Implementasi Kombinasi Algoritma Genetika dan Tabu Search untuk Penyelesaian *Travelling Salesman Problem*

Syarifah Elza Ramadhania
Program Studi Sarjana Informatika
Universitas Islam Indonesia
Jl. Kaliurang KM. 14.5, Sleman, Yogyakarta, Indonesia
17523027@students.uii.ac.id

Septia Rani
Program Studi Sarjana Informatika
Universitas Islam Indonesia
Jl. Kaliurang KM. 14.5, Sleman, Yogyakarta, Indonesia
septia.rani@uii.ac.id

***Abstrak*—Saat melakukan kunjungan ke beberapa tempat, seringkali seseorang kebingungan dalam menentukan urutan titik atau tempat yang akan dikunjungi agar seluruh tempat tersebut dapat dilalui dan jarak yang ditempuh sekecil mungkin. Permasalahan ini dikenal dengan *Travelling Salesman Problem* (TSP). TSP adalah sebuah persoalan klasik seorang *salesman* saat mencari rute atau jalur terpendek. Banyak algoritma yang dapat digunakan dalam penyelesaian TSP, namun setiap algoritma juga memiliki kelebihan dan kekurangan masing-masing. Tujuan dari penelitian ini ialah dapat meng-implementasi kombinasi Algoritma Genetika dan Tabu Search (GA-TS) dalam menyelesaikan kasus TSP simetris. Diharapkan dengan menggunakan kombinasi GA-TS, hasil akhir dari penyelesaian TSP yang didapatkan lebih optimal dibandingkan dengan Algoritma Genetika. Untuk mencapai tujuan penelitian hal yang pertama penulis lakukan ialah merancang sistem mulai dari analisis kebutuhan, perancangan sistem, implementasi, serta perancangan pengujian. Kombinasi GA-TS pada penelitian ini ialah memodifikasi Algoritma Genetika yaitu dengan meletakkan proses tabulist (yang ada pada Tabu Search) sebelum tahapan *crossover*. Hasil pada penelitian ini mampu menyelesaikan kasus TSP simetris dengan menggunakan 4 data yang memiliki ukuran yang berbeda. Dan sistem mampu membuktikan bahwa GA-TS lebih optimal dibandingkan Algoritma Genetika. Hasil yang ditampilkan pada sistem ini ialah berupa jarak terbaik, rute terbaik, nilai fitness terbaik serta menampilkan plot jarak berdasarkan generasi.**

Kata Kunci—Travelling Salesman Problem, Algoritma Genetika, Tabu Search, Tabulist

I. Latar Belakang

Permasalahan optimasi klasik dalam mencari rute kunjungan terpendek dikenal dengan istilah *Travelling Salesman Problem* (TSP) [1][2]. Seorang *salesman* terlibat dalam TSP, yang mana dia wajib mengunjungi beberapa kota untuk menawarkan hasil produksinya. Dalam melakukan kunjungan, *salesman* harus merencanakan suatu rute agar kota-kota tersebut hanya dilalui satu kali saja, kemudian *salesman* kembali ke kota asal. Untuk mendapatkan rute terpendek, beberapa algoritma yang memecahkan persoalan TSP mewajibkan memperhitungkan semua probabilitas rute yang dapat diperoleh [3].

TSP merupakan salah satu persoalan penting di dunia matematika dan informatika. Persoalan tersebut telah banyak diterapkan di berbagai kasus di dunia nyata [4][5]. Adapun contoh kasus yang mampu dipecahkan dengan TSP pada dunia nyata antara lain [2]: mencari rute bis agar dapat mengantarkan para siswa tepat waktu, mencari rute truk untuk mengantar parcel, serta mencari rute pengambilan tagihan telepon secara efisien.

Menurut [6], terdapat dua jenis kasus TSP. Yang pertama yaitu TSP asimetris. Pada kasus TSP asimetris, dana atau biaya dari kota 1 ke kota 2 berbeda dengan dana atau biaya dari kota 2 ke kota 1. Jenis yang kedua yaitu TSP simetris. Pada kasus TSP simetris, dana atau biaya dari kota 1 ke kota 2 setara dengan dana atau biaya dari kota 2 ke kota 1.

Banyak cara penyelesaian yang dapat diterapkan untuk mendapatkan rute terpendek dalam TSP, di antaranya yaitu menggunakan Algoritma *Artificial Bee Colony, Tabu Search,* Algoritma *Cheapest Insertion Heuristics,* Algoritma *Greedy,* Algoritma Genetika, dan yang lainnya. Setiap algoritma mempunyai keunggulan dan kekurangannya sendiri. Hasil dari sebuah algoritma pun berbeda-beda, dikarenakan belum pasti sebuah algoritma yang nilai optimasinya besar untuk sebuah masalah memiliki nilai optimasi yang besar pula untuk masalah lainnya [7].

