Implementasi Bellman-Ford untuk Optimasi Rute Pengambilan Sampah di Kota Palembang

Rezania Agramanisti Azdy[1], Febriyanti Darnis[2]

***Abstract*—The problem of waste in big cities in Indonesia is still the main focus of the related department. Generally, the problem is the lack of garbage transport facilities that can be used to transport waste from the temporary shelter (*Tempat Penampungan Sementara*, TPS) to the final processing site (*Tempat Pemrosesan Akhir*, TPA). This lack of garbage transport makes the garbage contained in TPS not fully transportable to TPA due to limited capacity and operational time of the related service. In overcoming this problem, optimization of garbage transport facility utilization can be carried out to optimize the capacity of the garbage transported by finding the shortest path that can be traversed by the garbage transporter. This paper aims to apply the Bellman-Ford algorithm for determining waste collection routes. The Bellman-Ford algorithm allows a negative weighting for its edge value, so that it can anticipate the possible costs to be incurred in the selection of the garbage collection path. The stages of this paper were data requirements analysis, design, implementation, and testing. The results of the study are trajectories with a minimum cost from the origin location to the destination location, although it does not pass through all TPS that must be visited.**

***Intisari*— Permasalahan sampah di kota besar di Indonesia masih menjadi fokus utama dinas terkait. Umumnya, masalah yang ditemui adalah kurangnya fasilitas pengangkut sampah yang dapat digunakan untuk mengangkut sampah dari Tempat Penampungan Sementara (TPS) hingga ke Tempat Pemrosesan Akhir (TPA). Kurangnya pengangkut sampah ini membuat sampah yang terdapat di TPS tidak sepenuhnya dapat diangkut menuju TPA karena keterbatasan daya tampung maupun waktu operasional dinas terkait. Dalam mengatasi permasalahan tersebut, dapat dilakukan optimasi terhadap pemakaian fasilitas pengangkut sampah agar kapasitas sampah yang diangkut menjadi optimal dengan mencari jalur terpendek yang dapat dilalui oleh pengangkut sampah. Makalah ini bertujuan menerapkan algoritme Bellman-Ford untuk penentuan rute pengambilan sampah. Algoritme Bellman-Ford memungkinkan adanya bobot yang negatif untuk nilai *edge*-nya, sehingga dapat mengantisipasi kemungkinan adanya *cost* yang harus dikeluarkan dalam pemilihan jalur pengambilan sampah. Tahapan yang dilakukan adalah analisis kebutuhan data, perancangan, implementasi, dan pengujian. Hasil yang dicapai adalah lintasan dengan *cost* minimum dari lokasi asal ke lokasi tujuan, meskipun tidak melewati seluruh TPS yang harus dikunjungi.**

Kata Kunci—***Shortest Path,* Tempat Penampungan Sementara, Tempat Pemrosesan Akhir.**

[1] *Program Studi Informatika STMIK Palcomtech, Jl. Basuki Rahmat No. 05 Palembang 30129 Indonesia (tlp: 0711-359092; e-mail: rezania@palcomtech.ac.id)*

[2] *Program Studi Sistem Informasi STMIK Palcomtech, Jl. Basuki Rahmat No. 05 Palembang 30129 Indonesia (e-mail: febriyanti_darnis@palcomtech.ac.id)*

I. PENDAHULUAN

Sampah merupakan permasalahan yang tidak akan ada habisnya dan mudah ditemui di seluruh kota di Indonesia, terutama di kota-kota besar maupun ibukota provinsi, seperti Palembang. Permasalahan sampah meningkat seiring dengan bertambahnya jumlah penduduk yang berpotensi menghasilkan sampah dan tidak diiringi dengan kemampuan pengelolaan sampah yang sebanding, sehingga penumpukan sampah tetap ada. Meskipun Palembang termasuk dalam sepuluh kota terbaik dalam pengelolaan sampah di seluruh Indonesia, tetapi dari total 1.300 ton sampah yang dihasilkan, hanya 800 ton sampah saja yang dapat diangkut ke Tempat Pemrosesan Akhir (TPA) dan meninggalkan sekitar 500 ton sampah yang tidak dapat diangkut dan tetap berada di Tempat Penampungan Sementara (TPS) [1]. Kepala Dinas Lingkungan Hidup dan Kebersihan (DLHK) Kota Palembang dalam menyatakan penyebab permasalahan tersebut adalah adanya keterbatasan personel dan mobil pengangkut sampah di lingkup DLHK Kota Palembang [1]. Berdasarkan kapasitas sampah yang dihasilkan per hari, idealnya DLHK Kota Palembang memiliki 190 hingga 250 truk pengangkut sampah. Namun, hingga bulan April 2018, DLHK Kota Palembang hanya memiliki 105 truk pengangkut sampah.

