Jurnal Riset Komputer (JURIKOM), Vol. 5 No. 2, April 2018
ISSN 2407-389X (Media Cetak)
Hal: 136-139

Perbandingan Algoritma Prim Dengan Algoritma Floyd-Warshall Dalam Menentukan Rute Terpendek (*Shortest Path Problem*)

Zuhri Ramadhan[1], Muhammad Zarlis[2], Syahril Efendi[2], Andysah Putera Utama Siahaan[1]

[1] Fakultas Sains dan Teknologi, Universitas Pembangunan Panca Budi, Medan, Indonesia
[2] Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Medan, Indonesia

Abstrak

Masalah optimasi menjadi hal yang kompleks dalam mencari jalur atau rute optimal, banyak metode yang menjadi indikator rute optimal salah satunya adalah rute terpendek. Pencarian rute terpendek (shortest path) merupakan salah satu metode untuk menyelesaikan masalah rute perjalanan, metode shortest path problem dapat menggunakan berbagai macam algoritma seperti algoritma prim dan algoritma Floyd-warshall, namun algoritma mana diantara keduanya yang paling optimum dalam menentukan masalah rute terpendek. Dengan proses pencarian menggunakan graf dan dianalisa hasil dengan tabel kebenaran maka akan didapat hasil paling optimum diantara kedua algoritma.

Kata Kunci: Rute Optimal, Shortest Path Problem

Abstract

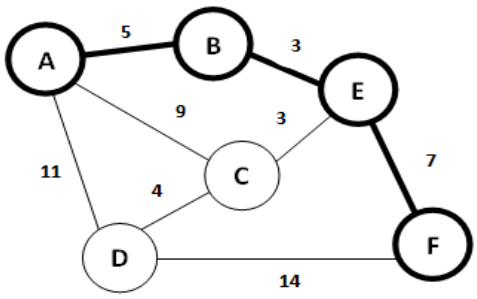
The optimization problem becomes complex in finding the optimal path or route, many methods become the optimal route indicator one of which is the shortest route. Shortest path search is one method to solve travel problem, shortest path problem method can use various algorithms such as algorithm prim and Floyd-Warshall algorithm, but which algorithm is the most optimum in determining the shortest route problem. With the search process using the graph and analyzed the results with the truth table will get the most optimum results between the two algorithms.

Keywords: Optimal Route, Shortest Path Problem

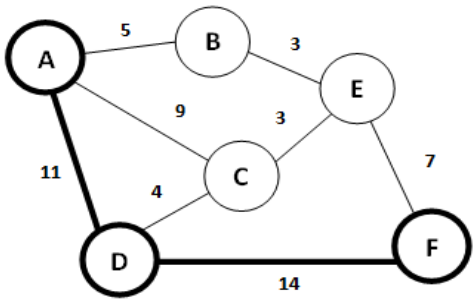
1. PENDAHULUAN

Banyak penelitian yang mengangkat persoalan optimasi dan metode - metode yang digunakan untuk menyelesaikan masalah tersebut. Seperti halnya dalam sebuah jaringan yang memiliki susunan garis edar (*path*) untuk menghubungkan banyak titik (*node*) [1]. Model jaringan yang akrab dalam kehidupan masyarakat seperti sistem jaringan jalan lintas dan tol, jaringan listrik dan telepon maupun jaringan rel kereta api. Optimasi merupakan upaya untuk memperoleh hasil yang terbaik dengan mempertimbangkan hambatan, kendala dan keterbatasan yang ada pada suatu persoalan, tujuannya adalah untuk meminimalkan atau menekan hal-hal yang merugikan dan memaksimalkan hal-hal yang dianggap menguntungkan [2].

Melihat dari definisi di atas, istilah optimasi dapat dikaitkan dengan persoalan rute optimum. Rute optimum dapat diartikan rute paling pendek atau rute paling efisien dari satu tempat ke tempat tujuan, akan tetapi kedua hal tersebut tidaklah sama, karena rute paling pendek belum tentu efisien dan rute paling efisien belum tentu rute paling pendek. Dilihat dari jumlah bobot (*weight*) setiap jalur (*edge*) dan jumlah simpul titik (*vertex*) yang dilalui. Semakin kecil jumlah bobot (Σw) dari jalur yang dilalui maka rute tersebut merupakan rute terpendek walaupun jumlah simpul titik (Σv) lebih banyak. Lihat gambar 1. Selain dari itu semakin kecil jumlah simpul titik (Σv) dari jalur yang dilalui maka rute tersebut merupakan rute paling efisien, lihat gambar 2.



