UNIVERSITAS INDONESIA

PENDEKATAN ALGORITMA *AUCTION* PADA PENJADWALAN KENDARAAN BUS *RAPID TRANSIT* DENGAN MEMPERHATIKAN JADWAL PENGISIAN BAHAN BAKAR DAN APLIKASINYA PADA PENJADWALAN KENDARAAN BUS TRANSJAKARTA

SKRIPSI

LAILI MIFTAHUR RIZQI
0806452204

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
NOVEMBER 2012

UNIVERSITAS INDONESIA

PENDEKATAN ALGORITMA *AUCTION* PADA PENJADWALAN KENDARAAN BUS *RAPID TRANSIT* DENGAN MEMPERHATIKAN JADWAL PENGISIAN BAHAN BAKAR DAN APLIKASINYA PADA PENJADWALAN KENDARAAN BUS TRANSJAKARTA

SKRIPSI
Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

LAILI MIFTAHUR RIZQI
0806452204

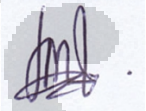
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
NOVEMBER 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber, baik yang dikutip maupun dirujuk,

telah saya nyatakan dengan benar.

Nama	: Laili Miftahur Rizqi
NPM	: 0806452204
Tanda Tangan	: 
Tanggal	: 20 November 2012

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Laili Miftahur Rizqi
NPM	: 0806452204
Program Studi	: Sarjana Matematika
Judul Skripsi	: Pendekatan Algoritma *Auction* pada Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar dan Aplikasinya pada Penjadwalan Kendaraan Bus TransJakarta

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Helen Burhan, M.Si.	( )
Penguji	: Arie Wibowo, M.Si.	( )
Penguji	: Dra. Denny Riama Silaban, M.Kom.	( )
Penguji	: Dhian Widya, M.Kom.	( )

Ditetapkan di	: Depok
Tanggal	: 20 November 2012

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah SWT atas segala rahmat dan karunia yang diberikan sehingga penulis dapat menyelesaikan tugas akhir ini. Penulisan tugas akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan tugas akhir ini, sangatlah sulit bagi penulis untuk menyelesaikan tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

(1) Ibu Helen Burhan, M.Si. selaku pembimbing tugas akhir penulis. Terima kasih atas kemudahan, pencerahan, dan ilmu yang dialirkan kepada penulis sehingga tugas akhir ini dapat selesai dengan baik.

(2) Ibu Dra. Saskya Mary Soemartodjo, M.Si. selaku pembimbing akademik penulis selama 4,5 tahun penulis menyelesaikan masa studi di Departemen Matematika ini. Terima kasih atas segala bimbingan dan nasihat-nasihat Ibu.

(3) Seluruh dosen Departemen Matematika FMIPAUI yang telah mengajarkan banyak ilmu kepada penulis.

(4) Seluruh karyawan di Departemen Matematika yang telah membantu dan memberikan kemudahan kepada penulis dalam berbagai keperluan.

(5) Mama dan Papa, sang pemberi semangat, sumber motivasi dan inspirasi. Terima kasih atas doa-doa dan semangatnya.

(6) Muhammad Maulana Abdullah, kakak penulis yang telah mendahului penulis menjadi sarjana. Terima kasih atas pemikiran, semangat, dan kerja kerasnya yang juga selalu menginspirasi.

(7) Keluarga Matematika UI 2008: Uni Aci, Ade, Adhi, Agnes, Agy, Bang Andy, Awe, Anggi, Anisah, Arief, Arkies, Arman, Aya, Bowo, Cindy, Citra, Danis, Dede, Dhea, Dheni, Dhila, Dian, Dini, Ega, Eka, Emy, Juni, Fani, Ko Hen, Ijut, Ica, Ifah, Ines, Janu, Lian, Luthfa, Maimun, Masykur, Maul, May TA, Mei, Mela, Nadia, Nita, Nora, Numa, Oline, Puput, Purwo, Ramos, Resti,

Risya, Mami Sita, Mba Siwi, PuJul, Rizkie, Tute, Uchi, Uci, Umbu, Vika, Wulan, Yulial, dan Zee. Terima kasih untuk semua kebersamaan dan kenangannya, juga untuk momen-momen senang dan sedih bersama. Semoga kita bisa terus berkumpul bersama.

(8) Nita Astuti yang sudah menjadi teman diskusi dan teman seperjuangan bagi penulis selama pembuatan tugas akhir ini, tak bosan-bosan memberikan semangat, solusi, dan saran sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

(9) Bang Andy yang sudah sangat banyak membantu penulis dalam menyelesaikan tugas akhir ini, menjadi mentor dalam pembuatan program, menjadi teman diskusi, hingga membantu memeriksa teknis penulisan.

(10) Ka Yanu, yang merelakan waktunya untuk membantu penulis dalam menyelesaikan program hingga diperoleh program yang sesuai dengan kebutuhan tugas akhir penulis.

(11) Widya Rhesi yang telah menjadi teman bagi penulis di berbagai suasana.

(12) Kakak-kakak angkatan 2004, 2005, 2006, dan 2007 yang telah memberikan semangat dan bantuan kepada penulis.

(13) Teman-teman dan adik-adik angkatan 2009, 2010, dan 2011 yang telah memberikan semangat, dukungan, dan doa kepada penulis.

(14) Keluarga Perwita: Mba Warti, Pak Le, Anis, Iqbal, Mei, Luthfa, Inah, Muti, Dhea, Dhipu, Bianca, Dewi, Mba Nurs, Widya, dan Mayda, yang telah banyak memberikan bantuan, dukungan, doa, dan tawa untuk penulis.

(15) Mba Nursih yang telah memperkenalkan penulis dengan "1 Night 2 Days" sehingga dapat menjadi hiburan bagi penulis selama masa penyelesaian tugas akhir ini.

(16) Semua pihak yang telah memberikan doa dan semangat kepada penulis, yang tidak bisa penulis sebutkan satu per satu.

Penulis berharap Allah berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga tugas akhir ini membawa manfaat bagi pengembangan ilmu.

Penulis

2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Laili Miftahur Rizqi
NPM	: 0806452204
Program Studi	: S1 Matematika
Departemen	: Matematika
Fakultas	: Fakultas Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul :
Pendekatan Algoritma *Auction* pada Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar dan Aplikasinya pada Penjadwalan Kendaraan Bus TransJakarta

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 20 November 2012

Yang menyatakan

(Laili Miftahur Rizqi)

ABSTRAK

Nama : Laili Miftahur Rizqi
Program Studi : Matematika
Judul : Pendekatan Algoritma *Auction* pada Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar dan Aplikasinya pada Penjadwalan Kendaraan Bus TransJakarta

Penjadwalan kendaraan (*vehicle scheduling*) merupakan proses pengaturan kendaraan terhadap himpunan perjalanan (*trip*) yang berasal dari jadwal keberangkatan (*timetable*) sedemikian sehingga meminimumkan biaya operasional. *Trip* merupakan perpindahan kendaraan dengan penumpang dari lokasi awal yang spesifik ke lokasi akhir yang spesifik pada waktu keberangkatan dan waktu kedatangan yang juga spesifik. Pada dasarnya, penjadwalan kendaraan telah mencakup jadwal pengisian bahan bakar. Akan tetapi pada pengoperasian bus TransJakarta terdapat hal yang perlu diperhatikan, yaitu stasiun pengisian bahan bakar gas yang jumlahnya hanya sedikit. Akibatnya bus hanya dapat mengisi bahan bakar di suatu lokasi tertentu. Selain itu, ketika bus akan mengisi bahan bakar, bus tersebut harus dalam kondisi tidak membawa penumpang. Oleh karena itu, penjadwalan kendaraan bus TransJakarta harus memenuhi aspek-aspek berikut:

- *Timetable* bus terpenuhi dengan memperhatikan jadwal pengisian bahan bakar.
- Biaya operasional yang dikeluarkan Unit Pengelola TransJakarta Busway minimum.

Masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar akan dimodelkan sebagai masalah *quasi-assignment*. Selanjutnya masalah tersebut akan diselesaikan menggunakan algoritma *auction* dan diaplikasikan pada masalah penjadwalan kendaraan bus TransJakarta. Keluaran dari masalah penjadwalan kendaraan pada tugas akhir ini ialah barisan perjalanan dan *deadhead* yang dijalankan oleh setiap kendaraan bus TransJakarta dan kapan kendaraan tersebut mengisi bahan bakar.

Kata Kunci : penjadwalan kendaraan bus *rapid transit*, penjadwalan kendaraan bus TransJakarta, jadwal pengisian bahan bakar, algoritma *auction*
xiv + 100 halaman : 7 gambar; 2 tabel
Daftar Pustaka : 9 (1980-2012)

ABSTRACT

Name : Laili Miftahur Rizqi
Study Program : Mathematics
Title : Auction Algorithm Approach for Rapid Transit Bus Vehicle Scheduling by Considering Fuel Filling Schedule and Its Application to TransJakarta Bus Vehicle Scheduling

Vehicle scheduling is the proses of assigning vehicle to a set of trips from predetermined departure schedule (timetable) in order to minimize operational cost. Trip is the movement of vehicle together with passengers from specified start location to specified end location at a specified departure time and arrival time. Basically, vehicle scheduling already includes fuel filling schedule. But in operating TransJakarta bus, there is one thing needed to be paid attention to, that is the small number of gas stations available for fuel filling. As a consequence, the bus can only fill up the gas tank at certain locations. Besides that, when a bus is going to fill up the gas tank, the bus should be in a condition where it contains no passenger. Because of that, TransJakarta bus vehicle scheduling must fulfill these aspects:

- The buses' timetable must be fulfilled by considering the fuel filling schedule.
- Operational cost spent by Unit Pengelola TransJakarta Busway is minimum.

Rapid transit vehicle scheduling problem by considering fuel filling schedule will be modeled as a quasi-assignment problem. The problem will be solved using auction algorithm and be applied to TransJakarta bus vehicle scheduling problem. Output from vehicle scheduling problem in this *skripsi* are sequences of trips and deadheads which will be executed by each TransJakarta bus and when the vehicle fill up the gas tank.