Dari permasalahan tersebut, makalah ini mencoba memberikan solusi terhadap permasalahan di atas dengan melakukan optimasi pengambilan sampah di TPS-TPS yang tersebar di Kota Palembang agar daya angkut sampah menuju TPA lebih optimal. Optimasi dapat dilakukan dengan menentukan rute terpendek yang dapat dilalui truk pengangkut sampah agar lamanya waktu truk di perjalanan menjadi minimum dan truk dapat melakukan pengambilan di TPS berkali-kali selama jam operasional. Penentuan rute terpendek dapat dilakukan dengan menghitung jarak terpendek yang harus dilalui truk pengangkut sampah menggunakan metode atau teknik yang telah ada. Beberapa penelitian sebelumnya membahas optimasi penentuan jalur menggunakan algoritme genetika [2] dan *ant colony* [3] yang menghasilkan jarak terpendek pada permasalahan *routing* maupun *traveling salesman problem*. Pada makalah ini, pencarian lintasan terpendek dilakukan dengan mengimplementasikan salah satu *shortest path algorithm*, yaitu algoritme Bellman-Ford. Algoritme Bellman-Ford memungkinkan adanya bobot yang negatif untuk nilai *edge*-nya, sehingga dapat mengantisipasi kemungkinan adanya *cost* yang harus dikeluarkan dalam pemilihan jalur pengambilan sampah.

Berdasarkan latar belakang yang telah dijabarkan, dapat dirumuskan permasalahan pada makalah ini, yaitu cara menerapkan algoritme Bellman-Ford pada sebuah sistem yang dapat menghasilkan keluaran berupa lintasan terpendek pengambilan sampah di Kota Palembang, yang dimulai dari lokasi sopir (*driver*) truk berada dan berakhir di lokasi TPA.

II. OPTIMASI RUTE PENANGANAN SAMPAH

Beberapa penelitian sebelumnya mengenai optimasi rute penanganan sampah pernah dilakukan. Algoritme genetika diterapkan untuk menentukan rute optimal bagi kendaraan pengangkut sampah dari kantor Dinas Kebersihan dan Pertamanan (DKP) menuju ke titik-titik tempat sampah dan berakhir di TPA [4]. Dari penelitian ini, solusi terbaik dapat ditemukan dengan adanya pengaturan parameter pada algoritme genetika, yaitu semakin banyak jumlah populasi yang terbatas akan menghasilkan nilai *fitness* yang semakin baik. Akan tetapi, jumlah populasi yang melampaui batas dan terlalu banyak membuat aplikasi berjalan semakin lambat, sehingga jumlah populasi yang mencapai nilai rata-rata *fitness* tertinggi adalah 100 populasi, dengan rata-rata nilai *fitness* 0,495. Sebanding dengan jumlah populasi, semakin banyak generasi yang dijalankan akan menghasilkan nilai rata-rata *fitness* yang semakin baik. Rata-rata nilai *fitness* tertinggi dihasilkan pada generasi ke-900 dengan rata-rata nilai fitness 0,462.

Penelitian berikutnya menentukan rute terpendek pengambilan sampah di Kota Merauke [5]. Algoritme yang digunakan pada penelitian ini adalah algoritme Dijkstra dengan menggunakan parameter berupa tempat asal dan tempat tujuan. Hasil dari penelitian yang mengombinasikan algoritme Dijkstra dengan metode *saving heuristic* ini adalah rute terpendek sesuai dengan kriteria yang diinginkan, yaitu adanya depot tempat kendaraan berangkat dan pulang, tiap konsumen tepat dilayani satu kali, dan kapasitas sampah yang diangkut tidak melebihi kapasitas maksimal kendaraan.

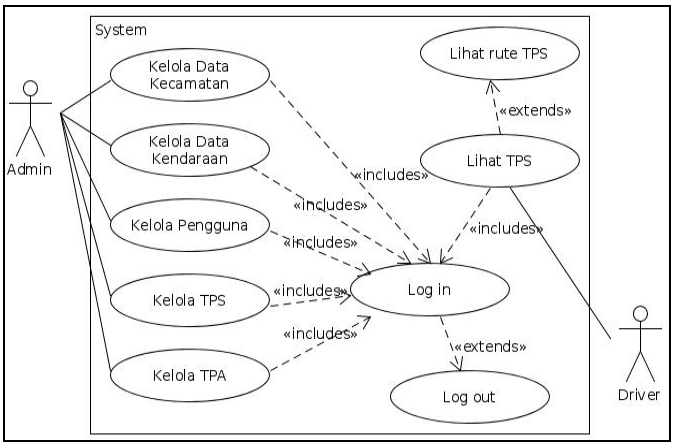
Penelitian lainnya menggunakan algoritme Floyd Warshall untuk menentukan lintasan terpendek pengangkutan sampah [6]. Pada penelitian tersebut, simpul merepresentasikan TPS dan TPA yang ada di Kabupaten Kubu Raya dan sisi merepresentasikan lintasan yang menghubungkan antar TPS dan/ke TPA. Lintasan truk sampah melalui TPS-TPS sebagai titik pengambilan sampah dan TPA sebagai tujuan akhir dengan panjang lintasan terpendek adalah 38,7 km.

Adapun perbedaan makalah ini dengan penelitian yang telah ada terletak pada algoritme optimasi yang digunakan. Pada makalah ini, optimasi dilakukan menggunakan algoritme Bellman-Ford, merepresentasikan permasalahan dalam bentuk graf, dengan *vertex* berupa lokasi sopir, lokasi TPS, dan lokasi TPA, dan *edge* berupa jarak antar lokasi *vertex*. Pemilihan algoritme Bellman-Ford selain dikarenakan kelebihannya dalam menangani permasalahan graf dalam *edge* yang bernilai negatif [7], algoritme ini juga dapat mendeteksi adanya *cycle* dengan bobot negatif [8], [9].

