TESIS - SM 142501

PENERAPAN ALGORITMA DISCRETE CUCKOO SEARCH PADA MULTIPLE TRAVELLING SALESMAN PROBLEM

HANDY FRANK WILY AMBARITA
NRP 0611 1650 010 020

DOSEN PEMBIMBING:
Prof. Dr. Mohammad Isa Irawan, M.T.
Dr. Budi Setiyono, S.Si., M.T.

PROGRAM MAGISTER
DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

THESIS - SM 142501

IMPLEMENTATION OF DISCRETE CUCKOO SEARCH ALGORITHM ON MULTIPLE TRAVELLING SALESMAN PROBLEM

HANDY FRANK WILY AMBARITA
NRP 0611 1650 010 020

SUPERVISORS:
Prof. Dr. Mohammad Isa Irawan, M.T.
Dr. Budi Setiyono, S.Si., M.T.

MASTER PROGRAM
DEPARTMENT OF MATHEMATICS
FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember

Oleh:
Handy Frank Wily Ambarita
NRP. 0611 1650 010 020

Tanggal Ujian : 30 Juli 2018
Periode Wisuda : September 2018

Disetujui oleh:

Prof. Dr. Mohammad Isa Irawan, M.T.
NIP. 19631225 198903 1 001
(Pembimbing)

Dr. Budi Setiyono, S.Si, M.T.
NIP. 19720207 199702 1 001
(Pembimbing)

Dr. Imam Mukhlash, S.Si, M.T.
NIP. 19700831 199403 1 003
(Penguji)

Dr. Hariyanto, M.Si.
NIP. 19530414 198203 1 002
(Penguji)

Dr. Dieky Adzkiya, S.Si, M.Si.
NIP. 19830517 200812 1 003
(Penguji)

Dekan FMKSD,

Prof. Dr. Basuki Widodo, M.Sc.
NIP. 19650605 198903 1 002

PENERAPAN ALGORITMA DISCRETE CUCKOO SEARCH PADA MULTIPLE TRAVELLING SALESMAN PROBLEM

Nama Mahasiswa	:	HANDY FRANK WILY AMBARITA
NRP	:	0611 1650 010 020
Pembimbing	:	1. Prof. Dr. Mohammad Isa Irawan, M.T.
		2. Dr. Budi Setiyono, S.Si., M.T.

ABSTRAK

Pada penelitian ini salah satu variasi dari Travelling Salesman Problem (TSP) yaitu Multiple Travelling Salesman Problem (MTSP) diselesaikan dengan algoritma Discrete Cuckoo Search dan algoritma k-means++. Untuk menyelesaikan permasalahan MTSP ini, pertama-tama dilakukan klustering dengan menggunakan k-means++ untuk mendapatkan kota yang membentuk rute dari setiap salesman.Kemudian setelah mendapatkan kluster-kluster, dilakukan pembentukan dan pengoptimalan rute pada kluster-kluster tersebut satu per satu menggunakan Discrete Cuckoo Search. Dari hasil percobaan didapat bahwa parameter: jumlah populasi (pop) dan banyak generasi (ngen), berpengaruh pada jarak terpendek yang didapat serta waktu komputasi. Sedangkan untuk parameter fraksi Cuckoo yang jelek (p_a), tidak berpengaruh terhadap jarak terpendek yang dihasilkan, tetapi berpengaruh pada waktu komputasi. Jika nilai fraksi yang diberikan semakin mendekati satu, waktu komputasi semakin bertambah dengan $p_a = 0$ memiliki rerata waktu komputasi terkecil dan $p_a = 1$ adalah fraksi dengan rerata waktu komputasi terbesar. Selain itu kombinasi k-means++ dan Discrete Cuckoo Search dibandingkan dengan Ant Colony Optimization dan Genetic Algorithm dari penelitian lain.

Kata-kunci: Discrete Cuckoo Search, Multiple Travelling Salesman Problem, k-means++

IMPLEMENTATION OF DISCRETE CUCKOO SEARCH ALGORITHM ON MULTIPLE TRAVELLING SALESMAN PROBLEM

Name	:	HANDY FRANK WILY AMBARITA
NRP	:	0611 1650 010 020
Supervisors	:	1. Prof. Dr. Mohammad Isa Irawan, M.T.
		2. Dr. Budi Setiyono, S.Si., M.T.

ABSTRACT

In this thesis writer combine Discrete Cuckoo Search and k-means++ to solve Multiple Travelling Salesman Problem (MTSP) with single depot which is nondeterministc polynomial-hard problem. k-means++ is used to find the city for route of every salesman. Once the route is found, Discrete Cuckoo Search is used to create route and optimize route. From the experiments it gets parameters such as number of population (pop) and number of generation (ngen) affect the solution from Discrete Cuckoo Search and its time of computation. Fraction of the worst Cukoo (p_a) does not affect the algorithm to get solution. It only affects the time of computation. Every increasing of value of p_a, get the time of computation bigger. The smallest time of computation is found on $p_a = 0$ and the biggest is on $p_a = 1$. Also, the combination of method is compared with Ant Colony Optimization and Genetic Algorithm.

Key-words: Discrete Cuckoo Search, Multiple Travelling Salesman Problem, k-means++

KATA PENGANTAR

Segala puji dan syukur bagi Tuhan Yang Maha Kuasa karena atas kuasa dan rahmatnya penulis diberikan kesempatan untuk menyelesaikan tesis yang berjudul

PENERAPAN ALGORITMA DISCRETE CUCKOO SEARCH PADA MULTIPLE TRAVELLING SALESMAN PROBLEM

Tesis ini dibuat untuk memenuhi salah satu syarat dalam memperoleh gelar Magister Program Magister Matematika, Fakultas Matematika, Komputasi dan Sains Data, Institut Teknologi Sepuluh Nopember.

Penyusunan tesis ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan terima kasih kepada pihak-pihak tersebut diantaranya:

1. Rektor Institut Teknologi Sepuluh Nopember.
2. Dekan Fakultas Matematika, Komputasi dan Sains Data, Institut Teknologi Sepuluh Nopember.
3. Kepala Departemen Matematika, Fakultas Matematika, Komputasi, dan Sains Data, Institut Teknologi Sepuluh Nopember.
4. Kepala Program Studi Magister Matematika, Fakultas Matematika, Komputasi dan Sains Data, Institut Teknologi Sepuluh Nopember.
5. Prof. Dr. Mohammad Isa Irawan, M.T dan Dr. Budi Setiyono, S.Si., M.T. selaku dosen yang senantiasa membimbing penulis dalam menyelesaikan penelitian tesis ini.
6. Dr. Imam Mukhlash, S.Si., M.T., Dr. Hariyanto, M.Si., dan Dr. Dieky Adzkiya, S.Si, M.Si. selaku dosen penguji yang telah banyak memberikan masukan dalam penyelesaian tesis ini.
7. Prof. Dr. Basuki Widodo, M.Sc. selaku dosen wali penulis selama menempuh program studi Magister Matematika.
8. Teman-teman seperjuangan di Program Studi Magister Matematika. Terima kasih banyak atas segala sesuatunya yang telah diberikan selama menjalani perkuliahan Magister Matematika.
9. Staff Pasca Sarjana Matematika, Mbak Resty dan Mas Afif, Mas Ali, Mas Agung, dan Pak Joko. Terima kasih banyak atas bantuan dalam menginformasikan keperluan administrasi dan bersedia menampung keluh kesah penulis selama proses penyelesaian tesis hingga kelulusan.

10. Bapak, Mamak, Desy, Hegar, dan Samuel. Kalian sebagai pengingat kenapa abang harus menyelesaikan tesis ini.

11. Teman-teman kontrakan (Anas, Hakam, Heri, dan Wiwid). Terima kasih untuk kebersamaannya.

12. Ministri ZEAL Surabaya, Bang Apul, Bang Jesman, dan Kak Friska, keluarga "Mukidi", terutama Dyah Kristina Ambarwati a.k.a Nana atas semangat, motivasi, dan perhatiannya untuk penulis.

Penulis berharap semoga tesis ini dapat bermanfaat bagi pembaca, dan semua yang telah dikerjakan ini menjadi berkat buat kita semua.

Surabaya, Agustus 2018

Handy Frank Wily Ambarita

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PENGESAHAN v

ABSTRAK vii

ABSTRACT ix

KATA PENGANTAR xi

DAFTAR ISI xiii

DAFTAR GAMBAR xv

DAFTAR TABEL xvii

BAB 1 PENDAHULUAN 1
- 1.1 Latar Belakang 1
- 1.2 Rumusan Masalah 4
- 1.3 Batasan Masalah 4
- 1.4 Tujuan Penelitian 4
- 1.5 Manfaat Penelitian 4

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI 7
- 2.1 Penelitian-Penelitian Terkait 7
- 2.2 Travelling Salesman Problem 7
- 2.3 Multiple Travelling Salesman Problem 9
- 2.4 Algoritma Cuckoo Search 11
  - 2.4.1 Analisis Konvergensi Algoritma Cuckoo Search 13
  - 2.4.2 Perbedaan Dengan Beberapa Algoritma Metaheuristik yang Lain 15
  - 2.4.3 Discrete Cuckoo Search 16
  - 2.4.4 Distribusi Levy 18
- 2.5 k-means++ 18

BAB 3 METODE PENELITIAN 21
- 3.1 Tahapan Penelitian 21

BAB 4 HASIL DAN PEMBAHASAN 25
- 4.1 Clustering Data 25
  - 4.1.1 Jumlah Maksimum Salesman 25
  - 4.1.2 Proses Klustering Dengan k-means++ 26
- 4.2 Cuckoo Search Untuk Hasil Clustering 29
  - 4.2.1 Pembentukan Solusi 30
  - 4.2.2 *Update* Solusi 32
  - 4.2.3 Ruang Pencarian 34

4.3 Pembangkitan Bilangan Acak Berdistribusi Levy 41
4.4 Hasil Percobaan 45
4.4.1 ngen=50 Dan pop=35 46
4.4.2 ngen=100 Dan pop=35 46
4.4.3 ngen=200 Dan pop=35 47
4.4.4 ngen=50 Dan pop=25 48
4.4.5 ngen=100 Dan pop=25 49
4.4.6 ngen=200 Dan pop=25 49
4.4.7 Diskusi Hasil 49
4.4.8 Perbandingan Hasil Penelitian Lain 54

BAB 5 KESIMPULAN 55

DAFTAR PUSTAKA 57

BIOGRAFI PENULIS 61

DAFTAR GAMBAR

Gambar 2.1 (Latah, Majd, 2016) Solusi yang diperoleh oleh Latah dengan menggunakan 50 kota dan 3 salesman. 8
Gambar 2.2 Ilustrasi satu wilayah dengan tujuh kota yang harus dikunjungi. . 8
Gambar 2.3 Salah satu solusi yang ditemukan. 11
Gambar 2.4 Sirkuit dengan 15 kota. 11
Gambar 2.5 2-opt. (**a**) Tur awal sebelum dikenakan operator. (**b**) Tur baru yang dihasilkan dari operator 2-opt [*edge* (a,b) dan (c,d) dibuang, kemudian diganti dengan *edge* (a,c) dan (b,d)] 13
Gambar 2.6 Sirkuit dengan 15 kota yang sudah dikenakan double-bridge. 13
Gambar 2.7 Contoh representasi sebuah telur Cuckoo untuk salah satu solusi dari MTSP 17

Gambar 3.1 Diagram blok metode penelitian 24

Gambar 4.1 24 titik yang belum dikluster. 26
Gambar 4.2 Titik kuning dengan tepian hitam terpilih sebagai centroid awal. 27
Gambar 4.3 Dua centroid awal lain terpilih. 27
Gambar 4.4 Hasil kluster 24 titik. 28
Gambar 4.5 Titik merah yang disisipkan pada kluster lain. 29
Gambar 4.6 29
Gambar 4.7 Ruang pencarian dengan menggunakan 2-opt pada 5 kota. 34
Gambar 4.8 Visualisasi 2-opt untuk 5 kota. 35
Gambar 4.9 Hasil operator 2-opt. 36
Gambar 4.10 Sirkuit dengan 9 kota. 38
Gambar 4.11 Sirkuit dengan 9 kota yang sudah dikenakan operator 2-opt. 38
Gambar 4.12 Sirkuit dengan 15 kota. 39
Gambar 4.13 Hasil operator double-bridge. 39
Gambar 4.14 Sirkuit dengan 15 kota yang sudah dikenakan double-bridge. 40
Gambar 4.15 Data dummy 24 kota 42
Gambar 4.16 Data kota yang dikluster. 43
Gambar 4.17 Hasil setelah dikenakan Cuckoo Search. 44
Gambar 4.18 Total jarak terpendek yang didapat dari masing-masing p_a untuk att48.xlsx 48

Gambar 4.19 Rerata waktu komputasi dari 3 generasi dengan jumlah populasi yang sama 48
Gambar 4.20 Rerata waktu dari $ngen = 100, 200$ dan $pop = 25, 35$ untuk att48.xlsx 50
Gambar 4.21 Rerata jarak dari beberapa generasi untuk data att48.xlsx 51
Gambar 4.22 Hasil terbaik dari att48.xlsx 52
Gambar 4.23 Rerata jarak dari $ngen = 100, 200$ dan $pop = 25, 35$ pada att48.xlsx 52
Gambar 4.24 Perbandingan rerata jarak terpendek dengan rerata waktu komputasi 53
Gambar 4.25 Hasil dari kluster [6,15,3] 53

DAFTAR TABEL

Tabel 2.1 Perbandingan CS, GA, dan PSO 16

Tabel 4.1 Hasil dengan menggunakan $ngen = 50$ dan $pop = 35$ 46
Tabel 4.2 Hasil dengan $ngen = 100$ dan $pop = 35$ 47
Tabel 4.3 Hasil dengan $ngen = 200$ dan $pop = 35$ 47
Tabel 4.4 Hasil dengan $ngen = 50$ dan $pop = 25$ 49
Tabel 4.5 Hasil dengan $ngen = 100$ dan $pop = 25$ 49
Tabel 4.6 Hasil dengan $ngen = 200$ dan $pop = 25$ 50
Tabel 4.7 Hasil dengan beberapa jenis kluster pada data dumm.xlsx 51
Tabel 4.8 Perbandingan dengan algoritma lain 54

BAB 1
PENDAHULUAN

Pada bab ini akan disampaikan latar belakang permasalahan beserta tujuan dan manfaat dari penyelesaiaan permasalahan.

1.1 Latar Belakang

Ada banyak metode untuk menyelesaikan permasalahan optimasi. Terkadang, permasalahan optimasi sulit untuk diselesaikan. Salah satu alasannya adalah karena ruang pencarian (*search space*) yang terlalu besar. Hal ini mengakibatkan metode matematika konvensional sulit untuk menyelesaikannya (Said, Gamal Abd El-Nasser A. et al, 2014).

Karena alasan tersebut menurut Azmi Alazam et al pada (Said, Gamal Abd El-Nasser A. et al, 2014) banyak algoritma metaheuristik yang telah dikembangkan untuk menyelesaikan permasalahan optimisasi. Algoritma ini bekerja pada permasalahan yang mana kita hanya mendapatkan sedikit petunjuk atas suatu persoalan. Bahkan terkadang tidak memerlukan asumsi seperti yang ada pada teknik Gradient Ascent (Luke, Sean, 2013).

Selain itu, menurut Gamal (Said, Gamal Abd El-Nasser A. et al, 2014) algoritma metaheuristik adalah proses mengkombinasikan konsep kecerdasan yang beragam untuk mengeksplorasi dan mengeksploitasi ruang pencarian, yang mana strategi pembelajaran digunakan untuk membentuk informasi (solusi) agar dapat menemukan solusi sekitar yang optimal dengan cara yang efisien. Xin-She (X.-S. Yang, 2010) mengatakan bahwa kecenderungan saat ini mendefinisikan algoritma metaheuristik adalah algoritma yang menggunakan konsep stokastik atau keteracakan dan pencarian lokal. Menurutnya konsep keteracakan ini memberikan cara yang baik untuk berpindah dari pencarian lokal ke pencarian yang berskala besar (global).

Salah satu algoritma metaheuristik yang sering digunakan adalah Algoritma Cuckoo Search. Algoritma ini terinspirasi dari proses perkembangbiakan burung Cuckoo terutama dalam proses ketika akan menaruh atau memilih sarang untuk telur dari burung tersebut. Salah satu keunggulan dari Algoritma Cuckoo Search adalah keseimbangan

pada eksploitasi dan eksplorasi dari ruang pencarian oleh Cuckoo dengan menggunakan *Levy flight*. Keunggulan lain dibanding algoritma lain seperti PSO (*Particle Swarm Optimization*) dan algoritma Genetika adalah pada kesederhanaan (jumlah dan *setting*-an) parameter yang digunakan (X.-S. Yang et al, 2009).Algoritma ini telah banyak digunakan dan dikembangkan/diimprovisasi (Civicioglu, Pinar et al, 2011), (I. Fister Jr. et al, 2014), dan (Joshi, Anuja.S. et al, 2017).

Travelling salesman problem adalah permasalahan yang membahas pencarian rute optimal dari seorang *salesman* yang berangkat dari sebuah titik awal; mengunjungi keseluruhan kota dari satu wilayah yang telah ditetapkan untuk kemudian kembali ke titik awal sedemikan hingga total jarak yang ditempuh adalah yang paling kecil (minimum) dan setiap kota hanya boleh dikunjungi satu kali saja. Permasalahan ini diformulasikan secara matematis pada tahun 1930 dan telah menjadi salah satu permasalahan yang intens/sering untuk dibahas di berbagai negara.

Masalah TSP memiliki kompleksitas waktu penyelesaiannya yang akan meningkat secara eksponensial terhadap ukuran dari input yang diberikan(Jati, Gilang Kusuma et al, 2011). Banyak metode atau algoritma untuk menyelesaikan permasalahan TSP. Misal Lin-Kernighan Heuristic, Algoritma Genetika, dan Discrete Cuckoo Search.

Algoritma Discrete Cuckoo Search adalah salah satu bentuk modifikasi dari algoritma Cuckoo Search. Algoritma ini bekerja mirip dengan algoritma Cuckoo Search. Yang menjadi pembeda adalah pada penggunaan persamaan yang disebut dengan *Levy Flight*. Algoritma Discrete Cuckoo Search bisa dikatakan sebagai algoritma Cuckoo Search pada permasalahan yang bersifat diskrit seperti pada masalah kombinatorik. Menurut Aziz (Ouaarab, Aziz et al, 2014a) algoritma Discrete Cuckoo Search yang digunakannya sangat baik dalam menyelesaikan permasalahan TSP.

Salah satu bentuk dari Travelling Salesman Problem adalah Multiple Travelling Salesman Problem (MTSP). MTSP adalah salah satu persoalan yang juga populer untuk dibahas. Permasalahan MTSP pernah diselesaikan dengan menggunakan algoritma Ant Colony System yang diajukan oleh (Kencana, Eka N. et al, 2017) dan (Necula, Raluca et al, 2015) dan M. Sedighpour dengan menggunakan algoritma Genetika.

Pada persoalan MTSP, terdapat m *salesman*. Masing-masing *salesman* mengunjungi sebuah himpunan kota yang berbeda dengan *salesman* yang lain dan setiap kota dalam sebuah himpunan kota tidak berada dalam himpunan

kota yang lain kecuali kota asal (depot). Setiap *salesman* berangkat dari satu depot yang sama dan kembali ke depot tersebut. Kemudian dicari rute terpendek bagi setiap salesman yang merupakan sirkuit Hamilton untuk mengunjungi setiap kota yang ada pada himpunan kota (Gutin G, Punen AP, 2002). Secara umum rute yang terbentuk dari setiap salesman akan membentuk kelompok kota yang berbeda-beda kecuali pada titik awal dari rute mereka. Pengelompokan ini secara tidak langsung menunjukkan proses pengklusteran.