Gambar 1. Graf Rute Paling Pendek



Gambar 2. Graf Rute Paling Efisien

Secara umum algoritma Floyd-Warsall banyak digunakan untuk menyelesaikan masalah jalur terpendek (*Shortesth Path Problem*). Seperti penelitian yang dilakukan oleh (Hougardy, 2010). *The Floyd-Warshall Algorithm on Graphs with Negative Cycles*. *Information Processing Letters*. (Magzhan dan Jani, 2013). *A Review dan Evaluations of Shortest Path Algorithms. International Journal of Scientific & Technology Research Volume*

2. (Singh dan Mishra, 2014). *Performance Analysis of Floyd Warshall Algorithm vs Rectangular Algorithm. International Journal of Computer Applications*.

Sementara algoritma prim sering digunakan menyelesaikan masalah MST (*Minimum Spanning Tree*) pada *graph* untuk meminimalkan rentang pohon atau mengurangi jumlah percabangan. Penelitian yang dilakukan oleh D. Kalpanadevi tahun 2013 dalam *International Journal of Computer and Organization Trends*, menggunakan algoritma prim untuk mencari jalur terpendek dan membuktikan bahwa algoritma prim dapat menemukan rute terpendek dengan jumlah *path* yang minimal dan durasi yang lebih kecil.

Yang menjadi permasalahan dalam penelitian ini adalah apakah algoritma prim dapat menyelesaikan masalah *shortest path problem* dan bagaimana membuktikan performa dari masing – masing algoritma dalam menyelesaikan masalah *shortest path problem.*

2. TEORITIS

2.1 Teori Graph

Teori *graph* merupakan salah satu cabang ilmu matematika yang sering diterapkan dan dimanfaatkan dalam kehidupan sehari-hari, ada beberapa contoh permasalahan yang bisa diselesaikan dengan menggunakan teori *graph*. Contohnya *travelling salesman problem*, pengaturan jadwal pegawai, arus lalu lintas, dll. Permasalahan di atas dapat dibentuk sebagai *graph* yang digambarkan dengan titik dan garis.

Graph adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lain melalui sisi/busur (*edges*). (Sedgewick. R, 2014). Suatu Graf G terdiri dari dua himpunan yaitu himpunan V dan himpunan E.
Verteks (simpul) : V = himpunan simpul yang terbatas dan tidak kosong.
Edge (sisi/ruas) : E = himpunan busur/garis yang menghubungkan sepasang simpul.

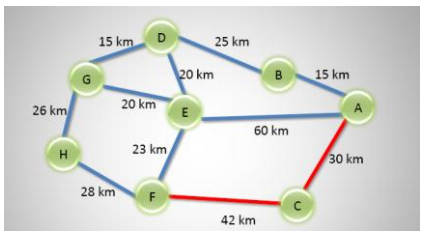
Simpul pada *graph* umumnya diberi identitas dengan huruf atau angka. Sedangkan sisi pada *graph* umumnya dinamai dengan himpunan simpul yang dihubungkan oleh sisi tersebut. Simpul-simpul pada *graph* dapat merupakan obyek sembarang seperti nama kota, atom atom suatu zat, nama anak, jenis buah, komponen alat elektronik dan sebagainya. Busur dapat menunjukkan hubungan (relasi) sembarang seperti rute penerbangan, jalan raya, sambungan telepon, jalur data, dan lain-lain.

2.2 Jalur Terpendek (*Shortest Path*)

Jalur terpendek (*shortest path*) adalah jalur optimum yang dapat diselesaikan dengan menggunakan *graph*. Jalur ini biasanya ditentukan oleh rute yang memiliki total biaya perjalanan yang paling kecil atau murah. Jika diaplikasikan dengan *graph* maka setiap garis pada simpul titik memiliki bobot berupa nilai dan apabila dijumlahkan bobot dari garis yang dilalui maka memiliki nilai yang minimal. Berikut adalah algoritma-algoritma untuk menyelesaikan masalah jalur terpendek.

2.3 Algoritma Floyd-Warshall

Algoritma ini ditemukan Robert W. Floyd pada tahun 1967. Algoritma Floyd-Warshall memiliki input graf berarah dan berbobot (*V,E*), yang berupa daftar titik (node/vertex *V*) dan daftar sisi (edge *E*). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada *E* diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi graf ini untuk memiliki siklus. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik. Implementasi algoritma ini seperti pada gambar 3, memperlihatkan jarak antar kota dari kota A untuk menuju kota F terdapat beberapa jalur, dapat melalui kota B terlebih dahulu, kota E, atau kota C. Pada algoritma ini dipilih jalur melalui kota C kemudian ke kota F sehingga jarak tempuh total adalah 72 km. Berbeda jika kita memilih kota B atau E terlebih dahulu, karena akan menghasilkan jarak tempuh yang lebih panjang.