Penelitian yang akan dilakukan pada makalah ini ialah implementasi kombinasi Algoritma Genetika dan Tabu Search (GA-TS) untuk penyelesaian *Travelling Salesman Problem* dengan kasus simetris. Sebelum ini sudah pernah dilakukan penelitian dengan kasus penjadwalan yang menggunakan GA-TS [8]. Dan hasil akhir membuktikan bahwa GA-TS lebih efisien dibandingkan Algoritma Genetika. Sehingga diharapkan pada penelitian ini, penggunaan GA-TS lebih optimal dibandingkan Algoritma Genetika.

II. Penelitian Sejenis

Dalam TSP banyak sekali algoritma yang dapat diterapkan. Beberapa di antaranya yaitu Algoritma *Artificial Bee Colony, Tabu Search,* Algoritma *Cheapest Insertion Heuristics,* Algoritma *Greedy,* Algoritma Genetika, dan masih banyak lagi. Terdapat banyak penelitian mengenai penerapan ataupun penyelesaian menggunakan algoritma-algoritma tersebut pada kasus TSP. Berikut ini akan disajikan pemaparannya.

Algoritma Greedy termasuk ke dalam jenis algoritma heuristik, dimana algoritma ini mencari *local optimal* dan mengoptimalkannya sehingga mendapatkan *global optima*. Berdasarkan penelitian oleh [9], 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 [10], nilai yang diperoleh tidak terus-menerus optimal, dikarenakan masih terpikat dalam *local optimal*. Sehingga 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 [1][11], *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. Namun 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 [2], *runtime* CIH jauh lebih cepat dibandingkan NNH. Kemudian pada penelitian [9], 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 [9][12].

Tabu *Search* termasuk ke dalam metode optimasi dengan teknik pencarian lokal. Dimana tujuan dari metode ini ialah agar tidak terjadi perulangan serta didapatkannya hasil yang serupa pada suatu iterasi dimana iterasi tersebut akan digunakan pada iterasi berikutnya [13]. Metode optimasi ini pun teruji efektif dan kerap digunakan untuk menyelesaikan permasalahan optimasi berskala besar. Berdasarkan penelitian oleh [14], 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. Dan hasilnya pun terbukti bahwa dengan kombinasi tersebut dapat meningkatkan hasil akurasi permasalahan TSP.

Held-karp oleh [15] membuktikan bahwa hasil *run time Held-karp* jauh lebih cepat dibandingkan dengan *brute force.* Walaupun terjadi peningkatan *runtime* setiap penambahan destinasi namun berjalan cukup stabil. Dan dapat dikatakan bahwa hasil penyelesaian menggunakan *Held-karp* lebih baik dari *brute force.* Kemudian [16] 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 [17], mengatakan bahwa ACO tidak dapat memperhitungkan jumlah iterasi yang baik, dikarenakan solusi yang dihasilkan tidak akan akurat bila jumlah iterasi terlalu kecil. Sedangkan solusi akan semakin besar jika jumlah iterasi terlalu besar. Dan 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 [18].

Berdasarkan penelitian-penelitian sebelumnya pada kasus TSP yang diselesaikan menggunakan berbagai macam algoritma, hasil yang didapatkan setiap algoritma berbeda-beda yaitu mempunyai kelebihan dan kekurangannya masing-masing. Selain kasus TSP, algoritma-algoritma tersebut dapat menyelesaikan kasus lain contohnya ialah kasus penjadwalan. Pada penelitian [8] 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 [8]. Sehingga berdasarkan penelitian tersebut untuk penelitian kali ini menggunakan kombinasi GA-TS dalam penyelesaian TSP dan diharapkan dapat lebih optimal dibandingkan Algoritma Genetika.

III. LANDASAN TEORI

A. *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 [14]. 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 [4]. Pada tahun 1985 Hoffman dan Wolfe menjelaskan bahwa TSP ialah masalah yang harus diselesaikan oleh seorang *salesman* dalam menjalankan perjalanan.

TSP merupakan sebuah persoalan dimana 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 [13]. Dan 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 [10].

B. Algoritma Genetika

Algoritma genetika (GA) termasuk ke dalam algoritma evolusi, dimana teknik optimasinya 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 [19]. Algoritma Genetika memiliki beberapa langkah, yaitu:

1) Inisialisasi Kromosom

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 [12].

2) Evaluasi Kromosom

Melakukan perhitungan nilai fitness terhadap kromosom. Setelah nilai fitness didapatkan, nilai tersebut akan digunakan untuk proses seleksi kromosom.

3) Seleksi Kromosom

Proses ini berguna agar dapat memilih kualitas kromosom yang baik sehingga dapat meneruskan proses *crossover.* Ada berbagai macam teknik seleksi yang dapat digunakan, di antaranya adalah *Roulette Wheel Selection, Elitism, Rank Base Selection,* dan *Steady State Selection*.