Terdapat beberapa algortima pengklusteran yang sering dgunakan,yaitu k-means dan k-means++. Kedua algoritma ini akan menghasilkan kelompok atau kluster berdasarkan kedekatan titik yang ada pada kluster dengan centroid yang didapat.

Algoritma k-means adalah metode yang popular untuk mengkluster data berbasis vektor secara otomatis. Metode ini digunakan pada estimasi kepadatan, pengkarakteran perilaku beban kerja, pemampatan citra, identifikasi topik secara otomatis dll. Algoritma ini juga termasuk sepuluh besar algoritma yang digunakan pada data mining. k-means++ adalah pengembangan dari metode k-means. k-means biasa akan memilih centroid awal (*initial centroid*) secara acak. Hal ini terkadang dapat menghasilkan hasil kluster yang buruk atau proses menuju konvergensi yang lambat. Maka dari itu D. Arthur dan S. Vassilvitskii mengusulkan cara pemilihan centroid awal yang berbeda. Metode k-means++ menggunakan peluang yang disebut D^2 weighting. Dan berdasarkan penelitan mereka, k-means++ lebih baik dibandingkan dengan k-means biasa. Proses pemilihan kota yang nantinya membentuk rute dari setiap *salesman* dilakukan dengan proses klustering menggunakan metode k-means++ (Arthur, David Vassilvitskii. Sergei, 2007).

Permasalahan MTSP mirip dengan *Capacitated Vehicle Routing Problem* (CVRP). Pada CVRP diberikan sejumlah truk dan sejumlah kota. Setiap truk berangkat dari satu kota (depot) yang sama dan kembali ke kota yang sama tersebut. Yang membedakan dengan MTSP yang dijelaskan sebelumnya adalah pada kendala yang berupa kapasitas angkut/antar dari setiap truk yang akan mengunjungi kota yang diberikan. Penelitian ini pernah dilakukan oleh Shahab (Shahab, Muhammad Luthfi et al, 2016) dengan menggunakan *Two-Step Genetic Algorithm.*

Berdasarkan penelitian yang sudah dilakukan terkait Cuckoo Search pada TSP, MTSP dan pengklusteran menggunakan k-means++ maka pada penelitian ini digunakan algoritma Discrete Cuckoo Search yang merupakan

salah satu pengembangan dari algoritma Cuckoo Search untuk menyelesaikan atau mencari rute terpendek dari masing-masing *salesman* untuk persoalan MTSP yang dipadukan dengan algoritma k-means++.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan yang telah dijelaskan terutama terkait dengan Discrete Cuckoo Search dan MTSP, rumusan masalah untuk penelitian ini adalah bagaimana menyelesaikan *Multiple Travelling Salesman Problem* (MTSP) dengan menggunakan algoritma Discrete Cuckoo Search dan k-means++?

1.3 Batasan Masalah

Di dalam penelitian ini batasan masalah yang digunakan adalah sebagai berikut:

1. Pada model MTSP di penelitian ini, sejumlah *salesman* telah diberikan terlebih dahulu.

2. Model MTSP yang digunakan adalah model *symmetric*-MTSP yaitu, jarak dari kota i ke kota j dan dari kota j ke kota i dalam satu wilayah yang diteliti adalah sama.

3. Pada model MTSP yang diteliti, setiap *salesman* berangkat dari satu kota (depot) yang sama dan kembali lagi pada kota yang sama tersebut.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah menyelesaikan *Multiple Travelling Salesman Problem* dengan menggunakan algoritma Discrete Cuckoo Search dan k-means++.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Membantu menyelesaikan masalah yang terkait dengan aplikasi MTSP dan klustering seperti pendistribusian barang dan pengaturan rute salesman.

2. Memberi kajian baru dari penggunaan teknik klustering seperti k-means++ dengan algoritma Discrete Cuckoo Search.

3. Menambah wawasan dalam permasalahan optimasi kombinatorik terutama yang terkait dengan *Travelling Salesman Problem*.

4. Menambah wawasan dalam penerapan algoritma metaheuristik Discrete Cuckoo Search pada masalah optimasi kombinatorik seperti pada *Travelling Salesman Problem*.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Dalam bab ini dibahas mengenai kajian pustaka dan dasar teori yang berkaitan dengan topik penelitian tesis.

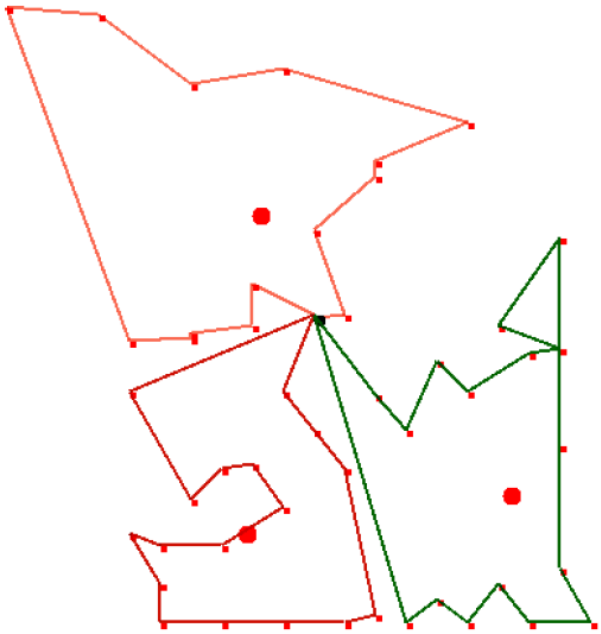
2.1 Penelitian-Penelitian Terkait

Sudah banyak penelitian yang dilakukakn terkait Cuckoo Search dan TSP. Pada tahun 2014 Ouaarab (Ouaarab, Aziz et al, 2014a) menyelesaikan persoalan Travelling Salesman Problem (TSP) dengan menggunakan Discrete Cuckoo Search. Di penelitian penulis ini dilakukan improvisasi pada algoritma Discrete Cuckoo Search dengan menambahkan sebuah kategori baru dari populasi burung Cuckoo yang disebut dengan *smart Cuckoo*. Kategori ini mampu memilih sarang yang lebih baik sehingga mencegah telur yang dihasilkan terbuang atau ditemukan oleh *host*. Sedangkan penelitian yang dilakukan oleh Jati (Jati, Gilang Kusuma et al, 2011), pada permasalahan yang sama, TSP, penulis melakukan modifikasi Discrete Cuckoo Search dengan tidak memakai*Levy flight*.

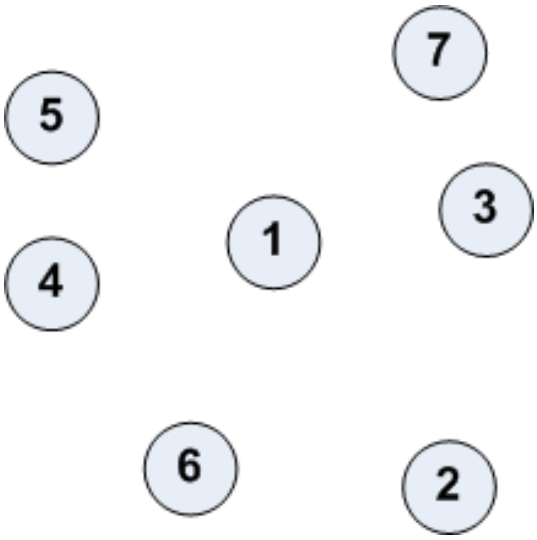
Variasi dari TSP yaitu MTSP juga banyak diteliti. Di tahun 2017 Kencana (Kencana, Eka N. et al, 2017) menyelesaikan salah satu variasi dari TSP, MTSP dengan menggunakan algoritma Ant System. Adapun Latah (Latah, Majd, 2016) di tahun 2016 menggunakan perpaduan k-Means dan algoritma Ant Colony System yang dimodifikasi. Gambar 2.1 adalah salah satu hasil dari penelitiannya. Necula (Necula, Raluca et al, 2015) juga melakukan penelitian MTSP dengan menggunakan Ant Colony System di tahun 2015.

2.2 Travelling Salesman Problem

Travelling Salesman Problem (TSP) adalah salah satu permasalahan optimasi klasik terkenal yang sulit untuk diselesaikan secara konvensional. Misalkan diberikan sebarang graf $\mathcal{G} = (\mathbb{V}, \mathbb{E})$. Untuk bobot setiap sisi $e \in \mathbb{E}$ adalah c_e, maka permasalahan TSP ialah mencari sebuah rute tur di $\mathcal{G}$ dengan jumlahan bobot dari semua sisi rute tersebut adalah yang paling kecil. Misalkan $\mathcal{G}$ adalah sebuah graf berarah, yang mana jika terdapat sisi yang hilang (*missing edge*), kita dapat asumsikan bahwa *edge* yang hilang



Gambar 2.1: (Latah, Majd, 2016) Solusi yang diperoleh oleh Latah dengan menggunakan 50 kota dan 3 salesman.



Gambar 2.2: Ilustrasi satu wilayah dengan tujuh kota yang harus dikunjungi.

tersebut memiliki nilai yang sangat besar. Misalkan himpunan titik $\mathbb{V} = 1, 2, \ldots, n$. Matriks $\mathbb{C} = (c_{ij})_{n \times n}$ disebut sebagi matriks biaya (*cost matrix*) yang bisa juga disebut matriks jarak atau matriks bobot, yang mana entri (i, j) yang disimbolkan oleh c_{ij} bersesuaian dengan biaya, bobot, atau jarak yang menghubungkan titik i ke titik j pada $\mathcal{G}$(Gutin G, Punen AP, 2002).
TSP dapat juga dinyatakan dengan formulasi berikut:

$$x_{i,j} = 1 \text{ , jika kota } j \text{ dikunjungi tepat dari kota } i \tag{2.1}$$

$$x_{i,j} = 0 \text{ , untuk yang lainnya.} \tag{2.2}$$

Minimalkan:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j}$$

dengan kendala:

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad , j = 2, \ldots, n \tag{2.3}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad , i = 2, \ldots, n \tag{2.4}$$

Kendala (2.3) dan (2.4)menyatakan bahwa setiap *salesman* hanya akan mengunjungi satu kali titik atau kota j dari kota i. Misalkan, untuk $j = 2$ dan $i = 4$ pada kendala (2.3), maka $x_{4,2}$ menyatakan bahwa *salesman* berangkat dari kota 4 ke kota 2. Tidak tujuan lain dari kota 4 selain kota 2. Sehingga untuk kendala (2.4), jumlahan tersebut akan bernilai 1 pada $i = 4$ dan $j = 2$ dan nol untuk j yang lain.

2.3 Multiple Travelling Salesman Problem

Pada kasus ini setiap *salesman* berangkat dari kota asal (depot) yang sama dan akan kembali ke kota tersebut. Setiap salesman mengunjungi kota yang berbeda-beda sebanyak satu kali. Permasalahan dari MTSP juga tidak jauh berbeda dengan TSP yaitu mencari total perjalanan terpendek dari setiap salesman.

Permasalahan MTSP dapat dinyatakan dengan ke dalam bentuk *Integer Linear Programming* sebagai berikut (Kencana, Eka N. et al, 2017) (Setiyono,Budi, 2002):

Diberikan variabel $x_{i,j}$ yang didefinisikan seperti pada (2.1) dan (2.2) dengan m menyatakan banyak *salesman* dan n adalah jumlah seluruh kota dari sebuah wilayah yang harus dikunjungi.
Minimalkan:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j}$$

dengan kendala:

$$\sum_{j=2}^{n} x_{1,j} = m \tag{2.5}$$

$$\sum_{j=2}^{n} x_{j,1} = m \tag{2.6}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad , j = 2, \ldots, n \tag{2.7}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad , i = 2, \ldots, n \tag{2.8}$$

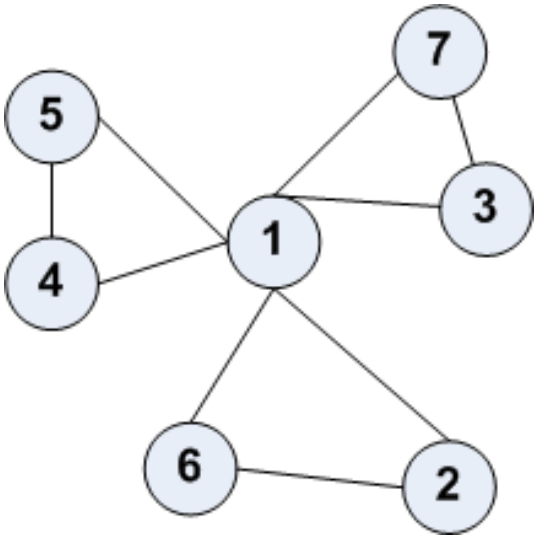
$$+ \text{ kendala eliminasi subtur} \tag{2.9}$$

$$x_{i,j} \in \{0, 1\}, \forall (i, j) \in A \tag{2.10}$$

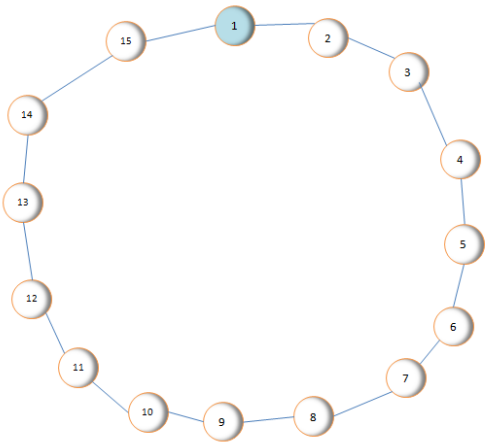
(2.7) dan (2.8) memastikan bahwa setiap kota hanya dikunjungi satu kali saja, kemudian (2.5) dan (2.6) menyatakan bahwa terdapat m *salesman* yang berkunjung dan kembali ke titik asal (dalam kasus ini dinyatakan dengan x_1) dan untuk (2.9) disebut dengan *subtour elimination constraints* (SEC).

Gambar 2.2 adalah salah satu contoh. Di contoh tersebut terdapat tujuh kota, dengan kota 1 (yang disimbolkan dengan lingkaran berisi angka 1) adalah depot atau kota asal dari masing-masing *salesman* dan pada contoh ini diberikan tiga *salesman*.

Kemudian jika diberikan tiga *salesman* pada wilayah yang ditunjukkan oleh Gambar 2.2, salah satu solusinya adalah seperti yang ditunjukkan oleh Gambar 2.3. Dari Gambar 2.3 dapat diperoleh informasi sebagai berikut: Salesman 1 memiliki rute perjalanan: kota 1→ kota 4→ kota 5→ kota 1, Salesman 2 memiliki rute perjalanan: kota 1→ kota 3→ kota 7→ kota 1, dan Salesman 3 memiliki rute perjalanan: kota 1→ kota 6→ kota 2→ kota 1.
Pada contoh ini, jarak antarkota dianggap sama. Sehingga rute kota 1→ kota 4→ kota 5→ kota 1 akan memiliki panjang lintasan yang sama dengan kota



Gambar 2.3: Salah satu solusi yang ditemukan.



Gambar 2.4: Sirkuit dengan 15 kota.

1→ kota 5→ kota 4→ kota 1 untuk salesman 1.

2.4 Algoritma Cuckoo Search

Algoritma Cuckoo Search dikembangkan oleh Xin-She Yang dan Suash Deb pada tahun 2009. Algoritma ini terinspirasi dari tingkah laku atau kebiasaan burung Cuckoo dalam proses perkembangbiakannya dan pencarian sarang untuk menaruh telur.

Dalam menggunakan algoritma Cuckoo Search diberikan tiga aturan:

1. Setiap burung Cuckoo menempatkan satu telur dalam satu waktu pada sebuah sarang secara acak;

2. Sarang terbaik adalah sarang yang memiliki telur dengan kualitas terbaik

dan telur tersebut nantinya lolos menjadi generasi selanjutnya;

3. Jumlah sarang yang tersedia tetap dan telur yang diletakkan oleh Cuckoo dapat ditemukan oleh burung pemilik sarang (*host*) dengan probabilitas $p_a \in [0, 1]$

Bisa saja *host* membuang telur atau membuat sebuah sarang yang baru. Dengan kalimat lain, sebanyak p_a dari n sarang digantikan oleh sarang baru.

Algorithm 1 Pseudo-Code Cuckoo Search

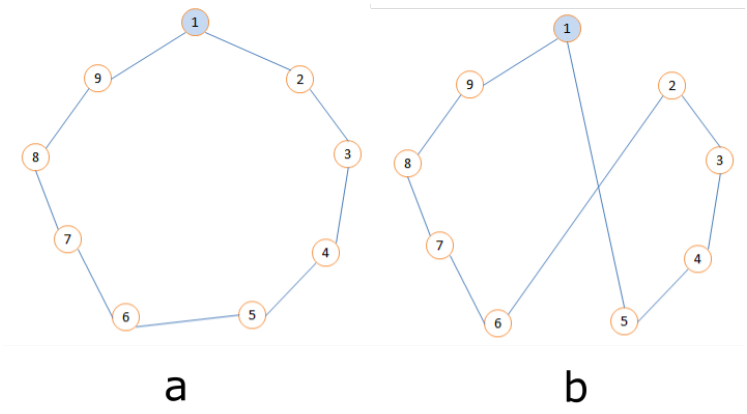
(Ouaarab, Aziz et al, 2014b)

INPUT: fungsi objektif $f(\mathbf{x}), \mathbf{x}_0 = (x_1, \ldots, x_d)^T$
populasi awal dari n sarang $x_i (i = 1, \ldots, n)$, MaksGenerasi, P_a
OUTPUT: **x** terbaik
while ($t \leq$ *MaksGenerasi* **or** *memenuhi kriteria pemberhentian lain*) **do**
  Ambil seekor Cuckoo secara acak dengan persamaan*Levy flight*.
  Hitung fitness f_i dari Cuckoo yang diambil.
  Pilih secara acak satu sarang dari antara n sarang telur (misalkan x_j).
  if $f_i > f_j$ **then**
    Ganti telur x_i dengan telur x_j ($x_i = x_j$),$f_i = f_j$
  end if
  for i=1 **to** n **do**
    Dengan peluang ditemukan p_a, sarang yang buruk dibuang dan dibentuk sarang yang baru.
  end for
  Simpan solusi yang terbaik.
  Urutkan solusi dan temukan solusi terbaik.
  t = t+1
end while
Hasil dan visualisasi

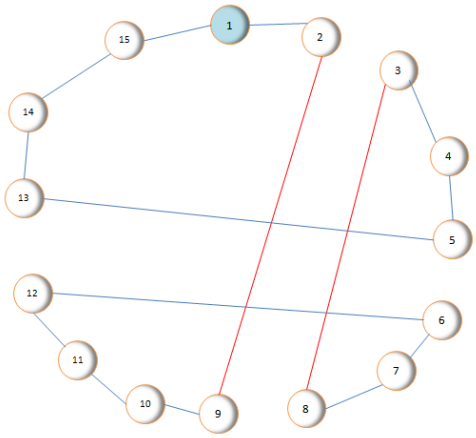
Setiap telur Cuckoo yang ada pada sarang menyatakan sebuah solusi atau calon solusi. Telur atau sarang berpeluang untuk diganti/dibuang atau lolos untuk menjadi generasi selanjutnya. Untuk mendapatkan calon solusi baru (Cuckoo) dari $t + 1$, yaitu $\mathbf{x}_i^{t+1}$, untuk seekor burung Cuckoo i, digunakan persamaan*Levy flight*:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \oplus \mathbf{L} \tag{2.11}$$

yang mana **L** adalah bilangan real berdistribusi Levy(s, λ) dan elemen lainnya adalah nol. $\alpha > 0$ menyatakan *step size*. Untuk Pseudo-code dari algoritma CS, dapat dilihat di Algorithm 1. *Levy flight* lebih efisien dari pada gerak



Gambar 2.5: 2-opt. (**a**) Tur awal sebelum dikenakan operator. (**b**) Tur baru yang dihasilkan dari operator 2-opt [*edge* (a,b) dan (c,d) dibuang, kemudian diganti dengan *edge* (a,c) dan (b,d)]



Gambar 2.6: Sirkuit dengan 15 kota yang sudah dikenakan double-bridge.

acak Brownian dalam mengeksplorasi ruang pencarian yang berukuran sangat besar. Salah satunya bisa diamati dari variansi dari*Levy flight* yaitu

$$\sigma^2(t) \sim t^{3-\beta}, 1 \leq \beta \leq 2$$

yang meningkat lebih cepat dibandingkan dengan variansi dari gerak acak Brownian ($\sigma^2(t) \sim t$).

