IMPLEMENTASI ALGORITMA GENETIKA DAN TABU *SEARCH* UNTUK *TRAVELLING SALESMAN PROBLEM*

Disusun Oleh:

N a m a : Syarifah Elza Ramadhania

NIM : 17523027

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI ALGORITMA GENETIKA DAN *TABU SEARCH* UNTUK *TRAVELLING SALESMAN PROBLEM*

TUGAS AKHIR

Disusun Oleh:

N a m a : Syarifah Elza Ramadhania
NIM : 17523027

Yogyakarta, 24 Desember 2020
Pembimbing,

(Septia Rani, S.T, M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI ALGORITMA GENETIKA DAN *TABU SEARCH* UNTUK *TRAVELLING SALESMAN PROBLEM*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 15 Januari 2021

Tim Penguji

Septia Rani, S.T, M.Cs. _____

Anggota 1

Zainudin Zukhri, S.T., M.I.T. _____

Anggota 2

Taufiq Hidayat, S.T., M.C.S. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Syarifah Elza Ramadhania

NIM : 17523027

Tugas akhir dengan judul:

IMPLEMENTASI ALGORITMA GENETIKA DAN *TABU SEARCH* UNTUK *TRAVELLING SALESMAN PROBLEM*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 24 Desember 2020

(Syarifah Elza Ramadhania)

HALAMAN PERSEMBAHAN

Tugas Akhir ini saya persembahkan untuk diri saya, kedua orang tua, saudara, serta teman-teman yang selalu mendukung, mendo'akan, memberi semangat kepada penulis, sehingga atas berkat dukungan dari semua pihak dapat saya jadikan motivasi untuk menyelesaikan Tugas Akhir ini.

Puji dan syukur saya panjatkan kepada Allah SWT yang memberikan saya nikmat dan hidayah-Nya yang tak henti-henti. Yang selalu ada jika saya ingin bercerita, memberikan saya Kesehatan, kekuatan hingga Tugas Akhir ini dapat terselesaikan.

HALAMAN MOTO

"Aku selalu berada di hari ini yang harus baik dari kemarin"

"Jika dalam proses mencari ilmu terdapat banyak kesulitan, maka seharusnya saat ini dirikulah yang dapat menikmati hasil dari kesulitan hari kemarin"

"Setiap orang ada masanya, tiap masa ada orangnya"

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Alhamdulillah, puji dan syukur saya panjatkan kepada Allah SWT, berkat rahmat dan hidayah-Nya saya sebagai penulis dapat menyelesaikan Tugas Akhir mengenai "Implementasi Algoritma Genetika dan Tabu *Search* untuk *Travelling Salesman Problem"* guna menyelesaikan pendidikan saya dijenjang sarjana. Dalam menyelesaikan Tugas Akhir ini tentunya saya mempunyai beberapa kendala namun alhamdulillah kendal-kendala tersebut dapat diselesaikan dengan bantuan, bimbingan serta do'a dari orang-orang sekitar saya. untuk pada kesempatan ini penulis ingin menyampaikan terimakasih kepada :

1. Allah SWT telah memberikan saya segala rahmat dan hidayah-Nya, serta senantiasa memberikan Kesehatan kepada penulis.
2. Ida Suryani, S.H. dan Helsan Zulkifli, S.T.,M.T. selaku orang tua dari penulis yang telah selalu memberikan do'a, kasih sayang, nasehat, semangat dan selalu sabar dalam mendidik penulis hingga detik ini, serta memberikan segala fasilitas yang dibutuhkan dalam mengerjakan Tugas Akhir ini.
3. Syarifah Resha Fadziella, S.Si dan Syarifah Sheilla Novira yaitu saudara penulis yang selalu mendukung serta memberikan semangat dalam menyelesaikan Tugas Akhir.
4. Ibu Septia Rani S.T., M.Cs., selaku dosen pembimbing yang telah bersedia meluangkan waktu, pikiran, serta telah sabar dalam memberikan arahan dan bimbingan sehingga penulis dapat menyelesaikan Tugas Akhir dengan baik.
5. Dosen-dosen Informatika yang telah berjasa dalam memberikan ilmu dengan baik selama masa perkuliahan
6. Reezcky Noer Allamsyah Santoso yang selalu memberi dukungan, waktu, nasehat, serta dorongan hingga penulis dapat menyelesaikan Tugas Akhir.
7. Shinta Dewi Kusumaningrum dan Safira Yuniar Putri Buana selaku teman dekat penulis pada masa kuliah yang selalu ada hingga saat ini. Yang selalu memberikan dukungan serta mau mendengarkan keluh kesah dari penulis dengan sabar.
8. Fatih Assidhiqi, Raisha Amini Dinda Salechah, Adhin Alifarchan, Rahmat Hidayat Saputro, dan teman-teman kontrakan yang selalu ada jika penulis membutuhkan sebuah hiburan, dukungan, nasehat, serta telah memberikan waktu yang sangat berarti untuk penulis.
9. Annisa Zahra dan Aldhiyatika Amwin, teman yang sangat baik dan selalu memberikan dukungan.

10. Teman-teman PIXEL yang telah mewarnai kisah penulis di masa perkuliahan.
11. Seluruh pihak yang telah membantu dalam menyelesaikan Tugas Akhir.

Saya sebagai penulis menyadari Tugas Akhir ini walaupun telah diselesaikan masih mempunyai kekurangan sehingga penulis sangat membutuhkan kritik dan saran agar penelitian ini dapat lebih baik lagi. Akhir kata saya ucapkan terimakasih kepada seluruh pihak, semoga apa yang saya berikan pada penelitian ini dapat bermanfaat untuk kedepannya.

Wassalamu 'alaikum Wr. Wb

Yogyakarta, 24 Desember 2020

(Syarifah Elza Ramadhania)

SARI

Pada penelitian ini untuk menyelesaikan permasalahan *Travelling Salesman Problem* menyarankan untuk menggunakan gabungan dari dua algoritma yaitu Algoritma Genetika dan Tabu *Search*. Penelitian ini memiliki tujuan yaitu dengan hibridasi terhadap Algoritma Genetika dengan Tabu *Search* mampu memberikan hasil yang lebih baik dibandingkan dengan Algoritma Genetika. Algoritma ini akan diimplementasi pada sistem, adapun tahapan agar sistem dapat dimplementasikan, yaitu analisis melakukan pengumpulan data, menganalisis kebutuhan, perancangan dan pemodelan sistem.

Setelah tahapan tersebut dilaksanakan penelitian ini mampu menghasilkan sebuah sistem yang dapat menyelesaikan permasalahan *Travelling Salesman Problem* dengan kasus simetris. Serta sistem mampu menghasilkan hasil yang sesuai dengan tujuan penelitian ini. Pengujian sistem yang dilakukan ialah pengujian performasi sistem dengan data yang berbeda-beda dan ukuran yang berbeda-beda. Pengujian performasi dilakukan menggunakan tiga populasi yang berbeda, dan berdasarkan hasil pengujian performasi, Algoritma Genetika dan Tabu *Search* mampu mendapatkan hasil yang lebih optimal dibandingkan dengan Algoritma Genetika walaupun waktu yang dibutuhkan lebih lama. Sehingga pada penelitian ini masih perlu perbaikan seperti meningkatkan performa sistem agar hasil yang didapatkan bisa lebih baik lagi.

Kata kunci: *travelling salesman problem*, *algoritma genetika*, *tabulist*.

GLOSARIUM

Pada penelitian ini terdapat kata yang memerlukan penjelasan. Berikut paparan penjelasan tersebut :

Dataset	Kumpulan data yang setiap kolom tabel mewakili variable tertentu, dan setiap baris mewakili catatan atau isi tertentu dari kumpulan data yang dimaksud.
Runtime	Waktu ketika suatu program sedang berjalan. Dimulai dari sebuah program dijalankan dan diakhiri dengan program keluar ataupun ditutup.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam dunia matematika dan ilmu komputer, dalam menyelesaikan sebuah permasalahan memerlukan sebuah algoritma atau tata cara tertentu. Contoh pada dunia komputer, sebelum menyelesaikan masalah menggunakan komputer, diperlukan sebuah perumusan langkah-langkah penyelesaian (algoritma). Kemudian mengukur seberapa efisien suatu algoritma tersebut, dengan cara membandingkan beberapa algoritma yang ditemukan atau dibuat menggunakan *time complexity* dan *space complexity*. *Time complexity* ialah mengukur seberapa lama waktu yang dibutuhkan algoritma untuk menyelesaikan suatu masalah, sedang *space complexity* ialah seberapa besar memori yang digunakan untuk menyelesaikan suatu masalah (Friesen, 2017).

Hampir seluruh masalah yang ditemukan dikehidupan sehari-hari atau di dunia nyata, tanpa disadari memerlukan langkah-langkah atau cara dalam penyelesaiannya. Contoh seperti saat guru ingin mengambil seluruh lembar jawaban siswa saat ujian, guru tersebut pastinya akan memilih solusi atau jalan yang menurutnya lebih cepat dengan membandingkan beberapa solusi yang ia dapatkan, sehingga terpilihlah sebuah solusi yang terbaik. Contoh lainnya ialah ketika orang hendak menyapu lantai, pastinya orang tersebut akan memilih urutan area yang menurutnya akan membuat pekerjaannya lebih cepat selesai. Ataupun ketika sebuah keluarga liburan ke suatu kota, mereka pastinya akan menentukan sebuah perjalanan wisata. Dalam menentukan perjalanan wisata pun diperlukan sebuah langkah-langkah agar tujuan yang dinginkan tercapai.

Permasalahan-permasalahan tersebut biasa dikenal dengan *Travelling Salesman Problem*, yaitu permasalahan optimasi klasik dalam mencari rute kunjungan terpendek dikenal dengan istilah *Travelling Salesman Problem* (TSP) (Andri, Suyandi, & WinWin, 2013; Kusrini & Istiyanto, 2007). Seorang *salesman* terlibat dalam TSP, yang memiliki permasalahan yaitu dia wajib mengunjungi beberapa kota untuk menawarkan hasil produksinya. Saat melakukan kunjungan, *salesman* harus merencanakan suatu rute agar kota-kota tersebut hanya dilalui satu kali saja, kemudian *salesman* kembali ke kota asal. Meskipun bernama *Travelling Salesman Problem*, namun penerapannya tidak selalu berhubungan dengan *salesman* atau pedagang (Nana, Prihandono, & Noviani, 2015). Untuk menyelesaikan permasalahan TSP, ada beberapa

algoritma yang dapat digunakan namun mewajibkan memperhitungkan semua probabilitas rute yang dapat diperoleh (Puspitorini, 2008).

Pada perkembangannya, ternyata TSP merupakan salah satu permasalahan penting dalam dunia matematika dan informatika. Persoalan tersebut telah banyak diterapkan di berbagai kasus di dunia nyata (Rafflesia, 2016; Siagian, 2010). Adapun beberapa contoh kasus yang mampu dipecahkan dengan TSP pada dunia nyata antara lain (Kusrini & Istiyanto, 2007) : mencari rute bis agar dapat mengantarkan para siswa tepat waktu, mencari rute truk untuk mengantar paket, serta mencari rute pengambilan tagihan telepon secara efisien. Menurut (Wiyanti, 2013) terdapat dua jenis kasus TSP, yang pertama yaitu TSP asimetris. Pada kasus TSP asimetris, biaya dari kota 1 ke kota 2 berbeda dengan biaya dari kota 2 ke kota 1. Sedangkan kasus TSP simetris, biaya dari kota 1 ke kota 2 setara dengan biaya dari kota 2 ke kota 1.

Masalah yang akan dibahas dalam penelitian ini adalah penyelesaian *Travelling Salesman Problem* dengan kasus simetris. Sedangkan untuk algoritma, banyak algoritma yang dapat digunakan untuk mendapatkan rute terpendek dalam *Travelling Salesman Problem* yaitu Algoritma *Greedy,* Algoritma *Artificial Bee Colony, Tabu Search,* Algoritma *Cheapest Insertion Heuristics,* Algoritma Genetika, dan masih banyak lagi. Algoritma-algoritma tersebut memiliki kelebihan dan kekurangannya masing-masing. Hasil dari sebuah algoritma pun berbeda-beda, dikarenakan belum pasti sebuah algoritma yang memiliki optimasi yang tinggi untuk sebuah masalah memiliki optimasi yang tinggi juga untuk kasus lainnya (Aristi, 2014).

Sebelum ini sudah pernah dilakukan penelitian mengenai kasus-kasus TSP menggunakan Algoritma Genetika, algoritma tersebut mampu mendapatkan hasil yang lebih optimal dibandingkan dengan algoritma lainnya jika jumlah kota yang digunakan cukup besar, dan juga waktu pemrosesannya pun cenderung stabil (Abdulkarim & Alshammari, 2015). Namun, untuk hasil yang didapatkan pada regenerasi cenderung tidak variatif atau adanya perulangan dan hasil yang sama. Oleh karena itu, pada penelitian ini menggabungkan Algoritma Genetika dengan teknik lain (hibridisasi) agar hasil yang didapatkan lebih optimal dan lebih bervariatif. Terdapat beberapa penelitian yang menggunakan hibradasi Algoritma Genetika dan Tabu *Search* yang dibahas lebih rinci pada kajian teori. Penelitian-penelitian tersebut menghasil hasil bahwa teknik penggabungan Algoritma Genetika dan Tabu *Search* lebih mendapatkan hasil yang lebih baik dibandingkan Algoritma yang lain. Salah satu contoh pada penelitian (Sari, 2013) pada tahapan pengujian penelitian Algoritma Genetika dan Tabu *Search* mampu mendapatkan satu jadwal optimal pada generasi ke-98, sedangkan dengan Algoritma Genetika

jadwal optimal didapatkan pada generasi ke-298. Adapun perbandingan kompleksitas waktu dari Algoritma Genetika dan Algoritma Genetika-Tabu *Search.*

Berdasarkan penelitian sebelumnya algoritma yang digunakan pada penelitian ini ialah dengan melakukan hibridisasi terhadap Algoritma Genetika dengan Tabu *Search*, agar solusi yang didapatkan lebih optimal dibandingkan dengan Algoritma Genetika dengan menghindari solusi yang sama (perulangan), dan juga mendapatkan solusi yang lebih bervariatif.

