Konstruksi Extreme Point Deterministic Algorithm Melalui Algoritma Kruskal dan Algoritma Prim pada Masalah Multi-Criteria Minimum Spanning Tree

Evawati Alisah*, Moh. Miftakhul Ulum*
* Jurusan Matematika, UIN Maulana Malik Ibrahim Malang

Email: eva.alisah@gmail.com, ululum@gmail.com

Info Artikel

Riwayat Artikel:

Diterima: 1 Oktober 2018
Direvisi: 1 November 2018
Diterbitkan: 1 Desember 2018

Kata Kunci:

Extreme point deterministic
algoritma Kruskal
algritma Prim
Multi-criteria spanning tree

ABSTRAK

Kajian MCMST merupakan pengembangan dari masalah optimasi MST dengan memuat dua kriteria atau lebih. Salah satu algoritma yang mampu untuk menyelesaikan masalah MCMST adalah EPDA. EPDA memiliki tiga tahapan. Sebagai fondasi awal, pada tahap pertama dibangun dari Algoritma Kruskal atau Algoritma Prim dengan memperhatikan kriteria yang bersesuaian satu per satu. Kemudian pada tahap kedua dan ketiga dilakukan proses mutasi sampai akhirnya didapatkan pohon merentang baru yang menjadi solusi efisien atau Pareto Front. Dengan perbedaan karakteristik yang dimiliki Algoritma Kruskal dan Algoritma Prim, penulis ingin menjelaskan perbandingan antara EPDA yang dibangun dari Algoritma Kruskal dan EPDA yang dibangun dari Algoritma Prim.

Secara umum, baik EPDA dengan Algoritma Kruskal maupun EPDA dengan Algoritma Prim menghasilkan solusi yang sama. Adapun perbedaan yang dihasilkan terdapat ada indeks yang digunakan. Kemudian untuk memperkecil banyaknya kemungkinan solusi yang diberikan, maka pada saat pemilihan sisi baik untuk Algoritma Kruskal maupun Algoritma Prim tidak hanya memperhatikan kriteria yang dikerjakan, namun sekaligus memperhatikan pertimbangan solusi yang termuat dalam tabel Edge List. Oleh karena itu, dengan diperoleh banyaknya kemungkinan solusi yang lebih sedikit, maka proses penyelesaian yang dilakukan menjadi lebih singkat.

Korespondensi:

Evawati Alisah,
Jurusan Matematika,
UIN Maulana Malik Ibrahim Malang,
Jl. Gajayana No. 50 Malang, Jawa Timur, Indonesia 65144
eva.alisah@gmail.com

1. PENDAHULUAN

Kajian Minimum Spanning Tree (MST) bertujuan untuk menyelesaikan masalah bobot minimum suatu graf. Masalah ini dapat diselesaikan secara eksak menggunakan beberapa algoritma berikut, yakni Algoritma Kruskal, Algoritma Prim, dan Algoritma Sollin. Algoritma tersebut telah populer untuk menyelesaikan masalah optimasi MST dalam kehidupan nyata, seperti konstruksi jalan dari beberapa lokasi berdasarkan jarak tempuh minimal, penentuan lintasan dengan biaya termurah, dan optimasi jaringan kabel listrik.

Sebagai pengembangan dari masalah optimasi MST, masalah Multi-Criteria Minimum Spanning Tree (MCMST) memuat kriteria lebih dari satu. Menurut Vianna [1] kriteria-kriteria yang termuat dalam masalah MCMST sering didapati saling berlawanan. Tujuan MCMST adalah untuk mendapatkan solusi yang memuat suatu himpunan solusi optimal yang diyakini tidak ada solusi lain yang lebih optimal dari solusi tersebut. Himpunan solusi tersebut dikenal sebagai Pareto Front (PF) atau himpunan solusi efisien [2]. Salah satu

algoritma yang mampu untuk menyelesaikan masalah MCMST adalah Extreme Point Deterministic Algorithm (EPDA).

EPDA memiliki tiga tahap penyelesaian masalah MCMST [2]. Sebagai fondasi awal algoritma ini, tahap pertama dibangun dari Algoritma Kruskal atau Algoritma Prim dengan memperhatikan kriteria yang bersesuaian satu per satu. Kemudian pada tahap kedua dan ketiga dilakukan proses mutasi sampai akhirnya diperoleh pohon merentang baru yang menjadi solusi efisien atau PF. Dengan perbedaan karakteristik yang dimiliki Algoritma Kruskal dan Algoritma Prim, penulis ingin menjelaskan perbandngan antara EPDA yang dibangun dari Algoritma Kruskal dan EPDA yang dibangun dari Algoritma Prim.

