APLIKASI TRAVELLING SALESMAN PROBLEM DENGAN METODE ARTIFICIAL BEE COLONY

Andri[1], Suyandi[2], WinWin[3]

STMIK Mikroskil
Jl. Thamrin No. 122, 124, 140 Medan 20212

andri@mikroskil.ac.id [1], suyandiz@gmail.com[2],
winzz990@yahoo.com[3]

Abstrak

Pada saat ini *Travelling Salesman Problem* (TSP) dikenal sebagai salah satu permasalahan optimasi klasik yang berat untuk dipecahkan secara konvensional. TSP melibatkan seorang *travelling salesman* yang harus melakukan kunjungan ke sejumlah kota dalam menjajakan produknya. Rangkaian kota-kota yang dikunjungi harus membentuk suatu jalur sedemikian sehingga kota-kota tersebut hanya boleh dilewati tepat satu kali dan kemudian kembali lagi ke kota awal.

Tujuan dari penelitian ini adalah membangun suatu aplikasi TSP dengan algoritma *Artificial Bee Colon*y dengan bahasa pemrograman Visual Basic 2008. Model graf yang digunakan adalah graf tidak berarah dan berbobot (simetris). Masukan program berupa jumlah kota (N), jumlah koloni, parameter α (alpha), β (beta) dan jumlah iterasi.

Keluaran program berupa panjang rute terpendek, proses perhitungan, grafik dan rute yang berbentuk graf. Pengujian yang dilakukan pada aplikasi ini berupa masukan beberapa parameter dengan nilai bervariasi seperti jumlah koloni, jumlah iterasi, K (faktor *scalar*), λ (lamda), α (Alpha) dan β (Beta). Hasil pengujian menunjukkan besarnya parameter yang diinput akan berdampak semakin optimal panjang jalur yang diperoleh. Aplikasi ini diharapkan dapat dimanfaatkan untuk menyelesaikan kasus seorang salesman dalam mencari rute terpendek.

Kata kunci : *Travelling Salesman Problem, Artificial Bee Colony, Rute Terpendek*

1. Pendahuluan

1.1. Latar Belakang

Travelling Salesman Problem (TSP) dikenal sebagai salah satu permasalahan optimasi klasik yang berat untuk dipecahkan secara konvensional sehingga penyelesaian eksak terhadap persoalan ini akan melibatkan algoritma yang mengharuskan untuk mencari kemungkinan semua solusi yang ada. Sebagai akibatnya, kompleksitas waktu dari eksekusi algoritma ini akan menjadi eksponensial terhadap ukuran dari masukan yang diberikan [2].

Permasalahan yang melibatkan algoritma demikian lebih dikenal sebagai permasalahan yang bersifat *Nondeterministic Polynomial-time Complete* (NP-*Complete*). TSP melibatkan seorang *travelling salesman* yang harus melakukan kunjungan ke sejumlah kota dalam menjajakan produknya. Rangkaian kota-kota yang dikunjungi harus membentuk suatu jalur sedemikian sehingga kota-kota tersebut hanya boleh dilewati tepat satu kali dan kemudian kembali lagi ke kota awal. Jika terdapat n kota yang harus dikunjungi, maka diperlukan proses pencarian sebanyak n!/2n rute, dengan cara ini waktu komputasi yang diperlukan akan jauh meningkat seiring dengan bertambahnya jumlah kota yang harus dikunjungi. Sebagai ilustrasi,

untuk 10 kota saja, diperlukan proses pencarian jalur sebanyak 181.440 rute [2]. Ada beberapa metode atau algoritma yang digunakan dalam penyelesaian kasus TSP seperti algoritma *Genetic Algorithm*, *Simulated Annealing*, *Neural Network*, *Greedy Heuristic*, dan lain-lain. Beberapa metode yang ada tersebut untuk menyelesaikan kasus *Travelling Salesman Problem* masih terdapat kekurangan khususnya pada penjelasan kasus di atas yang menunjukkan bahwa solusi eksak terhadap masalah TSP sangat sulit dilakukan. Oleh karena itu dibuat suatu perangkat lunak yang melibatkan *Artificial Bee Colony* (Kecerdasan Koloni Lebah) dengan metode pembelajaran untuk memberikan pemecahan alternatif yang lebih baik.

