PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP) DENGAN ALGORITMA *HYBRID CAT SWARM OPTIMIZATION* (hCSO)

SKRIPSI

Oleh
Tri Puji Lestari
NIM 121810101040

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP) DENGAN ALGORITMA *HYBRID CAT SWARM OPTIMIZATION* (hCSO)

SKRIPSI

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

Oleh
TRI PUJI LESTARI
NIM 121810101040

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ibunda Sulastri dan Ayahanda Samiran yang tercinta, terimakasih atas kesabarannya dalam mendidik, mendoakan dan memberikan kasih sayang serta pengorbanan selama ini;
2. Kakakku Samihadi, Kakakku Siswowiyono dan kakakku Risma Ike Yuliana yang tersayang;
3. Bapak dan ibu guru dari Taman Kanak-kanak sampai Perguruan Tinggi yang telah memberikan bekal ilmu, mendidik dengan tulus ikhlas agar menjadi pribadi yang berkarakter, berakhlak serta membimbing dengan sepenuh hati;
4. Teman-teman seperjuanganku di Jurusan Matematika FMIPA Universitas Jember angkatan 2012;
5. Almamater tercinta jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

MOTO

Dan bahwasanya seorang manusia tidak akan memperoleh selain apa yang telah diusahakannya.

(QS. An-Najm: 39)[1]

[1]Departement Agama Republik Indonesia.2006. Al Qur'an danTerjemahannya. Bandung: CV PenerbitDiponegoro

PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Tri Puji Lestari

NIM : 121810101040

menyatakan dengan sesungguhnya bahwa karya tulis ilmiah yang berjudul "Penyelesaian *Travelling Salesman Problem* (TSP) dengan Algoritma *Hybrid Cat Swarm Optimization* (HCSO)" adalah benar-benar hasil karya sendiri, kecuali jika disebutkan sumbernya dan belum pernah diajukan pada institusi manapun, serta bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenarann isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenar-benarnya, tanpa ada paksaan dan tekanan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata dikemudian hari pernyataan ini tidak benar.

Jember, Juni 2016
Yang menyatakan,

Tri Puji Lestari
NIM: 121810101040

SKRIPSI

PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP) DENGAN ALGORITMA *HYBRID CAT SWARM OPTIMIZATION* (hCSO)

Oleh

Tri Puji Lestari

NIM 121810101040

Pembimbing

Dosen Pembimbing I : M. Ziaul Arif, S.Si., M.Sc

Dosen Pembimbing II : Kusbudiono, S.Si., M.Si

PENGESAHAN

Skripsi berjudul "Penyelesaian *Travelling Salesman Problem* (TSP) Dengan Algoritma *Hybrid Cat Swarm Optimization* (HCSO) " telah diuji dan disahkan pada:

hari :

tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Sekertaris,
M. Ziaul Arif, S.Si., M.Sc	Kusbudiono, S.Si., M.Si
NIP. 198501112008121002	NIP. 19770432005011001
Anggota I	Anggota II
Ahmad Kamsyakawuni, S.Si., M.Kom	Dr. Mohamad Fatekurohman, S.Si., M.Si
NIP. 197211291998021001	NIP. 196906061998031001

Mengesahkan

Dekan Fakultas MIPA

Universitas Jember,

Drs. Sujito, Ph.D

NIP. 196102041987111001

RINGKASAN

PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP) DENGAN ALGORITMA *HYBRID CAT SWARM OPTIMIZATION* (hCSO); Tri Puji Lestari, 121810101040; 2016:54 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Kegiatan pengiriman barang merupakan salah satu rutinitas perusahaan guna untuk mendistribusikan barang kepada pelanggan. Dalam mengantarkan produk ke pelanggan terdapat banyak alternatif rute yang dapat ditempuh oleh *salesman* untuk sampai ke semua pelanggan tersebut. Hal ini menimbulkan permasalahan bagi seorang *salesman* karena harus menentukan rute terpendek dari banyaknya alternatif rute yang ada. Tentunya harus dipilih yang paling optimal sehingga semua pelanggan dapat dikunjungi dengan jarak tempuh yang minimum. Kegiatan pengiriman barang tersebut termasuk dalam masalah perjalanan *salesman* (*Travelling Salesman Problem*). Dalam penelitian ini *Travelling Salesman Problem* akan diselesaikan dengan algoritma gabungan dari algoritma *Cat Swarm Optimization* (CSO) dan algoritma *Harmony Search* (HS) yang kemudian disebut dengan algoritma *Hybrid Cat Swarm Optimization* (hCSO). Dengan tujuan meminimumkan jarak yang harus dilalui oleh *salesman*. Kemudian aplikasi algoritma hCSO pada program Matlab. Kemudian perbandingan hasil yang diperoleh dari algoritma CSO, HS dan hCSO.

Penelitian ini dilakukan beberapa langkah yaitu studi literatur yang berkaitan dengan referensi-referensi terkait algoritma CSO dan HS. Kemudian pengumpulan data yang diambil dari penelitian sebelumnya terkait kios pelanggan dari UD. Tani Lumintu Banyuwangi. Kemudian penyelesaian secara manual dengan data kecil yaitu 10 data. Kemudian pembuatan program dengan Matlab. Kemudian simulasi dan implementasi algoritma hCSO, dilakukan simulasi perubahan parameter *m*, *MR*, *c*, *SMP* dan *SRD* dengan masing-masing dicobakan sampai 3 buah nilai yang diulang

sebanyak 10 kali. Kemudian melakukan simulasi hasil dari CSO, HS dan HCSO. Kemudian hasilnya di analisis dan ditarik kesimpulan.

Dari penelitian yang telah dilakukan didapatkan bahwa Semakin besar parameter m maka jarak yang dihasilkan semakin minimum namun waktu *running* semakin lama. Semakin besar parameter *MR* yang diberikan maka waktu *running* yang dibutuhkan semakin cepat. Semakin besar parameter *SMP* yang diberikan maka jarak yang dihasilkan semakin minimum namun waktu semakin lama. Sedangkan parameter *c* dan *SRD* besarnya tidak mempengaruhi hasil. Selain itu dilakukan sebanyak 5 kali perulangan untuk perbandingan hasil dari CSO, HS dan HCSO. Hasil dari algoritma CSO diperoleh rata-rata jarak yang diperoleh yaitu 207.5, rata-rata waktu yang dibutuhkan untuk *running* yaitu 728.53424 dan rata-rata titik konvergen yaitu 1554.2. Hasil dar algoritma HS diperoleh rata-rata jarak yaitu 356.76, rata-rata waktu yang dibutuhkan untuk *running* yaitu 297045.6 dan rata-rata titik konvergen pada 1149. Hasil dari algoritma hCSO rata-rata jarak yaitu 193.52, rata-rata waktu yang dibutuhkan untuk *running* yaitu 5725.64704 dan rata-rata titik konvergen pada 1485.6.

Dari hasil tersebut dapat diperoleh bahwa algoritma yang lebih efektif untuk mencari jarak minimum adalah algoritma hCSO dibandingkan dengan algoritma CSO dan HS. Sedangkan algoritma yang lebih efektif dari segi waktu yaitu algoritma HS dibandingkan dengan algoritma CSO dan hCSO. Meskipun algoritma CSO, HS dan hCSO dapat dijamin konvergen, namun tingkat kekonvergenannya tidak dapat diprediksi setiap iterasinya. Hal ini disebabkan karena ketiga algoritma ini merupakan jenis algoritma metaheuristik yang setiap iterasinya selalu dibangkitkan oleh bilangan random. Jarak paling minimum yang diperoleh yaitu 151.2.

