PENERAPAN *CAT SWARM OPTIMIZATION ALGORITHM* (CSO) DAN *PARTICLE SWARM OPTIMIZATION ALGORITHM* (PSO) DALAM PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP)

SKRIPSI

Oleh
Asri Rizky Dyah Vitaloka
NIM 121810101084

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PENERAPAN *CAT SWARM OPTIMIZATION ALGORITHM* (CSO) DAN *PARTICLE SWARM OPTIMIZATION ALGORITHM* (PSO) DALAM PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP)

SKRIPSI

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

Oleh
Asri Rizky Dyah Vitaloka
NIM 121810101084

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Dengan kehadirat Allah S.W.T atas segala limpahan rahmat dan karunia-Nya serta sholawat salam kepada Nabi besar Muhammad S.A.W, saya persembahkan kebahagiaan atas perjuangan yang telah saya lakukan teriring rasa terima kasih kepada:

1. Ahmad Kamsyakawuni, S.Si., M.Kom selaku dosen pembimbing utama dan Kusbudiono, S.Si., M.Si selaku dosen pembimbing anggota yang telah membimbing saya selama menjadi mahasiswa serta meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Drs. Rusli Hidayat, M.Sc selaku dosen penguji I dan Kosala Dwidja Purnomo, S.Si., M.Si selaku dosen pembimbing akademik dan dosen penguji II yang telah meluangkan waktu, pikiran dan tenaga untuk perbaikan tulisan ini;
3. seluruh dosen Jurusan Matematika atas ilmu, bimbingan dan didikan selama menjadi mahasiswa Matematika;
4. guru-guru SD, SMP, dan SMA yang telah membimbing dan memberikan ilmu dengan penuh kesabaran;
5. ibu Hamida, Mama Asri Rizki, dan Ayah Tosari serta keluarga besar terima kasih banyak atas kasih sayang, dorongan moral maupun material, serta doa tiada putusnya yang telah diberikan kepada saya;
6. keluarga besar di Jember dan Surabaya yang turut mendoakan, memberikan dukungan, dan semangat dalam hidup saya;
7. sahabat saya Fitri Martha Mardhani dan Musdzalifah yang telah memberikan semangat, do'a, dan motivasi;
8. teman-teman Jurusan Matematika FMIPA Universitas Jember angkatan 2012 semuanya yang telah memberikan bentuk persahabatan yang indah dan motivasi dalam penyelesaian tugas akhir ini.

MOTO

"Mahkota seseorang adalah akalnya, derajat seseorang adalah agamanya, sedangkan kehormatannya adalah budi pekertinya"

(Umar bin Khatab)

"Kemenangan yang seindah-indahnya dan sesukar-sukarnya yang boleh direbut oleh manusia ialah menundukkan diri sendiri"

(Ibu Kartini)

---

*) @Tausiah_Hati

**) http://thefilosofi.blogspot.co.id/2014/07/kumpulan-motto-hidup-terbaik-lengkap.html?m=1

PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama : Asri Rizki Dyah Vitaloka

NIM : 121810101084

menyatakan dengan sesungguhnya bahwa skripsi yang berjudul: Penerapan *Cat Swarm Optimization Algorithm* (CSO) dan *Particle Swarm Optimization Algorithm* (PSO) dalam Penyelesaian *Travelling Salesman Problem* (TSP) adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, dan belum diajukan pada instansi manapun, serta 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 adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Desember 2016

Yang menyatakan,

Asri Rizky Dyah Vitaloka
NIM. 121810101084

SKRIPSI

PENERAPAN *CAT SWARM OPTIMIZATION ALGORITHM* (CSO)
DAN *PARTICLE SWARM OPTIMIZATION ALGORITHM* (PSO)
DALAM PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* (TSP)

Oleh

Asri Rizky Dyah Vitaloka

NIM 121810101084

Pembimbing

Dosen Pembimbing 1 : Ahmad Kamsyakawuni, S.Si., M.Kom

Dosen Pembimbing 2 : Kusbudiono, S.Si., M.Si

PENGESAHAN

Skripsi berjudul ”Penerapan *Cat Swarm Optimization Algorithm* (CSO) dan *Particle Swarm Optimization Algorithm* (PSO) dalam Penyelesaian *Travelling Salesman Problem* (TSP)” telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas MIPA Universitas Jember.

Tim Penguji :

Dosen Pembimbing Utama,	Dosen Pembimbing Anggota,
Ahmad Kamsyakawuni,S.Si., M.Kom. NIP 197211291998021001	Kusbudiono, S.Si., M.Si. NIP 197704302005011001
Dosen Penguji I,	Dosen Penguji I,
Drs. Rusli Hidayat, M.Sc. NIP 196610121993031001	Kosala Dwidja Purnomo, S.Si., M.Si. NIP 196908281998021001

Mengesahkan
Dekan,

Prof. Dr. Sujito, Ph.D.

NIP 196102041987111001

RINGKASAN

Penerapan *Cat Swarm Optimization Algorithm* (CSO) dan *Particle Swarm Optimization Algorithm* (PSO) dalam Penyelesaian *Travelling Salesman Problem* (TSP); Asri Rizki Dyah Vitaloka, 121810101084; 2016; 83 Halaman; Jurusan Matetika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Kehidupan manusia tidak akan lepas dari permasalahan dalam bidang transportasi, misalnya seorang *salesman*. *Salesman* yang harus mengantarkan barang kebeberapa tempat tujuan dengan banyak alternatif rute yang dapat ditempuh untuk sampai ke semua pelanggan. Dengan banyak alternatif rute tersebut menimbulkan permasalahan bagi *salesman* karena harus menentukan rute optimal yang akan ditempuh untuk sampai ke tempat pelanggan, permasalahan ini lebih sering disebut *Travelling Salesman Problem* (TSP) yaitu permasalahan seorang *salesman* yang harus menentukan rute terpendek dalam sebuah perjalanannya yang melalui sejumlah kota.

Tujuan dari tugas akhir ini yaitu menerapkan algoritma *Cat Swarm Optimization* (CSO) dan algoritma *Particle Swarm Optimization* (PSO) dalam penyelesaian *Travelling Salesman Problem* (TSP) dengan menggunakan data 50 kota. Penelitian dilakukan dengan beberapa langkah yaitu mulai, *Study Literature*, mengambil data mengenai permasalahan TSP sehingga menggunakan data yaitu 50 kota yang diambil dari *library* TSP *asymmetric*, kemudian mengolah data dengan algoritma CSO dan algoritma PSO, selanjutnya membuat program Matlab yang sesuai dengan algoritma CSO dan algoritma PSO, kemudian mensimulasi hasil dari program yang telah dijalankan, selanjutnya menganalisis hasil dari perbandingan hasil algoritma CSO dan algoritma PSO, selanjutnya menyimpulkan hasil yang didapat dari penerapan algoritma tersebut.

Algoritma CSO dan algoritma PSO merupakan algoritma yang tergolong metaheuristik yaitu proses pembangkitan solusi awal dilakukan dengan membangkitkan bilangan random. Percobaan dilakukan *running* sampai 10 kali, kemudian diambil jarak paling minimum yang sesuai dengan tujuan permasalahan TSP. Algoritma CSO untuk data 50 kota pada penelitian ini diperoleh jarak paling minimum yaitu 1808 km yang konvergen pada titik 1600 dengan iterasi maksimum 3000. Algoritma PSO untuk data 50 kota pada penelitian ini diperoleh jarak paling minimum yaitu 1808 km yang konvergen pada titik 200 dengan iterasi maksimum 3000.

Berdasarkan hasil yang telah diperoleh, algoritm*a Cat Swarm Optimization* (CSO) lebih efektif dibandingkan algoritm*a Particle Swarm Optimization* (PSO), karena meskipun total jarak tempuh yang dihasilkan sama jika dilihat dari hasil *running* program algoritma CSO selalu menghasilkan total jarak tempuh yang lebih minim dibandingkan algoritma PSO.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul Penerapan *Cat Swarm Optimization Algorithm* (CSO) dan *Particle Swarm Optimization Algorithm* (PSO) dalam Penyelesaian *Travelling Salesman Problem* (TSP). Skripsi ini disusun untuk memenuhi salah satu syarat untuk menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Pada kesempatan ini penulis mengucapkan terima kasih atas bantuan dan bimbingan dalam penyusunan skripsi ini, terutama kepada yang terhormat:

1. Drs. Sujito, Ph.D., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
2. Kusbudiono S.Si., M.Si., selaku Ketua Jurusan Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
3. Ahmad Kamsyakawuni, S.Si., M.Kom., selaku Dosen Pembimbing Utama, Kusbudiono S.Si., M.Si., selaku Dosen Pembimbing Anggota, Drs. Rusli Hidayat, M.Sc., selaku dosen Penguji I dan Kosala Dwidja Purnomo S.Si., M.Si., selaku dosen penguji II yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
4. ibu Hamida, Mama Asri Rizki, dan Ayah Tosari serta keluarga besar terima kasih banyak atas kasih sayang, dorongan moral maupun material, serta doa tiada putusnya yang telah diberikan kepada saya;
5. keluarga besar di Jember dan Surabaya yang turut mendoakan, memberikan dukungan, dan semangat dalam hidup saya;
6. sahabat saya Fitri Martha Mardhani dan Musdzalifah yang telah memberikan semangat, do'a, dan motivasi;

7. teman-teman Jurusan Matematika FMIPA Universitas Jember angkatan 2012 semuanya yang telah memberikan bentuk persahabatan yang indah dan motivasi dalam penyelesaia tugas akhir ini;
8. teman seperjuanganku Shelda Amy, Rafika, dan para pejuang riset yang telah membantu;
9. semua pihak yang telah membantu terselesaikannya skripsi ini.

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

Jember, Desember 2016 Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Manusia tidak akan lepas dari permasalahan dalam bidang transportasi, misalnya seorang *salesman*. *Salesman* yang harus mengantarkan barang kebeberapa tempat tujuan dengan banyak alternatif rute yang dapat ditempuh untuk sampai ke semua pelanggan. Dengan banyak alternatif rute tersebut menimbulkan permasalahan bagi *salesman* karena harus menentukan rute optimal yang akan ditempuh untuk sampai ke tempat pelanggan. Dalam menentukan rute yang optimal, tentunya permasalahan ini harus mempunyai penyelesaian yang baik sehingga *salesman* dapat meminimumkan total jarak tempuh. Permasalahan *salesman* tersebut lebih dikenal sebagai *Travelling Salesman Problem* (TSP).