1.2 Rumusan Masalah

Adapun masalah yang akan diselesaikan berdasarkan latar belakang masalah yang telah dijeleaskan :

a. Bagaimana implementasi hibridisasi Algoritma Genetika dan Tabu *Search* (GA-TS) dalam *Travelling Salesman Problem* dengan kasus simetris?
b. Bagaimana hasil perbandingan dari performansi antara GA-TS dan Algoritma Genetika dalam menyelesaikan *Travelling Salesman* Problem?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah :

a. Dihasilkan sebuah penyelesaian *Travelling Salesman Problem* dengan kasus simetris.
b. Didapatkan solusi yang lebih optimal dalam menyelesaikan *Travelling Salesman Problem* dengan menggunakan Algoritma Genetika dan Tabu *Search* dibandingkan dengan Algoritma Genetika.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

a. Dapat membantu penyelesaian *Travelling Salesman Problem* menggunakan GA-TS.
b. Dapat membantu agar mendapatkan solusi optimal dan terbaik menggunakan GA-TS dalam kasus simetris *Travelling Salesman Problem.*
c. Dapat menjadi tolak ukur dalam penelitian selanjutnya.

1.5 Batasan Masalah

Ruang lingkup pembahasan dalam penelitian ini sebagai berikut :

a. Penelitian ini berfokus pada penyelesaian *Travelling Salesman Problem* dengan kasus simestris.
b. Algoritma yang digunakan ialah hibridisasi terhadap Algoritma Genetika dengan Tabu *Search.*
c. Data yang digunakan merupakan koordinat titik !, # (bidang datar).

1.6 Sistematika Penulisan

Sistematika penulisan memuat tentang metode penulisan yang digunakan dalam penelitian. Berikut merupakan sistematika penulisan penelitian yang terbagi menjadi lima bab.

BAB I PENDAHULUAN

Berisi latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, Batasan masalah, dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi mengenai teori-teori yang digunakan dalam penelitian yang diambil dari berbagai sumber terutama penelitian-penelitian yang pernah dilakukan. Diantaranya ialah teori berdasarkan penelitian sebelumnya, teori mengenai *Travelling Salesman Problem,* dan teori mengenai Algoritma yang digunakan pada penelitian yaitu Algoritma Genetika dan Tabu *Search*.

BAB III METODE PENELITIAN

Berisi tentang uraian langkah-langkah yang dilakukan untuk menyelesaikan penelitian.

BAB IV HASIL DAN PEMBAHASAN

Berisi tentang pemaparan hasil implementasi dan hasil pengujian berdasarkan skenario pengujian pada BAB III.

BAB V KESIMPULAN DAN SARAN

Berisi tentang kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 Kajian Penelitian Sebelumnya

Travelling Salesman Problem (TSP) adalah sebuah permasalahan optimasi klasik yang sulit dipecahkan. Permasalahan TSP adalah bagaimana mencari perjalanan atau rute yang paling terbaik atau optimal tanpa mengunjungi suatu kota atau titik lebih dari satu kali. Dalam menyelesaikan permasalahan TSP diperlukanlah algoritma, algoritma yang dapat digunakan sangat banyak namun untuk setiap hasil akhir algoritma pun berbeda beda. Pada penelitian sebelumnya terdapat banyak algoritma yang telah digunakan untuk memecahkan permasalahan TSP, berikut pemaparannya.

Algoritma *Greedy* termasuk ke dalam jenis algoritma heuristik, algoritma ini mencari *local optimal* dan mengoptimalkannya sehingga mendapatkan *global optimal*. Berdasarkan penelitian oleh (Abdulkarim & Alshammari, 2015) walaupun Algoritma *Greedy* memiliki iterasi yang lebih banyak dibandingkan Algoritma Genetika dan *Nearest Neighbor*, hasil yang didapatkan pada Algoritma *Greedy* ialah yang paling mendekati hasil optimal. Sedangkan menurut (Lukman, AR, & Nurhayati, 2011) nilai yang diperoleh tidak terus-menerus optimal, dikarenakan masih terpikat dalam *local optimal*. Maka dari itu, Algoritma *Greedy* kurang tepat digunakan untuk jumlah kota yang besar.

Artificial Bee Colony termasuk ke dalam *swarm intelligence* yaitu Algoritma *Evolutionary Computation* (salah satu algoritma optimasi berbasis probabilistik). Algoritma ini terinspirasi oleh perilaku lebah yang hidupnya berkoloni. Seekor lebah mampu menjangkau sumber makanan sambil mengingat arah, tata letak, dan jarak dari sumber makanan. Berdasarkan hasil diskusi (Amri, Nababan, & Syahputra, 2012; Andri et al., 2013) *Artificial Bee Colony* ialah algoritma fleksibel, sederhana, dan juga mempunyai keahlian agar bebas dari *local minimum* dan secara kemampuan dimanfaatkan untuk multivariabel optimasi fungsi. Akan tetapi, apabila jumlah kota yang diproses semakin banyak maka tingkat *error* akan semakin besar.

Cheapest Insertion Heuristics (CIH) dan *Nearest Neighbor Heuristics* (NNH) merupakan metode heuristik yang cukup efektif memecahkan permasalahan TSP. Metode heuristik ini tidak selalu bisa memecahkan permasalahan, namun kerap memecahkan masalah dengan

cukup baik karena pada metode ini bertujuan untuk mengurangi jumlah pencarian solusi. Menurut (Kusrini & Istiyanto, 2007), *runtime* CIH jauh lebih cepat dibandingkan NNH. Kemudian pada penelitian (Abdulkarim & Alshammari, 2015) menyatakan bahwa ketika persoalan TSP memiliki kompleksitas besar dan jumlah iterasi besar, NNH mampu menemukan rute terpendek.

Algoritma Genetika adalah sebuah metode yang menirukan proses evolusi alam dan algoritma ini dikenal dalam menyelesaikan masalah kompleks. Waktu komputasi yang diperlukan pada Algoritma Genetika cenderung stabil, meskipun jumlah kota besar jika dibandingkan dengan algoritma lainnya. Algoritma Genetika sanggup memberikan rute terdekat, walaupun demikian, tingkat keberhasilan Algoritma Genetika sangat bergantung pada saat menetapkan parameter masukan, yaitu ukuran populasi, jumlah generasi, probabilitas *crossover*, dan probabilitas mutasi (Abdulkarim & Alshammari, 2015; Lukas, Anwar, & Yuliani, 2005).

Tabu *Search* termasuk ke dalam metode optimasi dengan teknik pencarian lokal. Tujuan dari metode ini ialah agar tidak terjadi perulangan serta didapatkannya hasil yang serupa pada suatu iterasi dan iterasi tersebut akan digunakan pada iterasi berikutnya (Fatmawati, Prihandono, & Noviani, 2015). Metode optimasi ini pun teruji efektif dan kerap digunakan untuk menyelesaikan permasalahan optimasi berskala besar. Berdasarkan penelitian oleh (Hay's, 2017), ia melakukan sebuah penelitian dengan menyelesaikan TSP menggunakan *Firefly Algorithm* namun terdapat kelemahan dalam penyelesaian optimasi berskala besar maka diperlukannya sebuah kombinasi dengan Tabu *Search* agar solusi yang didapatkan lebih akurat. Hasil akhir terbukti bahwa dengan kombinasi tersebut dapat meningkatkan hasil akurasi permasalahan TSP.

Held-karp oleh (Rani, Kurnia, Huda, & Ekamas, 2019) membuktikan bahwa hasil *runtime Held-karp* jauh lebih cepat dibandingkan dengan *brute force.* Walaupun terjadi peningkatan *runtime* setiap penambahan destinasi namun berjalan cukup stabil. Dapat dikatakan bahwa hasil penyelesaian menggunakan *Held-karp* lebih baik dari *brute force.* Kemudian (Nilsson, 2003) mengatakan bahwa dengan intensitas yang besar atau kota yang banyak tidaklah memungkinkan menggunakan algoritma ini dalam memecahkan masalah TSP.

Ant Colony Optimization (ACO) menirukan kelakuan semut saat mencari makanan dari sarangnya ke tempat sumber makanan dengan rute terpendek. Prinsip dasar ACO ialah pada rute yang dilalui saat perjalanan, semut selalu meninggalkan suatu feromon. Feronom tersebut

menjadi penunjuk jalan untuk semut yang lain dalam menyelesaikan perjalanan. Pada penelitian (Maria, Sinaga, & Iwo, n.d.), mengatakan bahwa ACO tidak dapat memperhitungkan jumlah iterasi yang baik, dikarenakan solusi yang dihasilkan tidak akan akurat bila jumlah iterasi terlalu kecil, dan solusi akan semakin besar jika jumlah iterasi terlalu besar. ACO tepat untuk jumlah simpul yang sangat banyak, dikarenakan ACO meletakkan semut pada setiap simpul, maka pencarian semakin cepat.

Branch and Bound adalah sebuah algoritma yang memiliki peranan untuk menyelesaikan permasalahan optimasi. Pendekatan yang digunakan dalam *branch and bound* ialah pendekatan numerisasi dengan menyingkirkan *search space* yang tidak tertuju pada pemecahan masalah. Hasil optimasi yang didapatkan menggunakan algoritma ini bergantung pada keakuratan pemilihan fungsi pembatas yang dipilih. Metode yang dipilih berdasarkan naluri dan pengetahuan maka seringkali hasil yang didapatkan tidak optimal. Akan tetapi pada aspek waktu, *branch and bound* dapat menjadi pilihan dalam menyelesaikan masalah optimasi kombinatorial (Prasetyo, 2017).

Tabel 2.1 Rangkuman Penelitian Sebelumnya

Peneliti	**Judul**	**Algoritma**	**Studi Kasus**
Abdulkarim, Haider A Alshammari, Ibrahim F 2015	*Comparison of Algorithms for Solving Traveling Salesman Problem*	*Nearest Neighbor Heuristic, Genetic Algorithm, Greedy Algorithm*	TSP kota-kota di USA
Lukman, Andi AR, Rubinah Nurhayati 2011	Penyelesaian *Travelling Salesman Problem* dengan Algoritma *Greedy*	Algoritma *Greedy*	Penyelesaian *Travelling Salesman Problem*
Amri, Faisal Nababan, Erna Budhiarti Syahputra, Mohammad Fadly 2012	*Artificial Bee Colony Algorithm* untuk Menyelesaikan *Travelling Salesman Problem*	*Artificial Bee Colony Algorithm*	Penyelesaian *Travelling Salesman Problem*
Andri Suyandi WinWin 2013	Aplikasi *Travelling Salesman Problem* dengan Metode *Artificial Bee Colony*	*Artificial Bee Colony Algorithm*	Penyelesaian *Travelling Salesman Problem*

Kusrini Istiyanto, Jazi Eko 2007	Penyelesaian *Travelling Salesman Problem* dengan Algoritma *Cheapest Insertion Heuristics* dan Basis Data	*Cheapest Insertion Heuristics*	Penyelesaian *Travelling Salesman Problem*
Lukas, Samuel Anwar, Toni Yuliani, Willi 2005	Penerapan Algoritma Genetika untuk *Travelling Salesman Problem* dengan Menggunakan Metode *Order Crossover* dan *Insertion Mutation*	Algoritma Genetika	Penyelesaian *Travelling Salesman Problem*
Fatmawati Prihandono, Bayu Noviani, Evi 2015	Penyelesaian *Travelling Salesman Problem* dengan Metode Tabu *Search*	Tabu *Search*	Penyelesaian *Travelling Salesman Problem*
Hay's, Riyan Naufal 2017	Implementasi *Firefly Algorithm* – Tabu *Search* untuk Penyelesaian *Travelling Salesman Problem*	*Firefly Algorithm* dan Tabu *Search*	Penyelesaian *Travelling Salesman Problem*
Rani, Septia Kurnia, Yoga Agung Huda, Sheila Nurul Ekamas, Sarah Ayu Safitri 2019	*Smart Travel Itinerary Planning Application using Held-Karp Algorithm and Balanced Clustering Approach*	*Held-Karp Algorithm*	Pembuatan *Itenarary* Wisata
Nilsson, Christian 2003	*Heuristics for The Traveling Salesman Problem*	*Heuristics Algorithm Category*	Penyelesaian *Travelling Salesman Problem*
Maria, Anna Sinaga, Elfira Yolanda Iwo, Maria Helena	Penyelesaian Masalah Travelling Salesman Problem Menggunakan *Ant Colony Optimization* (ACO)	*Ant Colony Optimization*	Penyelesaian *Travelling Salesman Problem*
Prasetyo, Yoga Dwi 2017	Penyelesaian Travelling Salesman Problem dengan Algoritma *Branch and Bound*	*Brand and Bound*	Penyelesaian *Travelling Salesman Problem*

Pada Tabel 2.1 merupakan rangkuman dari penelitian-penelitian tentang TSP. Berdasarkan penelitian-penelitian tersebut, banyak algoritma yang dapat menyelesaikan permasalahan TSP, hasil yang didapatkan setiap algoritma berbeda-beda yaitu mempunyai kelebihan dan kekurangannya masing-masing. Algoritma yang digunakan pada penelitian ini ialah Algoritma Genetika, berdasarkan penelitian sebelumnya algoritma ini mampu mendapatkan hasil yang optimal dibandingkan dengan algoritma lainnya, walaupun jumlah kota cenderung besar dan waktu yang didapatkan cenderung stabil. Namun agar hasil yang didapatkan lebih optimal penelitian ini dilakukan hibridisasi terhadap Algoritma Genetika yaitu menggunakan Tabu *Search*, agar solusi yang didapatkan terhindar dari perulangan. Adapun beberapa penelitian yang melakukan hibridisasi terhadap Algoritma Genetika dengan Tabu *Search*, berikut paparannya.