2.4.1 Analisis Konvergensi Algoritma Cuckoo Search

Wang dkk. membuktikan secara matematis konvergensi global (*global convergence*) dari Cuckoo Search dengan pendekatan teori Rantai Markov (Wang, F et al, 2012). Secara ringkas bukti matematisnya adalah sebagai berikut:

Untuk memudahkan analisis Wang dkk. menyederhanakan algoritma Cuckoo Search pada pembaruan solusi. Mereka menggunakan bilangan acak $r \in [0,1]$ yang dibandingkan dengan parameter penemuan oleh *host* p_a sehingga didapat

$$\begin{cases} \mathbf{x}_i^{t+1} = \mathbf{x}^t & r < p_a \\ \mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \otimes L(s,\lambda) & r > p_a \end{cases}$$

Karena Cuckoo Search adalah algoritma yang bersifat stokastik, maka algoritma itu dapat disederhanakan dengan langkah sebagai berikut:

1. Bangkitkan secara acak populasi awal dari n sarang, $\mathbf{X} = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \mathbf{x}_3^0, ..., \mathbf{x}_n^0\}$, kemudian evaluasi fitnessnya untuk mendapatkan solusi global terbaik sementara $\mathbf{g}_t^0$.

2. Perbarui solusi dengan

$$\mathbf{x}_i^{t+1} = \mathbf{x}^t + \alpha \otimes L(\lambda).$$

3. Bangkitkan bilangan acak r yang berdistribusi seragam $[0,1]$. Jika $r > p_a$, *up date* $\mathbf{x}_i^{t+1}$. Kemudian evaluasi solusi baru untuk mendapatkan solusi terbaik global yang baru $\mathbf{g}_t^*$.

4. Jika kondisi pemberhentian terpenuhi, maka $\mathbf{g}_t^*$ adalah solusi terbaik global yang ditemukan. Jika tidak, kembali ke (2).

Jika f terukur dan ruang solusi *feasible* Ω adalah subset dari $\mathcal{R}$, algoritma A memenuhi dua kondisi dengan barisan pencarian $\{\mathbf{x}_k\}_{k=0}^{\infty}$ maka

$$\lim_{k\to\infty} P(\mathbf{x}_k \in R_{\epsilon,M}) = 1.$$

Dapat dikatakan bahwa algoritma A konvergen secara global. $P(\mathbf{x}_k \in R_{\epsilon,M})$ adalah ukuran peluang dari solusi ke-k pada $R_{\epsilon,M}$ pada iterasi ke-k.

Sarang dan solusi global terbaiknya $\mathbf{g}$ pada catatan pencarian (*search history*) membentuk keadaan Cuckoo: $\mathbf{y} = (\mathbf{x}, \mathbf{g})$ dengan $\mathbf{x}, \mathbf{g} \in \Omega$ dan $f(\mathbf{g}) \leq f(\mathbf{x})$. Himpunan dari semua kemungkinan keadaan membentuk ruang keadaan yang dinyatakan oleh

$$\mathbf{Y} = \{\mathbf{y} = (\mathbf{x}, \mathbf{g}) | \mathbf{x}, \mathbf{g} \in \Omega, f(\mathbf{g}) \leq f(\mathbf{x})\}. \tag{2.12}$$

Keadaan dari seluruh n sarang membentuk keadaan dari grup yang dinyatakan oleh $\mathbf{q} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_n)$. Semua keadaan dari Cuckoo membentuk sebuah ruang keadaan untuk grup tersebut yang dinyatakan oleh

$$\mathbf{Q} = \{\mathbf{q} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_n), \mathbf{y}_i \in \mathbf{Y}, 1 \leq i \leq n\}.$$

Jelas bahwa $\mathbf{Q}$ memuat catatan solusi terbaik global $\mathbf{g}^*$ untuk seluruh populasi dan semua solusi terbaik dari individu $\mathbf{g}_i (1 \leq i \leq n)$ di catatan tersebut. Lalu, solusi terbaik global dari seluruh populasi adalah yang terbaik di antara semua $\mathbf{g}_i$, sehingga $f(\mathbf{g}^*) = \min(f(\mathbf{g}^*)), 1 \leq i \leq n$.

Peluang transisi dari keadaan $\mathbf{y}_1$ ke $\mathbf{y}_2$ dari Cuckoo adalah

$$P(T_y(\mathbf{y}_1) = \mathbf{y}_2) = P(\mathbf{x}_1 \to \mathbf{x}_1')P(\mathbf{g}_1 \to \mathbf{g}_1')P(\mathbf{x}_1' \to \mathbf{x}_2)P(\mathbf{g}_1' \to \mathbf{g}_2) \quad (2.13)$$

dengan $P(\mathbf{x}_1 \to \mathbf{x}_1')$ adalah peluang transisi pada langkah ke-2 pada penyederhanaan sebelumnya, dan $P(\mathbf{g}_1 \to \mathbf{g}_1')$ adalah peluang transisi untuk catatan terbaik global pada langkah ke-2. $P(\mathbf{x}_1' \to \mathbf{x}_2)$ adalah peluang transisi pada langkah ke-3, sementara $P(\mathbf{g}_1' \to \mathbf{g}_2)$ adalah peluang transisi catatan terbaik global untuk langkah ke-3.

Untuk solusi optimal secara global $\mathbf{g}_b$ untuk sebuah permasalahan optimasi $< \Omega, f >$, himpunan keadaan optimal didefinisikan sebagai $\mathbf{R} = \{\mathbf{y} = (\mathbf{x}, \mathbf{g}) | f(\mathbf{g}) = f(\mathbf{g}_b), \mathbf{y} \in \mathbf{Y}\}$.

Untuk solusi optimal secara global $\mathbf{g}_b$ untuk sebuah permasalahan optimasi $< \Omega, f >$, himpunan keadaan grup optimal didefinisikan sebagai $\mathbf{H} = \{\mathbf{q} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_n) | \exists \mathbf{y}_i \in \mathbf{R}, 1 \leq i \leq n\}$.

Analisis matematis lanjut yang lebih detil membuktikan bahwa ketika banyak iterasi cukup besar, barisan keadaan grup akan konvergen ke himpunan solusi yang optimal $\mathbf{H}$. Oleh karena itu, Cuckoo Search menjamin konvergen secara global (X.-S. Yang, 2014).

2.4.2 Perbedaan Dengan Beberapa Algoritma Metaheuristik yang Lain

Perbedaan antara Cuckoo Search dengan algoritma berbasis populasi seperti GA salah satunya pada proses pembentukan solusi atau individu baru. Pada GA, terdapat dua operasi yang dilakukan, yaitu operasi mutasi dan pindah silang. Mutasi hanya melibatkan satu kromosom, sedangakan pindah silang membutuhkan dua kromosom untuk mendapatakan sebuah solusi atau kromosom yang baru. Berbeda dengan CS, yang hanya melibatkan

Tabel 2.1: Perbandingan CS, GA, dan PSO

Parameter	**CS**	**GA**	**PSO**
populasi	kumpulan telur	kumpulan kromosom	*swarm*
pindah silang	tidak ada	ada	tidak ada
mutasi	mirip pada proses penggantian telur	ada	tidak ada
kecepatan individu	tidak ada	ada	ada
posisi individu	tidak ada	tidak ada	ada
generasi	ada	ada	tidak ada
pembentukan solusi baru	dengan pergerakan Levy dan melibatkan individu dalam satu iterasi	melibatkan sejumlah individu hasil pindah silang dan mutasi	berdasarkan posisi dan kecepatan individu

sebuah individu dan satu individu pembanding dalam melakukan operasi pembentukan solusi baru. Selain itu GA bekerja pada semua solusi secara bersamaan atau simultan, sedangkan CS hanya mengerjakan pada individu atau sebuah telur yang terpilih. PSO dan CS sekilas mirip. Karena sama-sama mengggunakan konsep pergerakan, tetapi pada CS tidak dikenal konsep kecepatan seperti pada PSO. Pada PSO, jika sebuah solusi sudah diduga optimal, maka pergerkan individunya akan melambat, agar tidak bergerak jauh dari solusi yang optimal. Sedangkan CS akan mengekplorasi ruang pencarian untuk mendapatkan sarang baru atau individu baru. Bergerak secara acak, sehingga bisa saja solusi yang awalnya diduga optimal, ternyata tidak terpilih akibat hasil ekplorasi yang mendapatkan solusi yang lebih optimal. Tabel 2.1 menunjukkan ringkasan dari perbandingan CS, GA, dan PSO.

2.4.3 Discrete Cuckoo Search

Travelling Salesman Problem adalah permasalahan yang bersifat diskrit. Permasalahan ini akan sulit diselesaikan dengan menggunakan algoritma Cuckoo Search biasa, terutama ketika menggunakan persamaan*Levy flight*. Yang membedakan algoritma Cuckoo Search dengan algoritma Discrete Cuckoo salah satunya adalah pada penggunaan*Levy flight* dalam pencarian atau pembentukan solusi yang baru terutama dalam menentukan ukuran dari *step* (*step size*). Aziz (Ouaarab, Aziz et al, 2014b) mendefinisikan ulang*Levy flight* (2.11) tersebut dengan menggunakan operator 2-opt dan double-bridge.

Jika nilai yang didapat dari distribusi Levy, **L** berada pada:

salesman	1		2		3	
telur (rute)	4	5	3	7	6	2

Gambar 2.7: Contoh representasi sebuah telur Cuckoo untuk salah satu solusi dari MTSP

1. $[0, i)$; lakukan sekali operasi 2-opt ($\mathbf{x}_i^{t+1}$ = 2-opt ($\mathbf{x}_i^t$))
2. $[(k-1)\times i, k\times i)$; lakukan k kali operasi 2-opt ($\mathbf{x}_i^{t+1}$ = 2-opt($\mathbf{x}_i^t$) sebanyak k kali)
3. $[k\times i, 1]$; lakukan sekali operasi double-bridge ($\mathbf{x}_i^{t+1}$ = double-bridge($\mathbf{x}_i^t$))

dengan $i = (1/(n+1))$, n adalah jumlah maksimum operasi (*step*) yang dilakukan dan $k \in (2, 3, 4, ..., n)$.

Operator 2-opt akan membentuk solusi baru dengan cara membuang dua *edge* dari sebuah rute atau *tour* kemudian menghubungkan titik dari kedua *edge* itu sehingga terbentuk *edge* baru. Sedangkan operator double bridge; seperti operator 2-opt; membedakan adalah double-bridge mengambil empat *edge* dan membentuk empat *edge* yang baru. Setiap telur pada Discrete Cuckoo Search menyatakan individu dari populasi dan juga merepresentasikan solusi atau kandidat solusi dari suatu permasalahan.

Dalam kasus MTSP, sebuah telur adalah solusi yang terdiri atas rute dari setiap *salesman*. Sebuah sarang beserta telur juga menyatakan sebuah solusi. Dalam hal ini, setiap sarang hanya diisi oleh satu telur Cuckoo saja. Sarang yang dimaksud adalah sarang dari burung spesies lain (bukan dari spesies Cuckoo). Sehingga ada kemungkinan untuk telur dari Cuckoo ditemukan oleh burung dari pemilik sarang (*host*) yang dinyatakan dengan parameter p_a. Jika pemilik sarang berhasil menemukan atau menyadari ada telur Cuckoo pada sarangnya, *host* akan membuang/merusak telur atau membuang sarang dan membentuk sarang baru. Dengan kata lain, proses ini akan menghasilkan telur (solusi) baru dan berpotensi lebih baik untuk menggantikan telur yang tidak terlalu baik dari sebuah sarang.

Kualitas dari telur atau sarang dapat dilihat dari nilai fungsi objektifnya. Untuk masalah minimisasi seperti pada MTSP maka telur yang terbaik adalah telur yang memiliki total rute yang terpendek.

Gambar 2.5 menunjukkan contoh hasil dari operator 2-opt. Untuk double-bridge, ditunjukkan oleh Gambar 2.6.

2.4.4 Distribusi Levy

Secara matematis, distribusi Levy yang sederhana dapat dinyatakan sebegai berikut:

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}}\exp(-\frac{\gamma}{2(s-\mu)})\frac{1}{(s-\mu)^{3/2}} & 0 < \mu < s < \infty \\ 0 & \text{untuk yang lain} \end{cases} \tag{2.14}$$

dengan $\mu > 0$ adalah step minimum dan γ adalah sebuah parameter skala. Jelas jika $s \to \infty$, didapat

$$L(s,\gamma,\mu) \approx \sqrt{\frac{\gamma}{2\pi}\frac{1}{s^{3/2}}}. \tag{2.15}$$

Ini adalah kasus khusus dari distribusi Levy yang diperumum.

Bilangan acak berdistribusi Levy dibangkitkan dengan menggunakan algoritma Mantegna. Bilangan acak ini nantinya digunakan untuk menentukan banyaknya langkah operator dari 2-opt dan double-bridge.

Bilangan acak s didapat dengan algoritma Mantegna:

$$s = \frac{u}{|v|^{\frac{1}{\beta}}} \tag{2.16}$$

dengan u dan v didapat dari distribusi normal, yaitu $u \sim N(0,\sigma_u^2)$ dan $v \sim N(0,\sigma_v^2)$ dan

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2})2^{\frac{\beta-1}{2}}\beta} \right\}^{\frac{1}{\beta}} \tag{2.17}$$

dan $\sigma_v = 1$. Nilai β yang digunakan adalah $\beta = \frac{3}{2}$ dan $\Gamma(z)$ menyatakan fungsi gamma yang dinyatakan oleh:

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx \tag{2.18}$$

untuk $z \in Z$(X.-S. Yang, 2010).

2.5 k-means++

Algoritma k-means++ merupakan pengembangan dari algoritma k-means biasa. Salah satu perbedaan dari algoritma k-means biasa adalah pada pemilihan centroid awal. k-means biasa akan memilih secara acak k centroid

awal yang akan digunakan. Sedangkan pada k-means++ dipilih sebuah centroid awal, kemudian untuk $k-1$ centroid yang lain dipilih dengan peluang yang disebut D^2 weighting. D^2 adalah jarak terpendek dari sebuah titik terhadap centroid terdekat yang sudah dipilih. Misalkan diberikan $X = 0, 1, 2, 3, 4$, maka $D^2(x = 1)$ untuk centroid $c = 2$ adalah $D^2(x = 1) = ||1 - 2||^2$. Probabilitas ini diukur dengan melihat perbandingan jarak minimum titik dengan centroid dikuadratkan dibanding total jarak minimum titik dengan keseluruhan centroid. Setelah mendapatkan k centroid, maka langkah selanjutnya adalah sama seperti pada algoritma k-means biasa.

Sebuah pengukuran jarak digunakan untuk menghitung nilai ketidaksamaan di antara dua titik. Pemilihan pengukuran ini penting dalam analisis *clustering*. Misalkan diberikan sebuah himpunan X yang terdiri atas n titik yaitu $X = x_1, x_2, x_3, ..., x_n$. jika d adalah dimensi dari data, maka $x_{pq}, p = 1, 2, ..., n, q = 1, 2, ..., d$ menyatakan nilai dari titik ke-p dimensi ke-q. Jarak dua titik x_i dan x_j dinyatakan dengan $D(x_i, x_j)$ dan memenuhi sifat-sifat berikut:

1. $D(x_i, x_j) \geq 0$ untuk semua $x_i, x_j \in X$ dan $D(x_i, x_j) = 0$ jika $i = j$.

2. $D(x_i, x_j) = D(x_j, x_i)$ untuk semua $x_i, x_j \in X$

3. $D(x_i, x_j) \leq D(x_i, x_l) + D(x_l, x_j)$ untuk $x_i, x_j, x_l \in X$

Jarak Euclid digunakan untuk pengukuran (*Euclidean Distance*). Jarak Euclid $D(x_i, x_j)$ di antara dua vektor x_i dan x_j dinyatakan dengan:

$$D(x_i, x_j) = \sqrt{\sum_{l=1}^{d} (x_{il} - x_{jl})^2} \tag{2.19}$$

k-means++ bekerja dengan langkah sebagai berikut:

1. bentuk sebuah himpunan kosong C untuk menyimpan k centroid yang terpilih

2. pilih centroid awal c_1 dari himpunan X dan simpan di C

3. Untuk setiap x_i yang tidak berada di C, cari jarak kuadrat terkecil $D(x_i, C)$ untuk setiap centroid di C.

4. untuk mendapatkan centroid selanjutnya yaitu c_p, maka pilih $x \in X$ dengan menggunakan peluang terbobot yaitu

$$P = \frac{D(c_p, C)^2}{\sum_{x_i} D(x_i, C)^2} \tag{2.20}$$

untuk memilih x_p dari X.

5. ulangi langkah 3-4 hingga didapat k centroid awal.

6. pasangkan titik $x_i, i = 1, 2, ..., n$ pada kluster $M_j, j \in 1, 2, 3, ..., k$ jika dan hanya jika $D(c_j, x_i) < D(c_p, x_i), p = 1, 2, .., k$ dan $j \neq p$.

7. centroid baru dari untuk setiap kluster yaitu $c_1^t, c_2^t, ..., c_k^t$ dihitung dengan rumus

$$c_i^t = \frac{1}{n_i} \sum_{x \in M_j} x_j \tag{2.21}$$

$i = 1, 2, ..., k, t = 1, 2, 3, ..., l$ dengan n_i adalah banyak elemen yang ada di kluster M_i.

8. jika $c_i^t = c^t + 1_i$ untuk $i = 1, 2, ..., k$ atau $t = l$;berhenti. Jika tidak ulang 6-7.

BAB 3
METODE PENELITIAN

Dalam bab ini dibahas mengenai tahapan penelitian tentang "Penerepan Discrete Cuckoo Search Pada Multiple Travelling Salesman Problem".

3.1 Tahapan Penelitian

Tahapan yang dilakukan dalam penelitian adalah:

1. Studi literatur
   Pada bagian ini dilakukan kajian literatur untuk mendalami lagi Travelling Salesman Problem, *Multiple Travelling Salesman Problem* (MTSP), k-means++, Cuckoo Search, dan Discrete Cuckoo Search (DCS). Sumber literatur dapat berupa buku, jurnal, artikel-artikel dari internet atau saran/masukan dari dosen pembimbing.

