *random* yang dibangkitkan bernilai 0.21 sedangkan HMCR bernilai 0.8, sehingga bilangan $rand < HMCR$. Selanjutnya dibangkitkan bilangan *random* yang kedua bernilai 0.56 sedangkan PAR bernilai 0.4, sehingga $rand > PAR$, maka menggunakan persamaan (2.9), (2.10), dan (2.11) diperoleh

$$d_1 = int[1 + (5 - 1)0.45]$$
$$d_1 = int[2.8]$$
$$d_1 = 3$$
$$d_2 = HM(d_1, i)$$
$$d_2 = HM(3,2)$$
$$d_2 = 0.11$$
$$x_2^{'} = d_2$$
$$x_2^{'} = 0.11$$

c) Pembangkitan variabel keputusan baru yang ketiga $i = 3$

Bilangan *random* yang dibangkitkan bernilai 0.31 sedangkan HMCR bernilai 0.8, sehingga bilangan $rand < HMCR$. Selanjutnya dibangkitkan bilangan *random* yang kedua bernilai 0.25 sedangkan PAR bernilai 0.4, sehingga $rand < PAR$, maka menggunakan persamaan (2.9), (2.10), (2.11), (2.12), dan (2.13) diperoleh

$$d_1 = int[1 + (5 - 1)0.76]$$
$$d_1 = int[4.04]$$
$$d_1 = 4$$
$$d_2 = HM(4,3)$$
$$d_2 = 0.65$$
$$d_3 = d_2 + bw \times \varepsilon$$
$$d_3 = 0.65 + 0.2 \times 0.01$$
$$d_3 = 0.67$$
$$x_3^{'} = d_3$$
$$x_3^{'} = 0.67$$

d) Pembangkitan variabel keputusan baru yang keempat $i = 3$

Bilangan *random* yang dibangkitkan bernilai 0.85 sedangkan HMCR bernilai 0.8, sehingga bilangan $rand > HMCR$, maka menggunakan persamaan (2.8) diperoleh $x_4^{'} = 0.97$.

e) Pembangkitan variabel keputusan baru yang kelima $i = 3$

Bilangan *random* yang dibangkitkan bernilai 0.35 sedangkan HMCR bernilai 0.8, sehingga bilangan $rand < HMCR$. Selanjutnya dibangkitkan bilangan *random* yang kedua bernilai 0.64 sedangkan PAR bernilai 0.4, sehingga $rand > PAR$, maka menggunakan persamaan (2.9), (2.10), dan (2.11) diperoleh

$$d_1 = int[1 + (5 - 1)0.82]$$
$$d_1 = int[4.28]$$
$$d_1 = 4$$

$$d_2 = HM(4,5)$$
$$d_2 = 0.31$$
$$x_5^{'} = d_2$$
$$x_5^{'} = 0.31$$

4) Langkah keempat dari algoritma HS adalah meng-*update harmony memory*. Apabila vektor solusi yang baru lebih baik nilainya, maka vektor solusi yang lama akan digantikan. Jika sebaliknya, maka tidak terjadi pergantian. Perbandingan nilai fungsi objektif vektor solusi yang lama dan yang baru, ditunjukkan pada Gambar 4.4 berikut ini.

$$\begin{bmatrix} VS1 \\ VS2 \\ VS3 \\ VS4 \\ VS5 \end{bmatrix} = \begin{bmatrix} 795000 & 795000 \\ 795000 & 795000 \\ 735000 & 910000 \\ 910000 & 910000 \\ 910000 & 910000 \end{bmatrix}$$

Gambar 4.4 Perbandingan Nilai Fungsi Objektif

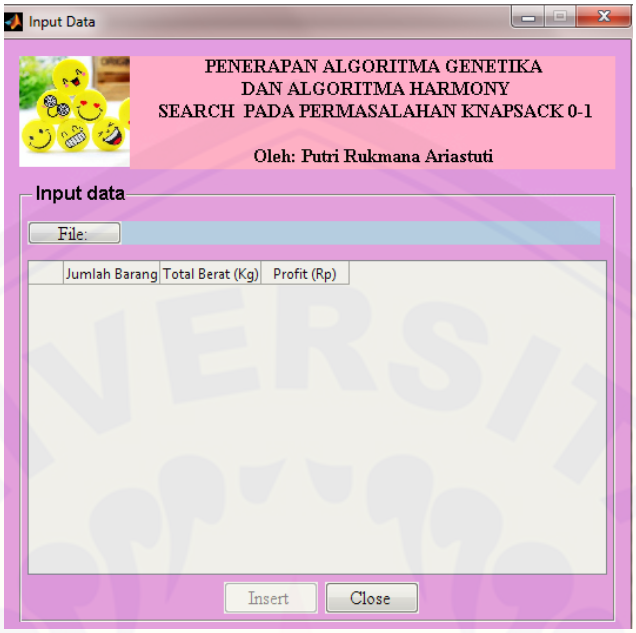
Berdasarkan Gambar 4.4 tersebut, vektor solusi 1 dan 2 merupakan vektor solusi terburuk, maka vektor solusi tersebut yang akan diproses pada iterasi selanjutnya.

5) Langkah selanjutnya adalah pengecekan kriteria pemberhentian. Apabila kriteria pemberhentian telah tercapai, maka iterasi dihentikan. Apabila belum tercapai, maka kembali ke langkah ketiga.

4.1.2 Program

Berdasarkan hasil program yang telah dibuat, akan dijelaskan langkah-langkah dalam pengoperasiannya.

Gambar 4.5 merupakan tampilan awal program. Pada jendela tersebut, terdapat menu *file* yang berfungsi untuk menemukan *file* yang berisi data barang dengan format .txt. Setelah itu pilih tombol *insert* untuk menyisipkan data pada program seperti ditunjukkan pada Gambar 4.6.