Pada penelitian (Sari, 2013) menggabungkan dua algoritma yaitu Algoritma Genetika dan Tabu *Search* (GA-TS), ia membuktikan bahwa kombinasi GA-TS hasilnya lebih efektif dibandingkan Algoritma Genetika pada kasus penjadwalan. Hasil pengujian membukti bahwa kombinasi Algoritma Genetika dan Tabu *Search* ketika jumlah generasi ke-98 sudah mendapatkan satu jadwal optimal, sedangkan dengan Algoritma Genetika jadwal optimal didapatkan pada generasi ke-298.

Kasus penjadwalan lainnya yang dilakukan oleh (Darmawan & Siliwangi, 2011) menyatakan bahwa hibridisasi Algoritma Genetika – Tabu *Search* mampu menjadi alternatif penyelesaian permasalahan SJMM (*scheduling jobs on multiple machines)*. Hasil dari performansi algoritma ini relatif baik, namun performansi semakin menurun jika ukuran data semakin bertambah. Selain itu adapun kasus mengenai penjadwalan *job-shop* yaitu pada penelitian membandingkan hasil akhir yang didapatkan dari Algoritma Genetika dan Tabu *Search* dengan Tabu *Search* (Thamilselvan & Balasubramanie, 2009). Hasil yang didapatkan ialah Algoritma Genetika - Tabu *Search* mampu memberikan hasil yang lebih baik menyelesaikan kasus penjadwalan dibandingkan menggunakan Tabu *Search*.

Menurut (Albar, 2013) dalam penelitian yang dilakukannya mengenai permasalahan penjadwalan kuliah, penelitian tersebut membandingkan hasil dari Algoritma Genetika dan Tabu *Search* dengan Algoritma Memetika, dan hasil keluaran kedua algoritma tersebut hampir sama, namun hasil dari Algoritma Genetika dan Tabu *Search* lebih unggul dibandingkan dengan Algoritma Memetika. Adapun penelitian pada kasus penjadwalan kuliah (Sumihar & Musdholifah, 2018) dengan menggunakan Algoritma Genetika dan Tabu *List*, pada penelitian tersebut didapatkan bahwa hasil penelitian berdasarkan segi waktu, Algoritma Genetika lebih

baik daripada kombinasi Algoritma Genetika dan Tabu *list* pada masing-masing parameter. Namun dari segi generasi Algoritma Genetika dan Tabu *list* lebih unggul, yaitu hanya membutuhkan 38 generasi sedangkan Algoritma Genetika membutuhkan lebih dari 300 generasi.

Berdasarkan hasil kajian penelitian yang telah dilakukan dan dipertimbangan, ialah diharapkan bahwa dengan adanya hibridisasi terhadap Algoritma Genetika dengan Tabu *Search*, kedua algoritma tersebut saling melengkapi kekurangan-kekurangan yang ada dan menghasilkan hasil yang lebih optimal dibanding Algoritma Genetika.

2.2 Travelling Salesman Problem

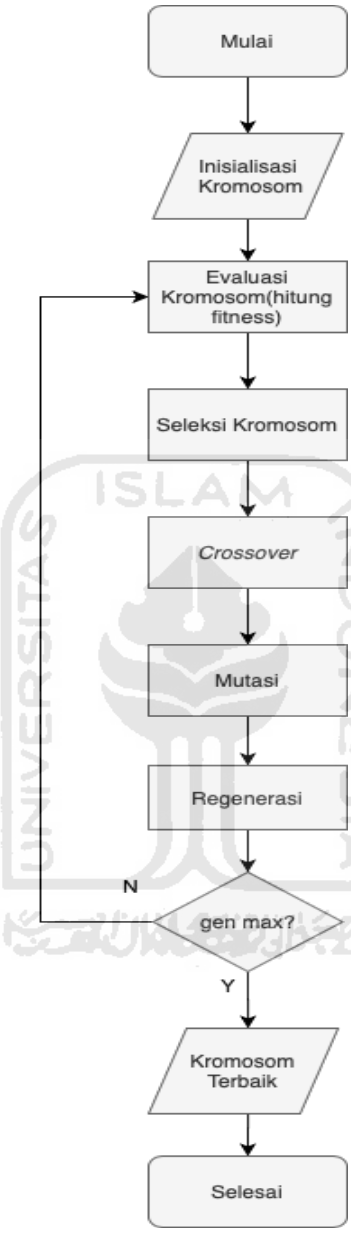
Pertama kali ide mengenai *Travelling Salesman Problem* disampaikan pada tahun 1800 oleh matematikawan Irlandia William Roman Hamilton dan matematikawan Inggris Thomas Penyngton (Hay's, 2017). Dimulai pada tahun 1930 para matematikawan mempelajari bentuk umum dari TSP. Karl Menger di Vienna dan Harvard mengawalinya, setelah itu permasalahan TSP dipublikasikan oleh Hassler Whitney dan Merrill Flood di Princenton (Rafflesia, 2016). Pada tahun 1985 Hoffman dan Wolfe menjelaskan bahwa TSP ialah masalah yang harus diselesaikan oleh seorang *salesman* dalam menjalankan perjalanan.

Travelling Salesman Problem termasuk persoalan teori graf. TSP merupakan sebuah persoalan seorang *salesman* harus menjajakan produknya ke kota-kota namun setiap kota hanya bisa dikunjungi satu kali saja dan kembali ke kota awal keberangkatan. Hal tersebut sering dikenal dengan nama sirkuit *hamilton*. Meskipun bernama *Travelling Salesman Problem*, namun penerapannya tidak selalu berhubungan dengan *salesman* atau pedagang (Nana et al., 2015). Penyelesaian dari kasus TSP ialah menemukan jarak terpendek dari kota atau tempat yang akan dituju. Cara penyelesaiannya ialah dengan menghitung seluruh peluang rute, setelah itu menentukan salah satu rute yang terpendek. Maka dari itu bila terdapat *n* kota yang harus dituju, maka terdapat *n*! kombinasi kota yang akan dibandingkan jaraknya. *Runtime* yang diperlukan tentu bertambah bersamaan dengan meningkatnya jumlah kota yang harus dituju (Lukman et al., 2011).

2.3 Algoritma Genetika

Algoritma Genetika (GA) termasuk ke dalam algoritma evolusi, teknik optimasi pada algoritma ini menirukan proses evolusi biologi. GA merupakan tipe algoritma evolusi yang paling terkenal dan banyak diimplementasikan pada permasalahan yang kompleks. Dalam

bidang biologi, sosiologi, fisika, ekonomi, Algoritma Genetika banyak digunakan hingga masalah optimasi yang model matematikanya sangat kompleks (Firdaus Mahmudy, 2013). Pada Gambar 2.1 merupakan *flowchart* algoritma genetika.



Gambar 2.1 *Flowchart* Algoritma Genetika

a. Inisialisasi

Inisialisasi dibuat secara acak untuk membangkitkan himpunan solusi baru yang terdiri dari sejumlah string kromosom dan disebut sebagai populasi. Kromosom mengandung informasi solusi dari sekian banyak kemungkinan solusi masalah yang dihadapi. Untuk mendapatkan solusi terbaik, setiap generasi akan menghasilkan kromosom-kromosom baru yang dibentuk dari generasi sebelumnya dengan menggunakan operator kawin silang dan

mutasi. Dalam satu individu terdiri dari beberapa gen yang direpresentasikan dalam bentuk angka-angka (*allele*), dan setiap angka tersebut mewakili titik-titik (tempat) tujuan yang akan dilalui. Setelah kromosom terbentuk, maka selanjutnya adalah perhitungan nilai *fitness* seperti persamaan dibawah ini. Istilah dari *TotalJarak* persamaan 2.1 ialah total jarak yang ditempuh dari titik awal hingga akhir (Lukas et al., 2005).

$$\$\%\&'() = \frac{1}{,-\&./0.1.2} \qquad (2.1)$$

b. Seleksi

Proses seleksi merupakan peranan penting dalam Algoritma Genetika. Proses ini digunakan agar hanya kromosom-kromosom yang memiliki kualitas baik dan dapat melanjutkan peranannya dalam proses algoritma genetika berikutnya. Ada berbagai macam teknik seleksi yang dapat digunakan, diantaranya adalah *Roulette Wheel Selection, Elitism, Rank Base Selection,* dan *Steady State Selection* (Lukas et al., 2005). Dalam penelitian ini Teknik seleksi yang akan digunakan ialah *Rank Based Selection* dan *Elitism*. Seleksi dengan *Roulette Wheel Selection* ialah kromosom yang sebelumnya melakukan perhitungan *fitness* akan diurutkan berdasarkan nilai terbesar hingga terkecil dan yang memiliki nilai *fitness* terbesar kemungkinan besar akan menjadi induk pada generasi selanjutnya. Kemudian kromosom tersebut akan diseleksi menggunakan *Elitism* dengan mempertahankan nilai *fitness* terbaik. Sedangkan untuk nilai *fitness* terburuk akan digantikan oleh individu baru dengan nilai *fitness* lebih besar. Setelah proses perhitungan fitness selesai, maka akan dipilih kromosom dengan nilai *fitness* terbaik dan akan menjadi induk untuk metode *crossover*.

c. Crossover

Crossover atau kawin silang menghasilkan dua kromosom anak dengan cara menukar informasi dari dua induk kromosom (Dini, 2015). Proses ini akan memilih dua kromosom induk yang memiliki nilai fitness terbaik kemudian akan mengalami proses kawin silang atau *crossover* secara acak dan menghasilkan kromosom anak. Metode *crossover* ada beberapa macam, di antaranya yaitu *crossover* banyak titik, *crossover* satu titik, *crossover* aritmatika, dan *crossover* untuk representasi kromosom permutasi. Pada *crossover* untuk representasi kromosom permutasi ada beberapa metode, seperti *order-based crossover, order crossover,*

cycle crossover, position-based crossover, partial mapped crossover, heuristic cross over, dan lain-lain (Melorose, Perroy, & Careas, 2015).

d. Mutasi

Mutasi adalah proses menghasilkan individu baru dengan melakukan modifikasi satu atau lebih gen dalam individu itu sendiri. Mutasi akan membuat populasi menjadi bervariasi dengan menukar gen yang hilang dari populasi selama proses seleksi serta menyimpan gen yang tidak ada dalam populasi awal (Priandani & Mahmudy, 2015). *Reciprocal exchange mutation* adalah metode mutasi yang paling sederhana. Metode ini berjalan dengan memilih dua posisi (*exchange point* / XP) secara acak kemudian menukarkan nilai pada posisi tersebut. Selain *reciprocal exchange mutation* ada metode lain yang dapat digunakan yaitu *insertion mutation*. Metode ini bekerja dengan memilih satu posisi (*selected point* / SP) secara acak kemudian mengambil dan menyisipkan nilainya pada posisi lain (*insertion point* / IP) secara acak (Firdaus Mahmudy, 2013).

e. Regenerasi

Setelah melewati tahap mutasi, hasil dari proses tersebut ditempatkan pada populasi yang baru atau generasi baru. Kemudian populasi baru ini akan diproses kembali dari tahapan ke-2 (menghitung nilai *fitness*) hingga kromosom mencapai iterasi atau perulangan yang telah ditentukan.

2.4 Tabu *Search*

Pada tahun 1970-an, Glover memperkenalkan Tabu *Search* (TS) pertama kalinya. Menurut Glover, TS adalah salah satu metode *metaheuristic* tingkat tinggi untuk penyelesaian permasalahan optimasi kombinatorial. TS menyimpan solusi terbaik dan tetap mencari berdasarkan solusi terakhir, untuk menjaga agar solusi terbaik tidak hilang. Selain itu algoritma ini mampu mengingat sebagian solusi yang pernah didapatkan dan mencegah menggunakan solusi yang telah ditelusuri untuk menghindari pengulangan yang sia-sia.

Tabu *Search* menggunakan struktur memori yang disebut *Tabu List* untuk menyimpan atribut dari sebagian *move* yang telah diterapkan pada iterasi-iterasi sebelumnya. Tujuan dari algoritma ini adalah mencegah terjadinya perulangan dan ditemukannya solusi yang sama pada suatu iterasi yang akan digunakan lagi pada tahap selanjutnya (Fernando, 2012). Adapun tahapan-tahapan dalam Tabu *Search* sebagai berikut:

a. Menentukan titik awal.

b. Menentukan solusi alternatif dengan menggabungkan dua titik berdasarkan jarak terkecil.

c. Megevaluasi solusi-solusi *alternative* dengan tabu *list* untuk melihat apakah solusi tersebut sudah ada pada tabu *list*. Untuk menghindari perulangan, jika solusi *alternative* sudah ada dalam tabu *list* maka solusi tersebut tidak akan dievaluasi lagi, dan begitu sebaliknya.

d. Memilih solusi terbaik dan menetapkannya sebagai solusi optimum baru.

e. Memperbarui tabu *list*.

f. Ulangi langkah-langkah tersebut hingga kriteria sudah terpenuhi.

Kriteria pemberhentian ialah berdasarkan dari kriteria yang telah dibuat atau ditentukan contohnya jika iterasi mencapai jumlah maksimum.