2. Analisis dan pembahasan algoritma DCS dan k-means++.
   Pada bagian ini dianalisis dan dibahas bagaimana algoritma DCS bekerja pada persoalan Travelling Salesman Problem dan MTSP.

3. Penerapan DCS dan k-means++ pada MTSP
   Pada bagian ini algoritma DCS dan k-means yang telah dikaji; diterapkan pada MTSP. Sebelum melakukan DCS pada MTSP, kota yang diberikan akan terlebih dahulu dikluster dengan menggunakan k-means++. Jumlah kluster menunjukan jumlah salesman yang digunakan.

   Kemudian akan ditentukan bentuk representasi solusi dan fungsi objektif yang digunakan serta pseudo-code dari algortima DCS untuk menyelesaikan MTSP. Untuk mempermudah penerapan Discrete Cuckoo Search maka, perlu penjelasan dari beberapa istilah pada Discrete Cuckoo Search seperti, telur, sarang, fungsi objektif, ruang pencarian, dan *Levy Flight*:

   (a) Telur
   Telur adalah solusi atau kandidat solusi dari MTSP yang dapat

dinyatakan sebagai permutasi bilangan.

(b) Sarang

Sarang beserta telur adalah kandidat solusi. Sehingga sebuah telur dapat menjadi solusi bergantung pada sarang. Jika sebuah sarang rusak atau *host* menyadari terdapat telur Cuckoo pada sarangnya maka telur itu akan dirusak atau dibuang. Dengan kalimat lain, solusi yang ada pada sarang itu tidak digunakan. Sebuah telur Cuckoo akan ditemukan *host* dengan probabilitas p_a.

(c) Fungsi objektif

Untuk menentukan kualitas dari telur maka diperhatikan nilai dari fungsi objektif yang dihasilkan. Dalam kasus ini, telur (solusi) yang menghasilkan rute terpendek adalah telur dengan kualitas terbaik.

(d) Ruang pencarian

Karena koordinat kota dalam wilayah tetap, maka pergerakan dalam ruang pencarian dapat dinyatakan sebagai urutan dari kota yang dikunjungi. Untuk mendapatkan solusi baru pada ruang pencarian dengan merubah urutan dari kota yang dikunjungi, maka digunakan operator 2-opt dan double-bridge *move* .

Operator 2-opt digunakan untuk menghasilkan perubahan yang tidak terlalu jauh dari solusi yang didapat sebelumnya. Sedangkan double-bridge *move* digunakan untuk membentuk solusi yang berbeda cukup signifikan dengan solusi yang didapat sebelumnya.

(e) *Levy flight*

Levy flight digunakan untuk menentukan penggunaan operator 2-opt dan double-bridge.

4. Simulasi solusi dari penerapan algoritma DCS pada persoalan MTSP.

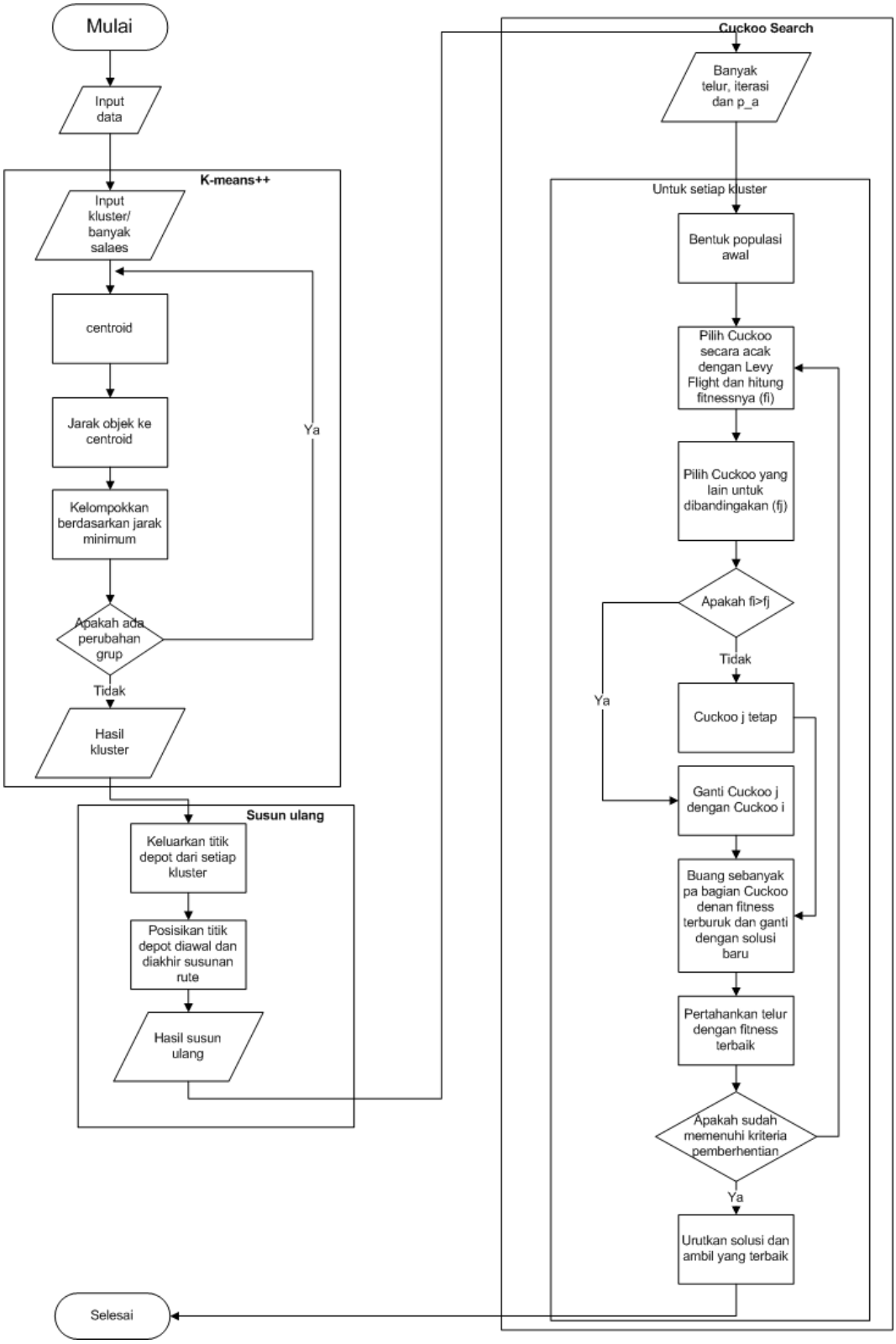
   Pada bagian ini dilakukan simulasi dari solusi yang diperoleh dari penerapan algoritma pada persoalan MTSP dengan menggunakan bahasa Python dan data untuk simulasi diperoleh dari data yang tersedia secara on-line yaitu TSPLIB dari http://elib.zib.de/pub/mp-testdata/tsp/.

5. Analisis dan pembahasan simulasi

   Pada bagian ini dianalisis dan dibahas bagaimana hasil dari simulasi algoritma DCS pada persoalan Travelling Salesman problem dan MTSP dengan cara melakukan percobaan sebanyak 30 kali. Dari hasil

percobaan tersebut akan diamati rerata jarak, jarak terpendek, deviasi standar, koefisisen variasi, dan rerata waktu komputasi.

Untuk memvalidasi solusi yang didapatkan, maka akan digunakan data dan hasil penelitian dari Majd Latah (Latah, Majd, 2016). Pada penelitian tersebut digunakan 5 data dan jumlah *salesman* yakni 3.



Gambar 3.1: Diagram blok k-means++ dan Cuckoo Search

BAB 4
HASIL DAN PEMBAHASAN

Untuk menyelesaikan permasalahan MTSP, pada penelitian ini digunakan metode klustering dan Cuckoo Search. Secara garis besar, permasalahan MTSP diselesaikan dengan terlebih dahulu melakukan *clustering* pada data dengan menggunakan k-means++ untuk mendapatkan kota yang akan membentuk rute dari *salesman*, kemudian dari hasil klustering tersebut akan dicari rute terpendek dengan menggunakan Cuckoo Search. Telur pada algoritma Cuckoo Search adalah permutasi dari kota yang didapat dari hasil *clustering*.

Salah satu yang diamati dari Cuckoo Search pada MTSP adalah pada bagaimana merepresentasikan dan membangkitan solusi. Jelas bahwa domain solusi dari MTSP adalah semua permutasi dari kota yang diberikan. Pembangkitan solusi pada Cuckoo Search menggunakan *Levy Flight*. Namun untuk kasus diskrit seperti MTSP, perlu pendefinisian ulang dari persamaan tersebut. Pada penelitian ini digunakan operator 2-opt dan double-bridge untuk pendefinisian ulang dari *Levy Flight*. Kedua operator ini digunakan untuk membentuk permutasi baru.

4.1 Clustering Data

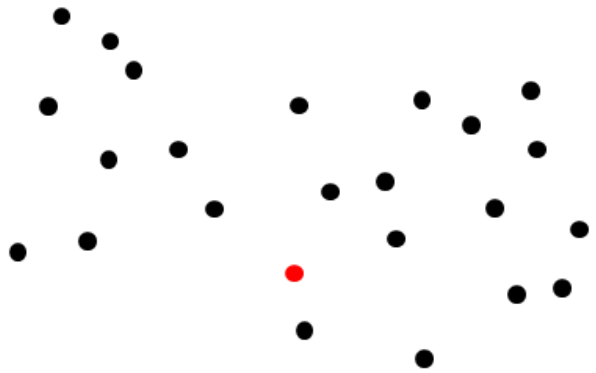
4.1.1 Jumlah Maksimum Salesman

Jumlah kluster sama dengan banyak *salesman* yang digunakan. Misalkan banyak kota yang diberikan adalah sebanyak r dan *salesman*, maka dugaan awal maksimum jumlah *salesman* sehingga mendapat kota yang berbeda-beda adalah r. Nantinya setiap *salesman* mendapatkan kota masing-masing sebanyak satu. Namun, jika diambil salah satu titik dari r kota sebagai titik depot maka akan ada satu *salesman* yang berangkat hanya dari satu kota saja. Misalkan titik itu adalah titik p. Maka *salesman* yang mendapat titik atau kota p akan berangkat dari titik p ke titik p saja. Karena setiap *salesman* mendapat satu kota, maka *salesman* yang mendapat kota p hanya akan berangkat dari satu titik, tidak seperti $r-1$ *salesman* yang lain. Maka dari itu *salesman* yang digunakan haruslah kurang 1 dari yang sebelumnya atau ada sebanyak $r-1$.

Hal ini memastikan bahwa *salesman* mendapat kota yang berbeda tetapi tetap menyediakan satu kota yang menjadi titik depot. Sehingga banyak *salesman* maksimum yang digunakan jika diberikan sebanyak r kota adalah sebanyak $r - 1$ *salesman* sehingga tetap memenuhi kendala (2.5)-(2.10).

4.1.2 Proses Klustering Dengan k-means++

Pertama-tama dilakukan *clustering* pada data yang diberikan. Tujuan dari proses *clustering* ini adalah untuk mendapatkan kota yang akan dikunjungi oleh setiap *salesman* yang nantinya membentuk rute. Teknik *clustering* yang digunakan adalah k-means++. Hasil dari klustering ini membuat bahwa setiap kota yang dikunjungi oleh setiap *salesman*, berbeda karena *clustering* dengan menggunakan k-means++ tidak bersifat *overlapping*. Sebagai contoh;

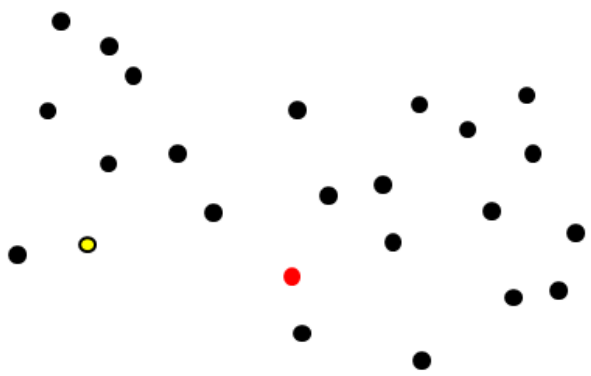


Gambar 4.1: 24 titik yang belum dikluster.

misalkan diberi 24 kota (titik) ditunjukkan pada Gambar 4.1 dengan titik merah merupakan kota depot. Akan dilakukan proses klustering dengan menggunakan nilai $k = 3$.

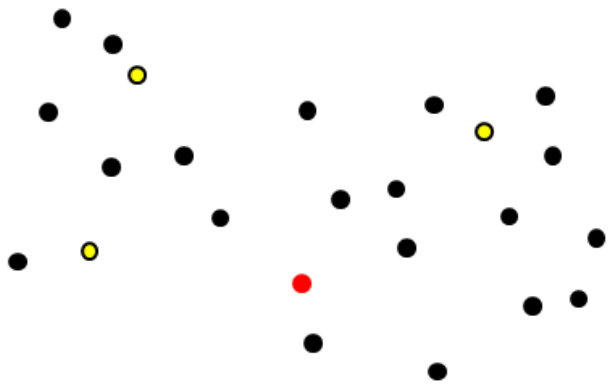
Pertama-tama pilih sebuah titik centroid awal secara acak pada Gambar 4.2 ditunjukkan oleh titik lingkaran kuning dengan garis tepi hitam. Pada k-means++ proses pemilihan centroid yang lain dengan menggunakan D^2 weighting. Lalu, 2 centroid awal yang lain dipilih dengan mengunakan peluang berdasarkan D^2 weighting. Sehingga didapat 3 centroid awal yang ditandai juga dengan lingkaran kuning garis tepi hitam pada Gambar 4.3.

Kemudian setelah mendapatkan 3 centroid awal, setiap data akan dihitung jaraknya dengan setiap centroid yang didapat tersebut. Proses kluster awal berlangsung di tahap ini. Pemilihan anggota kluster adalah berdasarkan kedekatan dengan centroid yang diberikan. Selanjutnya masuk ke tahap



Gambar 4.2: Titik kuning dengan tepian hitam terpilih sebagai centroid awal.

pemilihan centroid yang baru. Setelah mendapatkan anggota pada masing-masing kluster, dengan menggunakan (2.21) dicari bobot rata-rata yang nantinya menjadi titik centroid baru. Algoritma k-means++ dijalankan



Gambar 4.3: Dua centroid awal lain terpilih.

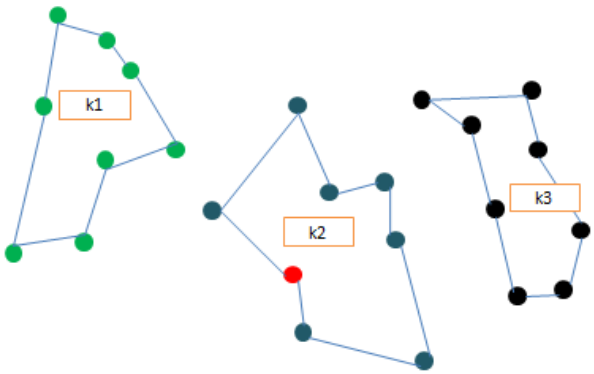
dengan sejumlah iterasi untuk mendapatkan nilai SSE (*sum of squared errors*) yang paling minimum. SSE digunakan untuk menilai kriteria kluster yang dihasilkan. Semakin kecil nilai SSE yang dihasilkan maka semakin baik kluster yang diperoleh. Nilai SSE dapat diperoleh dengan rumus

$$SSE = \sum_{k=1}^{K} \sum_{\forall x_j \in C_i} ||x_j - m_i||^2 \tag{4.1}$$

dengan K banyak kluster, C_i adalah himpunan data yang berada pada kluster i, dan m_i adalah centroid kluster i.

Selanjutnya didapat hasil kluster 24 titik yang diberikan seperti yang tampak pada Gambar 4.4. Masing-masing kluster memiliki 8 titik. Titik merah masuk dalam salah satu kluster yaitu kluster k2.

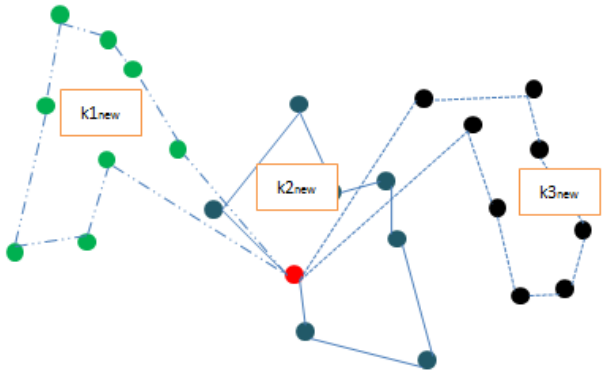
Hasil kluster dengan metode ini, masih menempatkan titik merah atau titik depot pada salah satu kluster. Seperti yang diketahui, bahwa titik depot adalah titik yang digunakan oleh semua *salesman* pada permasalahan MTSP. Maka nantinya titik depot akan ditambahkan pada kluster lain agar rute yang terbentuk sesuai seperti yang ditunjukkan oleh Gambar 4.5. Pada gambar tersebut terdapat 3 kluster, yang setiap klusternya memnuat titik merah. Hal ini ditandai dengan garis yang terhubung dari dan menuju titik merah untuk setiap kluster.



Gambar 4.4: Hasil kluster 24 titik.

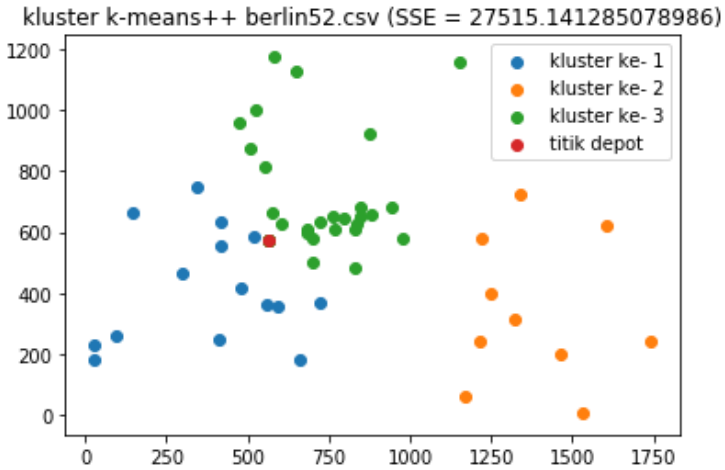
Gambar 4.6 menunjukkan hasil proses *clustering* dari data berlin52.csv. Pada data ini digunakan kota pertama (1) dengan koordinat (565.0, 575.0) sebagai kota depot, dengan banyak *salesman* yang digunakan sebanyak $m = 3$. Sehingga nilai k yang digunakan dalam proses *clustering* dengan k-means adalah $k = 3$. Dengan menggunakan parameter tersebut didapat nilai dari SSE adalah 27.515,1413.

Adapun kluster yang dihasilkan dengan menggunakan data berlin52.csv: kluster 1 = {1, 2, 3, 7, 16, 17, 18, 20, 21, 22, 23, 29, 30, 31, 42, 50} ada sebanyak 16 kota. Elemen yang terdapat dalam himpunan kluster 1 menunjukkan urutan kota yang berada pada daftar berlin.csv. Sebagai contoh, 1 artinya kota pertama pada daftar tersebut. Data kota dari berlin.xlsx, dapat dihimpun menjadi $S = \{n | n \leq 52, n \in \mathbb{N}^+\}$ Pada kluster lain yaitu kluster 2 = {11, 12, 13, 14, 26, 27, 28, 47, 51, 52} sebanyak 10 kota, dan kluster 3 = {4, 5, 6, 8, 9, 10, 15, 19, 24, 25, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 48,



Gambar 4.5: Titik merah yang disisipkan pada kluster lain.