PRAKATA

Syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga karya tulis berupa skripsi yang berjudul "Penyelesaian *Travelling Salesman Problem* (TSP) Dengan Algoritma *Hybrid Cat Swarm Optimization* (HCSO)" dapat terselesaikan. Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata (S1) pada program studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, secara khusus disampaikan terimakasih kepada:

1. Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan alam Bapak Kusbudiono, S.Si., M.Si;
2. M. Ziaul Arif, S.Si., M.Sc selaku dosen pembimbing utama dan Kusbudiono, S.Si., M.Si selaku dosen pembimbing anggota, yang telah meluangkan waktu, pikiran, tenaga serta perhatian dalam penulisan skripsi ini;
3. Ahmad Kamsyakawuni, S.Si., M.Kom selaku dosen penguji I dan Dr. Mohamad Fatekurohman, S.Si., M.Si selaku dosen penguji II yang telah meluangkan waktu, pikiran dan tenaga untuk perbaikan tulisan ini;
4. Seluruh dosen jurusan matematika atas ilmu, bimbingan dan didikan selama menjadi mahasiswa Matematika;
5. Kedua orang tuaku, kakak, adik, dan keluarga besarku di Jember terimakasih atas kasih sayang, doa, harapan, kesempatan, dukungan dan motivasi yang diberikan kepadaku;
6. Sahabatku-sahabatku di UKKI IONS terimakasih atas bantuan, hiburan, doa yang memberikan kenangan motivasi dan inspirasi kepadaku.
7. Teman-teman seperjuangan Matematika angkatan 2012, semuanya yang memberikan bentuk persahabatan yang indah;

8. Semua pihak yang tidak dapat disebutkan satu persatu yang telah memberikan bantuan dan dukungan dalam penyelesaian skripsi ini.

Jember, Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Kegiatan pengiriman barang merupakan salah satu rutinitas perusahaan guna untuk mendistribusikan barang kepada pelanggan. Dalam mengantarkan produk ke pelanggan terdapat banyak alternatif rute yang dapat ditempuh oleh *salesman* untuk sampai ke semua pelanggan tersebut. Hal ini menimbulkan permasalahan bagi seorang *salesman* karena harus menentukan rute terpendek dari banyaknya alternatif rute yang ada. Tentunya harus dipilih yang paling optimal sehingga semua pelanggan dapat dikunjungi dengan jarak tempuh yang minimum. Kegiatan pengiriman barang tersebut termasuk dalam masalah perjalanan *salesman* (*Travelling Salesman Problem*). *Travelling Salesman Problem* (TSP) adalah sebuah persoalan optimasi untuk mendapatkan rute terpendek yang harus dilalui oleh seorang pedagang keliling (*salesman*). *Salesman* tersebut harus mengunjungi sejumlah kota tepat satu kali untuk tiap kota dan kembali kekota asal dengan akumulasi biaya perjalanan yang minimum. Biaya tersebut dapat berupa jarak, waktu, konsumsi bahan bakar dan lain-lain. Masalah perjalanan *salesman* dapat diformulasikan dalam bentuk graf yaitu kota-kota yang dikunjungi dapat dinyatakan sebagai *vertex*, dan jalan yang menghubungkan kota dapat dinyatakan dengan *edge*. Untuk menyatakan jarak suatu kota maka graf diberi bobot (*weight*).

TSP banyak diaplikasikan dalam kehidupan sehari-hari. Pada permasalahan pendistribusian produk dapat dikategorikan sebagai TSP, dimana *salesman* harus mengunjungi beberapa toko tepat satu kali yang dimulai dari gudang menyimpanan dan kembali lagi ke gudang asal. Salah satu contoh pendistribusian barang adalah pendistribusian hasil produksi yang dilakukan oleh sebuah perusahaan. Perusahaan tersebut mendistribusikan hasil produksinya ke beberapa toko pelanggan. Hal menarik yang perlu dikaji adalah pola pendistribusian barang tersebut masih belum sepenuhnya memperhatikan masalah rute terpendek sehingga dimungkinkan

terjadinya inefiseiensi pengiriman barang. Pencarian rute terpendek, dapat diselesaikan dengan berbagai macam metode pendekatan. Contohnya metode heuristik dan metaheuristik.

Pada penelitian sebelumnya telah dilakukan penyelesaian rute terpendek dengan menggunakan beberapa metode heuristik. Purmawanto (2005) membandingkan algoritma *Dijikstra*, algoritma *Floyd* dan algoritma *Queues*, menghasilkan kesimpulan bahwa algoritma *Dijikstra lebih efisien*. Susani (2012) membandingkan algoritma *Dijikstra, Bellman-Ford* dan *Floyd-Warshall*, menghasilkan algoritma *Djikstra* lebih efisien. Setiawan (2010) memberikan kesimpulan bahwa algoritma *Djikstra* lebih cepat dan efisien namun kurang akurat dibandingkan dengan dengan algoritma *A**. Pribadi (2010) membandingkan algoritma *Depth First, Breath First* dan *Hill Climbing*, hasilnya bahwa yang paling efektif diterapkan pada jalur transportasi yang tersedia yaitu algoritma *Breadth First* dan algoritma *Hill Climbing*. Penelitian tentang pencarian rute terpendek dengan metode metaheuristik juga telah dilakukan dengan beberapa algoritma. Saptono dkk (2007) menarik kesimpulan bahwa algoritma *Ant Colony Optimization* terbukti lebih akurat dibandingkan dengan algoritma Genetika. Rahmatullah (2015) menyimpulkan bahwa algoritma *Cat Swarm Optimization* (CSO) lebih efektif dibangdingkan dengan algoritma *Particle Swarm Optimization* (PSO) dan *Ant Bee Colony* (ABC). Sedangkan Setiawan (2010) membandingkan algoritma *Harmony Search* (HS) dengan algoritma *Particle Swarm Optimization* (PSO) untuk menyelesaikan *Resource Constrained Project Scheduling Problem* (RCPS), penelitian tersebut menunjukkan HS memiliki tingkat efisien komputasi yang lebih baik.

Berdasarkan hal diatas peneliti tertarik untuk melakukan penggabungan algoritma yaitu algoritma *Cat Swarm Optimization* (CSO) dengan algoritma *Harmony Search* (HS) yang disebut algoritma Hybrid *Cat Swarm Optimization* (hCSO) untuk penyelesaian *Travelling Salesman Problem* (TSP).

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan, maka masalah yang akan dibahas adalah sebagai berikut:

a. Bagaimana cara penyelesaian TSP menggunakan algoritma *hybrid Cat Swarm Optimization* (hCSO)?
b. Bagaimana penerapan algoritma *hybrid Cat Swarm Optimization* (hCSO) pada program MatLab?
c. Bagaimana hasil dari algoritma hybrid *Cat Swarm Optimization* (hCSO) dibandingkan dengan hasil yang diperoleh *algoritma Cat Swarm Optimization* dan *algoritma Harmony Search* (HS) berdasarkan jarak minimum yang diperoleh, waktu yang dibutuhkan untuk *running* program dan tingkat kekonvergenan?

1.3 Batasan Masalah

Permasalahan dalam skripsi ini dibatasi oleh beberapa hal, yaitu:

a. Biaya perjalanan yang digunakan berupa jarak antar toko pelanggan
b. Jarak toko A ke toko B sama dengan jarak toko B ke toko A
c. Kondisi jalan dalam keadaan sama baik
d. Dalam kasus ini tidak mempertimbangkan waktu tempuh

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini yaitu:

a. Menyelesaikan TSP menggunakan algoritma *hybrid Cat Swarm Optimization* (hCSO)
b. Membuat program algoritma *hybrid Cat Swarm Optimization* (hCSO) untuk menyelesaikan TSP menggunakan Mat Lab.

c. Membandingkan hasil yang diperoleh dari algoritma *hybrid Cat Swarm Optimization* (hCSO) dengan hasil yang diperoleh dari algoritma *Cat Swarm Optimization* dan algoritma *Harmony Search* (HS).

