Pencarian Jalur Tercepat Rute Perjalanan Wisata Dengan Algoritma *Tabu Search*

Ivana Varita, Onny Setyawati dan Didik Rahadi

***Abstrak*- Salah satu permasalahan utama bagi wisatawan baik wisatawan domestik maupun manca negara adalah rute wisata yang harus mereka tempuh. Hal ini disebabkan oleh jumlah obyek wisata dan jalur alternatif yang banyak. Dalam penelitian ini, ditunjukkan bagaimana cara menyelesaikan kedua permasalahan ini dengan algoritma *Tabu Search* dan metode antrian yang diterapkan pada pencarian jalur tercepat wisata kota Malang. Algoritma ini dapat memberikan rute tercepat secara optimal dengan mendapatkan *cost* terendah tanpa perubahan nilai pada rentang iterasi 300 dengan percobaan pencarian jalur yang dipilih pada jalan-jalan pintu masuk kota Malang.**

***Kata kunci: wisata kota Malang, rute tercepat, tabu search*.**

I. PENDAHULUAN

WISATA adalah kegiatan perjalanan yang dilakukan oleh seseorang atau sekelompok orang dengan mengunjungi tempat tertentu untuk tujuan rekreasi, pengembangan pribadi, atau mempelajari keunikan daya tarik wisata yang dikunjungi dalam jangka waktu tertentu.

Akan tetapi, permasalahan yang sering muncul adalah minimnya informasi mengenai jalur menuju tempat wisata, tingkat kepadatan dan volume jalur tersebut. Dikarenakan waktu mereka yang terbatas, pencarian rute terpendek menjadi hal yang penting.

Pencarian rute terpendek adalah menentukan jalur yang paling optimal,yaitu jalur dengan rute terpendek dan biaya terkecil [1] dalam penerapannya dapat dilakukan pada satu atau lebih asal pencarian rute ke satu atau lebih tujuan melalui jaringan yang terhubung. Dalam beberapa aplikasi, juga bermanfaat untuk mengetahui jalur terpendek dua atau tiga alternatif tambahan misalnya, dalam rangka meningkatkan efektivitas pemberian informasi perjalanan, kebutuhan untuk memberikan beberapa jalur alternatif bagi pengguna jalan dalam mengemudi [2].

Dalam penentuan rute dapat memanfaatkan proses secara *metaheuristic* yaitu metoda untuk mencari solusi yang memadukan interaksi antara prosedur pencarian local dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik local optimal dan melakukan pencarian di ruang solusi untuk menemukan solusi global. Adapun varian dari metaheuristic antara lain *Simulated Annealing, Algoritma Genetika, Cross Entropy, Partical Swarm Optimation* dan *Tabu Search* [3]. Pencarian jalur tercepat rute perjalanan wisata kota Malang menggunakan *Tabu Search* karena *Tabu Search* menunjukkan kinerja yang baik untuk fungsi yang memiliki jumlah variabel lebih dari 10 [4], algoritma ini dapat mengoptimalkan biaya dan menghasilkan jalur biaya minimum dengan cakupan maksimum [5]., dalam kasus *vehicle routing problem* algoritma *hybrid Tabu Search* mampu meningkatkan solusi awal melalui prosedur *improving*. Hal tersebut diimplementasikan pada studi kasus tes 5-200 pelanggan. Algoritma ini bisa menemukan nilai optimum global sebesar 88% [6] dan lebih baik dari pada algoritma genetika[7].

Ivana Varita adalah Mahasiswa Program Magister Teknik Elektro, Universitas Brawijaya Malang, email: ivana_1924@yahoo.com
Onny Setyawati, adalah Dosen Jurusan Teknik Elektro, Program Magister Teknik Elektro, Fakultas Teknik, Universitas Brawijaya
Didik Rahadi S adalah dosen Jurusan Fisika, Fakultas MIPA, Universitas Brawijaya, Malang

II. TEORI *TABU SEARCH*

Algoritma *Tabu Search* adalah sebuah metode optimasi dengan prinsip dasar mengikuti kemampuan *local search* dengan tambahan kemampuan untuk menghindari optimum lokal dengan cara membiarkan *nonimproving value* bergerak kembali ke solusi sebelumnya dengan menggunakan memori yang disebut dengan *Tabu List*[8].