49} terdiri atas 26 kota. Untuk sementara didapat bahwa kota 1 sebagai titik



Gambar 4.6: Hasil klustering dengan k=3.

depot terdapat di salah satu kluster, yaitu kluster 1.

4.2 Cuckoo Search Untuk Hasil Clustering

Setelah proses *clustering* berhasil dilakukan dengan menggunakan k-means++, maka selanjutnya setiap kluster akan dikenakan Cuckoo Search untuk mendapatkan rute terpendek dari kota yang akan dikunjungi. Pencarian rute terpendek dari dari setiap kluster, mirip dengan persoalan pada TSP. Sehingga persoalan MTSP pada penelitian ini adalah persoalan TSP, yang terdiri atas sebanyak jumlah *salesman* yang digunakan. Antar satu kluster dengan kluster yang lain tidak berpengaruh. Kesamaan di antara kluster hanyalah satu titik depot.

Salah satu yang diamati dari penelitian ini adalah bagaimana menggunakan algoritma Cuckoo Search yang sebenarnya bekerja pada ruang pencarian kontinu ke permasalahan kombinatorik seperti pada TSP. Maka dari itu, notasi telur, sarang, fungsi objektif, ruang pencarian, dan *Levy flight* akan dijelaskan di subbab ini.

4.2.1 Pembentukan Solusi

Dalam Cuckoo Search, sebuah telur dalam sarang adalah sebuah solusi yang direpresentasikan oleh sebuah sarang/individu pada populasi. Jumlah sarang ada sebanyak jumlah populasi Cuckoo yang digunakan. Telur dengan kualitas terbaik akan menjadi individu baru. Kemudian individu itu akan menghasilkan telur baru atau solusi baru. Dan jika nanti terdapat sarang yang menghasilkan telur dengan kualitas yang buruk, telur pada sarang itu tidak akan menjadi individu yang baru. Sarang tersebut akan diganti dengan sarang yang berisi telur baru.

Setiap telur menunjukkan rute atau sikel Hamiltonian. Sebuah telur juga dapat merupakan sebuah calon solusi baru dari sarang yang dibuat oleh seekor Cuckoo dalam populasi atau dengan kata lain sikel hamiltonian yang baru.

Setiap telur yang dihasilkan mengandung kota apa saja yang akan dikunjungi oleh setiap *salesman*. Setiap telur merupakan permutasi dari kota yang didapat dari masing-masing kluster.

Rute dibentuk dengan menambahkan dan mengatur ulang kota 1 pada masing-masing kluster. Hal ini dilakukan karena kota depot hanya berada pada salah satu kluster saja. Pada contoh sebelumnya titik depot hanya berada pada kluster 1 saja.

kota 1 akan ditambahkan pada kluster yang sebelumnya tidak terdapat kota 1 atau kota depot. Sehingga nantinya kluster 1 = {**1**, 2, 3, 7, 16, 17, 18, 20, 21, 22, 23, 29, 30, 31, 42, 50}, kluster 2 = {**1**, 11, 12, 13, 14, 26, 27, 28, 47, 51, 52}, dan kluster 3 = {**1**, 4, 5, 6, 8, 9, 10, 15, 19, 24, 25, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 48, 49}.

Rute masing-masing *salesman* didapat dari permutasi kota dari setiap kluster. Misal, rute untuk *salesman* pertama adalah permutasi dari kluster 1. Salah satu rute atau telur yang mungkin dari *salesman* pertama adalah

$$r_1 = (1, 3, 2, 7, 16, 17, 18, 20, 21, 22, 23, 29, 30, 31, 42, 50) \tag{4.2}$$

yang merupakan sebuah sikel dengan panjang 16. Sedangkan untuk *salesman*

2, rute yang mungkin adalah

$$r_2 = (1, 11, 14, 13, 12, 26, 27, 28, 47, 51, 52)$$

yang adalah sikel dengan panjang 11. Cara yang juga berlaku pada rute dengan menggunakan kluster 3.

Dalam Cuckoo Search, diperhatikan pembentukan calon solusi (telur) yang akan digunakan. Misalkan dari hasil susun ulang kluster diperoleh sebanyak m kota (koordinat) maka representasi salah satu telur yang digunakan adalah $(c_1, c_2, c_3, ..., c_{m-1}, c_m)$ dengan c_1 adalah kota depot dari setiap rute dan c_i adalah kota yang ada di suatu kluster dengan $i = 2, 3, ..., m$. Sebagai contoh gunakan (4.2). Pada telur r_1, $c_1 = 1, c_2 = 3, c_3 = 2, c_4 = 7, ..., c_{m=16} = 50$.

Untuk melihat kualitas dari telur yang dihasilkan maka algoritma Cuckoo Search akan menilai berdasarkan fungsi objektif. Fungsi objektif yang dimaksud adalah jarak yang dihasilkan dari permutasi kota pada tiap telur. Jarak yang digunakan diukur dengan mengunakan metrik Euklid (Euclidian Metric), yaitu dinyatakan oleh $d(c_p, c_q)$ yaitu

$$d(c_p, c_q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \tag{4.3}$$

dengan $c_p = (x_p, y_p)$ dan $c_q = (x_q, y_q)$.

Sehingga fungsi objektif dari sebuah telur adalah

$$f(e) = \sum_{i=1}^{m-1} d(c_i, c_{i+1}) + d(c_1, c_m). \tag{4.4}$$

Telur dengan kualitas buruk atau memiliki nilai fungsi objektif yang besar tidak akan menjadi individu baru. Sarang yang memuat telur dengan kualitas terburuk tersebut akan diisi dengan telur baru. Dianalogikan ada Cuckoo baru yang menaruh telur pada sarang tersebut.

4.2.1.1 Pembentukan Solusi Awal

Solusi awal yang digunakan untuk setiap kluster adalah kombinasi kota dari masing-masing kluster tersebut yang didapat dengan mempertimbangkan kota depot. Seperti yang dijelaskan sebelumnya,solusi yang dihasilkan harus membuat kota depot berada pada urutan awal dan akhir. Misalkan untuk menyelesaikan permasalahan pada kluster 1, maka sebanyak *pop* solusi awal yang digunakan, diperoleh dari kombinasi kota di K_1 yang sudah disusun ulang. Misal salah satu solusi awal yang dihasilkan $e =$

$(1, 3, 2, 7, 16, 17, 18, 20, 21, 22, 23, 29, 30, 31, 42, 50)$ yang diambil dari (4.2). Telur ini memiliki 16 kota, yang terdiri dari 1 kota depot dan 15 kota lainnya.

Nilai *pop* dipilih dengan memperhatikan banyak kota dari setiap kluster. Jika dari hasil klustering dengan k-means++ dan kemudian dilakukan penyusunan ulang sehingga didapat kluster yang memiliki kota paling sedikit misalkan sebanyak b kota maka nilai *pop* akan berada pada

$$1 < pop \leq \frac{(b-1)!}{2} \tag{4.5}$$

Hal ini dapat dijelaskan dengan menghitung banyak permutasi siklik. Perhatikan bahwa banyak permutasi siklik yang dihasilkan dari sebuah himpunan dengan banyak anggota b adalah $(b-1)!$. Namun makna $(c_1, c_2, c_3, c_4, ..., c_{b-3}, c_{b-2}, c_{b-1}, c_b)$ sama dengan $(c_1, c_b, c_{b-1}, c_{b-2}..., c_4, c_3, c_2)$. Misalkan dari telur $e = (1, 3, 2, 7, 16, 17, 18, 20, 21, 22, 23, 29, 30, 31, 42, 50)$, akan menghasilkan rute dengan jarak yang dengan telur $e_2 = (1, 50, 42, 31, 30, 29, 23, 22, 21, 20, 18, 17, 16, 7, 2, 3)$. Sehingga jumlah permutasi yang dihasilkan nanti dapat disederhanakan menjadi separuhnya yaitu $\frac{(b-1)!}{2}$. Jika nilai *pop* yang digunakan melebihi $\frac{(b-1)!}{2}$, algoritma Cuckoo Search tetap dapat berjalan, tetapi terdapat calon solusi atau telur yang berisi kota sama. Hal ini sebaiknya dihindari agar algoritma bekerja lebih efektif dan mencegah perulangan yang tidak perlu. Sebagai contoh, jika banyak kota adalah 4, maka banyak calon kemungkinan calon solusi yang dihasikan adalah sebanyak $\frac{(4-1)!}{2} = 3$, yaitu $(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4)$.

4.2.2 *Update* Solusi

Untuk mendapatkan telur baru, seekor Cuckoo harus bergerak mencari posisi sarang yang baru. Hal demikan dilakukan dengan menggunakan *Levy Flight*. *Levy Flight* ini dinyatakan dalam persamaan matematika seperti yang ditunjukkan oleh persamaan (2.11). Perhatikan bahwa *Levy Flight* pada (2.11) tidak bisa diterapkan langsung pada permasalahan yang bersifat kombinatorik, maka dari itu perlu pendefinisian ulang persamaan *Levy Flight*. Pendefinisian ini tetap menggunakan bilangan acak berdistribusi Levy.

Pada pendefinisian ini, persamaan *Levy Flight* menggunakan operator 2-opt dan double bridge. Penggunaan kedua operator tersebut juga melibatkan bilangan acak berdistribusi Levy.

Untuk mendapatkan telur baru maka, solusi dikenakan salah satu operator tersebut dengan memperhatikan bilangan acak Levy yang dihasilkan.

Sebelumnya dibuat interval untuk menentukan kategori dan jumlah operasi operator yang digunakan. Jika nilai yang didapat dari distribusi Levy, $\mathbf{L}$ berada pada:

1. $[0, i)$; lakukan sekali 2-opt *move* ($\mathbf{e}_i^{t+1}$ = 2-opt move($\mathbf{e}_i^t$))
2. $[(k-1) \times i, k \times i)$; lakukan k kali 2-opt *move* ($\mathbf{e}_i^{t+1}$ = 2-opt *move*($\mathbf{e}_i^t$) sebanyak k kali)
3. $[k \times i, 1)$; lakukan sekali double-bridge *move* ($\mathbf{e}_i^{t+1}$ = double-bridge *move*($\mathbf{e}_i^t$))

dengan $i = (1/(n+1))$, n adalah jumlah maksimum operasi (*step*) yang dilakukan dan $k \in (2, 3, 4, ..., n)$. Pemilihan nilai n ditentukan berdasarkan jumlah kota yang ada pada telur. Misalkan pada satu telur terdapat 6 kota, maka nilai n yang digunakan adalah $\frac{6(6-3)}{2} = 9$. Nilai tersebut adalah banyak kombinasi pemilihan kota pada rute yang menyebabkan terjadinya persilangan pada sirkuit dengan operator 2-opt yaitu, $\frac{b(b-3)}{2}$, sehingga didapat

$$n = \frac{b(b-3)}{2}. \tag{4.6}$$

Rumus ini didapat dengan langkah sebagai berikut:

1. kota pertama yang dipilih adalah acak dan ada sebanyak b cara.
2. kemudian kota kedua; pilihan yang tersedia mula-mula $b-1$ sebab kota pertama tidak dipilih lagi. Kemudian, syarat lain kota yang dipilih haruslah berjarak minimal satu titik sebelum atau sesudah kota pertama yang terpilih. Hal ini menyebabkan banyak pilihan kota kedua berkurang menjadi $((b-1)-1)-1 = b-3$.
3. sehingga didapat bahwa banyak pilihan kota kedua menjadi $b-3$.
4. karena makna (c_1, c_2) dan (c_2, c_1) sama, maka didapat bahwa banyak cara pemilihan kota adalah $\frac{b(b-3)}{2}$.

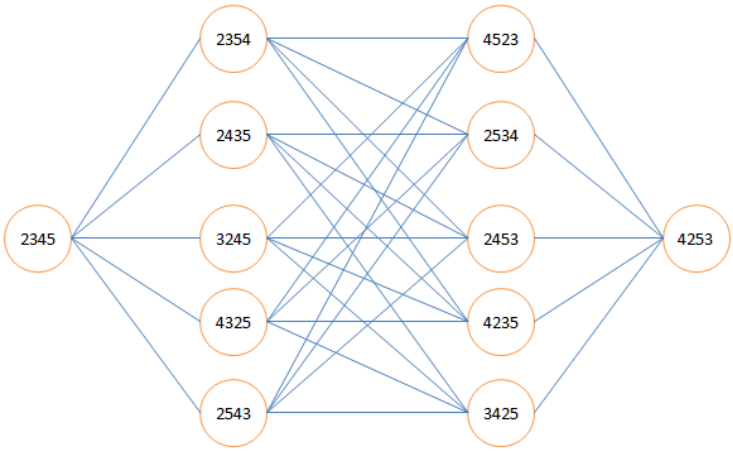
Sebagai contoh jika $n = 10$ dan bilangan acak Levy yang dihasilkan bernilai 2.345483641091, maka lakukan satu kali operasi double-bridge pada solusi $\mathbf{e}_i^t$). Sedangkan jika bilangan acak Levy yang dihasilkan bernilai 0.614398436457, maka lakukan 2-opt *move* sebanyak 6 kali(*step*), sebab nilai tersebut masuk pada interval $[0.6, 0.7)$.

4.2.3 Ruang Pencarian

Banyak cara membagikan c kota pada m *salesman* dapat adalah sebanyak $c\binom{(c-1)+m-1}{m-1} = c\frac{((c-1)+m-1)!}{((c-1+m-1)-(m-1))!(m-1)!} = c\frac{(c+m-2)!}{(c-1)!(m-1)!}$. Pada penelitian ini MTSP dapat dipandang sebagai kumpulan dari beberapa TSP yang terbentuk dari hasil klustering. Sehingga solusi dari MTSP dapat ditemukan dengan mencari rute terpendek dari tiap kluster.

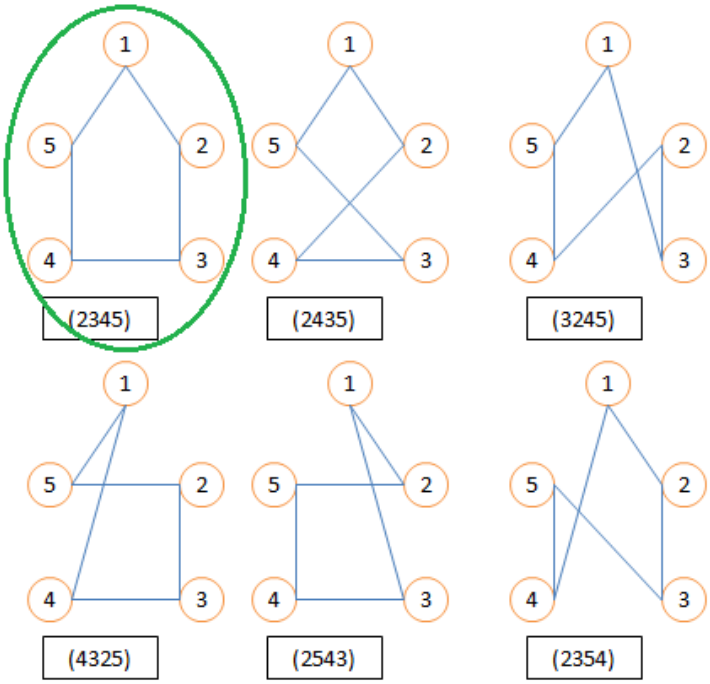
Setiap Cuckoo akan mencari posisi atau tempat untuk mendapatkan atau menghasilkan telur yang lebih baik. Ruang pencarian pada TSP adalah himpunan seluruh solusi atau permutasi dari kota. Banyak kemungkinan solusi yang berbeda pada ruang pencarian dengan menggunakan operator 2-opt adalah sebanyak $\frac{(g-1)!}{2}$ cara. Hal ini sudah didapat dipahami dengan menggunakan konsep permutasi siklik. Banyak permutasi siklik untuk g kota adalah $(g-1)!$.

Contoh ruang pencarian yang dihasilkan oleh operator 2-opt ditunjukkan pada Gambar 4.7. Setiap titik merepresentasikan sebuah rute yang berpotensi sebagai solusi dengan menggunakan 5 kota. Setiap titik tidak terhubung secara asal. Titik-titik yang saling terhubung menyatakan kemungkinan hasil jika sebuah titik dikenakan operator 2-opt.



Gambar 4.7: Ruang pencarian dengan menggunakan 2-opt pada 5 kota.

Pada gambar tersebut setiap titik merepresentasikan rute. Misal 2345, berarti dari kota 1 ke kota 2 hingga kota 5 untuk kembali lagi ke kota 1. Dan jika dinyatakan dengan telur menjadi $(1, 2, 3, 4, 5)$. Kota 1 yang disimbolkan dengan angka 1 tidak dituliskan. Telur $(1, 2, 3, 4, 5)$ ini jika dikenakan operator 2-opt, maka akan menghasilkan 5 permutasi yang lain, yaitu $(1, 2, 3, 5, 4), (1, 2, 4, 3, 5), (1, 3, 2, 4, 5), (1, 4, 3, 2, 5)$, dan $(1, 2, 5, 4, 3)$. Ini



Gambar 4.8: Visualisasi 2-opt untuk 5 kota.

tampak dari garis biru yang terhubung dengan 2345.

Banyak tetangga yang terhubung untuk setiap titik yang berada pada ruang pencarian adalah sebanyak $\frac{g(g-3)}{2}$. Rumus ini mirip seperti pada pencarian n pada persamaan (4.6).

4.2.3.1 Pergerakan Di Ruang Pencarian Dan Persekitaran

Koordinat dari setiap kota tidak berubah, maka pergerakan pada ruang pencarian adalah berdasarkan urutan dari kota yang dikunjungi. Pada penelitian ini operator 2-opt dan double-bridge digunakan untuk merubah susunan urutan dari kota yang ada pada telur sehingga membentuk solusi baru. 2-opt digunakan untuk pergerakan dengan langkah kecil (*small steps*) dan pergerakan dengan langkah yang besar (*big steps*) menggunakan double bridge. Perhatikan kembali Gambar 4.7. Jika solusi awal berada pada 3245 yang jika dinyatakan dalam telur (1,3,2,4,5), maka dengan menggunakan operator 2-opt pergerakan yang dihasilkan ada sebanyak 5 kemungkinan yaitu $4523(1,4,5,2,3), 2453(1,2,4,5,3), 4235(1,4,2,3,5), 3425(1,3,4,2,5)$ dan $2345(1,2,3,4,5)$. Gambar 4.8 memvisualisasikan 5 persekitaran dari 2345 atau telur $(1,2,3,4,5)$. untuk mendapatkan permutasi yang lain, maka cukup dengan mengubah susunan dari 2,3,4, dan 5 sehingga masing-masing telur dari kasus ini memiliki 5 hasil bentuk 2-opt. Pada algoritma Cuckoo Search dari 5

kemungkinan ini, dipilih yang menghasilkan rute dengan jarak paling pendek.

4.2.3.2 Pembentukan Solusi Menggunakan 2-opt

Susunan kota pada calon solusi atau solusi yang didapat membentuk sirkuit tertutup. Susunan ini nantinya akan diubah-ubah dengan operator 2-opt atau double-bridge.

2-opt *move* digunakan untuk mencari (pergerakan) solusi di persekitaran solusi awal (*neigbourhood*). Sedangkan double-bridge digunakan agar solusi yang terbentuk tidak terjebak pada kondisi *local optimum*.