1.2. Rumusan Masalah

Secara konvensional, penyelesaian dalam kasus TSP sangatlah sulit dilakukan karena melibatkan proses dan waktu komputasi yang kurang efisien sehingga untuk menyelesaikan permasalahan tersebut diperlukan suatu metode. Metode yang digunakan dalam tulisan ini adalah *Artificial Bee Colony*. Metode ini merupakan suatu metode pencarian nilai optimal yang terinspirasi dari kegiatan kawanan lebah dalam mencari makanan. Dengan menggunakan metode ini maka permasalahan TSP tersebut dapat diselesaikan dengan waktu komputasi yang jauh lebih efisien dari pada penyelesaian secara konvensional.

1.3. Tujuan dan Manfaat

Tujuan dari penelitian ini adalah membangun sebuah aplikasi TSP dengan metode *Artificial Bee Colony*.

Manfaatnya adalah:

a. Bagi perkembangan teknologi informasi, penelitian diharapkan dapat digunakan untuk membantu suatu lembaga atau perusahaan yang bergerak dibidang transportasi yang memerlukan efisiensi waktu dan biaya dalam mencari rute terpendek.
b. Sebagai tambahan referensi mengenai aplikasi TSP yang dapat digunakan oleh pihak-pihak yang memerlukan.
c. Memudahkan pengguna dalam pemecahan kasus TSP secara optimal.

1.4. Batasan Masalah

Ruang lingkup pembahasan pada penelitian ini sebagai berikut:

a. Model yang dibuat berupa graf yang tidak berarah dan berbobot (simetris)
b. Input program berupa jumlah kota (N), jumlah koloni, parameter α (alpha), β (beta), K (faktor *scalar*), λ (lamda) dan jumlah iterasi. λ bernilai antara 0 sampai 1 ($\lambda \neq 0$ dan $\lambda \neq 1$)
c. Minimal kota yang bisa diinput adalah sebanyak 5 kota.
d. Output yang dihasilkan berupa panjang rute yang terpendek, proses penghitungan, grafik dan rute yang berbentuk graf.
e. Perangkat lunak juga menyediakan teori pendukung yang berhubungan dengan TSP dan algoritma *bee colony*.

1.5 Metodologi Penelitian

Langkah-langkah yang dilakukan untuk menyelesaikan penelitian ini antara lain:

a. Menganalisa teori dan memahami cara kerja dari algoritma *Artificial Bee Colony* untuk kasus TSP dan mengimplementasikan ke dalam aplikasi.
b. Desain *prototype* terhadap aplikasi yang akan dibuat berupa tampilan *splash screen*, menu utama, menu teori dan menu aplikasi.

c. Pembuatan program aplikasi menggunakan bahasa pemrograman VB.NET 2008.
d. Melakukan pengujian terhadap beberapa parameter dalam mencari rute terpendek pada aplikasi.

2. Kajian Pustaka

2.1. Metode Penyelesaian Permasalahan TSP

Dalam penyelesaian masalah TSP kita dapat membagi dalam 2 metode, yaitu metode optimasi dan metode aproksimasi. Metode optimasi akan menghasilkan hasil yang optimal (minimum) sedangkan metode aproksimasi akan menghasilkan hasil yang mendekati optimal [1].

2.2.1 Metode Optimasi

Sejak permasalahan TSP ditemukan pada tahun 1800 oleh matematikawan Irlandia Sir William Rowan Hamilton dan matematikawan Inggris Thomas Penyngton Kirkman, pusat perhatian studi ini adalah menemukan secara pasti nilai minimum dari persoalan TSP dengan konsekuensi dibutuhkan waktu yang cukup lama untuk menyelesaikannya.

a. Complete Enumeration

Metode ini akan mengenumerasi setiap kemungkinan yang terdapat dalam graf, setelah itu algoritma ini akan membandingkan lintasan mana yang paling minimum.

b. Branch and Bound

Metode *Branch and Bound* adalah sebuah teknik algoritma yang secara khusus mempelajari bagaimana caranya memperkecil *Search Tree* menjadi sekecil mungkin. Sesuai dengan namanya, metode ini terdiri dari 2 langkah yaitu *Branch* yang artinya membangun semua cabang *tree* yang mungkin menuju solusi dan *Bound* yang artinya menghitung node mana yang merupakan *active node* (*E-node*) dan node mana yang merupakan *dead node* (*D-node*) dengan menggunakan syarat batas *constraint* (kendala) [1].

