JKMA Jurnal Kajian Matematika dan Aplikasinya
VOLUME 1 NOMOR 1, July 2020
e-ISSN: 2722-7650. http://journal2.um.ac.id/index.php/jkma

TWO PHASE HEURISTIC ALGORITHM (TPHA) PADA MULTIPLE TRAVELLING SALESMAN PROBLEM (MTSP) DAN IMPLEMENTASI PROGRAMNYA

Rahma Try Iriani[1], Sapti Wahyuningsih[1,*], Darmawan Satyananda[1]
[1] Jurusan Matematika, FMIPA, Universitas Negeri Malang

Email: rahmatriiriani@yahoo.com (*R. T. Iriani*), sapti.wahyuningsih.fmipa@um.ac.id (*S. Wahyuningsih*), darmawan.satyananda.fmipa@um.ac.id (*D. Satyananda*)
* Corresponding Author

Abstract
Multiple Traveling Salesman Problem (MTSP) is one variant of Traveling Salesman Problem (TSP) which involves several salesmen in making a trip to visit several customers. In this article, the Two-Phase Heuristic Algorithm (TPHA) is used to solve MTSP problems. The algorithm classifies customers into several regions using the K-Means algorithm, which will then find a route solution for each region using a genetic algorithm. The MTSP problems that were resolved using TPHA were implemented into the Borland Delphi 7.0 programming language. Application testing was conducted using 21, 32, and 46 point cases.

Keywords: *distribution, multiple traveling salesman problems, two phase heuristic algorithm.*

Submitted: 28 January 2020; Revised: 20 April 2020; Accepted Publication: 19 May 2020;
Published Online: July 2020
DOI: 10.17977/um055v1i1p10-17

PENDAHULUAN

Salah satu kajian dalam teori *graph* yang membahas tentang penentuan rute yang optimum adalah *Travelling Salesman Problem* (TSP). Dalam aplikasinya diperlukan varian TSP yang dapat diaplikasikan sesuai keperluan, misalnya dalam hal distribusi. Penentuan rute dalam bidang pendistribusian merupakan salah satu aspek penting yang akan mempengaruhi biaya dan tingkat efisiensi suatu proses distribusi. Implementasi varian TSP untuk masalah distribusi telah dibahas dalam penelitian yang dilakukan oleh Wahyuningsih, et al. (2016).

Multiple Travelling Salesman Problem (MTSP) merupakan salah satu varian TSP dengan tambahan kendala yaitu terdapat lebih dari satu *salesman*. Latah (2016) mengatakan bahwa MTSP dapat diaplikasikan dalam berbagai hal sebagai pemecahan suatu permasalahan seperti masalah perjalanan, pengambilan, pengiriman, penjadwalan percetakan, dan penjadwalan karyawan. Menurut Xu, et al. (2017), permasalahan MTSP ini melibatkan *m salesman* melakukan sebuah perjalanan mengunjungi *n customer* yang berawal pada suatu depot, dimana *m salesman* tersebut harus melakukan kunjungan tepat satu kali ke setiap *customer* dalam suatu wilayah yang sudah ditentukan untuk masing-masing *salesman* dan harus kembali ke depot dalam suatu lintasan dengan jarak tempuh dan biaya seminimum mungkin. Tujuan dari MTSP adalah meminimumkan total jarak yang ditempuh oleh setiap *salesman*.

Terdapat beberapa penelitian terdahulu terkait algoritma yang digunakan untuk menyelesaikan permasalahan MTSP diantaranya adalah Alves & Lopes (2015) membahas tentang *Genetic Algorithm*, Latah (2016) membahas tentang *K-Means and Crossover based Modified ACO Algorithm*, Bolanos et al. (2016) yang membahas tentang *A Population-Based Algorithm*, selanjutnya Nuriyeva & Kizilates (2017) membahas tentang *A New Heuristic*

Algorithm, dan Sathya & Muthukumaravel (2017) yang membahas tentang *Two Phase Hybrid AI-Heuristics*.