Perlu diperhatikan bahwa representasi sikel dari hasil operator 2-opt dan doble-bridge berbeda dengan representasi sikel dari solusi (real). Pada penggunaan operator 2-opt dan double-bridge, semua telur direpresentasikan sebagai sirkuit tertutup tanpa memiliki *edge* yang berpotongan. Sebagai contoh Gambar 4.10 adalah representasi dari telur yang akan dikenakan operator 2-opt ataupun operator double-bridge. Representasi real dari telur bisa saja tidak seperti itu atau menghasilkan sirkuit yang memiliki *edge* yang berpotongan. Kemudian pada Gambar 4.8 adalah hasil dari operator 2-opt yang bekerja pada (2345). Tampak bahwa hasil dari operator tersebut menghasilkan sirkuit dengan *edge* yang bersilangan.

2-opt bekerja dengan mengambil dua *edge* dengan syarat dua *edge* tersebut dipisah oleh minimal satu edge. Kemudian dari dua *edge* yang diambil tadi dihubungkan ulang. Misalkan *edge* yang diambil adalah ab dan pq maka sirkuit yang terbaru akan memiliki *edge* aq bp dan *edge* ab dan pq tidak ada lagi pada sirkuit yang baru. Sebagai contoh, digunakan telur $(1, 2, 3, 4, 5, 6, 7, 8, 9)$.

e	1	2	3	4	5	6	7	8	9
e_{2opt}	1	5	4	3	2	6	7	8	9

Gambar 4.9: Hasil operator 2-opt.

Dimulai dengan memilih dua bilangan bulat acak yang saling berbeda dari 1-9. Syarat pemilihan kedua bilangan acak tersebut adalah selisih dari keduanya lebih dari 1. Jika bilangan yang didapat 9 dan 1, lakukan pemilihan lagi hingga tidak didapat pasangan 16 dan 1. Sebab 16 yang merepresentasikan 9 langsung terhubung dengan 1 atau *edge* (k_9, k_1) dan(k_1, k_2) adalah *edge* yang tidak dipisah oleh *edge* lain. Setelah mendapat dua bilangan acak; misalkan p dan q dan $p < q$, maka selanjutnya adalah mengubah posisi elemen pada K_1

yaitu

$$e = (k_1, k_2, ..., k_p, k_{p+1}, ..., k_q, k_{q+1}, ..., k_9).$$

Secara sederhana: titik p akan terhubung dengan titik $q+1$ dan titik $p+1$ akan terhubung dengan titik q. Lebih jelasnya: elemen K_1 pada posisi pertama hingga p tetap, sedangkan elemen pada posisi $p+1$ hingga q diisi dengan cara elemen posisi $p+1$ diisi oleh elemen q, elemen posisi $p+2$ diisi oleh elemen $q-1$ dst. Untuk elemen posisi $q+1$ hingga elemen posisi 9 tidak mengalami perubahan. Sehingga K_1 yang baru setelah dikenakan operator 2-opt berubah menjadi

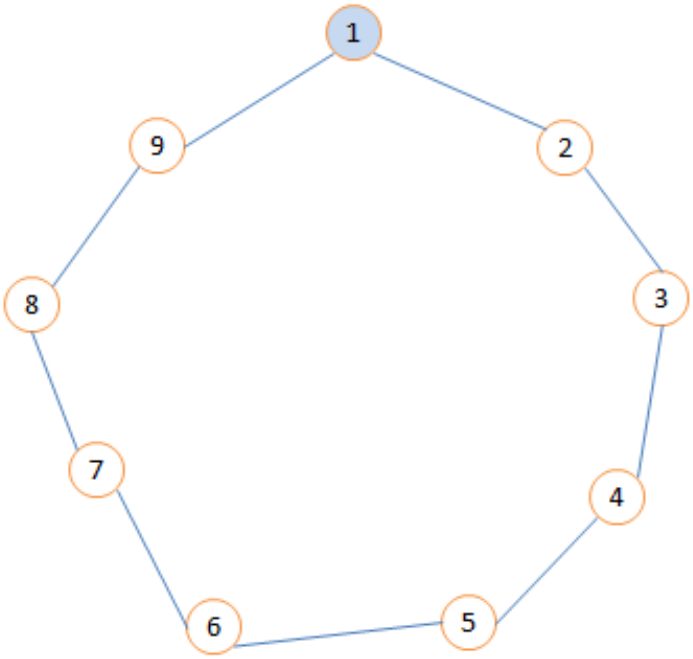
$$e_{2opt} = (k_1, ..., k_p, k_q, k_{q-1}, ..., k_{p+1}, k_{q+1}, , ..., k_9)$$

Algoritma 2-opt tidak bekerja jika jumlah titik atau kota pada sirkuit kurang dari 4. Hal ini didasari fakta bahwa 2-opt melibatkan penggunaan 4 titik yang berbeda. Gambar 4.9 menunjukkan telur awal dan hasil ketika telur dikenakan operator 2-opt. Telur yang belum dikenakan operator 2-opt disimbolkan dengan e. Setelah dikenakan operator:e_{2opt}. Pada contoh ini, titik atau posisi ke-2 dan ke-5 terpilih sebagai titik yang bertukar posisi dan titik di antara dan 2 dan 5 diurutkan ulang dengan posisi terbalik.

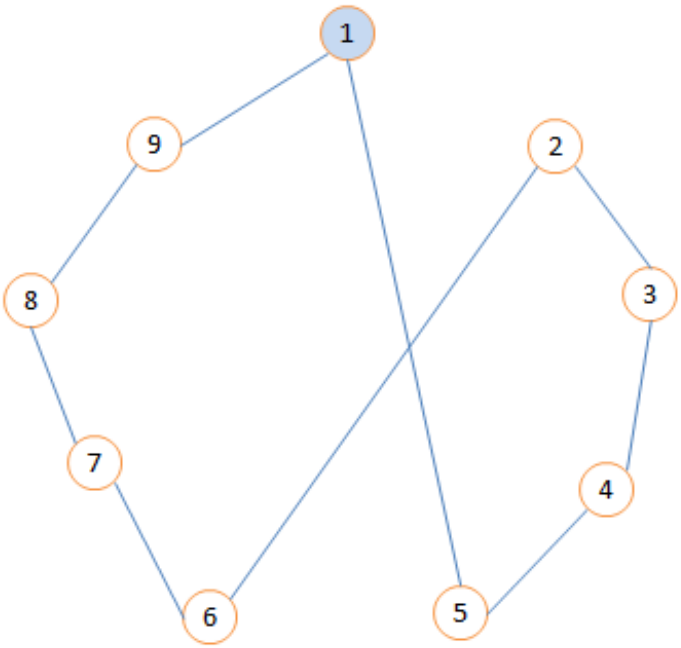
Pada Gambar 4.10 kota yang digunakan adalah sebanyak 9, dengan kota 1 sebagai titik depot. Sirkuit itu jika dinyatakan ke dalam telur menjadi $(1, 2, 3, 4, 5, 6, 7, 8, 9)$. Dengan $g = 9$, maka permutasi rute yang dihasilkan adalah $\frac{(9-1)!}{2} = 20.160$ cara dengan setiap permutasi memiliki persekitaran sebanyak $\frac{9(9-3)}{2} = 27$. Pada contoh ini terpilih $p = 1$ dan $q = 5$, sehingga *edge* $(1, 2)$ dan *edge* $(5, 6)$ dibuang dan dengan menggnakan 2-opt dibentuk dua *edge* baru yaitu $(1, 5)$ dan $(2, 6)$. Dari hasil operasi 2-opt terbentuk sirkuit atau telur baru yaitu $e_{2opt} = (1, 5, 4, 3, 2, 6, 7, 8, 9)$ dan ditunjukkan oleh Gambar 4.11.

4.2.3.3 Pembentukan Solusi Menggunakan Operator Double-Bridge

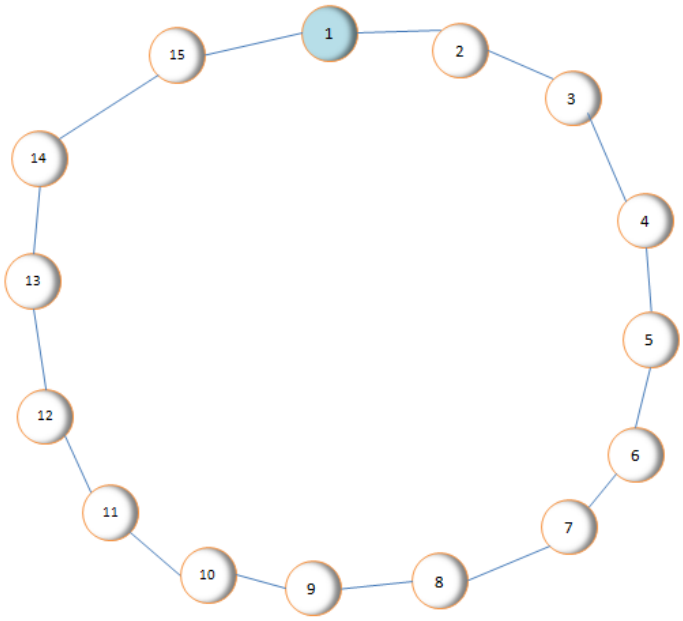
Operator double-bridge bekerja dengan mengambil 4 *edge* dan membentuk *edge* baru. Jelas bahwa pengambilan dan pembentukan ini melibatkan 8 titik yang berbeda. Sehingga operator double-bridge dapat bekerja jika jumlah kota yang diberikan minimal 8. Dengan operator double bridge, solusi yang dihasilkan diharapkan tidak berada pada sekitaran



Gambar 4.10: Sirkuit dengan 9 kota.



Gambar 4.11: Sirkuit dengan 9 kota yang sudah dikenakan operator 2-opt.



Gambar 4.12: Sirkuit dengan 15 kota.

dari solusi awal di ruang pencarian. Sebagai contoh, digunakan telur $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$ dan jika dinyatakan dalam bentuk sirkuit, dapat terlihat padaGambar 4.12. Pertama-tama akan dipilih 4

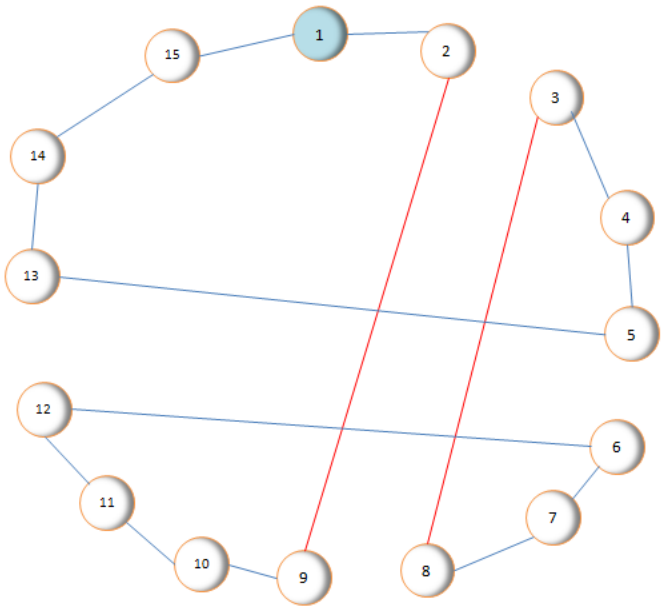
e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
e_{dbride}	1	2	9	10	11	12	6	7	8	3	4	5	13	14	15

Gambar 4.13: Hasil operator double-bridge.

bilangan acak berbeda di antara 1-15. Bilangan acak ini berfungsi sebagai indeks seperti yang digunakan pada operator 2-opt. Misalkan keempat bilangan acak itu adalah w, x, y, dan z dengan $w < x < y < z$, maka K_1 dapat ditulis menjadi

$$e = (k_1, k_2, ..., k_w, k_{w+1}, ..., k_x, k_{x+1}..., k_y, k_{y+1}, ..., k_z, k_{z+1}, ..., k_{15})$$

Untuk elemen posisi pertama hingga w tidak mengalami perubahan, untuk posisi selanjutnya diisi dengan elemen dari $y + 1$ hingga z, kemudian elemen dari $x+1$ hingga y, kemudian mulai dari $w+1$, hingga x dan sisanya diisi oleh



Gambar 4.14: Sirkuit dengan 15 kota yang sudah dikenakan double-bridge.

elemen $z+1$ hingga elemen terakhir. Sehingga diperoleh

$$e_{dbride} = (k_1, k_2..., k_w, k_{y+1}, ..., k_z, k_{x+1}, ..., k_y, k_{w+1}, ..., k_x, k_{z+1}, ..., k_{15})$$

Pada contoh ini terpilih $w = 2, x = 5, y = 8, z = 12$, sehingga dihasilkan telur $e_{dbride} = (1, 2, 9, 10, 11, 12, 6, 7, 8, 3, 4, 5, 13, 14, 15)$ dan perbedaannya dapat dilihat pada Gambar 4.13 serta bentuk sirkuit pada Gambar 4.14. e pada Gambar 4.13 divisualisasikan pada Gambar 4.12, sedangkan e_{dbride} tampak pada Gambar 4.14.

Dengan menggunakan kedua operator tersebut, algoritma Discrete Cuckoo Search akan menghasilkan solusi yang menuju ke arah optimal. Karena operator tersebut akan mengganti solusi yang baru jika solusi yang baru lebih baik dibandingkan solusi sebelum dikenakan operator double-bridge atau 2-opt. Selain itu, pada algoritma Discrete Cuckoo Search, untuk mengganti atau mendapatkan solusi baru, maka akan dipastikan bahwa solusi yang akan diganti adalah solusi yang buruk. Hal ini ada pada langkah pemilihan telur dan pembandingannya (telur x_i dan x_j) dan pada penggunaan parameter p_a. Parameter p_a menunjukkan seberapa besar peluang telur dengan kualitas buruk yang akan dibuang dari sarang. Proses ini diharapkan dapat memperbesar peluang mendapatkan solusi yang optimal. Pada proses penggantian atau pembuangan telur, operasi yang digunakan adalah double-

bridge. Sehingga, dipastikan bahwa telur yang dihasilkan setidaknya sama atau tidak lebih buruk dari telur yang akan dibuang.

4.3 Pembangkitan Bilangan Acak Berdistribusi Levy

Perlu diketahui bahwa nilai dari u pada (2.16) dapat bernilai negatif, sehingga menghasilkan step s yang bernilai negatif. Nilai negatif pada masalah permutasi terutama untuk operator 2-opt dan double-bridge sulit didefinisikan maka untuk menyederhanakannya dalam permasalahan diskrit, u akan dikenakan tanda mutlak $|\ |$. Maka dari itu (2.16) dimodifikasi menjadi

$$s = \frac{|u|}{|v|^{\frac{1}{\beta}}}. \tag{4.7}$$

Sebagai contoh, digunakan data dummy yang berisi 24 kota. Kemudian dilakukan proses klustering dengan mengunakan $k = 3$ yang berarti juga menggunakan 3 *salesman*. Didapat kluster1 berisi 3 kota, kluster2 15 kota dan kluster3 mendapat 6 kota. Pada kluster ini, titik depot masih berada pada salah satu kluster, yaitu kluster3. Hal ini dapat dilihat pada Gambar 4.15

Kemudian hasil dari kluster tersebut akan digunakan pada Cuckoo Search untuk menentukan rute terpendek. Sebelumnya dilakukan penyusunan ulang agar setiap kluster berisi titik depot dengan menyisipkan titik depot pada kluster1 dan kluster2. Setelah dilakukan penyisipan didapat kluster1= (kota-1, kota-5, kota-7, kota-23), kluster2= (kota-1, kota-2, kota-4, kota-6, kota-8, kota-11, kota-12, kota-13, kota-14, kota-15 , kota-17, kota-18, kota-19, kota-20, kota-21, kota-24) dan kluster3= (kota-1, kota-3, kota-9, kota-10, kota-16, kota-22).

Seperti yang dijelaskan sebelumnya antar satu kluster dengan kluster yang lain tidak saling berpengaruh. Maka untuk contoh pengerjaan Cuckoo Search pada kluster, digunakan kluster 1 yang terdiri atas 4 kota. Hasil kluster tampak pada Gambar 4.16. Centroid dari masing-masing kluster dinyatakan dengan tanda bintang. Centroid yang dihasilkan bukanlah kota yang ada pada data dummy.

Pertama-tama diinisialisasi parameter yang digunakan, yaitu, $p_a = 0.5$ dan banyak generasi($ngen$) = 200. Dari kluster1 akan menghasilkan sebanyak 3! permutasi solusi atau dengan kata lain akan menghasilkan sebanyak $\frac{(4-1)!}{2} = 3$ telur. Sebagai contoh populasi dari Cuckoo atau telur yang digunakan adalah

no.	x	y
1	98	198
2	145	56
3	118	193
4	87	118
5	24	158
6	101	73
7	25	184
8	159	43
9	87	183
10	123	175
11	131	76
12	86	56
13	183	41
14	147	121
15	188	47
16	171	151
17	100	93
18	40	39
19	143	65
20	57	64
21	131	101
22	126	165
23	32	108
24	95	91

Gambar 4.15: Data dummy 24 kota

3.

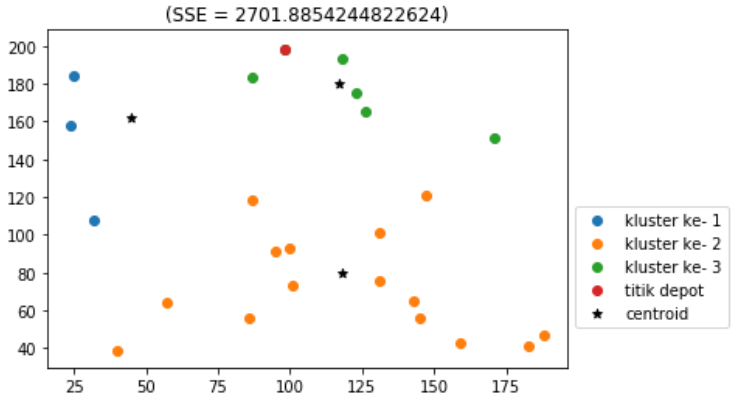
Selanjutnya, selama $t < ngen$ kerjakan:

langkah 1: pilih sebuah telur untuk dikenakan *Levy Flight* dengan membangkitkan bilangan acak menggunakan (2.16) dengan sebelumnya membangkitkan bilangan acak berdistribusi normal dengan varians yang didapat dari (2.17). Prosesnya ditunjukkan sebagai berikut.

$$\sigma_u = \left\{ \frac{\Gamma(1+\frac{3}{2})\sin(\frac{\pi\frac{3}{2}}{2})}{\Gamma(\frac{1+\frac{3}{2}}{2})2^{\frac{\frac{3}{2}-1}{2}}\frac{3}{2}} \right\}^{\frac{1}{\frac{3}{2}}} \tag{4.8}$$

sehingga diperoleh

$$\begin{aligned} \sigma_u &= 0.6965745 \\ \sigma_u^2 &= 0.485216 \end{aligned}$$



Gambar 4.16: Data kota yang dikluster.

dan selanjutnya didapat $u \sim N(0, 0.485216)$. Misalkan didapat $u = -1.0587047660005$ dan $v = -2.14487463862$. Selanjutnya untuk mendapatkan s, prosesnya adalah

$$\begin{aligned} s &= \frac{|u|}{|v|^{2/3}} \\ &= \frac{|-1.0587047660005|}{|-2.14487463862|^{2/3}} \\ &= 0.6365613392749903. \end{aligned}$$

Dengan $b = 3$ maka didapat $n = 2$ dari (4.6), selanjutnya untuk i:

$$i = \tfrac{1}{2+1} \tag{4.9}$$

$$= 0.\overline{3}. \tag{4.10}$$

dan $s \in [2 \times \frac{1}{3}, 1)$ sehingga operator yang dikenakan seharusnya adalah double-bridge, tetapi operator double-bridge membutuhkan minimal 8 titik, sehingga untuk menyiasatinya, operator yang digunakan adalah operator 2-opt yang dijalankan sebanyak 1 kali pada telur tersebut. Misalkan telur yang terpilih adalah $e_i = (1, 2, 3, 4)$, yang menunjukkan rute dari kota-1→kota-23→kota-7→kota-5→kota-1, maka kemungkinan telur yang didapat adalah sebanyak 2 saja yaitu $(1, 2, 4, 3)$ dan $(1, 3, 2, 4)$. Hal ini dapat ditentukan dengan menggunakan (4.6), yaitu

$$n = \frac{4(4-3)}{2}$$

langkah 2: hitung *fitness* dari telur e_i dengan menggunakan (4.4) dan nyatakan dengan f_i.