Gambar 3. *Graf Dengan Bobot Jarak*

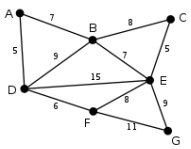
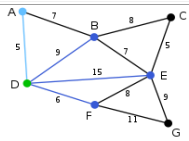
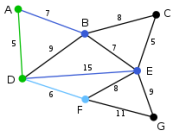
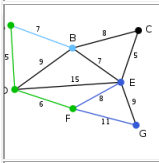
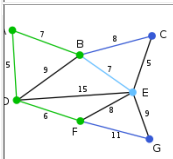
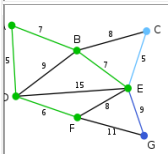
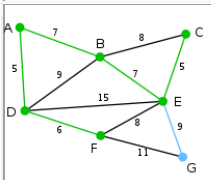
2.4 Algoritma Prim

Algoritma ini ditemukan pada tahun 1930 oleh matematikawan Vojtèch Jarnik dan kemudian secara terpisah oleh *computer scientist* Robert C. Prim pada tahun 1957 dan ditemukan kembali oleh Dijkstra pada tahun 1959. (Edmon, 2003). Algoritma prim merupakan algoritma dalam teori graf untuk mencari rentang pohon minimum untuk sebuah graf berbobot yang saling terhubung. Algoritma prim merupakan salah satu algoritma yang bekerja secara *greedy*.

Algoritma prim membentuk MST (*Minimum Spanning Tree*) langkah per langkah. Pada setiap langkah dipilih sisi graf G yang mempunyai bobot minimum dan terhubung dengan MST yang telah terbentuk. Algoritma prim digunakan untuk mencari pohon pembangkit minimum dari graf terhubung berbobot dengan cara mengambil sisi/ ruas garis yang memiliki bobot terkecil dari graf, diamana ruas garis tersebut bersisian dengan pohon terentang yang telah dibuat dan yang tidak membentuk siklus(Sedgewick, 2014).

Ini berarti sebuah himpunan bagian dari *edge* yang membentuk suatu pohon yang mengandung *node*, dimana bobot keseluruhan dari semua edge dalam pohon merupakan nilai minimal. Bila graf tersebut tidak terhubung, maka graf tersebut hanya memiliki satu pohon rentang minimum untuk satu dari komponen yang terhubung. Seperti pada tabel 1 berikut.

Tabel 1. Tabel Graph Pada Algoritma Prim

	Ini adalah graf berbobot awal. Graf ini bukan pohon karena ada circuit. Nama yang lebih tepat untuk diagram ini adalah graf atau jaringan. Angka-angka dekat garis penghubung adalah bobotnya. Belum ada garis yang ditandai, dan node **D** dipilih secara sembarang sebagai titik awal.
	Node kedua yang dipilih adalah yang terdekat ke **D**: **A** jauhnya 5, **B** 9, **E** 15, dan **F** 6. Dari ke empatnya, 5 adalah yang terkecil, jadi kita tandai node **A** dan cabang **DA**.
	Node berikutnya yang dipilih adalah yang terdekat dari **D** *atau* **A**. **B** jauhnya 9 dari **D** dan 7 dari **A**, **E** jauhnya 15 dan **F** 6. bobot 6 adalah yang terkecil, jadi kita tandai node **F** dan cabang **DF**.
	Algoritma ini berlanjut seperti di atas. Node **B**, yang jauhnya 7 dari **A**, ditandai. Di sini, cabang **DB** ditandai merah, karena baik node **B** dan node **D** telah ditandai hijau, sehingga DB tidak dapat digunakan.
	Dalam hal ini, kita dapat memilih antara **C**, **E**, dan **G**. **C** jauhnya 8 dari **B**, **E** 7 dari **B**, dan **G** 11 dari **F**. **E** yang terdekat, jadi kita tandai node **E** dan cabang **EB**. Dua cabang lain ditandai merah, karena kedua node yang terhubung telah digunakan.
	Di sini, node yang tersedia adalah **C** dan **G**. **C** jauhnya 5 dari **E**, dan **G** 9 dari **E**. **C**dipilih, jadi ditandai bersama dengan cabang **EC**. Cabang **BC** juga ditandai merah.
	Node **G** adalah satu-satunya yang tersisa. Jauhnya 11 dari **F**, dan 9 dari **E**. **E** lebih dekat, jadi kita tandai cabang **EG**. Sekarang semua node telah terhubung, dan pohon rentang minimum ditunjukkan dengan warna hijau, bobotnya 39.