2.2.2 Metode Heuristik

Metode Heuristik yaitu metode yang memberikan pendekatan untuk menyelesaikan permasalahan optimasi kombinatorial. *Combinatorial search* akan memberi kita hasil yang mungkin dan mencari yang hasil yang mendekati optimal dari hasil-hasil tersebut.

Algoritma yang biasa digunakan dalam metode heuristik adalah *genetic algorithm, simulated annealing,* dan *neural network.*

a. *Genetic Algorithm* yaitu algoritma yang inspirasi dari proses evolusi dan seleksi alam. Melalui proses seleksi alam, organisme atau mahluk hidup beradaptasi untuk mengoptimalkan kesempatan mereka untuk tetap bisa hidup didalam sebuah lingkungan. Mutasi secara random menghasilkan kode genetik dari mahluk hidup yang kemudian diturunkan kepada anaknya. Jika mutasi meningkatkan kualitas genetik, maka secara otomatis sang anak akan terbantu untuk selamat.
b. *Simulated Annealing* yaitu metode yang terinspirasi dari proses fisika mengenai pendinginan lelehan material yang berubah menjadi padat. Ketika lelehan besi didinginkan terlalu cepat, maka retakan dan gelembung udara akan muncul, menghancurkan permukaan dan integritas strukturnya. Oleh karena itu untuk menghasilkan besi yang baik, besi harus didinginkan secara pelan-pelan dan diberi jeda. *Annealing* adalah teknik metalurgi yang menggunakan ilmu penjadwalan proses pendinginan untuk menghasilkan efisiensi dalam menggunakan energi dan menghasilkan besi yang optimal.

c. *Neural Network* yaitu paradigma komputasional yang terinspirasi dari arsitektur otak manusia. Ketika otak memiliki intuisi yang sangat baik dalam memecahkan persoalan, maka mesin pun seharusnya dibangun seperti itu [1].

2.2 Algoritma Lebah untuk TSP

Algoritma Optimasi terbagi menjadi dua jenis, yaitu algoritma optimasi dengan pendekatan berbasis *deterministic* dan algoritma optimal dengan pendekatan berbasis *probabilistic*. Yang termasuk dalam algoritma berbasis *deterministic* diantaranya *State Space Search, Dynamic Programming*, dan *Branc and Bound*. Sedangkan algoritma optimasi yang termasuk dalam algoritma yang berbasis pendekatan *probabilistic* adalah algoritma *Monte Carlo* dengan berbagai macam turunannya [5].

Evolutionary Computation merupakan salah satu algoritma yang termasuk kedalam algoritma optimasi berbasis *probabilistic*. Definisi dari algoritma *Evolutionary Computation* adalah abstraksi dari teori evolusi biologis yang digunakan untuk membuat prosedur atau metodologi optimal, biasanya diterapkan pada komputer, yang digunakan untuk memecahkan masalah. Algoritma ini memiliki ide dasar dari bagaimana proses evolusi yang terjadi pada makhluk hidup yang menganggap bahwa hasil setiap evolusi itu akan membawa sesuatu menjadi lebih baik dan optimal [5].

Beberapa algoritma yang termasuk kedalam algoritma *Evolutionary Computation* diantaranya adalah *swarm intelligent* dimana algoritma ini didasarkan dari kecerdasan berkelompok. Dengan semakin banyaknya anggota kelompok dan terkumpulnya kecerdasan-kecerdasan individual maka akan menyebabkan terkumpulnya kecerdasan kelompok yang luar biasa. Beberapa yang termasuk dalam algoritma *swarm intelligent* diantaranya *Particel Swarm Optimization, Ant Colony Optimization, Artificial Bee Colony Optimization* [5].

Swarm intelligent merupakan sebuah metode penyelesaian masalah yang memanfaatkan prilaku sekumpulan agen yang saling bekerja sama. Khususnya *swarm intelligence* pada algoritma lebah (*Algorithm Bee Colony*) terinspirasi dari perilaku sosial koloni lebah dimana seekor lebah bisa menjangkau sumber makanan sekaligus mengingat letak, arah dan jarak dari sumber makanan. Sekembalinya dari pencarian sumber makanan maka seekor lebah akan melakukan *waggle dance*. *Waggle dance* merupakan alat komunikasi yang dilakukan oleh koloni lebah [5].