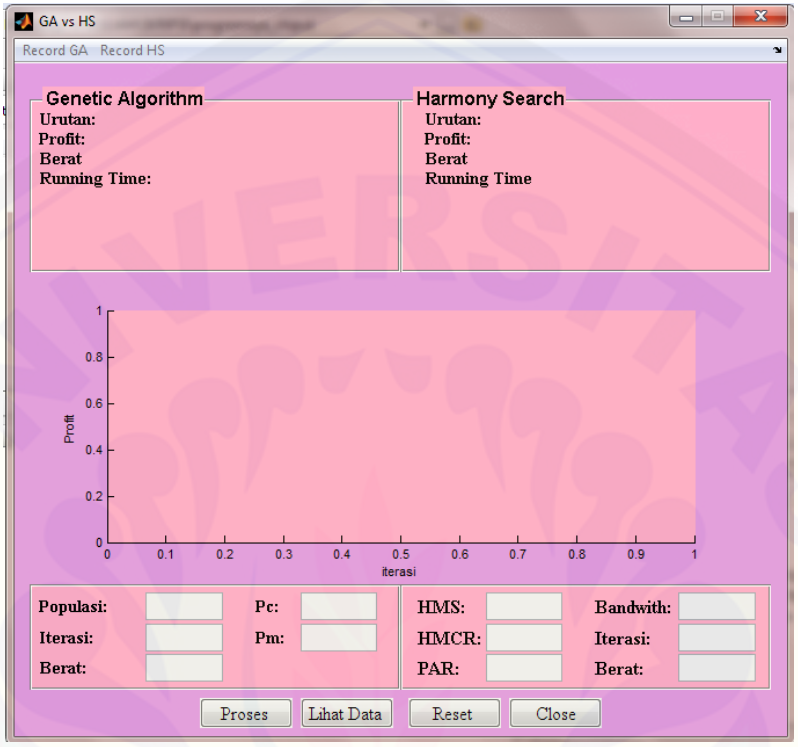


Gambar 4.5 Tampilan Awal Program

Gambar 4.6 Penyisipan Data Barang

Selanjutnya akan muncul jendela seperti pada Gambar 4.7 yang merupakan tampilan awal program dengan parameter masing-masing algoritma. Setelah meng-*input*kan parameter masing-masing algoritma, kemudian klik tombol Proses untuk

menjalankan program. Setelah proses selesai, maka akan ditampilkan barang yang terpilih, profit, berat, *running time*, dan konvergensi dari masing-masing algoritma.

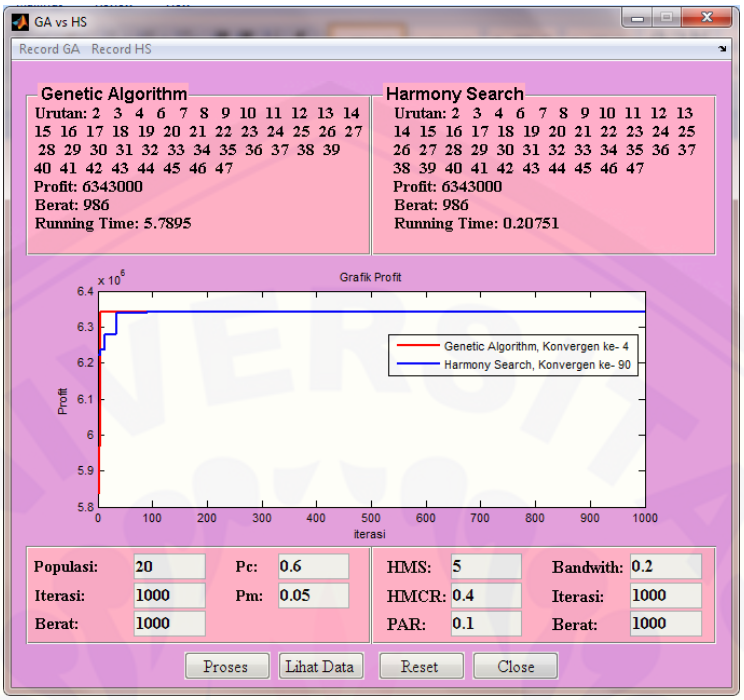


Gambar 4.7 Tampilan Awal Program dengan Parameter Algoritma

4.2 Pembahasan

Pada bagian ini, dilakukan percobaan sebanyak sepuluh kali dengan nilai parameter yang berbeda. Masing-masing percobaan dilakukan 10 kali *running*, dengan tujuan untuk mendapatkan nilai yang paling maksimal. Berikut ini adalah hasil percobaan.

a. Percobaan pertama menggunakan parameter algoritma Genetika yaitu *n* populasi 20, Pc 0,6,dan Pm 0,05, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,4, PAR 0,1, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.8 Hasil Percoban Pertama

Dari percobaan pertama, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5,7895 detik, dan konvergen pada iterasi ke-4. Sedangkan algoritma HS mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 0,20751 detik, dan konvergen pada iterasi ke-90.

Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.9 berikut.

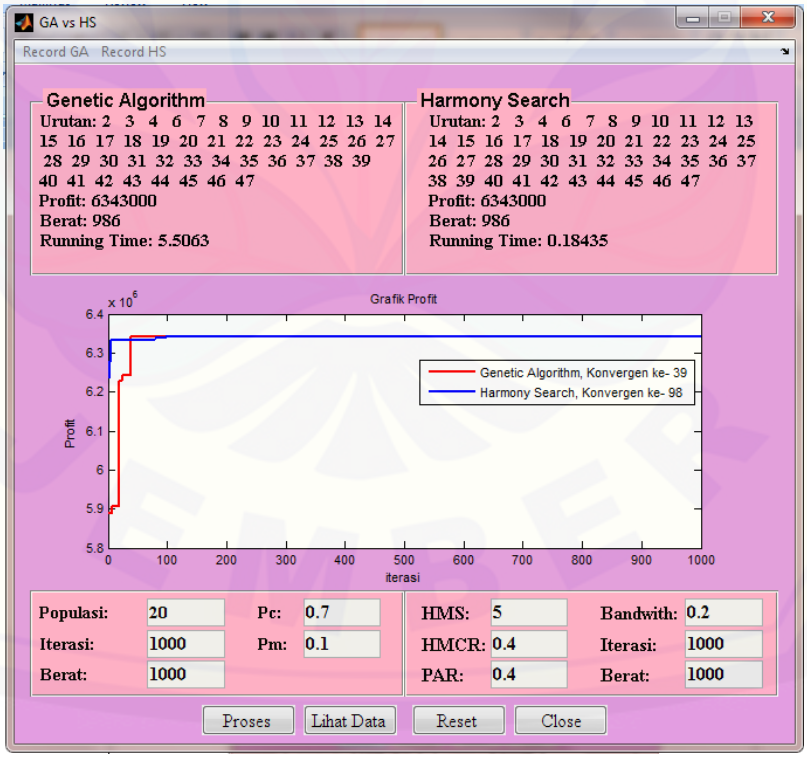
Tabel 4.9 Hasil *Running* Percobaan Pertama

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	4	5,7895	986	6.343.000	90	0,20751
2	986	6.343.000	135	5,6262	986	6.343.000	60	0,16378
3	986	6.343.000	13	5,3765	986	6.343.000	60	0,1604
4	986	6.343.000	3	5,323	986	6.343.000	134	0,15987
5	986	6.343.000	24	5,5334	986	6.343.000	23	0,16398
6	986	6.343.000	18	5,4661	986	6.343.000	112	0,16772

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)
7	986	6.343.000	12	5,6591	986	6.343.000	227	0,16172
8	986	6.343.000	27	5,6833	986	6.343.000	42	0,16325
9	986	6.343.000	63	5,6216	986	6.343.000	217	0,16325
10	986	6.343.000	10	5,5211	986	6.343.000	18	0,16201

Berdasarkan Tabel 4.9, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-3 selama 5,323 detik pada *running* ke-4. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-18 selama 0,16201 detik pada *running* ke-10.

b. Percobaan kedua menggunakan parameter algoritma Genetika yaitu *n* populasi 20, Pc 0,7, dan Pm 0,1, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,4, PAR 0,4, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.9 Hasil Percobaan Kedua

Dari percobaan kedua, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5,5063 detik, dan konvergen pada iterasi ke-39. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000 dengan berat 986 kg, *running time* 0,18435 detik, dan konvergen pada iterasi ke-98.