3. ANALISA DAN PEMBAHASAN

3.1 Perancangan Graph

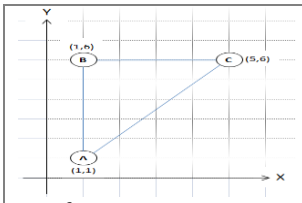
Graph yang digunakan dalam penelitian ini diperoleh dengan cara merancang berdasarkan sumbu koordinat X dan Y. Sumbu koordinat digunakan agar dapat menentukan bobot antar verteks dengan memanfaatkan metode *euclidean distance*. *Graph* yang akan digunakan sebagai sampel pengujian sebanyak sepuluh jenis *graph*, yang masing – masing memiliki bobot yang berbeda – beda dan akan dikelompokan ke dalam lima level. Hal ini dimaksudkan agar didapat pembuktian yang lebih empiris. Berikut data pengelompokan *graph* sesuai dengan jumlah verteks yang akan dibangkitkan pada setiap *graph* pada tabel 2 berikut.

Tabel 2. *Daftar Jumlah Verteks Setiap Graph*

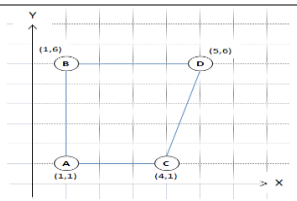
Level	Graph	Jlh Verteks (ΣV)	Jumlah Edge (Σe)	Notasi Verteks
I	G1	3	3	A, B, C
II	G2	4	4	A, B, C, D
	G3	4	5	A, B, C, D
III	G4	5	6	A, B, C, D, E
	G5	5	7	A, B, C, D, E
IV	G6	6	8	A, B, C, D, E, F
	G7	6	9	A, B, C, D, E, F
V	G8	7	10	A, B, C, D, E, F, G
	G9	7	11	A, B, C, D, E, F, G
	G10	7	12	A, B, C, D, E, F, G

Setelah jumlah verteks yang akan dibangkitkan sudah ditentukan maka langkah selanjutnya adalah mendesain *graph*. Sebagai acuan adalah sumbu koordinat maka digunakan tabel pada aplikasi AutoCAD yang disesuaikan lebar kolom dan row dengan jumlah pixel yang sama agar didapat jarak sumbu koordinat antara *x* dan *y* yang sama. Berikut gambar rancangan *graph* model 1 sampai dengan 10.

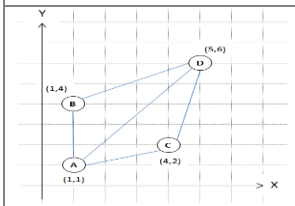
Tabel 3. *Graph Dengan Metode Euclidean Distance*



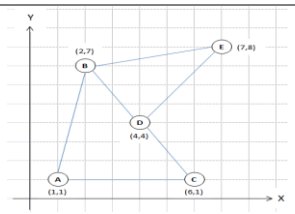
Graf 1



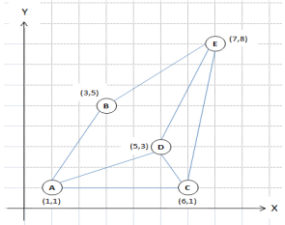
Graf 2



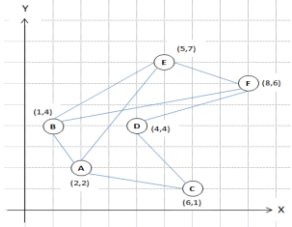
Graf 3



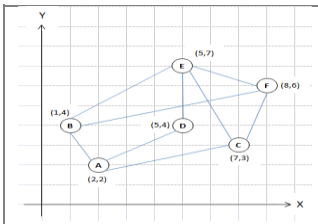
Graf 4



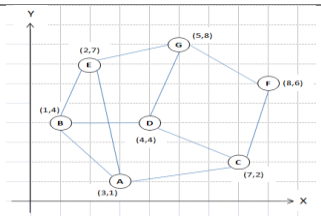
Graf 5



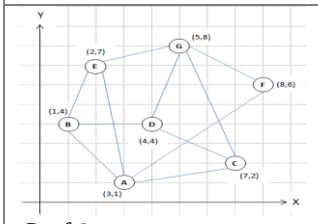
Graf 6



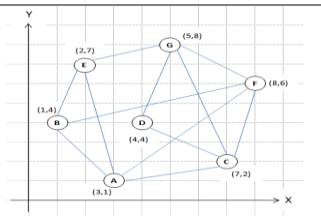
Graf 7



Graf 8



Graf 9



Graf 10

3.2 Pembobotan *Graph*

Pembobotan *graph* dilakukan dengan menggunakan metode *euclidean distance* yang biasa digunakan untuk menghitung panjang garis diagonal pada segitiga. Akan tetapi untuk mendapatkan jarak garis diagonal maupun garis lurus sejajar sumbu *x* dan *y*, titik – titik verteks harus dipresentasikan ke dalam bentuk koordinat 2 dimensi (*x* , *y*) sehingga dapat dihitung dengan menggunakan rumus sebagai berikut :