Keywords : rapid transit vehicle scheduling, TransJakarta bus vehicle scheduling, fuel filling schedule, auction algorithm
xiv + 100 pages : 7 pictures; 2 tables
Bibliography : 9 (1980-2012)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Salah satu masalah transportasi yang ada di DKI Jakarta adalah kemacetan. Penyebabnya antara lain pertumbuhan jumlah kendaraan yang tidak sebanding dengan pertumbuhan panjang jalan. Berdasarkan data Dishub DKI Jakarta, hingga akhir tahun 2010, pertumbuhan rata-rata jumlah kendaraan bermotor di Jakarta dalam lima tahun terakhir sebesar 9,5% per tahun. Sementara itu, pertumbuhan panjang jalan hanya sekitar 0,01% per tahun. Hal ini menyebabkan ruas jalan yang ada tidak cukup untuk menampung kendaraan yang berlalu lintas (Tarjuki, M. T. dan Nurachman, 2012).

Pertumbuhan kendaraan yang telah dipaparkan di atas didominasi oleh pertambahan kendaraan pribadi, yaitu sekitar 441 mobil baru dan 1.311 motor baru setiap harinya. Hal ini disebabkan oleh kurang layaknya pelayanan transportasi umum yang ada sehingga pertumbuhan kendaraan pribadi semakin melesat (Tarjuki, M. T. dan Nurachman, 2012). Untuk mengatasi masalah tersebut, Pemerintah Provinsi DKI Jakarta berusaha mengadakan sistem transportasi yang lebih nyaman dan tertib dengan mengelola layanan transportasi umum massal, salah satunya adalah bus TransJakarta. Bus TransJakarta merupakan bus yang menggunakan sistem bus *rapid transit*, yaitu sistem transportasi umum yang menggunakan kendaraan berkapasitas relatif besar, cepat, nyaman, aman, tepat waktu, dan memiliki jalur terpisah dengan kendaraan lainnya. Sistem pengoperasian bus TransJakarta mengadaptasi sistem TransMilenio yang sukses di Kolombia. Tujuan pembangunan sarana ini ialah meningkatkan pelayanan dan penyediaan jasa transportasi yang aman, terpadu, tertib, lancar, nyaman, ekonomis, efisien, efektif, dan terjangkau oleh masyarakat (Akbar, M., 2012).

Saat ini masih banyak kendala yang dihadapi oleh Unit Pengelola TransJakarta Busway dalam mengelola bus TransJakarta, salah satunya adalah

penjadwalan kendaraan. Penjadwalan kendaraan (*vehicle scheduling*) adalah proses pengaturan kendaraan terhadap himpunan perjalanan (*trip*), yang berasal dari jadwal keberangkatan (*timetable*), sehingga diperoleh biaya operasional yang minimum (Freling, Wagelmans, dan Paixão, 2001).

Pada dasarnya, penjadwalan kendaraan telah mencakup jadwal pengisian bahan bakar dan secara umum, ketika akan mengisi bahan bakar, kendaraan diperbolehkan dalam kondisi sedang membawa penumpang. Akan tetapi, pada pengoperasian bus TransJakarta terdapat beberapa hal khusus yang perlu diperhatikan, diantaranya adalah terbatasnya jumlah stasiun pengisian bahan bakar, akibatnya bus hanya dapat mengisi bahan bakar di lokasi tertentu. Selain itu, ketika bus akan mengisi bahan bakar, bus tersebut harus dalam kondisi tidak membawa penumpang. Oleh karena itu penjadwalan kendaraan bus TransJakarta harus memenuhi aspek-aspek berikut:

- Jadwal keberangkatan (*timetable*) bus terpenuhi dengan memperhatikan jadwal pengisian bahan bakar.
- Biaya operasional yang dikeluarkan oleh Unit Pengelola TransJakarta Busway minimum.

1.2 Perumusan Masalah dan Ruang Lingkup Masalah

Perumusan masalah yang diajukan pada tugas akhir ini adalah:

- Bagaimana memodifikasi model matematis untuk masalah penjadwalan kendaraan secara umum sehingga sesuai untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar sehingga diperoleh biaya operasional minimum.
- Bagaimana menyelesaikan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction*.
- Bagaimana mengaplikasikan algoritma *auction* untuk menyelesaikan masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar menggunakan bantuan perangkat lunak MATLAB R2009.

Ruang lingkup pembahasan masalah pada tugas akhir ini adalah:

- Hanya digunakan satu *depot*, yaitu tempat parkir bagi kendaraan yang sedang tidak digunakan untuk sementara, sehingga setiap kendaraan akan berangkat dari *depot* yang sama.
- Kendaraan diperbolehkan melakukan *interlining*, yaitu pergantian dari rute yang satu menjadi rute yang lainnya.
- Kendaraan yang digunakan homogen, yaitu bus berkapasitas 85 orang.
- Pada penjadwalan kendaraan bus TransJakarta, hanya digunakan dua rute, yaitu rute Dukuh Atas – Pulogadung dan rute Dukuh Atas – Ragunan.

1.3 Jenis Penelitian dan Metode yang Digunakan

Jenis penelitian yang digunakan dalam pembuatan tugas akhir ini adalah studi literatur. Kemudian akan dilakukan modifikasi terhadap model penjadwalan kendaraan secara umum sehingga sesuai untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Metode yang digunakan untuk menyelesaikan masalah tersebut dan aplikasinya pada penjadwalan kendaraan bus TransJakarta adalah algoritma *auction*. Selanjutnya untuk memperoleh solusi dari masalah penjadwalan tersebut digunakan bantuan perangkat lunak MATLAB R2009.

1.4 Tujuan Penulisan

Tujuan penulisan tugas akhir ini adalah:

- memodifikasi model matematis untuk masalah penjadwalan kendaraan secara umum sehingga sesuai untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar,
- menyelesaikan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction*,
- mengaplikasikan algoritma *auction* untuk menyelesaikan masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal

pengisian bahan bakar menggunakan bantuan perangkat lunak MATLAB R2009.

BAB 2
LANDASAN TEORI

Beberapa dasar teori yang akan digunakan dalam menyelesaikan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar akan dibahas pada bab 2 ini. Dasar-dasar teori tersebut meliputi konsep pemrograman linear (*linear programming*) yang akan dibahas pada Subbab 2.1, teori dalam jaringan (*network*) yang akan dibahas pada Subbab 2.2, masalah penugasan dan model *quasi-assignment* pada Subbab 2.6 dan 2.7, serta algoritma *auction* pada Subbab 2.8. Sementara itu dasar-dasar perencanaan transportasi untuk transportasi umum akan dibahas pada Subbab 2.3 – 2.5.

2.1 Pemrograman Linear (*Linear Programming*)

Masalah penjadwalan kendaraan merupakan salah satu contoh dari masalah pemrograman linear (*linear programming*). Konsep *linear programming*, yang selanjutnya disingkat menjadi LP, pertama kali diperkenalkan oleh George B. Dantzig pada tahun 1947. Masalah LP dapat didefinisikan sebagai masalah mengoptimalkan (memaksimumkan atau meminimumkan) suatu fungsi linear terhadap sejumlah berhingga kendala linear, dimana fungsi linear yang dioptimalkan tersebut dinamakan fungsi objektif (Hillier dan Lieberman, 1980).

Secara matematis, suatu masalah LP adalah suatu masalah untuk menemukan nilai-nilai $x_1, x_2, \dots, x_n \geq 0$ dan meminimalkan fungsi z terhadap fungsi-fungsi kendala. Maka masalah LP didefinisikan sebagai berikut:

$$\begin{aligned} (\min)\ z = c_1x_1 &+ c_2x_2 + \dots + c_nx_n \\ a_{11}x_1 &+ a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 &+ a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\ \vdots \quad &\quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{m1}x_1 &+ a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m, \end{aligned} \tag{2.1}$$

atau dalam notasi matriks-vektor, model (2.1) yang merupakan bentuk umum masalah LP, dapat dinyatakan sebagai berikut:

$$\text{min} \qquad z = \mathbf{c}^T\mathbf{x} \tag{2.2}$$

$$\text{dengan syarat} \qquad \mathbf{Ax} \geq \mathbf{b}, \tag{2.3}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{2.4}$$

dimana $\mathbf{c}^T\mathbf{x}$ adalah fungsi tujuan atau fungsi objektif, yaitu fungsi linear yang akan diminimalkan, dengan $\mathbf{x} \in \mathbb{R}^n$ merepresentasikan vektor dari variabel keputusan dan $\mathbf{c} \in \mathbb{R}^n$ adalah vektor biaya. $\mathbf{Ax} \geq \mathbf{b}$ merupakan kendala fungsional dengan $\mathbf{A} \in \mathbf{M}(m, n)$ dan $\mathbf{b} \in \mathbb{R}^m$ sedangkan $\mathbf{x} \geq \mathbf{0}$ merupakan kendala nonnegatif. Nilai-nilai $x_1, x_2, \ldots, x_n$ yang memenuhi semua kendala (2.3) dan (2.4) disebut sebagai solusi layak, sementara solusi layak yang memberikan nilai minimum terhadap fungsi objektif disebut sebagai solusi optimal (Hillier dan Lieberman, 1980).

Menurut Hillier dan Lieberman (1980), solusi optimal untuk setiap masalah LP terletak pada batas dari kendala (*constraint boundaries*), yaitu ketika kendala berupa persamaan. Oleh karena itu, masalah LP dengan bentuk umum (2.2) – (2.4) dapat diselesaikan dengan terlebih dahulu diubah menjadi bentuk baku berikut:

$$\text{min} \qquad z = \mathbf{c}^T\mathbf{x} \tag{2.5}$$

$$\text{dengan syarat} \qquad \mathbf{Ax} = \mathbf{b}, \tag{2.6}$$

$$\mathbf{x} \geq \mathbf{0}. \tag{2.7}$$

Kendala persamaan pada masalah LP (2.5) – (2.7) diperoleh dengan cara sebagai berikut:

- Jika kendala fungsional berupa pertidaksamaan $\leq$, kendala tersebut diubah menjadi bentuk persamaan dengan menambahkan variabel *slack* $\mathbf{s} \in \mathbb{R}^m$ pada ruas kiri dengan $\mathbf{s} \geq \mathbf{0}$. Kemudian pada fungsi tujuan ditambahkan $\mathbf{0}^T\mathbf{s}$.
- Jika kendala fungsional berupa pertidaksamaan $\geq$, kendala tersebut diubah menjadi bentuk persamaan dengan menambahkan variabel *surplus* $-\mathbf{e} \in \mathbb{R}^m$ pada ruas kiri dengan $\mathbf{e} \geq \mathbf{0}$. Kemudian pada fungsi tujuan ditambahkan $\mathbf{0}^T\mathbf{e}$.

