E-Jurnal Matematika Vol. 14(3), Agustus 2025, pp. 92-97 ISSN: 2303-1751
DOI: https://doi.org/10.24843/MTK.2025.v14.i03.p485

PENCARIAN LINTASAN TERPENDEK DENGAN ALGORITME DIJKSTRA DAN MINIMUM SPANNING TREE DENGAN ALGORITME SOLLIN TERHADAP PERJALANAN WISATA SEJARAH DI KABUPATEN SUMENEP

Lika Hanifa[1§], Luh Putu Ida Harini[2], G.K. Gandhiadi[3]

[1]Program Studi Matematika, Fakultas MIPA – Universitas Udayana [Email: likahanifa425@gmail.com]
[2]Program Studi Matematika, Fakultas MIPA – Universitas Udayana [Email: ballidah@unud.ac.id]
[3]Program Studi Matematika, Fakultas MIPA – Universitas Udayana [Email: gandhiadi@unud.ac.id]
[§]*Corresponding Author*

ABSTRACT

A graph is a diagram that contains specific information. One concept in graphs that can solve real-life problems is the concept of trees, which consists of various types of trees used to solve problems in life, such as finding the minimum path using the Dijkstra algorithm and the use of minimum spanning trees using the Sollin algorithm. This research produced the minimum path using Dijkstra's Algorithm and the minimum spanning tree using Sollin's Algorithm, which were applied to historical tourist routes in Sumenep Regency.

Keywords: *Dijkstra's Algorithm, Algorithm Sollin's, Minimum Spanning Trees*

1. PENDAHULUAN

Kabupaten Sumenep adalah salah satu Kabupaten yang terletak di pulau Madura, Jawa Timur. Menurut (Kabupaten Sumenep Dalam Angka, 2017) Kabupaten Sumenep memiliki luas sekitar 2.093 km² yang terdiri dari 27 Kecamatan, yakni 19 Kecamatan di daratan dan 8 Kecamatan yang meliputi pulau-pulau kecil. Selain itu Kabupaten Sumenep juga memiliki tempat wisata seperti wisata religi, sejarah, budaya, dan arsitektur yang memiliki daya tarik tersendiri untuk dikunjungi. Wisata religi, Sejarah, Budaya dan Arsitektur menurut Qurnia Tour and Travel yaitu: Asta Tinggi, Makam Pangeran Panembahan Joharsari, Asta Pangeran Lor Wetan (Asta Karang Sabu), Asta Katandur, Museum Keraton Sumenep, Masjid Jamik Sumenep, Kota Tua Kalianget, dan Benteng Kalimo'ok.

Namun, konektivitas antar destinasi wisata tersebut belum dioptimalkan sehingga perjalanan wisatawan kurang efisien. Perjalanan menuju ke lokasi wisata tersebut membutuhkan transportasi dan perencanaan perjalanan yang tepat. Permasalahan tersebut dapat diselesaikan dengan perancangan jalur yang lebih efisien seperti pemodelan graph dalam matematika diskret seperti lintasan terpendek *(minimum path)* dan *minimum spanning tree*.

Algoritme Dijkstra adalah metode *greedy* yang digunakan untuk menemukan jalur terpendek dari satu *vertex* awal ke semua *vertex* lain dalam graph berbobot non-negatif. Pada penelitian ini, algoritme yang diterapkan untuk mengoptimalkan rute wisata dari Masjid Jami' Sumenep ke destinasi lain di Kabupaten Sumenep. Berbeda dengan Algoritme Sollin yang menghasilkan *minimum spanning tree* (MST) untuk menghubungkan semua destinasi, algoritme dikstra fokus pada jalur terpendek ke destinasi spesifik.