Berdasarkan penelitian yang telah dilakukan oleh Xu, et al. (2017) menjelaskan bahwa MTSP dapat diselesaikan dengan menggunakan *Two Phase Heuristic Algorithm* (TPHA). Dalam tulisan tersebut dijelaskan bahwa TPHA menghasilkan solusi yang lebih optimum dibandingkan *Genetic Algorithm* (Algoritma Genetika), hal ini dikarenakan adanya perbedaan pada tahap inisialisasinya. Proses inisialisasi pengelompokan *customer* pada TPHA dilakukan dengan menggunakan algoritma K-Means, sedangkan pada algoritma genetika proses inisialisasi ini dilakukan dengan membangkitkan sejumlah bilangan random untuk tiap *salesman*.

Berdasarkan uraian di atas, pada artikel ini akan dikaji mengenai keunggulan TPHA pada MTSP dibanding algoritma genetika dan implementasi program TPHA pada MTSP.

METODE

TPHA pada MTSP

Two Phase Heuristic Algorithm merupakan salah satu metode untuk menyelesaikan permasalahan optimasi, dimana prinsip kerjanya adalah mengelompokan semua *customer* ke dalam beberapa wilayah dengan menggunakan algoritma K-Means, yang kemudian akan dicari solusi rute pada masing-masing wilayah dengan menggunakan algoritma genetika (Xu, et al. 2017). Penyelesaian dengan menggunakan TPHA pada permasalahan MTSP ini terdiri dari tahap inisialisasi pengelompokan titik, tahap proses pencarian solusi, dan kondisi optimum. Dalam tahap inisialisasi pengelompokan titik, algoritma K-Means yang merupakan algoritma pembentukan *cluster* (kelompok) digunakan untuk menempatkan *customer-customer* ke dalam beberapa wilayah. Selanjutnya proses pencarian solusi dilakukan dengan menggunakan algoritma genetika yang bertujuan mencari solusi rute untuk masing-masing wilayah. Proses pencarian solusi rute dengan menggunakan algoritma genetika dilakukan melalui beberapa tahapan utama yaitu pembangkitan populasi awal, evaluasi kromosom, seleksi kromosom, proses *crossover*, dan proses mutasi. Proses genetika ini dilakukan secara berulang sampai diperoleh kondisi optimum. Kondisi optimum tercapai saat semua generasi yang telah ditentukan sudah terlewati dan menghasilkan solusi terbaik berdasarkan nilai *fitness* yang diperoleh.

Berikut merupakan langkah-langkah penyelesaian permasalahan MTSP dengan menggunakan TPHA (Xu, et al. 2017).

1. Tahap Inisialisasi Pengelompokan Titik

Dalam mengelompokan titik ke dalam m himpunan dengan menggunakan algoritma K-Means, k titik dipilih secara random sebagai pusat *(centroid)* dari *cluster* yang berkorespondensi, yang selanjutnya mengidentifikasi semua titik-titik terdekat melalui fungsi *fitness* dan disesuaikan dengan lokasi *centroid*. Langkah ini diulang hingga kekonvergensian dari algoritma diperoleh. Misal $V = \{V_i : i = 1,2, \dots, n\}$ adalah himpunan dari n titik dan diasumsikan *centroid* pada *cluster* awal adalah $c_j, j = 1,2, \dots, k$. Berikut merupakan langkah-langkah pengelompokan titik dengan menggunakan algoritma K-Means:

Langkah 1: Menentukan jumlah *cluster.* Banyaknya *cluster* disesuaikan dengan jumlah *salesman*.

Langkah 2: Menentukan koordinat *centroid*. Dalam menentukan koordinat *centroid* untuk awal iterasi, dilakukan dengan memilih secara acak dari data koordinat titik. Sedangkan jika menentukan koordinat *centroid* yang merupakan tahap dari iterasi selanjutnya, maka koordinat diperbarui dengan menggunakan rumus sebagai berikut :

$$\begin{cases} \bar{x}_j = \dfrac{1}{|c_j|} \displaystyle\sum_{v_i \in c_j} x_i \\ \bar{y}_j = \dfrac{1}{|c_j|} \displaystyle\sum_{v_i \in c_j} y_i \end{cases}$$

dimana $\bar{x}_j$ adalah rata-rata *centroid* ke-j untuk variabel x, $\bar{y}_j$ adalah rata-rata *centroid* ke-j untuk variabel y, $|c_j|$ adalah jumlah titik pada *cluster* c_j, x_i adalah koordinat titik x ke i, dan y_i adalah koordinat titik y ke i.