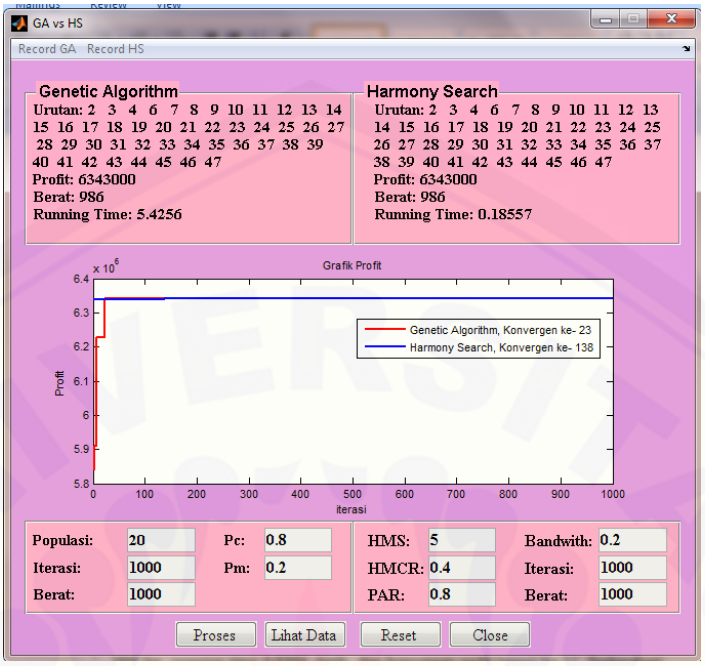
Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.10 berikut.

Tabel 4.10 Hasil *Running* Percobaan Kedua

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	39	5,5063	986	6.343.000	98	0,18435
2	986	6.343.000	19	5,5593	986	6.343.000	70	0,16414
3	986	6.343.000	48	5,5939	986	6.343.000	42	0,166
4	986	6.343.000	23	5,341	986	6.343.000	112	0,16567
5	986	6.343.000	20	5,5805	986	6.343.000	1	0,16949
6	986	6.343.000	18	5,6234	986	6.343.000	57	0,16472
7	986	6.343.000	18	5,6317	986	6.343.000	174	0,1643
8	986	6.343.000	69	5,4978	986	6.343.000	8	0,17695
9	986	6.343.000	5	5,3793	986	6.343.000	60	0,16523
10	986	6.343.000	12	5,6034	986	6.343.000	211	0,16544

Berdasarkan Tabel 4.10, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-5 selama 5,3793 detik pada *running* ke-9. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-1 selama 0,16949 detik pada *running* ke-5.

c. Percobaan ketiga menggunakan parameter algoritma Genetika yaitu n populasi 20, Pc 0,8, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,4, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.10 Hasil Percobaan Ketiga

Dari percobaan ketiga, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5,4256 detik, dan konvergen pada iterasi ke-23. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000 dengan berat 986 kg, *running time* 0,18557 detik, dan konvergen pada iterasi ke-138.

Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.11 berikut.

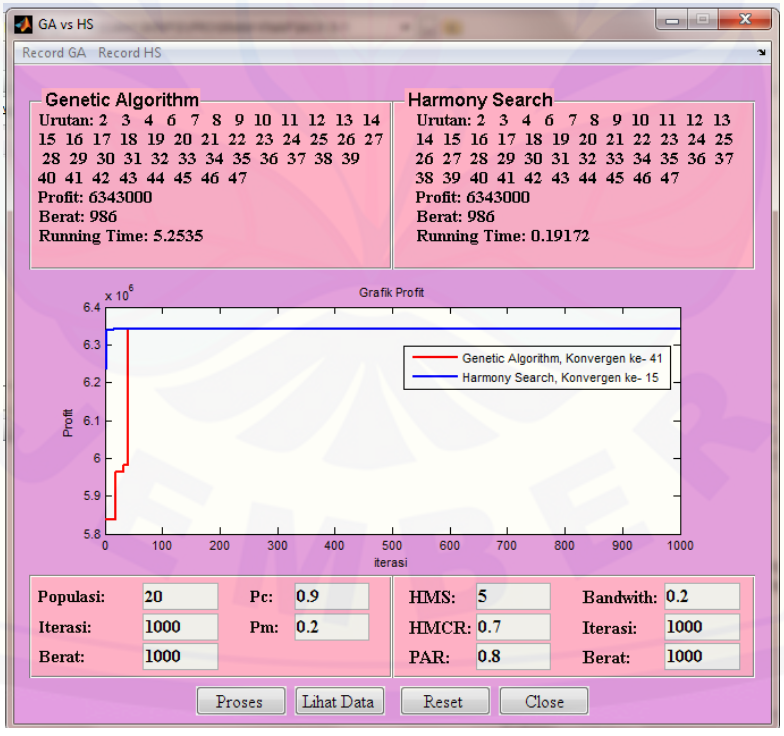
Tabel 4.11 Hasil *Running* Percobaan Ketiga

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	23	5,4256	986	6.343.000	138	0,18557
2	986	6.343.000	25	5,5296	986	6.343.000	82	0,17652
3	986	6.343.000	30	5,5079	986	6.338.000	22	0,17605
4	986	6.343.000	2	6,4023	986	6.338.000	59	0,17286
5	986	6.343.000	3	5,4139	986	6.343.000	48	0,17194
6	986	6.343.000	48	5,3617	986	6.343.000	60	0,17118
7	986	6.343.000	6	5,4826	986	6.343.000	64	0,1635

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)
8	986	6.343.000	14	5,1593	986	6.343.000	44	0,16436
9	986	6.343.000	25	5,2923	986	6.343.000	136	0,16318
10	986	6.343.000	12	5,4038	986	6.343.000	31	0,17055

Berdasarkan Tabel 4.11, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-2 selama 6,4023 detik pada *running* ke-4. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-22 selama 0,17605 detik pada *running* ke-3.

d. Percobaan keempat menggunakan parameter algoritma Genetika yaitu *n* populasi 20, Pc 0,9, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,7, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.11 Hasil Percobaan Keempat

Dari percobaan keempat, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5,2535 detik, dan konvergen pada iterasi ke-41. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000 dengan berat 986 kg, *running time* 0,19172 detik, dan konvergen pada iterasi ke-15.