Pada subbab ini, ada beberapa pembahasan lebih lanjut mengenai LP, salah satunya mengenai dualitas dari suatu masalah LP yang akan dibahas pada subbab berikut.

2.1.1 Dualitas Suatu Masalah LP

Setiap masalah LP dengan bentuk umum (2.2) – (2.4), yang juga disebut sebagai masalah primal, mempunyai masalah dual dengan bentuk sebagai berikut:

maks $v = \mathbf{b}^T\mathbf{y}$ (2.8)

dengan syarat $\mathbf{A}^T\mathbf{y} \leq \mathbf{c}$, (2.9)

$\mathbf{y} \geq \mathbf{0}$ dan $\mathbf{y} \in \mathbb{R}^m$ (2.10)

(Hillier dan Lieberman, 1980). Berikut ini beberapa teorema yang menjelaskan hubungan antara masalah primal LP dengan masalah dualnya:

Teorema 2.1 (Hillier dan Lieberman, 1980). Jika $\mathbf{x} \in \mathbb{R}^n$ adalah sembarang solusi layak untuk suatu masalah primal yang meminimumkan dan $\mathbf{y} \in \mathbb{R}^m$ adalah sembarang solusi layak untuk masalah dualnya maka $\mathbf{c}^T\mathbf{x} \geq \mathbf{b}^T\mathbf{y}$.

Teorema 2.2 (Hillier dan Lieberman, 1980). $\mathbf{x}^* \in \mathbb{R}^n$ adalah solusi optimal untuk suatu masalah primal LP dan $\mathbf{y}^* \in \mathbb{R}^m$ adalah solusi optimal untuk masalah dualnya jika dan hanya jika maka $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$.

Teorema 2.3 (Hillier dan Lieberman, 1980). Jika suatu masalah primal LP tidak terbatas maka masalah dualnya tidak layak.

Untuk masalah LP dengan variabel keputusan berupa bilangan bulat, masalah tersebut dinamakan pemrograman linear bilangan bulat (*integer linear programming*) yang dibahas dalam subbab berikut.

2.1.2 Pemrograman Linear Bilangan Bulat (*Integer Linear Programming*)

Masalah pemrograman bilangan bulat (*integer linear programming*), yang selanjutnya disingkat menjadi ILP, merupakan suatu bentuk masalah LP dimana terdapat tambahan kendala yaitu nilai dari variabel keputusan harus berupa bilangan bulat. Secara matematis, bentuk umum dari ILP adalah:

$$\text{min} \quad z = \mathbf{c}^T\mathbf{x} \tag{2.11}$$

$$\text{dengan syarat} \quad \mathbf{Ax} \geq \mathbf{b}, \tag{2.12}$$

$$\mathbf{x} \geq \mathbf{0} \text{ dan } \mathbf{x} \in \mathbb{Z}^n \tag{2.13}$$

(Hillier dan Lieberman, 1980). Sementara itu, jika nilai variabel-variabel tersebut dibatasi menjadi 0 atau 1 maka masalah ini disebut masalah pemrograman linear biner (*binary linear programming*) dan mempunyai bentuk matematis berikut:

$$\text{min} \quad z = \mathbf{c}^T\mathbf{x} \tag{2.14}$$

$$\text{dengan syarat} \quad \mathbf{Ax} \geq \mathbf{b}, \tag{2.15}$$

$$\mathbf{x} \in \{0,1\}^n \tag{2.16}$$

(Hillier dan Lieberman, 1980). Masalah penjadwalan kendaraan bus *rapid transit* termasuk dalam masalah ILP, khususnya masalah pemrograman linear biner.

Dengan adanya kendala tambahan berupa variabel keputusan yang harus berupa bilangan bulat, masalah LP menjadi lebih sulit diselesaikan. Oleh karena itu, kendala $\mathbf{x} \in \mathbb{Z}^n$ direlaksasi menjadi kendala yang lebih longgar.

2.1.3 Relaksasi Pemrograman Linear (*Linear Programming Relaxation*)

Ide dari relaksasi ini adalah untuk mengganti masalah ILP dengan masalah optimisasi yang lebih mudah, yaitu dengan menghilangkan kendala bahwa variabel keputusan harus berupa bilangan bulat.

Definisi 2.4 (Wolsey, 1998). Suatu masalah ILP berikut:

$$\text{min} \quad z = \mathbf{c}^T\mathbf{x} \tag{2.17}$$

$$\text{dengan syarat} \quad \mathbf{Ax} \geq \mathbf{b}, \tag{2.18}$$

$$\mathbf{x} \geq \mathbf{0} \text{ dan } \mathbf{x} \in \mathbb{Z}^n \tag{2.19}$$

memiliki bentuk relaksasi seperti di bawah ini:

$$\text{min} \quad z^R = \mathbf{c}^T\mathbf{x} \tag{2.20}$$

$$\text{dengan syarat} \quad \mathbf{Ax} \geq \mathbf{b}, \tag{2.21}$$

$$\mathbf{x} \geq \mathbf{0} \text{ dan } \mathbf{x} \in \mathbb{R}^n. \tag{2.22}$$

Teorema berikut ini menunjukkan hubungan antara masalah LP yang telah direlaksasi dengan keoptimalan masalah LP tersebut.

Teorema 2.5 (Wolsey, 1998).

- Jika suatu masalah relaksasi merupakan masalah yang tidak layak maka masalah awalnya (ILP) juga tidak layak.
- Misalkan $\mathbf{x}^*$ adalah solusi optimal untuk suatu masalah relaksasi dari masalah ILP. Jika $z^R = z$ maka $\mathbf{x}^*$ adalah solusi optimal untuk masalah ILP.

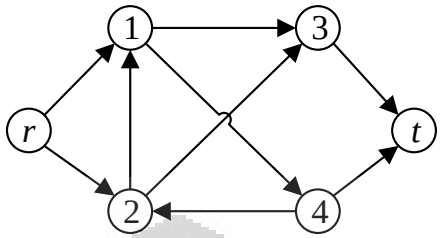
Untuk membuat model matematis dari masalah penjadwalan kendaraan, biasanya masalah tersebut terlebih dahulu direpresentasikan dalam bentuk jaringan.

2.2 Beberapa Istilah dalam Jaringan (*Network*)

Pada subbab ini akan dijelaskan mengenai beberapa istilah dari graf dan jaringan yang diperlukan dalam tugas akhir ini. Suatu graf terdiri dari suatu himpunan berhingga titik, yang disebut juga sebagai himpunan simpul, dan suatu himpunan pasangan simpul, yang disebut juga sebagai himpunan busur. Busur yang memiliki arah disebut busur berarah dan busur yang tidak memiliki arah disebut busur tak berarah. Misalkan terdapat busur (i, j), dimana i dan j keduanya merupakan elemen himpunan simpul. Jika busur tersebut merupakan busur berarah maka busur $(i, j) \neq$ busur (i, j). Jika busur (i, j) merupakan busur tidak berarah maka busur (i, j) = busur (i, j). Suatu jaringan adalah suatu graf yang memiliki arus (*flow*) pada busurnya. Suatu rantai (*chain*) antara simpul i dan simpul j adalah suatu barisan busur yang menghubungkan kedua simpul ini. Suatu lintasan (*path*) adalah barisan simpul di sepanjang rantai yang memiliki awal dan akhir yang spesifik. Suatu siklus (*cycle*) adalah suatu rantai yang berawal dan berakhir pada simpul yang sama (Hillier dan Lieberman, 1980).

Berikut ini diberikan gambaran mengenai istilah-istilah mengenai graf yang telah dijelaskan sebelumnya. Misalkan suatu graf G memiliki 6 simpul, yaitu simpul r, 1, 2, 3, 4, dan t. Misalkan busur-busur yang dimiliki oleh graf G ialah

busur berarah (*r*, 1), (*r*, 2), (1, 3), (1, 4), (2, 1), (2, 3), (3, *t*), (4, 2), dan (4, *t*). Maka graf G dapat digambarkan seperti di bawah ini:

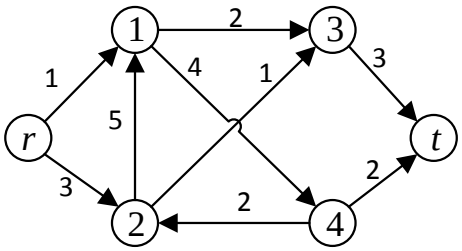


Gambar 2.1 Graf *G*

Contoh rantai dari gambar di atas ialah (1, 3) – (3, *t*). Rantai tersebut merupakan rantai yang menghubungkan simpul 1 dengan simpul *t*. Selanjutnya misalkan akan dicari lintasan dari simpul 1 ke simpul *t*. Maka lintasan dari simpul 1 ke simpul *t* ialah 1 – 3 – *t*. Contoh siklus dari graf *G* di atas ialah (1, 4) – (4, 2) – (2, 1).

Suatu jaringan dikatakan siklis jika jaringan tersebut memiliki *cycle*. Selanjutnya suatu jaringan dikatakan berarah jika setiap busurnya memiliki arah sedemikian sehingga salah satu simpul dianggap sebagai simpul asal dan simpul lainnya dianggap sebagai simpul tujuan. Kapasitas arus dari suatu busur adalah batas atas besarnya arus yang dapat melalui busur tersebut. Suatu simpul dalam jaringan disebut *source* jika simpul tersebut merupakan simpul asal bagi setiap busur yang menghubungkan simpul tersebut dengan simpul lain. Suatu simpul disebut *sink* jika simpul tersebut merupakan simpul tujuan bagi setiap busur yang menghubungkan simpul tersebut dengan simpul lain (Hillier dan Lieberman, 1980).