Langkah 3: Hitung jarak masing-masing titik $v_i \in V$ terhadap pusat *cluster* atau *centroid* dengan menggunakan jarak Euclidean yaitu:

$$\|v_i - c_j\| = \sqrt{\left(x_i - \bar{x}_j\right)^2 + \left(y_i - \bar{y}_j\right)^2};\ i = 1, \dots, n;\ j = 1, \dots, k$$

Langkah 4: Pengelompokan titik. Untuk menentukan anggota *cluster* adalah dengan memperhitungkan jarak minimum titik. Nilai yang diperoleh dalam keanggotaan data pada matriks jarak adalah 0 dan 1, dimana nilai 1 untuk data yang dialokasikan ke *cluster* dengan jarak paling minimum terhadap salah satu *centroid* sedangkan nilai 0 untuk data yang dialokasikan ke *cluster* lain.

Langkah 5: Ulangi langkah 2, lakukan hingga koordinat *centroid* yang dihasilkan tetap dan anggota *cluster* tidak berpindah ke *cluster* lain.

Proses pengelompokan titik dengan menggunakan algoritma K-Means berhenti ketika anggota *cluster* yang telah terbentuk pada suatu iterasi sama dengan anggota *cluster* yang terbentuk pada iterasi sebelumnya.

2. Tahap Proses Pencarian Solusi

Proses pencarian solusi dilakukan dengan menggunakan algoritma genetika untuk menghasilkan solusi rute masing-masing *cluster*. Terlebih dahulu harus ditentukan parameter algoritma genetika yaitu jumlah generasi, jumlah individu, probabilitas *crossover* (P_c), dan probabilitas mutasi (P_m). Berikut adalah langkah-langkah mencari solusi menggunakan algoritma genetika.

Langkah 1 (Pembangkitan Populasi Awal):

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Teknik yang digunakan dalam membangkitkan populasi awal pada pembahasan ini adalah permutasi Josephus.

Langkah 2 (Evaluasi Kromosom):

Pada tahap evaluasi kromosom, dilakukan perhitungan nilai *fitness* dari semua kromosom yang merupakan total bobot pada kromosom. Nilai *fitness* yang digunakan sebagai ukuran seberapa baik sebuah individu menjadi solusi adalah sebagai berikut:

$fitness = \sum_{i=1}^{n} w(e_i)$, dengan e_i adalah bobot sisi v_{ij} , dimana v_{ij} menghubungkan titik v_i dan titik v_j.

Langkah 3 (Seleksi Kromosom):

Pada tahap seleksi kromosom semua individu akan diseleksi dengan metode *roullete wheel*, langkah-langkahnya adalah sebagai berikut:

1. Menghitung nilai invers $Q[k]$ masing-masing kromosom. Rumus untuk menentukan invers kromosom ke-k adalah $Q[k] = \frac{1}{fitness[k]}$.
2. Menghitung probabilitas $P[k]$ masing-masing kromosom dengan rumus: $[k] = \frac{Q[k]}{\sum_{k=1}^{n} Q[k]}$, dimana n = ukuran populasi.
3. Menghitung nilai probabilitas kumulatif:
   $C[1] = P[1]$
   $C[k] = C[k-1] + P[k], k = 2,3, \dots, n$ (n = ukuran populasi).

4. Membangkitkan bilangan random $R[k]$ dengan $0 \leq R[k] \leq 1$ sebanyak jumlah populasi.
5. Menentukan populasi baru dengan membandingkan bilangan random tersebut dan probabilitas kumulatif bobot setiap kromosom. Jika bilangan random $R[k]$ masuk dalam proporsi probabilitas kumulatif $C[k-1] < R[k] < C[k]$, maka kromosom ke-k diganti dengan kromosom yang mempunyai nilai probabilitas kumulatif $C[k]$ tersebut. Misal $R[1]$ kurang dari $C[4]$ dan lebih dari $C[3]$ maka kromosom ke-1 diganti dengan kromosom ke-4 pada populasi awal.