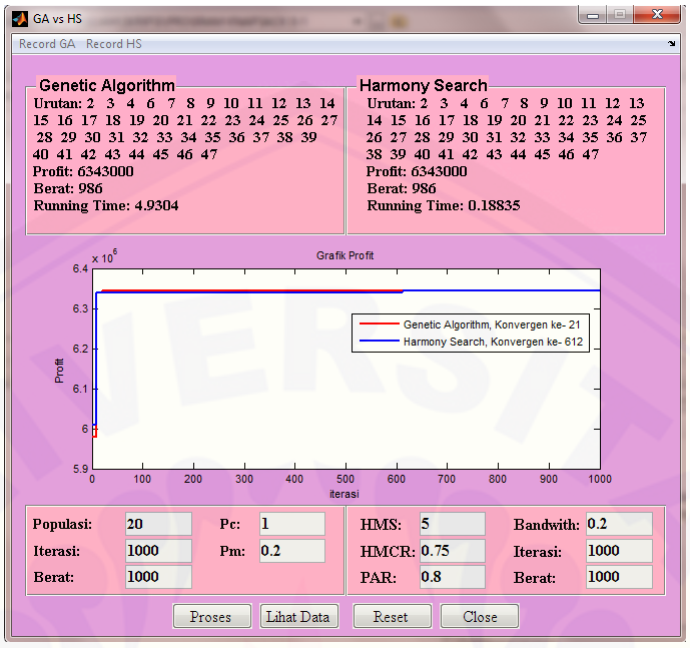
Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.12 berikut.

Tabel 4.12 Hasil *Running* Percobaan Keempat

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	41	5,2535	986	6.343.000	15	0,19172
2	986	6.338.000	40	5,0064	986	6.343.000	222	0,16939
3	986	6.343.000	68	5,3509	986	6.343.000	380	0,23188
4	986	6.343.000	32	4,9741	986	6.343.000	640	0,16993
5	986	6.343.000	11	5,0473	986	6.338.000	7	0,16839
6	986	6.343.000	31	4,9982	986	6.343.000	442	0,16621
7	986	6.343.000	10	5,0706	986	6.343.000	68	0,17093
8	986	6.343.000	42	5,1619	986	6.343.000	59	0,18213
9	986	6.343.000	8	5,093	986	6.343.000	15	0,16709
10	986	6.343.000	27	4,9811	986	6.343.000	16	0,17219

Berdasarkan Tabel 4.12, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-8 selama 5,093 detik pada *running* ke-9. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-15 selama 0,19172 detik pada *running* pertama.

e. Percobaan kelima menggunakan parameter algoritma Genetika yaitu *n* populasi 20, Pc 1, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,75, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.12 Hasil Percobaan Kelima

Dari percobaan kelima, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 4,9304 detik, dan konvergen pada iterasi ke-21. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000 dengan berat 986 kg, *running time* 0,18835 detik, dan konvergen pada iterasi ke-612.

Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.13 berikut.

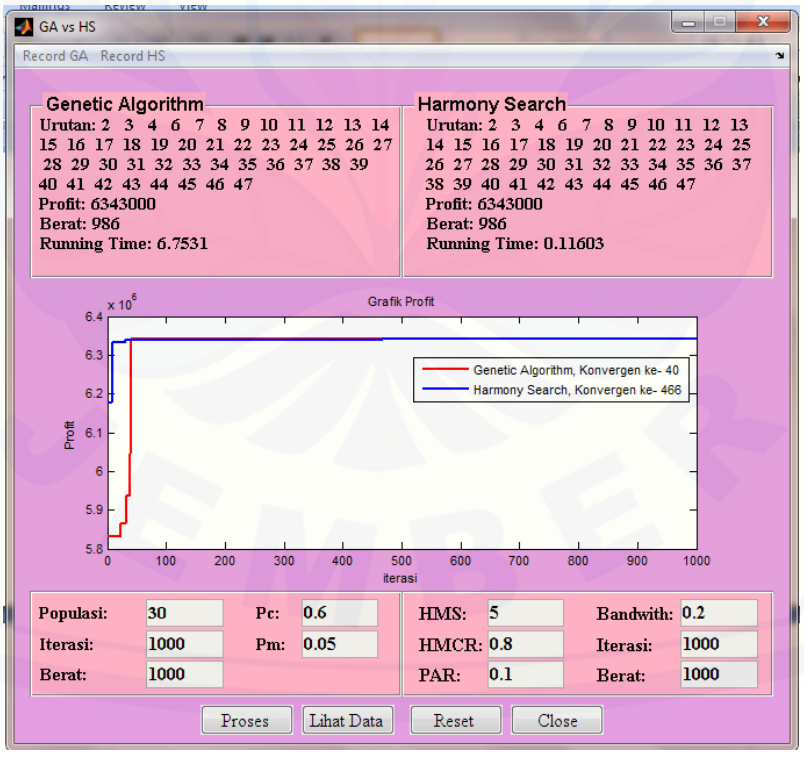
Tabel 4.13 Hasil *Running* Percobaan Kelima

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	21	4,9304	986	6.343.000	612	0,18835
2	986	6.343.000	26	4,9348	986	6.343.000	82	0,16895
3	986	6.343.000	29	4,8268	986	6.333.000	62	0,17305
4	986	6.343.000	21	4,8024	986	6.343.000	34	0,16822
5	986	6.343.000	25	4,9223	986	6.338.000	83	0,16806
6	986	6.343.000	28	4,9506	986	6.343.000	503	0,16843
7	986	6.343.000	34	4,9517	986	6.343.000	19	0,16764

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
8	986	6.343.000	22	4,7515	986	6.343.000	28	0,16732
9	986	6.343.000	7	4,9854	986	6.343.000	37	0,17126
10	986	6.343.000	7	4,8726	986	6.343.000	34	0,16834

Berdasarkan Tabel 4.13, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-7 selama 4,9854 detik pada *running* ke-9. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-28 selama 0,16732 detik pada *running* ke-8.

f. Percobaan keenam menggunakan parameter algoritma Genetika yaitu *n* populasi 30, Pc 0,6, dan Pm 0,05, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,8, PAR 0,1, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.13 Hasil Percobaan Keenam

Dari percobaan keenam, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 6,7531 detik, dan konvergen pada iterasi ke-40. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 0,11603 detik, dan konvergen pada iterasi ke-466.

