Penerapan Algoritma Nearest Neighbour untuk Menyelesaikan Travelling Salesman Problem

Imam Sutoyo

AMIK BSI JAKARTA
Jl. Raya Fatmawati No.24, Pd. Labu, Jakarta Selatan
e-mail: imam.ity@bsi.ac.id

Abstract *– Traveling Salesman Problem (TSP) is a problem that has gained much attention from researchers in the field of computer science and mathematics. Many algorithms have been introduced to solve this TSP problem. The solution to be obtained is the optimal solution in the sense of getting the shortest route that can be used to visit all points once only then back to the starting point of the journey. This paper discusses the application of one of the algorithms to obtain solutions for TSP problems, ie Nearest Neighbors (NN) algorithm. The application of the NN algorithm to solve TSP problems proved to be efficient even though there is no guarantee that the solution provided is the most optimal solution.*

Key Word: graph, algorithm, greedy, Nearest Neighbour, Traveling Salesman Problem

I. PENDAHULUAN

Travelling Salesman Problem (TSP) merupakan permasalahan yang kedengarannya sederhana namun telah menjadi permasalahan yang diteliti secara intensif di bidang matematika komputer (L. Applegate, 2007). Interpretasi yang paling umum dari permasalahan TSP yakni permasalahan seorang pedagang keliling yang berusaha mencari rute terpendek untuk mengunjungi sejumlah kota (Laporte, 1991).

Jadi, TSP adalah permasalahan dimana seorang salesman harus mengunjungi sejumlah kota untuk menjual barang dagangannya. Setiap kota hanya akan dikunjungi sebanyak satu kali dan setelah semua kota tersebut dikunjungi ia harus kembali ke tempat awal ia memulai perjalanan. Untuk efisiensi waktu, tenaga, dan biaya, harus dibuat rute yang optimal untuk melaksanakan perjalanan tersebut.

Rute yang optimal artinya rute yang paling hemat dalam hal penggunaan sumber daya. Rute yang paling hemat haruslah rute yang paling cepat sehingga efisien dalam hal waktu. Rute yang paling hemat juga haruslah rute yang paling mudah dilalui sehingga efisien dalam hal tenaga. Rute yang paling hemat juga haruslah rute yang paling murah sehingga efisien dalam hal biaya. Adapun secara sederhana, rute yang optimal adalah rute yang paling pendek. Penentuan rute yang paling pendek inilah solusi yang ingin didapatkan dalam banyak permasalahan TSP.

Prinsip dasar dalam menyelesaikan sebuah permasalahan dengan baik dan benar adalah dengan menggunakan algoritma yang efektif dan efisien. Efektif artinya algoritma dapat menyelesaikan permasalahan, sedangkan efisien artinya penggunaan sumber daya untuk menerapkan algoritma sebanding dengan manfaat yang didapatkan.

Pada paper ini akan dibahas penerapan algoritma *Nearest Neighbour* untuk menyelesaikan permasalahan TSP. Algoritma *Nearest Neighbour* merupakan alternatif dari algoritma *Brute Force* dalam menyelesaikan permasalahan TSP. Algoritma ini menawarkan penyelesaian permasalah TSP dengan efisiensi yang lebih tinggi meskipun tidak harus memberikan solusi yang paling optimal.

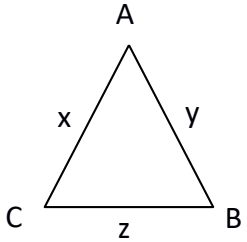
Pembahasan pada paper ini diatur secara sistematis. Pada bagian pertama akan dibahas landasan - landasan teori yang digunakan sebagai landasan penyelesaian permasalahan. Pada bagian kedua akan dibahas penerapan algoritma *Nearest Neighbour* untuk menentukan rute terpendek. Terakhir, akan dikemukakan kesimpulan yang diperoleh berdasarkan pembahasan pada bagian-bagian sebelumnya.