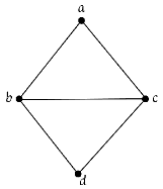
Berdasarkan uraian tersebut, penulis menerapkan Algoritme Dijkstra dalam menentukan jalur terpendek dan Algoritme Sollin untuk mencari *minimum spanning tree* terhadap perjalanan wisata religi, sejarah, budaya dan arsitektur di Kabupaten Sumenep

Adapun variabel yang digunakan pada penelitian ini adalah jalur antar tempat wisata sebagai *edge*, objek wisata sebagai *vertex*, serta jarak antar tempat wisata sebagai bobot.

Dalam matematika, graph sering merujuk pada grafik fungsi atau relasi. Namun, penelitian ini membahas graph dalam teori graph, yaitu struktur yang terdiri dari simpul (*vertex*) dan sisi (*edge*) yang menghubungkan *vertex* tersebut.

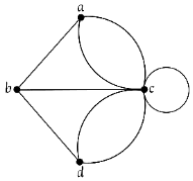
1.1. Graph Berdasarkan Ada Tidaknya Gelang atau *edge* Ganda

Menurut (Kusmira & Taufiqurrachman, 2017) jenis graph ini dibagi menjadi dua yaitu graph sederhana *(Simple Graph)* dan graph tak sederhana *(Non-simple Graph).* Graph sederhana adalah graph yang tidak memiliki gelang dan *edge* ganda. Dapat dilihat pada gambar 1 bahwa tidak ditemukan *edge* yang membentuk *loop* dan *edge* yang terdaftar dua kali yaitu $E(G) = \{ab, ac, bc, bd, cd\}$.



Gambar 1. Graph Sederhana *(Simple Graph)*

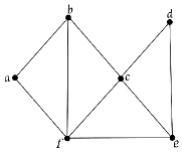
Graph tak sederhana adalah suatu graph yang memiliki *edge* ganda atau gelang. Graph tak sederhana dibagi menjadi dua, yaitu graph ganda dan graph semu. Graph ganda merupakan graph yang memiliki *edge* ganda. Sesuai pada gambar 2 ditemukan *edge* ganda yaitu *ac* dan *cd* Sedangkan graph semu adalah graph yang memiliki gelang. *Edge* yang terdapat pada graph semu dapat terhubung dengan dirinya sendiri seperti pada *edge cc* yang membentuk sebuah *loop.*



Gambar 2. Graph Tak Sederhana *(Non-simple Graph)*

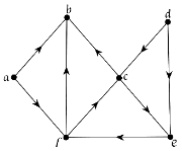
1.2. Graph Berdasarkan Orientasi Arah *Edge* Suatu Graph.

Menurut (Sam & Yuliani, 2016) graph terbagi menjadi graph tak berarah *(Non-directed graph)* dan graph berarah *(directed graph).* Graph dengan *edge* yang tidak memiliki orientasi arah disebut graph tak berarah. Pada graph tak berarah urutan pasangan *vertex* tidak diperhatikan, seperti pada gambar 3 diperoleh $(a, b) = (b, c)$.



Gambar 3. Graph Tak Berarah *(Non-directed Graph)*

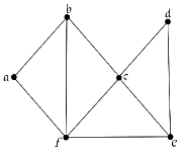
Graph berarah merupakan suatu graph dengan *edge* yang memiliki orientasi arah. Pada graph berarah, urutan pasangan sangat diperhatikan. Berdasarkan gambar 4 $(a, b) \neq (b, a)$ karena a mengarak ke b dan b tidak mengarak ke a, begitupun dengan $(a, f) \neq (f, a), (b, f) \neq (f, b),\ (b, c) \neq (c, b), (c, f) \neq (f, c),\ (c, d) \neq (d, c),\ (c, e) \neq (e, c), (d, e) \neq (e, d), dan\ (f, e) \neq (e, f)$.