$$D_{A-B} = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \quad \text{.....(1)}$$

atau

$$D_{A-B} = \textbf{abs}\sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \quad \text{.....(2)}$$

Persamaan (2) digunakan apabila titik awal berada lebih jauh dari titik tujuan (*reverse*), sehingga pengurangan $x_2 - x_1$ atau $y_2 - y_1$ menghasilkan nilai negatif. Fungsi **abs** (*absolute*) pada persamaan (2) berfungsi untuk mengkonversi nilai negatif menjadi nilai positif.

Keterangan :
- D_{A-B} : Jarak titik A (awal) ke titik B (tujuan)
- x_1 : Koordinat titik awal pada sumbu x
- x_2 : Koordinat titik tujuan pada sumbu x
- y_1 : Koordinat titik awal pada sumbu y
- y_2 : Koordinat titik tujuan pada sumbu y

Pada *graph* ke-1 untuk menentukan pembobotan pada verteks AB, verteks BC dan verteks AC dengan koordinat titik A (1,1), titik B (1,6) dan titik C (5,6) maka perhitungannya sebagai berikut :

Diketahui :
- Koordinat A : (1,1)
- Koordinat B : (1,6)
- Koordinat C : (5,6)

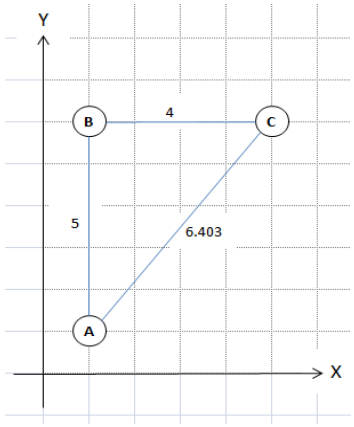
Ditanya :
- D_{A-B}
- D_{A-C}
- D_{B-C}

Jawab :
- ($x_1 = 1$) ($x_2 = 1$) ($y_1 = 1$) ($y_2 = 6$)
- $D_{A-B} = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$
  $= \sqrt{((1 - 1)^2 + (6 - 1)^2)}$
  $= \sqrt{(0^2 + 5^2)}$

$= \sqrt{25}$
$= 5$

- $(x_1 = 1)\ (x_2 = 5)\ (y_1 = 1)\ (y_2 = 6)$
- $D_{A-C} = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$
  $= \sqrt{((5-1)^2 + (6-1)^2)}$
  $= \sqrt{(4^2 + 5^2)}$
  $= \sqrt{16 + 25}$
  $= \sqrt{41}$
  $= 6.403$
- $(x_1 = 1)\ (x_2 = 5)\ (y_1 = 6)\ (y_2 = 6)$
- $D_{B-C} = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$
  $= \sqrt{((5-1)^2 + (6-6)^2)}$
  $= \sqrt{(4^2 + 0^2)}$
  $= \sqrt{16}$
  $= 4$

Maka bobot pada verteks AB = 5, verteks BC = 4 dan verteks AC = 6.403 seperti pada gambar 4 berikut :



Gambar 4. Bobot *Graph* ke-1 (G1)

3.3 Menentukan Bobot Optimum

Floyd- Warshall

Perhitungan dengan Floyd-Warshal terlebih dahulu menentukan titik awal dan titik akhir. Asumsinya semua titik pada *graph* dapat menjadi titik awal, tetapi dalam penelitian ini titik A menjadi titik awal dan titik dalam abjad terakhir menjadi titik akhir. Algoritma Floyd-Warshall menggunakan tabel *devide and concuer* dalam melakukan perhitungan yang jumlah kolom dan row-nya sesuai dengan jumlah verteks yang ada pad *graph*, seperti pada tabel 4.