TSP merupakan masalah seorang *salesman* dalam mengantarkan barang untuk mencari rute perjalanan ke sejumlah kota dengan tujuan mencari rute yang memiliki total jarak tempuh yang paling minimum. *Salesman* harus berangkat dari suatu kota kemudian mengunjungi kota-kota lain untuk mengirim barang dengan masing-masing satu kali kunjungan dan kembali ke kota asal. Rute kota yang dilalui oleh *salesman* dapat digambarkan oleh suatu graf, dimana kota yang dikunjungi oleh *salesman* sebagai *vertex*, jalan yang menghubungkan kota-kota tersebut sebagai *edge*, dan jarak antar kota sebagai bobot (*weight*). Sehingga dari gambaran graf tersebut, permasalahannya yaitu bagaimana mencari lintasan tertutup graf dengan jumlah bobot yang minimum.

Travelling Salesman Problem (TSP) dapat dibagi menjadi dua, yaitu TSP simetris dan TSP asimetris. Perbedaan TSP simetris dengan TSP asimetris yaitu masalah biaya, pada TSP simetris biaya dari kota 1 ke kota 2 sama dengan biaya dari kota 2 ke kota 1. Sedangkan untuk TSP asimetris biaya dari kota 1 ke kota 2 berbeda dengan biaya dari kota 2 ke kota 1. Pada penelitian ini akan menggunakan TSP

asimetris dimana terdapat perbedaan biaya dari kota 1 ke kota 2 dan begitu sebaliknya.

Banyak algoritma yang telah dikembangkan untuk penyelesaian permasalahan *salesman*, misalnya algoritma *Particle Swarm Optimization* (PSO), *Harmony Search Algorithm* (HS), *Cat Swarm Optimization* (CSO). Lestari (2016) yang telah menyelesaikan permasalahan TSP dengan menggunakan Algoritma *Hybrid Cat Swarm Optimization* (hCSO), selain itu juga dibandingkan algoritma HS, algoritma CSO, dan algoritma hCSO. Lestari menyimpulkan bahwa algoritma yang lebih efektif untuk mencari jarak minimum adalah hCSO dibandingkan dengan algoritma CSO dan HS. Sedangkan algoritma yang lebih efektif dari segi waktu yaitu algoritma HS dibandingan dengan algoritma CSO dan hCSO. Selain itu algoritma CSO juga dapat diterapkan untuk permasalahan lainnya, misalnya Baihaki (2016) telah menerapkan algoritma CSO untuk sistem persamaan non-linier. Disimpulkan bahwa algoritma CSO memiliki akurasi yang baik dalam pencarian solusi sistem persamaan non-linier karena solusi yang dihasilkan mendekati solusi eksak.

Algoritma CSO merupakan algoritma yang terinspirasi oleh perilaku *cat* (kucing) saat dia beristirahat dan menangkap mangsa. Algoritma ini termasuk algoritma metaheuristik yaitu metode untuk menyelesaikan persoalan optimasi secara efisien. Selain itu algoritma lain yang efektif dalam persoalan optimasi adalah *Particle Swarm Optimization Algorithm* (PSO). Algoritma ini terinspirasi oleh sekawanan burung atau ikan. Kelebihan dari algoritma PSO yaitu kecepatan dalam memproses atau menyelesaikan suatu permasalahan optimasi lebih cepat. Fatimah (2016) telah menerapkan algoritma PSO untuk *Vehicle Routing Problem With Time Windows* (VRPTW) pada kasus pendistribusian barang. Disimpulkan bahwa, algoritma PSO efektif untuk masalah penentuan rute terpendek seperti VRPTW, TSP, dan lainnya. Berdasarkan uraian diatas, menjadi bahan pertimbangan bagi penulis untuk melakukan penelitian tentang penerapan *Cat Swarm Optimization Algorithm* (CSO) dan *Particle Swarm Optimization Algorithm* (PSO) dalam penyelesaian *Travelling Salesman Problem* (TSP) untuk mencari rute yang memiliki total jarak

tempuh yang paling minimum. Sehingga dapat mengetahui bagaimana penyelesaian TSP dengan menggunakan algoritma CSO dan algoritma PSO, selain itu juga dapat mengetahui perbandingan hasil dari kedua algoritma.

1.1 Rumusan Masalah

Berdasarkan uraian pada latar belakang di atas, maka rumusan masalah dari penelitian ini adalah:

a. bagaimana penyelesaian *Travelling Salesman Problem* (TSP) dengan algoritma *Cat Swarm Optimization* (CSO)?
b. bagaimana penyelesaian *Travelling Salesman Problem* (TSP) dengan algoritma *Particle Swarm Optimization* (PSO)?
c. bagaimana perbandingan hasil pada Penerapan algoritma *Cat Swarm Optimization* (CSO) dan algoritma *Particle Swarm Optimization* (PSO) dalam penyelesaian *Travelling Salesman Problem* (TSP)?

1.2 Tujuan Penelitian

Berdasarkan uraian pada latar belakang dan rumusan masalah, maka tujuan dari penelitian ini adalah :

a. mengetahui penyelesaian *Travelling Salesman Problem* (TSP) dengan algoritma *Cat Swarm Optimization* (CSO);
b. mengetahui penyelesaian *Travelling Salesman Problem* (TSP) dengan algoritma *Particle Swarm Optimization* (PSO);
c. mengetahui hasil terbaik dari perbandingan hasil pada penerapan algoritma *Cat Swarm Optimization* (CSO) dan algoritma *Particle Swarm Optimization* (PSO) dalam penyelesaian *Travelling Salesman Problem* (TSP).

1.3 Manfaat Penelitian

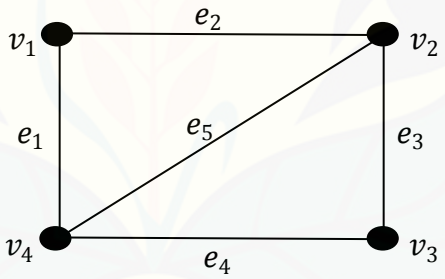
Berdasarkan uraian pada latar belakang, rumusan masalah dan tujuan penelitian maka manfaat dari penelitian ini adalah:

a. menambah wawasan dan referensi tentang algoritma *Cat Swarm Optimization* (CSO) dan algoritma *Particle Swarm Optimization* (PSO) dalam penyelesaian *Travelling Salesman Problem* (TSP) sehingga dapat mengembangkan algoritma lain yang lebih baik untuk menyelesaikan *Travelling Salesman Problem* (TSP);
b. secara khusus dapat memberikan gambaran tentang penyelesaian *Travelling Salesman Problem* (TSP) dan menjadi pertimbangan dalam penelitian selanjutnya mengenai *Travelling Salesman Problem* (TSP).

BAB 2. TI NJAUAN PUSTAKA

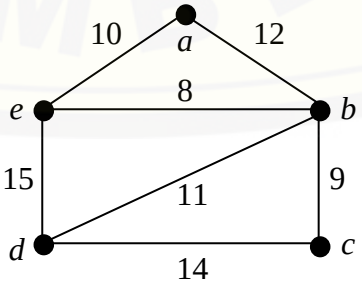
2.1 Teori Graf

Sebuah graf G adalah pasangan $(\mathrm{V}(G), E(G))$ dengan $V = \{v_1, v_2, v_3, \ldots, v_n\}$ adalah himpunan tak kosong yang anggotanya disebut titik (*vertex*), dan $E = \{e_1, e_2, e_3, \ldots, e_n\}$ adalah himpunan yang anggotanya adalah pasangan-pasangan tak berurut dan disebut dengan sisi (*edge*) yang himpunannya boleh kosong. Selanjutnya, graf G dapat dinyatakan dengan $G = (V, E)$ jika $V = \{v_1, v_2, v_3, \ldots, v_n\}$ dan *e* adalah *edge* yang menghubungkan *vertex* v_i dengan v_j maka *e* dapat dinyatakan dengan $e = v_1v_2$. Salah satu contoh graf G dapat dilihat pada Gambar 2.1



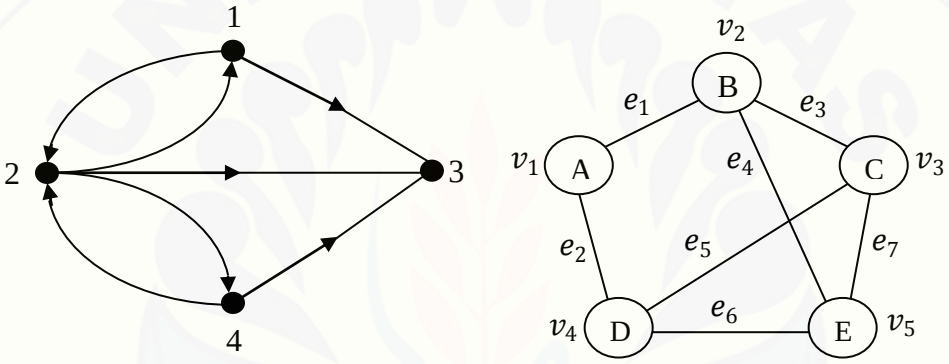
Gambar 2.1 Graf G dengan 4 titik dan 5 sisi

Graf berbobot adalah sebuah graf dengan bilangan riil *w* pada setiap sisinya. Jika sisi e_1 diberi label *w* (e_1), maka bobot dari sisi e_1 adalah *w* (e_1). Gambar 2.2 adalah contoh graf berbobot (Johnsonbaugh, 1998).



Gambar 2.2 Graf Berbobot

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah, sisi berarah biasanya disebut busur (*arc*). Pada graf berarah (*u*,*v*) dan (*v*, *u*) menyatakan dua buah busur yang berbeda, dengan kata lain (*u*,*v*) ≠ (*v*, *u*). Untuk busur (*u*,*v*), simpul *u* dinamakan simpul asal (*initial vertex*) dan simpul *v* dinamakan simpul terminal (*terminal vertex*). Sedangkan Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, (*u*, *v*) = (*v*, *u*) adalah sisi yang sama.