BAB III
METODOLOGI PENELITIAN

3.1 Pengumpulan Data

Data yang digunakan pada sistem ini adalah *dataset* TSP dengan kasus simetris yang memiliki koordinat titik !,#. Data yang diambil bervariasi yaitu dengan ukuran yang berbeda-beda dari kecil hingga besar. Pada data yang digunakan, sudah terdapat solusi optimumnya sehingga pada tahap evaluasi, hasil yang didapatkan dari algoritma yang diusulkan dapat dibandingkan dengan solusi yang optimum.

Dataset yang digunakan diambil melalui internet pada tautan http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/ . Tabel 3.1 merupakan contoh data TSP yang akan digunakan, pada kolom pertama merupakan kolom kota atau titik, kolom kedua ialah koordinat titik ! dan kolom terakhir ialah koordinat titik #. Sebagai contoh, pada kota 1, koordinat titik ! ialah 6734 sedangkan koordinat titik # ialah 1453.

Tabel 3.1 Contoh Data TSP

Kota	X	Y
1	6734	1453
2	2233	10
3	5530	1424
4	401	841
5	3082	1644
6	7608	4458

3.2 Analisis Kebutuhan

Analisis kebutuhan merupakan hal-hal apa saja yang dibutuhkan dalam pengembangan sistem berdasarkan kebutuhan masukan, kebutuhan proses, kebutuhan keluaran, kebutuhan perangkat keras, dan kebutuhan perangkat lunak.

3.2.1 Kebutuhan Masukan

Kebutuhan masukan yang diperlukan dalam pengembangan sistem ini ialah :

a. *User* memasukkan data yang akan digunakan.
b. *User* memasukkan ukuran populasi tiap generasi.

c. *User* memasukkan maksimal generasi.

3.2.2 Kebutuhan Proses

Kebutuhan proses yang ada dalam pengembangan sistem ini adalah:

a. Proses konversi data koordinat titik !,# menjadi jarak antar titik atau kota.
b. Proses pembuatan inisialisasi kromosom atau rute secara *random*.
c. Proses menghitung nilai *fitness* (evaluasi) kromosom atau rute.
d. Proses seleksi kromosom atau rute.
e. Proses pengecekan *tabulist*.
f. Proses *crossover*.
g. Proses mutasi.
h. Proses menghasilkan kromosom atau rute terbaik.
i. Proses menghitung *fitness* akhir.

3.2.3 Kebutuhan Keluaran

Keluaran yang dibutuhkan ialah program dapat menampilkan hasil permasalahan TSP dari setiap proses yang dilewati hingga menghasilkan rute terbaik dengan menggunakan Algoritma Genetika dan Tabu *Search*.

3.2.4 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak adalah kemampuan yang dimiliki oleh suatu perangkat lunak agar memenuhi apa yang diperlukan oleh pengguna. Ada beberapa kriteria yang dibutuhkan dalam mengembangkan sistem ini, yang pertama dari segi fungsional, perangkat lunak mampu menyajikan informasi hasil akhir dari permasalahan yang akan diselesaikan yaitu berupa rute terpendek atau terbaik. Kemudian perangkat lunak mampu menghasilkan waktu pemrosesan setiap data, dan dapat mengolah data yang cukup besar hingga hasil didapatkan. Serta perangkat lunak dapat menyajikan informasi lainnya berupa grafik dan juga waktu pemrosesan yang diperlukan oleh sistem. Dari analisis kebutuhan perangkat lunak tersebut, perangkat lunak yang digunakan dalam pembuatan sistem ini ialah *Jupyter Notebook* dengan pemograman berbahasa Python.

3.3 Perancangan dan Pemodelan Sistem

3.3.1 Kerangka Kerja Sistem Penyelesaian TSP

Dalam pemodelan sistem dibutuhkan sebuah kerangka kerja agar dapat mempermudah pembuatan sistem dan sesuai dengan analisis kebutuhan yang telah dilakukan. Ada tiga analisis kebutuhan yang diperlukan, yaitu analisis kebutuhan masukan, analisis kebutuhan proses, dan

analisis kebutuhan keluaran. Dalam kerangka kerja ini, masukan akan dilakukan oleh *user*, dan memasukkan data, jumlah populasi, dan maksimal generasi. Kemudian proses dan keluaran merupakan reaksi atau hasil dari aksi yang telah dimasukkan oleh *user*.

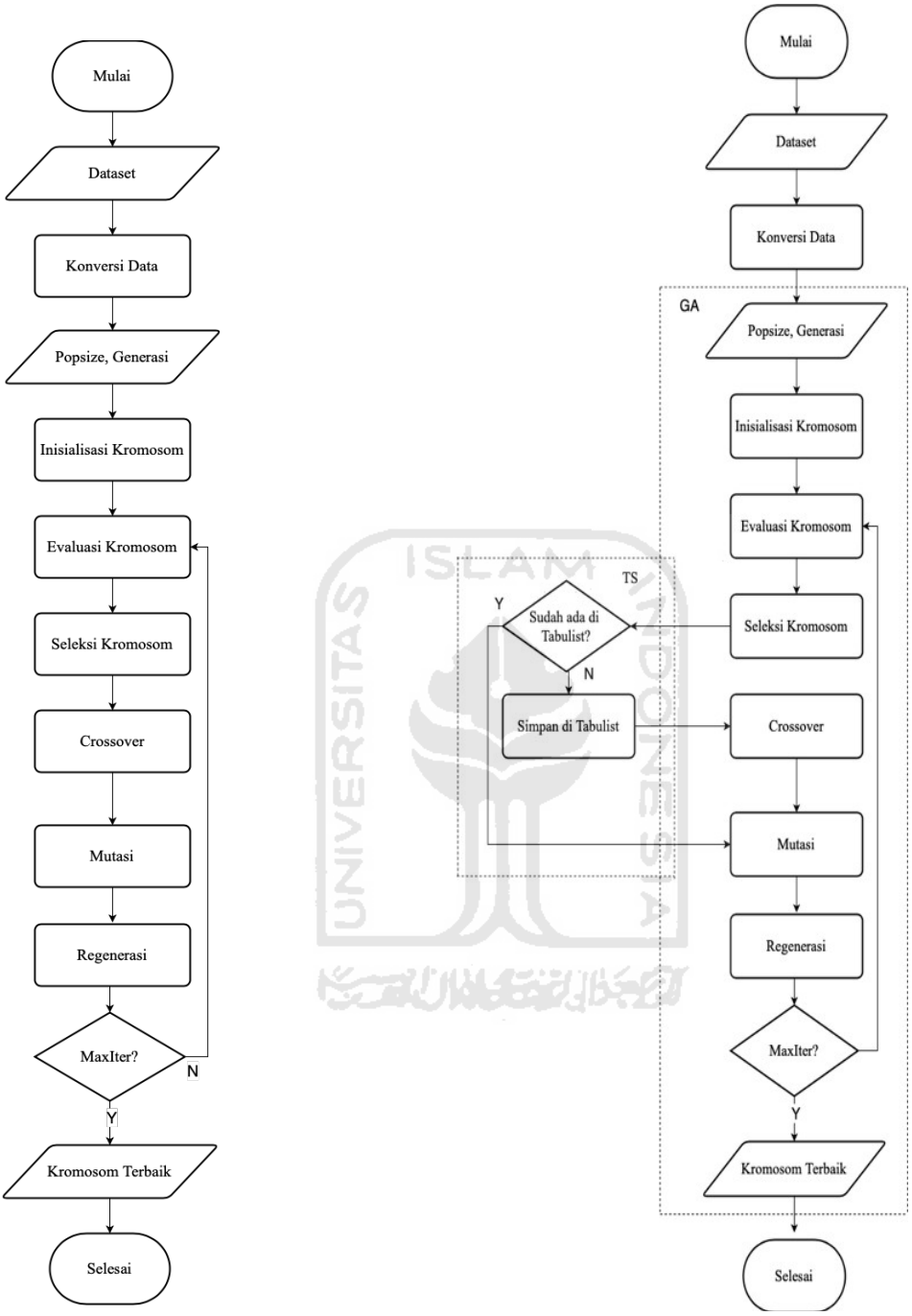
Sistem ini tidak menggunakan *database*, data yang diambil merupakan beberapa *dataset* untuk TSP yang dapat diakses melalui internet. Data yang dibutuhkan berupa koordinat titik setiap lokasi. Proses penyelesaian TSP ini akan dijalankan menggunakan kombinasi Algoritma Genetika dan Tabu *Search*.

3.3.2 *Flowchart* Sistem

Pada penelitian ini, sebelum dilakukannya implementasi sistem akan dibuat sebuah rancangan sistem. Rancangan sistem berguna untuk memberikan gambaran secara umum tentang sistem yang akan dibuat, seperti mengetahui alur kerja sistem. Rancangan sistem yang dibuat dalam penelitian menggunakan diagram *flowchart* (alir) dan diagram tersebut dibuat sebanyak dua yaitu *flowchart* Algoritma Genetika (GA) dan juga *flowchart* setelah dilakukan hibridisasi terhadap Algoritma Genetika dan Tabu *Search* (GA-TS).

Perbedaan antara kedua metode tersebut ialah pada hibridisasi GA-TS tujuannya agar hasil yang didapatkan lebih optimal atau lebih baik dibandingkan dengan GA. Dikarenakan pada GA hasil tidak terlalu bervariatif sehingga diharapkan dengan hibridisasi GA-TS dapat membuat hasil yang lebih bervariatif lagi. Agar tujuan tersebut tercapai bentuk dari sebuah hibrididasi dengan menggabungkan salah satu langkah pada Tabu *Search* ke dalam Algoritma Genetika, langkah tersebut ialah Tabu *List.* Tabu *list* memiliki kelebihan yaitu dapat menghindari perulangan solusi yang sama pada tahapan seleksi agar hasil seleksi tersebut dapat bervariatif dan nantinya akan digunakan untuk proses selanjutnya yaitu *crossover.*

Selanjutnya akan dijelaskan lebih *detail* mengenai setiap proses yang terdapat pada *flowchart*. Gambar 3.2 (bagian kiri) merupakan *flowchart* Algoritma Genetika sebelum digabungkan atau hibridisasi, sedangkan Gambar 3.2 (bagian kanan) merupakan *flowchart* rancangan sistem setelah Algoritma Genetika digabungkan dengan Tabu *Search*. Perubahan yang terjadi dari Algoritma Genetika menjadi GA-TS ialah pada tahapan setelah seleksi kromosom, terdapat dua blok utama pada rancangan sistem (bagian kanan), yaitu blok GA dan TS. Dari *flowchart* (bagian kanan) menunjukkan bahwa blok TS merupakan bagian dari blok GA. Hal ini menunjukkan bahwa proses hibridisasi antara GA dan TS dilakukan dengan menjadikan TS sebagai salah satu tahapan di dalam GA.



Gambar 3.1 *Flowchart* Sistem

Langkah pertama yang dilakukan dalam sistem ini ialah memasukkan data kemudian mengkonversi data yang awalnya berupa koordinat !,# menjadi data jarak antar kota/titik. Cara

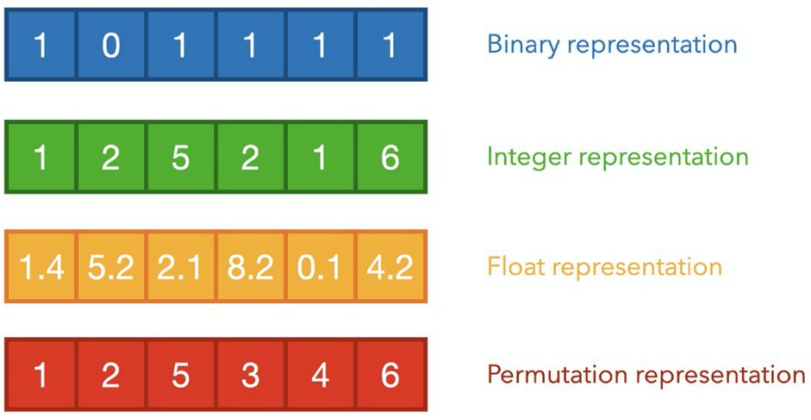
mengkonversi data yaitu dengan menggunakan rumus pada Persamaan 3.1 dengan $!_1$, $\#_1$ ialah koordinat kota/titik awal dan $!_2$, $"_2$ koordinat kota/titik akhir.

$$0.1.2 = \overline{4(\$_2 - \$_1)^2 + (\,"_2 - "_1\,)^2} \qquad (3.1)$$

Kemudian setelah jarak didapatkan maka *user* memasukkan ukuran populasi (Popsize), maksimum iterasi (generasi). Setelah itu Langkah dilakukan sesuai dengan urutan berikut ini :

a. Inisialisasi Kromosom

Kromosom mengandung informasi solusi dari sekian banyak kemungkinan solusi masalah yang dihadapi. Dalam permasalahan TSP, kromosom ditunjukkan sebagai rute perjalanan atau lebih jelasnya ialah dalam satu individu terdiri dari beberapa gen yang direpresentasikan dalam bentuk angka-angka, dan setiap angka tersebut mewakili titik-titik (tempat) tujuan yang akan dilalui. Pada Gambar 3.2 merupakan contoh ilustrasi representasi kromosom atau inisialisasi kromosom, pada ilustrasi tersebut untuk kasus penyelesaian TSP metode yang digunakan ialah representasi permutasi (*permutation encoding*). Metode tersebut menunjukan bahwa sebuah kromosom terdiri dari angka yang menunjukan sebuah posisi dalam suatu urutan. Dan pada Gambar 3.3 merupakan contoh ilustrasi inisialisasi kromosom pada kasus TSP, angka dari kromosom tersebut menunjukkan rute atau urutan perjalanan.