4) Crossover

Crossover atau kawin silang menghasilkan dua kromosom anak dengan cara menukar informasi dari dua induk kromosom [20]. 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 [21].

5) Mutasi

Melakukan modifikasi satu atau lebih gen dalam individu merupakan tahapan dari mutasi yang akan menghasilkan individu baru. 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 [22].

6) 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.

C. 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.

Adapun tahapan-tahapan dalam Tabu *Search* sebagai berikut [23]:

- Menentukan titik awal.
- Menentukan solusi alternatif dengan menggabungkan dua titik berdasarkan jarak terkecil.
- Mengevaluasi solusi-solusi alternatif dengan *tabu list* untuk melihat apakah solusi tersebut sudah ada pada *tabu list*. Untuk menghindari perulangan, solusi tidak akan dievaluasi jika sudah ada pada *tabu list* dan begitu sebaliknya.
- Mendapatkan solusi terbaik dan memutuskan sebagai solusi optimum baru.
- Tabu list diperbarui.
- Ulangi langkah-langkah tersebut hingga kriteria sudah terpenuhi.

IV. Metodologi Penelitian

A. Pengambilan Data

Data yang digunakan pada sistem ini adalah dataset TSP dengan kasus simetris yang memiliki koordinat titik x,y. 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/ atau https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html. Gambar 1 merupakan contoh data yang akan digunakan. Pada kolom pertama merupakan kolom kota/titik, kolom kedua ialah koordinat titik x dan kolom terakhir ialah koordinat titik y. Sebagai contoh, pada kota 1, koordinat titik x ialah 6734 sedangkan koordinat titik y ialah 1453.

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

Gambar 1. Contoh Data TSP

B. Analisis Kebutuhan

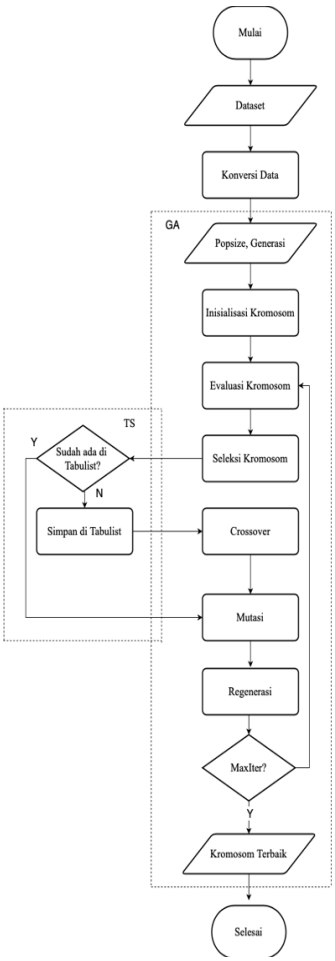
Kemudian dalam sistem yang akan dibuat diperlukan sebuah masukan berupa data yang digunakan, banyaknya iterasi yang akan dijalankan sistem, dan jumlah populasi yang akan dibuat. Adapun kebutuhan proses yang akan dilakukan pada sistem ini ialah:

- Proses konversi data koordinat titik x,y menjadi jarak antar titik atau kota.
- Proses pembuatan inisialisasi kromosom atau rute secara random.
- Proses menghitung nilai fitness (evaluasi) kromosom atau rute.
- Proses seleksi kromosom atau rute.
- Proses pengecekan tabulist.
- Proses *crossover.*
- Proses mutasi.
- Proses menghasilkan kromosom atau rute terbaik.
- Proses menghitung fitness dan jarak akhir.

Dari seluruh proses kombinasi GA-TS tersebut keluaran yang diharapkan dalam sistem ini nantinya ialah dapat menampilkan hasil dari persoalan TSP yaitu berupa rute terbaik.

C. Rancangan Sistem

Pada makalah ini, rancangan utama dari kombinasi Algoritma Genetika dan Tabu Search untuk penyelesaian TSP akan disusun dalam bentuk diagram *flowchart*. Selanjutnya akan dijelaskan lebih detail mengenai setiap proses yang terdapat pada *flowchart*. Gambar 2 merupakan *flowchart* rancangan sistem. Terdapat dua blok utama pada rancangan sistem model 1 dan 2, yaitu blok GA dan TS. Dari *flowchart* dapat dilihat bahwa blok TS merupakan bagian dari blok GA. Hal ini menunjukkan bahwa proses kombinasi antara GA dan TS dilakukan dengan menjadikan TS sebagai salah satu tahapan di dalam GA, yang mana TS berperan dalam proses seleksi kromosom.