Sebelumnya telah dijelaskan bahwa jaringan merupakan graf yang memiliki *flow*. Misalkan graf *G* pada Gambar 2.1 diberikan suatu *flow* di setiap busurnya. Pada suatu jaringan, total *flow* yang masuk ke suatu simpul harus sama dengan total *flow* yang keluar dari simpul tersebut. Selanjutnya misalkan graf *G* yang telah diberikan *flow* tersebut dinamakan jaringan G_1. Maka berikut ini contoh jaringan G_1 beserta *flow*-nya:



Gambar 2.2 Jaringan G_1

Jaringan G_1 memiliki siklus yaitu (1, 4) – (4,2) – (2, 1) sehingga G_1 disebut jaringan siklis. Selanjutnya karena setiap busur pada jaringan G_1 memiliki arah, maka G_1 disebut juga jaringan berarah. Simpul *r* memiliki busur (*r*, 1) dan (*r*, 2). Setiap busur tersebut menjadikan simpul *r* sebagai simpul asalnya. Oleh karena itu, simpul *r* disebut juga sebagai *source*. Simpul *t* memiliki busur (3, *t*) dan (4, *t*). Setiap busur tersebut menjadikan simpul *t* sebagai simpul tujuannya. Oleh karena itu, simpul *t* disebut juga sebagai *sink*.

Sebelum dibahas mengenai model matematis yang digunakan dalam membuat penjadwalan kendaraan, terlebih dahulu dipaparkan mengenai dasar-dasar teori transportasi yang mencakup istilah-istilah yang biasa digunakan dalam teori transportasi.

2.3 Dasar-Dasar Perencanaan Transportasi untuk Transportasi umum

Pada suatu perusahaan transportasi umum diperlukan suatu proses perencanaan pengoperasian transportasi umum yang baik, agar diperoleh keuntungan, baik bagi pemilik perusahaan maupun bagi para pelanggan jasa transportasi umum tersebut. Sebelum dijelaskan mengenai proses perencanaan tersebut, terlebih dahulu dipaparkan beberapa istilah dalam transportasi yang digunakan dalam tugas akhir ini:

a) Rute : jalan yang menghubungkan satu lokasi dengan lokasi lainnya

b) Frekuensi : menunjukkan berapa banyak kendaraan melewati suatu rute per satuan waktu (biasanya per jam)

c) *Headway* : selang waktu kedatangan dua kendaraan pada suatu lokasi

d) *Interlining* : pergantian rute yang dilakukan oleh suatu kendaraan

e) *Timetable* : tabel yang berisi daftar waktu kapan satu kendaraan berangkat dari lokasi yang spesifik dan tiba di lokasi yang spesifik

f) *Depot* : tempat parkir kendaraan ketika sedang tidak digunakan untuk sementara waktu

g) *Trip* : perpindahan kendaraan dengan penumpang antara lokasi keberangkatan yang spesifik ke lokasi kedatangan yang spesifik pada waktu keberangkatan dan waktu kedatangan yang spesifik

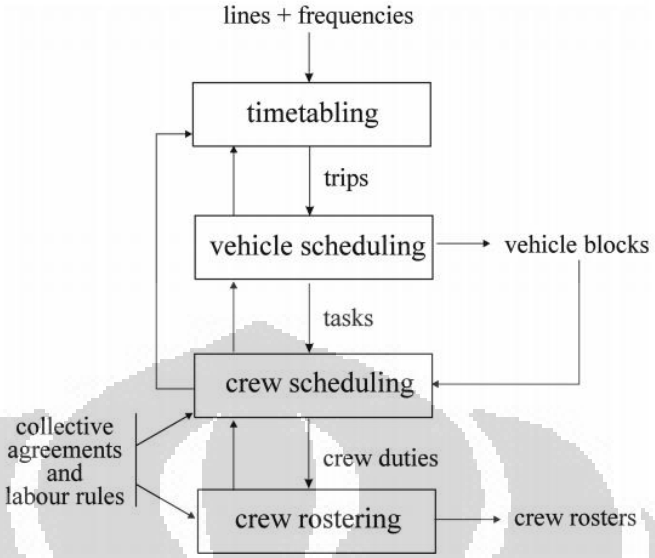
h) *Deadhead* : perpindahan kendaraan tanpa penumpang antara lokasi keberangkatan yang spesifik ke lokasi kedatangan yang spesifik pada waktu keberangkatan dan waktu kedatangan yang spesifik

i) *Compatible trip* : *trip* yang dapat dijalankan oleh suatu kendaraan setelah kendaraan tersebut selesai melakukan satu *trip* yang lain

j) *Vehicle block* : barisan *trip* dan *deadhead* yang dimulai dari dan berakhir di *depot*, yang dapat dieksekusi oleh satu kendaraan

(Huisman, 2004)

Berikut ini adalah bagan proses perencanaan pengoperasian transportasi umum menurut Huisman (2004):



Gambar 2.3 Proses Perencanaan Sistem Transportasi Umum
(sumber: Huisman, 2004)

Selanjutnya akan dipaparkan mengenai proses perencanaan transportasi umum pada Gambar 2.3 yang disertai dengan penjelasan dari beberapa istilah transportasi yang digunakan. Proses perencanaan pengoperasian transportasi umum dimulai dari penentuan rute (*lines*) yang akan dilalui suatu kendaraan dan juga penentuan frekuensi, yaitu berapa banyak kendaraan yang melewati rute yang telah ditentukan dalam satuan waktu (biasanya per jam). Kemudian dari rute dan frekuensi dapat dibuat jadwal keberangkatan (*timetable*). Misalkan untuk rute A – B dan rute B – C diketahui *timetable* sebagai berikut:

Tabel 2.1 Contoh *Timetable*

Lokasi Keberangkatan	**Lokasi Kedatangan**	**Waktu Keberangkatan**	**Waktu Kedatangan**
A	B	08.00	08.40
A	B	08.10	08.50
B	C	08.45	09.30
B	A	08.55	09.35

Kolom-kolom *timetable* terdiri dari lokasi keberangkatan, lokasi kedatangan, waktu keberangkatan kendaraan dari lokasi keberangkatan, dan waktu kedatangan kendaraan di lokasi kedatangan.

Berdasarkan Gambar 2.3, dari *timetable* dapat diperoleh *trip* dan berdasarkan definisi, setiap *trip* mengandung informasi berupa perpindahan kendaraan dari lokasi keberangkatan dan kedatangan yang spesifik pada waktu keberangkatan dan kedatangan yang juga spesifik. Oleh karena itu, setiap baris dari *timetable* pada Tabel 2.1 dapat disebut *trip* dan *trip* biasanya diurutkan berdasarkan waktu keberangkatan yang meningkat. Dari baris pertama Tabel 2.1 diperoleh *trip* pertama, yaitu jadwal perpindahan kendaraan dari lokasi keberangkatan A pada pukul 08.00 ke lokasi kedatangan B pada pukul 08.40. Secara keseluruhan, dari Tabel 2.1 diperoleh 4 *trip*. Penjelasan mengenai *deadhead* serupa dengan penjelasan mengenai *trip*, hanya saja pada *deadhead*, kendaraan berpindah dengan tidak membawa penumpang. Contoh *deadhead* dari Tabel 2.1 ialah perpindahan kendaraan dari lokasi B pada pukul 08.40 dan tiba di lokasi yang sama, yaitu B pada pukul 08.45.

Trip yang telah diperoleh dari *timetable* selanjutnya menjadi masukan untuk penjadwalan kendaraan. Penjadwalan kendaraan ialah proses pengaturan kendaraan terhadap himpunan perjalanan (*trip*), yang berasal dari jadwal keberangkatan (*timetable*), sehingga diperoleh biaya operasional yang minimum (Freling, Wagelmans, dan Paixão, 2001). Keluaran dari penjadwalan kendaraan ini ialah *vehicle blocks*, yaitu barisan *trip* dan *deadhead* yang dimulai dari dan berakhir di *depot*, yang dapat dieksekusi oleh satu kendaraan (Huisman, 2004). Misalkan dari penjadwalan kendaraan diperoleh bahwa *trip* 1 dan *trip* 3 dieksekusi oleh satu kendaraan, sedangkan *trip* 2 dan *trip* 4 dieksekusi oleh satu kendaraan lainnya. Maka *vehicle block* yang pertama ialah *depot* – *trip* 1 – *trip* 3 – *depot* dan *vehicle block* yang ke dua ialah *depot* – *trip* 2 – *trip* 4 – *depot*. Selanjutnya dari *vehicle block* pertama dapat dilihat bahwa setelah kendaraan melakukan *trip* 1 yang berawal di lokasi A dan berakhir di lokasi B, kendaraan tersebut melakukan *trip* 3 yang berawal di lokasi B dan berakhir di lokasi C. Karena kendaraan tersebut berganti rute dari A – B menjadi rute B – C, kendaraan tersebut dikatakan melakukan *interlining*.

Berikut ini akan dijelaskan perbedaan antara penjadwalan kendaraan (*vehicle scheduling*) dengan pembuatan rute kendaraan (*vehicle routing*). Pada penjadwalan kendaraan, yang menjadi masukan ialah *timetable* dan yang menjadi

keluaran ialah *vehicle blocks*, sedangkan pada pembuatan rute kendaraan, justru *timetable* yang menjadi keluaran dengan rute dan frekuensi sebagai masukan.

Sesuai dengan Gambar 2.1, proses perencanaan transportasi umum dapat dilakukan hingga penjadwalan pengemudi (*crew scheduling*), yaitu pengaturan pengemudi terhadap kendaraan. *Crew scheduling* merupakan pengaturan pengemudi terhadap kendaraan dalam satu hari sedangkan *crew scheduling* untuk jangka panjang, biasanya satu tahun atau setengah tahun, dinamakan *crew rostering*. Namun pada tugas akhir ini hanya akan dibahas penjadwalan kendaraan.