III. ALGORITME BELLMAN-FORD

Algoritme Bellman-Ford merupakan salah satu algoritme yang dapat digunakan untuk mencari lintasan terpendek pada graf berbobot [10]. Yang dimaksud dengan lintasan terpendek adalah lintasan minimum yang dapat dilalui untuk menempuh satu titik menuju titik lainnya. Algoritme Bellman-Ford termasuk dalam kategori *single source shortest path algorithm*, yaitu berasal dari satu titik (sumber) dan akan menghitung semua lintasan terpendek yang berawal dari titik tersebut. Langkah algoritme Bellman-Ford dalam menentukan lintasan terpendek adalah sebagai berikut [10].

1. Menentukan titik asal dan mendaftar semua titik atau sisi.
2. Memberi nilai pada titik asal sama dengan 0 dan titik lainnya dengan nilai tak hingga.
3. Memulai iterasi terhadap semua titik, diawali dari titik asal dan dilakukan hingga semua titik terjelajahi. Melakukan perhitungan UV = sisi yang menghubungkan U dengan V, dengan U adalah titik asal dan V adalah titik tujuan. Jika jarak V lebih besar dari jarak U+bobot UV, maka jarak V diisi dengan jarak U+bobot UV.
4. Melakukan iterasi pada semua sisi untuk memeriksa siklus negatif yang mungkin ada di dalam graf dengan langkah: untuk semua sisi UV, jika jarak V lebih besar dari jarak U+bobot UV, maka graf tersebut memiliki siklus negatif.



Gbr. 1 Diagram *use case* sistem.

Algoritme Bellman-Ford dapat dituliskan dalam bentuk formula seperti pada (1) [11].

$$M[I,y] = min\,[\,M(y),\,(\,M(i,s) + EdgeWeight) \quad (1)$$

dengan

M = *marked*
M(y) = nilai dalam *vertex* awal (*node Y*)
M(i,s) = nilai dalam *vertex* dari tujuan (*node i* dan *s*)
EdgeWeight = bobot dari sisi yang menghubungkan *vertex*.

IV. METODE

A. Analisis Kebutuhan

Analisis kebutuhan dilakukan melalui wawancara untuk memperoleh informasi mengenai prosedur pengambilan sampah, yaitu bahwa setiap sopir memiliki tanggung jawab untuk mengambil sampah di TPS-TPS yang telah ditentukan.

Dari analisis kebutuhan, dapat disimpulkan kebutuhan fungsional dari sistem yang dimodelkan menggunakan diagram *use case.* Pengguna sistem terdiri atas admin (administrator) dan sopir dengan kebutuhan fungsional masing-masing diperlihatkan pada Gbr. 1. Kendaraan angkut yang digunakan disimpan di rumah masing-masing sopir, sehingga lintasan yang dicari dimulai dari lokasi sopir berada, menelusuri TPS-TPS yang telah menjadi penugasannya, dan berakhir di TPA.

```
function BellmanFord(G, S) {
  dist[S] ← 0

  for each vertex V in G {
    dist[V] ← inf
    pre[V] ← null
  }

  for each vertex V in G {
    for each edge (U,V) in G {
      if dist[U] + weight(U,V) < dist[V]
        dist[V] ← dist[U] + weight(U.V)
        pre[V] ← U
      }
    }
  }

  for each edge (U,V) in G {
    if dist[U] + weight(U,V) < dist[V]
      Error: Negative Cycle Exist
    }
  }

  return dist[ ], pre[ ]
}
```

Gbr. 2 *Pseudo-code* algoritme Bellman-Ford.

B. Perancangan

Pada tahap ini, dilakukan perancangan sistem dengan memodelkan basis data, alur kerja (aktivitas), dan struktur statis pada sistem. Pada tahapan ini juga dilakukan pengambilan data sebagai sampel untuk pengolahan data pada sistem berupa titik-titik koordinat lokasi TPS dan TPA.

C. Implementasi

Model yang telah dibuat di tahap sebelumnya diterjemahkan ke dalam bentuk kode program yang sesuai. Pada tahapan ini, juga diterapkan algoritme Bellman-Ford ke dalam sistem, untuk dapat memberikan keluaran berupa lintasan terpendek yang dapat dilalui sopir.

Algoritme Bellman-Ford diimplementasikan menggunakan bahasa pemrograman PHP dengan menerjemahkan *pseudo-code* pada Gbr. 2. Fungsi menerima parameter sebuah graf yang berisi titik lokasi TPS, TPA, dan sopir sebagai *vertex*; dan jarak antara titik-titik tersebut sebagai *weight*. Pada algoritme Bellman-Ford, iterasi sering juga disebut dengan relaksasi yang dilakukan sebanyak (jumlah *vertex* – 1) kali. Fungsi relaksasi adalah memperbarui (*update*) nilai *distance* pada sebuah *vertex* dengan nilai yang lebih kecil dan menyimpan *vertex* pendahulu (*pre*) yang dapat mengubah nilai *distance*. Kompleksitas algoritme di atas adalah $O(V.E)$ dengan V adalah banyaknya *vertex* dan *E* adalah banyaknya *edge*.