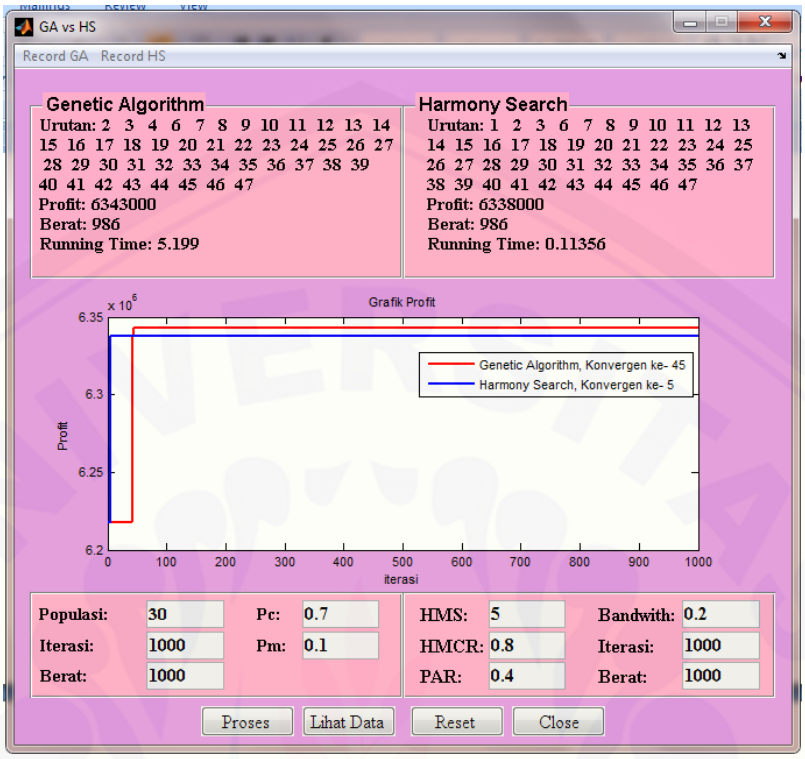
Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.14 berikut.

Tabel 4.14 Hasil *Running* Percobaan Keenam

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	40	6,7531	986	6.343.000	466	0,11603
2	986	6.343.000	22	7,49	986	6.343.000	37	0,1337
3	986	6.343.000	4	8,3209	986	6.343.000	57	0,19066
4	986	6.343.000	3	8,4837	986	6.343.000	55	0,16823
5	986	6.343.000	33	5,3451	986	6.343.000	88	0,099758
6	986	6.343.000	14	5,3932	986	6.343.000	248	0,10562
7	986	6.343.000	52	5,3019	986	6.343.000	1	0,099797
8	986	6.343.000	29	5,3692	986	6.343.000	67	0,099554
9	986	6.343.000	85	5,4455	986	6.343.000	162	0,099777
10	986	6.343.000	38	7,4923	986	6.343.000	74	0,1414

Berdasarkan Tabel 4.14, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-3 selama 8,4837 detik pada *running* ke-4. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-1 selama 0,099797 detik pada *running* ke-7.

g. Percobaan ketujuh menggunakan parameter algoritma Genetika yaitu *n* populasi 30, Pc 0,7, dan Pm 0,1, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,8, PAR 0,4, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.14 Hasil Percobaan Ketujuh

Dari percobaan ketujuh, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5,199 detik, dan konvergen pada iterasi ke-45. Sedangkan pada algoritma HS mencapai hasil 6338000 dengan berat 986 kg, *running time* 0,11356 detik, dan konvergen pada iterasi ke-5.

Dengan parameter tersebut, dilakukan 10 kali *running*. Hasil dari *running* tersebut dirangkum pada Tabel 4.13 berikut.

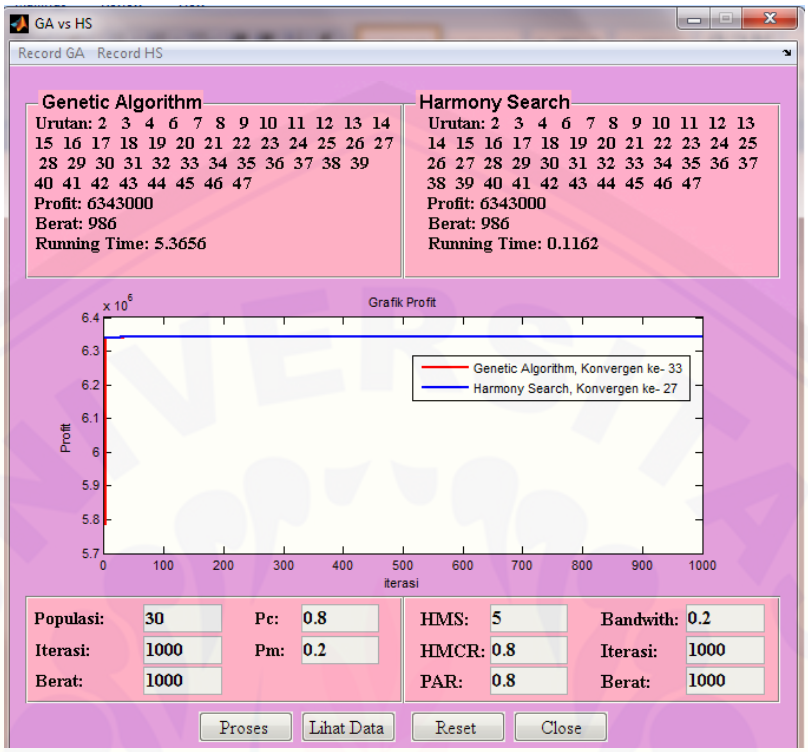
Tabel 4.15 Hasil *Running* Percobaan Ketujuh

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	45	5,199	986	6.338.000	5	0,11356
2	986	6.343.000	12	6,2678	986	6.343.000	35	0,12586
3	986	6.343.000	31	6,9139	986	6.343.000	56	0,12685
4	986	6.343.000	22	6,3335	986	6.343.000	94	0,12822

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)
5	986	6.343.000	3	6,6717	986	6.343.000	38	0,13678
6	986	6.343.000	17	5,9187	986	6.343.000	280	0,1036
7	986	6.343.000	45	6,4603	986	6.343.000	409	0,12553
8	986	6.343.000	8	6,9008	986	6.338.000	65	0,11439
9	986	6.343.000	45	6,2062	986	6.343.000	389	0,11466
10	986	6.343.000	33	6,6925	986	6.343.000	32	0,10287

Berdasarkan Tabel 4.15, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-3 selama 6,6717 detik pada *running* ke-5. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-32 selama 0,10287 detik pada *running* ke-10.

h. Percobaan kedelapan menggunakan parameter algoritma Genetika yaitu *n* populasi 30, Pc 0,8, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,8, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.15 Hasil Percobaan Kedelapan

Dari percobaan kedelapan, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 5.3656 detik, dan konvergen pada iterasi ke-33. Sedangkan pada algoritma HS mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 0.1162 detik, dan konvergen pada iterasi ke-27.