1.5 Manfaat

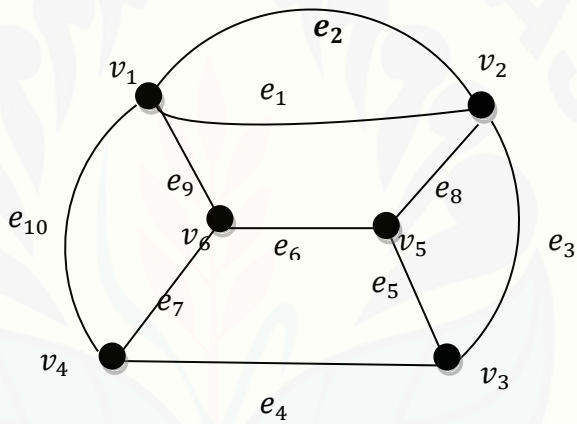
Adapun manfaat yang dapat diambil dari penulisan karya ilmiah ini antara lain:

a. Menambah wawasan tentang pengembangan algoritma gabungan antara *Cat Swarm Opimization* (CSO) dan *Harmony Search* (HS) yang diberi nama *Cat Swarm Optimization* (hCSO).
b. Diharapkan dapat mempermudah, menghemat waktu serta biaya pendistribusian barang.

BAB 2. TINJAUAN PUSTAKA

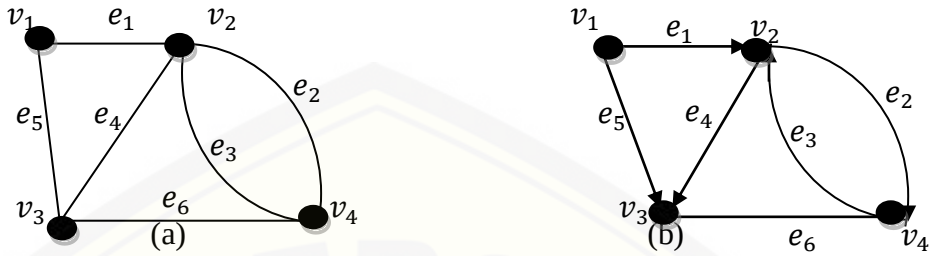
2.1 Teori Graf

Sebuah graf *G* didefinisikan sebagai pasangan himpunan (*V*(*G*), *E*(*G*)) dengan $V = \{v_1, v_2, \ldots v_n\}$ adalah himpunan tak kosong dari elemen-elemen yang disebut titik (vertex) dan $E = \{e_1, e_2, \ldots e_m\}$ adalah himpunan boleh kosong dari elemen-elemen yang disebut sisi (edge) dimana sisi adalah pasangan tak terurut dari dua titik di G(v_i, v_j) atau dapat ditulis $e = v_i, v_j$. Salah satu contoh graf dapat dilihat pada gambar 2.1.



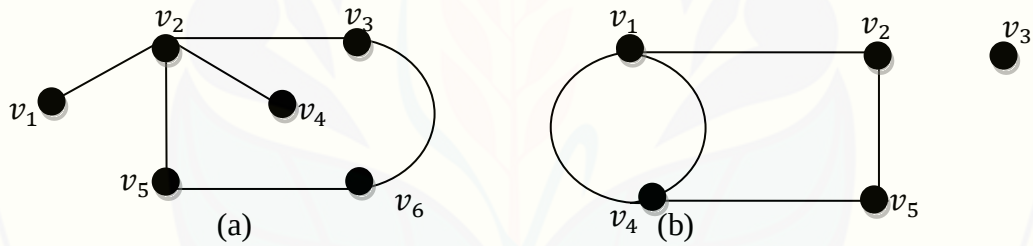
Gambar 2.1 Graf G dengan 6 titik dan 10 sisi

Graf tak berarah adalah graf yang sisinya tidak mempunyai orientasi arah. Pada graf tak berarah, urutan pasangan titik yang dihubungkan oleh sisi tidak diperhatikan. Jadi sisi (v_i, v_j) dan sisi(v_j, v_i) adalah dua sisi yang sama. Sedangkan graf berarah (directed graph atau digraph) adalah graf yang sisi (v_i, v_j) digambarkan oleh segmen garis berarah dari titik v_i ke titik v_j sehingga sisi(v_i, v_j) dan sisi (v_j, v_i) adalaha dua sisi yang berbeda.



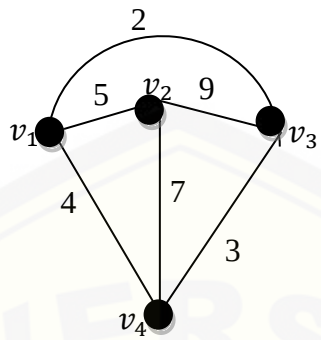
Gambar 2.2 Graf (a) tak berarah dan (b) berarah

Sebuah graf G dikatakan terhubung (connected) jika setiap pasang titik u dan v terdapat lintasan dari u dan v. jika tidak maka disebut graf tak terhubung (disconnected). Contoh graf terhubung dan tidak terhubung dapat dilihat pada gambar 2.3 dibawah ini.



Gambar 2.3. Graf (a) terhubung dan (b) tidak terhubung

Graf berbobot yaitu graf yang masing-masing sisi e diberikan suatu bilangan $w(e)$ yang disebut bobot. Apabila bobot dari v_i ke v_j sama dengan bobot dari v_j ke v_i maka disebut graf berbobot simetrik, jika tidak maka disebut graf berbobot asimetrik. Contoh gambar graf berbobot ditunjukkan pada gambar 2.4.



Gambar 2.4 Graf Berbobot

2.2 *Travelling Salesman Problem* (TSP)

Masalah perjalanan salesman merupakan salah satu persoalan optimasi untuk mendapatkan rute terpendek yang harus dilalui oleh seorang salesman. Salesman tersebut harus mengunjungi sejumlah titik tepat satu kali untuk setiap titik dan kembali kekota asal dengan biaya perjalanan yang minimum. Biaya tersebut berupa jarak antar titik. Terdapat dua jenis TSP, yaitu simetris dan asimetris. Penyelesaian masalah TSP dapat meggunakan beberapa metode diantaranya Algoritma Genetika, Pemrograman Linier, *Hill Climbing*, Algoritma Semut, *Tabu Search*, dan *Simulated Annealing*. Permasalahan dalam bidang transportasi darat merupakan salah satu penerapan TSP dengan tujuan jarak tempuh yang dilalui dan waktu perjalanan seminimum mungkin. Perjalanan tersebut dapat dimodelkan dalam bentuk graf. Graf adalah himpunan yang terdiri dari simpul dan sisi. Simpul merepresentasikan kota, sisi merepresentasikan jalur yang menghubungkan dua kota, dan bobot merepresentasikan biaya, jarak tempuh atau waktu perjalanan yang dikeluarkan. Graf yang digunakan adalah graf lengkap berbobot dan tak berarah.

2.3 Algoritma

Algoritma adalah sebuah himpunan terhingga dari instruksi yang mempunyai karakteristik berikut ini:

a. Presisi (*precision*) yaitu langkahnya dinyatakan dengan jelas
b. Unik (*uniqueness*) yaitu hasil lanjutan dari setiap langkah dari pelaksanaan didefinisikan secara tunggal dan semata-mata bergantung pada masukan dan hasil dari langkah sebelumnya.
c. Terhingga (*finiteness*) yaitu algoritma berhenti setelah beberapa instruksi terhingga dilaksanakan
d. Masukan (input) yaitu algoritma memerlukan masukan
e. Keluaran (output) yaitu algoritma menghasilkan keluaran
f. Umum (generality) yaitu algoritma berlaku pada himpunan masukan.