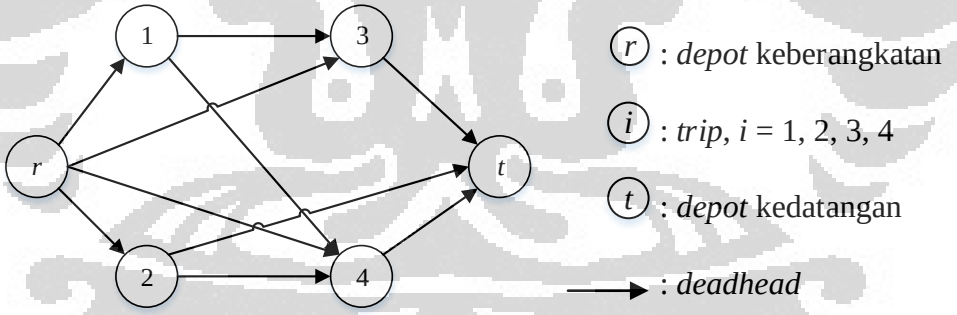
2.4 Representasi Masalah Penjadwalan Kendaraan dalam Bentuk Jaringan (*Network*)

Langkah awal dalam membuat representasi masalah penjadwalan dalam bentuk suatu jaringan $G(V, A)$ ialah dengan menentukan himpunan simpul V dan himpunan busur A dari G. Pada Subbab 2.3 telah dijelaskan bahwa untuk membuat penjadwalan kendaraan, digunakan himpunan *trip* sebagai masukan. Himpunan *trip* ini diperoleh dari *timetable* yang sudah ada dan diurutkan berdasarkan waktu keberangkatan yang meningkat. Selanjutnya setiap *trip* direpresentasikan sebagai simpul dalam jaringan. Misalkan dari *timetable* diperoleh sebanyak *n trip* per hari. Maka simpul 1 hingga simpul *n* dalam jaringan merepresentasikan *trip* 1 hingga *trip n* secara berurutan. Misalkan N adalah himpunan *trip*, maka $N = \{1, 2, ..., n\}$. Kemudian karena kendaraan harus berangkat dari dan kembali ke *depot*, maka *depot* juga direpresentasikan sebagai simpul dalam jaringan. Misalkan simpul *r* dan *t* masing-masing adalah simpul yang merepresentasikan *depot* yang sama, hanya saja simpul *r* menunjukkan *depot* sebagai lokasi keberangkatan awal dan simpul *t* menunjukkan *depot* sebagai lokasi kedatangan akhir. Maka diperoleh himpunan simpul pada jaringan G ialah $V = N \cup \{r, t\}$.

Selanjutnya akan dibentuk himpunan busur A untuk jaringan G. Busur antara dua *trip* ada jika kedua *trip* tersebut *compatible*. *Trip i* dan *trip j* disebut *compatible* jika waktu kedatangan *trip i* ditambah waktu *deadhead* dari lokasi

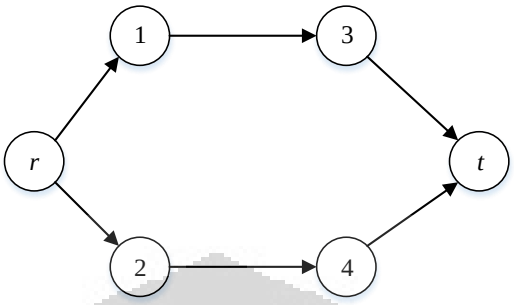
kedatangan *trip i* ke lokasi keberangkatan *trip j* kurang dari atau sama dengan waktu keberangkatan *trip j*. Dengan kata lain, *trip i* dan *trip j* dikatakan *compatible* jika $et_i + trav(i, j) \leq st_j$, dimana et_i adalah waktu kedatangan *trip i*, $trav(i, j)$ adalah waktu untuk melakukan *deadhead* dari lokasi kedatangan *trip i* ke lokasi keberangkatan *trip j*, dan st_j adalah waktu keberangkatan *trip j*. Misalkan E adalah himpunan busur antar-*trip* dimana $E = \{(i, j) \mid i, j \text{ compatible}, i, j \in N\}$. Kemudian diasumsikan dari *depot* kendaraan dapat melakukan *trip* manapun. Maka busur dari *depot* ke setiap *trip* atau $\{r\} \times N$ ada. Setelah selesai melakukan suatu *trip*, kendaraan juga dimungkinkan untuk kembali ke *depot* sehingga busur dari setiap *trip* ke *depot* atau $N \times \{t\}$ ada. Diperoleh himpunan busur untuk jaringan G yaitu $A = E \cup (\{r\} \times N) \cup (N \times \{t\})$.

Dari penjelasan di atas, masalah penjadwalan kendaraan dapat direpresentasikan sebagai jaringan $G(V, A)$ dengan $V = N \cup \{r, t\}$ dan $A = E \cup (\{r\} \times N) \cup (N \times \{t\})$. Berikut ini penggambaran masalah penjadwalan kendaraan dalam bentuk jaringan untuk contoh masalah pada Subbab 2.3, yaitu terdapat *depot* sebagai lokasi keberangkatan awal, 4 *trip*, dan *depot* sebagai lokasi kedatangan akhir, beserta dengan busur yang merupakan *deadhead*:



Gambar 2.4 Contoh Representasi Masalah Penjadwalan Kendaraan dalam Bentuk Jaringan

Berdasarkan keluaran yang juga dimisalkan pada Subbab 2.3, maka *vehicle blocks* yang diperoleh dapat digambarkan sebagai berikut:



Gambar 2.5 Contoh *Vehicle Blocks*

dengan *vehicle block* pertama merupakan *depot – trip* 1 – *trip* 3 – *depot* dan *vehicle block* ke dua merupakan *depot – trip* 2 – *trip* 4 – *depot*. Karena diperoleh 2 *vehicle block* sebagai keluaran, maka terdapat 2 barisan *trip* dan *deadhead*, yang bermula di *depot* dan kembali ke *depot*, yang harus dijalankan oleh 2 kendaraan berbeda. Oleh karena itu diketahui bahwa untuk menjalankan 4 *trip* yang diperoleh dari *timetable* pada Tabel 2.1 diperlukan 2 kendaraan.

2.5 Masalah Penjadwalan Kendaraan Satu *Depot* (Single-*Depot Vehicle Scheduling Problem*)

Penjadwalan kendaraan (*vehicle scheduling*) adalah suatu proses pengaturan kendaraan terhadap himpunan perjalanan (*trip*) yang telah ditentukan sedemikian sehingga diperoleh biaya operasional yang minimum (Freling, Wagelmans, dan Paixão, 2001). Jika dilihat dari banyaknya *depot* tempat kendaraan berasal, penjadwalan kendaraan dibagi menjadi 2 macam, yaitu penjadwalan kendaraan satu *depot* dan penjadwalan kendaraan lebih dari satu *depot*. Pada tugas akhir ini akan dibahas masalah penjadwalan kendaraan dengan satu *depot*.

Masalah penjadwalan kendaraan satu *depot* (*single-depot vehicle scheduling problem*), yang selanjutnya disingkat menjadi SDVSP, didefinisikan sebagai berikut:

Diberikan satu *depot* dan suatu himpunan *trip*. Akan dicari jadwal yang layak dengan biaya operasional yang minimum sedemikian sehingga:

a) Setiap *trip* dijalankan oleh tepat satu kendaraan,
b) Setiap kendaraan melakukan barisan *trip* yang layak beserta *deadhead* yang diperlukan.

SDVSP ini diketahui dapat diselesaikan dalam waktu polinomial (Freling, Wagelmans, dan Paixão, 2001).

SDVSP nantinya akan dimodelkan ke dalam masalah *quasi-assignment*. Masalah *quasi-assignment* merupakan salah satu bentuk khusus dari masalah penugasan linear (*linear assignment*). Pada subbab selanjutnya akan dijelaskan mengenai masalah penugasan linear secara umum dan masalah *quasi-assignment*.

2.6 Masalah Penugasan Linear (*Linear Assignment Problem*) dan Masalah *Quasi-Assignment*

Pada masalah penugasan linear (*linear assignment problem*), yang selanjutnya disingkat menjadi LAP, akan dicari penugasan (*assignment*) satu-satu yang lengkap dari n pekerja terhadap n tugas. Misalkan terdapat biaya c_{ij} yang dikeluarkan karena menugaskan pekerja i ke tugas j dan A adalah himpunan pasangan pekerja i dengan tugas j yang dapat berpasangan. Proses penugasan tersebut dilakukan sedemikian sehingga meminimumkan total biaya. Misalkan S adalah himpunan pasangan pekerja-tugas (i, j) sedemikian sehingga masing-masing pekerja i dan tugas j terdapat di paling banyak satu pasang di S. Jika jumlah pasangan di S ada sebanyak n, yang berarti bahwa setiap pekerja berpasangan dengan tugas yang berbeda, maka S dikatakan sebagai himpunan yang layak. Selanjutnya pada LAP akan dicari penugasan yang layak, yaitu himpunan pasangan pekerja-tugas $(1, j_1)$, $(2, j_2)$, …, (n, j_n) dari A dengan $j_1, j_2, \ldots, j_n$ yang semuanya merupakan tugas yang berbeda, sedemikian sehingga meminimumkan total biaya $\sum_{i=1}^{n} c_{ij_i}$.

Masalah *quasi-assignment* (*quasi-assignment problem*), yang selanjutnya disingkat menjadi QAP, merupakan bentuk khusus dari LAP. QAP didefinisikan sebagai masalah memasangkan m pekerja terhadap n tugas, dengan $m < n$, dimana terdapat suatu *sink* sedemikian sehingga setiap pekerja dapat berpasangan dengan

sink tersebut dan juga terdapat suatu *source* sedemikian sehingga setiap tugas dapat berpasangan dengan *source* tersebut. Oleh karena itu, pada QAP terdapat 3 jenis penugasan, yaitu:

- penugasan pekerja-tugas, yang didefinisikan dalam himpunan $S = \{(i, j) \mid i$ adalah pekerja, j adalah tugas, $(i, j) \in A\}$,
- penugasan *source*, yang didefinisikan dalam himpunan $D_r = \{j \mid j$ adalah tugas, j berpasangan dengan *source*$\}$,
- penugasan *sink*, yang didefinisikan dalam himpunan $D_t = \{i \mid i$ adalah pekerja, i berpasangan dengan *sink*$\}$,

dengan A merupakan himpunan semua pasangan pekerja dengan tugas yang mungkin berpasangan. Diketahui c_{ij} adalah biaya yang dikeluarkan karena memasangkan i dengan j. Selanjutnya pada QAP, akan dicari penugasan yang layak, yaitu himpunan pasangan (i, j), dimana setiap pekerja i berpasangan dengan tugas yang berbeda atau dengan *sink* dan setiap tugas j berpasangan dengan pekerja yang berbeda atau dengan *source*, sedemikian sehingga meminimumkan total biaya $\sum_{(i,j)\in A} c_{ij}$.