Gambar 3.2 Ilustrasi Representasi Kromosom atau Inisialisasi Kromosom

Sumber: (Salman, 2020)

Kromosom 1

1	3	2	4	5

Kromosom 2

1	3	2	5	4

Kromosom 3

4	1	2	3	5

Kromosom 4

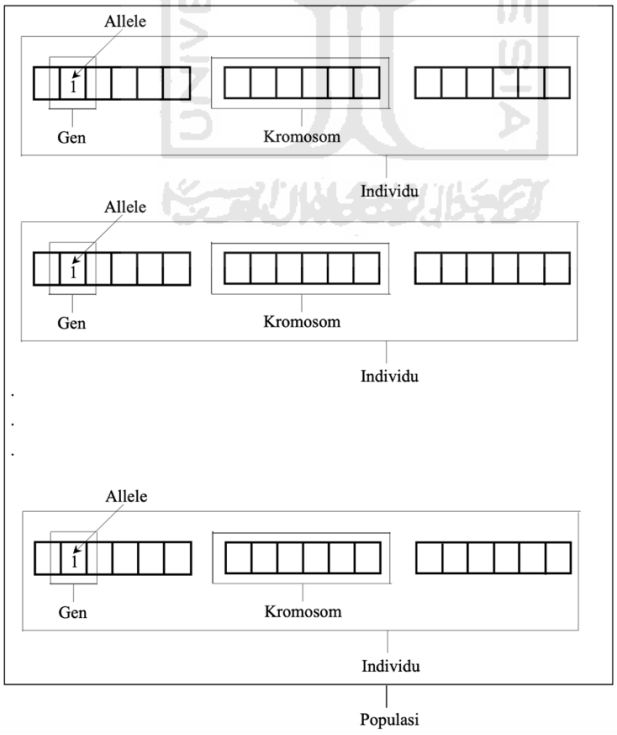
2	4	3	5	1

Kromosom 5

5	2	1	3	4

Gambar 3.3 Ilustrasi Inisialisasi Kromsom pada Kasus TSP

Pada Gambar 3.4 merupakan ilustrasi dari sebuah populasi. Populasi terdiri dari beberapa individu, individu berisi sebuah solusi yang mungkin dalam suatu permasalahan. Kemudian kromosom berisi gen-gen yang memiliki nilai (*allele*).



Gambar 3.4 Ilustrasi Suatu Populasi

Sumber: (Melorose et al., 2015)

b. Evaluasi Kromosom

Setelah kromosom terbentuk maka setiap kromosom dilakukan perhitungan nilai *fitness* seperti pada persamaan 2.1.

c. Seleksi Kromosom

Seleksi yang dipilih dalam penelitian ini ialah *Rank Base Selection*. Nilai *fitness* yang telah didapatkan akan diurutkan berdasarkan nilai terbesar hingga terkecil, kemudian urutan yang telah didapat akan diseleksi menggunakan *elitism* yaitu menyimpan 2 kromosom (*parent*) dengan nilai yang baik. Hasil seleksi kromosom tersebut akan digunakan untuk proses selanjutnya.

d. Cek *Tabulist*

Cek *tabulist* memiliki peranan penting dalam kombinasi GA-TS yaitu agar tidak didapatkannya solusi yang sama dikarenakan oleh perulangan. Dan solusi tersebut akan digunakan pada tahap selanjutnya. Setelah hasil seleksi didapatkan maka untuk iterasi pertama kromosom akan disimpan pada *tabulist*, dan untuk iterasi selanjutnya kromosom akan dicek terlebih dahulu menggunakan *tabulist*. Apabila kromosom sudah terdapat pada *tabulist*, maka kromosom akan dimutasi untuk menghindari perulangan, jika belum ada maka akan disimpan dan dilakukan *crossover*.

e. Crossover

Pada *crossover,* dua kromosom induk akan mengalami proses kawin silang. Pada tahap ini kromosom tidak dapat memiliki gen atau kota yang sama untuk itu dapat menggunakan metode *ordered crossover* dengan memilih secara acak beberapa gen di induk 1 kemudian sisa gen pada induk 1 akan diisi oleh induk 2 dengan persyaratan yaitu gen tidak boleh sama. Proses *crossover* akan menghasilkan *offspring/children* baru. Pada kasus TSP hasil dari kawin silang / *crossover* 2 induk hanya akan menghasilkan satu anak / *offspring.* Gambar 3.5 merupakan ilustrasi *ordered crossover* pada kasus TSP, blok berwarna hijau pada *parent1* ialah beberapa gen yang terpilih untuk mengisi *offspring,* kemudian gen sisanya akan diisi oleh gen pada *parent2* yang belum terdapat pada *offspring* tersebut.

Parent1

1	3	2	5	4

Parent2

3	4	1	5	2

Offspring

1	3	2	4	5

Gambar 3.5 Ilustrasi *Ordered Crossover*

f. Mutasi

Setelah terjadinya *crossover* ataupun pengecekan *tabulist* kromosom akan dimutasi menggunakan metode *reciprocal exchange mutation* dengan memilih dua posisi secara *random* kemudian menukarkan kedua nilai tersebut. Gambar 3.6 merupakan ilustrasi dari *reciprocal exchange mutation,* dengan blok berwarna hijau merupakan gen yang akan ditukar posisinya (bagian atas), kemudian hasil dapat dilihat pada bagian bawah yaitu blok hijau merupakan hasil penukaran antara dua gen dan didapatkan sebuah rute baru yang akan ditempatkan pada generasi baru.

Sebelum dimutasi

1	3	2	4	5

Setelah dimutasi

1	4	2	3	5

Gambar 3.6 Ilustrasi *Reciprocal Exchange Mutation*

g. Regenerasi

Setelah proses mutasi selesai, maka hasil dari proses tersebut menjadi populasi yang baru atau generasi baru. Kemudian generasi tersebut akan diproses kembali dari tahapan pertama hingga akhir.

h. Kromosom Terbaik

Proses akan berhenti, jika iterasi yang ditetapkan telah selesai.

Contoh Penyelesaian TSP dengan GA-TS

Tabel 3.2 merupakan contoh *dataset* yang digunakan, data tersebut memiliki koordinat titik !,#.

Tabel 3.2 Contoh *Dataset*

Kota	x	y
1	20	-10
2	10	15
3	-40	60
4	72	10
5	50	50
6	68	38

Pertama yang dilakukan ialah mengubah koordinat-koordinat menjadi satuan jarak dengan menggunakan rumus pada Persamaan 3.1.Dan hasil dari konversi tersebut tercantum pada Tabel 3.3.

Tabel 3.3 Contoh Hasil Konversi Data

	1	2	3	4	5	6
1	0	26,9	95,1	55,7	67,08	67,8
2	26,9	0	67,2	62,2	53,1	62,3
3	95,1	67,2	0	122,6	90,5	110,21
4	55,7	62,2	122,6	0	45,6	28,2
5	67,08	53,1	90,5	45,6	0	21,6
6	67,8	62,3	110,21	28,2	21,6	0

Pada Tabel 3.3 merupakan contoh hasil konversi data, yang semula berupa koordinat titik !,# berubah menjadi jarak antar titik. Dapat dilihat bahwa jarak antar titik A-B dan B-A adalah 26,9 dan begitu seterusnya. Pada kasus ini jumlah generasi dan jumlah populasi menggunakan GA-TS ialah 5. Pertama yang akan dilakukan ialah inisialisasi kromosom secara *random,* Pada Tabel 3.4 merupakan inisialisasi kromosom serta perhitungan nilai *fitness* masing-masing kromosom menggunakan Persamaan 2.1, pada kolom pertama merupakan representasi kromosom.

Populasi Awal:

Tabel 3.4 Contoh Inisialisasi Kromosom

Kromosom	**Jarak**	**Fitness**
Kromosom 1	388,99	$2{,}5707 \times 10^{-3}$

(5-1-2-4-3-6)		
Kromosom 2 (1-4-5-3-6-2)	364,31	$2{,}7449 \times 10^{-3}$
Kromosom 3 (3-1-2-5-6-4)	224,9	$4{,}4464 \times 10^{-3}$
Kromosom 4 (3-5-2-1-4-6)	254,4	$3{,}9308 \times 10^{-3}$
Kromosom 5 (2-1-3-5-4-6)	286,3	$3{,}4928 \times 10^{-3}$

Setelah kromosom terbentuk maka dilakukan seleksi dengan metode *Rank Based Selection* (Tabel 3.5) dengan mengurutkan kromosom berdasarkan nilai *fitness* terbesar hingga terkecil. Kemudian menjadikan dua kromosom sebagai induk berdasarkan nilai *fitness* tertinggi atau terbaik menggunakan *elitism* (Tabel 3.6).

Tabel 3.5 Seleksi Kromosom dengan *Rank Based Selection*

Kromosom	**Jarak**	**Fitness**
(3-1-2-5-6-4)	224,9	$4{,}4464 \times 10^{-3}$
(3-5-2-1-4-6)	254,4	$3{,}9308 \times 10^{-3}$
(2-1-3-5-4-6)	286,3	$3{,}4928 \times 10^{-3}$
(1-4-5-3-6-2)	364,31	$2{,}7449 \times 10^{-3}$
(5-1-2-4-3-6)	388,99	$2{,}5707 \times 10^{-3}$

Tabel 3.6 Kromosom Induk

Induk 1	3	1	2	5	6	4
Induk 2	3	5	2	1	4	6

Setelah terbentuk kromosom induk, maka dua kromosom tersebut akan disimpan atau dicek pada *tabulist*. Pada Tabel 3.7 merupakan ilustrasi penyimpanan kromosom pada *tabulist*.

Tabel 3.7 Ilustrasi *Tabulist*

Tabulist
3-1-2-5-6-4
3-5-2-1-4-6

Kemudian akan dilakukan *ordered crossover* jika kromosom belum ada pada *tabulist*, pada Tabel 3.8 merupakan contoh *ordered crossover* yang akan menghasilkan satu *children* yaitu dengan memilih secara random bagian dari induk 1, kemudian diisi pada kromosom *children* 1 dan sisanya akan diisi oleh induk 2 namun tidak boleh ada yang sama.

Tabel 3.8 Contoh *ordered crossover*

Induk 1	3	1	2	5	6	4
Induk 2	3	5	2	1	4	6

Children 1	3	1	2	5	4	6

Setelah proses *crossover* atau jika kromosom ada pada *tabulist*, maka hasil *crossover* atau *children* akan melewati proses mutasi menggunakan metode *reciprocal exchange mutation* dengan probabilitas yang ditentukan. Pada Gambar 3.9 merupakan contoh *reciprocal exchange mutation,* yaitu dengan memilih dua posisi (*exchange point* / XP) secara acak kemudian menukarkan nilai pada posisi tersebut.

Tabel 3.9 Contoh *Reciprocal Exchange Mutation*

		XP 1 ↓			XP 2 ↓	
Children	3	1	2	5	4	6
Mutation	3	4	2	5	1	6

Children akan menjadi populasi baru pada generasi berikutnya dan mengulang dari proses seleksi hingga generasi sesuai dengan yang telah ditentukan atau telah mencapai nilai optimal. Tabel 3.10 merupakan contoh populasi baru dengan menggantikan kromosom terburuk menjadi hasil dari proses *crossover* dan mutasi.

Tabel 3.10 Populasi Baru

Kromosom	**Jarak**	**Fitness**
(3-1-2-5-6-4)	224,9	$4{,}4464 \times 10^{-3}$
(3-5-2-1-4-6)	254,4	$3{,}9308 \times 10^{-3}$
(2-1-3-5-4-6)	286,3	$3{,}4928 \times 10^{-3}$

(1-4-5-3-6-2)	364,31	$2,7449 \times 10^{-3}$
(3-4-2-5-1-6)	372,78	$2,6825 \times 10^{-3}$

3.4 Implementasi Sistem

Setelah proses perancangan selesai dibuat, tahapan selanjutnya yang akan dilakukan ialah tahap implementasi sistem. Tahapan ini dibuat dengan berdasarkan perancangan sistem serta dapat mewujudkan tujuan yang akan dicapai.

3.5 Skenario Pengujian

Skenario Pengujian yang akan diterapkan pada penelitian ini ialah pengujian performa sistem. Pengujian performa yang dilakukan adalah analisis waktu pemrosesan sistem serta hasil akhir. Beberapa faktor dapat mempengaruhi waktu pemrosesan yaitu kecepatan prosesor, kapasitas RAM, generasi serta jumlah populasi. Rancangan pengujian ini dapat dilihat pada tabel 3.10.

Tabel 3.11 Skenario Pengujian Performa

Dataset	**Generasi**	**Rata-Rata Hasil Akhir (Jarak)**		**Rata-Rata Waktu Pemrosesan**	
		GA	GA-TS	GA	GA-TS
	50				
	100				
	500				
	1000				
	50				
	100				
	500				
	1000				

Setelah dilakukan pengujian performa sistem, hasil pengujian tersebut akan dilakukan digunakan untuk T-Test yaitu sebuah pengujian untuk menguji kebenaran atau kepalsuan sebuah hipotesis.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Penyelesaian *Travelling Salesman Problem* pada sistem yang dibuat ialah menggabungkan Algoritma Genetika dengan Tabu *Search*. Adapun masukan yang diperlukan oleh sistem yaitu berupa data, ukuran populasi, dan jumlah generasi. Dan masukan tersebut akan diproses oleh sistem sehingga sistem dapat menampilkan rute terbaik dari sebuah data.

4.2 Batasan Implementasi

Dalam pembuatan sistem *Travelling Salesman Problem* dengan kombinasi Algoritma Genetika dan Tabu *Search*, penelitian ini memiliki batasan implementasi berupa asumsi-asumsi sebagai berikut :

a. *Dataset* yang diambil berupa data TSP dengan kasus simetris.
b. Data berisi titik koordinat !, #.
c. Algoritma yang digunakan ialah kombinasi Algoritma Genetika dan Tabu *Search*.
d. *User* telah menentukan ukuran populasi.
e. *User* telah menentukan jumlah generasi.
f. Rute akan kembali pada titik awal.
g. Inisialisasi rute dibuat secara acak.