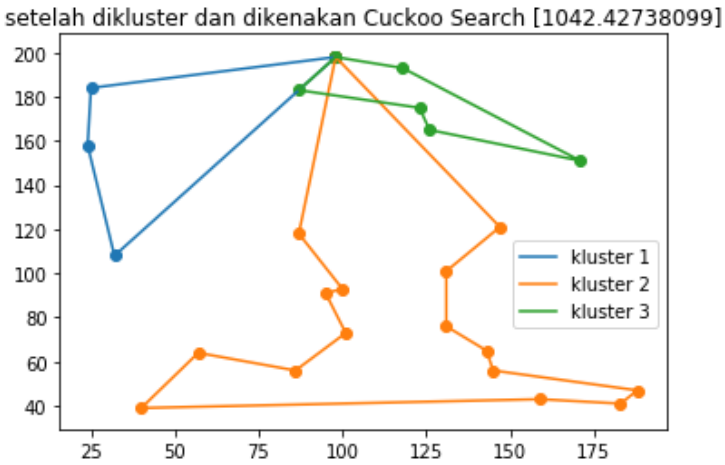
langkah 3: pilih secara acak sebuah telur dari populasi/sarang yang tersedia. Misalkan $e_j = (1, 2, 4, 3)$. Hitung *fitness*nya dan nyatakan sebagai f_j.

langkah 4: jika $f_i > f_j$ maka ganti telur e_j dengan telur e_i. Misal ternyata $f_i > f_j$ maka $e_j = e_i$.

langkah 5: sebanyak p_a bagian dari populasi yang buruk dibuang dan dibentuk sarang baru. Prosesnya adalah dengan membangkitkan sebuah bilangan acak. Jika bilangan acak terpilih kurang dari p_a maka dapatkan telur baru dengan mengerjakan sekali operator 2-opt pada telur yang terpilih. Misal, setelah membangkitkan bilangan acak, didapat telur $e_w = (1, 3, 2, 4)$ maka telur ini dikenakan operator 2-opt satu kali dan didapt e_w yang baru adalah $e_w = (1, 2, 4, 3)$.

langkah 6: urutkan telur atau solusi dengan melihat *fitness* dari masing-masing. Kemudian dapatkan sebuah telur dengan kualitas terbaik dan $t = t + 1$.

Lakukan langkah 1-langkah 6 hingga banyak generasi maksimal terpenuhi. Dari langkah tersebut nantinya diperoleh telur terbaik $e = (1, 4, 3, 2)$ yang merepresentasikan rute kota-1→kota-5→kota-7→kota-23→kota-1 adalah rute yang sudah dioptimasi dengan Cukoo Search dari kluster1 dan menghasilkan jarak 262.592.



Gambar 4.17: Hasil setelah dikenakan Cuckoo Search.

Langkah yang sama juga berlaku untuk kluster 2 serta kluster 3 dan masing-masing berturut-turut menghasilkan jarak terpendek 578.549 dan

201.287. Sehingga secara keseluruhan menghasilkan total jarak 1042.427. Hasil dari Cuckoo Search pada contoh ini tampak pada Gambar 4.17. Algoritma dari penggunaan k-means++ dan Discrete Cuckoo Search dapat dilihat pada Algorithm 2.

Algorithm 2 Pseudo-Code k-means++-Cuckoo Search

```
INPUT: data koordinat kota, jumlah salesman (m),
fungsi objektif f(x), x_0 = (x_1, ..., x_d).
banyak populasi n, MaksGenerasi (ngen), p_a
OUTPUT: x terbaik.
Lakukan klustering pada data kota yang diberikan dengan k-means++.
Sisipkan kota depot yang pada tiap kluster yang tidak terdapat titik depot.
for kluster ke-1 to kluster ke-m do
  Bentuk telur sebanyak n yang merupakan permutasi dari kota dan telah
  berisi titik depot
  while (t ≤ MaksGenerasi) do
    Ambil seekor Cuckoo secara acak dengan persamaan Levy flight.
    Hitung fitness f_i dari telur yang diambil.
    Pilih secara acak satu telur dari antara n sarang telur (misalkan x_j).
    if f_i > f_j then
      Ganti telur x_i dengan telur x_j (x_i = x_j),f_i = f_j.
    end if
    for i=1 to n do
      Dengan peluang ditemukan p_a, sarang yang berisi telur buruk
      dibuang dan dibentuk sarang yang baru.
    end for
    Simpan solusi yang terbaik.
    Urutkan solusi dan temukan solusi terbaik.
    t = t+1
  end while
  Dapatkan solusi terbaik dari kluster
end for
Jumlahkan fitness telur/solusi terbaik yang diperoleh per kluster.
Hasil dan visualisasi
```

4.4 Hasil Percobaan

Untuk menguji performa dari Cuckoo Search, dilakukan percobaan sebanyak 30 kali. Kemudian dilihat rata-rata, deviasi standar, koefisien variasi dari jarak yang didapat dan rerata waktu komputasi.

Pada Cuckoo Search, digunakan 3 parameter utama yaitu, p_a, banyak Cuckoo yang disimbolkan dengan *pop*, dan banyak generasi Cuckoo yang disimbolkan dengan *ngen*. Untuk parameter p_a, nilai yang digunakan adalah

$0, 0.25, 0.5, 0.75, 0.9$, dan 1. Pemilihan parameter $p_a = 0$ berarti tidak telur yang terbuang atau ditemukan oleh *host*. Sedangkan $p_a = 1$ berarti setiap telur yang dihasilkan, ditemukan oleh pemilik sarang sehingga Sebagai contoh digunakan data dari att48.xlxs sebagai percobaan untuk melihat pengaruh dari ketiga parameter tersebut. Pada contoh ini dilakukan sebanyak 30 kali percobaan. Hal yang diperhatikan dari setiap percobaan adalah jarak terpendek yang ditempuh dari 30 kali percobaan, rata-rata jarak terpendek dari 30 kali percobaan, deviasi standar, koefisien variasi, dan rata-rata waktu komputasi dari 30 kali percobaan.

4.4.1 ngen=50 Dan pop=35

Dapat dilihat dari Tabel 4.1 bahwa waktu komputasi yang dihasilkan dari keenam parameter p_a menghasilkan waktu komputasi yang bertambah. Rerata jarak yang terbaik yang dihasilkan berasal dari parameter $p_a = 0.25$; untuk jarak tependek didapat dengan menggunakan $p_a = 0$. Dengan melihat koefisien variasi yang dihasilkan metode Cuckoo Search cukup presisi. Tingkat presisi paling tinggi didapat dari $p_a = 0.9$ tetapi untuk jarak terpendek memang didapat pada baris $p_a = 0$ tetapi $p_a = 0$ menghasilkan rerata terbesar dan deviasi standar yang paling besar di antara 5 parameter p_a yang lain. Namun, sebenarnya jarak terpendek yang dihasilkan oleh $p_a = 0$ tidak jauh berbeda dengan $p_a = 0.25$ dan $p_a = 0.75$. Hanya selisih sekitar 8.537 satuan jarak.

Tabel 4.1: Hasil dengan menggunakan $ngen = 50$ dan $pop = 35$

p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	**41930.459**	43551.083	1458.351	0.033	4.394
0.25	41938.996	**43317.589**	848.137	0.020	4.862
0.5	42112.689	43348.067	858.053	0.020	5.467
0.75	41938.996	43377.392	842.378	0.019	6.062
0.9	42035.825	43426.885	773.471	0.018	6.375
1	42006.189	43467.130	863.747	0.020	6.632

4.4.2 ngen=100 Dan pop=35

Dari Tabel 4.2 dengan $ngen = 100$ dan $pop = 35$ diperoleh metode Cuckoo Search semakin presisi terlihat dari koefisien variasi yang kurang dari 0.014 tetapi sebagai kompensasi terdapat peningkatan waktu komputasi sekitar dua kali lipat dibanding dengan percobaan dengan menggunakan parameter $ngen = 50$ dan $npop = 35$ untuk setiap parameter p_a. Adapun jarak terpendek yang berhasil didapat secara umum lebih baik dibandingkan

dengan penggunaan $ngen = 50$ dan $pop = 35$. Perubahan paling siginifikan terjadi pada parameter $p_a = 0.5$. Terjadi pengurangan jarak hingga 586.61 satuan jarak. Secara umum dengan parameter $ngen = 100$ dan $pop = 35$ menghasilkan rerata jarak yang lebih baik dibanding dengan menggunakan parameter $ngen = 50$ dan $pop = 35$ walaupun waktu komputasi meningkat sekitar dua kali lipat.

Tabel 4.2: Hasil dengan $ngen = 100$ dan $pop = 35$

p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	41641.926	42579.116	532.003	0.012	8.729
0.25	41760.960	42476.073	408.461	0.010	9.773
0.5	**41526.079**	42624.666	534.480	0.013	10.922
0.75	41752.358	**42459.746**	409.583	0.010	12.055
0.9	41597.296	42703.919	462.785	0.011	12.808
1	41724.229	42580.077	396.544	0.009	13.267

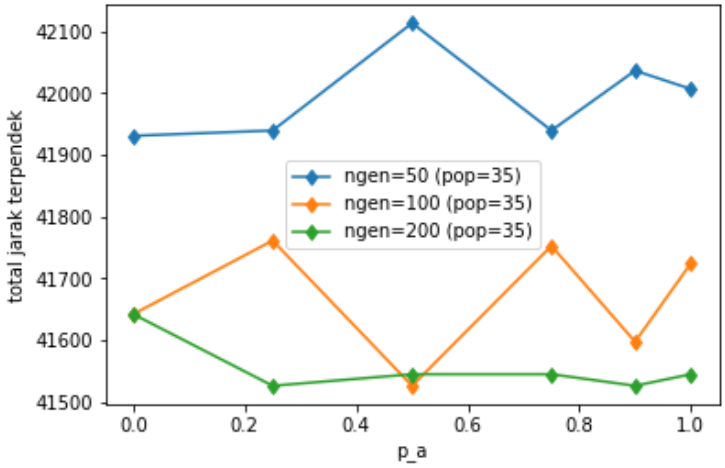
4.4.3 ngen=200 Dan pop=35

Tabel 4.3 menunjukan hasil untuk $ngen = 200$ dan $pop = 35$. Dari tabel tersebut diperoleh hasil jarak terpendek pada parameter $p_a = 0.25, 0.9$ sebesar 41526.079 dan rata-rata terkecil didapat pada $p_a = 0.9$ dengan nilai 42055.467. Koefisien variasi yang didapat secara keseluruhan untuk setiap parameter p_a lebih baik dibanding dengan $ngen = 50$ dan $pop = 35$, tetapi tidak jauh berbeda dengan $ngen = 100$ dan $pop = 35$. Namun untuk rata-rata waktu komputasi, dengan $ngen = 200, pop = 35$ menghasilkan nilai yang lebih besar dibanding dengan $ngen = 50$ dan $ngen = 100$ untuk $pop = 35$.

Tabel 4.3: Hasil dengan $ngen = 200$ dan $pop = 35$

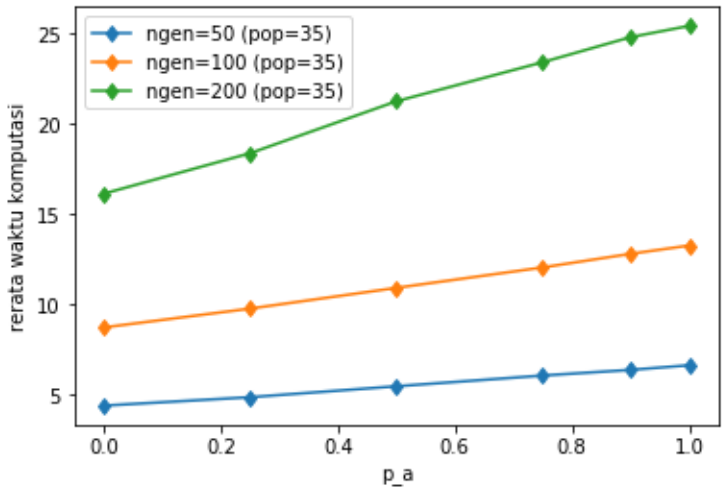
p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	41641.926	42308.793	359.320	0.008	16.144
0.25	**41526.079**	42174.770	332.182	0.008	18.387
0.5	41544.769	42074.756	291.493	0.007	21.266
0.75	41544.769	42112.461	283.292	0.007	23.431
0.9	**41526.079**	42055.467	278.892	0.007	24.818
1	41544.769	42116.472	236.863	0.006	25.443

Dengan melihat Gambar 4.18 dapat disimpulkan bahwa parameter p_a tidak berpengaruh pada jarak terpendek yang didapat. Misal, untuk $p_a = 0.25$ menghasilkan jarak terpendek di antara nilai p_a yang lain dengan $ngen = 200$,



Gambar 4.18: Total jarak terpendek yang didapat dari masing-masing p_a untuk att48.xlsx

tetapi tidak demikian dengan $ngen = 50$ dan $ngen = 100$. Pada Gambar 4.19 terlihat bahwa dengan nilai p_a yang bertambah, rata-rata waktu komputasi juga bertambah. Rata-rata terkecil didapat pada $p_a = 0$ untuk setiap $ngen$. Tampak juga bahwa $ngen$ yang semakin besar juga menghasilkan rata-rata waktu komputasi yang semakin besar.



Gambar 4.19: Rerata waktu komputasi dari 3 generasi dengan jumlah populasi yang sama

4.4.4 ngen=50 Dan pop=25

Dengan menggunakan $pop = 25$ dan $ngen = 50$ didapat jarak terpendek sebesar 42157.576 dari $p_a = 0.5$ dan rata-rata terkecil 43881.686 untuk p_a. Jika

dibanding dengan $pop = 35$, $pop = 25$ menghasilkan rerata dan jarak terpendek yang tidak lebih kecil. Selain itu deviasi standar yang dihasilkan secara umum lebih besar untuk hampir semua nilai p_a. Rata-rata waktu komputasi yang lebih kecil dihasilkan Tabel 4.4 untuk setiap p_a dibanding dengan Tabel 4.1.

Tabel 4.4: Hasil dengan $ngen = 50$ dan $pop = 25$

p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	42336.374	44307.065	1153.576	0.026	3.270
0.25	42675.715	44706.693	1307.852	0.029	3.442
0.5	**42157.576**	44216.047	1032.380	0.023	4.070
0.75	42322.261	**43881.686**	1094.638	0.025	4.378
0.9	42504.455	44466.806	1240.597	0.028	4.536
1	42269.258	44096.167	1096.833	0.025	4.778

4.4.5 ngen=100 Dan pop=25

Dengan $ngen = 100$ dan $pop = 25$ diperoleh hasil jarak terpendek yang lebih baik dibanding $ngen = 25$ dan juga rata-rata waktu komputasi. Namun terjadi peningkatan rata-rata waktu komputasi sekitar dua kali lipat dibanding $pop = 25$ tetapi deviasi standar yang didapat lebih kecil.

Tabel 4.5: Hasil dengan $ngen = 100$ dan $pop = 25$

p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	**41526.079**	42736.597	542.410	0.013	6.290
0.25	41760.959	42748.878	688.930	0.016	7.148
0.5	41625.930	42642.769	379.481	0.009	7.881
0.75	41722.224	42641.715	551.839	0.013	8.736
0.9	41544.769	**42571.493**	666.516	0.016	9.019
1	41979.943	42662.620	543.583	0.013	9.300

4.4.6 ngen=200 Dan pop=25

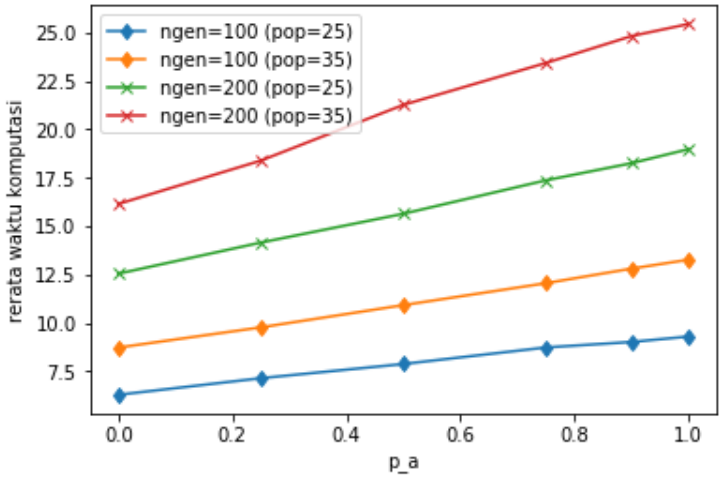
Untuk $pop = 25, ngen = 200$ diperoleh hasil yang mirip dengan percobaan parameter sebelumnya tetapi kali ini parameter $p_a = 0.9$ berhasil mendapatkan jarak paling pendek di Tabel 4.6. Selain itu rata-rata waktu komputasi lebih kecil dibandingkan dengan $pop = 35$ dan $ngen$ yang sama.