Gambar 4. Graph Berarah *(Directed Graph)*

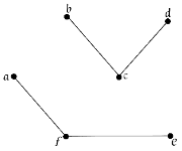
1.3. Graph Berdasarkan Ada Tidaknya Keterhubungan

Menurut Sam & Yuliani (2016) jenis graph berdasarkan ada tidaknya keterhubungan terbagi menjadi Graph terhubung *(Connected Graph)* dan Graph tak terhubung *(Non-connected Graph).* Graph dapat dikatakan terhubung apabila semua *vertex* terhubung oleh *edge* terhadap *vertex* lainnya. Pada gambar 5 semua *vertex* saling terhubung oleh *edge.*



Gambar 5. Graph Terhubung *(Connected Graph)*

Graph dikatakan tak terhubung apabila terdapat satu atau lebih *vertex* yang tidak terhubung oleh *edge* terhadap *vertex* lainnya. Pada gambar 6 $V(G) = \{a, f, e\}$ tidak terhubung pada $V(G) = \{b, c, d\}$.



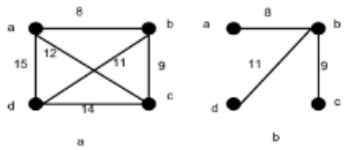
Gambar 6. Graph Terhubung *(Connected Graph)*

Pohon merupakan suatu graph yang terhubung, serta termasuk graph tak berarah yang tidak memiliki sirkuit (Didiharyono & Soraya, 2018). Pohon memiliki struktur data

non-linier dengan hubungan yang bersifat hirarki.

Pohon merentang adalah suatu subgraph dari graph asal yang melibatkan seluruh *vertex* di dalamnya. Subgraph ini berbentuk pohon karena memiliki sifat terhubung dan tidak mengandung siklus. (Afrianto & Jamilah, 2012).

Subgraph dengan jalur minimal yang menghubungkan semua *vertex* pada graph pohon merupakan pohon merentang minimum *(minimum spanning tree)* yang terdapat pada suatu graph pohon. Pohon merentang dengan bobot terkecil pada suatu graph pohon merentang dapat dikatakan sebagai pohon merentang minimum *(minimum spanning tree)* (Afrianto & Jamilah, 2012) terlihat pada gambar 7 $E(G) = \{ab, bc, bd\}$ merupakan *minimum spanning tree* dari $E(G) = \{ab, ad, ac, bc, bd, cd\}$.



Gambar 7. *Skewed Binary Tree*

Algoritme Dijkstra merupakan suatu algoritme yang menerapkan graph berarah dan berbobot, dengan jarak antar *vertex* diatur sebagai bobot. Algoritme Dijkstra akan mencari lintasan yang paling minimum antara *vertex* yang satu dengan *vertex* yang lainnya serta beroperasi secara menyeluruh terhadap semua alternatif fungsi yang ada, sehingga menghasilkan lintasan terpendek dari semua *vertex* yang dapat digunakan untuk menghitung total biaya atau total jarak dari lintasan terpendek yang telah terbentuk (Yusuf at all, 2017).

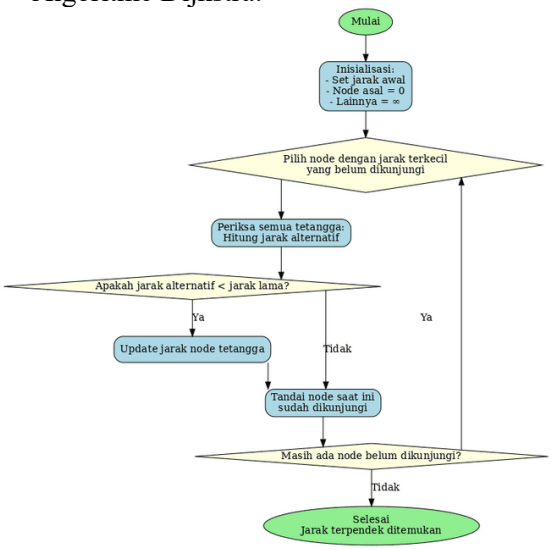
Algoritme Sollin merupakan gabungan Algoritme Kruskal dan Prim, konsep dari Algoritme Sollin adalah memilih wakil *edge* dari masing-masing antar *vertex* dengan nilai bobot terkecil untuk menemukan solusi dari *minimum spanning tree (MST)* dan pada Algoritme Sollin juga dapat ditemukan kemungkinan bahwa *edge* yang terpilih akan sama atau duplikat, sehingga akan ada proses penghapusan duplikat Wamiliana (2014).