Terdapat tiga strategi utama yang digunakan dalam *Tabu Search*[9], yaitu :

a. Strategi pelarangan (*the forbidding strategy*) untuk mengontrol apa saja yang boleh masuk ke *Tabu List.*
b. Strategi pembebasan (*the freeing strategy*) untuk memutuskan apa saja yang boleh dikeluarkan dari *Tabu List* dan kapan pengeluaran dilakukan
c. Strategi jangka pendek (*the short-term* strategy) yang mengatur interaksi antara strategi pelarangan dan strategi pembebasan untuk membangkitkan dan menyeleksi solusi-solusi percobaan.

Tabu Search memiliki lima parameter utama, yaitu :

a. Prosedur *local search*,
b. Struktur *neighbourhood*, adalah suatu fungsi yang memetakan setiap solusi layak *S* ke solusi-solusi yang lainnya. Jumlah solusi layak dalam *neighborhood* biasanya dibatasi dengan menggunakan berbagai kriteria untuk

mengurangi waktu proses pencarian solusi.

c. Kondisi tabu, adalah pelarangan penggunaan solusi yang telah ditemukan sebelumnya.
d. Kondisi aspirasi, adalah pengecualian pengambilan solusi yang telah masuk dalam tabu
e. Kriteria penghentian.
Algoritma *Tabu Search* bisa dihentikan berdasarkan kriteria tertentu, misalnya sejumlah iterasi yang ditentukan *user*, sejumlah waktu CPU tertentu, atau sejumlah iterasi berurutan tanpa peningkatan nilai fungsi objektif terbaik.[9]

Parameter *Tabu Search* tidak benar-benar independen dari satu sama lain dan memungkinkan berefek pada parameter yang lain[10]. Pemberian antrian pada pencarian *neighbourhood* dapat mengurangi kompleksitas langkah pencarian [11].

Berikut disajikan pseudocode dari algoritma *Tabu Search* yang merupakan salah satu metode menuliskan algoritma yang menyerupai kode pemrograman sebenarnya.

```
s ← s0
        sBest ← s
       tabuList ← null
       while (not stoppingCondition())
         candidateList ← null
         for(sCandidate in sNeighborhood)
          if(not containsTabuElements(sCandidate,
   tabuList))
                candidateList ← candidateList +
   sCandidate
           end    end
      sCandidate ← LocateBestCandidate(candidateList)
        if(fitness(sCandidate) < fitness(sBest))
        then  Check Tabu status of swab
        If (swab tabu)
        Then Check aspiration criteria
            tabuList ← featureDifferences(sCandidate,
   sBest)
           sBest ← sCandidate
           while(size(tabuList) > maxTabuListSize)
              ExpireFeatures(tabuList)
           end
     end
end
      return(sBest)
```

III. METODE PENELITIAN

Pencarian jalur tercepat lokasi wisata kota Malang memerlukan beberapa data. Pada penelitian ini digunakan data sebagai berikut:

1. Data jalan yang diperoleh dari dinas Perhubungan
2. Data lokasi wisata yang diperoleh dari dinas Pariwisata.

Data jalan meliputi nama jalan, panjang jalan, kepadatan dan volume, sedangkan data lokasi wisata berupa alamat objek wisata yang terdapat di kota Malang.

A. Metode Pengolahan Data

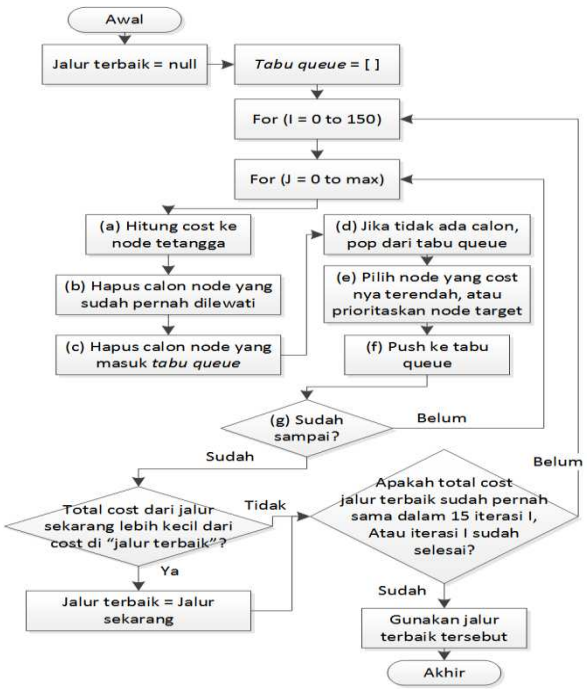
Pada tahap ini dilakukan berbagai pengumpulan informasi terkait beberapa hal berikut:

1. Pengumpulan informasi tentang jurnal atau hasil penelitian yang terkait dengan topik yang penelitian kemudian direview sebagai penelitian pendahulu.
2. Pengumpulan informasi tentang bagaimana membangun aplikasi pencarian jalur tercepat.
3. Pengumpulan informasi tentang metode yang akan digunakan.

B. Langkah Penelitian Dengan Algoritma Tabu Search.

Setelah mendapatkan data dan metode maka didapatkan konsep solusi dari aplikasi Algoritma *Tabu Search* untuk menentuan rute tercepat pencarian objek wisata dan variabel-variabel penelitian. Langkah penelitian dijabarkan sebagai berikut:

1. Pengumpulan Data
Data yang dibutuhkan untuk pengujian algoritma adalah data jalan dan lokasi wisata.
2. Penentuan Parameter
Pada tahap ini dilakukan penetapan parameter *Tabu Search*, diantaranya adalah jumlah populasi tiap iterasi (N), TabuListMemberCount (jumlah rute yang dibangkitkan *Tabu Search* dan disimpan dalam *Tabu List*), dan itmax_*Tabu* (iterasi maksimum *Tabu Search*).
3. Pembangkitan kandidat solusi *Tabu Search*
Pembangkitan kandidat solusi *Tabu Search* menggunakan pertukaran tetangga (*neighbourhood* selection) kemudian dilakukan pertukaran tetangga sehingga akan muncul n kemungkinan solusi.



Gambar 1. Kerangka Solusi Dengan Algoritma *Tabu Search*

4. Pemilihan solusi terbaik dari kandidat solusi. Kandidat solusi yang dibangkitkan akan dievaluasi fitness satu per satu dan akan diurutkan dari yang terkecil hingga terbesar. Rute terbaik akan dievaluasi apakah bisa masuk dalam *Tabu List* atau tidak. Namun dalam proses ini ditambahkan fungsi antrian dalam penentuan *neighbourhood*.
5. Update *Tabu List*
   Rute terbaik dari langkah 4 akan dilakukan evaluasi apakah lebih baik daripada rute yang sudah ada dalam *Tabu List* atau tidak. Jika rute yang dibangkitkan lebih baik daripada rute yang ada dalam *Tabu List*, maka rute akan dimasukkan dalam *Tabu List* menggantikan rute yang lebih buruk dalam *Tabu List*. Rute terbaik tersebut akan digunakan untuk pembangkitan kandidat solusi pada iterasi berikutnya. Namun, jika rute yang dibangkitkan tidak lebih baik daripada rute yang ada dalam *Tabu List*, maka solusi tidak akan dimasukkan dalam *Tabu List* dan akan dilakukan pembangkitan rute secara random untuk pertukaran tetangga pada iterasi berikutnya.
6. Pengecekan kriteria pemberhentian dengan mempertimbangkan maksimum iterasi yang ditetapkan pada langkah 2. Deskripsi tersebut direpresentasikan pada Gambar1.

IV. PENGUJIAN DAN PEMBAHASAN

A. *Persiapan* Data

Proses untuk menghitung nilai *cost* dari jalan menggunakan pembobotan rata-rata kemudian normalisasi , artinya sebuah nilai akan dibagi dengan nilai maksimal dari suatu *field*. Hasil nilai *cost* adalah semakin besar *cost* berarti semakin lama perjalanan menempuh jalan tersebut dan sebaliknya. Sehingga, algoritma pencarian jalur akan mencarikan *cost* terendah untuk ditempuh.