Algoritma juga diartikan sebagai metode langkah demi langkah dari pemecahan suatu masalah. Langkah-langkah dari suatu algoritma harus dinyatakan dengan jelas sehingga dapat ditulis dalam bahasa pemrograman dan dijalakan computer (Johnsonbaugh, 1998). Algoritma optimasi (*optimization algoritHMS*) dapat didefinisikan sebagai algoritma atau metode numerik untuk menemukan nilai x sedemikian hingga menghasilkan $f(x)$ yang bernilai (sekecil atau sebesar) mungkin untuk suatu fungsi f yang diberikan, yang mungkin disertai dengan beberapa batasan pada x. Disini, x bisa berupa scalar atau vector dari nilai-nilai kontinu maupun diskrit (Suyanto, 2010). Terdapat banyak cara pengklasifikasian algoritma optimasi yang bisa dilakukan, diantaranya adalah: berdasarkan metode operasinya, berdasarkan akurasi dan kecepatannya, serta berdasarkan analogi dan nama yang digunakan.

a. Berdasarkan metode operasinya

Berdasarkan metode operasinya, algoritma optimasi (AO) dapat dibagi ke dalam dua kelas besar, yaitu algoritma deterministik dan algoritma probabilistik. Pada setiap langkash eksekusi dalam algoritma deterministik, terdapat maksimum satu jalan untuk diproses. Jika tidak ada jalan, berarti algoritma sudah selesai. Algoritma deterministik sering digunakan untuk masalah yang memiliki relasi yang jelas antara karakteristik calon solusi dengan utilitasnya. Tetapi, jika relasi antara

suatu kandidat solusi dan *"fitness"*–nya tidak dapat dipahami atau diamati, maka masalah ini akan lebih sulit diselesaikan dengan secara deterministik. Untuk permasalahan dengan ruang pencarian yang sangat besar, biasanya *PAR*a praktisi lebih sering menggunakan algoritma probabilistik.

b. Berdasarkan akurasi dan kecepatan

*PAR*a praktisi yang sedang menghadapi suatu masalah cenderung memilih algoritma optimasi berdasarkan dua hal yaitu akurasi dan kecepatan. Klasifikasi yang bisa digunakan adalah membedakan algoritma optimasi menjadi dua, yaitu optimasi *online* dan optimasi *offline*. Optimasi *Online*, optimasi ini ditujukan untuk permasalahan yang membutuhkan solusi dalam waktu cepat dan biasanya permasalahan tersebut terjadi secara berulang-ulang. Misalnya, penentuan lokasi robot *(robot localization)*, penyeimbangan beban *(load balancing)*. Pada permasalahan tersebut, biasanya hanya diberikan waktu yang singkat.

Optimasi *Offline*, optimasi ini ditujukan untuk permasalahan yang membutuhkan solusi tidak dalam waktu cepat dan biasanya masalah terjadi dalam periode waktu yang lama. Misalnya optimasi dalam proses perancangan *(desgn optimization)*, data mining, pembuatan jadwal kuliah setiap semester dan sebagaingnya.

c. Berdasarkan analogi yang digunakan

Bedasarkan analogi dan nama yang digunakan, algoritma optimasi bisa dibedakan menjadi dua yaitu algoritma minimasi dan algoritma maksimasi. Algoritma minimasi merupakan algoritma yang menggunakan analogi meminimalkan sesuatu pada dunia nyata. Sedangkan algoritma maksimasi merupakan algoritma yang menggunakan analogi memaksimalkan sesuatu pada dunia nyata.

2.4 Cat Swarm Optimization

Cat Swarm Optimization adalah algoritma yang diususlkan oleh Shu-Chan Chu dan Pei-Wei Tsai pada tahun 2006, yang didapat melalui pengamatan terhadap perilaku sekumpulan kucing. Dalam CSO, sekumpulan kucing dan model perilakunya digunakan untuk menyelesaikan permasalahan optimasi. Algoritma CSO dibagi ke dalam dua sub model yang berdasarkan dari dua perilaku utama kucing, yaitu *seeking mode* dan *tracing mode* (Dhanasaputra dan Santosa, 2010).

2.4.1 *Seeking Mode*

Seeking Mode digunakan untuk memedelkan situasi kucing ketika dalam keadaan beristirahat, melihat sekeliling dan mencari posisi berikutnya untuk bergerak. Dalam *seeking mode*, didefinisikan empat faktor penting. yaitu: *seeking memory pool* (*SMP*), *seeking range of the selected dimension* (*SRD*) atau mencari rentang dimensi terpilih, *count of dimension to change* (*CDC*) atau menghitung dimensi yang akan berubah dan *self-position considering* (SPC) atau mempertimbangkan posisi (Dhanasaputra dan Santosa, 2010).

SMP digunakan untuk mendefinisikan ukuran memori pencarian untuk masing-masing kucing, yang mengindikasikan titik-titik yang telah dicoba oleh kucing. Kucing tersebut kemudian akan memilih titik dari kelompok memori berdasarkan aturan yang akan dijelaskan kemudian. *SRD* menyatakan rentang perpindahan dalam dimensi terpilih. Dalam *seeking mode,* jika suatu dimensi diputuskan perpindah, selisih antara nilai baru dengan yang lama tidak melebihi suatu rentang, yaitu rentang yang didefinisikan oleh *SRD*. *CDC* memperlihatkan berapa besar dimensi yang akan berubah. Keseluruhan faktor inilah yang memegang peran penting dalam *seeking mode* (Dhanasaputra dan Santosa, 2009).

SPC merupakan variable Boolean (bernilai benar atau salah), untuk memutuskan apakah suatu titik, yang pernah menjadi posisi kucing, akan menjadi kandidat posisi untuk bergerak. Bagaimanapun nilai *SPC*, entah benar ataupun salah,

nilai *SMP* tidak akan terpengaruh. Langkah-langkah *seeking mode* dapat dideskripsikan dalam 5 tahap:

a. Bangkitkan *j* tiruan dari posisi saat ini kucing$_k$, dimana *j*=*SMP*. Jika nilai SPC benar, maka *j*=(*SMP*-1), kemudian pertahankan posisi saat ini sebagai salah satu kandidat.
b. Untuk setiap tiruan,disesuaikan dengan *CDC*, tambahkan atau kurangkan *SRD* persen dari nilai saat ini secara acak dan gantikan nilai yang sebelumnya.
c. Hitung nilai yang kecocokan (FS) untuk semua titik kandidat.
d. Jika semua FS tidak benar-benar sama, hitung probabilitas terpilih masing-masing titik kandidat dengan menggunakan persamaan (2.1), setidaknya atur probabilitas terpilih untuk semua titik sama dengan 1.
e. Secara acak pilih titik untuk bergerak dari titik-titik kandidat, dan pindahkan posisi kucing$_k$.

$$Probabilitas\ Terpilih_i = \frac{|fitness_i - fitness_{max}|}{fitness_{max} - fitness_{min}} \quad (2.1)$$

2.4.2 Tracing Mode

Tracing mode adalah sub model yang menggambarkan keadaan ketika kucing sedang mengikuti jejak targetnya. Sekali kucing memasuki *tracing mode*, kucing tersebuat akan bergerak sesaui dengan kecepatannya untuk tiap dimensi.

Tahapan *tracing mode* dapat dijabarkan dalam 3 langkah sebagai berikut:

a. Perbaharuhi nilai kecepatan untuk setiap dimensi ($v_{k,d}$) berdasarkan persamaan (2.2).
b. Periksa apakah kecepatan berada dalam rentang kecepatan maksimum. Jika kecepatan yang baru melebihi rentang, tetapkan nilai sama dengan batas.
c. Perbaharui posisi kucing$_k$ berdasarkan persamaan (2.3).

$$v'_{k,d} = v_{k,d} + r\ x\ c\ x\ \left(x_{best,d} - x_{k,d}\right), \quad (2.2)$$

dimana, *k*=1,2, . . . *m* dan *d*=1,2, . . . *n*

$x_{best,d}$ adalah posisi kucing yang memiliki nilai kecocokan terbesar; $x_{k,d}$ adalah posisi kucing$_k$, c_1 adalah konstanta dan r_1 adalah nilai acak dalam rentang [0,1].

$$x'_{k,d} = x_{k,d} + v'_{k,d} \quad (2.3)$$

Seperti yang telah dijabarkan sebelumnya, CSO terdiri dari dua sub model. Untuk mengkombinasikan kedua mode dalam satu algoritma, perlu didefinisikan rasio campuran/*mixture ratio* (MR) untuk menggabungkan *seeking mode* dan *tracing mode*.

Dengan mengamati perilaku kucing, dapat diketahui bahwa kucing menghabiskan sebagian besar waktunya untuk beristirahat. Selama berstirahat, kucing mengubah posisinya perlahan dan berhati-hati, terkadang bahkan tetap pada tempatnya. Untuk menerapkan perilaku ini digunakan *seeking mode*.

Perilaku mengejar target diaplikasikan dalam tracing mode. Karena itu MR harus bernilai kecil untuk memastikan bahwa kucing menghabiskan sebagian besar waktu dalam *seeking mode*, seperti dikehidupan nyata.