Gambar 2. *Flowchart* Rancangan Sistem

Langkah pertama yang dilakukan dalam sistem ini ialah memasukkan data kemudian mengkonversi data yang awalnya berupa koordinat x,y menjadi data jarak antar kota atau titik. Cara mengkonversi data yaitu dengan menggunakan rumus pada Persamaan 1 dengan x_1, y_1 ialah koordinat kota/titik awal dan x_2, y_2 koordinat kota/titik akhir [12]. Tabel 1 menunjukkan contoh gambaran sebuah hasil dari konversi data koordinat titik x,y menjadi jarak antar kota atau titik.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

TABEL I. CONTOH HASIL KONVERSI DATA

	1	2	3	4	5	6
1	0	1450	47	643	223	62
2	1450	0	1417	833	1634	4451
3	47	1417	0	605	233	3034
4	643	833	605	0	807	3624
5	223	1634	233	807	0	2817
6	62	4451	3034	3624	2817	0

Kemudian setelah jarak didapatkan, maka *user* memasukkan ukuran populasi (popsize) dan maksimum iterasi (generasi). Setelah itu, langkah-langkah yang dilakukan sesuai dengan urutan berikut:

1) 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. Dimana setiap angka tersebut mewakili titik-titik (tempat) tujuan yang akan dilalui.

2	3	1	5	4	6

Gambar 3. Contoh Representasi Kromosom

Gambar 3 merupakan contoh representasi kromosom. Dapat dilihat bahwa rute yang akan dilalui ialah dari titik 2-3-1-5-4-6. Tabel 2 merupakan contoh inisialisasi kromosom dengan ukuran populasi yaitu sebesar 5.

TABEL II. INISIALISASI KROMOSOM

Kromosom	Rute
1	(5-1-2-4-3-6)
2	(1-4-5-3-6-2)
3	(3-1-2-5-6-4)
4	(3-5-2-1-4-6)
5	(2-1-3-5-4-6)

2) Evaluasi Kromosom

Setelah kromosom terbentuk maka setiap kromosom dilakukan perhitungan nilai fitness menggunakan Persamaan 2 [13]. *TotalJarak* merupakan total jarak yang ditempuh dari titik awal hingga akhir setiap kromosom. Pada Tabel 3 merupakan contoh perhitungan *fitness* setiap kromosom.

$$fitness = \frac{1}{TotalJarak} \quad (2)$$

TABEL III. EVALUASI KROMOSOM

Kromosom	Jarak	Fitness
(5-1-2-4-3-6)	6145	0,00016273
(1-4-5-3-6-2)	9168	0,00010908
(3-1-2-5-6-4)	9572	0,00010447
(3-5-2-1-4-6)	7584	0,00013186
(2-1-3-5-4-6)	6161	0,00016231

3) 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. Kromosom tersebut akan digunakan untuk proses selanjutnya. Tabel 4 merupakan contoh hasil seleksi kromosom.

TABEL IV. CONTOH HASIL SELEKSI KROMOSOM

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

4) 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*. Tabel 5 merupakan contoh ilustrasi penyimpanan kromosom pada *tabulist*.

TABEL V. ILUSTRASI TABULIST

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

5) Crossover

Pada *crossover,* dua kromosom induk akan mengalami proses kawin silang dengan probabilitas *crossover*. Pada tahap ini kromosom tidak dapat memiliki gen atau kota yang sama. Oleh karena itu, dapat digunakan metode *ordered crossover* dengan memilih secara acak beberapa gen di induk 1, kemudian sisa gen pada induk 1 akan diisi oleh induk 2 dimana gen tidak boleh sama. Proses *crossover* akan menghasilkan *offspring/children* baru. Gambar 4 merupakan contoh *ordered crossover.*

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

Children	3	1	2	4	5	6

Gambar 3. Contoh *Ordered Crossover*

6) Mutasi

Setelah terjadinya *crossover* ataupun pengecekan tabu *list,* kromosom akan dimutasi sesuai dengan probabilitas mutasi menggunakan metode *reciprocal exchange mutation* yaitu dengan memilih dua posisi secara acak kemudian menukarkan kedua nilai tersebut. Gambar 5 merupakan contoh *reciprocal exchange mutation.*

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

Gambar 4. Contoh *Reciprocal Exchange Mutation*

7) 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. Pada Tabel 6 merupakan contoh populasi baru atau generasi baru.

TABEL VI. POPULASI BARU

Kromosom	Jarak	Fitness
(5-1-2-4-3-6)	6145	0,00016273
(1-4-5-3-6-2)	9168	0,00010908
(3-5-2-4-1-6)	3405	0,00029369
(3-5-2-1-4-6)	7584	0,00013186
(2-1-3-5-4-6)	6161	0,00016231

8) Kromosom Terbaik

Proses akan berhenti, jika iterasi yang ditetapkan telah selesai.

D. *Implementasi*

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.

E. *Rancangan Pengujian*

Skenario pengujian yang akan diterapkan nantinya pada penelitian ini ialah pengujian performa sistem. Pengujian performa yang dilakukan adalah analisis hasil waktu pemrosesan berdasarkan jumlah iterasi, analisis hasil akhir berupa jarak berdasarkan jumlah iterasi, dan analisis perbandingan antara GA-TS dan Algoritma Genetika (tanpa TS).