Langkah – langkah penyelesaian algoritma lebah sebagai berikut [3]:

1. Forage

Pada awalnya harus ditentukan jumlah lebah (N). Jumlah lebah ini sama dengan jumlah kota pada masalah TSP. Setelah itu dilanjutkan tahapan *forage* yaitu lebah melakukan pergerakan dari suatu kota ke kota lain untuk membentuk sebuah jalur TSP. Ketika semua lebah sudah menyelesaikan perjalanan (membentuk jalur), maka dianggap lengkaplah siklus lebah. Karena pada siklus pertama perjalanan, kawanan lebah tidak menari dalam menentukan perjalanan ke kota berikutnya, maka mereka melakukan perjalanan ke kota berikutnya secara acak. Berikutnya lebah harus mengikuti aturan dalam membuat keputusan dalam memilih kota kunjungan berikutnya. $P_{ij,n}$ adalah besarnya kemungkinan lebah bergerak dari kota i ke kota j setelah transisi ke-n yang dirumuskan sebagai berikut:

$$P_{ij,n} = \frac{[\rho_{ij,n}]^{\alpha} \cdot [\frac{1}{d_{ij}}]^{\beta}}{\sum_{j \in A_{i,n}} [\rho_{ij,n}]^{\alpha} \cdot [\frac{1}{d_{ij}}]^{\beta}}$$

Keterangan:

$\rho_{ij,n}$ (*rho*) menunjukkan *arc fitness* dari kota i ke kota j setelah transisi ke-n.
$d_{i,j}$ menunjukkan jarak antara kota i dengan kota j.
$A_{i,n}$ adalah deretan kota (yang belum dikunjungi) yang dapat dicapai dari i pada transisi ke-n.
α (Alpha) dan β (Beta) adalah parameter yang menentukan signifikan relatif dari *arc fitness* dengan jarak heuristik.

Pij mempunyai nilai berbanding terbalik dengan *dij*. Semakin dekat jarak antara i dan j, semakin besar probabilitas j untuk dipilih sebagai kota berikutnya untuk dikunjungi. $\rho_{ij,n}$ mengukur probabilitas sesuai keberadaan antara dua buah kota. Secara alami, ketika lebah menemukan sumber makanan baru, mereka akan melakukan *waggle dance*. Jika lebah lain tertarik dengan tarian tersebut, mereka cenderung melakukan *forage* (mencari makanan) di daerah tersebut. Dalam algoritma *Bee Colony Optimization* (BCO), hal ini dimodelkan melalui implementasi sebuah *preferred path* yang dilambangkan dengan θ. θ adalah sebuah permutasi gerakan dari seekor lebah untuk melakukan pengamatan dari sarang lainnya. Jadi panduan dalam tahapan *forage* didefinisikan dengan rumus berikut ini:

$$\rho_{ij,n} = \left\{ \begin{array}{cc} \lambda & , j \in F_{i,n}, \left|A_{i,n}\right| > 1 \\ \dfrac{1-\lambda\left|A_{i,n} \cap F_{i,n}\right|}{\left|A_{i,n} - F_{i,n}\right|} & , j \notin F_{i,n}, \left|A_{i,n}\right| > 1 \\ 1 & , \left|A_{i,n}\right| = 1 \end{array} \right\} \quad 0 \le \lambda \le 1$$

Dimana λ menunjukkan probabilitas dari sebuah kota yang diikuti dalam θ. $F_{i,n}$ adalah suatu himpunan yang berisi suatu kota yang lebih disukai oleh lebah untuk berpindah dari pada transisi ke-n, seperti yang direkomendasikan oleh θ.

2. *Waggle Dance*

Ketika lebah menemukan sumber makanan baru, lebah akan melakukan *waggle dance*. Dalam algoritma BCO, tidak semua lebah yang menyelesaikan jalur akan menari. Hanya lebah yang menghasilkan panjang jalur terpendek dibandingkan dengan panjang jalur terbaik sebelumnya yang diijinkan untuk menari. Tetapi lebah dalam algoritma BCO dilengkapi dengan *memory* untuk mengingat panjang tur terbaik personal mereka yang diperoleh selama eksekusi algoritma. Jika lebah menari, *waggle dance* berakhir selama durasi tertentu yang dirumuskan dengan fungsi linear D_i. Menurut fungsi linear, jika lebah *i* memiliki Pf_i yang lebih tinggi, maka akan diberikan kesempatan untuk menari yang lebih lama (tarian muncul dengan lebih banyak iterasi). Jika tidak, maka lebah menari hanya dalam periode singkat. Pf_i melambangkan skor profitabilitas lebah *i*. Pf_{colony} menunjukkan profitabilitas rata-rata koloni lebah dan diperbarui setelah masing-masing lebah menyelesaikan jalur. *K* didefinisikan sebagai skala faktor yang mengendalikan besarnya durasi.