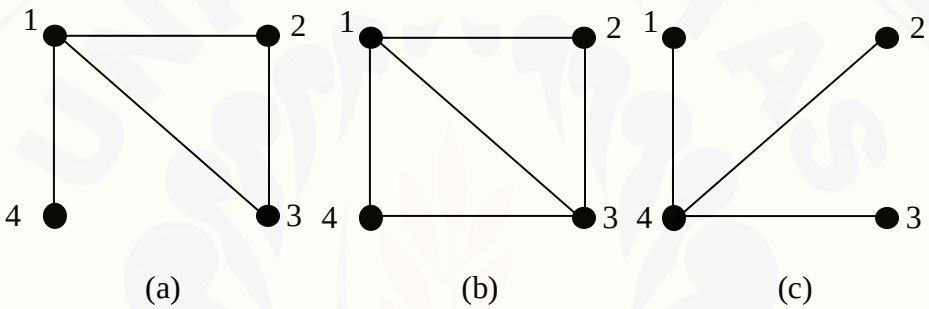


(a) (b)

Gambar 2.3 Graf (a) berarah dan Graf (b) tak berarah (Munir,2003).

Jalan (*walk*) pada graf G dinotasikan dengan $W(G)$ adalah suatu barisan berhingga yang diawali dan diakhiri oleh titik yang unsur-unsurnya bergantian antara titik dan sisi. Jalan $W(G)$ dengan $v_1 = v_n$ disebut jalan tertutup, sedangakan jika $v_1 \neq v_n$ disebut jalan terbuka. Suatu jalan dengan titik yang tidak diulang disebut lintasan (*path*), sedangkan suatu jalan dengan sisi yang tidak diulang disebut jejak (*trail*). Sebuah jalan tertutup tanpa pengulangan titik kecuali titik awal dan titik akhir disebut sikel (*cycle*).

Lintasan hamilton adalah lintasan yang melalui tiap simpul di dalam graf tepat satu kali. lintasan tersebut kembali ke simpul asal membentuk lintasan tertutup (sirkuit), sehingga lintasan tertutup disebut sirkuit hamilton. Sirkuit hamilton merupakan sirkuit yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali. Graf yang memiliki sirkuit hamilton disebut graf-hamilton, sedangkan graf yang hanya memiliki lintasan hamilton disebut graf semi-hamilton. Gambar 2.4 merupakan contoh graf yang mengandung lintasan yaitu:



Gambar 2.4 (a) graf yang memiliki lintasan hamilton (3, 2, 1, 4), (b) Graf yang memiliki sirkuit hamilton (1, 2, 3, 4, 1), (c) Graf yang tidak memiliki lintasan maupun sirkuit hamilton.

(Munir,2003).

2.2 *Travelling Salesman Problem* (TSP)

Permasalahan TSP ditemukan sejak tahun 1800 oleh matematikawan Irlandia Sir William Rowan Hamilton dan matematikawan Inggris Thomas Penyngton Kirkman, bentuk umum dari TSP pertama kali dipelajari oleh para matematikawan mulai tahun 1930-an oleh Karl Menger di Vienna dan Harvard. TSP dikenal sebagai suatu permasalahan optimasi yang bersifat klasik dan *Non-Deterministic Polynomial-time* (NPC), dimana tidak ada penyelesaian yang paling optimal selain mencoba seluruh kemungkinan penyelesaian. Tujuannya yaitu menemukan secara pasti nilai minimum dari perjalanan seorang salesman. *Travelling Salesman Problem* (TSP)

merupakan permasalahan seorang *salesman* yang harus menentukan rute terpendek dalam sebuah perjalanannya yang melalui sejumlah kota (Andri, 2013).

Salesman harus mengunjungi beberapa kota dalam setiap kali kunjungan, yang harus berangkat dari suatu kota kemudian mengunjungi kota lain masing-masing satu kali kunjungan dan kembali ke kota semula dengan biaya perjalanan minimum. Salesman harus memilih rute agar total jarak antar kota (satuan kilometer) yang ditempuh menjadi minimum, sehingga total waktu sekaligus ongkos perjalanan juga dapat di minimumkan atau di tekan.

Terdapat dua jenis TSP, yaitu simetris dan asimetris. Perbedaannya terletak pada biaya dari sebuah perjalanan seorang salesman. TSP simetris, biaya dari kota 1 ke kota 2 sama dengan biaya dari kota 2 ke kota 1. Sedangkan pada TSP asimetris, biaya dari kota 1 ke kota 2 tidak sama dengan biaya dari kota 2 ke kota 1. Untuk TSP asimetris, jumlah jalur yang mungkin adalah permutasi dari jumlah kota yang harus di lalui dibagi dengan jumlah kota yang dilalui. Hal ini dapat dipahami karena secara siklus, sebuah jalur dengan urutan 1-2-3 adalah sama dengan jalur 2-3-1 dan jalur 3-1-2. Tetapi jalur dengan urutan 1-2-3 tidaklah sama dengan jalur 3-2-1. Untuk memilih jarak total minimum diantara semua kemungkinan menggunakan persamaan (2.1).

$$Z = \Sigma_{(i,j)\epsilon A} C_{ij}\ x_{ij} \qquad (2.1)$$

Dengan batas : $\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}$ = 1, j = 0, 1, 2, ..., n

$\sum_{\substack{j=0 \\ j \neq i}}^{n} x_{ij}$ = 1, i = 0, 1, 2, ..., n

dimana : n = jumlah kota

C_{ij} = jarak *travelling* dari kota *i* ke kota *j*

$x_{ij} = \begin{cases} 1 \text{ jika sales melakukan perjalanan dari kota } i \text{ ke kota } j \\ 0 \text{ jika sales melakukan perjalanan dari kota } j \text{ ke kota } i \end{cases}$

(Suyanto, 2005).

2.3 *Cat Swarm Optimization* (CSO)

Cat Swarm Optimization (CSO) merupakan salah satu algoritma metaheuristik untuk memecahkan masalah optimasi. Algoritma CSO pertama kali diusulkan oleh Shu-Chuan Chu dan Pei-Wei Tsai (Taiwan) pada tahun 2006. Ketertarikan Shu-Chuan Chu dan Pei-Wei Tsai berawal pada pengamatan terhadap perilaku sekumpulan *Cat* (kucing), namun *Cat* yang dimaksud bukanlah hanya seekor kucing saja. *Cat* termasuk kelas binatang dari *felid* yang memiliki sekitar 30 spesies, *Cat* (kucing) merupakan salah satu dari spesies tersebut. Sehingga Shu-Chuan Chu dan Pei-Wei Tsai menggunakan istilah "kucing" untuk mewakili semua spesies *Cat* dan digambarkan pada algoritma CSO.

Kucing memiliki pola perilaku yang sama, yaitu keterampilan dalam hal berburu dan rasa ingin tahunya yang besar terhadap mangsa (objek buruan). Meskipun memiliki rasa ingin tahu yang besar, kucing menghabiskan waktu secara tidak aktif misalnya hanya bermalas-malasan, beristirahat, bahkan ketika kucing terjaga tampak seolah tidak aktif (diam). Namun kucing memiliki tingkat kewaspadaan yang tinggi, terutama saat dalam kondisi istirahat dan terjaga. Kucing memiliki mata yang besar, dan akan lebih besar dari biasanya saat melakukan pengamatan jika ada mangsa buruan. Kucing mengamati dengan seksama dan mencari apakah ada mangsa yang dapat dikejar. Setelah menemukan mangsa atau objek buruan, kucing akan melakukan sergapan terhadap mangsa. Pada saat melakukan pengamatan, kucing sedang menyusun langkah atau strategi apa yang akan dilakukan secara efektif, singkat, dan tepat agar mangsa buruannya tidak lepas dari cengkramannya.

Dalam algoritma CSO kucing dan model perilakunya digunakan untuk mneyelesaikan masalah optimasi, artinya kucing digambarkan sebagai set solusi. Berdasarkan perilaku kucing tersebut algoritma CSO terbagi menjadi 2 langkah dalam menyelesaikan masalah optimasi, yaitu *Seeking Mode* (SM) yang menggambarkan kucing saat istirahat, melihat sekeliling, menyusun strategi

selanjutnya dan *Tracking Mode* (TM) yang menggambarkan kucing saat mengikuti mangsa buruan.

2.3.1 *Seeking Mode*

Seeking Mode merupakan gambaran kucing saat istirahat namun waspada melihat sekeliling dan menyusun strategi atau mencari posisi berikutnya untuk bergerak. *Seeking Mode* dibagi menjadi empat faktor yaitu *seeking memory pool* (SMP), *seeking range of the select dimension* (SRD) atau mencari dimensi terpilih, *counts of dimension to cange* (CDC) atau menghitung dimensi yang akan berubah, dan *self position considering* (SPC) atau mempertimbangkan posisi.

SMP digunakan untuk mendefinisikan ukuran memori pencarian masing-masing kucing, mengindikasikan titik-titik yang telah dicoba oleh kucing. Kucing tersebut kemudian akan memilih titik dari kelompok memori. SRD menyatakan rentang perpindahan dalam dimensi terpilih. Dalam *seeking mode* jika suatu dimensi diputuskan berpindah, selisih antara nilai baru dengan yang lama tidak boleh melebihi suatu rentang yaitu SRD. CDC memperlihatkan beberapa dimensi yang akan berubah, SPC merupakan variabel *Boolean* (bernilai benar atau salah). Untuk memutuskan suatu titik yang pernah menjadi posisi kucing akan menjadi kandidat posisi untuk bergerak. Bagaimanapun nilai SPC, benar atau salah nilai SMP tidak akan terpengaruh. Langkah-langkah *seeking mode* yaitu sebagai berikut:

a. bangkitkan *j* tiruan dari posisi saat ini, dimana *j* sama dengan SMP. Jika nilai SPC benar maka j = (SMP-1). Kemudian pertahankan posisi saat ini sebagai salah satu kandidat. Menghitung nilai *fitness* atau nilai fungsi tujuan sebagai solusi awal yang telah dibangkitkan dengan menggunakan persamaan 2.2

$$Fitness = \frac{1}{z_i} \tag{2.2}$$

b. untuk setiap tiruan disesuaikan dengan CDC, tambahkan atau kurangkan SRD persen dari nilai saat ini secara acak dan gantikan nilai yang sebelumnya. Untuk

menghitung jumlah dimensi yang akan dimodifikasi yaitu dengan menggunakan persamaan 2.3

$$\text{Jumlah modifikasi} = \text{CDC} * n \quad (2.3)$$

selanjutnya menghitung nilai modifikasi dari setiap dimensi yang terpilih dengan menggunakan persamaan 2.4

$$\text{Modifikasi}_{i,d} = x_{i,d} + (-1)^m * \text{SRD} * x_{i,d} \quad (2.4)$$

c. hitung nilai kecocokan (FS) untuk semua titik kandidat;

d. jika semua FS tidak benar-benar sama, hitung probabilitas terpilih masing-masing titik kandidat dengan menggunakan persamaan 2.5

$$p_i = \frac{|FS_i - FS_{min}|}{FS_{max} - FS_{min}} \quad (2.5)$$

dan begitu juga sebaliknya atur probabilitas terpilih untuk semua titik sama dengan 1;

e. secara acak pilih titik untuk bergerak dari titik-titik kandidat, dan pindahkan posisi $kucing_k$.