Pada data jalan, nilai panjang berkisar antara 0.08-4.27 km, kepadatan 12.76-133.33 satuan mobil penumpang (smp)/km, dan volume antara 299-2795 smp/jam. Sehingga, *cost* dari jalan dapat dirumuskan[12] :

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$$

$\bar{x}$ = rata-rata tertimbang
xi = nilai data ke-*i*
wi = bobot data ke-*i*
n = jumlah data

Dengan konsep normalisasi ini, nilai adalah antara 0 sampai 1. Khusus untuk volume, nilainya diinvers dimana volume besar akan menghasilkan nilai kecil dan volume kecil akan menghasilkan nilai *cost* besar. Karena semakin kecil volume akan semakin lama menempuh jalan tersebut.

Dengan rumus diatas, jika diurai, maka hasil akhirnya bernilai antara 0 sampai 100. Angka 100 merupakan konstanta untuk mempermudah perhitungan *cost*, sehingga tidak menyebabkan banyak angka di belakang koma.

B. *Pengujian* Algoritma

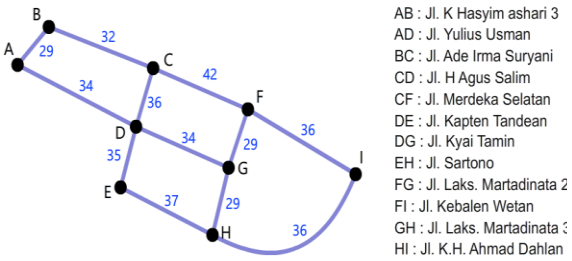
Pengujian ini bertujuan untuk mengetahui unjuk kerja dari algoritma *tabu search*.

Dalam proses pengujian ditetapkan parameter utama, yaitu:

a. Proses dihentikan berdasarkan sejumlah iterasi yang ditentukan. Iterasi dihentikan setelah berulang hingga iterasi berurutan tanpa peningkatan nilai fungsi objektif terbaik sebanyak 15 kali.
b. Klasifikasi suatu sub himpunan langkah di dalam suatu ketetanggan sebagai larangan atau *tabu* diimplementasikan dengan melarang jalur yang telah dilewati seperti jalur A ke B maka AB masuk kedalam *tabu*.
c. Suatu *Tabu List* mencatat langkah-langkah terlarang atau *tabu moves*.
d. Pengecualian batasan-batasan tabu bisa diberikan ketika suatu langkah *tabu* memberikan suatu solusi yang lebih baik dalam bentuk *cost* dibandingkan semua langkah terbaik sebelumnya, maka status *tabu* dari langkah tersebut bisa diabaikan (artinya: statusnya diubah dari *tabu* menjadi tidak *tabu*). Kondisi atau kriteria pengabaian status *tabu* ini disebut kondisi aspirasi.

Data diujicobakan dengan memberikan *threshold* 15 dan iterasi 30-300. *Threshold* dalam penelitian merupakan batasan pengulangan pencarian *cost* terendah. Iterasi merupakan proses pengulangan pencarian keseluruhan *cost* terendah.

Iterasi dalam penelitian ini digambarkan pada perncarian jalur dari A ke I pada Gambar 2. Dengan urutan langkah dari a hingga f akan berbeda tergantung kondisi node yang dilewati.



Gambar 2. Graf A-I

Proses iterasi graf A ke I adalah sebagai berikut :

a. Iterasi I-0 : Cari Jalur dari A ke I

1. Iterasi J-0
   Posisi di node A. Node tetangganya B dan D.*Cost* AB = 29, AD = 34*Tabu queue* masih kosong. Sehingga AB dan BC layak untuk dijalani.AB *cost* nya paling kecil. Maka bergerak ke B.*Tabu queue* sekarang = [AB]
   Jalur saat ini : A-B
2. Iterasi J-1
   Posisi di node B. Node tetangganya A dan C.*Cost* BA = 29, BC = 32. A sudah pernah dilewati. Hapus AB. BC ada di tabu queue?