Proses CSO dapat digambarkan dalam 6 langkah sebagai berikut:

a. Bangkitkan N kucing dalam proses
b. Sebarkan kucing secara acak dalam ruang solusi berdimensi D dan secara acak pula pilih nilai dalam rentang kecepatan maksimum untuk menjadi kecepatan kucing. Kemudian pilih sejumlah kucing secara sembarang dan masukkan dalam tracing mode sesuai MR, sisanya dimasukkan dalam *seeking mode*.
c. Hitung nilai kecocokan masing-masing kucing dengan memasukkan nilai posisi kucing kedalam fungsi kecocokan, yang menunjukkan criteria tujuan dan simpan kucing terbaik dalam memori. Perlu diingat bahwa yang perlu disimpan adalah posisi kucing terbaik ($x_{best,d}$) karena kucing terbaik sejauh ini mewakili solusi terbaik.

d. Pindahkan kucing sesuai benderanya, jika kucing$_k$ berada dalam *seeking mode*, selanjutnya perlakukan sesuai tracing mode. Proses masing-masing telah dijelaskan sebelumnya.
e. Pilih lagi sejumlah kucing dan masukkan dalam *tracing mode* sesuai MR, sisanya masukkan dalam *seeking mode.*
f. Perhatikan *terminating condition*-nya. Jika telah memuaskan, hentikan program. Sebaliknya ulangi langkah c-e..

2.5 Seleksi *Roulette Wheel*

Seleksi *roullete wheel* menirukan permainan roulette wheel (roda *roullete wheel*) dimana masing-masing chromosome menepati potongan lingkaran pada roda *roulette wheel* secara proposional secara proposional sesuai dengan nilai fitnessnya. *Chromosome* dengan nilai fitness tinggi akan menepati potongan lingkaran yang lebih besar dibandingkan dengan nilai fitness yang lebih kecil (Sundarningsih dkk, 2015). Berikut adalah tahap-tahap seleksi *roulette wheel*:

a. Hitung nilai fitness dari masing-masing individu
b. Hitung total fitness dari semua individu
c. Hitung probabilitas tersebut, hitung jatah masing-masing individu pada angka 0-1
d. Bangkitkan bilangan acak antara 0-1
e. Dari bilangan acak yang dihasilkan, tentukan individu mana yang terpilih dalam seleksi

2.6 Algoritma *Harmony Search* (HS)

Geem mengenalkan algoritma ini pada tahun 2001 yang terinspirasi oleh proses perbaikan harmoni musik yang dilakukan oleh kelompok paduan musik (Hanggraeni, 2013). Terdapat tiga kemungkinan pilihan ketika kelompok paduan musik melakukan perbaikan, yaitu memainkan musik yang terkenal berdasarkan ingatan mereka, memainkan harmoni musik yang serupa dengan harmoni musik yang

terkenal namun ada sedikit penyesuaian, atau membuat harmoni musik yang baru (Suyanto, 2010).

Terdapat tiga kemungkinan pilihan ketika kelompok paduan musik melakukan perbaikan, yaitu memainkan musik yang terkenal berdasarkan ingatan mereka, memainkan harmoni musik yang serupa dengan harmoni musik yang terkenal namun ada sedikit penyesuaian, atau membuat harmoni musik yang baru (Suyanto, 2010). Penggunaan *harmony memory* sangat penting karena *harmony memory* tersebut bisa menjamin bahwa *harmony* yang bagus akan dipertimbangkan sebagai elemen-elemen dari vector posisi yang baru. Agar *harmony memory* dapat digunakan secara efektif, algoritma ini mengadopsi sebuah *PAR*ameter yang disebut *Harmony Memory Considering Rate* (*HMCR*). Jika rate ini terlalu rendah maka, maka hanya sedikit harmony yang akan terpilih dan juga dapat menyebabkan proses konvergensi terlalu lambat. Jika rate terlalu besar, maka akan menyebabkan nada-nada pada *Harmony Memory* banyak terpakai dan tidak sempat mengeksplorasi nada lain, yang pada akhirnya sulit mencapai solusi yang bagus. Oleh karena itu biasanya digunakan *HMCR*=0.7∼0.95. komponen lainnya yaitu *Harmony Memory Size* (*HMS*) yaitu jumlah baris dari matrik *harmony memory* yang akan digunakan pada saat inisalisasi *harmony memory*. *HMS* merupakan sampel kemungkinan dari jumlah solusi yang ada.

Komponen selanjutnya adalah penyesuaian nada dimana mempunyai beberapa *PAR*ameter seperti *bandwidth* (*bw*) dan *Pitch Adjusting Rate* (*PAR*). Penyesuaian nada musik berarti pengubahan frekuensi nada, hal itu berarti membangkitkan nilai yang sedikit berbeda pada algoritma HS. *PAR* yang bernilai rendah dengan *bandwidth* yang sempit dapat menyebabkan proses konvergensi lambat dikarenakan keterbatasan eksplorasi pada ruang pencarian yang besar. Sebaliknya jika *PAR* yang bernilai tinggi dengan *bandwidth* yang lebar menyebabkan solusi-solusi yang ada terlalu menyebar dari potensi solusi optimal. Oleh karena itu biasanya digunakan *PAR*= 0.1∼0.5.

Algoritma HS disusun dengan 5 langkah yang terinspirasi oleh proses pencarian harmoni terbaik yaitu:

a. Inisiasi masalah dan *PAR*ameter algoritma

Pada tahap ini akan dicari jarak dari titik awal ke titik-titik tujuan ataupun jarak antar titik tujuan, rute perjalanan. Mendefinisikan *PAR*ameter-*PAR*ameter yang ada yaitu *HMS* (*harmony Memory Size*), *HMCR* (*Harmony Memory Consideration Rate*), *PAR* (*Pitch Adjustment Rate)* dan *NI* (Nilai Iterasi).

b. Inisiasi Memori Rute

Dalam tahap penentuan memori rute, yang dilakukan adalah menghasilkan rute-rute yang diacak dari rute kluster. Memory rute sendiri adalah kumpulan dari hasil random rute kluster masing-masing hasil random tersebut dinamakan dengan vector rute memory. Menurut Setiawan (2010) pada tahap ini akan dibangkitkan sebuah matriks harmony memory secara acak yang berisikan vector-vektor solusi sebanyak *HMS*. Berikut ini adalah contoh matriks *Harmony Memory* yang ditunjukkan pada gambar 2.6. sedangkan nilai fungsi ditunjukkan pada gambar 2.7.

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix}$$

Gambar 2.5 Matriks Harmony Memory

Dimana masing-masing vector solusi akan dievaluasi nilai fungsinya,

$$f(x) = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x^{HMS-1}) \\ f(x^{HMS}) \end{bmatrix}$$

Gambar 2.6 Nilai Fungsi

dengan,

$f(x)$= fungsi tujuan

HM= Matriks *Harmony memory*

N = jumlah variable keputusan

HMS= jumlah baris matrik *Harmony Memory*

x^{HMS}= variable keputusan ke-N pada sebanyak *HMS*

c. Membuat vector solusi yang baru

Untuk membentuk vector solusi yang baru $x' = (x_1^1, x_2^1, \ldots x_N^1)$, didasarkan pada aturan tertentu yang dilakukan dengan membandingkan nilai yang dibangkitkan secara random dengan *HMCR* dan *PAR*. Pembangkitan vector solusi yang baru secara random didasarkan pada aturan berikut:

1.Penggunaan *Harmony Memory*

Pembangkitan vector solusi yang baru diambil dari HM, jika nilai pertama dari bilangan random yang dibangkitkan lebih kecil dari *HMCR* dan nilai kedua dari bilangan random yang dibangkitkan lebih besar dari *PAR*. Formulanya sebagai berikut:

$$d_1 = int[1 + (HMS - 1)rand]$$
$$d_2 = HM(d, i)$$
$$x_1' = d_2$$

dengan,

$d_1 =$ nilai yang menyatakan pemilihan lokasi variable keputusan pada harmony memory secara random