2.3.2 *Tracking Mode*

Tracking mode merupakan model yang menggambarkan keadaan ketika kucing sedang mengikuti jejak targetnya. Sekali kucing memasuki *tracking mode*, kucing tersebut akan bergerak sesuai dengan kecepatannya untuk setiap dimensi. Tahapan *tracking mode* dapat dijelaskan dalam 3 langkah berikut yaitu:

a. perbarui nilai kecepatan untuk setiap dimensi $V_{k,d}$ berdasarkan persamaan 2.6

$$V'_{k,d} = V_{k,d} + r_1 \times c_1 \, (X_{best,d} - X_{k,d}) \quad (2.6)$$

dimana: $d = 1, 2, \ldots, n$ dan $k = 1, 2, \ldots, m$

$X_{best,d}$ 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];

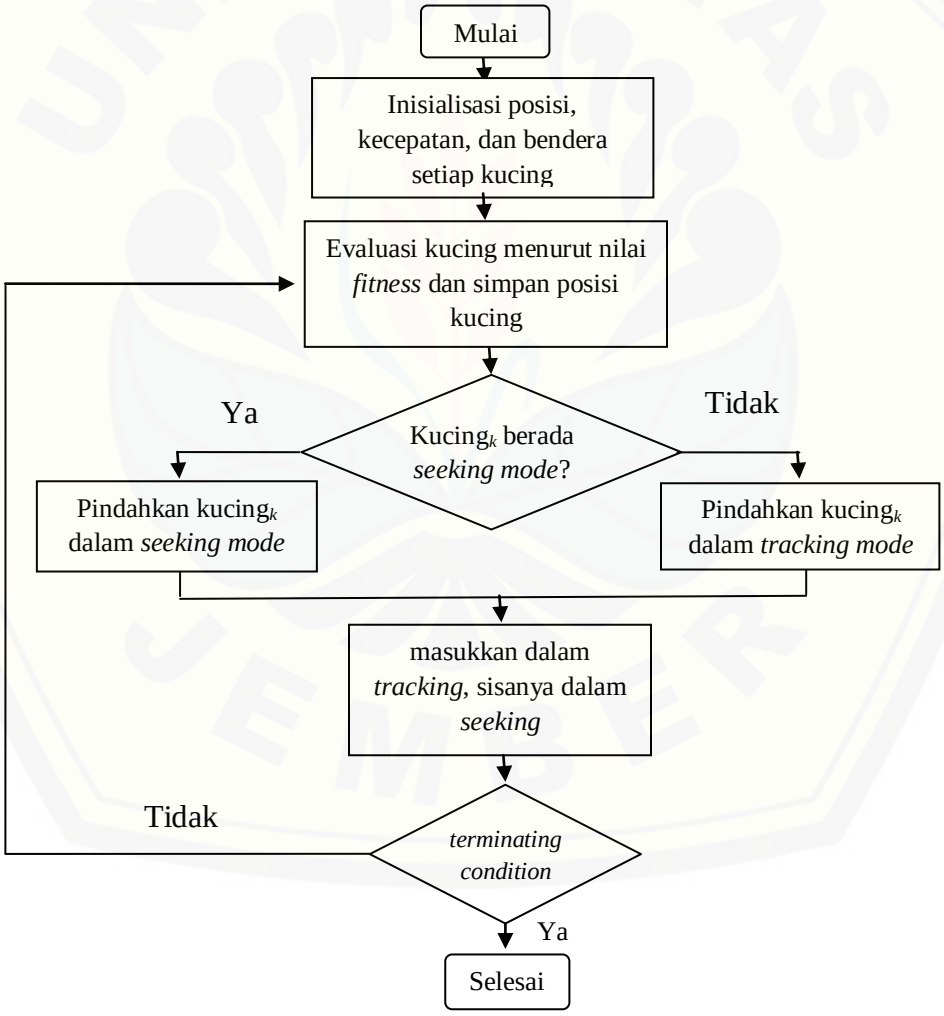
b. periksa apakah kecepatan berada dalam rentang kecepatan maksimum. Jika kecepatan yang baru melebihi rentang, maka tetapka nilai sama dengan batas;

c. perbarui posisi $kucing_k$ berdasarkan persamaan 2.7

$$X'_{k,d} = X_{k,d} + V_{k,d} \quad (2.7)$$

dengan mengamati perilaku kucing, dapat diketahui bahwa kucing menghabiskan sebagian besar waktunya untuk beristirahat. Selama beristirahat, kucing mengubah posisinya perlahan dan berhati-hati, terkadang bahkan tetap pada tempatnya. Untuk menerapkan perilaku ini CSO maka digunakan *seeking mode*. Sedangkan perilaku mengejar target diaplikasikan dalam *tracking mode*. Oleh karena itu *micture ratio*/rasio campuran harus bernilai kecil untuk memastikan bahwa kucing menghabiskan sebagian besar waktu dalam *seeking mode*.

Langkah-langkah Algoritma *Cat Swarm Optimation* (CSO) dapat digambarkan sebagai berikut:



Gambar 2.7 *Flowchart* Algoritma CSO

Penjelasan *flowchart* algoritma CSO pada Gambar 2.7 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 *tracking mode* sesuai MR, sisanya dimasukkan dalam *seeking mode*. Untuk menghitung jumlah *tracking* yaitu dengan menggunakan persamaan 2.8

   Jumlah *tracking* = MR * *m* (2.8)

   dan untuk menghitung jumlah *seeking* menggunakan persamaan 2.9

   Jumlah *Seeking* = *m* – jumlah *tracking* (2.9)

c. hitung nilai kecocokn masing-masing kucing dengan memasukkan nilai posisi kucing ke dalam fungsi kecocokan yang menunjukkan kriteria tujuan dan simpan kucing terbaik dalam memori. Yang perlu disimpan adalah posisi kucing terbaik (X_{best}) karena kucing terbaik mewakili solusi terbaik;
d. pindahkan kucing sesuai benderanya, jika $kucing_k$ berada dalam *seeking mode* maka ambil langkah-langkah sesuai *seeking mode* dan begitu juga sebaliknya ambil langkah-langkah sesuai *tracking mode*;
e. pilih lagi sejumlah kucing dan masukkan dalam *tracking mode* sesuai MR, sisanya masukkan ke dalam *seeking mode*;
f. perhatikan *terminating condition*-nya. Jika telah optimal hentikan program. Sebaliknya ulangi langkah 3 hingga 5.

2.4 Seleksi *Roulette Wheel*

Seleksi *roulette wheel* menirukan permainan roda *roulette wheel* dimana masing – masing *chromosome* menempati potongan lingkaran pada roda *roulette wheel* secara proposional sesuai dengan nilai *fitness*nya. Langkah – langkah seleksi *roulette wheel* yaitu:

1. menghitung nilai *fitness* dari masing – masing kucing;
2. menghitung nilai *fitness* dari semua kucing;
3. menghitung probabilitas, hitung masing – masing kucing antara [0,1];
4. bangkitkan bilangan acak [0,1];
5. dari bilangan acak yang dihasilkan, tentukan kucing mana yang terpilih dalam seleksi.

2.5 *Particle Swarm Optimization* PSO)

Particle Swarm Optimization (PSO) merupakan salah satu algoritma metaheuristik yang didasarkan pada perilaku sekawanan burung atau ikan. Algoritma PSO pertama kali dikembangkan oleh James Kennedy dan Russel Eberhart pada tahun 1995 yang meniru perilaku sosial sekawanan burung atau ikan. Dalam algoritma PSO burung disebut sebagai partikel. James Kennedy dan Russel Eberhart terinspirasi oleh sekawanan burung atau ikan karena partikel ini berperilaku dengan cara menggunakan kecerdasan (*intelligence*) dan dipengaruhi perilaku kelompok. "partikel" dimaksudkan sebagai seekor burung dalam kawanan burung atau seekor ikan dalam kawanan ikan. James dan Russel menggunakan istilah partikel karena pada dasarnya banyak makhluk hidup yang memiliki kecerdasan seperti burung dan ikan, misalnya belalang, lebah, dan lain sebagainya. *Particle* yang berarti individu sedangkan *Swarm* yang berarti populasi (Santoso, 2011).

Particle Swarm Optimization (PSO) mensimulasikan perilaku dari sekolompok burung atau ikan. Misalnya, sekelompok burung sedang terbang secara acak mencari makanan di sebuah area tetapi hanya ada satu potong makanan pada area tersebut. Semua burung tidak mengetahui dimana lokasi makanan yang sebenarnya, hanya saja mereka dapat mengetahui seberapa jauh letak makanan yang ada pada setiap pencarian. Strategi terbaik yang dapat digunakan untuk mencari makanan adalah dengan cara mengikuti burung yang lokasinya paling dekat dengan makanan tersebut. Satu burung menyatakan satu solusi di dalam ruang masalah dan burung tersebut direpresentasikan sebagai "partikel" dalam algoritma PSO. Algoritma