Dengan parameter tersebut, dilakukan 10 kali percobaan. Hasil dari percobaan tersebut dirangkum pada Tabel 4.16 berikut.

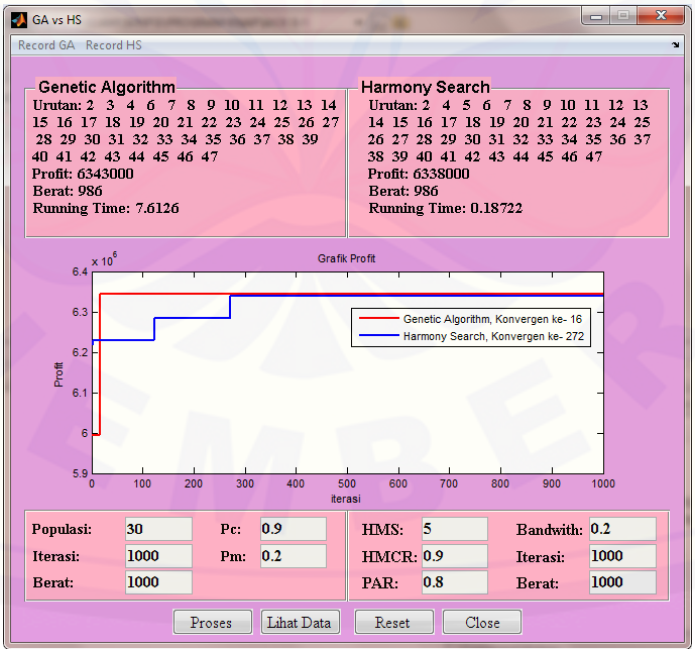
Tabel 4.16 Hasil *Running* Percobaan Kedelapan

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	33	5,3656	986	6.343.000	27	0,1162
2	986	6.343.000	21	4,9759	986	6.343.000	236	0,10327
3	986	6.343.000	67	5,0553	986	6.343.000	661	0,10435
4	986	6.343.000	18	5,1385	986	6.343.000	5	0,1039
5	986	6.343.000	19	5,0809	986	6.343.000	151	0,1038

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running Time* (s)
6	986	6.343.000	26	5,2632	986	6.338.000	7	0,10831
7	986	6.343.000	55	5,1975	986	6.338.000	19	0,10609
8	986	6.343.000	23	5,5173	986	6.338.000	27	0,10426
9	986	6.343.000	15	5,2305	986	6.338.000	6	0,10356
10	986	6.343.000	29	5,2653	986	6.338.000	1	0,10351

Berdasarkan Tabel 4.16, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-15 selama 5,2305 detik pada *running* ke-9. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-5 selama 0,1039 detik pada *running* ke-4.

i. Percobaan kesembilan menggunakan parameter algoritma Genetika yaitu *n* populasi 30, Pc 0,9, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,9, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.16 Hasil Percobaan Kesembilan

Dari percobaan kesembilan, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 7,6126 detik, dan konvergen pada iterasi ke-16. Sedangkan pada algoritma HS mencapai profit Rp 6.338.000,- dengan berat 986 kg, *running time* 0.18722 detik, dan konvergen pada iterasi ke-272.

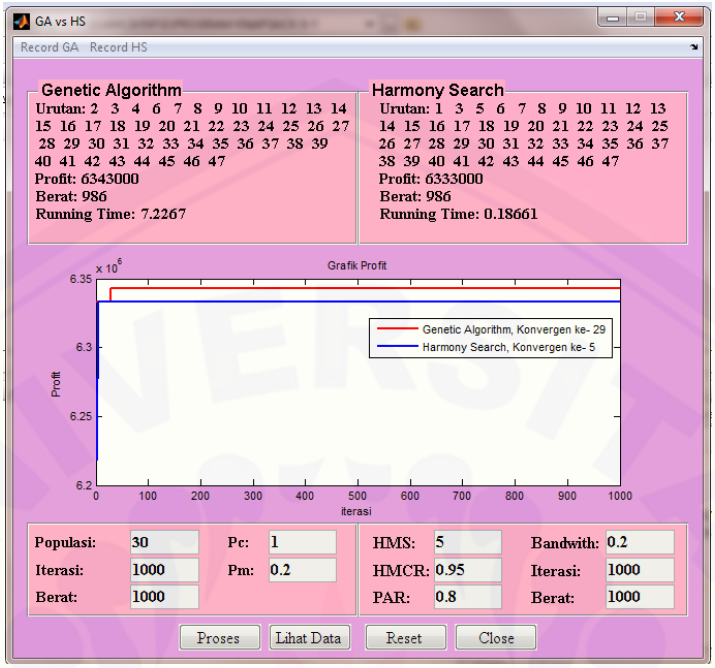
Dengan parameter tersebut, dilakukan 10 kali percobaan. Hasil dari percobaan tersebut dirangkum pada Tabel 4.17 berikut.

Tabel 4.17 Hasil *Running* Percobaan Kesembilan

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	16	7,6126	986	6.338.000	272	0,18722
2	986	6.343.000	30	7,6365	986	6.343.000	64	0,17296
3	986	6.343.000	44	7,6448	966	6.218.000	1	0,18071
4	986	6.343.000	23	7,6072	986	6.343.000	5	0,17698
5	986	6.343.000	16	7,5177	986	6.338.000	1	0,17678
6	986	6.343.000	44	7,5536	986	6.338.000	25	0,17495
7	986	6.343.000	36	7,6198	986	6.343.000	99	0,17572
8	986	6.343.000	22	7,5783	986	6.343.000	11	0,17958
9	986	6.343.000	18	7,6425	986	6.338.000	46	0,17203
10	986	6.343.000	15	7,4667	986	6.338.000	14	0,17183

Berdasarkan Tabel 4.17, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-15 selama 7,4667 detik pada *running* ke-10. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-5 selama 0.17698 detik pada *running* ke-4.

j. Percobaan kesepuluh menggunakan parameter algoritma Genetika yaitu *n* populasi 30, Pc 1, dan Pm 0,2, sedangkan parameter algoritma HS yaitu HMS 5, HMCR 0,95, PAR 0,8, *bw* 0,2 dengan berat maksimum 1000 kg, dan jumlah iterasinya 1000.



Gambar 4.17 Hasil Percobaan Kesepuluh

Dari percobaan kesepuluh, algoritma Genetika mencapai profit Rp 6.343.000,- dengan berat 986 kg, *running time* 7,2267 detik, dan konvergen pada iterasi ke-29. Sedangkan pada algoritma HS mencapai profit Rp 6.333.000,- dengan berat 986 kg, *running time* 0.18661 detik, dan konvergen pada iterasi ke-5.