4.4.7 Diskusi Hasil

Dapat disimpulkan bahwa solusi yang dihasilkan dan waktu komputasi sangat dipengaruhi oleh jumlah generasi dan banyak populasi yang digunakan,

Tabel 4.6: Hasil dengan $ngen = 200$ dan $pop = 25$

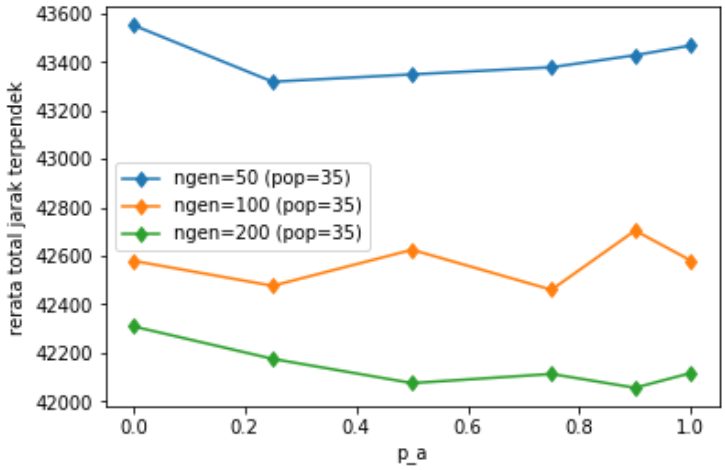
p_a	**jarak terpendek**	**rerata jarak**	**std**	**cv**	**rerata waktu komputasi**
0	41597.296	42552.128	391.536	0.009	12.545
0.25	41876.806	42358.463	332.289	0.008	14.153
0.5	41544.769	42364.533	438.651	0.010	15.641
0.75	41544.769	42194.899	459.095	0.011	17.369
0.9	**41526.079**	42183.665	414.896	0.010	18.252
1	41597.296	42298.010	406.900	0.010	18.977



Gambar 4.20: Rerata waktu dari $ngen = 100, 200$ dan $pop = 25, 35$ untuk att48.xlsx

sedangkan parameter p_a hanya mempengaruhi waktu komputasi dan tidak terlalu berpengaruh pada solusi yang dihasilkan. Sebagai contoh hasil total jarak terpendek pada Tabel 4.2 ada pada $p_a = 0.5$ dengan waktu komputasi 10.922 detik, sedangkan Tabel 4.5 dengan populasi yang berbeda, hasil terbaik ada pada $p_a = 0$ dengan waktu 6.290 detik. Pada Tabel 4.6 parameter $p_a = 0.9$ menghasilkan jarak terpendek. Semua parameter tersebut menghasilkan jarak terpendek yang sama yaitu, 41526.079 satuan jarak dan waktu terbaik ada pada Tabel 4.5 dengan waktu 6.290 detik. Gambar 4.22 menunjukkan hasil terbaik dengan jarak 41526.079. Selain itu, semakin besar nilai *ngen* dan *pop* yang digunakan semakin kecil nilai deviasi standar dan koefisien variasi yang digunakan. Ini menunjukkan bahwa algoritma Discrete Cuckoo Search akan semakin dekat pada hasil optimal jika parameter *ngen* dan *pop* semakin besar, tetapi waktu komputasi semakin bertambah.

Tampak pada Gambar 4.20 waktu komputasi dari kombinasi antara *ngen*



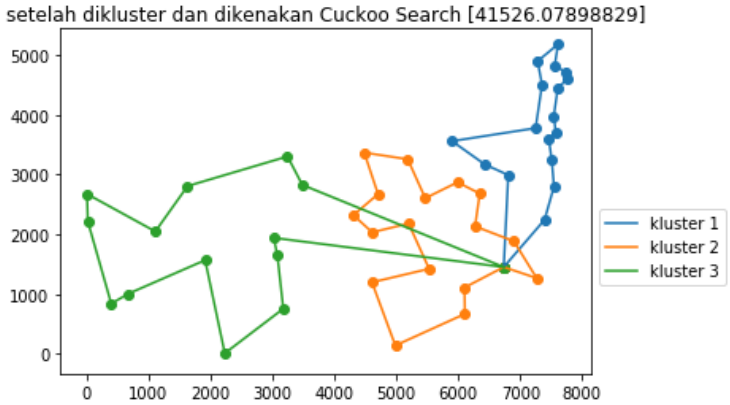
Gambar 4.21: Rerata jarak dari beberapa generasi untuk data att48.xlsx

Tabel 4.7: Hasil dengan beberapa jenis kluster pada data dumm.xlsx

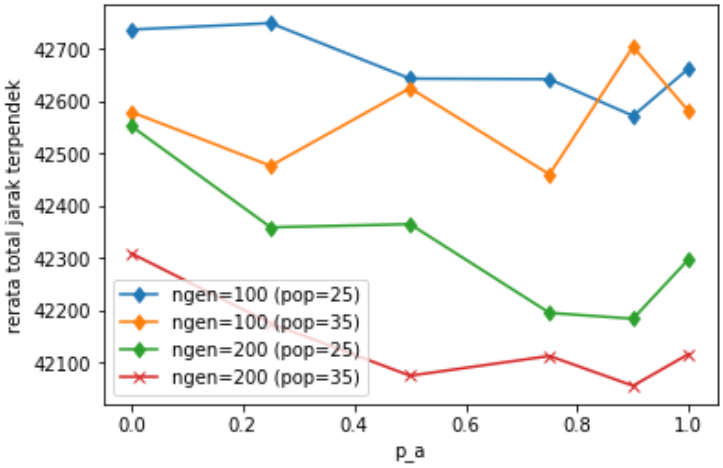
tipe kluster	jarak terpendek	rerata jarak	deviasi standar	rerata waktu komputasi
[9,6,9]	1058.515	1064.208	7.693	4.993
[14,6,4]	1049.936	1057.501	5.113	4.435
[6,15,3]	**1028.800**	1036.018	6.164	4.288

dan *pop*. Semakin besar nilai parameter yang diberikan pada masing-masing parameter, baik itu $ngen, pop$, dan p_a, semakin besar waktu komputasi yang dihasilkan. Namun untuk rerata total jarak yang dihasilkan, semakin besar nilai parameter *ngen* dan *pop*, rerata jarak yang dihasilkan cenderung semakin kecil. Sedangkan pada Gambar 4.21 dengan jumlah populasi yang sama dan *ngen* yang berbeda terlihat bahwa parameter p_a tidak berpengaruh pada rata-rata jarak terpendek yang ditempuh. Misalkan untuk $ngen = 200$ didapat rata-rata terpendek ada pada $p_a = 0.75$ sedangkan untuk $ngen = 50$, rata-rata terbaik ada pada $p_a = 0.25$. Pada Gambar 4.23 didapat informasi bahwa rerata jarak terpendek yang didapat akan semakin baik jika nilai $ngen = 200$. Sekali lagi terlihat pengaruh dari p_a tidak sama dengan untuk setiap *ngen*.

Gambar 4.24 menunjukkan perbandingan rerata jarak terpendek dengan rerata waktu komputasi (*pc*). Dari gambar tersebut dengan populasi yang sama didapat bahwa jarak $p_a = 1$ menghasilkan nilai paling kecil untuk masing-masing *ngen*. Ini berarti walaupun waktu relatif besar, namun diperoleh nilai jarak terpendek yang lebih kecil jika dilakukan percobaan yang banyak dibandingkan dengan parameter p_a yang lain. Maka untuk mendapat solusi yang baik dengan menjalankan algoritma metode Cuckoo Search lebih dari



Gambar 4.22: Hasil terbaik dari att48.xlsx



1

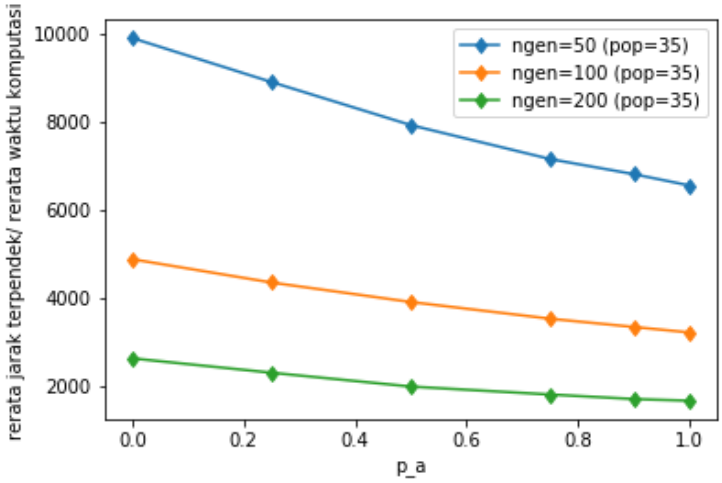
Gambar 4.23: Rerata jarak dari $ngen = 100, 200$ dan $pop = 25, 35$ pada att48.xlsx

satu kali akan lebih baik jika menggunakan paramater $p_a = 1$. Perbandingan itu dinyatakan sebagai berikut:

$$pc = \frac{\text{rerata jarak terpendek}}{\text{rerata waktu komputasi}} \tag{4.11}$$

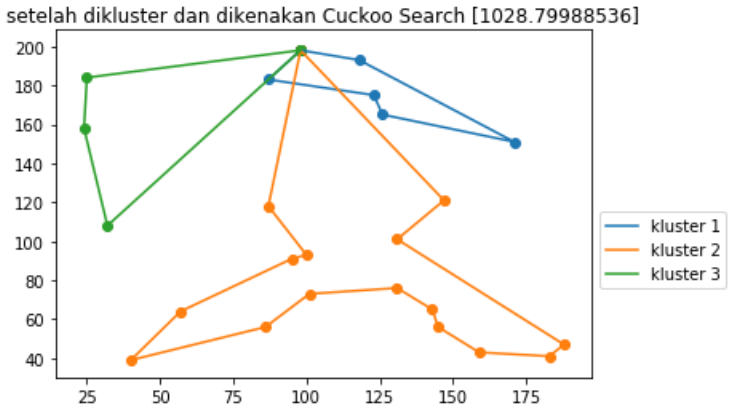
Untuk masing-masing $ngen = 50, 100, 200$, didapat pc terkecil adalah 6554.15, 3209.47, dan 1655.33.

Didapati juga bahwa hasil dari kluster dengan menggunakan k-means++ mempengaruhi hasil akhir atau jarak terpendek yang didapat. Sebagai contoh digunakan data dummy.xlxs sebagai contoh. Pada data ini memuat 24 kota. Mula-mula pengklusteran dengan $k = 3$, didapat kluster1 terdiri atas 9 kota, kluster2 6 kota dan kluster3 9 kota, dan kluster ini dinyatakan dengan [9,6,9].



Gambar 4.24: Perbandingan rerata jarak terpendek dengan rerata waktu komputasi

Dengan menggunakan parameter $p_a = 0, ngen = 100, pop = 35$, dan percobaan dilakukan sebanyak 30 kali, didapat: jarak terpendek sebesar 1058.515 dengan deviasi standar sebesar 7.693, rerata jarak adalah 1064.208 dan rerata waktu komputasi adalah 4.993 detik. Kemudian dengan data yang sama dilakukan pengklusteran dan diperoleh kluster yang sama tetapi dengan jumlah kota pada masing-masing kluster berbeda dengan percobaan sebelumnya, yaitu kluster1 14 kota, kluster2 6 kota dan kluster3 4 kota. Dengan parameter dan jumlah percobaan yang sama didapat: jarak terpendek 1049.936, deviasi standar 5.113, rerata jarak 1057.501, koefisien variasi 0.005 dan rerata waktu komputasi 4.435 detik. Pada Tabel 4.7 terlihat bahwa dengan kluster [6,15,3]



Gambar 4.25: Hasil dari kluster [6,15,3]

didapat total jarak terpendek dan rata-rata jarak paling kecil serta rerata

waktu komputasi yang digunakan juga paling kecil. Gambar 4.25 menunjukkan lintasan terpendek dari kluster [6,15,3].

4.4.8 Perbandingan Hasil Penelitian Lain

Pada penelitian ini digunakan data dari pr76.xlsx, rat99.xlsx, berlin52.xlsx dan bier127.xlsx dengan parameter Cuckoo Search yang digunakan adalah $p_a = 0.25$, $pop = 25$ jumlah populasi Cuckoo atau banyak telur yang dihasilkan serta $ngen = 200$ adalah banyak generasi yang dihasilkan (solusi). Selanjutnya hasil dari k-means++ dan Cuckoo Search dibandingkan dengan penelitian dari (Latah, Majd, 2016)

Tabel 4.8: Perbandingan dengan algoritma lain

data	jarak terpendek	rerata jarak	ACO	GA
berlin52.xlsx	8995.385	9169.158	11726.73	10898.75
rat99.xlsx	1737.224	1843.297	3328.02	1970.48
pr76.xlsx	139539.366	146890.744	265550.49	170857.76
bier127.xlsx	152768.335	160670.601	349770.9	257228.63

Pada data berlin52.xlsx setelah 30 kali percobaan didapat rata-rata jarak 9169.158 dengan jarak terpendeknya adalah 8995.385, nilai deviasi standar 160.116, koefisien variasi 0.0175 serta rerata komputasi: 13.700 detik. Hasil ini lebih baik dibanding dengan metode Algoritma Genetika dan Ant Colony Optimization pada penelitian Majd Latah. Dengan jumlah *salesman* yang sama, diperoleh hasil terbaiknya adalah 10898.750 dan 11726.730. Sedangkan pada rat99.xlsx didapat rata-rata jarak terpendek 1843.297 dengan jarak terpendek 1737.224, rerata waktu komputasi 26.637 detik. Untuk penelitian pada Majd Latah: jarak terpendeknya adalah 3328.020 untuk ACO dan 1970.480 untuk Algoritma Genetika. Untuk pr76.xlsx diperoleh rata-rata jarak terpendek 146890.744, dengan jarak terpendeknya 139539.366, deviasi standarnya 3005.354, dan rata-rata waktu komputasi 21.287 detik dan untuk bier127.xlsx, didapat rata-rata jarak terpendeknya adalah 160670.601 dan jarak terpendek yang didapat adalah 152768.335, dengan deviasi standar sebesar 4860.590 dan koefisien variasinya adalah 0.030 serta rata-rata waktu komputasi adalah 13.554 detik. Baik jarak terpendek yang ditemukan ataupun rata-rata jarak terpendek; nilai tersebut masih lebih baik dibanding dengan ACO dan GA.

BAB 5
KESIMPULAN

Pada bab ini akan disampaikan kesimpulan yang diperoleh dari analsis hasil pembahasan di bab sebelumnya.

Dari hasil penelitian yang dilakukan, didapat bahwa untuk menyelesaikan persoalan MTSP dengan k-means++ dan Discrete Cuckoo Search langkah-langkah yang dilakukan adalah: pertama-tama melakukan *clustering* pada data kota yang diberikan, kemudian dilakukan penyisipan titik depot pada setiap kluster. Kedua tahap itu dilakukan dengan menggunakan k-means++. Selanjutnya adalah melakukan optimasi pada setiap kluster yang didapat dengan menggunakan Discrete Cuckoo Search. Pada tahap ini, telur adalah permutasi kota dari kluster yang diberikan. Selain itu digunakan 2-opt dan double-bridge untuk modifikasi dari *Levy Flight*.

Diperoleh bahwa solusi yang didapat dari Discrete Cuckoo Search lebih baik dibanding GA dan ACO. Diamati parameter jumlah telur yang ditemukan host p_a, jumlah generasi $ngen$, dan banyak populasi dari Cuckoo pop. Setelah dilakukan percobaan sebanyak 30 kali pada data didapat bahwa

1. parameter p_a tidak berpengaruh signifikan terhadap solusi atau jarak terpendek yang didapat. Namun, parameter p_a berpengaruh pada waktu komputasi. Nilai $p_a = 0$ menghasilkan rata-rata waktu komputasi terkecil sedangkan $p_a = 1$ menghasilkan rata-rata waktu komputasi terbesar.

2. parameter $ngen$ dan pop berpengaruh pada waktu komputasi dan juga jarak terpendek yang dihasilkan. Semakin besar nilai dari kedua parameter tersebut, semakin besar waktu komputasi dan semakin kecil rata-rata jarak terpendek yang dihasilkan serta semakin besar kemungkinan untuk mendapatkan jarak terpendek yang ditunjukkan oleh nilai dari koefisien variasi dan variasi standar yang semakin kecil.

DAFTAR PUSTAKA

Joshi, Anuja.S. Kulkarni, Omkar M.,Kakandikar G. (2017), *Cuckoo Search Optimization- A Review*,International Conference on Advancements in Aeromechanical Materials for Manufacturing, Elsevier Ltd.

Kencana, Eka N. Harini, Ida Mayuliana, K. (2017), *The Performance of Ant System in Solving Multiple Traveling Salesmen Problem*, 4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017, Bali, Indonesia.

Latah, M. (2016), *Solving Multiple TSP Problem by K-Means and Crossover based Modified ACO Algorithm*International Journal of Engineering Research & Technology (IJERT) Vol. 5 Issue 02, February-2016.

Necula, Raluca Breaban, Mihaela dan Raschip, Madalina (2015), *Performance Evaluation of Ant Colony Systems for the Single-Depot Multiple Traveling Salesman Problem*, Lecture Notes on Artificial Intelligence (LNAI),Hybrid Artificial Intelligent Systems, Springer.

Shahab, Muhammad Luthfi Utomo, Daryono Budi Irawan, Mohammad Isa (2016), *Decomposing and Solving Capacitated Vehicle Routing Problem (CVRP) Using Two-Step Genetic Algorithm (TSGA)*, Journal of Theoretical and Applied Information Technology. Vol. 87 No. 3.

Said, Gamal Abd El-Nasser A. Mahmoud, Abeer M. El-Horbaty, El-Sayed M. (2014),*A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem*International Journal of Advanced Computer Science and Applications, Vol 5 No.1.

I. Fister Jr., X. S. Yang, D. Fister, I. Fister (2014), *Cuckoo search: A brief literature review*, in: Cuckoo Search and Firefly Algorithm: Theory and Applications, Studies in Computational Intelligence, vol. 516, pp. 49-62.

X.-S. Yang,(2014), *Cuckoo Search and Firefly Algorithm: Overview and Analysis*,Studies in Computational Intelligence, vol. 516, pp. 1-26.

Jati, Gilang Kusuma Manurung, Hisar Maruli Suyanto, Suyanto (2011), *Discrete Cuckoo Search for Traveling Salesman Problem*, IEEE Conference.

Ouaarab, Aziz Ahiod, Belaid Yang, Xin-She (2014a), *Discrete cuckoo search algorithm for the travelling salesman problem.* Neural Comput & Applic (2014)24:1659-1669.

Ouaarab, Aziz Ahiod, Belaid Yang, Xin-She (2014b), *Improved and Discrete Cuckoo Search for Solving the Travelling Salesman Problem.*Springer International Publishing Switzerland.

Luke, Sean (2013), *Essentials of Metaheuristics*, Lulu, second edition.

Wang, F., He, X.-S.,Wang, Y., Yang, S.M. (2012),*Markov model and convergence analysis based on cuckoo search algorithm.* Comput. Eng. 38(11), 180185

Civicioglu, Pinar Besdok, Erkan (2011), *A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms*, Artif Intell Rev, Springer,DOI: 10.1007/s10462-011-9276-0.

Sedighpour, Mohammad Yousefikhoshbakht, Majid Darani, Narges Mahmoodi, (2011), *An Effective Genetic Algorithm for Solving the Multiple Traveling Salesman Problem*, Journal of Optimization in Industrial Engineering 8 (2011) 73-79.

X.-S. Yang (2010), *Nature-Inspired Metaheuristic Algorithms*,Second Edition, Luniver Press.

X.-S. Yang, S. Deb (2009) *Cuckoo search via Levy flights*, in: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009),December 2009, India. IEEE Publications, USA, pp. 210-214.

David Arthur and Sergei Vassilvitskii (2007). k-means++: the advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1027-1035.

Gutin G, Punen AP (2002), *The Traveling salesman problem and its variations*, vol. 12. Springer, New York.

Setiyono, Budi (2002), *Multi Traveling Salesman Problem*. Tesis Jurusan Informatika ITB, Bandung.

BIOGRAFI PENULIS

Penulis bernama Handy Frank Wily Ambarita. Dilahirkan di Kendari, Sulawesi Tenggara 3 Februari 1993. Penulis menempuh pendidikan formal di SD Negeri 205 Kota Jambi tahun 1999-2005, SMP Negeri 7 Kota Jambi 2005-2008 dan SMA Negeri 5 Kota Jambi tahun 2008-2011. Setelah lulus dari SMA penulis melanjutkan studi S1 Jurusan Matematika di Institut Teknologi Sepuluh Nopember pada tahun 2011-2015. Silakan hubungi penulis melalui handyambarita@gmail.com.