II. METODOLOGI PENELITIAN

A. Teori Graf

A.1. Definisi Graf

Graf adalah himpunan dari titik - titik dengan garis - garis yang menghubungkan sebagian titik - titik tersebut (Sipser, 2013). Titik pada graf disebut *node* atau *vertice*, sedangkan garis disebut *edge*. Jumlah dari garis yang terhubung ke sebuah titik disebut derajat dari titik tersebut (Sipser, 2013). Perhatikan contoh *graf* pada gambar 1 berikut ini.



Gambar 1. Contoh Graf sederhana

Graf tersebut memiliki tiga buah titik, yaitu A, B, dan C dan tiga buah garis, yaitu x, y, z. Garis pada graf tersebut juga dapat disimbolkan dengan dua titik yang dihubungkannya, yaitu AC, AB, dan BC.

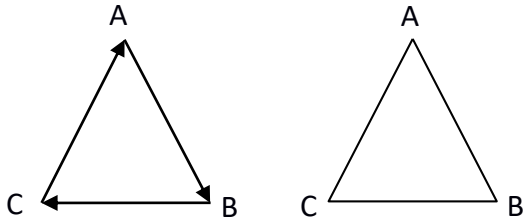
Secara matematis, graf dapat ditulis dalam bentuk formal: *G = (V, E)*. *V* adalah himpunan dari titik - titik, dan *E* adalah himpunan dari garis. Graf pada gambar 1 dapat ditulis dalam bentuk formal:

G = ({A, B, C},{(AC),(AB),(BC)})

A.2. Jenis - jenis Graf

a. Graf Terarah dan Graf Tidak Terarah

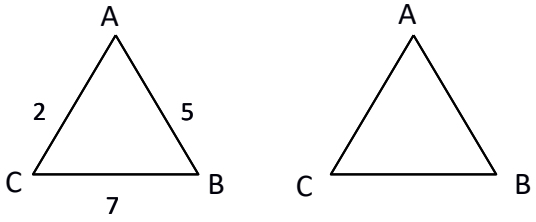
Graf terarah adalah graf yang garisnya digambarkan dengan tanda panah (Sipser, 2013). Tanda panah tersebut menunjukan arah tertentu. Graf tidak terarah tidak memiliki tanda panah pada garisnya. Perhatikan perbedaan kedua jenis graf ini pada gambar 2.



Gambar 2.a Gambar 2.b

b. Graf Berlabel dan Graf Tidak Berlabel

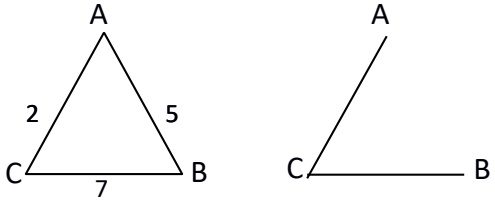
Graf berlabel adalah graf yang titik atau garisnya diberi label (Sipser, 2013). Label ini menunjukkan ukuran tertentu dari apa yang dimodelkan oleh titik atau garis pada graf tersebut. Perhatikan perbedaan kedua jenis graf ini pada gambar 3.



Gambar 3.a Gambar 3.b

c. Graf Lengkap dan Graf Tidak Lengkap

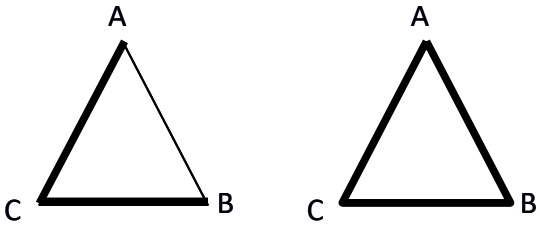
Graf dikatakan lengkap jika setiap titik pada graf memiliki hubungan lansung dengan semua titik lainnya pada graf tersebut. Jika kurang satu garis saja maka graf dikatakan tidak lengkap. Perhatikan perbedaan kedua jenis graf ini pada gambar 4.