V. PENGUJIAN DAN PEMBAHASAN

Data yang diperlukan pada makalah ini adalah data berupa titik lokasi sopir, lokasi TPS, dan lokasi TPA, dilihat dari posisinya pada permukaan bumi berdasarkan garis bujur dan lintang.

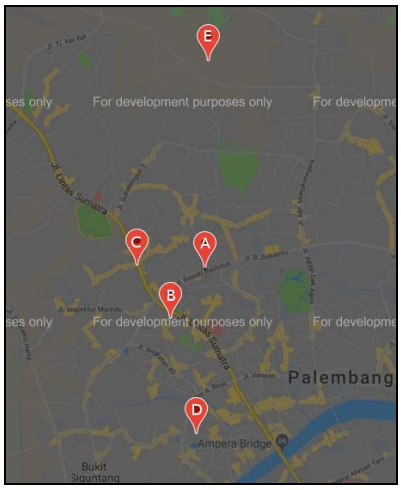
TABEL I
DATA TPS

No. Kendaraan	Rute Angkutan
BG 83XX MZ	TPS Pasar Pahlawan (Kontainer)
	TPS Pasar KM 5 (Kontainer)
	TPS Bukit Kecil (Pasar Kubah) (Kontainer)

Sumber: DLHK Kota Palembang, 2018 [12]

TABEL II
LOKASI DALAM NILAI BUJUR DAN LINTANG

Lokasi	Bujur	Lintang	*Vertex*
Sopir (STMIK Palcomtech)	-2.954299	104.748283	A
TPS Pasar Pahlawan (Kontainer)	-2.964408	104.741428	B
TPS Pasar KM 5 (Kontainer)	-2.953833	104.734691	C
TPS Bukit Kecil (Pasar Kubah) (Kontainer)	-2.987493	104.746541	D
TPA Sukawinatan	-2.913106	104.748894	E

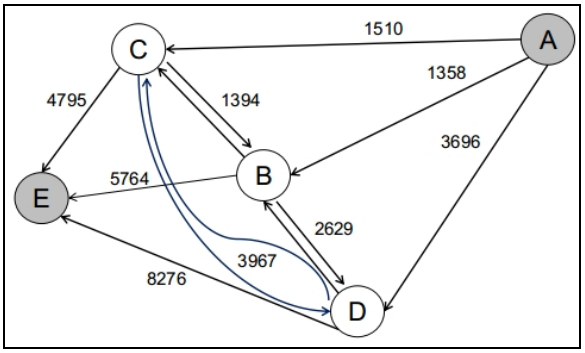


Gbr. 3 Titik lokasi pada peta.

Data yang digunakan sebagai sampel pengujian adalah data TPS yang harus dikunjungi dan menjadi tanggung jawab sopir kendaraan pengangkut sampah dengan nomor polisi BG 83XY MZ, seperti ditunjukkan pada Tabel I. Jika diasumsikan posisi sopir berada di STMIK Palcomtech dan TPA yang dituju adalah TPA Sukawinatan, maka lokasi sopir, TPS, dan TPA dituliskan titik koordinatnya dalam bentuk bujur (*longitude*) dan lintang (*latitude*) seperti pada Tabel II.

Dalam menentukan lintasan terpendek menggunakan algoritme Bellman-Ford, permasalahan yang ada dituangkan dalam bentuk graf berbobot dengan *vertex* pada graf diambil dari titik lokasi pada Tabel II, seperti ditunjukkan pada Gbr. 3. Selain memberikan nilai kembalian berupa jarak (*dist*) setiap *vertex*, fungsi juga mengembalikan nilai berupa *vertex* pendahulu (*pre*).

Bobot masing-masing *edge* yang menghubungkan antar *vertex* diambil dari jarak antara dua titik lokasi dan ditentukan



Gbr. 4 Graf berbobot representasi permasalah lintasan pengambilan sampah.

TABEL III
JARAK ANTAR LOKASI SEBAGAI BOBOT (*WEIGHT*) PADA GRAF

Lokasi asal	Lokasi Tujuan	Jarak (pembulatan dalam meter)
A	B	1.358
A	C	1.510
A	D	3.696
B	C	1.394
B	D	2.629
B	E	5.764
C	B	1.394
C	D	3.967
C	E	4.795
D	B	2.629
D	C	3.967
D	E	8.276

menggunakan formula *haversine*. Formula *haversine* menghitung jarak antar dua titik berdasarkan letaknya terhadap garis bujur dan lintang [13]. Formula *haversine* yang dimaksud adalah sebagai berikut [14].

$$a = \sin^2\left(\frac{\Delta \text{lat}}{2}\right) + cos(lat2) \,.\, cos(lat1) \,.\, \sqrt{\sin^2\left(\frac{\Delta \text{long}}{2}\right)} \quad (2)$$

$$d = 2r.arcsin(\sqrt{a}) \,. \quad (3)$$

Contoh perhitungan jarak antara lokasi sopir yang berada di titik A dengan TPS Pasar Pahlawan yang berada di titik B adalah

Δlat = (-2,964408*0,0174532925)-(-2,954299*0,0174532925)
= -0,000176435334

$\Delta long$ = (104,741428*0,0174532925) -
(104,748283*0,0174532925)
= -0,000119642320