Langkah 4 (Proses *Crossover*):

Kemungkinan suatu kromosom akan mengalami proses *crossover* dan menjadi induk dipilih berdasarkan pada parameter probabilitas *crossover* (P_c) yang telah ditentukan pada awal tahap. Teknik *crossover* yang digunakan adalah teknik *Order Crossover*, langkah-langkahnya adalah sebagai berikut:

1. Membangkitkan bilangan random $R[k]$ dengan $0 \leq R[k] \leq 1$ sebanyak jumlah populasi.
2. Kromosom ke-k terpilih untuk menjalani proses *crossover* jika $R[k] < P_c$.
3. Membangkitkan bilangan random $R[k]$ antara 1 sampai banyaknya gen dalam satu kromosom untuk masing-masing kromosom yang terpilih untuk menjalani proses *crossover*.
4. Gen pada kromosom induk pertama diambil sesuai dengan bilangan random yang telah dibangkitkan kemudian ditukar dengan gen pada kromosom induk kedua yang belum ada pada kromosom induk pertama dengan tetap memperhatikan urutan.
5. Kromosom yang telah melalui proses *crossover* masuk dalam populasi baru

Langkah 5 (Proses Mutasi):

Pada proses mutasi jumlah kromosom yang mengalami mutasi dalam satu populasi ditentukan berdasarkan parameter probabilitas mutasi (P_m). Teknik yang digunakan adalah *swapping mutation*, langkah-langkahnya adalah sebagai berikut:

1. Menghitung panjang total gen yaitu mengalikan jumlah gen dalam 1 kromosom dengan jumlah individu.
2. Menghitung jumlah gen yang akan dimutasi yaitu mengalikan P_m dengan panjang total gen.
3. Membangkitkan bilangan random antara 1 sampai panjang total gen, sebanyak jumlah gen yang akan dimutasi dan menentukan posisi gen yang akan dimutasi.
4. Proses mutasi dilakukan dengan menukar gen yang telah dipilih dengan gen sesudahnya. Jika gen tersebut berada di akhir kromosom, maka gen itu ditukar dengan gen yang pertama.

3. Kondisi Optimum

Proses pencarian solusi pada masing-masing *cluster* akan membentuk sebuah populasi baru pada tiap generasi, yang kemudian populasi baru tersebut akan menjalani proses evaluasi kromosom, seleksi kromosom, *crossover*, dan mutasi kembali. Proses ini diulangi kembali hingga diperoleh individu yang memiliki solusi paling optimal setelah perulangan berturut-turut berdasarkan jumlah generasi yang telah ditentukan. Sehingga kondisi optimum tercapai saat semua generasi yang telah ditentukan sudah dilewati dan diperoleh solusi terbaik berdasarkan nilai *fitness* yang diperoleh.

Algoritma Genetika pada MTSP

Berikut merupakan langkah-langkah proses algoritma genetika (Mahmudy, 2008):

1. Tahap Inisialisasi Pengelompokan Titik

Tentukan terlebih dahulu parameter algoritma genetika yang terdiri dari jumlah generasi, jumlah individu, probabilitas *crossover* (P_c), dan probabilitas mutasi (P_m).

Langkah 1 : Pembangkitan populasi awal. Pengelompokan titik titik-titik ke dalam beberapa himpunan dilakukan pada pembangkitan populasi awal yang dilakukan berdasarkan nilai random yang dibangkitkan untuk masing-masing *salesman*. Dalam tahap inisialisasi ini, rute

yang akan dilalui oleh *salesman* direpresentasikan ke dalam kromosom yang terbagi menjadi dua segmen (bagian). Segmen pertama menunjukkan kumpulan rute yang akan dilalui masing-masing *salesman,* dimana masing-masing rute berisikan *customer-customer* yang telah dikelompokkan pada segmen kedua*,* pengelompokan *customer* pada segmen kedua ini ditentukan berdasarkan pembangkitan nilai random untuk masing-masing *salesman*.

2. Proses Pencarian Solusi

Langkah 2 : Evaluasi nilai *fitness* dari setiap kromosom.

Langkah 3 : Seleksi dengan metode *roullete wheel*, digunakan untuk mendapatkan kumpulan kromosom baru calon anggota populasi berikutnya.