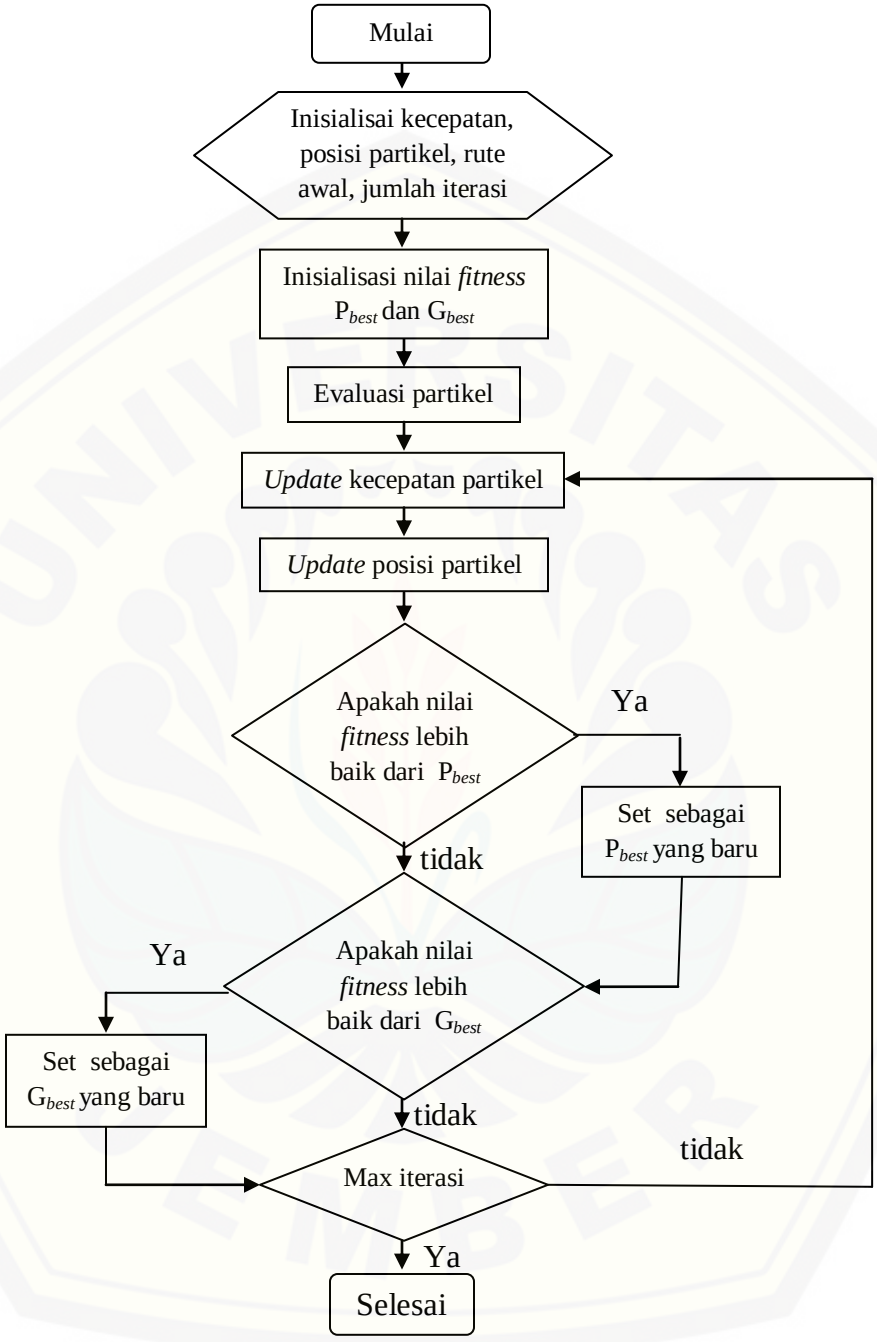
PSO menggunakan populasi dari sekumpulan partikel, dimana setiap partikel mewakili sebuah solusi yang mungkin untuk sebuah permasalahan optimasi. Semua partikel memiliki nilai *fitness* yang dievaluasi oleh fungsi *fitness* yang akan dioptimalkan, dan memiliki kecepatan (*velocity*) setiap partikel berpindah dengan kecepatan yang diadaptasi dari daerah pencarian dan disimpan sebagai posisi terbaik yang pernah dicapai. Sehingga dari skenario kumpulan burung tersebut yang kemudian menggunakannya untuk memecahkan masalah optimasi (Erny, 2013).

Sehingga *Particle Swarm Optimization* (PSO) dikembangkan dengan berdasarkan pada model:

a. ketika seekor burung mendekati target atau makan (bisa minimum atau maksimum suatu fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawanan tertentu;
b. burung yang lain akan mengikuti arah menuju makanan tetapi tidak secara langsung;
c. ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewati pada waktu sebelumnya.

Dalam algoritma PSO kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak disuatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki karakteristik yaitu posisi dan kecepatan. Setiap partikel bergerak dalam ruang tertentu dengan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel lain dan menyesuaikan posisi kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut.

Langkah-langkah Algoritma *Particle Swarm Optimation* (PSO) dapat digambarkan sebagai berikut:



Gambar 2.8 *Flowchart* Algoritma PSO

Penjelasan *flowchart* algoritma PSO pada gambar 2.8 sebagai berikut:

a. melakukan inisialisasi jumlah partikel, rute awal, posisi awal C_1 (*learning rates eksplorasi* lokal), C_2 (*learning rates* eksplorasi global), jumlah iterasi, kecepatan awal partikel v_i, inersia;
b. evaluasi jarak total dari masing-masing rute yang dihasilkan dari setiap partikel;
c. menentukan posisi terbaik partikel dengan jarak total terkecil dan menetapkan partikel ini sebagai G_{best} untuk setiap partikel menggunakan nilai awalnya sebagai P_{best};
d. mengulangi langkah berikut sampai stopping kriteria terpenuhi:
   1) menggunakan P_{best} dan G_{best} yang ada, memperbarui kecepatan setiap partikel menggunakan persamaan 2.10

   $$v_i'(i) = \theta v_j(i-1) + c_1 r_1[p_{best} - c_1] + c_2 r_2[g_{best} - c_2] \quad (2.10)$$

   Dimana p_{best} merupakan nilai partikel terbaik p_{best} merupakan posisi terbaik dari setiap partikel, c_1 dan c_2 adalah konstanta yang bernilai positif, r_1 dan r_2 bilangan random yang bernilai 0 sampai 1.

   θ merupakan nilai bobot inersia dengan persamaan 2.11

   $$\theta(i) = \theta_{max} - \left(\frac{\theta_{max} - \theta_{min}}{i_{max}}\right) i \quad (2.11)$$

   kemudian dengan kecepatan baru yang didapat, update posisi dari setiap partikel menggunakan persamaan 2.12

   $$c_i' = c_i + v_i' \quad (2.12)$$

   2) evaluasi kembali jarak total dari masing-masing rute, jika kurang dari p_{best} maka menggunakan persamaan 2.13

   $$g_{best} = \begin{cases} p_{ibest}, f(p_{ibest}) < f(g_{best}) \\ g_{best}, f(p_{ibest}) \geq f(g_{best}) \end{cases} \quad (2.13)$$

   3) evaluasi jarak dari setiap rute yang dihasilkan oleh setiap partikel;
   4) menentukan partikel dengan jarak total minimum dan menetapkan partikel yang bersangkutan sebagai G_{best}. Untuk setiap partikel menentukan P_{best} dengan

membandingkan partikel sekarang dengan P_{best} dari iterasi sebelumnya berdasarkan jarak total minimumnya;

5) cek *stopping criteria*, jika dipenuhi berhenti jika tidak kembali ke langkah awal. (Santosa, 2011).

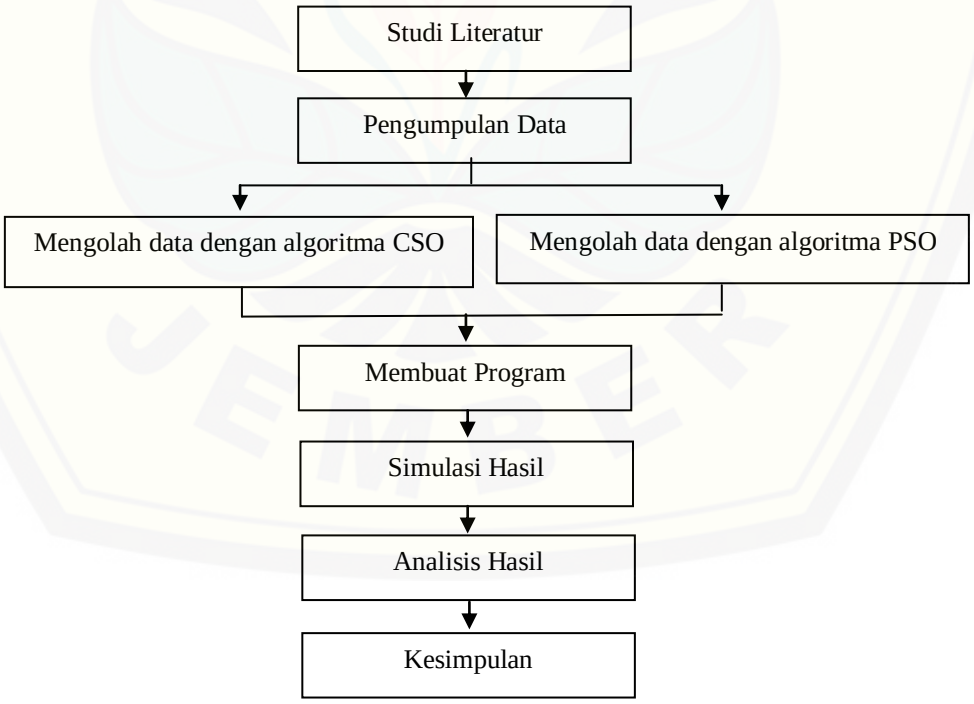
BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam penelitian ini adalah data sekunder yang diperoleh dari *library Travelling Salesman Problem* (TSP) *asymmetric*. Data dalam penelitian ini berupa sejumlah kota (*n*) dan jarak antar kota-kota yang akan dikunjungi oleh sales. Jumlah kota yang akan digunakan sebanyak 50 kota dengan jarak antar kota yang dapat dilihat pada lampiran.

3.2 Langkah-langkah Penyelesaian

Langkah – langkah penelitian yang akan dilakukan untuk menyelesaikan *Travelling Salesman Problem* (TSP) dapat dilihat pada Gambar 3.1



Gambar 3.2 Skema Metode Penelitian

Berikut penjelasan dari skema langkah – langkah penelitian pada Gambar 3.1 diatas yaitu sebagai berikut:

a. Studi Literatur

Langkah awal yang dilakukan pada metode penelitian ini yaitu mencari dan mengumpulkan studi literatur. Studi literatur digunakan untuk mendapatkan informasi dari buku, jurnal, dan skripsi yang terkait dengan algoritma (CSO), algoritma (PSO), dan *Travelling Salesman Problem* (TSP).

b. Pengumpulan Data

Langkah selanjutnya yaitu mengumpulkan data, diambil untuk studi literatur menjadikan objek menyelesaikan masalah, pada penelitian ini dengan menggunakan algoritma (CSO) dan algoritma (PSO). Data yang digunakan yaitu data jarak antar kota yang diperoleh dari *library* TSP *asymmetric.*

c. Menerapkan Algoritma *Cat Swarm Optimation* (CSO) Pada Permasalahan *Travelling Salesman Problem* (TSP)

Langkah keempat yaitu menerapkan Algoritma *Cat Swarm Optimation* (CSO) pada permasalahan *Travelling Salesman Problem* (TSP), adapun langkahnya sebagai berikut:

1) Input dan inisialisasi parameter

Langkah awal penerapan algoritma (CSO) untuk input data dan menentukan parameter-parameter yang dibutuhkan dalam algoritma (CSO) yaitu jumlah populasi kucing (*m*), jumlah titik yang dikunjungi (*n*), SMP, CDC, SRD, MR, *c*.