$sin2(\Delta lat/2)$ = $sin2$(-0,000176435334/2)
= $sin2$(-0,0000882177)
= 0,000000007782

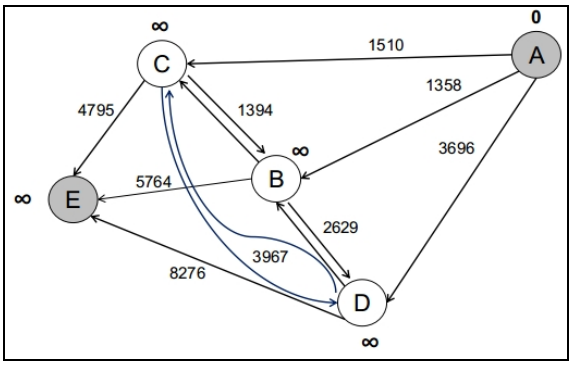
$cos(latitude2)$ = cos(-2,964408*0,0174532925)
= cos(-0,05173868)
= 0,998661853046

$cos(latitude1)$ = cos(-2,954299*0,0174532925)
= cos(-0,051562245)
= 0,998670961962

$sin2(\Delta long/2)$ = $sin2$(-0,000119642320/2)
= $sin2$(-0,0000598212)
= 0,000000003579

a = 0,000000007782 +
(0,998661853046 * 0,998670961962 * 0,000000003579)
= 0,000000011351

d = 2*6371*$arcsin(\sqrt{a})$
= 2*6371*$arcsin$(√0,000000011351)
= 2*6371*$arcsin$(0,000106543)
= 2*6371*0,000106543
= 1,357570906 km.



Gbr. 5 Nilai *distance* untuk seluruh *vertex* di awal perhitungan algoritme.

TABEL IV
KEADAAN AWAL GRAF

	A	B	C	D	E
dist	0	∞	∞	∞	∞
pre	*null*	*null*	*null*	*null*	*null*

Hasil perhitungan jarak antar titik beserta arah lintasan seluruhnya diperlihatkan pada Tabel III. Gbr. 4 merupakan terjemahan lokasi dan jarak pada sebuah graf berbobot. Posisi sopir akan selalu menjadi titik awal, sehingga lintasan (arah graf) selalu keluar dari *vertex* ini menuju ke *vertex* lain yang merepresentasikan lokasi TPS. Lintasan (*edge*) dari setiap TPS ke TPS lainnya tidak memiliki arah karena sifatnya yang bolak-balik dan lokasi TPA menjadi titik tujuan atau akhir, sehingga seluruh arah graf dari *vertex* yang merepresentasikan TPS selalu berakhir di *vertex* ini.

Berdasarkan algoritme yang terdapat pada Gbr. 2, langkah awal dalam melakukan perhitungan menggunakan algoritme Bellman-Ford adalah dengan memberikan nilai *distance* untuk *vertex* awal 0 dan untuk *vertex* lainnya *infinity* (∞), seperti pada Gbr. 5, serta *previous vertex* untuk *vertex* selain titik awal adalah *null*, seperti pada Tabel IV.

TABEL V
PERHITUNGAN ALGORITME BELLMAN-FORD PADA ITERASI I

(U,V)	*dist*[U]	*weight*[U,V]	*dist*[V]	*dist*[V] terbaru	*pre*[V]
(A,B)	*dist*[A] = 0	*weight*[A,B] = 1.358	*dist*[B] = ∞	*dist*[B] = 1.358	A
(A,C)	*dist*[A] = 0	*weight*[A,C] = 1.510	*dist*[C] = ∞	*dist*[C] = 1.510	A
(A,D)	*dist*[A] = 0	*weight*[A,D] = 3.696	*dist*[D] = ∞	*dist*[D] = 3.696	A
(B,C)	*dist*[B] = 1.358	*weight*[B,C] = 1.394	*dist*[C] = 1.510	*dist*[C] = 1.510	A
(B,D)	*dist*[B] = 1.358	*weight*[B,D] = 2.629	*dist*[D] = 3.696	*dist*[D] = 3.696	A
(B,E)	*dist*[B] = 1.358	*weight*[B,E] = 5.764	*dist*[E] = ∞	*dist*[E] = 7.122	B
(C,B)	*dist*[C] = 1.510	*weight*[C,B] = 1.394	*dist*[B] = 1.358	*dist*[B] = 1.358	A
(C,D)	*dist*[C] = 1.510	*weight*[C,D] = 3.967	*dist*[D] = 3.696	*dist*[D] = 3.696	A
(C,E)	*dist*[C] = 1.510	*weight*[C,E] = 4.795	*dist*[E] = 7.122	*dist*[E] = 6.305	C
(D,B)	*dist*[D] = 3.696	*weight*[D,B] = 2.629	*dist*[B] = 1.358	*dist*[B] = 1.358	A
(D,C)	*dist*[D] = 3.696	*weight*[D,C] = 3.967	*dist*[C] = 1.510	*dist*[C] = 1.510	A
(D,E)	*dist*[D] = 3.696	*weight*[D,E] = 8.276	*dist*[E] = 6.305	*dist*[E] = 6.305	C

TABEL VI
HASIL PERHITUNGAN ALGORITME BELLMAN-FORD SETELAH ITERASI I

	A	B	C	D	E
dist	0	1.358	1.510	3.696	6.305
pre	-	A	A	A	C

Langkah berikutnya adalah melakukan relaksasi dengan mengubah nilai *distance* dan *pre vertex* dengan algortime sebagai berikut.

```
if dist[U] + weight(U,V) < dist[V]
       dist[V] <- dist[U] + weight(U,V)
       pre[V] <- U.
```

Pada graf terdapat lima buah *vertex*, sehingga relaksasi dilakukan sebanyak empat kali.