Tidak. Hanya BC yang tersisa. Maka bergerak ke C.*Tabu queue* sekarang = [AB, BC]
Jalur saat ini : A-B-C
Dan seterusnya hingga Iterasi J-5, ditemukan jalur A-B-C-D-G-F-I, total *cost* 196. Tabu queue sekarang [AB, BC, CD, DG, GF, FI]
Jalur terbaik adalah A-B-C-D-G-F-I

b. Iterasi I-1 : Cari Jalur Alternatif dari A ke I

1. Iterasi J-0
   Posisi di node A. Node tetangganya B dan D.*Cost* AB = 29, AD = 34. AB ada dalam *tabu queue*. Hapus AB. Hanya AD yang tersisa. Maka bergerak ke AD. *Tabu queue* sekarang = [AB, BC, CD, DG, GF, FI, AD]
   Jalur saat ini : A-D
2. Iterasi J-1
   Posisi di node D. Node tetangganya A, C, E, G. *Cost* DA = 34, DC = 29, DE = 35, DG = 34. DA sudah pernah dilewati. Hapus DA. DG ada dalam *tabu queue*. Hapus DG. (Catatan : CD ada juga dalam *tabu queue*, namun karena bersifat *bidirectional*, CD dianggap berbeda dengan DC). Tersisa DC, DE. DE memiliki *cost* lebih rendah. Maka bergerak ke DE. *Tabu queue* sekarang = [AB, BC, CD, DG, GF, FI, AD, DE].
   Jalur saat ini : A-D-E

Dan seterusnya hingga Iterasi J-3, ditemukan jalur A-D-E-H-I, total *cost* 142. *Tabu queue* sekarang [AB, BC, CD, DG, GF, FI, AD, DE, EH, HI]. Jalur ini *cost* nya lebih rendah daripada "jalur terbaik". Maka jalur terbaik saat ini adalah A-D-E-H-I.

c. Iterasi I-2 : Cari Jalur Alternatif dari A ke I

1. Iterasi J-0
   Posisi di node A. Node tetangganya B dan D. *Cost* AB = 29, AD = 34. AB dan AD ada dalam tabu queue. Hapus AB dan AD.Tidak ada node tersisa, maka :Ambil node dari tabu queue dengan cara pop. Ambil AB karena berada di index terawal dibanding AD. Maka bergerak ke AB.Setelah pop AB, AB dimasukkan kembali dengan cara push. Maka tabu queue menjadi [BC, CD, DG, GF, FI, AD, DE, EH, HI, AB]
   Jalur saat ini : A-B
2. Iterasi J-1
   Posisi di node B. Node tetangganya A dan C. *Cost* BA = 29, BC = 32. A sudah pernah dilewati. Hapus AB. BC ada dalam tabu queue. Hapus BC. Tidak ada node tersisa, maka : Ambil node dari tabu queue dengan cara pop. Ambil BC, bergerak ke BC. Masukkan kembali BC ke tabu queue, menjadi [CD, DG, GF, FI, AD, DE, EH, HI, AB, BC]
   Jalur saat ini : A-B-C
3. Iterasi J-2
   Posisi di node C. Node tetangganya B, D dan F. *Cost* CB = 32, CD = 36, CF = 42. CB sudah pernah dilewati. Hapus CB. CD ada dalam tabu queue. Hapus CD. Tersisa jalur CF. Maka bergerak ke F.Daftarkan CF ke tabu queue, menjadi [CD, DG, GF, FI, AD, DE, EH, HI, AB, BC, CF].
   Jalur saat ini : A-B-C-F
4. Iterasi J-3
   Posisi di node F. Node tetangganya C, G dan I. Karena I merupakan node target, maka bergerak ke I, dan lanjut ke langkah f. Daftarkan FI ke *tabu queue*. Namun karena FI sudah pernah ada, *pop* dulu FI dan letakkan dengan *push*. *Tabu queue* sekarang [DG, GF, AD, DE, EH, HI, AB, BC, CF, FI].
   Jalur saat ini A-B-C-F-I, total *cost* 139.

Jalur ini memiliki *cost* lebih rendah dari pada "jalur terbaik". Maka jalur terbaik saat ini adalah A-B-C-F-I. Setelah melewati 50 iterasi, ditemukan alternatif-alternatif jalur sebagai berikut per iterasi I :

0. A-B-C-D-G-F-I, *cost* 196
1. A-D-E-H-I, *cost* 142
2. A-B-C-F-I, *cost* 139
3. A-D-C-F-I, *cost* 148
4. A-B-C-D-E-H-I, *cost* 205
5. A-D-E-H-I, *cost* 142
6. A-B-C-F-I, *cost* 139
7. A-D-C-F-I, *cost* 148
8. A-B-C-D-G-H-I, *cost* 196
9. A-D-E-H-I, *cost* 142
10. A-B-C-F-I, *cost* 139

Dalam kasus contoh ini, karena *edge* dan *node* yang dilalui sangat terbatas jumlahnya. Dalam kasus contoh ini, terdapat *cost* terbaik yaitu 139, angka 139 sudah menjadi *cost* terbaik mulai dari iterasi 6,7,8,9,10. Sehingga iterasi dihentikan. Maka jalur terbaik yang dipakai adalah A-B-C-F-I dengan *cost* 139.