Tabel 4. Perhitungan Dengan *Floyd-Warshall*

I	A	B	C
A	0	5	6.403
B	5	0	4
C	6.403	4	0

II	A	B	C
A	0	5	6.403
B	5	0	4
C	6.403	4	0

III	A	B	C
A	0	5	6.403
B	5	0	4
C	6.403	4	0

Hasil penjumlahan *graph* ke-1 dengan algoritma floyd-warshall dengan A sebagai titik awal dan C sebagai titik akhir. Verteks yang dibangkitkan adalah

- Hasil tabel I : (BC = 4); (CB = 4)
- Hasil tabel II : (AC = 6.403); (CA = 6.403)
- Hasil tabel III : (AB = 5); (BA = 5)

Jika dihitung *cost* nya adalah

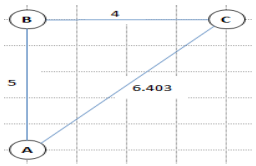
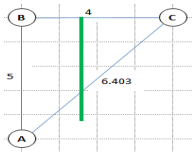
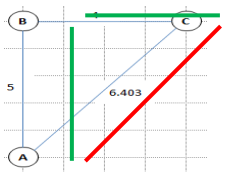
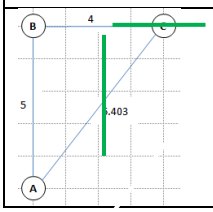
- Path I : A –B – C = 9
- Path II : A – C = 6.403

Hasil optimum rute A => C pada *graph* ke-1 adalah 6.403. karena *cost* lebih kecil dan jumlah verteks yang dibangkitkan juga lebih sedikit.

3.4 Prim

Dalam perhitungan algoritma prim untuk mencari bobot minimum biasanya menggunakan teknik meminimalkan jumlah *edge* pada *graph*. Jika pada graph memiliki verteks yang berecabang maka untuk mengambil keputusan rute mana yang harus dilalui terlebih dahulu adalah dengan memilih *edge* yang bobotnya lebih kecil, langkah ini biasa digunakan dalam algoritma *greedy*. Algoritma prim memanfaatkan fungsi heuristik dalam mengambil keputusan dan menggunakan teknik *backtracking* yakni dengan menjelajahi rute verteks yang masih memiliki pecabangan dengan jarak atau bobot yang lebih minimal lalu dilanjutkan ke verteks berikutnya atau kembali ke verteks sebelumnya jika tidak dijumpai bobot yang lebih kecil. Jalur yang akan dilalui disebut dengan *open list* sedangkan jalur didepannya yang akan dilalui disebut dengan *close list.* Dengan fungsi heuristik yang dimiliki maka algoritma prim dapat menghitung rute mana yang akan dilalui terlebih dahulu dengan mempertimbangkan bobot yang lebih kecil. Berikut adalah contoh *graph* yang akan dihitung bobot minimumnya. Lihat tabel 5 berikut.

Tabel 5. Perhitungan Dengan *Prim*

	Asumsikan titik awal adalah verteks A dan titik akhir adalah verteks C. Pada verteks A terdapat 2 rute yang akan di lalui yakni A – B dan A – C.
	Pilih bobot yang paling kecil untuk menentukan jalur pertama yang dilalui. Verteks pertama yang dipilih adalah B karena jarak yang lebih dekat A – B = 5. Jarak A – C dan B – C akan ditandai sebagai jalur *open list*.
	Selanjutnya adalah algoritma akan memilih jalur didepannya dan menimbang dengan jalur *open list* mana yang lebih dekat jaraknya. Maka jalur B – C lah yang dipilih dengan bobot 4. Karena jalur A – C merupakan *loop* mengingat sifat dari algoritma prim maka jalur A – C ditutup.
	Sekarang semua verteks telah terhubung dan jalur yang optimum adalah A – B – C = 9. Dengan verteks yang dibangkitkan yakni verteks A – B dan B – C.

3.5 Pengujian

Untuk memulai pengujian dengan menggunakan algoritma Floyd dimana semua titik pada *graph* dapat menjadi titik awal, tetapi dalam penelitian ini titik A menjadi titik awal dan titik dalam abjad terakhir menjadi titik akhir. Pengujian dilakukan dengan menggunakan 10 sampel *graph* dengan nama G1, G2, G3, G4, G5, G6, G7 G8, G9 dan G10. Yang ingin dicapai dalam pengujian ini adalah jalur optimum dengan jarak yang paling pendek. Hasil dapat dilihat pada tabel 6.