4.3 Hasil Implementasi Sistem

a. Pengumpulan Data

Data yang diambil merupakan *dataset* untuk TSP kasus simetris yang dapat diakses melalui tautan yang telah disampaikan pada metodologi. Data diambil memiliki ukuran yang bervariasi, dari kecil ke besar. Serta data yang digunakan berformat CSV dan Tabel 4.1 merupakan data yang digunakan.

Tabel 4.1 *Dataset* yang Digunakan

Dataset	***Size***	**Nilai Optimal**
Eli51	51 titik	426
Eli76	76 titik	538
Ch130	130 titik	6110

Lin318	318 titik	42029

b. Memasukkan Data dan Konversi Data

Tahapan awal setelah mendapatkan data ialah data di *import* ke sistem, kemudian sistem akan mengkonversi data yang awalnya berupa koordinat titik !,# menjadi data jarak antar kota atau titik. Data yang dimasukkan pada sistem diinisialisasi menjadi cityList. Pada Gambar 4.1 merupakan contoh kode membaca *dataset* dan Gambar 4.2 ialah contoh cara konversi data.

```
city= []
city = pd.read_csv("dataset.csv", header=None , sep=' ')
print(city.head(5))
```

Gambar 4.1 Contoh Kode Membaca *Dataset*

```
def distance(self, city):
    distance = np.sqrt((abs(self.x - city.x)** 2) +
                       (abs(self.y - city.y) ** 2))
    return distance
```

Gambar 4.2 Contoh Kode Konversi Data

c. Memasukkan Nilai Populasi dan Jumlah Generasi

Setelah itu *user* memasukkan jumlah populasi (popSize) dan jumlah generasi (generations) yang diinginkan. Pada Gambar 4.3 merupakan gambaran masukkan pada sistem yaitu *popSize* dan *generations*. Dapat diartikan bahwa pada populasi bernama *cityList* akan dibuat sebuah populasi sebanyak 10 dan akan membuat generasi / terjadi perulangan sebanyak 100 kali.

```
geneticAlgorithm(population=cityList, popSize=10, generations=100)
```

Gambar 4.3 Contoh Gambaran Masukan pada Sistem

d. Inisialisasi dan Evaluasi Kromosom

Berdasarkan popSize yang dimasukkan, maka sistem akan membuat populasi secara *random* sebanyak popSize, populasi tersebut berupa rute perjalanan. Dan setiap kromosom atau rute akan dievaluasi sehingga menghasilkan sebuah nilai *fitness*. Gambar 4.4 merupakan kode cara evaluasi atau menghitung nilai *fitness* setiap rute. Dan pada Gambar 4.5 merupakan tampilan hasil inisialisasi kromosom atau rute beserta dengan total jarak dan nilai *fitness*.

```
def routeFitness(self):
    if self.fitness == 0:
        self.fitness = 1 / float(self.routeDistance())
    return self.fitness
```

Gambar 4.4 Kode Evaluasi atau Menghitung Nilai *Fitness*

```
populasi ke-0 ([13, 15, 2, 8, 3, 4, 6, 7, 10, 12, 14, 11, 1, 9, 5]) Jarak
= 519.3567319577853 Fitness = 0.0019254588194714739
populasi ke-1 ([13, 4, 6, 11, 9, 8, 7, 10, 1, 15, 5, 3, 14, 12, 2]) Jarak
= 536.7054531397602 Fitness = 0.0018632193769411846
populasi ke-2 ([5, 8, 15, 3, 11, 12, 10, 14, 7, 9, 6, 4, 1, 13, 2]) Jarak
= 586.9471547126543 Fitness = 0.0017037308929277624
populasi ke-3 ([2, 4, 12, 6, 1, 7, 3, 13, 14, 10, 9, 5, 15, 11, 8]) Jarak
= 622.2202708413519 Fitness = 0.0016071478973962439
populasi ke-4 ([7, 4, 1, 13, 8, 5, 2, 9, 11, 14, 10, 3, 6, 12, 15]) Jarak
= 641.6880437700239 Fitness = 0.00155838964074324
populasi ke-5 ([6, 15, 4, 2, 5, 8, 7, 3, 13, 9, 12, 10, 14, 11, 1]) Jarak
= 651.3204989326175 Fitness = 0.0015353424337769157
populasi ke-6 ([11, 6, 3, 8, 4, 12, 7, 1, 5, 10, 2, 9, 14, 15, 13]) Jarak
= 663.738874732362 Fitness = 0.0015066165898497776
populasi ke-7 ([2, 7, 1, 13, 8, 9, 15, 12, 3, 11, 10, 6, 14, 5, 4]) Jarak
= 671.6556773661644 Fitness = 0.0014888581064652167
populasi ke-8 ([2, 3, 8, 11, 13, 7, 14, 10, 5, 12, 9, 4, 15, 1, 6]) Jarak
= 713.1107156574363 Fitness = 0.0014023067919798015
populasi ke-9 ([10, 1, 7, 8, 15, 14, 4, 12, 13, 9, 3, 2, 11, 6, 5]) Jarak
= 738.4551327054803 Fitness = 0.0013541784134349462
```

Gambar 4.5 Contoh Tampilan Inisialisasi dan Evaluasi Kromosom

e. Seleksi Kromosom

Seleksi pada sistem ini ialah mengurutkan kromosom atau rute berdasarkan nilai fitness terbesar hingga terkecil menggunakan metode *Rank Based Selection* dan mengambil atau mempertahankan 2 kromosom atau rute terbaik dengan metode *elitism.* Pada Gambar 4.6 merupakan contoh kode mengambil 2 kromosom terbaik menggunakan *elitism*, dan Gambar 4.7 merupakan contoh hasil seleksi kromosom atau rute.

```
def Selection(population):
    selection = []
    # mengambil 2 individu terbaik berdasarkan nilai fitness
    for i in range(0,2):
        selection.append(population[i])
        print("seleksi ke-" + str(i) + " " + str(Fitness(selection[i]).printKota()) +
              " Jarak = " + str(Fitness(selection[i]).routeDistance()) + " Fitness = " + str(
            Fitness(selection[i]).routeFitness()))
    return selection
```

Gambar 4.6 Contoh Kode Mengambil 2 Kromosom Terbaik (Hasil Seleksi)

```
seleksi ke-0 ([13, 15, 2, 8, 3, 4, 6, 7, 10, 12, 14, 11, 1, 9, 5]) Jarak = 519.3567319577853 Fitness = 0.001925458819
4714739
seleksi ke-1 ([13, 4, 6, 11, 9, 8, 7, 10, 1, 15, 5, 3, 14, 12, 2]) Jarak = 536.7054531397602 Fitness = 0.001863219376
9411846
```

Gambar 4.7 Contoh Hasil Seleksi Kromosom atau Rute

f. Cek *Tabulist*

Hasil Seleksi akan dilakukan pengecekan pada *tabulist*, jika hasil seleksi belum ada pada *tabulist* maka akan disimpan terlebih dahulu pada tabulist sebelum *crossover*. Namun jika hasil seleksi sudah terdapat pada *tabulist*, maka tahapan *crossover* tidak dilakukan dan hasil seleksi pertama atau yang terbaik akan dimutasi. Pada Gambar 4.8 merupakan contoh tampilan isi *tabulist*. Pada gambar 4.9 merupakan contoh kode fungsi *tabulist* pada sistem. Dan pada Gambar 4.11 merupakan contoh jika *tabulist false* (tidak terdapat pada *tabulist*).

```
TABULIST :  [[(13), (15), (2), (8), (3), (4), (6), (7), (10), (12), (14), (11), (1), (9),
(5)], [(13), (4), (6), (11), (9), (8), (7), (10), (1), (15), (5), (3), (14), (12), (2)]]
```

Gambar 4.8 Tampilan *Tabulist*

```
def Tabulist(selectionResults,tabulist):
    isTabulist1 = False
    isTabulist2 = False
    isTabulist = False
    #cek
    for j in range(0,len(tabulist)):
        if selectionResults[0] == tabulist[j]:
            isTabulist1 = True
        if selectionResults[1] == tabulist[j]:
            isTabulist2 = True

    if isTabulist1 == False:
        tabulist.append(selectionResults[0])
    if isTabulist2 == False:
        tabulist.append(selectionResults[1])

    if isTabulist1 == True and isTabulist2 == True:
        isTabulist = True

    return isTabulist,tabulist
```

Gambar 4.9 Contoh Kode Fungsi *Tabulist*

g. Crossover

Metode *crossover* yang digunakan pada sistem ini ialah metode *ordered crossover.* Pemilihan letak gen pada induk pertama yang akan di *crossover* dilakukan secara acak oleh sistem. Kemudian hasil pemilihan tersebut, gen yang masih belum terisi akan diisi oleh induk kedua seperti pada Gambar 4.10 merupakan contoh kode proses *crossover.* Pada Gambar 4.11 merupakan contoh tampilan hasil *crossover*, *startgene* merupakan gen awal dan *endgene* merupakan gen akhir, dan dapat dilihat bahwa pada induk pertama gen pada nomor 2 hingga 12 akan dipilih untuk di *crossover,* kemudian akan menampilkan hasil dari proses *crossover.*

```
for item in parent2:
    if item not in childP1:
        childP2.append(item)

for i in range(0,len(childP2)):
    if i==startGene:
        for j in range(startGene,endGene):
            child.append(parent1[j])
    child.append(childP2[i])
```

Gambar 4.10 Contoh Kode dalam Proses *Crossover*

```
stargene = 2, endGene = 12
Tabulis False
==========================================================================================>
hasil crossover = [(13), (9), (2), (8), (3), (4), (6), (7), (10), (12), (14), (11), (1), (15), (5)]
```

Gambar 4.11 Tampilan Hasil *Crossover* (*Tabulist False*)

h. Mutasi

Jika *tabulist true* ataupun terdapat hasil *crossover,* hasil tersebut akan dimutasi menggunakan metode *reciprocal exchange mutation.* Sistem akan memilih secara acak 2 gen yang akan ditukar pada suatu individu. Dan pada Gambar 4.12 merupakan contoh kode proses mutasi serta Gambar 4.13 merupakan tampilan hasil dari proses mutasi, *swapped* merupakan letak gen yang akan ditukar dengan letak gen *swap with*.

```
def Mutate(individual):
    #mengacak nomor gen yang akan di tukar
    swapped = int(random.random() * len(individual))
    swapWith = int(random.random() * len(individual))
    ## melakukan perulangan agar tidak muncul nilai sama
    while swapped == swapWith:
        swapWith = int(random.random() * len(individual))

    city1 = individual[swapped]
    city2 = individual[swapWith]

    print("swapped = "+str(swapped) + " , swap with = " + str(swapWith))
    individual[swapped] = city2
    individual[swapWith] = city1
    return individual
```

Gambar 4.12 Contoh Kode Proses Mutasi

```
swapped = 11 , swap with = 13
hasil mutasi = [(13), (9), (2), (8), (3), (4), (6), (7), (10), (12), (14), (15), (1), (11), (5)]
```

Gambar 4.13 Tampilan Hasil Mutasi

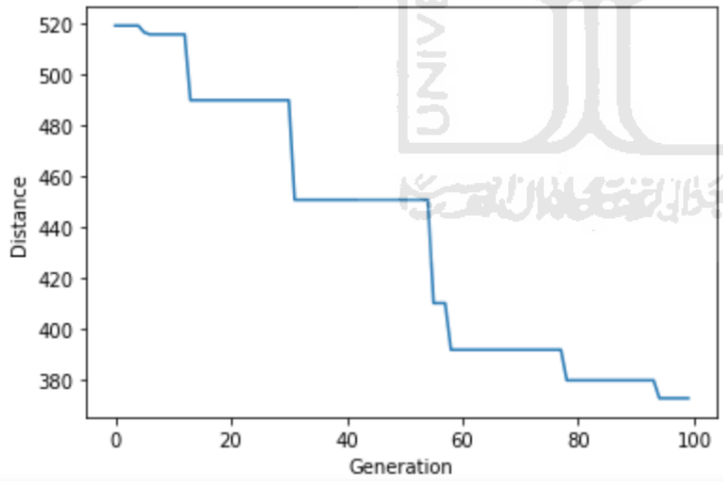
i. Regenerasi dan Hasil Akhir

Regenerasi dilakukan dengan menggantikan nilai terendah pada populasi menjadi hasil dari mutasi. Gambar 4.14 merupakan contoh hasil dari regenerasi. Jika generasi telah mencapai nilai maksimum maka sistem akan berhenti dan memberikan keluaran berupa individu terbaik beserta dengan jarak dan nilai *fitness* individu tersebut serta menampilkan diagram garis berdasarkan nilai generasi dan jarak (Gambar 4.15).

```
Generasi baru ke-0 ([13, 15, 2, 8, 3, 4, 6, 7, 10, 12, 14, 11, 1, 9, 5]) Jarak = 519.3567319577853 Fitness = 0.001925
4588194714739
Generasi baru ke-1 ([13, 4, 6, 11, 9, 8, 7, 10, 1, 15, 5, 3, 14, 12, 2]) Jarak = 536.7054531397602 Fitness = 0.001863
2193769411846
Generasi baru ke-2 ([5, 8, 15, 3, 11, 12, 10, 14, 7, 9, 6, 4, 1, 13, 2]) Jarak = 586.9471547126543 Fitness = 0.001703
7308929277624
Generasi baru ke-3 ([2, 4, 12, 6, 1, 7, 3, 13, 14, 10, 9, 5, 15, 11, 8]) Jarak = 622.2202708413519 Fitness = 0.001607
1478973962439
Generasi baru ke-4 ([7, 4, 1, 13, 8, 5, 2, 9, 11, 14, 10, 3, 6, 12, 15]) Jarak = 641.6880437700239 Fitness = 0.001558
38964074324
Generasi baru ke-5 ([6, 15, 4, 2, 5, 8, 7, 3, 13, 9, 12, 10, 14, 11, 1]) Jarak = 651.3204989326175 Fitness = 0.001535
3424337769157
Generasi baru ke-6 ([11, 6, 3, 8, 4, 12, 7, 1, 5, 10, 2, 9, 14, 15, 13]) Jarak = 663.738874732362 Fitness = 0.0015066
165898497776
Generasi baru ke-7 ([2, 7, 1, 13, 8, 9, 15, 12, 3, 11, 10, 6, 14, 5, 4]) Jarak = 671.6556773661644 Fitness = 0.001488
8581064652167
Generasi baru ke-8 ([2, 3, 8, 11, 13, 7, 14, 10, 5, 12, 9, 4, 15, 1, 6]) Jarak = 713.1107156574363 Fitness = 0.001402
3067919798015
Generasi baru ke-9 ([13, 9, 2, 8, 3, 4, 6, 7, 10, 12, 14, 15, 1, 11, 5]) Jarak = 584.6131834863276 Fitness = 0.001710
```