V. HASIL DAN PEMBAHASAN

Data yang diambil merupakan dataset untuk TSP kasus simetris yang dapat diakses melalui tautan yang telah disampaikan pada metodologi, dan Tabel 7 merupakan data yang digunakan :

TABEL VII. DATASET

Dataset	Size	Nilai Optimal
P01	15 titik	291
Eli51	51 titik	426
Ch130	130 titik	6110
Lin318	318 titik	42029

A. *Implementasi Sistem*

1) Masukkan Data dan Konversi Data

Data yang digunakan memiliki format file CSV dan di *import* pada sistem, kemudian data diinisialisasi dengan nama city (Gambar 6). City atau data yang awalnya berisi koordinat titik x,y diubah menjadi jarak antar titik melalui sistem. Dan data city dibuat atau diubah menjadi format list yang dinamakan cityList (Gambar 7).

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

Gambar 5. Contoh Kode Inisialisasi Dataset

```
cityList = []
for i in range(0,len(city)):
    cityList.append(City(name = city.iloc[i,0],x=city.iloc[i][1],y=city.iloc[i][2]))

print(cityList)
```

Gambar 6. Contoh Kode Membuat Data City menjadi List

2) Masukkan Nilai Populasi dan Jumlah Generasi

Setelah hasil konversi didapatkan selanjutnya ialah memasukan jumlah populasi (popsize) dan jumlah generasi (generations). Pada Gambar 6 merupakan gambaran masukan popsize dan generations. Dapat diartikan bahwa pada populasi cityList dibuat sebuah populasi berjumlah 10 dan akan dilakukan perulangan sebanyak 500 kali.

```
geneticAlgorithm(population=cityList, popSize=10 , generations=500)
```

Gambar 7. Contoh Masukan popsize dan generations

3) Inisialisasi dan Evaluasi Kromosom

Berdasarkan jumlah populasi yang dimasukkan, maka sistem akan membuat populasi secara *random* sebanyak popsize dari data cityList dan mengevaluasi setiap kromosom agar menghasilkan sebuah nilai fitness. Dikarenakan inisialisasi kromosom atau rute dilakukan secara acak maka setiap kali menjalankan program, hasil akhir yang didapatkan akan berbeda. Pada Gambar 7 merupakan contoh hasil inisialisasi dan evaluasi rute dimana nilai popsize ialah 5, serta total jarak setiap kromosom ialah berdasarkan hasil konversi data, kemudian dilakukan perhitungan fitness menggunakan Persamaan 2.

```
populasi ke-0 ([5, 9, 15, 2, 12, 6, 10, 13, 14, 1, 11, 4, 8, 3, 7])
 Jarak = 513.3094587293826
 Fitness = 0.0019481425541530907
populasi ke-1 ([10, 14, 7, 6, 13, 5, 15, 12, 3, 4, 1, 11, 2, 8, 9])
 Jarak = 592.4500309267612
 Fitness = 0.0016879060643067469
populasi ke-2 ([1, 10, 2, 13, 8, 7, 4, 3, 12, 14, 9, 5, 6, 15, 11])
 Jarak = 636.5470624987515
 Fitness = 0.0015709757516977959
populasi ke-3 ([1, 6, 7, 11, 9, 15, 12, 14, 3, 10, 8, 13, 5, 4, 2])
 Jarak = 637.4959870574916
 Fitness = 0.0015686373252571026
populasi ke-4 ([12, 5, 13, 6, 2, 1, 11, 10, 7, 8, 3, 4, 14, 15, 9])
 Jarak = 649.3635196328919
 Fitness = 0.0015399694774435979
```

Gambar 8. Contoh Tampilan Hasil Inisialisasi dan Evaluasi Kromosom

4) Seleksi Kromosom

Seleksi pada sistem ini ialah mengurutkan kromosm 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 8 ialah contoh hasil seleksi kromosom.

```
seleksi ke-0 ([5, 9, 15, 2, 12, 6, 10, 13, 14, 1, 11, 4, 8, 3, 7])
 Jarak = 513.3094587293826
 Fitness = 0.0019481425541530907
seleksi ke-1 ([10, 14, 7, 6, 13, 5, 15, 12, 3, 4, 1, 11, 2, 8, 9])
 Jarak = 592.4500309267612
 Fitness = 0.0016879060643067469
```

Gambar 9. Contoh Tampilan Hasil Seleksi Kromosom

5) Tabulist, Crossover, dan Mutasi

Hasil seleksi akan dicek terlebih dahulu pada tabulist, jika tidak ada dalam tabulist (tabulis false) maka hasil seleksi akan disimpan dalam tabulist dan melanjutkan proses *crossover* yaitu menggunakan metode *ordered crossover.* Jika ada pada tabulist (tabulist true) maka hasil seleksi terbaik akan dimutasi. Proses mutasi dilakukan dengan memilih 2 titik secara acak dan menukar kedua nilai tersebut atau disebut dengan metode *reciprocal exchange mutation*. Pada Gambar 9 merupakan contoh jika tabulist false dan Gambar 10 ialah contoh isi tabulist, dan Gambar 11 contoh hasil mutasi dimana swapped merupakan posisi gen yang akan ditukar dengan posisi gen swap with.