2) Pembangkitan populasi awal

Setiap individu 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.

3) Pembangkitan kecepatan awal

Setiap dimensi dari setiap individu juga memiliki inisialisasisi lain yakni kecepatan awal (*velocity*). Kecepatan awal adalah suatu bilangan acak yang

terkait dimensi dari setip individu. Pembangkitan kecepatan awal dilakukan dengan membuat matriks brukuran *m* x *n*.

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).

5) Menghitung fungsi tujuan

   Permasalahan TSP dibuat dengan tujuan untuk meminimumkan total jarak tempuh yaitu dengan menggunakan pengurutan biasa dengan bilangan acak terkecil disusun.

6) Menghitung nilai *fitness*

   Nilai fitness yang didapat oleh individu tergantung pada jarak tempuh yang didapat dari representasi permutasi. Cara menghitung nilai fitness kucing ke-*i* yaitu satu dibagi jarak yang diperoleh setiap set solusi kucing ke-*i*. Selanjutnya nilai fitness diurutkan dari yang terbesar hingga terkecil.

7) Menghitung *Self Position Considering* (SPC)

   Penempatan *Self Position Considering* (SPC) dilakukan dengan cara penempatan label SPC=1 terhadap kucing yang mempunyai nilai fitness tertinggi dan sisanya mempunyai label SPC=0 secara otomatis.

8) Menentukan Flag

   Jumlah kucing yang dimasukkan ke dalam masing-masing solusi dengan nilai MR.

9) Melakukan Proses *Tracing Mode*

   Setiap kucing akan mengubah representasi kecepatan yang lama menjadi representasi kecepatan yang baru. Proses update kecepatan dipengaruhi oleh memoryyang dipengaruhi juga oleh setiap kucing.

10) Melakukan Proses *Seeking Mode*

Modifikasi *memory pool* yang diperuntukkan untuk melakukan modifikasi terhadap set solusi dari tiruan individu. Hasil dari modifikasi akan diproses dalam *memory pool*. Modifikasi yang dilakukan tergantung parameter CDC, SRD, dan SMP.

11) Simulasi Perubahan Parameter

Melakukan simulasi perubahan parameter pada algoritma CSO, karena proses penyelesaiannya dilakukan secara random yang dibangkitkan oleh bilangan acak. Parameter yang dilakukan simulasi perubahan parameter yaitu MR, *c*, CDC, dan SRD.

d. Menerapkan Algoritma *Particle Swarm Optimation* (PSO) Pada Permasalahan *Travelling Salesman Problem* (TSP)

Langkah keempat yaitu menerapkan algoritma PSO pada permasalahan TSP, adapun langkahnya sebagai berikut:

1) melakukan inisialisasi jumlah partikel, rute awal, *C1* (*learning rates eksplorasi* lokal), *C2* (*learning rates* eksplorasi global), jumlah iterasi, kecepatan awal, inersia;
2) evaluasi jarak total dari masing-masing rute yang dihasilkan dari setiap partikel berdasarkan matrik jarak;
3) menentukan partikel dengan jarak total terkecil dan menetapkan partikel ini sebagai G_{best} untuk setiap partikel menggunakan nilai awalnya sebagai P_{best};
4) mengulangi langkah berikut sampai *stopping criteria* terpenuhi:
   a) menggunakan P_{best} dan G_{best} yang ada, memperbarui kecepatan setiap partikel menggunakan persamaan (2.10). Menghitung bobot inersia dengan persamaan (2.11) Kemudian dengan kecepatan baru yang didapat, diperbarui nilai dari setiap partikel menggunakan persamaan (2.12);

b) evaluasi kembali jarak total dari masing-masing rute, jika kurang dari p_{best} maka menggunakan persamaan (2.13)
c) evaluasi jarak dari setiap rute yang dihasilkan oleh setiap partikel;
d) menentukan partikel dengan jarak total minimum dan menetapkan partikel yang bersangkutan sebagai G_{best}. Untuk setiap partikel menentukan P_{best} dengan membandingkan partikel sekarang dengan P_{best} dari iterasi sebelumnya berdasarkan jarak total minimumnya;
e) cek *stopping criteria*, jika dipenuhi berhenti jika tidak kembali ke langkah awal.
f) Melakukan simulasi perubahan parameter pada algoritma PSO, karena proses penyelesaiannya dilakukan secara random yang dibangkitkan oleh bilangan acak. Parameter yang dilakukan simulasi perubahan parameter yaitu $c1$ dan $c2$. Untuk $\theta_{\max}$ dan $\theta_{\min}$ tidak dilakukan simulasi perubahan parameter karena nilai yang digunakan telah ditetapkan yaitu $\theta_{\max}$ = 0,9 sedangkan $\theta_{\min}$ = 0,4.

e. Membuat Program

Peneliti akan membuat *script* program algoritma CSO dan algoritma PSO menggunakan program MATLAB untuk menyelesaikan TSP.

f. Simulasi Hasil

Menjalankan program yang telah dibuat pada point e untuk menyelesaikan data yang telah dikumpulkan pada point b.

g. Analisis Hasil

Membandingkan hasil dari algoritma CSO dengan hasil dari algoritma PSO. Hasil yang dibandingkan berupa total jarak tempuh.

h. Kesimpulan

Mengambil kesimpulan akhir perbandingan hasil dari algoritma CSO dan algoritma PSO.

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada penelitian ini, maka dapat diambil kesimpulan sebagai berikut:

a. Penerapan Algoritma *Cat Swarm Optimization* (CSO) untuk *Travelling Salesman Problem* (TSP) untuk data 50 kota berdasarkan percobaan yang telah dilakukan menghasilkan rute sales dengan total jarak tempuh 1808 km.
b. Penerapan Algoritma *Particle Swarm Optimization* (PSO) untuk *Travelling Salesman Problem* (TSP) untuk data 50 kota berdasarkan percobaan yang telah dilakukan menghasilkan rute sales dengan total jarak tempuh 1808 km.
c. Berdasarkan hasil yang telah diperoleh, algoritm*a Cat Swarm Optimization* (CSO) lebih baik (efektif) dibandingkan algoritm*a Particle Swarm Optimization* (PSO), karena meskipun total jarak tempuh yang dihasilkan sama jika dilihat dari hasil *running* program algoritma CSO selalu menghasilkan total jarak tempuh yang lebih minimum dibandingkan algoritma PSO.

5.2 Saran

Penelitian selanjutnya dapat menerapkan algoritma *Cat Swarm Optimization* (CSO) pada permasalahan yang lain, misalnya Multi *Travelling Salesman Problem* (TSP), selain itu dapat membandingkan hasil dengan algoritma yang lain.

DAFTAR PUSTAKA

Andri, Suyandi, dan Winwin. 2013. *Aplikasi Travelling Salesman Problem Dengan Metode Artificial Bee Colony*. Medan: STMIK Mikroskil.

Apriliani, I. 2011. *Penyelesaian Travelling Salesman Problem* (TSP) *Menggunakan Algoritma Semut Dan Algoritma Cheapest Insertion Heuristic* (CIH). Tidak dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Atiyatna, V. 2012. *Penyelesaian Travellig Salesman Problem (TSP) Asimetris Dengan Algoritma Genetik Commonality*. Tidak dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Baihaki, A. N. 2016. *Penerapan Algoritma Cat Swarm Optimization* (*CSO*) *Pada penerapan Penyelesaian Sistem Persamaan Non-Linier*. Tidak Dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Chartrand, G dan Oellermann, O.R. 1993. *Applied and Algoritmic Graph Theory*. New York : Mc Graw-Hill, Inc.

Dhanasaputra, N dan Santosa, B. 2010. *Pengembangan Algoritma Cat Swarm Optimization (CSO) Untuk Klasifikasi*. Surabaya : ITS.

Erny, 2013. *Optimasi Pola Penyusunan Barang Dalam Peti Kemas Menggunakan Algoritma Particle Swarm Optimization*. Makasar : UNHAS.

Fatimah, M. N. 2016. *Penerapan Algoritma Particle Swarm Optimization Untuk Vehicle Routing Problem With Time Windows Pada Kasus Pendistribusian Barang*. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Johnsonbaugh, R. 1998. *Matematika Diskrit Edisi 4 Jilid 1*. Jakarta: PT. Prenhalliondo.

Lestari, T.P. 2016. *Penyelesaian Travelling Salesman Problem* (TSP) *Dengan Algoritma Hybrid Cat Swarm Optimization* (hCSO). Tidak dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Munir, R. 2010. *Matematika Diskrit*. Bandung: Informatika.

Rachman, R. A., Syarif, D., dan Sari, R. P. 2012. *Analisa Penerapan Metode Particle Swarm Optimization Pada Optimasi Penjadwalan Kuliah*. Pekanbaru: Politeknik Caltex Riau.

Santoso, B. 2011. *Particle Swarm Optimization*. Surabaya : ITS.

Setiyawan, TE. 2005. *Perbandingan Algoritma Semut dan Algoritma Greedy dalam Menyelesaikan Masalah Perjalanan Salesman*. Tidak Dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Suyanto. 2005. *Algoritma Genetika Dalam Matlab*. Yogyakarta: Andi.