2. KAJIAN TEORI

2.1. Multi-Criteria Minimum Spanning Tree

Masalah Multi-Criteria Minimum Spanning Tree (MCMST) tidak semudah mengubah MST dengan satu kriteria ke multi kriteria. Pada umumnya, kriteria yang termuat dalam masalah MCMST saling bertentangan. Sehingga solusi optimal dari masalah tersebut tidak mudah untuk ditentukan [3].

Misalkan $x = (x_1, x_2, \dots, x_m)$ didefinisikan sebagai berikut:

$$x_i = \begin{cases} 1, & \text{jika sisi } e_1 \text{ dipilih} \\ 0, & \text{untuk yang lain.} \end{cases}$$

Kemudian pohon merentang dari graf G dapat dinyatakan oleh vektor x. Misalkan X adalah himpunan dari setiap vektor yang bersesuaian terhadap pohon merentang dalam graf G, maka masalah MCMST dapat diformulasikan sebagai berikut:

$$\begin{aligned} \min z_1(x) &= \sum_{i=1}^{m} w_{1i}x_i \\ \min z_2(x) &= \sum_{i=1}^{m} w_{2i}x_i \\ &\vdots \\ \min z_p(x) &= \sum_{i=1}^{m} w_{pi}x_i \; ; x \in X \end{aligned}$$

dengan $z_i(x)$ adalah objek ke-i untuk diminimalkan dalam masalah MCMST [3].

MCMST bertujuan untuk mendapatkan solusi yang memuat suatu himpunan solusi optimal yang diyakini bahwa tidak ada solusi lain yang lebih optimal dari solusi yang diperoleh. Kumpulan tersebut dikenal sebagai Pareto Front (PF) atau kumpulan solusi efisien [2].

2.2. Macam-Macam Solusi MCMST

Keshavarz [4] memberikan beberapa definisi terkait macam-macam solusi pada masalah optimasi multi tujuan. Misalkan S adalah himpunan dari solusi yang mungkin terjadi atau himpunan kemungkinan dalam ruang keputusan dan $Z = \{(cx, ty) | (x, y) \in S\}$ tujuan maka:

Misalkan $(x, y), (x', y') \in S$. Jika $cx \leq cx', ty \leq ty'$, dan $(cx, ty) \neq (cx', ty')$ maka (x, y) mendominasi (x', y') dalam ruang keputusan dan (cx, ty) mendominasi (cx', ty') dalam ruang tujuan.

Solusi $(x^*, y^*) \in S$ disebut sebagai solusi efisien atau PF, jika tidak ada $(x, y) \in S$ sedemikian sehingga (x, y) mendominasi (x^*, y^*). Jika (x^*, y^*) adalah solusi efisien, maka vektor (cx^*, ty^*) dikatakan sebagai titik non-dominated dalam ruang tujuan. Himpunan solusi efisien dinotasikan sebagai S_E dan bayangan dari S_E di Z disebut sebagai himpunan non-dominated Z_N.

Solusi efisien $(x^*, y^*) \in S_E$ adalah solusi efisien yang supported jika solusi tersebut merupakan solusi optimal dengan menjumlahkan bobotnya yang memenuhi kondisi berikut

$$\min\{\lambda_1 cx + \lambda_2 ty | (x, y) \in S\}$$

untuk $\lambda_1 > 0$ dan $\lambda_2 > 0$. Jika (x^*, y^*) adalah solusi efisien yang supported, maka (cx^*, ty^*) dinamakan titik supported non-dominated. Adapun titik supported non-dominated terletak pada batas tepi berbentuk lengkungan cembung.

Solusi efisien $(x^*, y^*) \in S_E$ adalah solusi efisien yang non-supported jika tidak terdapat nilai positif dari λ_1 dan λ_2 sedemikian sehingga (x^*, y^*) memenuhi kondisi pada poin c.

2.3. Extreme Point Deterministic Algorithm

Terdapat tiga tahapan yang dimiliki oleh EPDA dalam menyelesaikan masalah MCMST. Berikut adalah tahapan-tahapan yang perlu dilakukan [2].

Tahap 1

Membuat daftar semua sisi dari graf yang bersesuaian ke dalam satu tabel dengan memperhatikan kriteria p. Tabel tersebut diberi nama Edge List$[i]$, dengan i adalah indeks dari banyaknya tabel yang digunakan. Tabel Edge List diurutkan berdasarkan kriteria yang dikerjakan. Kemudian MST sementara (MSTs) ditemukan menggunakan Algoritma Kruskal atau Algoritma Prim dengan memperhatikan kriteria satu per satu.