2. METODE PENELITIAN

Penelitian ini menggunakan metode Algoritme Sollin dalam mencari *minimum spanning tree*. Penelitian ini dilakukan pada Agustus 2025. Data yang diambil adalah jarak objek wisata religi, sejarah, budaya, dan arsitektur berdasarkan Qurnia Tour and Travel.

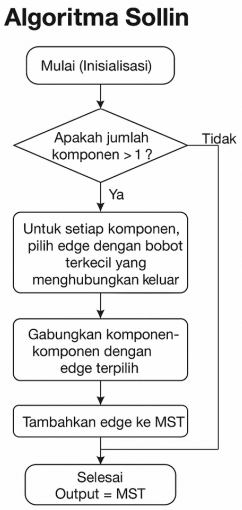
Langkah-langkah analisis data yang dilakukan pada penelitian ini yaitu:

1. Menentukan objek wisata religi, sejarah, budaya dan arsitektur menurut Qurnia Tour n Travel yang akan digunakan sebagai *vertex.*
2. Menentukan simbol pada *vertex.*
3. Mencari rute dan jarak antar lokasi menggunakan *google maps* yang akan digunakan sebagai *edge* dan bobot *edge*.
4. Merepresentasikan data yang diperoleh dalam bentuk tabel.
5. Melakukan penghitungan menggunakan Algoritme Dijkstra.



Gambar 8. Flowchart Alagoritma Dijkstra

6. Melakukan penghitungan menggunakan Algoritme Sollin.



Gambar 9. Flowchart Alagoritma Sollin

3. HASIL DAN PEMBAHASAN

3.1. Pengambilan dan Menentukan Simbol Model Pengukuran

Menentukan objek wisata religi, sejarah, budaya dan arsitektur menurut Qurnia Tour n Travel yang mewakili sebagai *vertex*, menentukan rute dan jarak antar objek wisata menggunakan *Google Maps* yang mewakili sebagai *edge* dan bobot. setiap *vertex* yang disimbolkan sebagai berikut:

1. Masjid Jamik Sumenep = a
2. Museum Keraton Sumenep = b
3. Asta Tinggi = c
4. Asta Katandur = d
5. Makam Pangeran Panembahan Joharsari = e
6. Asta Pangeran Lor Wetan (Asta Karang Sabu) = f
7. Kota Tua Kalianget = g
8. Benteng Kalimo'ok = h

3.2. Representasi Data

Representasi Data jarak antar destinasi wisata direpresentasikan dalam tabel matriks dengan satuan kilometer (km)

Tabel 1. Representasi Tabel Jarak

vertex	a	b	c	d	e	f	g	h
a	0	0.6	2.7	2.5	14	0.55	11	7.1
b	0.6	0	2.7	2.5	3	0.9	10	6.8
c	2.7	2.7	0	4.8	16	2.2	13	9.4
d	2.5	2.5	4.8	0	16	2.7	11	7.7
e	14	3	16	16	0	14	22	19
f	0.55	0.9	2.2	2.7	14	0	11	7.2
g	11	10	13	11	22	11	0	3.7
h	7.1	6.8	9.4	7.7	19	7.2	3.7	0

Matriks ini menunjukkan graph tak berarah yang terhubung, dimana jarak antar *vertex* (bobot *edge*) simetris (misalnya, a-b = b-a = 0,6 km). Data ini digunakan sebagai dasar penerapan Algoritme Sollin.