Profitabilitas lebah ke-i :

$$Pf_i = \frac{1}{L_i} \quad L_i = \text{tour length}$$

Profitabilitas rata-rata koloni lebah :

$$Pf_{colony} = \frac{1}{N} \sum_{i=1}^{N} Pf_i$$

Durasi Dance (menunjukkan waktu yang dibutuhkan lebah untuk melalui suatu rute):

$$Di = K.\frac{Pf_i}{Pf_{colony}}$$

K = faktor skalar
N = jumlah lebah yang ada pada *dance list*

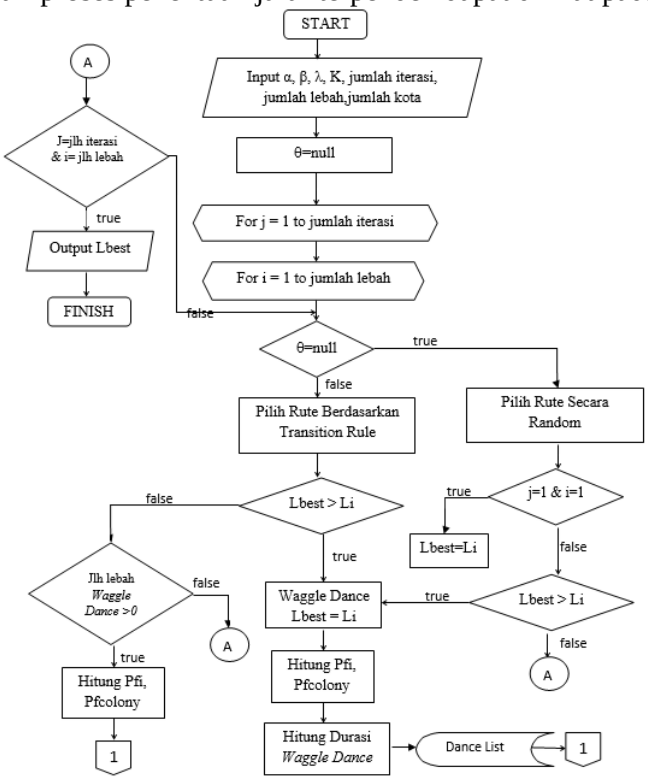
Pf_i dapat ditafsirkan sebagai kuantitas nektar yang dikumpulkan oleh lebah *i*. Kuantitas nektar yang lebih tinggi/banyak akan dikumpulkan jika lebah melakukan perjalanan dengan rute lebih pendek. Dengan demikian, Pfi didefinisikan berbanding terbalik dengan panjang tur. Sebelum lebah meninggalkan sarangnya, lebah akan melakukan pengamatan dan mengikuti tarian dari penari sebelumnya dengan probabilitas P_{follow}. Probabilitas P_{follow} disesuaikan secara dinamis mengikuti skor profitabilitas lebah dan koloninya berdasarkan Tabel 1. Lebah kemungkinan besar melakukan pengamatan secara acak dan mengikuti mengikuti *waggle dance* jika rating probabilitasnya rendah jika dibandingkan dengan probabilitas rata-rata koloninya [3].

Tabel 1. Tabel persentase lebah mengikuti dance list

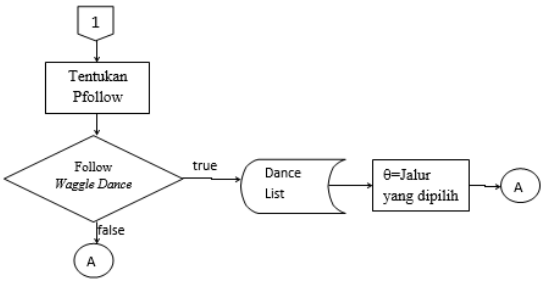
Profitability Scores	***Pfollow***
Pfi < 0.95 Pfcolony	0.80
0.95 Pfcolony ≤ Pfi < 0.975 Pfcolony	0.20
0.975 Pfcolony ≤ Pfi < 0.99 Pfcolony	0.02
0.99 Pfcolony ≤ Pfi	0.00