Dengan menggunakan Boolean flag, untuk setiap sisi yang tidak dipilih bernilai 0 dan yang terpilih bernilai 1. Keduanya termuat dalam Edge List. Adapun kumpulan sisi yang terpilih didefinisikan sebagai In Tree.

Tahap 2

Himpunan pertama dari pohon merentang sementara atau STs dibuat dengan mengganti hanya satu sisi dari MSTs. STs yang baru ini adalah tetangga dari MSTs yang disebut sebagai sisi karakteristik dengan ketentuan bahwa ia tidak akan tergantikan pada langkah selanjutnya agar tidak terjadi duplikasi. Semua tetangga dari $\text{MST}_i, i = 1, \dots, p$ yang non-dominated dihitung dengan aturan berikut. Untuk setiap $(u, v) \in \text{MST}_i$ yang dilepas, algoritma ini memindai sisi (r, s) dalam Edge List$[i]$ sebanyak mungkin yang akan menggantikan (u, v) dengan ketentuan bahwa (r, s) memenuhi tiga kondisi berikut:

1. $(r, s) \notin In\ Tree$.
2. Menambahkan (r, s) tidak mengakibatkan adanya cycle.
3. $C_j(r, s) < C_j(u, v)$ untuk setidaknya satu j, dengan $j = 1, \dots, p$ dan $j \neq i$.

Jika (r, s) memenuhi kondisi di atas, maka (r, s) ditandai sebagai sisi karakteristik dengan mengatu flag karakteristiknya sama dengan indeks dari sisi (u, v). Kemudian Total Costs (TC) dari STs yang baru dihitung dengan menggunakan relasi berikut:

$$\forall j, TC_j(\text{STs baru}) = TC_j(\text{MST}_i) - C_j(u, v) + C_j(r, s)$$

Kemudian TC dari STs baru dibandingkan dengan TC dari STs non-dominated pada Approximate Pareto Set (APS) atau hampiran dari PF. Jika STs baru tidak didominasi, maka ia ditambahkan ke APS.

Tahap 3

Setiap pohon merentang dalam APS dipilih untuk membuat STs yang baru, seperti pada tahap 2, kecuali bahwa untuk $j \neq i$ pada kondisi ketiga ditiadakan dalam tahap ini. Untuk masing-masing STs yang dipilih, sisi yang dimutasi adalah sisi yang bukan karakteristik. Langkah tersebut dilakukan sampai diperoleh STs yang valid dan masuk ke dalam APS sebagai solusi efisien atau PF.

3. HASIL DAN PEMBAHASAN

Data yang digunakan pada penelitian ini adalah data beberapa ruas jalan di sekitar kampus UIN Maulana Malik Ibrahim Malang yang diakses secara online melalui aplikasi Waze pada tanggal 26 November 2017.

Tabel 1. Beberapa Jalan di Sekitar UIN Maulana Malik Ibrahim Malang

Sisi	Nama Jalan	Jarak (m)	Waktu (menit)
AB	Jl. Tlogo Indah	921	4
BC	Jl. Joyo Utomo	356	2
CD	Jl. Mertojoyo	706	3
AD	Jl. MT. Hariono 1	314	1
CE	Jl. Joyo Tambaksari	737	3
CI	Jl. Sunan Kalijaga – Jl. Bend. Sigura-gura	2500	10
EI	Jl. Sumbersari	1190	4
DF	Jl. MT. Hariono 2	644	3
EF	Jl. Gajayana	474	3
FG	Jl. MT. Hariono 3	937	3
GH	Jl. Soekarno Hatta	749	3
GJ	Jl. DI. Panjaitan 1	1240	3
JK	Jl. Bogor	305	4
JL	Jl. DI. Panjaitan 2	662	2
LK	Jl. Bandung	542	2
IK	Jl. Veteran	1110	4
Jumlah		13387	54