$int =$integer

d_2= nilai yang menyatakan pemilihan lokasi *variabel* keputusan yang diambil dari *Harmony Memory*

2.Penyesuaian nada

Pembangkitan vektor solusi yang baru dilakukan melalui proses penyesuaian nada, jika nilai pertama dari bilangan random yang dibangkitkan lebih kecil dari

HMCR dan nilai kedua dari bilangan random yang dibangkitkan lebih kecil dari *PAR*. Rumus penyesuaian nada yaitu:

$$d_3 = d_2 + bw \times rand[-1,1]$$

$$x_1' = d_3$$

dengan,

x_1'= nada baru setelah dilakuka penyesuaian nada

bw= bandwidth

3.Proses pembangkitan secara random

Pembangkitan vector solusi yang baru secara random dengan nilai internal yang memungkinkan $(x_1 \in X_i)$. Hal ini terjadi apabila nilai pertama dari bilangan random yang dibangkitkan lebih besar dari *HMCR*, tidak perlu membangkitkan nilai kedua karena nilai yang dibangkitkan pertama sudah lebih besar dari *HMCR*.

d. Update Memori Rute

Pada tahap ini dilakukan perhitungan nilai fumgsi objektif pada vector memori rute baru. Jika hasil perhitungan vector rute baru menghasilkan nilai fungsi objektif yang lebih baik daripada vector memori rute terburuk di HM, maka vector memori rute baru akan dimasukkan kedalam HM dan vector memori rute terburuk dalam HM dikeluarkan dari HM.

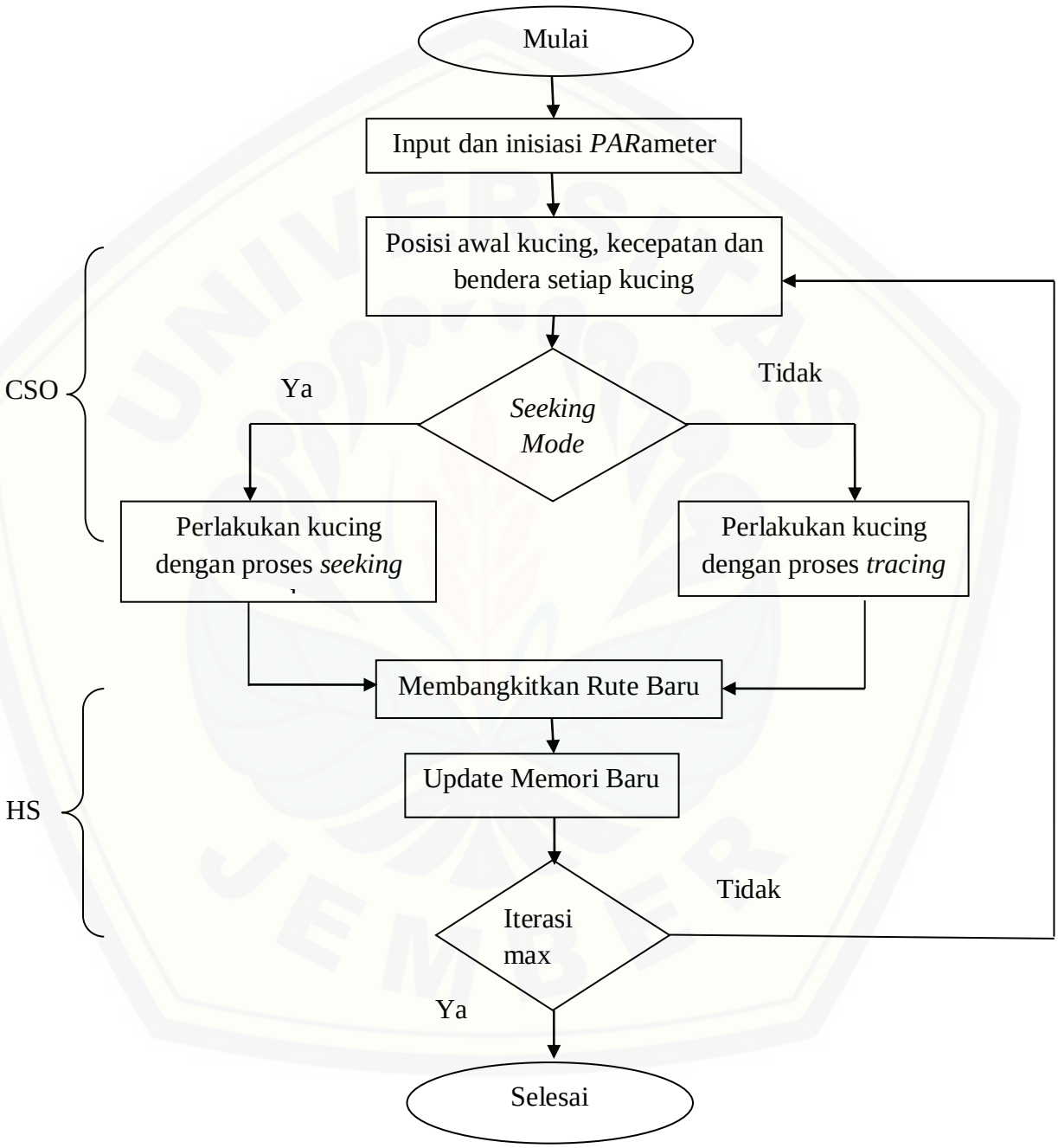
e. Pemberhentian

Pemberhentian merupakan langkah dimana jika jumlah iterasi sudah dipenuhi, maka perhitungan berhenti jika tidak dilanjutkan ke langkah ke-3 dan ke-4.

2.7 *Hybrid Cat Swarm Optimization* (hCSO)

Hybrid Cat Swarm Optimization (hCSO) merupakan gabungan dari *algoritma Cat Swarm Optimization* (CSO) dengan algoritma *Harmony Search* (HS). Proses algoritma *Harmony Search* (HS) diletakkan setelah proses algoritma *Cat Swarm Optimization* (CSO) sebelum dilakukan iterasi berikutnya agar didapatkan solusi

yang lebih baik. Adapun *flowchat* dari algoritma hCSO dapat dilihat pada Gambar 2.8.



Gambar 2.7 *Flowchart* Algoritma hCSO

2.8 Kriteria Kekonvergenan

Pemberhentian atau kriteria kekonvergenan pada suatu algoritma dapat dituliskan sebagai berikut:

a. Iterasi maksimum

Suatu algoritma akan berhenti ketika mencapai iterasi maksimum yang telah ditentukan.

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 fungsi objektif tidak mengalami perubahan

Proses algoritma akan berhenti jika tidak ada perubahan pada nilai fungsi objektif yang optimal.

2.9 Pengantar Matlab

Matlab (*Matrix Laboratory*) merupakan suatu bahasa pemrograman lanjutan yang dibentuk dengan dasar pemikiran menggunakan sifat dan bentuk dari matriks. Matlab awalnya dikembangkan dengan menggunakan bahasa pemrograman Fortran, namun sekarang ini sudah merupakan produk komersial dari perusahaan Mathworks.Inc yang dalam perkembangan selanjutnya dikembangkan bahasa C++ dan Assembler (terutama untuk fungsi-fungsi dasar) (Farida, 2006).

Matlab merupakan bahasa canggih untuk komputasi teknik. Matlab merupakan integrasi dari komputasi, visualisasi dan pemrograman dalam suatu lingkungan yang mudah digunakan, karena permasalahan dan pemecahannya dinyatakan dalam notasi matematika biasa. Kegunaan Matlab secara umum adalah untuk:

a. Matematika dan Komputasi

b. Pengembangan algoritma

c. Pemodelan, simulasi dan pembuatan prototype

d. Analisa data, eksplorasi dan visualisasi

e. Pembuatan aplikasi termasuk pembuatan GUI (*Graphical User Interface*)

Matlab adalah system interaktif dengan elemen dasar array yang merupakan basis datanya. Array tersebut tidak perlu dinyatakan khusus seperti di bahasa pemrogrman yang ada sekarang. Hal ini memungkinkan untuk memecahkan banyak masalah perhitungan teknik, khususnya yang melibatkan matriks dan vektor dengan waktu yang lebih singkat dari waktu yang dibutuhkan untuk menulis program dalam bahasa C atau *Fortran*. Untuk memahami Matlab, terlebih dahulu harus sudah paham mengenai matematika terutama operasi vector dan matriks, karena operasi matriks merupakan inti utama dari Matlab.