Dengan parameter tersebut, dilakukan 10 kali percobaan. Hasil dari percobaan tersebut dirangkum pada Tabel 4.18 berikut.

Tabel 4.18 Hasil *Running* Percobaan Kesepuluh

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	986	6.343.000	29	7,2267	986	6.333.000	5	0,18661
2	986	6.343.000	27	7,4683	986	6.343.000	15	0,17048
3	986	6.343.000	28	7,0621	986	6.343.000	626	0,17922
4	986	6.343.000	2	7,2342	986	6.333.000	3	0,17447
5	986	6.343.000	22	7,1074	986	6.333.000	65	0,17445
6	986	6.343.000	8	7,1409	986	6.343.000	119	0,1724

No.	Genetika				*Harmony Search*			
	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
7	986	6.343.000	12	7,283	986	6.338.000	1	0,17011
8	986	6.343.000	45	7,2097	986	6.343.000	21	0,17471
9	986	6.343.000	19	7,1122	986	6.338.000	19	0,17357
10	986	6.343.000	37	7,1633	986	6.343.000	6	0,17119

Berdasarkan Tabel 4.18, dapat dilihat bahwa pada algoritma Genetika mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-2 selama 7,2342 detik pada *running* ke-4. Sedangkan pada algoritma *Harmony Search* mencapai hasil optimal dengan konvergensi terbaik yaitu pada iterasi ke-6 selama 0,17119 detik pada *running* ke-10.

Rangkuman dari sepuluh kali percobaan dengan algoritma Genetika dan algoritma *Harmony Search* dapat dilihat pada Tabel 4.19 dan Tabel 4.20. Kedua tabel tersebut diperoleh dari masing-masing percobaan mencapai hasil optimal dengan konvergensi terbaik.

Tabel 4.19 Rangkuman Sepuluh Percobaan dengan Algoritma Genetika

Percobaan	Parameter			Hasil			
	n populasi	*Pc*	*Pm*	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	20	0,6	0,05	986	6.343.000	3	5,3230
2	20	0,7	0,1	986	6.343.000	5	5,3793
3	20	0,8	0,2	986	6.343.000	2	6,4023
4	20	0,9	0,2	986	6.343.000	8	5,0930
5	20	1	0,2	986	6.343.000	7	4,9854
6	30	0,6	0,05	986	6.343.000	3	8,4837
7	30	0,7	0,1	986	6.343.000	3	6,6717
8	30	0,8	0,2	986	6.343.000	15	5,2305
9	30	0,9	0,2	986	6.343.000	15	7,4667
10	30	1	0,2	986	6.343.000	2	7,2342

Tabel 4.20 Rangkuman Sepuluh Percobaan dengan Algoritma *Harmony Search*

Percobaan	Parameter				Hasil			
	HMS	HMCR	PAR	*bw*	Berat (kg)	Profit (Rp)	Konvergen	*Running time* (s)
1	5	0,4	0,1	0,2	986	6.343.000	18	0,16201
2	5	0,4	0,4	0,2	986	6.343.000	1	0,16949
3	5	0,4	0,8	0,2	986	6.343.000	22	0,17605
4	5	0,7	0,8	0,2	986	6.343.000	15	0,19172
5	5	0,75	0,8	0,2	986	6.343.000	28	0,16732
6	5	0,8	0,1	0,2	986	6.343.000	1	0,09979
7	5	0,8	0,4	0,2	986	6.343.000	32	0,10287
8	5	0,8	0,8	0,2	986	6.343.000	5	0,10390
9	5	0,9	0,8	0,2	986	6.343.000	5	0,17698
10	5	0,95	0,8	0,2	986	6.343.000	6	0,17119

Berdasarkan percobaan yang telah dilakukan pada algoritma Genetika dan algoritma *Harmony Search*, terlihat bahwa secara umum berat total dan keuntungan total yang dihasilkan dari setiap *running* tidak sama. Hal ini karena bilangan *random* yang dibangkitkan, jika nilai bilangan bilangan *random* yang dibangkitkan tidak mendukung untuk mencapai nilai keuntungan maksimal, maka akan sulit dihasilkan nilai keuntungan yang maksimal. Parameter-parameter algoritma Genetika dan algoritma *Harmony Search* yang digunakan pada penelitian sudah dapat mewakili selang yang telah ditentukan. Banyaknya *running* pada masing-masing percobaan ditentukan 10 kali, karena nilai yang dihasilkan fluktuatif sehingga dicukupkan sebanyak 10 kali *running* saja seperti yang dilakukan pada penelitian Zou *et al* (2011).

Dari sepuluh kali percobaan terlihat bahwa hasil *running* dari algoritma *Harmony Search* mengalami fluktuatif pada profit yang dihasilkan namun berat totalnya sudah optimum, sedangkan hasil *running* algoritma Genetika lebih stabil sejak percobaan pertama hingga percobaan kesepuluh yaitu sebesar Rp 6.343.000,- dengan berat total 986 kg. Hal ini menunjukkan bahwa algoritma Genetika lebih baik

daripada algoritma *Harmony Search* bila diterapkan pada permasalahan *knapsack* 0-1 karena menunjukkan hasil yang lebih optimal.

Algoritma genetika memiliki *running time* yang lebih lama daripada algoritma *Harmony Search*, namun algoritma Genetika lebih cepat konvergen dibandingkan algoritma *Harmony Search*. *Running time* yang lama pada algoritma Genetika disebabkan oleh proses mutasinya. Pada proses mutasi, selain menggunakan bilangan *random* yang nilainya kurang dari sama dengan *Pm*, setiap kromosom dievaluasi, apakah melanggar fungsi kendala yaitu berupa kapasitas maksimum *knapsack*. Apabila ada kromosom yang berat dan keuntungannya belum maksimal, maka akan terjadi proses mutasi dengan memprioritaskan barang yang memiliki keuntungan yang besar tiap kilogramnya. Setelah itu dilakukan evaluasi kembali, apakah beratnya melebihi kapasitas *knapsack*, jika ya, maka akan terjadi mutasi kembali dengan mengurangkan barang yang menyebabkan kelebihan muatan. Hal itu dilakukan terus menerus setiap kromosom dan setiap iterasi, maka hal inilah yang menyebabkan *running time* algoritma Genetika lebih lama dibandingkan algoritma *Harmony Search*.

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada tugas akhir ini, maka dapat diambil kesimpulan sebagai berikut.

a. Berdasarkan hasil simulasi pada studi kasus di UD. Permata Indah Situbondo, hasil profit yang ditunjukkan oleh algoritma Genetika lebih stabil daripada algoritma *Harmony Search* yaitu sebesar Rp 6.343.000,- dengan berat total 986 kg. Hal ini menunjukkan bahwa algoritma Genetika lebih baik daripada algoritma *Harmony Search* dalam menyelesaikan permasalahan *knapsack* 0-1.
b. Dilihat dari segi *running time*, algoritma *Harmony Search* memiliki *running time* yang lebih cepat daripada algoritma Genetika sehingga algoritma *Harmony Search* lebih baik daripada algoritma Genetika.
c. Dilihat dari segi konvergensinya, algoritma Genetika lebih cepat konvergen dibandingkan dengan algoritma *Harmony Search* sehingga algoritma Genetika lebih baik daripada algoritma *Harmony Search*.