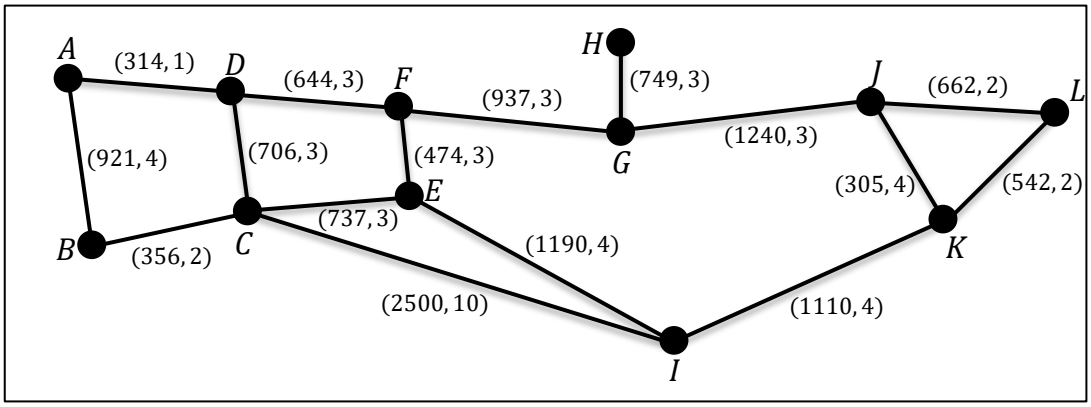
Graf pada Gambar 1 memiliki 12 titik dan 16 sisi dengan jumlah bobotnya sebesar $TC = (13387, 54)$.

3.1. EPDA dengan Algoritma Kruskal

Tahap 1

Pada tahap ini akan ditentukan dua MST yang dihasilkan melalui Algoritma Kruskal dengan memperhatikan kriteria satu per satu secara bergantian. Kemudian MST tersebut dimasukkan ke dalam tabel Edge List.

MST_1 dengan $TC = (7327, 32)$diperoleh melalui Algoritma Kruskal dengan memperhatikan kriteria C_1. sedangkan MST_2 dengan $TC = (7734, 29)$ diperoleh melalui Algoritma Kruskal dengan memperhatikan kriteria C_2.



Gambar 1. Beberapa Jalan di Sekitar UIN Maulana Malik Ibrahim Malang dalam Bentuk Graf

Tabel 2. Edge List[1] dari EPDA dengan Algoritma Kruskal

Indeks	Sisi	C_1	C_2	*In Tree*
1	JK	305	4	1
2	AD	314	1	1
3	BC	356	2	1
4	EF	474	3	1
5	LK	542	2	1
6	DF	644	3	1
7	JL	662	2	0
8	CD	706	3	1
9	CE	737	3	0
10	GH	749	3	1
11	AB	921	4	0
12	FG	937	3	1
13	IK	1110	4	1
14	EI	1190	4	1
15	GJ	1240	3	0
16	CI	2500	10	0

Tabel 3. Edge List[2] dari EPDA dengan Algoritma Kruskal

Indeks	Sisi	C_1	C_2	*In Tree*
1	AD	314	1	1
2	BC	356	2	1
3	LK	542	2	1
4	JL	662	2	1
5	EF	474	3	1
6	DF	644	3	1
7	CD	706	3	1
8	CE	737	3	0
9	GH	749	3	1
10	FG	937	3	1
11	GJ	1240	3	1
12	JK	305	4	0
13	AB	921	4	0
14	IK	1110	4	1
15	EI	1190	4	0
16	CI	2500	10	0

Tahap 2

Pada tahap ini akan dilakukan proses mutasi setiap sisi dari MST_1 dan MST_2 dengan mengganti sisi yang bersesuaian yang memenuhi 3 kondisi pada subbab 2.3.

Mutasi MST_1

- Hapus sisi (1) JK
  - Masukkan sisi (7) JL dan bangun MST_3 dengan $TC = (7684, 30)$.
  - MST_3 adalah solusi efisien dengan karakteristik sisi = 1.
  - Masukkan sisi (15) GJ dan bangun MST_4 dengan $TC = (8262, 31)$.
  - MST_4 adalah solusi dominated karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (1) JK.
- Hapus sisi (13) IK
  - Masukkan sisi (15) GJ dan bangun MST_5 dengan $TC = (7457, 31)$.
  - MST_5 adalah solusi dominated karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (13) IK.
- Hapus sisi (14) EI
  - Masukkan sisi (15) GJ dan bangun MST_6 dengan $TC = (7377, 31)$.
  - MST_6 adalah solusi efisien dengan karakteristik sisi = 14.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (14) EI.
- Adapun sisi (2) AD, (3) BC, (4) EF, (5) LK, (6) DF, (8) CD, (10) GH, dan (12) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.

Mutasi MST_2

- Hapus sisi (3) LK
  - Masukkan sisi (12) JK dan bangun MST_7 dengan $TC = (7497, 31)$.
  - MST_7 adalah solusi dominated karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (3) LK.