Selanjutnya dilakukan analisa pencarian jalur yang telah dilakukan dengan proses komputasi:

1. Percobaan pertama :
   Perjalanan dari Jl. Mayjend Haryono 1 ke Wisata Belanja Tugu sebanyak 10 kali dengan 5 nilai iterasi yang berbeda berkisar 30-300 kali dan threshold 15, sebagaimana pada Tabel I.
2. Percobaan kedua :
   Perjalanan dari dari Jl. Kol. Sugiono 1 ke Wisata Taman Rekreasi Tlogomas, sebanyak 5 kali dengan 5 nilai iterasi yang berbeda berkisar 30-300 kali dan threshold 15, sebagaimana pada Tabel II.
3. Percobaan ketiga
   Perjalanan dari dari Jl. Raden Intan 1 ke Wisata Religi Klenteng En Ang Kiong sebanyak 5 kali dengan 5 nilai iterasi yang berbeda berkisar 30-300 dan threshold 15, sebagaimana padaTabel III.
4. Percobaan keempat
   Perbandingan hasil pencarian jalur tercepat Algoritma *Djikstra* dan *Tabu Search*. Hasil dari pengujian dapat dilihat pada Tabel IV. Percobaan pencarian jalur tercepat dipilih pada jalur masuk kota Malang dengan dengan pertimbangan wisatawan memasuki kota Malang dapat melalui jalur tersebut.

Dari hasil percobaan pertama, kedua dan ketiga menunjukkan nilai jumlah iterasi mempengaruhi jumlah *cost*[9]. Semakin besar iterasi akan mendapatkan *cost*

TABEL I
PERCOBAAN PENCARIAN JALUR TERCEPAT PERJALANAN DARI JL. MAYJEND HARYONO 1 KE WISATA BELANJA TUGU

No	Iterasi	Threshold	*Cost*
1	30	15	199.54
2	60	15	199.54
3	90	15	178.27
4	120	15	178.27
5	150	15	178.27
6	180	15	178.27
7	210	15	178.27
8	240	15	178.27
9	270	15	178.27
10	300	15	178.27

TABEL II
PERCOBAAN PENCARIAN JALUR TERCEPAT PERJALANAN DARI JL. KOL. SUGIONO 1 KE WISATA TAMAN REKREASI TLOGOMAS

No	Iterasi	Thresehold	*Cost*
1	30	15	756,04
2	60	15	756,04
3	90	15	756,04
4	120	15	710,12
5	150	15	710,12
6	180	15	710,12
7	210	15	710,12
8	240	15	710,12
9	270	15	710,12
10	300	15	710,12

TABEL III
PERCOBAAN PENCARIAN JALUR TERCEPAT PERJALANAN DARI JL. RADEN INTAN KE WISATA RELIGI EN ANG KIONG

No	Iterasi	Threshold	Cost
1	30	15	638,84
2	60	15	638,84
3	90	15	638,84
4	120	15	454,94
5	150	15	454,94
6	180	15	454,94
7	210	15	454,94
8	240	15	454,94
9	270	15	454,94
10	300	15	454,94

TABEL IV
PERCOBAAN PENCARIAN JALUR TERCEPAT 5.2.2 PERBANDINGAN ALGORITMA DJIKSTRA DAN TABU SEARCH

Pencarian	*Cost Tabu Search*	*Cost Dijkstra*
1	526	571
2	428.01	438
3	493.72	529
4	374.09	398
5	319.61	335

yang lebih rendah sehingga didapatkan jalur tercepat dengan *cost* terendah. Sebagaimana ketentuan *tabu search*, dalam penelitian ini jika hasil *cost* jalur terbaik sudah pernah sama sebanyak 15 kali. *Cost* terendah secara konsisten didapatkan pada iterasi ke 90 hingga iterasi ke 300 dengan nilai *cost* 178.27 pada percobaan pertama, *cost* terendah secara konsisten didapatkan pada iterasi ke 120 hingga iterasi ke 300 dengan nilai *cost* 710,12 dan *cost* terendah secara konsisten didapatkan pada iterasi ke 120 hingga iterasi ke 300 dengan nilai *cost* 454,94 pada percobaan ketiga.