Relaksasi pada iterasi I diperlihatkan pada Tabel V. Untuk setiap lintasan dari *vertex* U ke *vertex* V, dilakukan perbandingan nilai *distance*[U]+*weight*[U,V] dengan nilai *distance*[V]. Jika nilainya lebih kecil, maka nilai *distance*[V] diganti dengan nilai *distance*[U]+*weight*[U,V] dan nilai *pre*[V] diperbarui dengan U. Relaksasi dilakukan dengan mendata seluruh *edge* yang terdapat pada graf.

Nilai *distance* dan *pre* untuk setiap *vertex* setelah iterasi I diperlihatkan pada Tabel VI. Relaksasi pada iterasi II melakukan perhitungan dengan cara yang sama dan mengambil nilai *distance* yang dihasilkan di iterasi sebelumnya. Perhitungan iterasi II diperlihatkan pada Tabel VII. Nilai *distance* dan *pre* untuk setiap *vertex* setelah iterasi kedua diperlihatkan pada Tabel VIII.

TABEL VII
PERHITUNGAN ALGORITME BELLMAN-FORD PADA ITERASI II

(U,V)	*dist*[U]	*weight*[U,V]	*dist*[V]	*dist*[V] terbaru	*pre*[V]
(A,B)	*dist*[A] = 0	*weight*[A,B] = 1.358	*dist*[B] = 1.358	*dist*[B] = 1.358	A
(A,C)	*dist*[A] = 0	*weight*[A,C] = 1.510	*dist*[C] = 1.510	*dist*[C] = 1.510	A
(A,D)	*dist*[A] = 0	*weight*[A,D] = 3.696	*dist*[D] = 3.696	*dist*[D] = 3.696	A
(B,C)	*dist*[B] = 1.358	*weight*[B,C] = 1.394	*dist*[C] = 1.510	*dist*[C] = 1.510	A
(B,D)	*dist*[B] = 1.358	*weight*[B,D] = 2.629	*dist*[D] = 3.696	*dist*[D] = 3.696	A
(B,E)	*dist*[B] = 1.358	*weight*[B,E] = 5.764	*dist*[E] = 6.305	*dist*[E] = 6.305	C
(C,B)	*dist*[C] = 1.510	*weight*[C,B] = 1.394	*dist*[B] = 1.358	*dist*[B] = 1.358	A
(C,D)	*dist*[C] = 1.510	*weight*[C,D] = 3.967	*dist*[D] = 3.696	*dist*[D] = 3.696	A
(C,E)	*dist*[C] = 1.510	*weight*[C,E] = 4.795	*dist*[E] = 6.305	*dist*[E] = 6.305	C
(D,B)	*dist*[D] = 3.696	*weight*[D,B] = 2.629	*dist*[B] = 1.358	*dist*[B] = 1.358	A
(D,C)	*dist*[D] = 3.696	*weight*[D,C] = 3.967	*dist*[C] = 1.510	*dist*[C] = 1.510	A
(D,E)	*dist*[D] = 3.696	*weight*[D,E] = 8.276	*dist*[E] = 6.305	*dist*[E] = 6.305	C

TABEL VIII
HASIL PERHITUNGAN ALGORITME BELLMAN-FORD SETELAH ITERASI II

	A	B	C	D	E
dist	0	1.358	1.510	3.696	6.305
pre	-	A	A	A	C

Dari Tabel VIII, dapat dilihat hasil perhitungan algoritme Bellman-Ford setelah iterasi II tidak berbeda dengan hasil perhitungan pada iterasi I di Tabel VI. Jika diteruskan lagi ke iterasi berikutnya, hasil perhitungannya akan tetap, karena nilai *distance* dan *weight*-nya selalu sama, sehingga iterasi dapat dihentikan karena hanya akan menghasilkan nilai yang sama dari iterasi sebelumnya. Setelah iterasi dihentikan, langkah berikutnya adalah melakukan pengujian untuk menentukan ketepatan perhitungan yang telah dilakukan. Dari hasil pengujian yang dilakukan, seperti ditunjukkan pada Tabel IX, perhitungan algoritme Bellman-Ford telah berhasil diimplementasikan dengan baik.