- Hapus sisi (4) JL
  - Masukkan sisi (12) JK dan bangun MST_8 dengan $TC = (7377, 31)$.
  - MST_8 adalah solusi dominated karena memiliki TC sama dengan MST_6.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (4) JL.
- Hapus sisi (11) GJ
  - Masukkan sisi (15) EI dan bangun MST_9 dengan $TC = (7684, 30)$.
  - MST_9 adalah solusi dominated karena memiliki TC sama dengan MST_3.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (11) GJ.
- Adapun sisi (1) AD, (2) BC, (5) EF, (6) DF, (7) CD, (9) GH, (10) FG, dan (14) IK jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.

Tahap 3

Untuk sementara MST yang masuk ke dalam APS adalah MST_1, MST_2, MST_3, dan MST_6. Selanjutnya MST tersebut dimutasi kembali dengan meniadakan syarat $j \neq i$ dalam kondisi ketiga kecuali MST_1 dan MST_2. Karena sudah optimal jika kondisi tersebut diberlakukan.

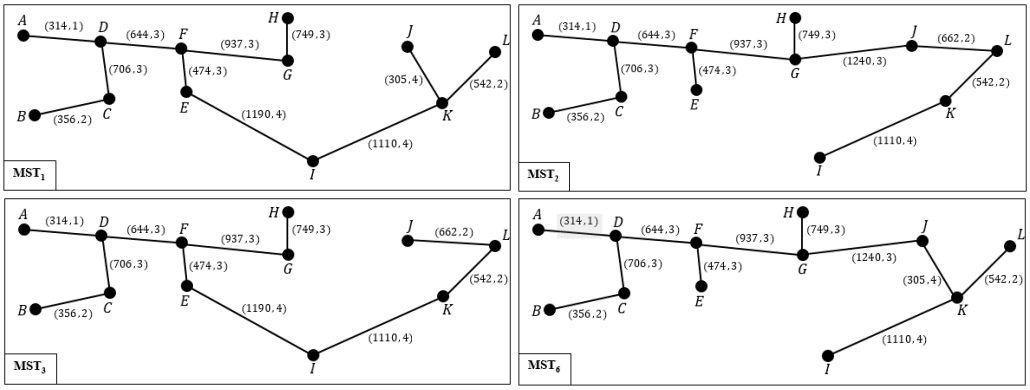
Mutasi MST_3

- Hapus sisi (5) LK
  - Masukkan sisi (1) JK dan bangun MST_{10} dengan $TC = (7447, 32)$.
  - MST_{10} adalah solusi dominated karena memiliki TC lebih besar dari MST_1.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (5) LK.
- Hapus sisi (13) IK
  - Masukkan sisi (1) JK dan bangun MST_{11} dengan $TC = (7814, 29)$.
  - MST_{11} adalah solusi dominated karena memiliki TC lebih besar dari MST_2.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (13) IK.
- Hapus sisi (14) EI
  - Masukkan sisi (15) GJ dan bangun MST_{12} dengan $TC = (7734, 29)$.
  - MST_{12} adalah solusi dominated karena memiliki TC sama dengan MST_2.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (14) EI.
- Adapun sisi (2) AD, (3) BC, (4) EF, (6) DF, (8) CD, (10) GH, dan (12) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.
- Adapun sisi (7) tidak dihapus karena sisi JL merupakan sisi karakteristik.

Mutasi MST_6

- Hapus sisi (1) JK
  - Masukkan sisi (7) JL dan bangun MST_{13} dengan $TC = (7734, 29)$.
  - MST_{13} adalah solusi dominated karena memiliki TC sama dengan MST_2.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (1) JK.
- Adapun sisi (2) AD, (3) BC, (4) EF, (5) LK, (6) DF, (8) CD, (10) GH, (12) FG, dan (13) IK jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.
- Adapun sisi (15) tidak dihapus karena sisi GJ merupakan sisi karakteristik.

Karena tidak ada lagi MST baru yang memenuhi kondisi yang disyaratkan, maka MST yang menjadi solusi efisien atau PF untuk masalah MCMST dari graf pada **Error! Reference source not found.** adalah MST_1 dengan $TC = (7327, 32)$, MST_2 dengan $TC = (7734, 29)$, MST_3 dengan $TC = (7684, 30)$, dan MST_6 dengan $TC = (7377, 31)$.



Gambar 2. Solusi Efisien Menurut EPDA dengan Algortima Kruskal

3.2. EPDA dengan Algoritma Prim

Tahap 1

Pada tahap ini akan ditentukan dua MST yang dihasulkan melalui Algoritma Prim dengan memperhatikan kriteria satu per satu secara bergantian. Kemudian MST tersebut dimasukkan ke dalam tabel Edge List.

MST_1 dengan $TC = (7327, 32)$ diperoleh melalui Algoritma Prim dengan memperhatikan kriteria C_1. Sedangkan MST_2 dengan $TC = (8645, 31)$ diperoleh melalui Algoritma Prim dengan memperhatikan kriteria C_2.