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

Gambar 10. Contoh Tampilan jika Tabulist False

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

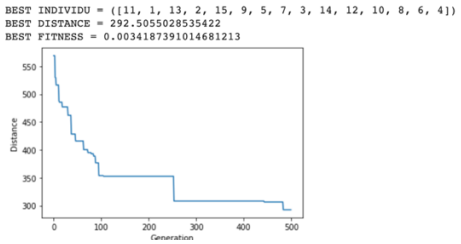
Gambar 11. Contoh Tampilan Isi Tabulist

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

Gambar 12. Contoh Tampilan Hasil Mutasi

6) Regenerasi dan Hasil Akhir

Regenerasi ialah dengan menggantikan nilai terendah dalam populasi menjadi hasil mutasi. Dan sistem akan berhenti jika generasi telah mencapai jumlah generasi yang telah ditentukan. Hasil akhir ialah berupa individu terbaik berserta dengan jarak dan nilai fitness individu tersebut dan sistem menampilkan diagram garis dari nilai akhir setiap generasi secara keseluruhan yang dapat dilihat pada Gambar 12.



Gambar 13. Contoh Hasil Akhir Sistem

B. Hasil Implementasi

Setelah setiap langkah diimplementasi dan mendapatkan sebuah hasil, hasil tersebut akan dievaluasi. Pada tahapan evaluasi, nilai populasi yang berikan ialah sebesar 10 populasi. Dan pada Tabel 8-11 merupakan hasil pengujian setiap dataset dan membandingkan Algoritma Genetika (GA) dan GA-TS berdasarkan hasil akhir (jarak) dan *runtime* dalam satuan detik (s) sebagai berikut :

1) P01

TABEL VIII. HASIL UJI TERHADAP *DATASET* P01

Generasi	Hasil Akhir (Jarak)		*Runtime*	
	GA	GA-TS	GA	GA-TS
50	473,808	435,166	1,14 s	1,01 s
100	406,299	356,63	2,08 s	1,72 s
500	329,709	329,96	9,63 s	8,54 s
1000	301,358	342,96	19,2 s	16,8 s

2) ELI51

TABEL IX. HASIL UJI TERHADAP *DATASET* ELI51

Generasi	Hasil Akhir (Jarak)		*Runtime*	
	GA	GA-TS	GA	GA-TS
50	1305,107	1288,42	2,59 s	2,24 s
100	1186,096	1065,479	4,99 s	4,42 s
500	908,111	808,755	24,5 s	22,9 s
1000	823,98	770,22	48,7 s	50,3 s

3) CH130

TABEL X. HASIL UJI TERHADAP *DATASET* CH130

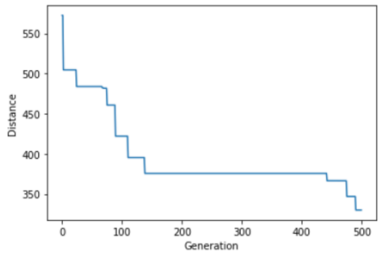
Generasi	Hasil Akhir (Jarak)		*Runtime*	
	GA	GA-TS	GA	GA-TS
50	40002,49	37492,146	5,62 s	4,87 s
100	36329,925	36534,473	11,4 s	9,61 s
500	30705	26611,872	55 s	56,4 s
1000	27328	22899,155	1min 49s	2 min

4) Lin318

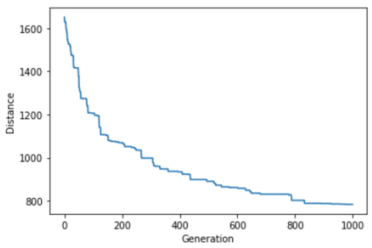
TABEL XI. HASIL UJI TERHADAP *DATASET* LIN318

Generasi	Hasil Akhir (Jarak)		*Runtime*	
	GA	GA-TS	GA	GA-TS
50	525748,577	547158	10,3 s	11,4 s
100	524665,672	508255,91	20,2 s	24,4 s
500	417851,014	404042,575	1min 42 s	1min 38 s
1000	407928	371753,773	3min 20 s	3min 18 s

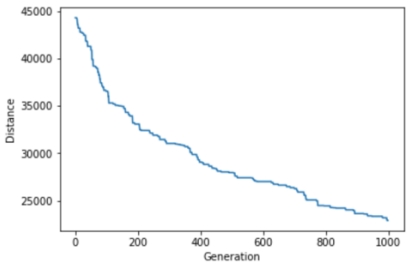
Dan pada Gambar 13-16 merupakan grafik hasil terbaik pada setiap dataset.Jika garis pada grafik menurun artinya hasil akan semakin kecil atau semakin baik. Dan pada algoritma ini garis tidak pernah naik dikarenakan setiap rute atau kromosom telah melewati proses seleksi.