Pada intinya Matlab merupakan sekumpulan fungsi-fungsi yang dapat dipanggil dan dieksekusi. Fungsi-fungsi tersebut dibagi-bagi berdasarkan kegunaannya yang dikelompokkan didalam *toolbox-toolbox* yang ada pada Matlab. Toolbox merupakan kumpulan koleksi dari fungsi-fungsi Matlab (M-files) yang memperluas lingkungan Matlab untuk memecahkan masalah-masalah tertentu. *Toolbox-toolbox* yang tersedia pada Matlab antara lain:

a. *Signal Processing Toolbox*

b. *Control Systems Toolsbox*

c. *Neural Networks Toolbox*

d. *Fuzzy Logic Toolbox*

e. *Wavalets Toolbox*

f. *Simulation Toolbox*

g. *Image Processing Toolbox*

Matlab juga memiliki sifat *extensible*, dalam arti bahwa pengguna dari Matlab dapat membuat suatu fungsi baru untuk ditambahkan kedalam library jika fungsi-fungsi *built-in* yang tersedia tidak dapat melakukan tugas tertentu.

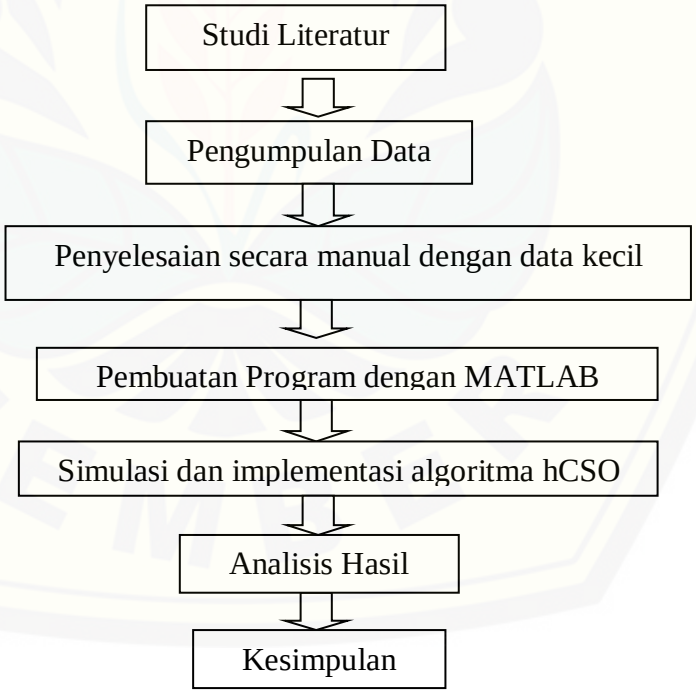
BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam penelitian ini adalah data sekunder yang diperoleh dari penelitian sebelumnya yang dilakukan oleh Sari (2011). Data dalam penelitian ini berupa data lokasi beberapa kios dan jarak antar kios dari pelanggan UD. Tani Lumintu. Kios-kios tersebut dipresentasikan sebagai titik dan kode titik. sedangkan jarak dipresentasikan sebagai bobot sisi. Data jarak antar titik disajikan pada Lampiran 1.

3.2 Langkah-langkah Penelitian

Langkah-langkah yang dilakukan untuk menyelesaikan pencarian rute terpendek pada kasus rute tempat pariwisata di Jember dapat dilihat di Gambar 3.1 berikut:



Gambar 3.1 Langkah Penelitian

Adapun penjelasan dari skema langkah-langkah penelitian pada Gambar 3.1 adalah sebagai berikut:

a. Studi Literatur

Langkah awal yang dilakukan adalah mengumpulkan literature tentang algoritma-algoritma metaheuristik khususnya *Cat Swarm Optimization* dan *Harmony Search*. Dan literature tentang pencarian rute terpendek atau *Shortest Path* dari internet maupun buku-buku.

b. Pengambilan dan Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data sekunder yang diperoleh dari penelitian sebelumnya yang dilakukan oleh Sari (2011). Data dalam penelitian ini berupa data lokasi beberapa kios dan jarak antar kios dari pelanggan UD. Tani Lumintu

c. Penyelesaian secara manual dengan data kecil

Pada langkah ini akan dilakukan penyelesaian secara manual mencari rute terpendek dengan algoritma CSO dan HS menggunakan 10 data.

1) Penerapan algoritma Cat Swarm Optimization untuk pencarian rute terpendek

Adapun langkah-langkah yang harus dilakukan dalam algoritma *Cat Swarm Optimization* adalah sebagai berikut:

Langkah 1: Input dan inisiasi parameter

Prosedur ini merupakan prosedur untuk input data dan menentukan parameter-parameter yang dibutuhkan dalam algoritma hCSO. Yaitu diantaranya jumlah populasi kucing (*m*), jumlah titik yang dikunjungi (*n*), *SMP*, *CDC*, *SRD*, *MR*, *c*, *HMR*, *HMCR*, *PAR*, *bw*, iterasi maksimum.

Langkah 2: Pembangkitan populasi awal

Setiap individu (kucing) merupakan representasi dari satu set solusi yang terdiri atas *n* dimensi. Untuk pembangkitan populasi awal, dibuat matrik dengan ukuran *m* x *n*. Matrik ini berisikan bilangan acak yang mempunyai nilai antara 0 dan 1.

Langkah 3: Pembangkitan kecepatan awal

Setiap dimensi dari setiap individu juga memiliki inisiasi lain yakni kecepatan awal (*initial velocity*). Kecepiatan awal adalah suatu bilangan acak yang terkait dimensi dari setiap individu setiap individu (kucing). Pembangkitan kecepatan awal dilakukan dengan membuat matriks berukuran *m* x *n*.

Langkah 4: Representasi permutasi

Langkah ini bertujuan untuk menghitung nilai fungsi tujuan dari setiap solusi awal yang telah dibangkitkan. Sebelum menghitung nilai fungsi tujuan langkah yang harus dilakukan terlebih dahulu adalah mengubah bilangan *real* ke bentuk permutasi secara *descending* (kecil-besar).

Langkah 5: Menghitung fungsi tujuan.

Permasalahan *Travelling Salesman Problem* dibuat dengan tujuan untuk meminimumkan total jarak tempuh. Yaitu dengan menggunakan pengurutan biasa dengan bilangan acak terkecil disusun

Langkah 6: Menghitung nilai *fitness*

Dalam skripsi ini, nilai *fitness* (*fitness value*) yang didapat setiap individu tergantung pada jarak tempuh yang didapat dari representasi permutasi. Cara menghitung nilai *fitness* kucing ke-*i* adalah 1 dibagi jarak yang diperoleh tiap set solusi kucing ke-*i*. Selanjutnya nilai *fitness* diurutkan dari yang terbesar hingga terkecil.

Langkah 7: Menghitung *Self Position Considering* (*SPC*)

SPC merupakan nilai *boolean* yang melabeli setiap individu sebelum melakukan proses seeking dan tracing. Dalam penempatan *SPC* dilakukan dengan cara menempatan label *SPC*=1 terhadap individu yang mempunyai nilai *fitness* tertinggi dan sisanya mempunya label *SPC*=0 secara otomatis.

Langkah 8: Menentukan *Flag*

Setiap individu akan ditempatkan kedalam suatu mode yaitu mode seeking dan mode tracing. Jumlah kucing yang masuk kedalam masing-masing mode sesuai dengan nilai *MR*.

Langkah 9: Melakukan Proses *Mode Tracing*

Pada mode ini setiap individu akan mengubah representasi kecepatan yang lama menjadi representasi kecepatan yang baru. Proses update kecepatan ini

dipengaruhi oleh memory yang dipengaruhi oleh setiap kucing. Adapun rumus untuk mengupdate kecepatan baru menggunakan persamaan 2.2. kemudian mengubah posisinya dengan persamaan 2.3.