Tabel 4. Edge List[1] *dari EPDA dengan Algoritma Prim*

Indeks	Sisi	C_1	C_2	$In\ Tree$
1	JK	305	4	1
2	AD	314	1	1
3	BC	356	2	1
4	EF	474	3	1
5	LK	542	2	1
6	DF	644	3	1
7	JL	662	2	0
8	CD	706	3	1
9	CE	737	3	0
10	GH	749	3	1
11	AB	921	4	0
12	FG	937	3	1
13	IK	1110	4	1
14	EI	1190	4	1
15	GJ	1240	3	0
16	CI	2500	10	0

Tabel 5. Edge List[2] *dari EPDA dengan Algoritma Prim*

Indeks	Sisi	C_1	C_2	$In\ Tree$
1	AD	314	1	1
2	BC	356	2	1
3	LK	542	2	1
4	JL	662	2	1
5	EF	474	3	0
6	DF	644	3	1
7	CD	706	3	1
8	CE	737	3	1
9	GH	749	3	1
10	FG	937	3	1
11	GJ	1240	3	1
12	JK	305	4	0
13	AB	921	4	0
14	IK	1110	4	0
15	EI	1190	4	1
16	CI	2500	10	0

Tahap 2

Pada tahap ini akan dilakukan proses mutasi setiap sisi dari MST_1 dan MST_2 dengan mengganti sisi yang bersesuaian yang memenuhi 3 kondisi pada subbab 2.3.

Mutasi MST_1

- Hapus sisi (1) JK
  - Masukkan sisi (7) JL dan bangun MST_3 dengan $TC = (7684, 30)$.
  - MST_3 adalah solusi efisien dengan karakteristik sisi = 1.
  - Masukkan sisi (15) GJ dan bangun MST_4 dengan $TC = (8262, 31)$.
  - MST_4 adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (1) JK.
- Hapus sisi (13) IK
  - Masukkan sisi (15) GJ dan bangun MST_5 dengan $TC = (7457, 31)$.
  - MST_5 adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (13) IK.
- Hapus sisi (14) EI
  - Masukkan sisi (15) GJ dan bangun MST_6 dengan $TC = (7377, 31)$.
  - MST_6 adalah solusi efisien dengan karakteristik sisi = 14.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (14) EI.
- Adapun sisi (2) AD, (3) BC, (4) EF, (5) LK, (6) DF, (8) CD, (10) GH, dan (12) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi

Mutasi MST_2

- Hapus sisi (3) LK
  - Masukkan sisi (12) JK dan bangun MST_7 dengan $TC = (7840, 31)$.
  - MST_7 adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (3) LK.
- Hapus sisi (4) JL
  - Masukkan sisi (12) JK dan bangun MST_8 dengan $TC = (7720, 31)$.
  - MST_8 adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (4) JL.
- Hapus sisi (6) DF
  - Masukkan sisi (5) EF dan bangun MST_9 dengan $TC = (7907, 29)$.
  - MST_9 adalah solusi *dominated* karena memiliki TC lebih besar dari MST_{11}.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (6) DF.

- Hapus sisi (7) CD
  - Masukkan sisi (5) EF dan bangun MST_{10} dengan $TC = (7845, 29)$.
  - MST_{10} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_{11}.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (7) CD.
- Hapus sisi (8) CE
  - Masukkan sisi (5) EF dan bangun MST_{11} dengan $TC = (7814, 29)$.
  - MST_{11} adalah solusi efisien dengan karakteristik sisi = 8.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (8) CE.
- Hapus sisi (11) GJ
  - Masukkan sisi (14) IK dan bangun MST_{12} dengan $TC = (7947, 30)$.
  - MST_{12} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_3.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (11) GJ.
- Hapus sisi (15) EI
  - Masukkan sisi (14) IK dan bangun MST_{13} dengan $TC = (7997, 29)$.
  - MST_{13} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_{11}.
  - Tidak ditemukan lagi sisi yang memenuhi kondisi, kembalikan sisi (15) EI.
- Adapun sisi (1) AD, (2) BC, (9) GH, dan (10) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi

Tahap 3

Untuk sementara MST yang masuk ke dalam APS adalah MST_1, MST_3, MST_6, dan MST_{11}. Selanjutnya MST tersebut dimutasi kembali dengan meniadakan syarat $j \neq i$ dalam kondisi ketiga kecuali MST_1. Karena sudah optimal jika kondisi tersebut diberlakukan.