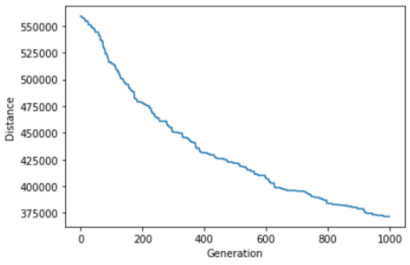
Gambar 14. Grafik Hasil Terbaik pada Dataset P01



Gambar 15. Grafik Hasil Terbaik pada Dataset Eli51



Gambar 16. Grafik Hasil Terbaik pada Dataset Ch130



Gambar 17. Grafik Hasil Terbaik pada Dataset Lin318

Berdasarkan tabel hasil evaluasi data kasus TSP simestris dengan *size* yang berbeda-beda dapat diselesaikan menggunakan GA-TS. Dan hasil dari perbandingan menunjukkan bahwa hasil terbaik dari ke-empat dataset selalu berada pada GA-TS. Hasil akhir dan *runtime* setiap data ataupun generasi selalu berbeda. Jika data memiliki size besar maka waktu yang dibutuhkan cenderung besar pula. Dan pada GA-TS waktu proses atau *runtime* cenderung lebih kecil daripada menggunakan Algorima Genetika, begitupula hasil akhir yang didapatkan yaitu cenderung lebih mendekati optimal. Namun masih memiliki kekurangan jika data yang memiliki *size* besar maka hasil masih cenderung jauh dari nilai optimal.

C. *Pembahasan*

Pada tahap implementasi bahasa pemograman yang digunakan ialah Python dan menggunakan Google Colaboratory (Colab) sebagai perangkat lunak yang diakses melalui *browser* Google Chrome secara *online*.

Penelitian ini memiliki kontribusi pada kasus TSP yaitu melakukan modifikasi pada Algoritma Genetika dengan melakukan kombinasi terhadap Tabu Search. Proses modifikasi dilakukan dengan kombinasi salah satu langkah yang ada pada Algoritma Genetika dengan Tabu Search. Langkah tersebut adalah proses seleksi kromosom. Proses seleksi kromosom dengan memanfaatkan Tabu Search memiliki peranan penting dalam penyelesaian kasus TSP yaitu agar terhindar dari perulangan agar hasil tidak sama, sehingga solusi optimal dapat lebih cepat diperoleh.

VI. KESIMPULAN

Penelitian ini memiliki tujuan agar kombinasi Algoritma Genetika dan Tabu Search (GA-TS) dapat menghasilkan hasil yang lebih optimal dibandingkan dengan Algoritma Genetika. Pada tahap implementasi sistem dapat disimpulkan bahwa penelitian ini mampu mengeluarkan hasil akhir berupa individu terbaik, jarak terbaik, nilai fitness terbaik, serta diagram garis. Dari hasil yang didapatkan membuktikan bahwa GA-TS lebih optimal dan memiliki runtime yang cenderung lebih kecil dibandingkan dengan Algoritma Genetika. Hasil tersebut dapat dilihat pada tahapan evaluasi, yaitu hasil penelitian menggunakan GA-TS dibandingkan dengan Algoritma Genetika.

Namun, masih terdapat beberapa kekurangan yaitu tidak semua data dapat mendekati hasil yang optimal terutama untuk kasus yang memiliki size besar.Penelitian ini masih akan dilanjutkan yaitu dengan membuat model kedua dari kombinasi GA-TS. Dan akan melakukan pengujian menggunakan data dengan *size* lebih besar lagi. Diharapkan untuk penelitian selanjutnya juga dapat meningkatkan performa sistem ini sehingga hasil yang didapatkan lebih optimal.

REFERENSI

[1] Andri, Suyandi, and WinWin, "Aplikasi Travelling Salesman Problem dengan Metode Artificial Bee Colony," vol. 14, no. 1, pp. 59–68, 2013.

[2] Kusrini and J. E. Istiyanto, "Penyelesaian Travelling Salesman Problem Dengan Algoritma Cheapest Insertion Heuristics Dan Basis Data," *J. Inform.*, vol. 8, no. 2, pp. 114–114, 2007, doi: 10.9744/informatika.8.2.pp.109-114.

[3] S. Puspitorini, "Penyelesaian Masalah Traveling Salesman Problem dengan Jaringan Saraf Self Organizing," *Media Inform.*, vol. 6, no. 1, pp. 39–55, 2008, doi: 10.20885/informatika.vol6.iss1.art3.