Gambar 4.a Gambar 4.b

d. *Path* dan *Cycle*

Path adalah sederetan titik - titik yang dihubungkan dengan garis (Sipser, 2013). *Cycle* atau Sirkuit adalah *Path* yang titik awal dan akhirnya sama (Sipser, 2013). Perhatikan contoh *Path* dan *Cycle* pada pada gambar 5. ACB adalah *Path* sedangkan ACBA adalah *Cycle* atau Sirkuit.

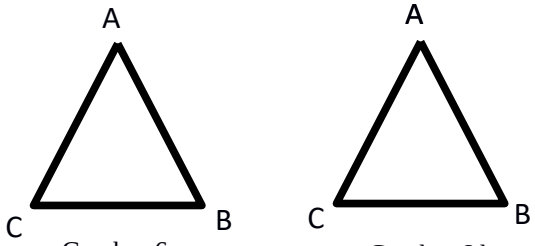


Gambar 5.a Path Gambar 5.b Cycle

Gambar 5. Perbandingan *Path* (5.a) dengan *Cycle* (5.b)

e. Sirkuit *Hamilton*

Sirkuit *Hamilton* adalah sirkuit pada graf yang melewati setiap titik pada graf hanya satu kali. Perhatikan contoh Sirkuit *Hamilton* pada gambar 6. ABCA dan ACBA adalah Sirkuit *Hamilton*.



Gambar 6.a ABCA Gambar 6.b ACBA

Gambar 6. Dua Sirkuit *Hamilton* pada sebuah Graf yang sama

B. Algoritma *Greedy*

B.1. Definisi Algoritma *Greedy*

Algoritma *Greedy* adalah salah satu pendekatan umum atau strategi untuk menyelesaikan permasalahan yang dapat diimplementasikan untuk menyelesaikan beragam permasalahan komputasi. Algoritma *Greedy* termasuk ke dalam jenis algoritma *heuristik*, yakni sebuah teknik algoritma yang didesain untuk menyelesaikan sebuah permasalahan secara cepat atau mencari pendekatan solusi jika tidak diketemukan solusi yang optimal untuk sebuah permasalahan.

Prinsip dasar algoritma *Greedy* adalah mengambil keputusan terbaik dengan cepat berdasarkan pertimbangan yang ada pada saat ini dengan berharap akan mendapatkan solusi terbaik tanpa mempertimbangkan seluruh pilihan yang ada.

 p-ISSN 1410-5063, e-ISSN: 2579-3500

Berdasarkan prinsip dasar tersebut algoritma *Greedy* merupakan algoritma yang cepat namun tidak menjamin akan mendapatkan solusi yang terbaik yang seharusnya bisa didapatkan. Meskipun demikian, algoritma ini terkadang menjadi pilihan satu-satunya pada permasalahan yang terlalu kompleks.

B.2. Algoritma *Nearest Neighbour*

Algoritma *Nearest Neighbour* merupakan contoh algoritma yang menerapkan prinsip algoritma *Greedy* untuk menyelesaikan permasalahan. Algoritma *Nearest Neighbour* mengambil pilihan terbaik berdasarkan data yang ada pada saat ini tanpa mempertimbangkan keseluruhan data yang ada. Algoritma *Nearest Neighbour* terus menunjukkan urgensinya pada banyak bidang ilmu pengetahuan dan teknik (Sameer, 1995).

Nearest Neighbour (NN) merupakan algoritma sederhana yang mudah untuk dipahami. Karena kesederhanaannya, NN mudah dan cepat untuk diimplementasikan. Dua hal inilah kelebihan utama dari NN.

Adapun timbal balik dari NN, ia tidak menjamin akan menemukan solusi yang terbaik yang sebenarnya solusi tersebut tersedia. Hal ini dikarenakan, NN tidak mempertimbangkan seluruh pilihan dari data yang tersedia dalam mengambil keputusan. Bahkan mungkin saja, NN akan memberikan solusi yang terburuk dari sebuah permasalahan.