Gambar 4.14 Contoh Tampilan Hasil Regenerasi

```
BEST INDIVIDU = ([15, 9, 7, 3, 12, 14, 4, 6, 10, 8, 2, 11, 1, 13, 5])
BEST DISTANCE = 372.4049389061262
BEST FITNESS = 0.0026852490274090444
```



Gambar 4.15 Tampilan Hasil Akhir Sistem

4.4 Hasil Pengujian Sistem

4.4.1 Pengujian Performa Sistem

Pengujian performa yang dilakukan ialah dengan menganalisis hasil akhir berupa jarak serta waktu pemrosesan yang diperlukan pada sistem. Percobaan dilakukan dengan menggunakan nilai *popSize* (jumlah populasi) sebanyak tiga ukuran yaitu sebesar 10, 15, dan

20 populasi serta generasi sebanyak lima ukuran, 50, 100, 500, 1000, dan 3000 generasi. Setiap percobaan tersebut akan dilakukan sebanyak tiga kali dengan *sample/*inisialisasi dilakukan *pure random*, kemudian hasil akhir serta waktu pemrosesan akan dihitung rata-ratanya. Beberapa faktor dapat mempengaruhi waktu pemrosesan sehingga hasilnya akan berbeda-beda tergantung dengan kecepatan prosessor, serta besaran masukan pada sistem. Dapat dilihat pada Tabel 4.2, Tabel 4.3, Tabel 4.4 merupakan hasil pengukuran performasi dengan *dataset* yang ukurannya berbeda-beda dan populasi yang beda-beda. Berikut jabaran hasil dari pengukuran performasi sistem.

Tabel 4.2 Hasil Pengukuran Performansi Algoritma dengan Jumlah Populasi Sebesar 10

Dataset	**Generasi**	**Rata-Rata Hasil Akhir (Jarak)**		**Rata-Rata Waktu Pemrosesan (detik)**	
		GA	GA-TS	GA	GA-TS
ELI51	50	1326,338	1349,77	1,56	1,88
	100	1216,981	1141,13	3	3,63
	500	932,58	833,255	14,53	18,15
	1000	781,298	733,173	28,83	36,8
	3000	689,589	635,461	85,3	113,3
Eli76	50	2108,281	2083,05	2,24	2,63
	100	1920,766	1846,974	4,23	5,12
	500	1546,726	1316,11	20,7	26,23
	1000	1417,24	1209,344	41,66	53,8
	3000	1097,458	1006,374	124	173
CH130	50	39915,07	38207,594	2,9	4,04
	100	36862,861	37143,682	5,59	8,21
	500	29661,153	29062,194	27,9	42,43
	1000	26456,488	23519,764	56,13	88,66
	3000	21970,072	18359,626	167	199,66
LIN318	50	538957,622	539497,538	7,18	8,66
	100	510661,49	510483,83	13,9	17,1
	500	436214,799	417665,936	69,33	84,6
	1000	379425,505	361070,343	139,33	168,33
	3000	313439,34	269428,56	416	498

Tabel 4.3 Hasil Pengukuran Performansi Algoritma dengan Jumlah Populasi Sebesar 15

Dataset	**Generasi**	**Rata-rata Hasil Akhir (Jarak)**		**Rata-Rata Waktu Pemrosesan (detik)**	
		GA	GA-TS	GA	GA-TS
ELI51	50	1284,268	1292,21	2,46	2,96
	100	1172,208	1177,86	4,76	5,85
	500	942,52	845,665	23,46	28,9
	1000	827,643	782,522	46,6	57,8
	3000	665,298	617,467	140	170,66
Eli76	50	2079,562	1979,143	3,54	4,26
	100	1879,649	1785,162	6,92	8,24
	500	1565,61	1370,47	35,6	40,66
	1000	1333,956	1214,988	69,6	81,3
	3000	1111,473	943,234	208,33	244,33
CH130	50	39779,611	39294,393	5,7	6,63
	100	36661,658	36292,06	11,2	13,3
	500	29249,53	27318,49	56,23	65,33
	1000	26186,318	23149,365	110,66	130,66
	3000	20652,691	18725,48	328	393,66
LIN318	50	520087,127	536431,605	14,36	16,83
	100	510130,109	510731,569	28,7	33,13
	500	436281,02	411000,724	141,33	163,33
	1000	387869,769	352701,157	281,66	327,66
	3000	329405,706	286869,632	834	972

Tabel 4.4 Hasil Pengukuran Performansi Algoritma dengan Jumlah Populasi Sebesar 20

Dataset	**Generasi**	**Rata-rata Hasil Akhir (Jarak)**		**Rata-Rata Waktu Pemrosesan (detik)**	
		GA	GA-TS	GA	GA-TS
ELI51	50	1297,253	1258,409	4,18	4,81
	100	1202,743	1150,535	8,3	9,4
	500	926,513	839,881	40	45,63
	1000	808,643	719,406	80,33	91

	3000	732,839	644,641	241,33	274
Eli76	50	2043,203	2049,912	6,05	6,9
	100	1847,513	1880,766	12,06	13,76
	500	1561,061	1381,155	60	67,66
	1000	1401,429	1170,65	119	136
	3000	1096,628	977,053	356	404,66
CH130	50	40892,332	39761,489	9,46	11,03
	100	38591,293	36452,94	18,73	21,56
	500	30894,111	27522,804	90,33	107,33
	1000	25354,654	22686,748	182,33	218,33
	3000	20262,867	18043	547,5	649
LIN318	50	537706,85	525806,773	23,6	26,46
	100	511604,034	517046,051	46,86	52,73
	500	430362,247	409194,988	231,3	262,33
	1000	394400,614	359400,99	465	527
	3000	336367,507	276670,561	1384	1579

a. Analisis ukuran dataset

Berdasarkan hasil pengujian jika dataset yang digunakan semakin besar maka waktu yang diperlukan untuk memecahkan kasus akan semakin tinggi pula. Pada perbandingan Algoritma Genetika dan Tabu *Search* waktu pemrosesan GA lebih cenderung lebih kecil dibandingkan dengan GA-TS namun hasil yang optimal terdapat pada GA-TS, seperti pada Tabel 4.2, dataset Eli51 untuk mendapatkan hasil optimal GA sebesar 689,589 memerlukan waktu 85,3 detik, sedangkan untuk mendapatkan hasil optimal GA-TS sebesar 635,461 memerlukan waktu 113,3 detik.

b. Analisis banyaknya generasi

Sedangkan jika generasi yang digunakan besar maka hasil akhir yang didapatkan akan semakin optimal, namun terkadang juga terjebak pada lokal optimum. Pada hasil analisis pada Tabel 4.4 data Eli51 menunjukkan bahwa GA-TS pada generasi 50 jarak atau hasil yang didapatkan ialah 1258,409 dan waktu yang diperlukan ialah 4,81 detik, sedangkan pada generasi 3000 hasil yang didapatkan jauh lebih optimal yaitu 644,641 namun waktu pemrosesan pun lebih lama yaitu sebesar 274 detik.

c. Analisis jumlah populasi

Dan pada pengujian menggunakan jumlah populasi yang berbeda hasil akhir atau jarak yang didapatkan cenderung lebih baik jika jumlah populasi yang digunakan lebih besar, contoh pada Tabel 4.4 data Ch130 GA-TS di generasi 3000 hasil akhir lebih baik yaitu 18043 dibanding dengan hasil data Ch130 GA-TS di generasi 3000 pada Tabel 4.2 dan Tabel 4.3 yaitu 18359,626 dan 18725,48. Sama seperti analisis generasi, jika populasi semakin besar maka waktu yang diperlukan juga akan semakin besar pula.

d. Analisis *Time-complexity* antara GA dan GA-TS

Analasisi *time-complexity* dilakukan ialah untuk mengukur seberapa besar waktu yang diperlukan suatu algoritma dalam menyelesaikan masalah. Berikut Tabel 4.5 merupakan hasil perbandingan *time-complexity* antara Algoritma Genetika dan GA-TS.

Tabel 4.5 Perbandingan *Time-Complexity* Algoritma Genetika dan GA-TS

Algoritma Genetika	**Algoritma Genetika - Tabu *Search***
a. Inisialisasi (()) =) . , b. Evaluasi (()) =) . , c. Seleksi (()) =) -./) *d. Crossover* !(#) = #(' + ' !) e. Mutasi (()) =) . , f. Regenerasi (()) =)	a. Inisialisasi (()) =) . , b. Evaluasi (()) =) . , c. Seleksi (()) =) -./) *d. Tabulist* (()) =)(, + , #) *e. Crossover* (()) =)(, + , #) f. Mutase (()) =) . , g. Regenerasi (()) =)
3Total keseluruhan T(n) : !(#) = #. ' + *(3#' + #log# + # ' ! + #) !(#) = O(max((#. '),(3*#'),(* #log#),(*# ' !),(*#) !(#) = O(*# ' !) * = *4#45678 96#* :8;5<747 ' = =6>?@ 4>74>@78 74A6#96> B@ ' C6*h* ><?6	**Total keseluruhan T(n) :** !(#) = #' + *(4#' + #C<* # + 2# ' ! + #) !(#) = O(max((#. '),(4*#'),(* #log#),(2*# ' !),(*#)) !(#) = O(2*# ' !) * = *4#45678 96#* :8;5<747

# = A6#96>#96 ;<;@C678	▼ = =6>?@ 4>74>@78 74A6#96> B@ ▼ C6h><?6 # = A6#96>#96 ;<;@C678

Pada hasil *time-complexity* pada Tabel 4.5 terdapat perbedaan antara Algoritma Genetika dan GA-TS, pada GA-TS waktu yang diperlukan akan lebih lama dibandingkan dengan Algoritma Genetika dikarenakan oleh setelah dilakukan hibridisasi Algoritma Genetika tersebut memiliki tambahan satu langkah yaitu yang terdapat pada Tabu *Search* (*tabulist*). Guna dari mengukur *time-complexity* ialah agar dapat menentukan kecepatan peningkatan waktu yang diperlukan sebuah algoritma seiring meningkatnya ukuran masukan. Untuk menghitung *time-complexity* ini dapat menggunakan notasi big O (O) yaitu dengan mencari batas atas kompleksitas waktu suatu algoritma atau kata lainnya ialah untuk mengetahui waktu eksekusi paling lama yang mungkin terjadi. Untuk hasil *time-complexity* menggunakan notasi big O (O) menyatakan bahwa kedua algoritma ini memiliki hasil waktu eksekusi dengan jenis yang sama yaitu *quadratic.*

e. Analisis hasil optimasi

Kemudian yang terakhir ialah membandingkan hasil optimasi sistem dengan nilai optimal yang sebenarnya, pada Tabel 4.6 merupakan hasil perbandingan tersebut.

Tabel 4.6 Hasil Perbandingan Nilai Optimal

Dataset	**Size**	**Nilai Optimal Sebenarnya**	**Nilai Optimal Hasil GA-TS**
Eli51	51 titik	426	617,467
Eli76	76 titik	538	943,234
Ch130	130 titik	6110	18043
Lin318	318 titik	42029	269428,56

Berdasarkan hasil perbandingan nilai optimal, akan sulit mencapai nilai yang mendekati optimum jika data atau kota berjumlah. Namun jika data yang digunakan cenderung kecil GA-TS mampu mendapatkan nilai yang mendekati optimal.

4.4.2 T-Test

Pengujian T atau *T-Test* dilakukan dengan mengambil sampel pada hasil performa sistem dan dilakukan sebanyak jumlah populasi yang digunakan yaitu 3. Pengujian ini menggunakan SPSS, dengan hipotesis yang digunakan dalam pengujian T seperti pada Gambar 4.16.

Hipotesis:

Ho = Rata-rata antara GA dan GA-TS tidak ada perbedaan yang signifikan

H1 = Rata-rata antara GA dan GA-TS ada perbedaan yang signifikan

Jika,

Sig > 6, maka Ho diterima, H1 ditolak

Sig < 6, maka Ho ditolak, H1 diterima

Gambar 4.16 Hipotesis dalan Pengujian T

Berikut merupakan hasil dari pengujian T disetiap populasi :

a. Populasi 10

Pada Tabel 4.7 menunjukan nilai statistik dari GA dan GA-TS pada populasi 10 dengan jumlah sampel adalah 20, nilai rata-rata GA sebelum dihibridisasi ialah 117330.083, sedangkan setelah dihibridisasi dengan Tabu *Search* ialah 112829.685. Standar deviasi GA adalah 193777.983 dan GA-TS adalah 189047.475, kemudian ada standar *error* GA ialah 43330.0742 dan GA-TS ialah 42272.3005. Pada Tabel 4.8 menunjukkan tingkat hubungan variable GA dan GA-TS. Nilai korelasi GA dan GA-TS ialah 0.999 yang berarti memiliki hubungan yang sangat erat.