Tabel 6. Hasil Sampel *Graph* Dengan *Floyd-Warshall*

Graph	Aktif Verteks	Rute Optimum	Cost
G1	2	A -C	6.403
G2	2	A - C - D	8.099
G3	3	A - D	6.403
G4	4	A - B - E	11.181
G5	4	A - B - E	9.472
G6	4	A - E - F	8.992
G7	5	A - C - F	8.261

Graph	Aktif Verteks	Rute Optimum	Cost
G8	6	A – E – G	9.244
G9	7	A – E – G	9.244
G10	8	A – E – G	9.244

Untuk pengujian dengan algoritma prim akan digunakan tabel untuk setiap tahapan *graph* dalam mencari rute optimum dan meminimalkan aktif verteksnya. Pengujian dilakukan dengan menggunakan 10 sampel *graph* dengan nama G1, G2, G3, G4, G5, G6, G7 G8, G9 dan G10. Yang ingin dicapai dalam pengujian ini adalah jalur optimum dengan jarak yang paling pendek. Hasil dapat dilihat pada tabel 7.

Tabel 7. Hasil Sampel *Graph* Dengan *Prim*

Graph	Aktif Verteks	Rute Optimum	Cost
G1	2	A – B – C	9
G2	3	A – B – D	9
G3	3	A – C – D	7.285
G4	4	A – C – D – E	13.605
G5	4	A – D – E	9.857
G6	5	A – C – D – F	11.423
G7	5	A – D – E – F	8.992
G8	6	A – B – E – F – G	13.534
G9	6	A – B – E – F – G	13.534
G10	6	A – B – E – F – G	13.534

3.6 Perbandingan

Dari hasil pengujian *graph* 1 sampai 10 pada masing – masing algoritma maka didapat hasil berupa data aktif verteks yang dibangkitkan, jalur optimum yang di lalui beserta *cost* untuk jalur optimum tersebut. Dapat dilihat pada tabel 8 dimana menunjukan perbandingan aktif verteks yang dibangkitkan oleh masing – masing algoritma, pada *graph* 1 sampai dengan *graph* 10 setiap algoritma menunjukan performa yang berbeda – beda dalam mencari rute optimum dengan membangkitkan aktif verteks. Agar dapat mengetahui algoritma mana yang lebih optimum dalam membangkitkan aktif verteks maka diambil rerata dari hasil keseluruhan *graph*. Jika diambil rerata dari jumlah aktif verteks pada setiap *graph* maka didapat hasil dimana *floyd* mendapatkan nilai rerata 4,5 sedangkan untuk algoritma prim mendapatkan nilai rerata 4,4. Ini artinya algoritma prim lebih cepat dalam menentukan rute optimum dengan membangkitkan jumlah aktif verteks lebih sedikit.

Tabel 8. Perbandingan Aktif Verteks

	Floyd	Prim
G1	2	2
G2	2	3
G3	3	3
G4	4	4
G5	4	4
G6	4	5
G7	5	5
G8	6	6
G9	7	6
G10	8	6
SCORE	**4,5**	**4,4**

Pada tabel 8 menunjukan perbandingan aktif verteks yang dibangkitkan oleh masing – masing algoritma, pada *graph* 1 sampai dengan *graph* 10 setiap algoritma menunjukan performa yang berbeda – beda dalam mencari rute optimum dengan membangkitkan aktif verteks. Agar dapat mengetahui algoritma mana yang lebih optimum dalam membangkitkan aktif verteks maka diambil rerata dari hasil keseluruhan *graph*. Jika diambil rerata dari jumlah aktif verteks pada setiap *graph* maka didapat hasil dimana *floyd* mendapatkan nilai rerata 4,5 sedangkan

untuk algoritma prim mendapatkan nilai rerata 4,4. Ini artinya algoritma prim lebih cepat dalam menentukan rute optimum dengan membangkitkan jumlah aktif verteks lebih sedikit.
Untuk perbandingan jalur optimum yang di lalui pada algoritma prim dan floyd dapat dilihat pada tabel 9 berikut.

Tabel 9. Perbandingan Jumlah Verteks Optimum

	Floyd		**Prim**	
G1	A - C	2	A – B – C	3
G2	A - C - D	3	A – B – D	3
G3	A - D	2	A – C – D	3
G4	A - B - E	3	A – C – D – E	4
G5	A - B - E	3	A – D – E	3
G6	A - E - F	3	A – C – D – F	4
G7	A - C - F	3	A – D – E – F	4
G8	A – E – G	3	A – B – E – F – G	5
G9	A – E – G	3	A – B – E – F – G	5
G10	A – E – G	3	A – B – E – F – G	5
SCORE	2,8		3,9	

Pada tabel 9 menunjukan perbandingan jumlah verteks yang dilalui jalur optimum, jika diambil reratanya maka dalam hal ini algoritma floyd memiliki keunggulan dengan nilai rerata 2,8 dan prim memiliki nilai rerata 3,9. Ini artinya algoritma floyd memiliki jalur lebih cepat dalam mencapai tujuan akhir karena jumlah verteks yang dilalui pada jalur optimum lebih sedikit.
Selanjutnya perbandingan jumlah bobot atau *cost* rute optimum yang dilalui pada masing – masing algoritma dapat dilihat pada tabel 10.