III. HASIL DAN PEMBAHASAN

A. Permasalahan

A.1. Definisi Permasalahan

Permasalahan yang akan dibahas untuk dicari solusinya pada paper ini adalah permasalahan sederhana yang sangat mudah untuk dipahami dan dijabarkan. Namun, solusi terbaik untuk permasalahan ini ternyata memiliki kerumitan yang tinggi, terlebih pada saat permasalahannya semakin kompleks. Perhatikan ilustrasi dari permasalahan pada gambar 7 berikut ini.

Gambar 7. Sketsa permasalahan

Sketsa pada gambar 7 adalah sketsa dari sebuah wilayah yang memiliki lima kompleks perumahan. Seorang pedagang keliling berharap dapat mengunjungi seluruh perumahan tersebut untuk menjual barang dagangannya.

Untuk efisiensi waktu, tenaga, dan biaya, tiap perumahan hanya akan dikunjungi satu kali dalam sebuah rute perjalanan. Solusi terbaik untuk permasalahan ini adalah rute terpendek untuk mengunjungi seluruh perumahan yang dimulai dari rumah pedagang dan kembali lagi ke rumah pedagang.

A.2. Pengumpulan Data

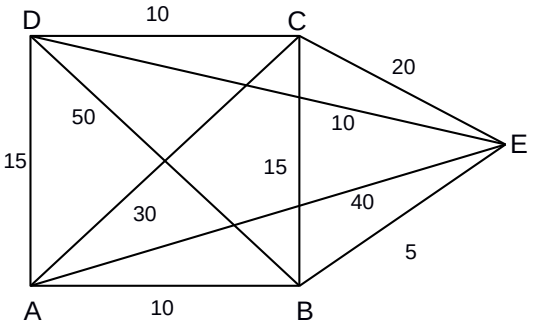
Langkah pertama adalah pengumpulan data dengan cara menghitung seluruh jarak antar perumahan. Data tersebut dapat ditampilkan dalam bentuk tabel berikut ini:

Tabel 1. Data jarak antarperumahan

	A	B	C	D	E
A	0	10	30	15	40
B	10	0	15	50	5
C	30	15	0	10	20
D	15	50	10	0	10
E	40	5	20	10	0

A.3. Permodelan Permasalahan

Langkah kedua adalah permodelan permasalahan menggunakan teori graf. Permasalahan ini sesuai dengan sketsa dan tabel data dapat ditampilkan dalam bentuk graf berikut ini:



Gambar 8. Model permasalahan

Graf permasalahan TSP merupakan graph lengkap yang tidak berarah (J.Rosenkrantz, 1977). Graf permodelan permasalahan pada gambar 8 menunjukkan hal ini. Permasalahan dimodelkan sebagai graf tidak terarah karena jalur antar perumahan dapat dilewati dalam dua arah yang berlawanan.

Graf permodelan permasalahan tersebut merupakan graf berlabel. Label pada garis graf menunjukkan jarak antar perumahan.

Graf permodelan permasalahan tersebut merupakan graf lengkap. Permasalahan dimodelkan

sebagai graf lengkap karena setiap perumahan memiliki jalur yang menghubungkannya dengan perumahan lainnya pada wilayah tersebut.

Graf permodelan permasalahan tersebut merupakan graf simetrik. Permasalahan dimodelkan sebagai graf simetrik karena jarak antar perumahan sama meskipun ditempuh dengan arah yang berlawanan. Contoh, jarak dari A ke B sama dengan jarak dari B ke A.

A.4. Solusi Permasalahan

Langkah ketiga adalah mencari solusi dari permasalahan. Solusi dari TSP yang telah dimodelkan dalam bentuk graf adalah sebuah Sirkuit *Hamilton* Terpendek (SHT). Untuk mendapatkan SHT tersebut kita gunakan algoritma *Nearest Neighbor* (NN).