Tabel 4.7 Nilai Statistik GA dan GA-TS pada Populasi Sebesar 10

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	GA	117330.083	20	193777.983	43330.0742
	GA-TS	112829.685	20	189047.475	42272.3005

Tabel 4.8 Tingkat Hubungan GA dan GA-TS pada Populasi sebesar 10

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	GA & GA-TS	20	.999	<.001

Tabel 4.9 Hasil Uji T GA dan GA-TS pada Populasi Sebesar 10

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	GA - GA-TS	4500.39745	10853.8558	2426.99594	-579.36343	9580.15833	1.854	19	.079

Tabel 4.9 adalah hasil dari pengujian T. dapat dilihat bahwa ada perbedaan rata-rata GA dan GA-TS yaitu sebesar 4500.39745 dengan nilai standar deviasi 10853.8558 dan standar *error* 2426.99594. Nilai T hitung adalah 1.854 dan df = 19 maka diperoleh sig (2 *tailed*) atau p-*value* sebesar 0.079. Nilai p-*value* tersebut lebih besar dibandingkan dengan nilai alfa (0.025). Maka dapat disimpulkan bahwa tidak ada perbedaan rata-rata yang signifikan antara GA dan GA-TS.

b. Populasi 15

Pada Tabel 4.10 menunjukan nilai statistik dari GA dan GA-TS pada populasi 15 dengan jumlah sampel adalah 20, nilai rata-rata GA sebelum dihibridisasi ialah 117458.286 sedangkan setelah dihibrididasi dengan Tabu *Search* ialah 112726.160. Standar deviasi GA adalah 193172.846 dan GA-TS adalah 188426.429, kemudian ada standar *error* GA ialah 43194.7616 dan GA-TS ialah 42133.4303. Pada Tabel 4.11 menunjukkan tingkat hubungan variabel GA dan GA-TS. Nilai korelasi GA dan GA-TS ialah 0.998 yang berarti memiliki hubungan yang sangat erat.

Tabel 4.10 Nilai Statistik GA dan GA-TS pada Populasi Sebesar 15

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	GA	117458.286	20	193172.846	43194.7616
	GA-TS	112726.160	20	188426.429	42133.4303

Tabel 4.11 Tingkat Hubungan GA dan GA-TS pada Populasi Sebesar 15

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	GA & GA-TS	20	.998	<.001

Tabel 4.12 Hasil Uji T GA dan GA-TS pada Populasi Sebesar 15

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	GA - GA-TS	4732.12650	13615.0788	3044.42417	-1639.9265	11104.1795	1.554	19	.137

Tabel 4.12 adalah hasil dari pengujian T. Dapat dilihat bahwa ada perbedaan rata-rata GA dan GA-TS yaitu sebesar 4732.12650 dengan nilai standar deviasi 13615.0788 dan standar *error* 3044.42417. Nilai T hitung adalah 1.554 dan df = 19 maka diperoleh sig (2 *tailed*) atau p-*value* sebesar 0.137. Nilai p-*value* tersebut lebih besar dibandingkan dengan nilai alfa (0.025). Maka dapat disimpulkan bahwa tidak ada perbedaan rata-rata yang signifikan antara GA dan GA-TS.

c. Populasi 20

Pada Tabel 4.13 menunjukan nilai statistik dari GA dan GA-TS pada populasi 20 dengan jumlah sampel adalah 20, nilai rata-rata GA sebelum dihibridisasi ialah 118967.717 sedangkan setelah dihibrididasi dengan Tabu *Search* ialah 112232.938. Standar deviasi GA adalah 195595.626 dan GA-TS adalah 187723.901 kemudian ada standar error GA ialah 43736.5116 dan GA-TS ialah 41976.3404. Pada Tabel 4.14 menunjukkan tingkat hubungan variable GA dan GA-TS. Nilai korelasi GA dan GA-TS ialah 0.998 yang berarti memiliki hubungan yang sangat erat.

Tabel 4.13 Nilai Statistik GA dan GA-TS pada Populasi Sebesar 20

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	GA	118967.717	20	195595.626	43736.5116
	GA-TS	112232.938	20	187723.901	41976.3404

Tabel 4.14 Tingkat Hubungan GA dan GA-TS pada Populasi Sebesar 20

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	GA & GA-TS	20	.998	<.001

Tabel 4.15 Hasil Uji T GA dan GA-TS pada Populasi Sebesar 20

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	GA - GA-TS	6734.77910	15441.2732	3452.77365	-491.95921	13961.5174	1.951	19	.066

Tabel 4.15 adalah hasil dari pengujian T. Dapat dilihat bahwa ada perbedaan rata-rata GA dan GA-TS yaitu sebesar 6734.77910 dengan nilai standar deviasi 15441.2732 dan standar *error* 3452.77365. Nilai T hitung adalah 1.951 dan df = 19 maka diperoleh sig (2 tailed) atau p-*value* sebesar 0.066. Nilai p-*value* tersebut lebih besar dibandingkan dengan nilai alfa (0.025). Maka dapat disimpulkan bahwa tidak ada perbedaan rata-rata yang signifikan antara GA dan GA-TS.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Sistem untuk menyelesaikan kasus *Travelling Salesman Problem* dibuat menggunakan kombinasi Algoritma Genetika dan Tabu *Search*. Data yang digunakan pada sistem ialah *dataset* TSP dengan kasus simetris, data berisikan informasi mengenai titik koordinat x,y yang dapat diakses melalui internet. Setelah implementasi serta pengujian telah selesai dilakukan, maka dari itu dapat ditarik kesimpulan sebagai berikut :

a. Sistem mampu menampilkan hasil akhir sesuai dengan tujuan dari penelitian ini yaitu dapat menyelesaikan permasalahan TSP dengan memberikan rute terbaik dari sebuah data, namun tidak selalu dapat menghasilkan nilai optimal atau bukan solusi optimum sebenarnya.
b. Hasil akhir yang diberikan ialah berupa rute terbaik serta total jarak dari kota awal dan kembali lagi ke kota akhir dan nilai fitness rute terbaik. Serta sistem menampilkan grafik garis berdasarkan jumlah generasi dan jarak setiap generasinya di setiap percobaan.
c. Kombinasi Algoritma Genetika dan Tabu *Search* terbukti mendapatkan yang lebih optimal dibandingkan dengan Algoritma Genetika walaupun waktu yang dibutuhkan cenderung lama.
d. Sistem memerlukan waktu pemrosesan yang besar seiring bertambah besarnya jumlah masukan serta data yang digunakan.
e. Solusi akan semakin jauh dari nilai optimum jika ukuran data semakin besar.

5.2 Saran

Berdasarkan hasil implementasi serta pengujian yang telah dilakukan, pada penelitian ini masih terdapat beberapa kekurangan sehingga penelitian ini masih memerlukan perbaikan, berikut saran yang dapat berikan :

a. Melakukan penelitian yang lebih mendalam mengenai penyelesaian *Travelling Salesman Problem.*
b. Melakukan modifikasi terhadap kombinasi Algoritma Genetika dan Tabu *Search* sehingga diharapkan hasil modifikasi tersebut lebih baik.
c. Data yang digunakan merupakan kasus *real* contohnya mencari rute pengantaran paket pada sebuah kota.

DAFTAR PUSTAKA

Abdulkarim, H. A., & Alshammari, I. F. (2015). Comparison of Algorithms for Solving Traveling Salesman Problem. *International Journal of Engineering and Advanced Technology*, *4*(6), 76–79.

Albar, M. A. (2013). Algoritma Genetik Tabu Search Dan Memetika Pada Permasalahan Penjadwalan Kuliah. *Semnasteknomedia Online*, *1*(1), 18–45. Retrieved from https://ojs.amikom.ac.id/index.php/semnasteknomedia/article/view/609

Amri, F., Nababan, E. B., & Syahputra, M. F. (2012). Artificial Bee Colony Algorithm untuk Menyelesaikan Travelling Salesman Problem. *Jurnal Dunia Teknologi Informasi*, *1*(1), 8–13.

Andri, Suyandi, & WinWin. (2013). *Aplikasi Travelling Salesman Problem dengan Metode Artificial Bee Colony*. *14*(1), 59–68.

Aristi, G. (2014). Perbandingan Algoritma Greedy, Algoritma Cheapest Insertion Heuristics Dan Dynamic Programming Dalam Penyelesaian Travelling Salesman Problem. *Paradigma*, *XVI*(2), 52–58.

Darmawan, I., & Siliwangi, U. (2011). *Hibridisasi Genetic-Tabu Search Algorithm Untuk Penjadwalan*. *2011*(Snati), 17–18.

Dini, N. S. (2015). *Produk Air Minum Kemasan Galon Menggunakan Kombinasi Algoritma Genetika Dan Pencarian Tabu Di Depot Air Minum Isi Ulang Banyu Belik*.

Fatmawati, Prihandono, B., & Noviani, E. (2015). Penyelesaian Travelling Salesman Problem Dengan Metode Tabu Search. *Buletin Ilmiah Mat. Stat. Dan Terapannya (Bimaster)*, *04* no. *1*(1), 17–24.

Fernando, E. (2012). *Kombinasi Algoritma Genetik Dan Tabu Search*. *6*(1).

Firdaus Mahmudy, W. (2013). *Algoritma Evolusi*. (September).

Friesen, J. (2017). Data Structures and Algorithms in Java, Part 1: Overview.

Hay's, R. N. (2017). Implementasi Firefly Algorithm - Tabu Search Untuk Penyelesaian Traveling Salesman Problem. *Jurnal Online Informatika*, *2*(1), 42–48. Retrieved from http://join.if.uinsgd.ac.id/index.php/join/article/view/v2i18/51

Kusrini, & Istiyanto, J. E. (2007). Penyelesaian Travelling Salesman Problem Dengan Algoritma Cheapest Insertion Heuristics Dan Basis Data. *Jurnal Informatika*, *8*(2), 114–114. https://doi.org/10.9744/informatika.8.2.pp.109-114

Lukas, S., Anwar, T., & Yuliani, W. (2005). Penerapan Algoritma Genetika Untuk Traveling

Salesman Problem Dengan Menggunakan Metode Order Crossover Dan Insertion Mutation. *Seminar Nasional Aplikasi Dan Teknologi Informasi (SNATI 2005)*, *2005*(Snati), 1–5.

Lukman, A., AR, R., & Nurhayati. (2011). *Penyelesaian Travelling Salesman Problem dengan Algoritma Greedy*. *04*(December 2011), 1–5.

Maria, A., Sinaga, E. Y., & Iwo, M. H. (n.d.). *Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO) Departemen Teknik Informatika , Institut Teknologi Bandung*.

Melorose, J., Perroy, R., & Careas, S. (2015). Bab 7 Algoritma Genetika. *Statewide Agricultural Land Use Baseline 2015, 1.* Retrieved from https://doi.org/10.1017/CBO9781107415324.004

Nana, K. K., Prihandono, B., & Noviani, E. (2015). *Penyelesaian Travelling Salesman Problem Menggunakan Metode Simple Hill Climbing*. *04*(3), 173–180.

Nilsson, C. (2003). heuristics for the traveling salesman problem. *Mathematical Programming*, *19*(1), 111–114. https://doi.org/10.1007/BF01581633

Prasetyo, Y. D. (2017). *Penyelesaian Travelling Salesman Problem dengan Algoritma Branch and Bound*. *04*(Maret 2017), 1–5.

Priandani, N. D., & Mahmudy, W. F. (2015). Optimasi Travelling Salesman Problem With Time Windows (Tsp-Tw) Pada Penjadwalan Paket Rute Wisata. *Seminar Nasional Sistem Informasi Indonesia*, (November), 2–3.

Puspitorini, S. (2008). Penyelesaian Masalah Traveling Salesman Problem dengan Jaringan Saraf Self Organizing. *Media Informatika*, *6*(1), 39–55. https://doi.org/10.20885/informatika.vol6.iss1.art3

Rafflesia, U. (2016). Travelling Salesperson Problem dengan Pendekatan Heuristik. *Jurnal Gradien*, *12*(2), 1171–1174.

Rani, S., Kurnia, Y. A., Huda, S. N., & Ekamas, S. A. S. (2019). Smart travel itinerary planning application using held-karp algorithm and balanced clustering approach. *ACM International Conference Proceeding Series*, 106–110. https://doi.org/10.1145/3377817.3377847

Salman, F. (2020). Panduan Ilustrasi tentang Algoritma Genetika.

Sari, E. D. (2013). Optimasi Penjadwalan Personalia Rumah sakit Berbasis AGENT Menggunakan Kombinasi Algoritma Genetika dan Tabu Search. *Optimasi Penjadwalan Personalia Rumah Sakit Berbasis AGENT Menggunakan Kombinasi Algoritma Genetika*

Dan Tabu Search, *1*.

Siagian, N. R. N. (2010). *Studi Metode Program Dinamik Dalam Mencari Solusi Optimal Pada Persoalan Travelling Salesman Problem (TSP)*. 1–12.

Sumihar, Y. P., & Musdholifah, A. (2018). Kombinasi Algoritma Genetika dan Tabu List pada Kasus Penjadwalan Ujian (Studi Kasus: Universitas Kristen Immanuel). *Berkala MIPA*, *25*(3).

Thamilselvan, R., & Balasubramanie, P. (2009). A Genetic Algorithm with a Tabu Search (GTA) for Traveling Salesman Problem. *International Journal of Recent Trends in Engineering*, *1*(1), 2–5. Retrieved from https://www.researchgate.net/publication/229046115

Wiyanti, D. T. (2013). Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem. *Jurnal Transformatika*, *11*(1), 1. https://doi.org/10.26623/transformatika.v11i1.76

LAMPIRAN