Langkah 4 : *Crossover* dengan teknik *order crossover*, menghasilkan kromosom baru perkawinan silang dari induknya.

Langkah 5 : Mutasi dengan teknik *swapping mutation*, dilakukan dengan mengubah karakter-karakter dari kromosom secara acak.

3. Kondisi Optimum

Langkah 2 sampai langkah 5 di atas diulangi kembali hingga semua generasi yang telah ditentukan sudah dilewati, yang selanjutnya akan dihasilkan kromosom dengan nilai *fitness* tertentu yang merupakan solusi terbaik yang dihasilkan oleh algoritma genetika.

Implementasi Program

Berdasarkan langkah-langkah TPHA yang telah dipaparkan pada penjelasan di atas, tahap selanjutnya adalah merancang diagram alir (*flowchart*) dari TPHA secara sistematis. *Flowchart* merupakan suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. Melalui perancangan *flowchart* yang sistematis, langkah selanjutnya adalah mengubah *flowchart* tersebut ke dalam suatu bahasa pemrograman yaitu bahasa pemrograman *Borland Delphi 7.0.* Terdapat komponen-komponen utama di dalam program aplikasi MTSP-TPHA yaitu berupa *tabsheet* (lembar kerja) untuk input titik, hasil perhitungan K-Means, input parameter genetika, hasil perhitungan akhir yang menunjukkan solusi dari TPHA, dan *graph* hasil untuk memperlihatkan rute masing-masing *salesman*.

HASIL DAN PEMBAHASAN

Prosedur penyelesaian permasahan MTSP menggunakan TPHA dan algoritma genetika terdiri dari tahap inisialisasi pengelompokan titik, proses pencarian solusi, hingga diperoleh kondisi optimum. Terdapat perbedaan di antara TPHA dan algoritma genetika yaitu pada tahap inisialisasi pengelompokan titik, dimana pada TPHA pengelompokan titik dilakukan dengan menggunakan algoritma K-Means, sehingga semua *customer* akan dikelompokkan ke dalam beberapa wilayah sesuai dengan jumlah *salesman* yang ada. Sedangkan pengelompokan titik pada algoritma genetika dilakukan berdasarkan nilai random yang dibangkitkan untuk masing-masing *salesman*.

Berdasarkan perbedaan pada tahap inisialisasi ini, TPHA memiliki keunggulan dibandingkan dengan algoritma genetika, yaitu TPHA mampu menentukan *customer-customer* terdekat yang kemudian dikelompokkan ke dalam beberapa wilayah dengan menggunakan algoritma K-Means dan masing-masing wilayah akan ditugaskan pada masing-masing *salesman*. Sedangkan pada algoritma genetika, dengan adanya pembangkitan nilai random untuk menentukan pengelompokan *customer*-nya maka terdapat kemungkinan perubahan kelompok *customer* di setiap generasinya, sehingga memungkinkan untuk *salesman* harus mengunjungi *customer* yang berada sangat jauh dibandingkan *customer* lain yang berada dalam satu wilayah.

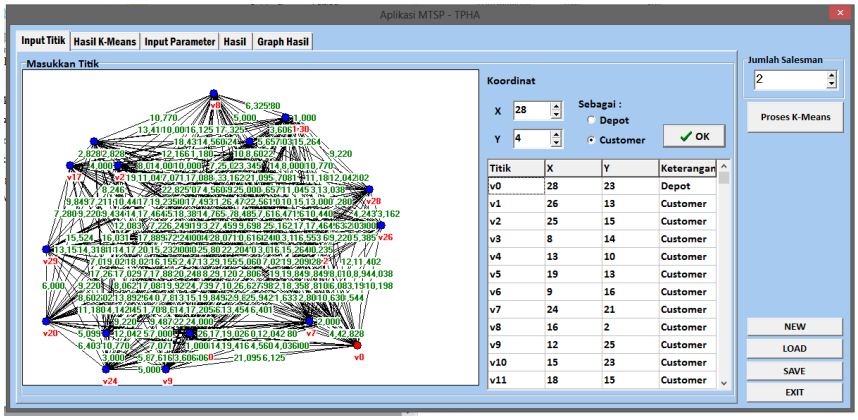
Berikut ini merupakan prosedur untuk proses perhitungan TPHA pada tahap inisialisasi pengelompokan titik dalam bahasa pemrograman *Borland Delphi 7.0.*

```
NoK :=True;
InitializeKMeans;
while NoK do
begin
    Inc(NumStep);
    UpdateMin;
    CalculateCentroid;
    UpdateSolution;
    if (tempCluster<> lastCluster ) then
       NoK := True
    else
    NoK := False;
    lastCluster := tempCluster;
end;
```