Berikut langkah - langkah untuk mendapatkan SHT menggunakan algoritma NN:

1. Tentukan titik awal sirkuit yang sekaligus menjadi titik akhir dari sirkuit.
2. Dari titik awal tersebut tentukan titik - titik lain yang terhubung dengannya dan belum dikunjungi.
3. Pilih bobot terkecil dari garis yang menghubungkan titik awal dengan semua titik yang belum dikunjungi tersebut.
4. Titik dari garis yang terpilih menjadi titik awal pencarian berikutnya kemudian titik tersebut ditandai telah dikunjungi.
5. Ulangi langkah 2 sampai dengan 4 hingga tidak diketemukan lagi titik yang belum bertanda sudah dikunjungi.

Perjalanan mulai dari titik A. Titik A ini juga nanti akan menjadi titik akhir perjalanan, yakni perjalanan akan kembali ke titik awal perjalanan membentuk sebuah sirkuit.

Dari titik A ini maka ada 4 titik lain yang terhubung lansung, yaitu B, C, D, dan E. Keempat titik tersebut belum satupun pernah dikunjungi. Buat daftar jarak dari titik A ke keempat titik tersebut dengan merujuk ke Tabel 1. Berdasarkan daftar jarak dapat ditentukan jarak terkecil, yaitu dari titik A ke titik B sebesar 10.

Dengan demikian maka jalur pertama yang dilalui adalah A → B. Selanjutnya B akan menjadi titik awal penentuan jalur berikutnya dengan mengikuti algoritma penentuan jalur yang sama.

Dari titik B ada 4 titik lain yang terhubung lansung, yaitu A, C, D, dan E. Dari keempat titik tersebut maka titik C, D, dan E yang belum pernah dikunjungi sekalipun. Buat daftar jarak dari titik B ke ketiga titik tersebut dengan merujuk ke Tabel 1. Berdasarkan daftar jarak dapat ditentukan jarak terkecil, yaitu dari titik B ke titik E sebesar 5.

Dengan demikian maka jalur kedua yang dilalui adalah B → E. Selanjutnya, E akan menjadi titik awal penentuan jalur berikutnya dengan mengikuti algoritma penentuan jalur yang sama.

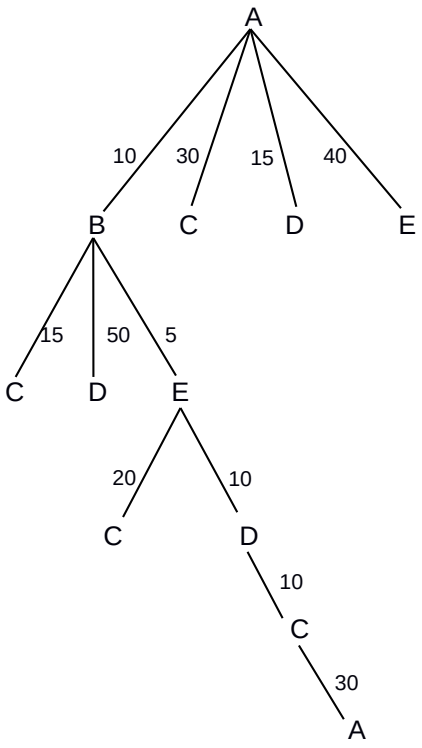
Dari titik E ada 4 titik lain yang terhubung lansung, yaitu A, B, C, dan D. Dari keempat titik tersebut maka hanya titik C dan D yang belum pernah dikunjungi sekalipun. Buat daftar jarak dari titik E ke kedua titik tersebut dengan merujuk ke Tabel 1. Berdasarkan daftar jarak dapat ditentukan jarak terkecil, yaitu dari titik E ke titik D sebesar 10.

Dengan demikian maka jalur ketiga yang dilalui adalah E → D. Selanjutnya, D akan menjadi titik awal penentuan jalur berikutnya dengan mengikuti algoritma penentuan jalur yang sama.