[4] U. Rafflesia, "Travelling Salesperson Problem dengan Pendekatan Heuristik," *J. Gradien*, vol. 12, no. 2, pp. 1171–1174, 2016.

[5] S. N, "Studi Metode Program Dinamik Dalam Mencari Solusi Optimal Pada Persoalan Travelling Salesman Problem (TSP)," pp. 1–12, 1993.

[6] D. T. Wiyanti, "Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem," *J. Transform.*, vol. 11, no. 1, p. 1, 2013, doi: 10.26623/transformatika.v11i1.76.

[7] G. Aristi, "Perbandingan Algoritma Greedy, Algoritma Cheapest Insertion Heuristics Dan Dynamic Programming Dalam Penyelesaian Travelling Salesman Problem," *Paradigma*, vol. XVI, no. 2, pp. 52–58, 2014.

[8] E. D. Sari, "Optimasi Penjadwalan Personalia Rumah sakit Berbasis AGENT Menggunakan Kombinasi Algoritma Genetika dan Tabu Search," *Optimasi Penjadwalan Pers. Rumah sakit Berbas. AGENT Menggunakan Komb. Algoritm. Genet. dan Tabu Search*, vol. 1, 2013.

[9] H. A. Abdulkarim and I. F. Alshammari, "Comparison of Algorithms for Solving Traveling Salesman Problem," *Int. J. Eng. Adv. Technol.*, vol. 4, no. 6, pp. 76–79, 2015.

[10] A. Lukman, R. AR, and Nurhayati, "Penyelesaian Travelling Salesman Problem dengan Algoritma Greedy," vol. 04, no. December 2011, pp. 1–5, 2011.

[11] F. Amri, E. B. Nababan, and M. F. Syahputra, "Artificial Bee Colony Algorithm untuk Menyelesaikan Travelling Salesman Problem," *J. Dunia Teknol. Inf.*, vol. 1, no. 1, pp. 8–13, 2012.

[12] S. Lukas, T. Anwar, and W. Yuliani, "Penerapan Algoritma Genetika Untuk Traveling Salesman Problem Dengan Menggunakan Metode Order Crossover Dan Insertion Mutation," *Semin. Nas. Apl. dan Teknol. Inf. (SNATI 2005)*, vol. 2005, no. Snati, pp. 1–5, 2005.

[13] Fatmawati, B. Prihandono, and E. Noviani, "Penyelesaian Travelling Salesman Problem Dengan Metode Tabu Search," *Bul. Ilm. Mat. Stat. Dan Ter.*, vol. 04 no. 1, no. 1, pp. 17–24, 2015.

[14] R. N. Hay's, "Implementasi Firefly Algorithm - Tabu Search Untuk Penyelesaian Traveling Salesman Problem," *J. Online Inform.*, vol. 2, no. 1, pp. 42–48, 2017, [Online]. Available: http://join.if.uinsgd.ac.id/index.php/join/article/view/v2i18/51.

[15] S. Rani, Y. A. Kurnia, S. N. Huda, and S. A. S. Ekamas, "Smart travel itinerary planning application using held-karp algorithm and balanced clustering approach," *ACM Int. Conf. Proceeding Ser.*, pp. 106–110, 2019, doi: 10.1145/3377817.3377847.

[16] C. Nilsson, "heuristics for the traveling salesman problem," *Math. Program.*, vol. 19, no. 1, pp. 111–114, 2003, doi: 10.1007/BF01581633.

[17] A. Maria, E. Y. Sinaga, and M. H. Iwo, "Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO) Departemen Teknik Informatika , Institut Teknologi Bandung."

[18] Y. D. Prasetyo, "Penyelesaian Travelling Salesman Problem dengan Algoritma Branch and Bound," vol. 04, no. Maret 2017, pp. 1–5, 2017.

[19] W. Firdaus Mahmudy, "Algoritma Evolusi," no. September, 2013.

[20] N. S. Dini, "Produk Air Minum Kemasan Galon Menggunakan Kombinasi Algoritma Genetika Dan Pencarian Tabu Di Depot Air Minum Isi Ulang Banyu Belik," 2015.

[21] J. Melorose, R. Perroy, and S. Careas, "Bab 7 Algoritma Genetika," *Statew. Agric. L. Use Baseline 2015, 1.*, 2015, [Online]. Available: https://doi.org/10.1017/CBO9781107415324.004.

[22] N. D. Priandani and W. F. Mahmudy, "Optimasi Travelling Salesman Problem With Time Windows (Tsp-Tw) Pada Penjadwalan Paket Rute Wisata," *Semin. Nas. Sist. Inf. Indones.*, no. November, pp. 2–3, 2015.

[23] E. Fernando, "Kombinasi Algoritma Genetik Dan Tabu Search," vol. 6, no. 1, 20