Mutasi MST_3

- Hapus sisi (5) LK
  - Masukkan sisi (1) JK dan bangun MST_{14} dengan $TC = (7447, 32)$.
  - MST_{14} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_1.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (5) LK.
- Hapus sisi (13) IK
  - Masukkan sisi (15) GJ dan bangun MST_{15} dengan $TC = (7814, 29)$.
  - MST_{15} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_2.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (13) IK.
- Hapus sisi (14) EI
  - Masukkan sisi (15) GJ dan bangun MST_{16} dengan $TC = (7734, 29)$.
  - MST_{16} adalah solusi efisien dengan karakteristik sisi = 14.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (14) EI.
- Adapun sisi (2) AD, (3) BC, (4) EF, (6) DF, (8) CD, (10) GH, dan (12) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.
- Adapun sisi (7) tidak dihapus karena sisi JL merupakan sisi karakteristik.

Mutasi MST_6

- Hapus sisi (1) JK
  - Masukkan sisi (7) JL dan bangun MST_{17} dengan $TC = (7734, 29)$.
  - MST_{17} adalah solusi *dominated* karena memiliki TC sama dengan MST_{16}.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (1) JK.
- Adapun sisi (2) AD, (3) BC, (4) EF, (5) LK, (6) DF, (8) CD, (10) GH, (12) FG, dan (13) IK jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.
- Adapun sisi (15) tidak dihapus karena sisi GJ merupakan sisi karakteristik.

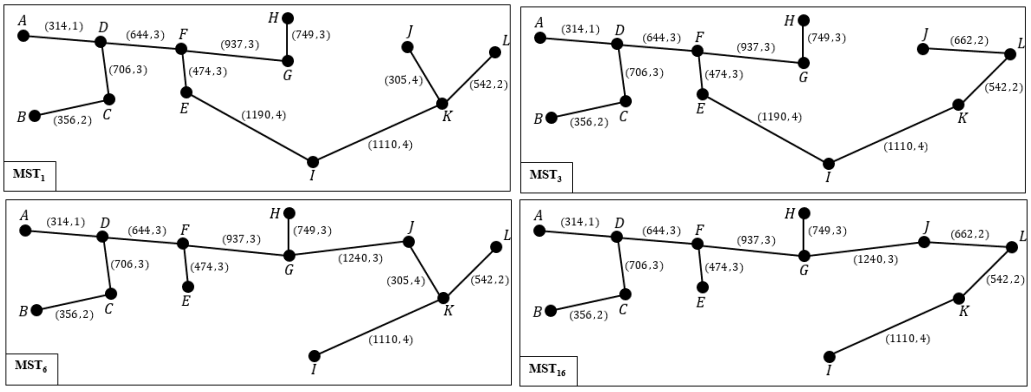
Mutasi MST_{11}

- Hapus sisi (3) LK
  - Masukkan sisi (12) JK dan bangun MST_{18} dengan $TC = (7577, 31)$.
  - MST_{18} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (3) LK.
- Hapus sisi (4) JL
  - Masukkan sisi (12) JK dan bangun MST_{19} dengan $TC = (7457, 31)$.
  - MST_{19} adalah solusi *dominated* karena memiliki TC lebih besar dari MST_6.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (4) JL.
- Hapus sisi (11) GJ
  - Masukkan sisi (14) IK dan bangun MST_{20} dengan $TC = (7684, 30)$.

- MST_{20} adalah solusi *dominated* karena memiliki TC sama dengan MST_3.
- Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (11) GJ.

- Hapus sisi (15) EI
  - Masukkan sisi (14) IK dan bangun MST_{21} dengan $TC = (7997, 29)$.
  - MST_{21} adalah solusi *dominated* karena memiliki TC sama dengan MST_{16}.
  - Tidak ditemukan sisi yang memenuhi kondisi, kembalikan sisi (15) EI.
- Adapun sisi (1) AD, (2) BC, (6) DF, (7) CD, (9) GH, dan (10) FG jika dihapus satu per satu maka tidak ditemukan sisi yang memenuhi kondisi.
- Adapun sisi (5) tidak dihapus karena sisi EF merupakan sisi karakteristik.

Pada tahap ini ditemukan MST baru yang dapat menggantikan MST yang terdapat dalam APS yakni MST_{16} menjadi solusi efisien menggantikan MST_{11}. Sehingga MST_{16} masuk ke dalam APS dan MST_{11} dikeluarkan. Karena tidak ada lagi MST baru yang memenuhi kondisi yang disyaratkan, maka MST yang menjadi solusi efisien atau PF untuk masalah MCMST dari graf pada gambar 3.1 adalah MST_1 dengan $TC = (7327, 32)$, MST_3 dengan $TC = (7684, 30)$, MST_6 dengan $TC = (7377, 31)$, dan MST_{16} dengan $TC = (7734, 29)$.