Gbr. 6 merupakan representasi graf dari hasil perhitungan akhir Bellman-Ford. Rute yang dapat dilalui dari suatu titik menuju titik lain diperlihatkan dengan *edge* yang bergaris tebal. Misalnya, untuk mencapai lokasi akhir (*vertex* E) dari lokasi awal (*vertex* A), lintasan yang dapat dilalui adalah A-C-E. Dari representasi graf tersebut, dapat dilihat bahwa dari lintasan yang dihasilkan, seluruh *vertex* yang merupakan representasi dari TPS tidak dapat dikunjungi seluruhnya. Berdasarkan lintasan yang terbentuk, jika sopir ingin menuju

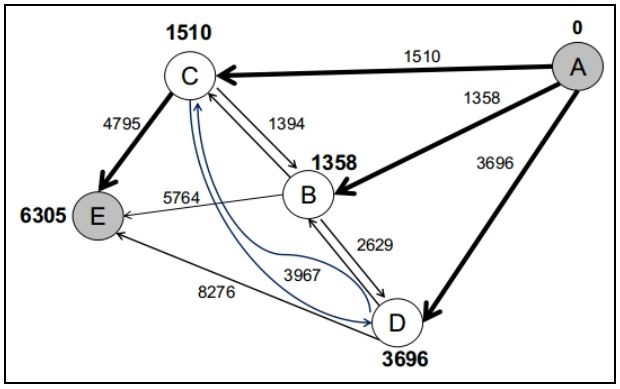
TABEL IX
PENGUJIAN TERHADAP HASIL PERHITUNGAN

(U,V)	***dist*[V] ≤ *dist*[U]+*weight*[U,V]**	Sesuai/Tidak
(A,B)	*dist*[B] ≤ *dist*[A]+*weight*[A,B] 1.358 ≤ 0 + 1.358 1.358 ≤ 1.358	Sesuai
(A,C)	*dist*[C] ≤ *dist*[A]+*weight*[A,C] 1.510 ≤ 0 + 1.510 1.510 ≤ 1.510	Sesuai
(A,D)	*dist*[D] ≤ *dist*[A]+*weight*[A,D] 63.035 ≤ 0 + 6.305 6.305 ≤ 6.305	Sesuai
(B,C)	*dist*[C] ≤ *dist*[B]+*weight*[B,C] 1.510 ≤ 1.358 + 1.394 1.510 ≤ 2.752	Sesuai
(B,D)	*dist*[D] ≤ *dist*[B]+*weight*[B,D] 3.696 ≤ 1.358 + 2.629 3.696 ≤ 3.987	Sesuai
(B,E)	*dist*[E] ≤ *dist*[B]+*weight*[B,E] 6.305 ≤ 1.358 + 5.764 6.305 ≤ 7.122	Sesuai
(C,B)	*dist*[B] ≤ *dist*[C]+*weight*[C,B] 1.358 ≤ 1.510 + 1.394 1.358 ≤ 2.904	Sesuai
(C,D)	*dist*[D] ≤ *dist*[C]+*weight*[C,D] 3.696 ≤ 1.510 + 3.967 3.696 ≤ 5.477	Sesuai
(C,E)	*dist*[E] ≤ *dist*[C]+*weight*[C,E] 6.305 ≤ 1.510 + 4.795 6.305 ≤ 6.305	Sesuai
(D,B)	*dist*[B] ≤ *dist*[D]+*weight*[D,B] 3.696 ≤ 3.696 + 2.629 3.696 ≤ 6.325	Sesuai
(D,C)	*dist*[C] ≤ *dist*[D]+*weight*[D,C] 1.510 ≤ 3.696 + 3.967 1.510 ≤ 7.663	Sesuai
(D,E)	*dist*[E] ≤ *dist*[D]+*weight*[D,E] 6.305 ≤ 3.696 + 8.276 6.305 ≤ 11.972	Sesuai

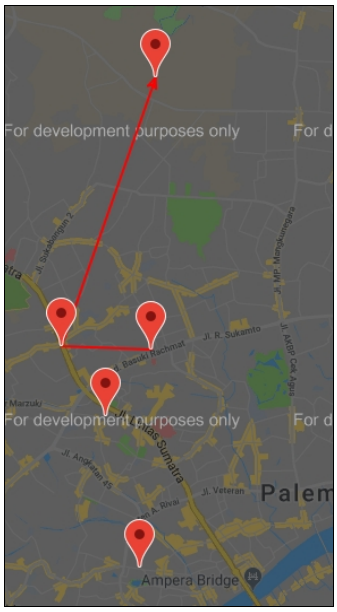
ke lokasi B, maka sopir dapat langsung menuju lokasi B dengan jarak minimum 1.358. Akan tetapi, dari lokasi B tidak ada lintasan yang dapat dilalui untuk menelusuri TPS lainnya, untuk tetap mempertahankan *cost* yang dikeluarkan dalam melakukan kunjungan ke seluruh TPS dan berakhir di TPA tetap minimum. Berdasarkan implementasi algoritme Bellman-Ford pada data TPS yang harus dikunjungi sopir truk BG 83XX MZ, rute yang dapat dilalui dari lokasi awal sopir menuju ke TPA diperlihatkan pada Gbr. 7.

VI. KESIMPULAN

Berdasarkan pembahasan pada bagian sebelumnya, dari titik awal (lokasi sopir) menuju ke titik akhir (lokasi TPA) dapat ditelusuri dengan urutan melalui TPS Pasar KM5 terlebih dahulu. Algoritme Bellman-Ford yang telah diterapkan dapat menghasilkan keluaran berupa lintasan dari lokasi awal hingga lokasi akhir dengan nilai *cost* berupa jarak minimum, tetapi tidak menjamin kunjungan ke seluruh TPS dilakukan.