3. Metode Penelitian

Prosedur kerja dari proses penentuan jalur terpendek dapat dilihat pada Gambar 1 berikut:



Gambar 1(a). *Flowchart Bee Colony*

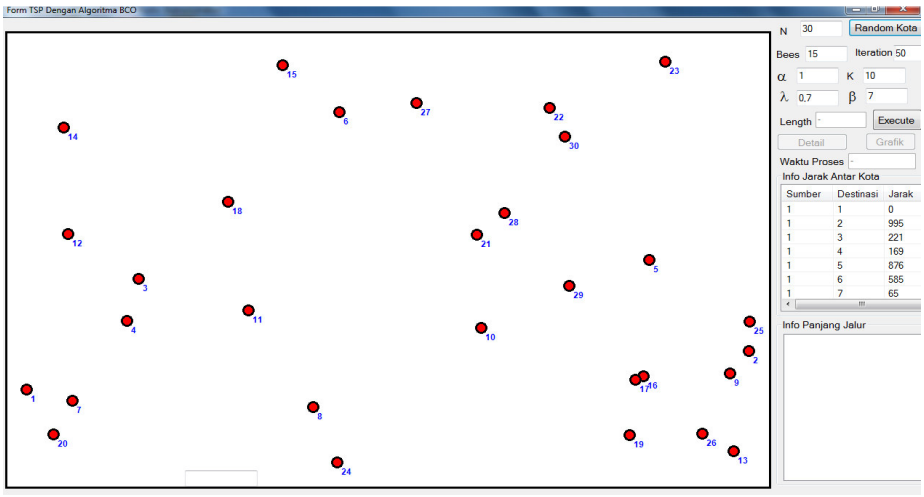


Gambar 1(b). *Flowchart Bee Colony*

4. Hasil dan Pembahasan

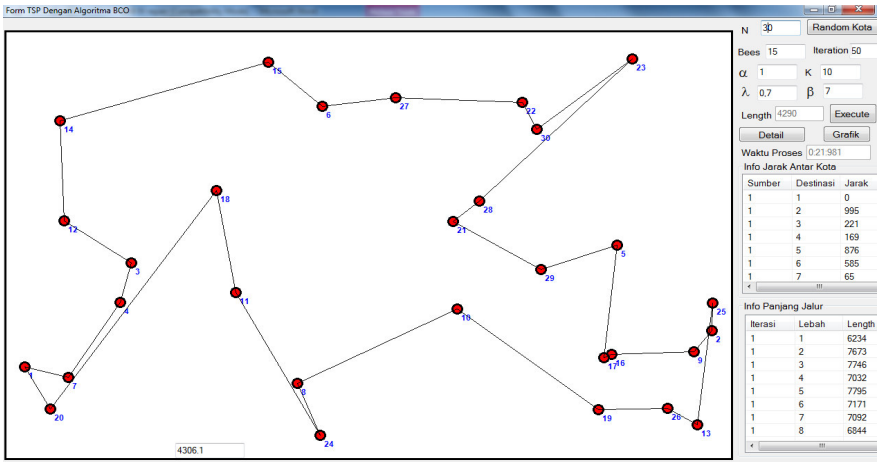
4.1. Hasil

Cara kerja aplikasi ini adalah input jumlah kota, lalu klik tombol *random* untuk mengacak N buah kota. Setelah itu akan muncul titik titik kota yang akan dilalui seperti Gambar 2.