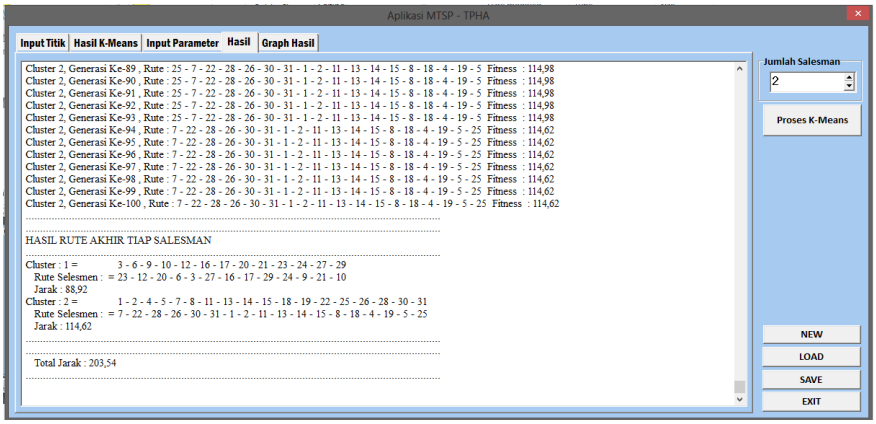
Implementasi TPHA pada MTSP dalam bahasa pemrograman *Borland Delphi 7.0* diuji coba menggunakan kasus 21 titik, 32 titik, dan 46 titik. Uji coba kasus 21 titik merupakan kasus yang dirujuk dari hasil penelitian Muzid (2014), sedangkan uji coba kasus 32 titik dan 46 titik merupakan data yang dihasilkan dari random koordinat titik pada program yang bertujuan untuk mengetahui kemampuan program aplikasi MTSP-TPHA (bisa dijalankan atau tidak) jika menggunakan inputan titik yang banyak. Uji coba semua kasus tersebut menggunakan nilai parameter genetika yang sama yaitu jumlah generasi sebanyak 100, jumlah individu sebanyak 100, probabilitas *crossover* adalah 0.1, dan probabilitas mutasi adalah 0.05 (Muzid, 2014).

Solusi yang baik akan diperoleh jika nilai parameter genetika dapat dirancang dengan baik. Ketika jumlah individu dan jumlah generasi semakin besar maka kombinasi kromosom semakin beragam dan peluang terpilihnya kromosom dengan *fitness* terbaik lebih besar. Serta perancangan yang baik untuk nilai probabilitas *crossover* dan mutasi, mengakibatkan semakin besar kesempatan kromosom untuk melakukan proses perkawinan, sehingga kemungkinan terbentuk solusi yang optimal akan semakin besar. Berdasarkan uji coba pengaruh perubahan nilai parameter genetika pada kasus 21 titik, diperoleh solusi optimum dengan 80 individu, 100 generasi, probabilitas *crossover* yaitu 0,95 dan probabilitas mutasi yaitu 0,1.