Lampiran A. Data 50 Kota

KOTA	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18
K1	0	80	65	19	72	81	44	89	86	81	60	50	23	19	86	93	50	45
K2	32	0	87	81	76	75	58	32	44	73	99	53	68	63	26	96	56	52
K3	28	89	0	20	80	90	69	28	22	13	48	62	99	18	44	91	59	35
K4	32	54	80	0	79	19	99	86	42	51	53	32	55	62	78	83	91	80
K5	70	95	41	51	0	65	67	77	79	30	72	11	84	48	54	45	54	97
K6	50	27	58	14	97	0	43	15	60	58	98	75	98	71	53	57	42	46
K7	23	13	49	92	90	27	0	69	96	23	44	24	51	19	76	92	98	83
K8	55	17	92	58	54	36	14	0	54	74	52	16	57	60	34	28	51	79
K9	85	37	87	64	66	79	25	68	0	75	35	46	77	31	73	41	25	49
K10	76	43	92	39	14	24	14	62	81	0	70	13	99	89	73	82	58	89
K11	87	97	51	94	35	32	71	23	100	15	0	15	40	48	55	40	22	29
K12	41	47	20	17	87	18	99	22	38	39	34	0	79	76	87	22	53	53
K13	80	73	97	50	69	45	45	67	34	38	70	19	0	92	71	28	42	76
K14	19	92	54	30	21	54	43	29	48	74	11	69	42	0	75	20	88	58
K15	68	50	19	44	65	82	22	50	62	43	61	57	53	54	0	19	75	70
K16	65	20	53	41	22	65	88	35	94	68	33	52	56	49	90	0	29	66
K17	24	50	65	13	69	70	81	69	66	97	97	70	99	74	95	29	0	35

KOTA	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18
K18	42	65	36	29	85	82	69	68	48	74	70	42	86	44	31	37	68	0
K19	78	44	57	38	20	77	22	51	49	95	72	31	83	35	89	26	76	58
K20	38	42	14	18	14	47	84	59	60	62	70	73	37	58	88	78	100	66
K21	25	73	26	31	50	54	44	38	53	35	62	93	91	44	77	11	33	54
K22	63	92	84	28	35	24	56	82	42	65	39	89	54	97	12	60	47	30
K23	15	52	61	20	46	57	23	98	100	93	72	43	75	85	62	84	21	52
K24	99	25	91	26	92	71	63	28	49	69	36	88	21	38	96	28	28	26
K25	89	33	44	64	38	75	29	51	29	48	70	98	60	97	38	24	24	98
K26	67	48	80	96	96	55	77	14	51	86	24	65	26	21	26	27	21	89
K27	32	38	67	13	52	53	30	70	19	21	25	35	44	34	63	90	12	70

KOTA	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18
K28	46	80	42	79	18	61	95	100	86	14	57	73	42	55	43	12	77	15
K29	86	66	30	86	87	63	11	78	49	40	22	41	64	81	62	71	28	21
K30	67	32	17	11	37	71	59	51	91	47	29	22	24	87	34	19	46	82
K31	25	87	98	49	87	29	88	35	65	58	55	18	86	32	76	26	68	67
K32	38	95	32	100	63	25	81	14	28	91	60	21	56	98	78	50	25	14
K33	94	44	40	49	28	89	78	27	79	54	100	60	92	12	59	62	19	75
K34	22	62	72	46	43	53	97	80	16	98	91	20	99	62	24	36	19	81
K35	13	34	56	48	15	75	43	76	44	74	63	87	73	79	52	31	69	86
K36	57	45	34	39	95	78	72	89	14	99	61	38	75	67	27	39	14	99
K37	54	31	70	77	40	13	54	82	44	67	99	94	95	12	47	69	38	79
K38	42	68	53	62	92	35	38	74	16	16	57	95	32	11	54	92	43	30
K39	63	80	55	57	79	18	42	100	89	67	92	47	63	92	25	27	83	63
K40	27	66	13	69	54	48	34	44	72	22	89	25	52	20	44	84	61	91
K41	35	20	39	74	14	24	34	90	19	70	76	76	44	45	74	53	33	58
K42	37	86	63	43	18	81	24	92	76	15	68	43	54	32	47	17	95	69
K43	60	66	75	31	80	15	49	44	60	66	11	19	27	93	33	56	96	14
K44	23	67	80	82	35	100	34	43	32	53	94	75	79	100	77	74	70	23
K45	44	46	94	40	59	18	57	90	49	25	29	93	58	50	53	13	11	36
K46	23	67	95	96	62	65	36	49	17	22	23	22	29	86	81	28	88	23
K47	46	58	23	54	54	80	29	69	61	60	99	93	15	33	76	82	89	63
K48	78	61	14	71	25	13	20	55	58	12	17	58	31	69	71	59	33	73
K49	93	83	29	58	44	78	100	36	48	87	92	35	99	46	77	94	98	37
K50	99	58	96	53	93	91	73	56	26	61	51	63	49	59	54	23	13	39

KOTA	K19	K20	K21	K22	K23	K24	K25	K26	K27	K28	K29	K30	K31	K32	K33	K34	K35	K36
K1	88	20	11	55	82	30	49	33	32	94	92	34	86	33	38	59	35	49
K2	42	97	43	75	98	75	64	44	18	22	96	49	21	78	58	33	82	66
K3	30	80	25	38	37	68	62	42	19	13	23	84	18	13	100	64	35	69
K4	76	19	61	98	93	55	37	52	25	28	34	58	21	16	26	60	100	45
K5	94	40	85	98	59	57	37	32	98	12	11	61	73	83	77	95	17	66
K6	27	45	85	29	49	72	75	71	62	45	44	27	70	53	75	55	19	42
K7	17	55	30	48	100	96	54	72	73	24	22	33	20	80	90	95	83	82
K8	65	44	21	48	18	46	76	25	88	76	22	12	61	80	66	38	51	96
K9	85	28	79	54	36	27	51	26	42	27	80	27	89	43	82	71	67	11
K10	51	59	68	89	78	83	78	35	12	26	12	29	35	13	67	11	69	84
K11	68	57	30	78	70	33	42	94	57	61	23	65	28	61	28	22	40	35
K12	69	71	30	28	40	63	32	31	65	29	81	51	64	73	30	37	39	75
K13	23	22	86	37	34	78	36	94	55	59	63	47	52	67	40	87	23	26
K14	54	86	63	70	38	56	75	51	43	47	21	67	88	34	84	79	99	21
K15	52	87	54	100	52	56	71	80	83	88	80	57	100	34	60	28	74	34
K16	94	26	78	60	65	62	48	29	78	50	25	26	72	36	22	91	85	34
K17	77	75	100	91	89	67	75	44	55	89	63	16	70	38	91	84	83	27
K18	24	82	84	55	13	44	76	98	14	24	42	89	40	13	26	12	96	31
K19	0	63	22	75	55	13	67	99	78	82	71	17	73	71	92	30	62	69
K20	73	0	33	77	98	81	54	27	94	33	21	88	29	29	95	54	57	68
K21	84	77	0	14	87	12	54	38	23	25	32	39	53	44	49	40	50	96
K22	73	85	100	0	22	98	48	40	19	39	14	73	21	54	43	69	72	11
K23	46	24	77	22	0	33	41	70	26	22	14	28	81	23	28	90	69	74
K24	79	25	36	66	85	0	70	21	86	46	41	95	13	32	65	56	57	40
K25	23	41	48	89	28	87	0	40	92	52	54	97	97	22	74	96	26	74

KOTA	K19	K20	K21	K22	K23	K24	K25	K26	K27	K28	K29	K30	K31	K32	K33	K34	K35	K36
K26	98	83	71	70	71	29	72	0	95	29	100	45	68	30	43	81	79	88
K27	93	99	71	74	44	39	47	22	0	29	45	84	37	90	81	15	77	39
K28	49	78	72	38	74	25	19	81	31	0	48	52	45	42	25	93	78	47
K29	70	61	13	29	28	30	17	63	42	42	0	65	90	13	28	96	94	62
K30	65	37	77	34	78	22	26	12	47	94	52	0	12	84	61	60	56	50
K31	91	32	98	44	47	78	89	90	51	31	14	38	0	94	95	37	17	54
K32	14	27	48	61	56	23	40	58	74	28	97	88	80	0	96	43	95	43
K33	62	46	86	29	76	94	57	91	29	37	37	71	81	38	0	15	24	67
K34	78	23	35	16	79	58	37	28	34	73	21	58	76	26	52	0	60	38
K35	58	23	30	56	43	58	74	43	67	58	88	33	47	57	40	34	0	93
K36	40	69	96	36	40	21	30	22	36	95	25	17	48	49	36	79	100	0
K37	97	23	56	52	55	19	73	60	81	61	39	55	22	54	54	82	23	97
K38	28	24	40	79	43	68	33	87	49	31	24	41	99	22	95	47	89	97
K39	17	61	74	73	68	95	44	27	73	70	12	52	22	44	80	11	98	19
K40	76	78	23	82	22	69	37	51	45	66	50	66	14	71	99	37	98	76
K41	35	79	56	68	37	50	91	54	46	69	35	59	36	34	33	94	75	19
K42	30	42	38	91	99	56	16	58	48	11	17	89	85	79	43	97	54	60
K43	54	24	40	30	95	82	48	19	51	54	32	95	73	52	52	31	99	27
K44	71	39	65	75	28	92	40	90	68	41	19	97	87	35	47	95	35	53
K45	41	94	52	96	43	28	71	74	70	55	81	59	35	41	70	51	53	68
K46	20	50	57	57	34	23	70	14	36	55	48	58	27	50	65	85	79	67
K47	60	79	39	24	92	65	100	90	36	100	37	84	69	73	93	54	15	77
K48	66	62	69	35	19	95	90	53	49	54	41	70	59	38	32	57	52	87
K49	17	63	70	69	37	89	80	61	72	93	94	73	36	78	47	85	89	72
K50	82	23	27	55	43	77	35	75	75	85	97	19	74	70	54	48	28	13

KOTA	K37	K38	K39	K40	K41	K42	K43	K44	K45	K46	K47	K48	K49	K50
K1	52	44	38	30	39	72	23	30	17	63	100	77	97	32
K2	43	15	66	98	91	12	71	62	53	22	44	41	38	66
K3	65	18	26	46	74	35	12	24	76	36	15	19	37	87
K5	38	31	65	93	56	45	19	72	74	34	18	90	88	16
K6	88	46	52	62	61	33	27	46	38	82	75	74	46	25
K7	82	66	79	68	77	38	51	100	48	71	82	98	16	21
K8	88	56	52	96	68	77	76	38	86	52	60	65	38	52
K9	41	37	92	66	50	99	53	49	95	72	52	18	28	11
K10	56	88	33	15	90	80	84	23	90	30	56	27	12	96
K11	32	18	33	33	62	20	74	45	36	89	90	17	52	92
K12	73	79	55	34	37	23	70	65	76	100	47	49	21	59
K13	23	25	11	23	69	96	61	40	38	56	90	90	82	69
K14	27	13	28	43	27	52	24	50	24	37	20	76	84	81
K15	89	90	63	19	85	23	97	74	73	38	94	63	34	89
K16	75	52	63	77	17	17	37	81	54	99	25	43	96	27
K17	15	80	42	95	85	38	17	18	81	46	74	14	27	71
K18	59	20	59	35	54	69	99	82	53	29	97	78	42	76
K19	66	85	51	65	48	18	66	69	56	94	18	75	92	13
K20	44	92	13	77	14	31	82	24	71	54	77	16	36	51
K21	81	74	70	49	76	91	94	86	20	99	41	100	41	76
K22	42	17	64	81	27	83	61	85	38	42	12	48	28	46
K23	28	14	78	89	70	42	11	46	36	83	95	43	62	99
K24	49	83	94	67	19	69	31	61	35	23	17	81	83	88