Dari titik D ada 4 titik lain yang terhubung lansung, yaitu A, B, C, dan E. Dari keempat titik tersebut maka hanya tinggal titik C yang belum pernah dikunjungi sekalipun. Dengan demikian tidak ada pilihan jalur lain kecuali D → C. Jarak D → C sebesar 10

Dengan demikian maka jalur keempat yang dilalui adalah D → C. Selanjutnya, C akan menjadi titik awal penentuan jalur berikutnya dengan mengikuti algoritma penentuan jalur yang sama. Namun, karena semua titik pada graf sekarang telah dikunjungi semua maka tidak tersisa satu jalurpun kecuali jalur C → A dengan jarak sebesar 30. Atau dengan kata lain, jalur perjalanan harus ditutup dengan kembali ke titik awalnya atau titik A untuk membentuk Sirkuit *Hamilton*.

Berikut ilustrasi dari langkah - langkah di atas untuk mendapatkan Sirkuit *Hamilton* menggunakan algoritma NN:



Gambar 9. Ilustrasi kerja algoritma NN

Berdasarkan ilustrasi pada gambar maka rute yang terpilih adalah ABEDCA. Total jarak rute tersebut adalah 10 + 5 + 10 + 10 + 30 = 65.

Rute ABEDCA dengan total jarak sebesar 65 ini adalah jarak yang harus ditempuh untuk

p-ISSN 1410-5063, e-ISSN: 2579-3500

mengunjungi seluruh titik pada graf dan kembali ke titik awal yang didapatkan menggunakan algoritma NN. Tidak ada jaminan bahwasanya rute tersebut adalah rute terpendek atau SHT yang bisa dibentuk pada graf.

Hal tersebut dikarenakan tidak semua kemungkinan jalur sudah dihitung. Untuk mendapatkan SHT yang sesungguhnya tidak mungkin dilaksanakan kecuali dengan membuat seluruh permutasi rute untuk membentuk SHT yang mungkin dibuat pada graf. Algoritma yang bekerja dengan membuat kemudian menghitung seluruh permutasi rute untuk membentuk SHT yang mungkin dibuat pada graf masuk ke golongan algoritma *Brute Force*.

Brute Force ini masuk ke dalam golongan yang disebut *exact algorithm*, yakni algoritma yang selalu menyelesaikan permasalahan optimisasi dengan solusi yang pasti optimal. Jadi, jika menggunakan *Brute Force* maka solusi yang didapatkan pasti dijamin optimal.

Lantas mengapa algoritma NN masih saja digunakan padahal algoritma ini tidak menjamin akan mendapatkan solusi yang optimal? Jawabannya adalah karena algoritma NN ini sangat efisien dalam metode penentuan rute, yakni penentuan rute pada NN dilaksanakan hanya dengan sekali jalan.

Untuk jumlah titik yang sedikit maka menghitung jarak perjalanan dari seluruh SHT yang bisa dibentuk pada graf bisa dilaksanakan demi untuk mendapatkan SHT yang paling optimal. Adapun, untuk jumlah titik yang makin banyak maka hal ini akan memakan waktu yang makin lama karena makin banyaknya permutasi rute yang mungkin untuk membentuk SHT pada graf.

Makin banyaknya permutasi rute yang mungkin untuk membentuk SHT pada graf untuk permodelan permasalahan TSP membuat solusi permasalahan TSP dikatakan semakin kompleks. Kompleksitas permasalahan tersebut bahkan bisa saja sampai pada titik dimana solusi optimal tidak mungkin didapatkan. Inilah mengapa permasalahan TSP masuk ke dalam golongan permasalahan *NP-hard*.

Untuk menyelesaikan permasalahan golongan *NP-hard* tidak bisa selalu digunakan *Brute Force*. Namun, bisa menggunakan *approximation algorithm* atau algoritma pendekatan. *Approximation algorithm* ini tujuannya adalah mendapatkan solusi dari permasalahan dengan efisien meskipun tidak dapat dipastikan solusi tersebut optimal.