2.7 Model *Quasi-Assignment* untuk SDVSP

SDVSP (*Single-Depot Vehicle Scheduling Problem*) dapat dimodelkan sebagai model penugasan linear, model transportasi, model *minimum cost flow*, model *quasi-assignment*, atau model pencocokan (*matching model*). Pada tugas akhir ini, SDVSP akan dimodelkan ke dalam bentuk model *quasi-assignment*, dimana kendaraan berperan sebagai pekerja sedangkan *trip* berperan sebagai tugas. Seperti yang telah dijelaskan pada Subbab 2.4, ketika SDVSP direpresentasikan dalam bentuk jaringan, yang menjadi simpul ialah *trip* dan *depot*, dimana *source* adalah *depot* keberangkatan dan *sink* adalah *depot* kedatangan.

Berikut ini akan dijelaskan mengenai pembentukan model *quasi-assignment* untuk SDVSP. Misalkan $N = \{1, 2, ..., n\}$ adalah himpunan *trip* yang diurutkan berdasarkan waktu keberangkatan yang meningkat dan $E = \{(i, j) \mid i < j$ *compatible*, $i \in N, j \in N\}$ adalah himpunan busur yang menunjukkan *deadhead*

antara dua *trip*. Misalkan simpul *r* dan *t* adalah simpul untuk *depot*. Keduanya merepresentasikan *depot* yang sama, hanya saja simpul *r* menunjukkan *depot* sebagai lokasi keberangkatan awal dan simpul *t* menunjukkan *depot* sebagai lokasi kedatangan akhir.

Didefinisikan jaringan untuk suatu SDVSP, yaitu $G = (V, A)$ yang merupakan jaringan tak siklis berarah dengan himpunan simpul $V = N \cup \{r, t\}$ dan himpunan busur $A = E \cup (\{r\} \times N) \cup (N \times \{t\})$. Suatu *vehicle block* untuk suatu kendaraan dapat diperoleh dari suatu lintasan (*path*) dari *r* ke *t* pada jaringan. Satu lintasan dari *r* ke *t* tersebut merepresentasikan jadwal yang layak untuk satu kendaraan. Oleh karena itu, jadwal layak yang lengkap sedemikian sehingga menyelesaikan masalah penjadwalan kendaraan adalah lintasan-lintasan yang saling lepas (*disjoint*) dari *r* ke *t* sedemikian sehingga setiap simpul dalam *N* tercakup. Misalkan c_{ij} adalah biaya untuk busur $(i, j) \in A$. Jika *i* dan *j* masing-masing adalah *trip* maka c_{ij} adalah biaya untuk melakukan *deadhead* dari lokasi kedatangan *trip i* ke lokasi keberangkatan *trip j* dan biaya untuk melakukan *trip j*. Jika *i* adalah *depot* keberangkatan dan *j* adalah *trip* maka c_{ij} adalah biaya karena menggunakan satu kendaraan, melakukan *deadhead* dari lokasi kedatangan *trip i* ke lokasi keberangkatan *trip j*, dan melakukan *trip j*. Jika *i* adalah *trip* dan *j* adalah *depot* kedatangan maka c_{ij} adalah biaya untuk melakukan *deadhead* dari lokasi kedatangan *trip i* ke *depot* kedatangan. Selanjutnya y_{ij} merupakan variabel keputusan yang digunakan dengan $y_{ij} = 1$ jika suatu kendaraan melakukan *trip j* tepat setelah melakukan *trip i* dan $y_{ij} = 0$ jika tidak. SDVSP dapat diformulasikan sebagai berikut:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{2.23}$$

$$\text{d.s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{2.24}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{2.25}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \tag{2.26}$$

(Freling, Wagelmans, dan Paixão, 2001).

Fungsi tujuan (2.23) ialah fungsi yang akan diminimumkan, yaitu biaya total yang dikeluarkan untuk melakukan semua *trip*, melakukan *deadhead* yang diperlukan, dan juga menggunakan kendaraan. Berdasarkan Subbab 2.5, dalam SDVSP setiap *trip* harus dijalankan oleh tepat satu kendaraan. Selanjutnya karena kendaraan harus bermula dan berakhir di *depot* dan satu kendaraan mungkin dapat melakukan lebih dari satu *trip*, maka setiap *trip* yang dijalankan oleh kendaraan harus memiliki *predecessor* dan *successor*. Kendaraan harus memiliki *predecessor* maksudnya setiap *trip* yang dilakukan oleh kendaraan tersebut harus diketahui asalnya, apakah dari *depot* keberangkatan atau dari *trip* lainnya. Kendaraan harus memiliki *successor* maksudnya setelah suatu *trip* dilakukan, harus diketahui apakah kendaraan akan melakukan *trip* lain atau ke *depot* kedatangan.

Kendala (2.24) adalah kendala yang menjamin bahwa setiap *trip* memiliki tepat satu *predecessor* dan kandidat *predecessor trip* tersebut dapat berupa *depot* keberangkatan atau *trip* lainnya. Kendala (2.25) adalah kendala yang menjamin bahwa setiap *trip* memiliki tepat satu *successor* dan kandidat *successor trip* tersebut dapat berupa *trip* lainnya atau *depot* kedatangan. Kendala (2.24) dan (2.25) juga menjamin bahwa jaringan G dipartisi ke dalam suatu himpunan lintasan dari r ke t yang saling lepas. Barisan *trip* yang layak bagi suatu kendaraan nantinya dapat dibentuk dari masing-masing *predecessor* dan *successor* setiap *trip*. Kendala (2.26) menunjukkan bahwa variabel keputusan y_{ij} harus bernilai 0 atau 1 yang berarti model *quasi-assignment* ini merupakan salah satu masalah *binary integer linear programming*.

Berdasarkan jenis-jenis penugasan pada QAP yang telah dijelaskan pada subbab sebelumnya, maka pada SDVSP yang dimodelkan menjadi QAP juga terdapat 3 jenis penugasan, yaitu:

- penugasan *trip-trip*, yang didefinisikan dalam himpunan $S = \{(i, j) \mid i, j \in N, (i, j) \in E, i \text{ dan } j \text{ berpasangan}\}$,
- penugasan *depot* keberangkatan, yang didefinisikan dalam himpunan $D_r = \{j \mid j \in N, j \text{ berpasangan dengan } r\}$,
- penugasan *depot* kedatangan, yang didefinisikan dalam himpunan $D_t = \{i \mid i \in N, i \text{ berpasangan dengan } t\}$.

Pada tugas akhir ini, untuk menyelesaikan masalah primal (2.23) – (2.26), diperlukan juga bentuk dual dari masalah tersebut. Untuk memperoleh masalah dual dari masalah primal (2.23) – (2.26), perlu dibentuk variabel dual yang bersesuaian dengan setiap kendala, yaitu kendala (2.24) – (2.26). Sebelum membentuk variabel dual yang bersesuaian dengan setiap kendala pada masalah primal, akan dilakukan relaksasi terhadap kendala (2.26). Berdasarkan Subbab 2.1.3, masalah primal (2.23) – (2.26) memiliki bentuk masalah relaksasi sebagai berikut:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{2.27}$$

$$\text{d.s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{2.28}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{2.29}$$

$$y_{ij} \in \mathbb{R} \qquad \forall\, (i,j) \in A \tag{2.30}$$

Setelah SDVSP yang dimodelkan menjadi QAP direlaksasi menjadi masalah (2.27) – (2.30), selanjutnya akan dicari masalah dual untuk masalah (2.27) – (2.30) tersebut. Sebelum dicari variabel dual yang bersesuaian dengan setiap kendala dari masalah primal (2.27) – (2.30), masalah primal tersebut diubah menjadi masalah memaksimumkan yang ekuivalen, yaitu:

$$-\text{maks} \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{2.31}$$

$$\text{d.s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{2.32}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{2.33}$$

$$y_{ij} \in \mathbb{R} \qquad \forall\, (i,j) \in A \tag{2.34}$$

Misalkan $a_{ij} = -c_{ij}$ untuk setiap $(i, j) \in A$, dengan a_{ij} ialah keuntungan yang diperoleh karena memilih busur (i, j), maka masalah (2.31) – (2.34) menjadi:

$$\text{maks} \sum_{(i,j)\in A} a_{ij} y_{ij} \tag{2.35}$$

$$\text{d.s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{2.36}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{2.37}$$

$$y_{ij} \in \mathbb{R} \qquad \forall\, (i,j) \in A \tag{2.38}$$

Masalah (2.35) – (2.38) selanjutnya menjadi masalah primal dengan tujuan untuk memaksimumkan total keuntungan, yaitu maks $\sum_{(i,j)\in A} a_{ij} y_{ij}$. Kemudian untuk dapat menyelesaikannya menggunakan algoritma *auction*, akan dicari bentuk dual dari masalah primal tersebut.

Misalkan p_j adalah variabel dual yang berkorespondensi dengan kendala (2.36) untuk setiap $j \in N$ dan misalkan π_i adalah variabel dual yang berkorespondensi dengan kendala (2.37) untuk setiap $i \in N$. Maka berdasarkan dengan teori dualitas yang telah dibahas pada Subbab 2.1.1, diperoleh masalah dual dari masalah primal (2.35) – (2.38) seperti berikut:

$$\min \left(\sum_{i\in N} \pi_i + \sum_{j\in N} p_j \right) \tag{2.39}$$

$$\text{d.s. } \pi_i + p_j \geq a_{ij} \qquad \forall\, (i,j) \in E \tag{2.40}$$

$$\pi_i \geq a_{it} \qquad \forall\, i \in N \tag{2.41}$$

$$p_j \geq a_{rj} \qquad \forall\, j \in N \tag{2.42}$$

Selanjutnya masalah (2.39) – (2.42) menjadi masalah dual dari masalah primal SDVSP (2.35) – (2.38).

Algoritma *auction* yang akan digunakan untuk menyelesaikan SDVSP merupakan algoritma primal-dual. Algoritma ini akan menyelesaikan masalah primal (2.35) – (2.38) dan masalah dual (2.39) – (2.42) sekaligus. Penjelasan mengenai teori algoritma *auction* dan penggunaannya untuk menyelesaikan masalah primal dan dual dari SDVSP akan dipaparkan pada subbab berikut ini.