KOTA	K37	K38	K39	K40	K41	K42	K43	K44	K45	K46	K47	K48	K49	K50
K25	16	96	92	84	88	54	31	77	38	94	13	58	79	41
K26	98	15	98	28	51	84	79	17	47	65	54	53	70	23
K27	36	75	36	50	20	72	51	71	17	34	17	68	84	93
K28	97	76	66	22	15	28	35	58	28	52	44	61	98	83
K29	56	66	15	25	59	38	86	15	15	85	36	91	92	45
K30	56	24	96	35	84	32	46	92	94	44	49	59	66	76
K31	76	89	93	14	17	38	88	87	21	27	76	92	98	21
K33	77	30	69	32	38	35	61	100	56	59	100	36	35	24
K34	32	82	46	98	61	15	65	54	67	88	48	87	50	65
K35	37	44	47	21	43	33	20	26	100	98	83	46	54	32
K36	47	90	88	71	88	92	44	47	13	34	32	35	61	28
K37	0	83	12	76	12	44	33	82	29	73	59	23	63	17
K38	26	0	45	37	49	27	53	75	93	48	42	11	58	14
K39	93	19	0	88	23	19	84	92	98	90	76	75	44	22
K40	94	82	14	0	45	30	79	29	72	65	35	65	89	29
K41	88	47	92	64	0	15	60	92	44	100	26	53	48	67
K42	83	34	37	16	46	0	35	36	66	28	33	45	52	74
K43	53	67	23	72	25	56	0	98	72	57	36	47	23	57
K44	51	17	42	36	92	26	62	0	99	12	12	49	91	33
K45	39	71	29	58	19	55	72	18	0	24	95	31	33	30
K46	38	85	65	45	33	90	84	35	100	0	31	49	67	32
K47	77	53	64	49	20	87	20	40	92	50	0	86	16	12
K48	85	46	45	20	63	75	14	83	38	93	56	0	75	69
K49	72	82	90	77	25	26	23	15	56	31	51	52	0	13
K50	37	38	47	87	83	46	62	41	25	54	20	32	73	0

Lampiran B.

Tabel 2.22 Percobaan 1 Uji MR = 0,1

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2827	121563	800
2	2894	121503	450
3	2873	122595	380
4	2543	1208	640
5	2675	121259	480
6	2769	119964	680
7	2878	120499	470
8	2709	122213	890
9	3423	121669	360
10	2927	120775	920
Rata- rata	2851,8	109324,8	607

Tabel 2.23 Percobaan 2 Uji MR = 0,2

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2539	112067	620
2	3238	98592	700
3	2959	99694	290
4	3354	100524	700
5	2978	99252	730
6	3025	98797	790
7	2613	98047	650
8	2542	98514	290
9	2973	98629	380
10	2828	98343	500
Rata- rata	2904,9	100245,9	565

Tabel 2.24 Percobaan 3 Uji MR = 0,3

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2963	98163	710
2	3238	98592	700
3	2959	99694	290
4	3354	100524	700
5	2978	99252	730
6	3025	98797	790

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
7	2613	98047	650
8	2542	98514	290
9	2973	98629	380
10	2828	98343	500
Rata- rata	2947,3	98855,5	510

Tabel 4.25 Percobaan 4 Uji MR = 0,49

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2989	75486	505
2	2259	7445	720
3	2415	75605	620
4	2513	75358	730
5	3198	75451	520
6	2601	74853	680
7	2682	75754	630
8	2746	76432	320
9	2311	76594	830
10	2722	75818	710
Rata- rata	2643,6	68879,6	626,5

Tabel 2.26 Percobaan 1 Uji CDC = 0,1

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2869	7.0471	620
2	2742	7.0424	290
3	2629	7.0704	890
4	3131	7.1086	780
5	2822	6.7459	900
6	2921	7.0166	300
7	2957	7.0853	720
8	2879	6.6245	130
9	2787	7.0429	950
10	2579	7.072	670
Rata- rata	2831,6	63490,9	625

Tabel 2.27 Percobaan 2 Uji CDC = 0,2

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2812	7.3362	520
2	3090	7.1203	870
3	2869	7.2501	880
4	2786	7.2046	880
5	2761	7.1947	580
6	2557	7.1573	800
7	2957	7.0853	720
8	2855	7.1718	840
9	2772	7.2802	860
10	2889	7.3978	530
Rata- rata	2834,8	72198,3	748

Tabel 2.28 Percobaan 3 Uji CDC = 0,5

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	3188	7.3789	80
2	2564	7.3592	180
3	3227	7.258	550
4	2726	7.2519	930
5	2978	7.2502	720
6	2813	7.3112	460
7	3109	7.2788	550
8	2752	7.3344	400
9	2711	7.2572	950
10	2927	120775	920
Rata- rata	2899,5	71225,1	574

Tabel 2.29 Percobaan 4 Uji CDC = 0,8

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2809	7.54	440
2	2654	7.3919	280
3	2141	7.4856	880
4	2786	7.2046	880
5	3393	7.4796	720
6	2600	7.3501	830

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
7	2293	7.3854	660
8	2362	7.5524	890
9	2875	7.3947	620
10	2779	7.3938	800
Rata- rata	2669,2	67713,5	700

Tabel 4.30 Percobaan 1 Uji $c = 0.5$

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2582	7.4405	780
2	2638	7.3981	770
3	2866	7.3517	880
4	2867	7.4657	780
5	3109	7.419	680
6	2235	7.416	820
7	2556	7.4519	600
8	2665	7.4509	500
9	3131	7.4556	780
10	2713	7.4958	720
Rata- rata	2736,2	60993,7	731

Tabel 4.31 Percobaan 2 Uji $c = 0.1$

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	3739	10844	70
2	3565	11899	80
3	3918	1094	30
4	3677	11015	160
5	3730	10835	90
6	3792	11001	20
7	3329	1126	70
8	3861	11142	320
9	3860	11132	190
10	4164	112	20
Rata- rata	3763,5	8020	105

Tabel 2.32 Percobaan 3 Uji c = 0.05

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	3793	10986	30
2	3995	10893	20
3	3871	10569	80
4	4107	11583	30
5	3757	10833	90
6	3966	11432	50
7	4102	10879	40
8	3757	10777	120
9	4065	1064	90
10	3892	10919	70
Rata- rata	3930,5	9993,5	62

Tabel 2.33 Percobaan 4 Uji c = 0.01

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	2249	11728	100
2	2241	12177	20
3	2303	11692	80
4	2277	11824	40
5	2187	11673	80
6	2174	11972	100
7	2003	11924	50
8	2301	11813	180
9	2073	12	120
10	2041	12146	20
Rata- rata	2184,9	10696,1	79

Tabel 4.34 Percobaan 2 Uji SRD = 0,1

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2320	7.4165	700
2	2545	7.5437	770
3	2457	7.4368	800
4	2682	5208	600
5	2270	7.4468	605
6	2218	7.4643	900

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
7	2742	7.4817	850
8	2333	7.4255	860
9	2176	7.3844	580
10	2403	7.4131	560
Rata- rata	2414,6	66553,6	722,5

Tabel 4.35 Percobaan 2 Uji SRD = 0,2

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2720	7.5259	690
2	2626	7.5105	780
3	2294	7.5115	910
4	2654	7.5336	280
5	2141	7.5922	880
6	3393	7.496	730
7	2600	7.558	840
8	2293	7.5762	670
9	2362	7.5	880
10	2519	7.5085	890
Rata- rata	2560,2	54271,3	755

Tabel 4.36 Percobaan 3 Uji SRD = 0,3

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	2889	7.6152	870
2	2620	7.6151	740
3	2908	7.5356	940
4	2841	7.595	920
5	2310	7.6885	840
6	2140	7.519	870
7	2498	7.4632	790
8	2834	7.4981	280
9	2413	7.5247	800
10	2173	7.5833	720
Rata- rata	2562,6	62035,1	777

Tabel 4.37 Percobaan 4 Uji SRD = 0,5

No.	Algoritma CSO		
	Jarak	Waktu	Kekonvergenan
1	1876	79033	730
2	1857	83024	380
3	1769	81882	780
4	1860	79063	280
5	1794	79322	550
6	1863	79389	880
7	1903	80581	620
8	1898	79685	280
9	1890	79577	380
10	1920	79448	320
Rata- rata	1863	801	520

Tabel 4.38 Percobaan 1 Uji c1 = 0.5

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	3695	1.116	350
2	3883	1.1353	80
3	3634	1.1424	190
4	3815	1.1188	70
5	4099	1.1479	390
6	3863	1.1576	100
7	3663	1.1416	280
8	3561	1.159	200
9	3781	1.1622	40
10	3581	1.1414	120
Rata- rata	3757,5	9374,7	182

Tabel 4.39 Percobaan 2 Uji c1 = 0.1

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	3739	10844	70
2	3565	11899	80
3	3918	1094	30
4	3677	11015	160
5	3730	10835	90

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
7	3329	1126	70
6	3792	11001	20
8	3861	11142	320
9	3860	11132	190
10	4164	112	20
Rata- rata	3763,5	8020	105

Tabel 4.40 Percobaan 3 Uji c1 = 0.05

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	3793	10986	30
2	3995	10893	20
3	3871	10569	80
4	4107	11583	30
5	3757	10833	90
6	3966	11432	50
7	4102	10879	40
8	3757	10777	120
9	4065	1064	90
10	3892	10919	70
Rata- rata	3930,5	9993,5	62

Tabel 4.41 Percobaan 4 Uji c1 = 0.01

No.	Algoritma PSO		
	Jarak	Waktu	Kekonvergenan
1	2249	11728	100
2	2241	12177	20
3	2303	11692	80
4	2277	11824	40
5	2187	11673	80
6	2174	11972	100
7	2003	11924	50
8	2301	11813	180
9	2073	12	120
10	2041	12146	20
Rata- rata	2184,9	10696,1	79