3.3. Penerapan Algoritme Dijkstra

a. Diasumsikan Tidak ada gelang (misalnya, a ke a) atau *edge* ganda antar *vertex*.
b. Diasumsikan sebagai graph tak berarah yaitu Jarak simetris (misalnya, a-f = f-a = 0,55 km).
c. Diasumsiakan *vertex* awal adalah *a* (Masjid Jami' Sumenep).
d. Diasumsikan semua bobot *edge* positif, sesuai syarat Algoritme Dijkstra.
e. Diasumsikan hanya untuk menemukan jalur terpendek dengan jarak, dari *a* ke setiap *vertex* lain.

1. Inisialisasi

Tabel 2. Tabel Inisialisasi

vertex	Jarak dari a	Status	Predecessor
a	0	Belum	None
b	∞	Belum	None
c	∞	Belum	None
d	∞	Belum	None
e	∞	Belum	None
f	∞	Belum	None
g	∞	Belum	None
h	∞	Belum	None

2. Iterasi 1: pilih *vertex* a (jarak = 0)

Vertex dipilih: a (Masjid Jami' Sumenep), karena memiliki jarak terkecil (0 km). Kemudian perbarui jarak sesuai dengan *edge* yang terhubung dengan *vertex* terpilih.

3. Iterasi 2: Pilih *vertex* f (Jarak = 0,55)

Vertex Dipilih: *f* (Asta Lor Wetan), jarak terkecil di antara *vertex* belum dikunjungi (0,55 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

4. Iterasi 3: Pilih *vertex* b (Jarak = 0,6)

Vertex Dipilih: *b* (Museum Keraton), jarak terkecil (0,6 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih. Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

5. Iterasi 4: Pilih *vertex* d (Jarak = 2,5)

Vertex Dipilih: *d* (Asta Katandur), jarak terkecil (2,5 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

6. Iterasi 5: Pilih *vertex* c (Jarak = 2,7)

Vertex Dipilih: *c* (Asta Tinggi), jarak terkecil (2,7 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

7. Iterasi 6: Pilih *vertex* h (Jarak = 7,1)

Vertex Dipilih: *h* (Benteng Kalimo'ok), jarak terkecil (7,1 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih. Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

8. Iterasi 7: Pilih *vertex* g (Jarak = 10,6)

Vertex Dipilih: *g* (Kota Tua Kalianget), jarak terkecil (10,6 km). Kemudian perbarui jarak sesuai denga *edge* yang terhubung dengan *vertex* terpilih.

9. Iterasi 8: Pilih *vertex* e (Jarak = 3,6).

Vertex Dipilih: *e* (Makam Pangeran), jarak terkecil (3.6 km) dan tidak ada pembaruan.

Tabel 3. Hasil Iterasi 8

vertex	Jarak dari a	Status	Predecessor
a	0	Dikunjungi	None
b	0,6	Dikunjungi	a
c	2,7	Dikunjungi	a
d	2,5	Dikunjungi	a
e	3,6	Dikunjungi	b
f	0,55	Dikunjungi	a
g	10,6	Dikunjungi	b
h	7,1	Dikunjungi	a

Semua *vertex* telah dikunjungi, sehingga algoritme selesai.

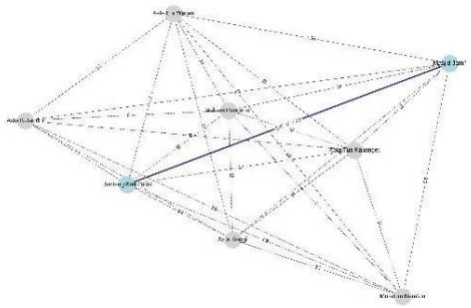
10. Hasil Akhir

Berdasarkan tabel akhir, jarak terpendek dan jalur dari Masjid Jami' Sumenep (*a*) ke setiap *vertex* adalah:

Tabel 4. Hasil lintasan Terpendek

Destinasi	Jarak (km)	Jalur
Masjid Jami' (a)	0	Masjid Jami'
Museum Keraton (b)	0.6	Masjid Jami' → Museum Keraton
Asta Tinggi (c)	2.7	Masjid Jami' → Asta Tinggi
Asta Katandur (d)	2.5	Masjid Jami' → Asta Katandur
Makam Pangeran (e)	3.6	Masjid Jami' → Museum Keraton → Makam Pangeran
Asta Lor Wetan (f)	0.55	Masjid Jami' → Asta Lor Wetan
Kota Tua Kalianget (g)	10.6	Masjid Jami' → Museum Keraton → Kota Tua Kalianget
Benteng Kalimo'ok (h)	7.1	Masjid Jami' → Benteng Kalimo'ok

Misalkan akan dicari rute dari masjid jami' Sumenep menggunakan Algoritme Dijkstra dengan asumsi kecepatan yang digunakan adalah 40 km/jam maka rute terpendek yang dihasilkan adalah 7.1 km dan dapat ditempuh dalam waktu ≈ 10,65 menit.



Gambar 10. Lintasan Terpendek dari a-h

3.4. Penerapan Algoritme Sollin

1. Data disingkat, hanya *edge* utama dengan bobot kecil yang berpotensi masuk *minimum spanning tree.*

Tabel 5. Urutan Data Berdasarkan Bobot Yang Terkecil

Dari	Ke	Bobot	Dari	Ke	Bobot
a	f	0.55	D	h	7.7
a	b	0.6	C	h	9.4
b	f	0.9	B	g	10
c	f	2.2	A	g	11
b	d	2.5	F	g	11
a	d	2.5	D	g	11
a	c	2.7	C	g	13
b	c	2.7	A	e	14.3
d	f	2.7	E	f	14.3
b	e	3	C	e	16.2
g	h	3.7	D	e	16.3
b	h	6.8	E	h	18.9
a	h	7.1	E	g	22
f	h	7.2			

2. Iterasi 1 (setiap *vertex* memilih *edge* terkecil keluar):

Tabel 6. Iterasi 1 Algoritme Sollin

Vertex	*Edge* Minimum	Bobot (km)	Gabungan Awal
a	a-f	0.6	{a, f}
b	b-a	0.6	{b, a}
c	c-f	2.2	{c, f}
d	d-a	2.5	{d, a}
e	e-b	3.0	{e, b}
f	f-a	0.6	{f, a}
g	g-h	3.7	{g, h}
h	h-g	3.7	{h, g}

Penggabungan Komponen dengan *Edge* f-a dan h-g adalah duplikat dari a-f dan g-h, sehingga diabaikan untuk menghindari redundansi.
Hasil Iterasi 1:
Komponen 1: {a, f, b, c, d,e}
Komponen 2: {g, h}
Edge minimum spanning tree sementara: a–f (0.55), a–b (0.6), c–f (2.2), d–a (2.5), e–b (3), g–h (3.7)

3. Iterasi 2 (tiap komponen pilih *edge* keluar terkecil):

Tabel 7. Iterasi 2 Algoritme Sollin

Komponen 1	Komponen 2	Bobot (km)	Penghubung
{a,b,c,d,f}	{g,h}	6.8	b–h
{a,b,c,d,f}	{g,h}	7.1	a–h
{a,b,c,d,f}	{g,h}	7.2	f–h
{a,b,c,d,f}	{g,h}	7.7	d–h
{a,b,c,d,f}	{g,h}	9.4	c–h
{a,b,c,d,f}	{g,h}	10	b–g
{a,b,c,d,f}	{g,h}	11	a–g
{a,b,c,d,f}	{g,h}	11	d–g
{a,b,c,d,f}	{g,h}	11	f–g
{a,b,c,d,f}	{g,h}	13	c–g
{a,b,c,d,f}	{g,h}	19	e–h
{a,b,c,d,f}	{g,h}	22	e–g