Langkah 10: Melakukan Proses *Mode Seeking*

Pada Mode *Seeking* terdapat Modifikasi Memory Pool yang diperuntukkan untuk melakukan modifikasi terhadap set solusi dari tiruan individu. Hasil modifikasi akan diproses dalam *Memory Pool*. Modifikasi yang dilakukan ini tergantung pada parameter *CDC, SRD,* dan *SMP*.

2) Penerapan algoritma *Harmony Search* untuk pencarian rute terpendek

Langkah 11: Membangkitkan rute baru

Pada proses ini membuat vektor solusi baru dengan cara membangkitkan bilangan acak. Langkah ini dipengaruhi oleh parameter *HMCR, PAR, bw.*

Langkah 12: Update memori rute

Pada tahap ini dilakukan perhitungan nilai fumgsi objektif pada vector memori rute baru. Jika hasil perhitungan vector rute baru menghasilkan nilai fungsi objektif yang lebih baik daripada vector memori rute terburuk di HM, maka vector memori rute baru akan dimasukkan kedalam HM dan vector memori rute terburuk dalam HM dikeluarkan dari HM.

Langkah 13: Pemberhentian

Pemberhentian merupakan langkah dimana jika jumlah iterasi (NI) sudah dipenuhi, maka perhitungan berhenti, jika tidak dilanjutkan ke langkah ke-9.

d. Pembuatan Program dengan MATLAB

Langkah penelitian ini yaitu pembuatan program dengan menggunakan software matematika yaitu Matlab. Pada langkah ini peneliti membuat script program dan desain program berupa tampilan pada GUI berdasarkan metode yang digunakan. Program ini menggunakan algoritma *hybrid Cat Swarm Optimization* (hCSO) untuk menyelesaikan rute terpendek.

e. Simulasi dan Implementasi Algoritma hCSO

Pada langkah akan dilakukan simulasi dan implementasi dari program yang telah dibuat dengan input data yang telah diperoleh.

f. Analisis Hasil

Pada langkah ini akan dilakukan analisis terhadap keluaran program. Algoritma ini dianggap baik jika didapatkan nilai yang minimum dan konvergen. Dalam hal ini peneliti juga membandingkan hasil yang diperoleh hCSO dengan algoritma CSO dan HS.

g. Kesimpulan

Peneliti akan menarik kesimpulan dari hasil analisis yang diperoleh

Bab 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan diperoleh kesimpulan :

a. Langkah penyelesaian TSP dengan algoritma hCSO adalah input dan inisiasi parameter, pembangkitan populasi awal, pembangkitan kecepatan awal, representasi permutasi, menghitung fungsi tujuan, menghitung nilai *fitness*, menghitung *Self Position Considering* (*SPC*), menentukan penentuan *Flag*, melakukan Proses *Mode Tracing*, melakukan proses *Mode Seeking*, membangkitkan rute baru, update memori rute dan pemberhentian.
b. Semakin besar parameter *m* maka jarak yang dihasilkan semakin minimum namun waktu *running* semakin lama. Semakin besar parameter *MR* yang diberikan maka waktu *running* yang dibutuhkan semakin cepat. Semakin besar parameter *SMP* yang diberikan maka jarak yang dihasilkan semakin minimum namun waktu semakin lama. Jarak paling minimum yang diperoleh yaitu 151.2 dengan rute yaitu 1-20-11-19-10-28-29-30-31-26-25-24-22-23-21-8-7-6-5-4-3-2-12-13-14-15-16-17-18-27-9-1 atau UD Tani Lumintu – Sari Bumi – Toko Krisna Makmur – UD Hasil Tani – Mbah Jenggot – UD Wasilah – Kios Subur – Al-Ikhlas Mitra Tani – UD Panorama Mbah Taujan – Rizki Abadi – Sumber Abadi – Tani Gemilang – Anisa Jaya – UD Mitra Amanah – Jaya Tani – Abdi Tani – Mitra Tani – Sumber Rejeki (1) – Usaha Tani – Sahabat Tani – Sinar Tani (1) – Tani Lumintu (3) – Sri Rejeki – Cahaya Tani – UD Tani Mulyo – Toko Rizki – Witanto – Sinar Tani (2) – Sumber Rejeki (2) – Muncar Tani – Sampoerna - UD Tani Lumintu.
c. Dari hasil tersebut dapat diperoleh bahwa algoritma yang lebih efektif untuk mencari jarak minimum adalah algoritma hCSO dibandingkan dengan algoritma

CSO dan HS. Sedangkan algoritma yang lebih efektif dari segi waktu yaitu algoritma HS dibandingkan dengan algoritma CSO dan hCSO.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, disarankan untuk peneliti selanjutnya untuk melakukan penggabungan algoritma CSO dengan algoritma yang lain guna untuk mencari hasil yang optimum.

DAFTAR PUSTAKA

Dhanasaputra, N & Santosa, B. 2010. *Pengembangan Algoritma Cat Swarm Optimization (CSO) untuk Klasifikasi*. Artikel. http://www.researchgate.net/publication/267242295_Pengembangan_Algoritma_Cat_Swarm_Optimization_Untuk_Klasifikasi. Diunduh pada 20 Oktober 2015.

Farida. 2006. Pengklasifikasian Gender Dengan Menentukan Titik -Titik Penting Pada Sistem Pengenalan Wajah Menggunakan Matlab 6.5. Artikel. http://www.gunadarma.ac.id/library/articles/graduate/industrialtechnology/2006/Artikel_50402391.pdf. Diunduh pada 15 Februari 2016.

Ningati, R. 2014. *Aplikasi Pencarian Rute Terpendek Daerah Wisata Kota Kediri menggunakan Algoritma Dijikstra*. Skripsi. Universitas Nusantara PGRI Kediri.

Pribadi, F. S & Mulwinda, Anggraini. 2010. Pencarian Rute Terpendek dengan Menggunakan Algoritma Depth First dan Hill Climbing (Study Comparative). *Jurnal Kompetensi Teknik*. B (1):57-64.

.

Saptono, F., Mutakhirih, I., Hidayat, T., & Fauziah, A. 2007. Perbandingan Performansi Algoritma Genetika dan Algoritma Semut untuk Penyelesaian Shortest Path Problem . *Seminar Nasional Sistem dan Informatika*. SNS107-043:246-251.

Saptono, F. & Hidayat, T. 2007. Perancangan Algoritma Genetika untuk Menentukan Jalur Terpendek. *Seminar Nasional Aplikasi Teknologi Informasi*. B75-B79.

Sari, R. 2011. Optimasi *Rute Travelling Salesman Problem* (TSP) dengan algoritma A*. Tidak Dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Setiawan, W. 2010. *Pembahasan Pencarian Lintasan Terpendek Menggunakan Algoritma Dijikstra dan A**. Makalah IF3051 Srategi Algoritma-Sem.I Tahun 2010/2011.

Sundarningsih, D. 2015. *Penerapan Algoritma Genetika untuk optimasi Vehicle Routing Problem with Time Window (VRPTW) Studi kasus Air Minum kemasan*. Tidak Dipublikasikan. Skripsi. Malang: Jurusan Teknik Informatika Fakultas PTIIK Universitas Brawijaya.

Susani, 2012. *Perbandingan Algoritma Djikstra, Bellman-Ford, dan Floyd-Warshall untuk mencari Rute Terpendek*. Skripsi. Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Purwanto, Y., Purwitasari, D. & Wibowo, A. 2005. Implementasi dan Analisis Algoritma Pencarian Rute Terpendek di Kota Surabaya. *Jurnal Penelitian dan Pengembangan Telekomunikasi*. **10** (2): 94-101.

Puspitasari, Irinne. 2013. *Algoritma Harmony Search dalam Optimalisasi Vehicle Routing Problem with Time window* (VRPTW). Tidak Dipublikasikan. Skripsi. Malang: Jurusan Matematika Fakultas MIPA Universitas Negeri Malang.

Suyanto. 2010. *Algoritma Optimasi*. Yogyakarta: Graha Ilmu.