5.2 Saran

Masalah *knapsack* 0-1 merupakan permasalahan optimasi yang dapat dikerjakan dengan berbagai macam metode dan algoritma. Masih terbuka bagi peneliti yang lain untuk menerapkan algoritma metaheuristik yang lain untuk menyelesaikan permasalahan *knapsack* 0-1. Selain menggunakan pemrograman Matlab, peneliti yang lain juga dapat menggunakan bahasa pemrograman yang lain seperti *Java*, *Visual Basic* dan sebagainya.

DAFTAR PUSTAKA

Chen, K.C., Ian H, dan Cao A.W. 2003. *A Genetic Algorithm for Minimum Tetrahedralization of a Convex Polyhedron*. Canadian Conference on Computational Geometry (CCCG): pp. 115-119. http://www.cccg.ca/proceedings/2003/29.pdf [20 Januari 2015]

Diah, K., Fadhli, M., dan Sutanto, C. 2010. *Penyelesaian Knapsack Problem Menggunakan Algoritma Genetika*. Riau: Politeknik Caltex Riau Pekanbaru.

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

Geem, Z. W. & Lee, K. S. 2005. A New Meta-heuristic Algorithm for Continous Engineering Optimizations: Harmony Search Theory and Practice. *Comput Methods Application Mechanical Engineering*. **194**: 3902-3933.

Geem, Z. W. & Williams, J. C. 2007. Harmony Search and Ecological Optimization. *International Journal of Energy and Environment*, Issue 2, **1**: 150-154. http://www.naun.org/multimedia/NAUN/energyenvironment/ee-26.pdf [7 Januari 2015]

Gen, M. & R. Cheng. 1997. *Genetic Algorithm and Engineering Design*. USA: John Wile &Sons, Inc.

Goldberg, D. E. 1989. *Genetic Algorithm in Search Optimization, and Machine Learning*. USA: Addison-Weasley Publishing Company, Inc.

Martello, S. 2006. *New Trends in Exact Algorithms for the 0-1 Knapsack Problem*. http://www.cise.ufl.edu/~sahni/papers/knap.pdf. [7 Januari 2015]

Paryati. 2009. *Optimasi Strategi Algoritma Greedy untuk Menyelesaikan Permasalahan Knapsack 0-1*. Yogyakarta: UPN "Veteran" Yogyakarta.

Ridho, M. 2014. "Penyelesaian Masalah Knapsack 0-1 dengan Algoritma Harmony Search dan Dynamic Programming." Tidak Diterbitkan. Skripsi. Jember: Jurusan Matematika FMIPA Universitas Jember.

Santosa, B. & Willy, P. 2011. *Metoda Metaheuristik: Konsep Implementasi*. Surabaya: Guna Widya.

Setiawan, A. 2010. *Penerapan Algoritma Harmony Search dalam Penyelesaian Resource-Constrained Project Scheduling Problem*. http://digilib.its.ac.id/public/ITS-Undergraduate-12614-Paper.pdf [20 Januari 2015]

Setiowati, Y. 2003. *Penyelesaian Persoalan Multidimensional Knapsack (PMK) dengan Algoritma Genetika*. Surabaya: Politeknik Elektronika Negeri Surabaya-ITS.

Sivanandam, S.N. & Deepa, S.N. 2008. *Introduction to Genetic Algorithms*. India: PSG College of Technology Coimbatore.

Zou, D., Gao, L., Li, S. & Wu, J. 2011. Solving 0-1 Knapsack Problem By a Novel Global Harmony Search Algorithm. *Applied Soft Computing*, **11**(2), 1556-1564.

LAMPIRAN

Daftar Barang

No.	Nama Barang (*i*)	Jumlah Barang	Harga beli (Rp)	Harga Jual (Rp)
1.	CIHERANG 10kg	5	76000	80000
2.	PADI 64 10kg	5	75000	80000
3.	WAY APU 10kg	5	75500	80500
4.	TOWATI 10kg	5	76000	81000
5.	SITUBAGEDID 10kg	5	76000	80000
6.	BISI 2 20kg	3	880000	925000
7.	P21 20kg	3	1200000	1260000
8.	P27 20kg	3	1220000	1260000
9.	PAC 224 20kg	3	1190000	1230000
10.	TM BA 1kg	2	105000	130000
11.	TM JAWARA 1kg	2	106000	130000
12.	TM PANAMAS 1kg	2	160000	180000
13.	PURADON 40kg	3	400000	455000
14.	REAGENT 10kg	4	160000	180000
15.	REAGENT CAIR 1kg	5	210000	245000
16.	SEDRIK 5kg	5	400000	495000
17.	SEDRIK 8kg	5	600000	680000
18.	DURSBAN 1kg	5	95000	120000
19.	DURSBAN 2.5kg	5	210000	240000
20.	DURSBAN 5kg	5	310000	345000
21.	DURSBAN 10kg	1	500000	550000
22.	URACRON 1kg	5	210000	245000
23.	URACRON 2.5kg	5	500000	540000
24.	FASTACT 1kg	5	75000	90000
25.	FASTACT 2.5kg	5	173000	195000
26.	FASTACT 4kg	5	280000	310000
27.	POLIDOR 1kg	5	160000	180000
28.	POLIDOR 2kg	5	300000	325000
29.	POLIDOR 4kg	5	520000	555000
30.	GAMATOC 1kg	5	120000	150000
31.	GAMATOC 2.5kg	5	280000	320000
32.	AKODAN 1kg	5	170000	200000
33.	DARMAIBAS 1kg	5	100000	115000
34.	ASUDRIN 1kg	5	190000	215000
35.	ASUDRIN 5kg	5	510000	550000
36.	DIPUSTAR 2kg	5	160000	180000
37.	DIPUSTAR 4kg	5	310000	350000
38.	SPONTAN 2kg	5	150000	180000
39.	SPONSOR 2kg	5	145000	180000
40.	SEPIN 2kg	5	160000	185000
41.	CONDIFOR 2kg	5	200000	250000
42.	GRANTONIC 10kg	1	240000	320000
43.	GRANTONIC 20kg	1	320000	500000
44.	SEPLASH 10kg	1	520000	600000
45.	SEPLASH 20kg	1	720000	900000
46.	PEREKAT 10kg	1	280000	400000
47.	PEREKAT 20kg	1	590000	720000