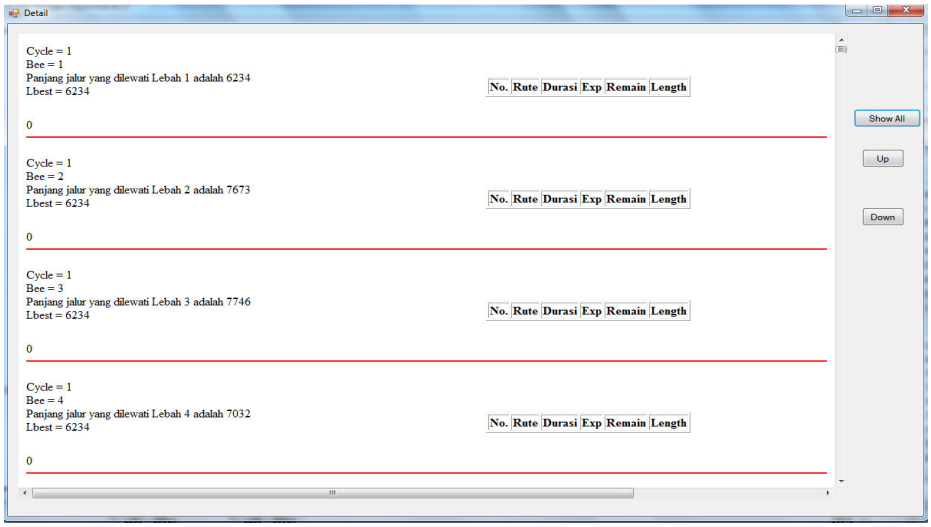
Gambar 2. Acak N Buah Kota

Setelah nilai parameter-parameter dimasukkan dan tombol *Execute* diklik maka akan dimunculkan berupa rute berupa graf yang akan dilalui seperti Gambar 3.



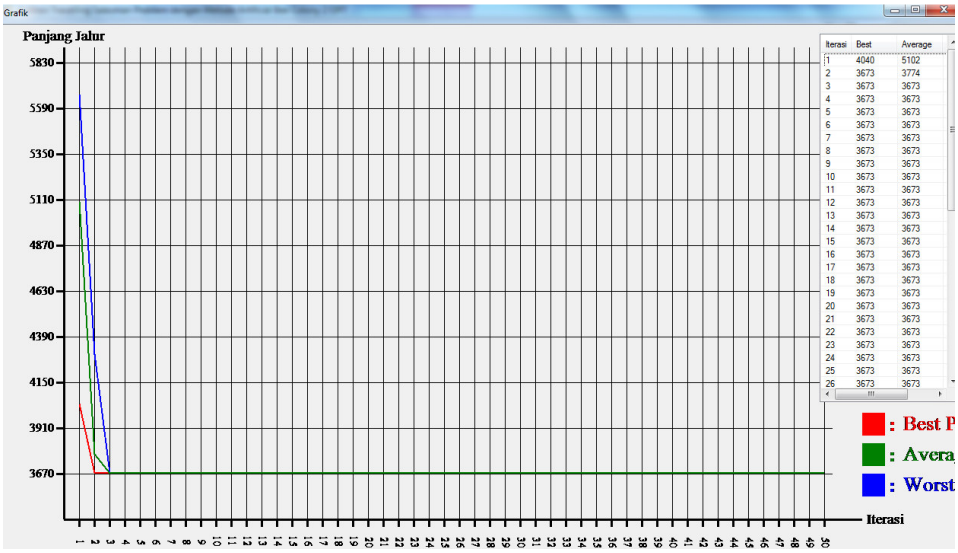
Gambar 3. Tampilan Rute berupa Graf

Pada saat tombol *Detail* diklik maka akan dimunculkan form yang berisi proses perhitungan seperti yang terlihat pada Gambar 4.



Gambar 4. Tampilan Proses Perhitungan

Pada saat tombol *Grafik* pada Gambar 3 diklik, maka akan dimunculkan berupa grafik panjang jalur dari setiap iterasi seperti terlihat pada Gambar 5 berikut:



Gambar 5. Tampilan Grafik dari Setiap Iterasi

4.2 Pembahasan

a. Hasil Pengujian Terhadap Jumlah Koloni yang Berbeda

Tabel 2 berikut ini merupakan hasil pengujian dengan jumlah koloni yang berbeda tetapi parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 50, Iterasi = 50, K = 100, λ (lamda) = 0.7, α (alpha) = 1, dan β (beta) = 7.

Tabel 2. Tabel Pengujian dengan Jumlah Koloni yang Berbeda

Percobaan	Nilai Masukan	Panjang jalur
1	5	5350.3
2	10	5342.4
3	15	5335.1
4	20	5334.4
5	25	5328.8
6	30	5326.6
7	35	5309.9
8	40	5202.3
9	45	5156
10	50	5113.4

b. Pengujian Terhadap Jumlah Iterasi yang Berbeda

Tabel 3 berikut ini merupakan hasil pengujian jumlah iterasi yang berbeda tetapi parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 50, Koloni = 50, K = 100, λ (lamda) = 0.7, α (alpha) = 1, dan β (beta) = 7.