Gbr. 6 Nilai *distance* untuk seluruh *vertex* di awal perhitungan algoritme.



Gbr. 7 Lintasan sopir truk BG 83XY MZ dari lokasi awal ke TPA.

Dari kesimpulan tersebut, hal yang dapat dilakukan untuk perbaikan pada penelitian berikutnya adalah dengan menentukan terlebih dahulu TPS yang akan dituju, kemudian mengubah titik awal dengan lokasi sopir, *vertex* dengan titik-titik setiap (batas) jalan yang mungkin untuk menuju ke TPS, dan titik akhirnya adalah lokasi TPS tersebut.

UCAPAN TERIMA KASIH

Terima kasih disampaikan kepada pihak DLHK Kota Palembang yang telah memberikan data penelitian yang dibutuhkan, serta Kementrian Riset, Teknologi, dan Pendidikan Tinggi yang telah mendanai penelitian ini dalam skema Penelitian Dosen Pemula dengan nomor kontrak 108/SP2H/LT/DRPM/2019 tanggal 11 Maret 2019.

REFERENSI

[1] Y. Triansyah, (2018) “500 Ton Sampah di Palembang Per Harinya tak Bisa Diangkut Ini Alasannya,” [Online], https://palembang.tribunnews.com/2018/04/26/500-ton-sampah-di-palembang-per-harinya-tak-bisa-diangkut-ini-alasannya, tanggal akses: 4-Sep-2019.

[2] M. Lestandy, S.H. Pramono, dan M.Aswin, "Optimasi Routing pada Metropolitan Mesh Network Menggunakan Adaptive Mutation Genetic

Algorithm," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, Vol. 6, No. 4, hal. 430-435, Nov. 2017.

[3] A.A. Ismail, S. Herdjunanto, dan Priyatmadi, "Penerapan Algoritme Ant System dalam Menemukan Jalur Optimal pada Traveling Salesman Problem (TSP) dengan Kekangan Kondisi Jalan," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, Vol. 1, No. 3, hal. 43-48, Nov. 2012.

[4] R.M. Krishnanda, B.D. Setiawan, dan Marji, "Optimasi Penentuan Rute Terpendek Pengambilan Sampah Menggunakan Multi Travelling Salesman Problem," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, Vol. 2, No. 6, hal. 2227-2234, Jun. 2018.

[5] S. Andayani dan E.W. Perwitasari, "Penentuan Rute Terpendek Pengambilan Sampah di Kota Merauke Menggunakan algoritme Dijkstra," *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2014 (SEMANTIK 2014)*, 2014, hal. 164-170.

[6] V. Setiawan, M. Kiftiah, dan W.B. Partiwi, "Analisis Algoritme Floyd Warshall untuk Menentukan Lintasan Terpendek Pengangkutan Sampah (Studi Kasus: Pengangkutan Sampah di Kabupaten Kubu Raya)," *Buletin Ilmiah Math. Stat. dan Terapannya (Bimaster)*, Vol. 6, No. 3, hal. 221-230, 2017.

[7] R. Pramudita dan N. Safitri, "Algoritma Bellman-Ford untuk Menentukan Jalur Tercepat dalam Sistem Informasi Geografis," *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic*, Vol. 6, No. 2, hlm. 105-114, Sep. 2018.

[8] S. Hamdi dan Prihandoko, "Analisis Algoritma Dijktra dan Algoritma Bellman-Ford sebagai Penentuan Jalur Terpendek Menuju Lokasi Kebakaran (Studi Kasus: Kecamatan Praya Kota)," *ENERGY*, Vol. 8, No. 1, hal. 26-32, Mei 2018.

[9] A. Serdano, M. Zarlis, dan D. Hartama, "Perbandingan Algoritma Dijkstra dan Bellman-Ford Dalam Pencarian Jarak Terpendek Pada SPBU," *Prosiding SENSASI 2019*, 2019, hal. 259-264.

[10] P.M. Hasugian, "Analisa dan Implementasi Algoritme Bellman Ford dalam Menentukan Jalur Terpendek Pengantaran Barang dalam Kota," *Jurnal Mantik Penusa*, Vol. 18, No. 2, hal.118-123, Des. 2015.

[11] S. Wijaya, "Implementasi Algoritma Bellman Ford pada Aplikasi Pencarian Pengobatan Patah Tulang Kem Kem Terdekat di Kota Medan Berbasis Android," *Jurnal Riset Komputer (JURIKOM)*, Vol. 6, No. 1, hal. 30-36, Feb. 2019.

[12] H.R. Harja, "Penyampaian Data Jumlah TPS," Dinas Lingkungan Hidup dan Kebersihan Kota Palembang, Feb. 2018.

[13] Gintoro, I.W. Suharto, F. Rachman, dan D. Halim, "Analisis dan Perancangan Sistem Pencarian Taksi Terdekat dengan Pelanggan Menggunakan Layanan Berbasis Lokasi," *Prosiding Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010)*, 2010, hal. B-34-38.

[14] C.N. Alam, K. Manaf, A.R. Atmadja, dan D.K. Aurum, "Implementation of Haversine Formula for Counting Event Visitor in the Radius Based on Android Application," *4th Int. Conf. on Cyber and IT Service Management*, 2016, hal. 1-6.