Tabel 10. Perbandingan *Cost Route Optimum*

	Floyd	**Prim**
G1	6.403	9
G2	8.099	9
G3	6.403	7.285
G4	11.181	13.605
G5	9.472	9.857
G6	8.992	11.423
G7	8.261	8.992
G8	9.244	13.534
G9	9.244	13.534
G10	9.244	13.534
SCORE	**8654,3**	**9178,2**

Hasil analisis menunjukan jumlah bobot atau biaya yang didapat untuk melalui jalur optimum pada *graph* ke-1 sampai dengan *graph* ke-2 bahwasanya algoritma *floyd* diyakini lebih unggul dibandingkan algoritma prim, dilihat dari hasil nilai rerata pada algoritma Prim sebesar 9178,2 sedangkan algoritma Floyd sebesar 8654,3. Ini artinya algoritma floyd lebih efisien dalam menghasilkan *cost* rute optimum.

4. KESIMPULAN

Berdasarkan hasil dan pembahasan sebelumnya serta teori – teori pendukung lainnya, maka dapat ditarik kesimpulan, sebagai berikut :

1. Algoritma Prim yang banyak digunakan untuk menyelesaikan masalah MST – *Minimmum Spanning Tree* telah terbukti dapat menyelesaikan masalah SPP – *Shortest Path Problem*.
2. Algoritma prim lebih cepat dalam menentukan rute optimum dengan membangkitkan jumlah aktif verteks lebih sedikit. Paling cepat dalam mengambil keputusan.
3. Algoritma floyd memiliki jalur lebih cepat dalam mencapai tujuan akhir karena jumlah verteks yang dilalui

pada jalur optimum lebih sedikit dan algoritma floyd lebih efisien dalam menghasilkan *cost* rute optimum.

4. Algoritma floyd masih lebih baik dalam menyelesaikan masalah rute terpendek tetapi lambat dalam menentukan mana yang menjadi rute optimum sedangkan algoritma prim lebih cepat dalam menentukan rute optimum tetapi tidak efisien dan rute optimumnya bukan menjadi jalur terpendek dan tercepat.

REFERENCES

Cormen Thomas H., Leiserson Charles E., Rivest Ronald L., Stein Clifford . 2001. Introduction to Algorithm. McGraw-Hill.

Edmond., Jeff. 2008. How to Think About Algorithms. Cambridge University Press. Cambridge. UK.

Sedgewick, R. "Algorithms, 4th Edition".Addison – Wesley Professional, England 2011.

Hougardy, S. "The Floyd–Warshall Algorithm on Graphs with Negative Cycles, Elsevier Journal of Information Processing Letters, Vol.110, pp.279-281, April 2010.

Magzhan, K. Jani, H M. "A Review dan Evaluations of Shortest Path Algorithms". International Journal of Scientific and Technology Research (IJSTR 2013), Vol 2. Pp. 99-104, June 2013.

Singh, A. Mishra, P K. "Performance Analysis of Floyd Warshall Algorithm vs Rectangular Algorithm". International Journal of Computer Applications (IJCA), vol.107, pp.23-27, December 2014.

Sartono B, Rahardiantoro S, "An Implementation of Genetic Algorithm to Generate the Most Compromised Decision when Information of the Alternativesis Incomplete" Journal of Advances on Statistical Methods in Engineering, Science, Economy, Education and Disaster Management (SESEE 2015), no. 9, pp. 49-52, September 2015.

Adipranata R, Handojo A, Setiawan H,. Aplikasi Pencarian Rute Optimum Pada Peta Guna Meningkatkan Efisiensi Waktu Tempuh Pengguna Jalan Dengan Menggunakan A* dan Best First Search, Jurnal Informatika, Vol.8, no.2, pp.94-99, November 2007.

M. Iqbal, A. P. U. Siahaan, N. E. Purba and D. Purwanto, "Prim's Algorithm for Optimizing Fiber Optic Trajectory Planning," International Journal of Scientific Research in Science and Technology, vol. 3, no. 6, pp. 504-509, 2017.

A. P. U. Siahaan, "Heuristic Function Influence to the Global Optimum Value in Shortest Path Problem," IOSR Journal of Computer Engineering, vol. 18, no. 5, pp. 39-48, 2016.

A. P. U. Siahaan, "Adjustable Knapsack in Travelling Salesman Problem Using Genetic Process," The International Journal Of Science & Technoledge, vol. 4, no. 9, pp. 46-55, 2016.