Tabel 3. Tabel Pengujian dengan Jumlah Iterasi yang Berbeda

Percobaan	Nilai Masukan	Panjang jalur
1	10	5029.3
2	20	4926
3	30	4920.1
4	40	4914.9
5	50	4889.4
6	60	4868.8
7	70	4908
8	80	4840.8
9	90	4788.4
10	100	4704

c. Pengujian Terhadap Jumlah K (Faktor *Scalar*) yang Berbeda

Tabel 4 berikut ini merupakan hasil pengujian jumlah K (faktor *scalar*) yang berbeda tetapi parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 50, Koloni = 20, iterasi = 100, λ (lamda) = 0.7, α (alpha) = 1, dan β (beta) = 7.

Tabel 4. Tabel Pengujian dengan Jumlah K yang Berbeda

Percobaan	Nilai Masukan	Panjang jalur
1	10	4677.8
2	50	4680.8
3	100	4729.3

d. Pengujian Terhadap Jumlah λ (Lamda) yang Berbeda

Tabel 5 berikut ini merupakan hasil jumlah λ (lamda) yang berbeda dan parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 20, Koloni = 10, iterasi = 100, K = 100, α (alpha) = 1, dan β (beta) = 7.

Tabel 5. Tabel Pengujian dengan Jumlah λ yang Berbeda

Percobaan	Nilai Masukan	Panjang jalur
1	0.2	4522.6
2	0.5	4425.5
3	0.6	4437.5
4	0.7	4421.5
5	0.8	4477.8
6	0.9	4554.3

e. Pengujian Terhadap Jumlah α (Alpha) yang Berbeda

Tabel 6 berikut ini merupakan hasil pengujian jumlah α (alpha) yang berbeda tetapi parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 20, Koloni = 10, iterasi = 100, K = 100, λ (lamda) = 0.8, dan β (beta) = 8.

Tabel 6. Tabel Pengujian dengan Jumlah α yang Berbeda

Percobaan	Nilai yang di input	Panjang jalur
1	0	5276.9
2	1	5011.5

f. Pengujian Terhadap Jumlah β (beta) yang Berbeda

Table 7 berikut ini merupakan hasil pengujian jumlah β (beta) yang berbeda dan parameter lainnya yang bernilai sama yaitu N (jumlah kota) = 50, Koloni = 20, iterasi = 100, K = 20, λ (lamda) = 0.7, dan α (alpha) = 1.

Tabel 7. Tabel Pengujian dengan Jumlah β yang Berbeda

Percobaan	Nilai yang di input	Panjang jalur
1	2	4589.3
2	5	3995.6
3	7	3963.2
4	10	3977
5	15	3990.7
6	20	4014.4

5 Kesimpulan

Kesimpulan yang bisa diambil adalah:

a) Semakin besar nilai koloni, iterasi dan α (alpha) maka panjang jalur yang dihasilkan akan semakin optimal.
b) Jika nilai K (faktor skalar), λ (lamda) dan β (beta) yang diinput terlalu besar maka hasil yang diperoleh berupa panjang jalur akan kurang optimal.
c) Waktu proses dengan menggunakan algoritma *bee colony* dalam menyelesaikan kasus *Travelling Salesman Problem* jauh lebih cepat dibandingkan menggunakan proses manual.

Referensi

[1] Amin, A. dkk, 2005, *Travelling Salesman*, http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/MakalahStmik30.pdf [30 November 2011].
[2] Puspitorini, S., 2008, *Penyelesaian Masalah Travelling Salesman Problem dengan Jaringan Saraf Self Organizing*, http://journal.uii.ac.id/index.php/media-informatika/article/viewFile/108/68 [17 September 2011].
[3] Wong, L.P. dkk, 2008, *Bee Colony Optimization with Local Search for Travelling Salesman Problem* [6 Februari 2012].
[4] Wong, L.P. dkk, 2008, *.A Bee Colony Optimization Algorithm for Traveling Salesman Problem* [28 juli 2012].
[5] Nurdiana dan Dian, *Implementasi Algoritma Lebah untuk Pencarian Jalur Terpendek dengan Mempertimbangkan Heuristik,* http://repository.upi.edu/operator/upload/s_d545_060211_chapter2.pdf, [31 Juli 2012]