Gambar 3. Solusi Efisien Menurut EPDA dengan Algortima Prim

3.3. Perbandingan EPDA dengan Algoritma Kruskal dan EPDA dengan Algoritma Prim

Berdasarkan penyelesaian yang dilakukan EPDA yang diterapkan pada masalah optimasi jarak dan waktu untuk beberapa jalan di sekitar kampus UIN Maulana Malik Ibrahim Malang diperoleh bahwa antara EPDA dengan Algrotima Kruskal dan EPDA dengan Algoritma Prim memiliki solusi yang sama. Adapun untuk EPDA dengan Algoritma Kruskal hasil yang diperoleh adalah MST_1 dengan $TC = (7327, 32)$, MST_2 dengan $TC = (7734, 29)$, MST_3 dengan $TC = (7684, 30)$, dan MST_6 dengan $TC = (7377, 31)$. Sedangkan untuk EPDA dengan Algoritma Prim hasil yang diperoleh adalah MST_1 dengan $TC = (7327, 32)$, MST_3 dengan $TC = (7684, 30)$, MST_6 dengan $TC = (7377, 31)$, dan MST_{16} dengan $TC = (7734, 29)$. Sehingga secara sederhana dapat dikatakan bahwa meskipun terdapat perbedaan pada indeksnya, baik EPDA yang dibangun dari Algoritma Kruskal maupun Algoritma Prim menghasilkan solusi yang sama.

Selanjutnya jika memperhatikan banyaknya kemungkinan solusi yang dihasilkan, kedua algoritma ini terdapat perbedaan dalam segi kuantitasnya yakni banyaknya kemungkinan solusi yang dihasilkan oleh EPDA dengan Algoritma Prim ditemukan lebih banyak dari pada EPDA dengan Kruskal. Adapun untuk EPDA dengan Algoritma Prim menghasilkan 21 MST. Sedangkan untuk EPDA dengan Algoritma Kruskal hanya menghasilkan MST sebanyak 13. Artinya EPDA dengan Algoritma Kruskal memiliki proses penyelesaian yang lebih singkat, karena banyaknya kemungkinan solusi yang dicek relatif lebih sedikit dibandingkan dengan EPDA menggunakan Algoritma Prim. Hal ini dikarenakan pada pemilihan sisi yang dilakukan oleh EPDA dengan Algoritma Kruskal tidak hanya memperhatikan kriteria yang dikerjakan, namun sekaligus memperhatikan pertimbangan bobot yang termuat dalam tabel Edge List.

4. KESIMPULAN DAN SARAN

Pemilihan sisi yang dilakukan oleh Algoritma Kruskal dalam konstruksi EPDA tidak hanya memperhatikan kriteria yang dikerjakan, namun sekaligus memperhatikan pertimbangan bobot yang termuat dalam tabel Edge List. Karena Algoritma Prim hanya memperhatikan kriteria yang dikerjakan, maka berakibat banyaknya kemungkinan solusi yang dihasilkan menjadi lebih banyak. EPDA dengan Algoritma Kruskal menghasilkan 13 MST sedangkan EPDA dengan Algoritma Prim menghasilkan 21 MST.

Sebagai lanjutan dari penelitian ini, penulis mengharapkan peneliti selanjutnya menggunakan algoritma lain yang setara dengan Algoritma Kruskal dan Algoritma Prim, misalnya Algoritma Sollin dan Algoritma Boruvka. Kemudian penulis juga mengharapkan penelitian selanjutnya dilakukan dengan menambahkan

program baik dalam bentuk instan maupun manual dengan tujuan untuk mempermudah penyelesaian masalah MCMST.

REFERENSI

[1] D. S. Vianna, J. E. C. Arroyo, P. S. Vieira dan T. R. Azeredo, "Parallel Strategies for a Multi-criteria GRASP Algorithm," *Producao,* vol. XVII, pp. 84-93, 2007.

[2] M. D. Moradkhan, "Multi-Criterion Optimization in Minimum Spanning Trees," *Studia Informatica Universalis,* pp. 185-208, 2010.

[3] G. Zhou dan M. Gen, "Genetic Algorithm Approach on Multi-criteria Minimum Spanning Tree Problem," *European Journal Operations Research,* pp. 141-152, 1999.

[4] E. Keshavarz dan M. Toloo, "A Bi-Objective Minimum Cost-Time Network Flow Problem," *Procedia Economics and Finance,* vol. XXIII, pp. 3-8, 2015.