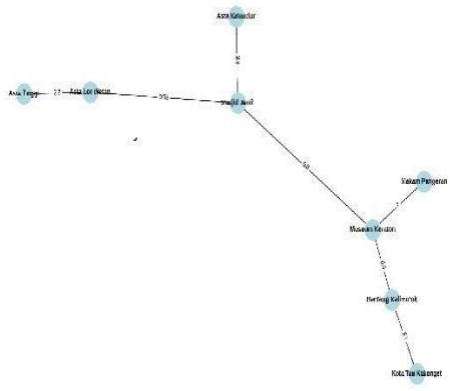
Hasil Iterasi 2:

Komponen 1: {a,b,c,d,e,f,g,h}

Tambahan *edge minimum spanning tree*: b–h (6.8). Semua *vertex* sudah tergabung sehingga algoritme selesai.

4. Hasil *minimum spanning tree* (Dengan Algoritme Sollin)

Edge yang terpilih:
Masjid Jami' - Asta Lor Wetan : 0.55 km.
Museum Keraton - Masjid Jami' : 0.6 km.
Asta Tinggi - Asta Lor Wetan : 2.2 km.
Asta Katandur - Masjid Jami' : 2.5 km.
Makam Pangeran - Museum Keraton : 3 km.
Kota Tua Kalianget - Benteng Kalimo'ok : 3.7 km.
Museum Keraton - Benteng Kalimo'ok : 6.8 km.
Total bobot *minimum spanning tree* = 19.35 km.



Gambar 11. *Minimum Spanning Tree*

4. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa Algoritme Dijkstra dapat digunakan untuk mencari lintasan terpendek dan Algoritme Sollin dapat digunakan untuk mencari *minimum spanning tree* pada jaringan perjalanan wisata religi, sejarah, budaya, dan arsitektur di Kabupaten Sumenep. Hasil *minimum spanning tree* memberikan jalur optimal yang dapat menjadi dasar pengembangan paket wisata. Berdasarkan dari hasil penelitian yang diperoleh penulis menyarankan untuk Penelitian selanjutnya dapat memperluas analisis dengan mempertimbangkan faktor waktu tempuh, kondisi jalan, serta preferensi wisatawan serta menggunakan konsep algoritme lain dalam menentukan lintasan terpendek dan *minimum spanning tree*.

DAFTAR PUSTAKA

Afrianto, I., Jamilah, E.W. 2012. "Penyelesaian Masalah Minimum Spanning Tree (MST) Menggunakan Ant Colony System (ACS)." *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)* I (2): 35–40.

Didiharyono, Soraya, S. 2018. "Penerapan Algoritma Greedy dalam Menentukan Minimum Spanning Trees Pada Optimisasi Jaringan Listrik Jala." *Jurnal Varian* 1 (2): 1–10.

Kusmira, M. dan Taufiqurrachman. 2017. "Pemanfaatan Aplikasi Graf pada Pembuatan Jalur Angkot 05 Tasikmalaya." *Seminar Nasional sains dan teknologi jurnal UMJ*.

Sam, M, Yuliani. 2016. "Penerapan Algoritma Prim untuk Membangun Pohon Merentang Minimum (Minimum Spanning Tree) dalam Pengoptimalan Jaringan Transmisi Nasional Provinsi Sulawesi Selatan." *Jurnal Dinamika* 7 (1): 50–61.

Wamiliana.dkk. 2014. "Perbandingan Kompleksitas Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin untuk Menyelesaikan Masalah Minimum Spanning Tree." *Jurnal komputasi* 2 (1): 60–74.

Yusuf, M. S. dkk. 2017. "Implementasi Algoritma Dijkstra Dalam Menemukan Jarak Terdekat DariLokasi Pengguna Ke Tanaman Yang Di Tuju Berbasis Android (Studi Kasus di Kebun Raya Purwodadi)." *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* 1 (12): 1779–87.