Jadi, *approximation algorithm* ini merupakan alternatif dari *exact algorithm*. *Approximation algorithm* digunakan saat *exact algorithm* tidak bisa dilaksanakan dikarenakan permasalahannya terlalu kompleks atau masuk ke dalam kategori permasalahan *NP-hard*.

Prinsip *approximation algorithm* adalah mendapatkan solusi yang mendekati optimal dengan cara yang efisien. Jadi, *approximation algorithm* tidak bertujuan mendapatkan solusi yang paling optimal melainkan mencari solusi yang mendekati solusi optimal dengan cara yang efisien. Demikianlah cara kerja algoritma NN untuk menyelesaikan permasalahan TSP.

Dengan demikian, algoritma NN merupakan algoritma alternatif yang bisa digunakan untuk menyelesaikan permasalahan TSP disaat jumlah titik yang harus dikunjungi semakin banyak karena menggunakan algoritma *Brute Force* menjadi sangat tidak efisien.

Untuk penelitian selanjutnya bisa diteliti perbandingan antara *exact algorithm*, misalnya *Brute Force* dengan *approximation algorithm*, misalnya NN dalam berbagai permasalahan untuk mendapatkan algoritma yang paling sesuai dengan permasalahan yang ada.

IV. KESIMPULAN

Berdasarkan pembahasan yang telah dipaparkan dapat ditarik beberapa kesimpulan sebagai berikut:

1. Untuk menyelesaikan sebuah permasalahan dengan efektif dan efisien harus menggunakan algoritma yang efektif dan efisien pula.
2. Untuk menyelesaikan sebuah permasalahan kita tidak hanya berusaha mencari solusi yang paling optimal, namun perlu juga diperhitungkan kompleksitas permasalahan tersebut.
3. Jika permasalahan terlalu kompleks sehingga sulit untuk mencari solusi yang paling optimal maka masalah tersebut dapat diselesaikan dengan pendekatan dengan menggunakan algoritma yang lebih cepat meskipun tidak menjamin didapatkannya solusi yang optimal.
4. Algoritma *Nearest Neighbor* merupakan alternatif dari algoritma *Brute Force*.
5. Algoritma *Nearest Neighbor* meskipun tidak menjamin untuk mendapatkan solusi yag optimal namun mudah dan cepat untuk diterapkan.
6. Permasalahan TSP dapat diselesaikan dengan lebih cepat menggunakan algoritma *Nearest Neighbor*.
7. Menyelesaikan permasalahan TSP menggunakan algoritma *Nearest Neighbor* tidak menjamin didapatkannya rute yang paling optimal.
8. Meskipun algoritma *Nearest Neighbor* tidak menjamin didapatkannya rute yang paling optimal untuk permasalahan TSP, namun ia merupakan alternatif yang tepat disaat permasalahan TSP terlalu kompleks.

REFERENSI

J.Rosenkrantz, Daniel, Richard E. Stearn, & Philipp M. Lewis II (1977). *An Analysis of Several Heuristics*

for Traveling Salesman Problem. SIAM J. Computing, Vol. 6, No. 3, Sept. 1977, pp. 563–581.

L. Applegate, David, Robert E. Bixby, Vasek Chvátal & William J. Cook (2007). *The Traveling Salesman Problem*. Princeton University Press.

Laporte, Gilbert (1991). *The Traveling Salesman Problem: An overview of exact and approximate algorithms*. European Journal of Operational Research 59 (1992) 231-247.

Malkevitch, Joe (2015). *Sales and Chips*. Feature Column from the American Mathematical Society (AMS).

Sameer, A. Nene & Shree K. Nayar (1995). *A Simple Algorithm for Nearest Neighbor Search in High Dimensions*. Department of Computer Science Columbia University.

Sipser, Michael (2013). *Introduction to the Theory of Computation Third Edition*. Cengage Learning.

 p-ISSN 1410-5063, e-ISSN: 2579-3500