2.8 Algoritma *Auction*

Algoritma *auction* merupakan suatu algoritma primal-dual untuk menyelesaikan masalah *network flow*. Masalah *network flow* adalah masalah optimisasi yang direpresentasikan dalam bentuk jaringan (*network*), contohnya ialah masalah penjadwalan kendaraan. Algoritma primal-dual maksudnya algoritma ini menyelesaikan masalah primal dan dual sekaligus. Algoritma ini merupakan algoritma iteratif, dimana kedua biaya primal atau dualnya mungkin semakin memburuk di suatu iterasi, namun pada akhirnya akan diperoleh suatu solusi optimal. Terdapat 3 jenis algoritma *auction*, yaitu algoritma *forward auction*, algoritma *reverse auction*, dan algoritma kombinasi *forward auction* dengan *reverse auction*. Secara umum pada algoritma *forward auction*, pekerja akan menawar tugas dengan meningkatkan harga (*price*) tugas tersebut, sedangkan pada algoritma *reverse auction*, tugas berkompetisi untuk mendapatkan pekerja. Terlebih dahulu akan dijelaskan mengenai algoritma *auction* yang digunakan untuk menyelesaikan LAP.

2.8.1 Algoritma *Auction* untuk Menyelesaikan LAP

Algoritma *auction* untuk menyelesaikan LAP merupakan suatu konsep algoritma yang berdasarkan pada konsep lelang (*auction*). Konsep lelang ini digunakan karena masalah penugasan ekuivalen dengan masalah mencari keseimbangan (*equilibrium*) dalam bidang ekonomi. Misalkan akan dilakukan proses pemasangan n pekerja dengan n tugas melalui mekanisme pasar, yaitu melihat setiap pekerja sebagai suatu agen ekonomi yang akan memilih tugas yang memberikan keuntungan maksimum. Misalkan a_{ij} adalah keuntungan yang akan diperoleh pekerja i jika memilih tugas j. Misalkan didefinisikan A adalah himpunan pasangan (i, j) yang mungkin berpasangan. Misalkan suatu tugas j memiliki harga (*price*) p_j dimana pekerja yang memilih tugas tersebut harus membayar *price* sebesar p_j. Maka keuntungan bersih yang diperoleh pekerja i karena memilih tugas j adalah $a_{ij} - p_j$. Kemudian karena setiap pekerja i akan memilih tugas j_i yang memberikan keuntungan maksimum, maka diperoleh:

$$a_{ij_i} - p_{j_i} = \underset{(i,j)\in A}{\text{maks}}\{a_{ij_i} - p_j\}. \quad (2.43)$$

Persamaan (2.43) disebut juga sebagai *complementary slackness*, yang selanjutnya disingkat menjadi CS. Sistem ekonomi akan mencapai titik keseimbangan ketika setiap pekerja telah berpasangan dengan tugas yang memberikan keuntungan maksimum bagi pekerja tersebut.

Pada paragraf sebelumnya telah disebutkan bahwa masalah mencari keseimbangan dalam bidang ekonomi ekuivalen dengan masalah penugasan. Hubungan antara masalah keseimbangan dengan masalah penugasan ialah jika masalah keseimbangan dapat diselesaikan dan diperoleh total keuntungan maksimum maka masalah penugasan yang berkorespondensi juga dapat diselesaikan dan diperoleh solusi optimal. Kemudian himpunan harga (*price*) tugas yang diperoleh dari masalah keseimbangan berasosiasi dengan masalah dual dari masalah penugasan tersebut.

Berikut ini akan dijelaskan proses untuk memperoleh keseimbangan beserta himpunan harga (*price*) dari setiap tugas menggunakan algoritma *auction*. Pada setiap iterasi, algoritma *auction* memperbaharui vektor *price* **p** dan barisan penugasan (*assignment*). Vektor *price* pada algoritma *auction* merupakan vektor dengan elemen sebanyak *n*, *n* adalah banyaknya tugas, dimana elemen ke-*j* menunjukkan *price* tugas *j*. Pada setiap awal iterasi, kondisi (2.43) dipenuhi untuk setiap pasangan (i, j_i) yang telah ada dari masalah penugasan. Jika semua pekerja telah memiliki pasangan, algoritma berhenti. Jika tidak maka lakukan langkah-langkah berikut:

Misalkan *I* adalah himpunan tak kosong dari pekerja yang belum memiliki pasangan.

Fase penawaran : setiap pekerja $i \in I$ mencari tugas j_i yang menawarkan nilai maksimum, yaitu

$$j_i = \arg\underset{(i,j)\in A}{\text{maks}}\{a_{ij} - p_j\}, \quad (2.44)$$

dan menghitung peningkatan penawaran (*bidding increment*)

$$\gamma_i = \alpha_i - \beta_i, \quad (2.45)$$

dimana α_i adalah keuntungan bersih terbaik, yaitu

$$\alpha_i = \underset{(i,j)\in A}{\text{maks}}\{a_{ij} - p_j\}, \tag{2.46}$$

dan β_i adalah keuntungan bersih terbaik ke dua, yaitu

$$\beta_i = \underset{(i,j)\in A, j\neq j_i}{\text{maks}}\{a_{ij} - p_j\}. \tag{2.47}$$

Namun jika *j* adalah satu-satunya tugas yang dapat berpasangan dengan *i*, nilai β_i didefinisikan menjadi -∞ sehingga nilai *increment* γ_i menjadi ∞. Jika nilai β_i adalah -∞ maka *increment* dari *price j* ialah ∞ sehingga tugas *j* tersebut akan menjadi sangat mahal bagi pekerja lainnya. Hal tersebut mengakibatkan tugas *j* tidak akan direbut dari pekerja *i*.

Fase pemasangan : Misalkan *Q*(*j*) adalah subhimpunan tidak kosong dari pekerja yang ada di *I*, yaitu himpunan pekerja yang dapat berpasangan dengan tugas *j*. Setiap tugas *j* yang dipilih sebagai tugas terbaik oleh pekerja *i* menentukan penawar (*bidder*) terbaik, yaitu:

$$i_j = \arg\underset{i\in Q(j)}{\text{maks}}\,\gamma_i, \tag{2.48}$$

dan meningkatkan *price* tugas *j* sebesar *bidding increment* tertinggi, yaitu $\underset{i\in Q(j)}{\text{maks}}\,\gamma_i$. Selanjutnya tugas *j* tersebut dipasangkan dengan penawar tertinggi i_j.

Jika terdapat pekerja lain yang pada iterasi sebelumnya telah dipasangkan dengan tugas *j* maka pekerja tersebut menjadi tidak memiliki pasangan.

Algoritma berhenti ketika semua pekerja telah memiliki pasangan tugas.

Nilai γ_i tidak mungkin negatif karena $\alpha_i \geq \beta_i$ sehingga *price* setiap tugas pasti meningkat. Jika *i* adalah satu-satunya penawar maka γ_i adalah *bidding increment* tertinggi dari pemasangan pekerja *i* dengan tugas yang diinginkannya. Kemudian jika *bidding increment* terlalu tinggi, *price* suatu tugas bisa jadi terlalu mahal bagi penawar lainnya. Hal ini seperti pada konsep lelang yang sesungguhnya, *bidding increment* dan *price* meningkatkan kompetisi dengan membuat barang yang diinginkan oleh penawar menjadi kurang menarik bagi penawar potensial lainnya.

2.8.2 ε-Complementary Slackness pada Algoritma Auction untuk LAP

Konsep algoritma *auction* yang dijelaskan pada Subbab 2.8.1 ada kemungkinan tidak berjalan dengan baik, yaitu ketika *bidding increment* γ_i bernilai 0. Hal tersebut terjadi ketika terdapat lebih dari satu tugas menawarkan nilai maksimum kepada penawar *i*. Jika *bidding increment* bernilai 0, beberapa pekerja akan memilih tugas tersebut tanpa meningkatkan *price*-nya. Hal ini dapat mengakibatkan tugas tersebut mungkin direbut oleh pekerja lain. Jika kondisi tersebut terus berlangsung, dapat mengakibatkan siklus (*cycle*) yang tidak berhenti. Untuk menghindari *cycle* tersebut, dibuat suatu mekanisme yang juga sesuai dengan konsep lelang yang sesungguhnya dimana setiap penawaran untuk suatu tugas harus meningkatkan *price* tugas tersebut, minimal sebesar suatu *increment* positif. Selanjutnya penawar tugas tersebut harus mengambil resiko untuk memenangkan tugas tersebut.

Misalkan ε adalah suatu bilangan skalar positif. Maka suatu masalah penugasan beserta vektor *price* **p** dikatakan memenuhi ε-*complementary slackness*, yang selanjutnya disingkat menjadi ε-CS, jika

$$a_{ij_i} - p_{j_i} \geq \underset{j \in \mathcal{M}(i)}{\text{maks}}\{a_{ij} - p_j\} - \varepsilon, \tag{2.49}$$

untuk setiap pasangan (i, j_i). *Bidding increment* pada persamaan (2.45) diformulasi ulang sedemikian sehingga nilai *bidding increment* ada minimal sebesar ε dan diperoleh

$$\gamma_i = \alpha_i - \beta_i + \varepsilon. \tag{2.50}$$

Dengan pemilihan *bidding increment* seperti pada persamaan (2.50), kondisi ε-CS pada pertidaksamaan (2.49) akan terpenuhi.

Misalkan suatu tugas menerima penawaran dalam *k* iterasi, maka *price* tugas tersebut akan meningkat minimal sebesar $k\varepsilon$ dibandingkan dengan *price* awalnya. Untuk suatu nilai *k* yang cukup besar, tugas tersebut bisa menjadi terlalu mahal, terutama jika dibandingkan dengan tugas lainnya yang belum pernah memperoleh penawaran. Oleh karena itu, sebaiknya ditentukan bahwa suatu tugas hanya dapat menerima tawaran dalam sejumlah iterasi yang terbatas. Algoritma *auction* berhenti ketika setiap tugas telah menerima minimal satu penawaran.

Ketika algoritma *auction* berhenti, diperoleh penugasan yang memenuhi ε-CS. Namun keoptimalan algoritma ini bergantung pada nilai ε. Dalam kasus pelelangan yang sesungguhnya, penawar tidak akan menaikkan nilai penawarannya terlalu tinggi agar ia tidak perlu memenangkan suatu tugas dengan harga (*price*) yang terlalu tinggi. Oleh karena itu, jika nilai ε kecil, penugasan yang diperoleh akan hampir optimal. Hal ini dikarenakan ketika suatu penugasan yang layak beserta suatu himpunan *price* memenuhi kondisi ε-CS dengan nilai ε yang kecil, maka kondisi CS juga akan dipenuhi. Hal tersebut berarti diperoleh solusi optimal untuk masalah primal dan dual dengan keuntungan a_{ij} seperti yang telah dijelaskan sebelumnya. Berikut ini adalah teorema yang menjamin keoptimalan algoritma *auction*.