Kondisi aspirasi diperoleh jika tidak terdapat node yang dapat dilewati.

V. KESIMPULAN

Beberapa hal yang dapat disimpulkan dari hasil desain, implementasi dan pengujian dari penelitian ini adalah pencarian jalur tercepat dengan parameter panjang, volume dan kepadatan jalan dapat diaplikasikan dengan algoritma *Tabu Search* dengan hasil jumlah iterasi dalam algoritma *Tabu Search* mempengaruhi jumlah *cost*. Semakin besar iterasi akan mendapatkan *cost* yang lebih rendah sehingga didapatkan jalur tercepat dengan *cost* terendah, yaitu iterasi I dibatasi 300, atau jika hasil *cost* jalur terbaik sudah pernah sama sebanyak 15 kali. Penambahan fungsi antrian yang diimplementasikan dalam *neighbourhood* berperan dalam mengurangi kompleksitas iterasi. Karena setiap parameter dalam *Tabu Search* mempengaruhi satu sama lain. Parameter dalam penelitian ini adalah iterasi , *threshold*, dan data yang digunakan.

Perbandingan hasil pencarian jalur tercepat dengan *Tabu Search* dan *Djikstra* menunjukkan unjuk kerja *Tabu Search* lebih baik dalam bentuk *cost* yang lebih rendah. Penelitian selanjutnya dapat menambahkan parameter jalan seperti kontur, aturan jalan searah – dua arah sehingga didapatkan hasil pencarian jalur yang mendekati fakta sebenarnya, sedangkan parameter kepadatan dan volume, dapat diukur secara berkala agar mendapatkan hasil yang lebih optimal

VI. REFERENSI

[1] Maki, M and Jose.I.2012. *Shortest Path Optimization Algorithms In Federated Cloud-Based Wireless Sensor Networks***.** International Journal of Electrical,Electronics and Data Communication. Vol.4,pp.41-46.

[2] Lim.Yand Kim.H.2005. *A Shortest Path Algorithm For Real Road Network Based On Path Overlap.* Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, pp. 1426 – 1438

[3] Santosa.B dan Willy.P. Metode Metaheuristik, Konsep dan Implementasi.GunaWidya.2011

[4] Rachid Chelouah and Patrick Siarry. 2000. *Tabu Search Applied to Global Optimization*. European Journal of Operational Research 123 .pp.256-270.

[5] Sharma.A, Jadhaf.A, Srivastava.P. R and Goyal.R, 2010. *Test Cost Optimization Using Tabu Search*. Software Engineering & Aplications 3,pp.477-486.

[6] Fard.M.K and Akbari.M.Reza.2013. *A Hybrid Tabu Search Algorithm for The Vehicle Routing Problem With Simultaneous Pickup and Delivery and Maximum Tour Time Length.* African Journal of Business Management Vol. 7(11), pp. 801-810.

[7] Bajeh,A.O,Abolarinwa, K. O. 2011. *Optimization: A Comparative Study of Genetic and Tabu Search Algorithms.* International Journal of Computer Applications (0975 – 8887) Volume 31– No.5.pp 43-48.

[8] Glover, F. 1989. *Tabu Search- Part I*, ORSA Journal on Computing1, 190-206.s

[9] Suyanto.2010.Algoritma Optimasi Deterministik atau Probabilistik. Yogyakarta: Graha Ilmu

[10] G. Paul . 2011 . *An Efficient Implementation of The Robust Tabu Search Heuristic for Sparse Quadratic Assignment Problems.* European Journal of Operational Research 209 215–218D

[11] Siamak Sarmady.*An Investigation on Tabu Search Parameters*. School of Computer Sciences, Universiti Sains Malaysia. sarmady.com /siamak/ papers/tabu-search.pdf.

[12] Syamsudin.2002.*Statistik Deskriptif*. Surakarta: Muhammadiyah University Press.