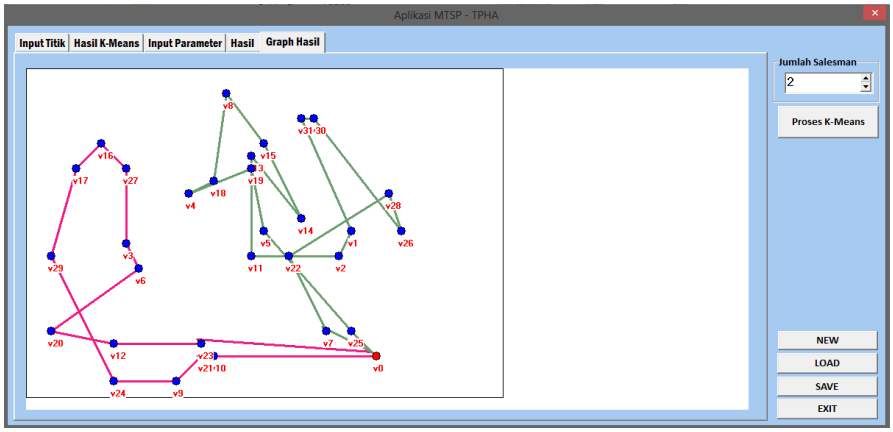
Tampilan hasil uji coba permasalahan MTSP yang diselesaikan dengan menggunakan aplikasi MTSP-TPHA untuk kasus 32 titik secara berturut-turut ditunjukkan pada Gambar 1, Gambar 2, dan Gambar 3 berikut.



Gambar 1. Tampilan Graph Kasus 32 titik



Gambar 2. Tampilan Hasil Akhir



Gambar 3. Tampilan Rute Tiap *Salesman*

PENUTUP

Penyelesaian dengan menggunakan TPHA pada permasalahan MTSP ini memiliki keunggulan dibandingkan algoritma genetika yang terletak pada tahap pengelompokan titik, dimana TPHA mampu mengelompokkan *customer* terdekat. Implementasi TPHA pada MTSP dalam bahasa pemrograman Borland Delphi 7.0 diuji coba menggunakan kasus 21 titik, 32 titik, dan 46 titik. Diharapkan untuk peneliti selanjutnya dapat mengembangkan algoritma yang digunakan pada tahap pengelompokan titik dan tahap pencarian solusi dengan dimodifikasi bersama algoritma lain untuk menyelesaikan permasalahan varian TSP lainnya.

DAFTAR RUJUKAN

Alves, R.M.F. & Lopes, C.R. 2015. Using Genetic Algorithms to Minimize The Distance and Balance The Routes For The Multiple Traveling Salesman Problem. *Faculty of Computing. Federal University of Uberlandia, UFU*. 3171-3178.

Bolanos, R.I., Toro, E.M., & Granada, M. 2016. A Population-Based Algorithm for The Multi Travelling Saleman Problem. *International Journal of Industrial Engineering Computation*. 245-256.

Latah, M. 2016. Solving Multiple TSP Problem by K-Means and Crossover based Modified ACO Algorithm. *International Journal of Engineering Research & Technology (IJERT), 5*(2), 430-434.

Mahmudy, W.F. 2008. Optimasi *Multi Travelling Salesman Problem* (M-TSP) Menggunakan Algoritma Genetika. *Seminar Nasional Basic Science V*, *1*(5), 1-6.

Muzid, S. 2014. Dinamisasi Parameter Algoritma Genetika Menggunakan *Population Resizing On Fitness Improvement Fuzzy Evolutionary Algorithm* (PROFIFEA). *Prosiding SNATIF Ke-1 Tahun 2014*.

Nuriyeva, F. & Kizilates, G. 2017. A New Heuristic Algorithm For Multiple Traveling Salesman Problem. *TWMS J. App. Eng. Math. 6*(1), 101-109.

Ponraj, R. & Amalanathan, G. 2014. Optimizing Multiple Travelling Salesman Problem Considering The Road Capacity. *Journal of Computer Science*, *10*(4), 680-688.

Sathya, N. & Muthukumaravel. Dr.A. 2017. Two Phase Hybrid AI-Heuristics for Multiple Travelling Salesman Problem. *International Journal of Applied Engineering Research*. *12*(22), 12659-12664.

Wahyuningsih, S., Satyananda, D., & Hasanah, D. 2016. Implementations of TSP-VRP Varian for Distribution Problem. *Global Journal of Pure and Applied Mathematic, 12*(1), 723-732.

Xu, X., Yuan, H., Liptrott, M., & Trovati, M. 2017. Two Phase Heuristic Algorithm For The Multiple-Travelling Salesman Problem. *Soft Computing*. Dari https://doi.org/10.1007/s00500-017-2705-5.