Teorema 2.6 (Bertsekas, 1992). Suatu penugasan layak yang memenuhi ε-CS bersama dengan suatu vektor *price* akan optimal dengan maksimal galat kesalahan sebesar $n\varepsilon$.

Misalkan a_{ij} merupakan bilangan bulat, maka total keuntungan untuk sembarang penugasan pasti merupakan bilangan bulat. Oleh karena itu, jika $n\varepsilon < 1$, sembarang penugasan lengkap yang optimal dengan galat kesalahan sebesar $n\varepsilon$ pasti optimal. Pernyataan tersebut dinyatakan dalam teorema berikut.

Teorema 2.7 (Bertsekas, 1992). Misalkan suatu masalah penugasan yang layak dengan keuntungan a_{ij} yang berupa bilangan bulat. Jika $\varepsilon < 1/n$, algoritma *auction* berhenti di sejumlah berhingga iterasi dengan penugasan yang optimal.

Konsep algoritma *auction* untuk LAP yang telah dibahas pada Subbab 2.8.1 beserta dengan konsep ε-CS pada Subbab 2.8.2 akan disesuaikan untuk penyelesaian QAP yang dibahas pada subbab berikut ini.

2.8.3 Algoritma *Auction* untuk Menyelesaikan SDVSP yang Dimodelkan Menjadi QAP

Ketika menyelesaikan SDVSP yang dimodelkan menjadi QAP menggunakan algoritma *auction*, SDVSP diformulasikan sebagai masalah memaksimumkan dengan mendefinisikan keuntungan $a_{ij} = -c_{ij}$ untuk setiap $(i, j) \in A$ dengan a_{ij} adalah bilangan bulat. Pada SDVSP, dalam mencari *successor* untuk setiap *trip*, *trip* tersebut tidak hanya dapat dipasangkan dengan *trip* lain tetapi juga dengan *depot* kedatangan, dan dalam mencari *predecessor* untuk setiap *trip*, *trip* tersebut tidak hanya dapat dipasangkan dengan *trip* lain tetapi juga dengan *depot* keberangkatan, maka untuk dapat memperoleh penugasan yang layak, hanya algoritma kombinasi *forward auction* dengan *reverse auction* yang dapat digunakan. Dari masalah dual QAP untuk SDVSP (2.39) – (2.42), diperoleh dua jenis variabel dual, yaitu p_j dan π_i. Pada algoritma *auction* untuk menyelesaikan SDVSP yang dimodelkan menjadi QAP, p_j ini menunjukkan *price* masing-masing *trip j* sedangkan π_i menunjukkan keuntungan bersih maksimum (*profit*) yang diperoleh suatu kendaraan karena melakukan *trip* j_i tepat setelah kendaraan tersebut melakukan *trip i*, yaitu $\pi_i = a_{ij_i} - p_{j_i}$.

2.8.4 ε-Complementary Slackness pada Algoritma Auction untuk SDVSP yang Dimodelkan Menjadi QAP

Seperti konsep ε-CS pada LAP, konsep ε-CS juga digunakan untuk SDVSP yang dimodelkan menjadi QAP. Konsep ε-CS untuk SDVSP tersebut disebutkan dalam definisi berikut ini:

Definisi 2.10 (Freling, Paixão, dan Wagelmans, 2001). Suatu SDVSP yang dimodelkan menjadi QAP beserta pasangan *profit-price* (π, p) dikatakan memenuhi ε-*CS* jika untuk suatu $\varepsilon > 0$:

$$\pi_i + p_j \geq a_{ij} - \varepsilon \qquad \forall\, (i,j) \in E \qquad (2.51)$$

$$\pi_i + p_j = a_{ij} \qquad \forall\, (i,j) \in S \qquad (2.52)$$

$$p_j \geq a_{rj} \qquad \forall\, j \in N \qquad (2.53)$$

$$p_j = a_{rj} \qquad \forall\, j \in D_r \qquad (2.54)$$

$$\pi_i \geq a_{it} \qquad \forall\, i \in N \qquad (2.55)$$

$$\pi_i = a_{it} \qquad \forall\, i \in D_t. \qquad (2.56)$$

Nilai ε tidak diperlukan untuk penawaran *depot* keberangkatan maupun *depot* kedatangan. Hal ini dikarenakan pada *depot*, penugasan ganda (*multiple assignments*) diperbolehkan. Pada Subbab 2.8.2 telah dijamin bahwa suatu masalah penugasan layak yang memenuhi ε-CS (2.49) bersama dengan suatu vektor *price* akan optimal dengan galat kesalahan sebesar $n\varepsilon$ dimana n adalah banyaknya tugas yang akan dipasangkan. Kemudian untuk menjamin keoptimalan solusi penugasan tersebut, diperlukan syarat cukup berupa $\varepsilon < 1/n$. Pada teorema berikut ini, akan dijamin bahwa hal tersebut juga berlaku untuk SDVSP yang dimodelkan menjadi QAP.

Teorema 2.11 (Freling, Paixão, dan Wagelmans, 2001). Misalkan suatu solusi SDVSP yang dimodelkan menjadi QAP bersama dengan pasangan *profit-price* (π, p) memenuhi ε-CS (2.51) – (2.56), maka solusi tersebut optimal dengan galat kesalahan maksimal sebesar $n\varepsilon$. Selanjutnya didefinisikan $\overline{p_j} = p_j + \varepsilon$ untuk setiap $j \in N$. Maka solusi layak untuk masalah dual QAP juga optimal dengan galat kesalahan maksimal sebesar $n\varepsilon$.

2.8.5 Algoritma Kombinasi *Forward Auction* dengan *Reverse Auction* untuk Menyelesaikan SDVSP

Pada Subbab 2.7 telah dijelaskan bahwa masalah penjadwalan kendaraan satu *depot* (SDVSP) dapat dimodelkan menjadi masalah *quasi-assignment* (QAP). Oleh karena itu seperti yang telah dijelaskan pada Subbab 2.8.3, SDVSP akan diselesaikan menggunakan algoritma kombinasi *forward auction* dengan *reverse auction*. Misalkan $G(V, A)$ adalah jaringan yang merepresentasikan suatu SDVSP dengan $V = N \cup \{r, t\}$ dan $A = E \cup (\{r\} \times N) \cup (N \times \{t\})$ dimana N adalah

himpunan *trip* dan *E* adalah himpunan *deadhead* antara dua *trip* dengan jenis-jenis penugasan pada SDVSP seperti yang telah dijelaskan pada Subbab 2.7.

SDVSP yang akan diselesaikan ialah SDVSP yang telah dimodelkan menjadi QAP dan diubah menjadi masalah memaksimumkan dengan memisalkan $a_{ij} = -c_{ij}$ untuk setiap $(i, j) \in A$, yaitu masalah primal (2.35) – (2.38). Masalah dual dari masalah primal (2.35) – (2.38) adalah masalah (2.39) – (2.42). Selanjutnya, masalah primal dan dual tersebutlah yang akan diselesaikan menggunakan algoritma kombinasi *forward auction* dengan *reverse auction*.

Algoritma kombinasi *forward auction* dengan *reverse auction* menghasilkan barisan vektor pasangan *profit-price* (π, p) sedemikian sehingga solusi penugasan yang layak diperoleh dengan memenuhi kondisi ε-CS untuk setiap $(i, j) \in A$. Pada kondisi awal algoritma *auction*, digunakan sembarang himpunan *price* dan setiap *trip* belum ada yang memiliki pasangan. Selanjutnya pada iterasi *forward auction*, *trip* yang belum memiliki *successor* akan melakukan penawaran terhadap *trip* lain atau *depot* kedatangan. Misalkan $f_{ij} = a_{ij} - p_j$ adalah nilai penawaran *trip i* untuk *trip* lain *j* dan misalkan $f_{it} = a_{it} + \varepsilon$ adalah nilai penawaran *trip i* untuk *depot* kedatangan, dengan nilai ε seperti yang telah dijelaskan pada Subbab 2.8.4. Maka selanjutnya akan dipaparkan mengenai *forward auction* dan *reverse auction*.

Berikut ini langkah-langkah algoritma *forward auction* untuk SDVSP:

Langkah 1. Cari *trip i* yang belum memiliki *successor*. Jika tidak ada maka algoritma berhenti. Jika ada maka lanjutkan langkah ini. *Trip* lain atau *depot* kedatangan yang memberikan keuntungan bersih terbaik adalah

$$j_i = \arg \operatorname*{maks}_{(i,j)\in A} f_{ij}, \tag{2.64}$$

dengan nilai

$$\alpha_i = \operatorname*{maks}_{(i,j)\in A} f_{ij}, \tag{2.65}$$

dan jika terdapat lebih dari satu kandidat *successor*,

$$\beta_i = \operatorname*{maks}_{(i,j)\in A, j\neq j_i} f_{ij}, \tag{2.66}$$

jika tidak, $\beta_i \approx -\infty$.

Jika $j_i \neq t$, yang merupakan *depot* kedatangan, maka lanjut ke Langkah 2. Jika tidak lanjut ke Langkah 3.

Langkah 2. Perbaharui *price* dan *profit*

$$p_{j_i} = p_{j_i} + \alpha_i - \beta_i + \varepsilon, \tag{2.67}$$

$$\pi_i = a_{i_{ji}} - p_{j_i}, \tag{2.68}$$

dan perbaharui penugasan: pasangan (i, j_i) menjadi elemen S. Jika *trip* j_i sebelumnya telah berpasangan dengan *trip* lain maka pasangan *trip* tersebut dengan *trip* j_i dikeluarkan dari S. Kemungkinan lain, jika tugas j_i sebelumnya telah berpasangan dengan *depot* keberangkatan, maka j_i dikeluarkan dari D_r. Kembali ke Langkah 1.

Langkah 3. Perbaharui *profit*: $\pi_i = a_{it}$, dan perbaharui penugasan: i menjadi elemen D_t. Kembali ke Langkah 1.

Algoritma *reverse auction* merupakan prosedur untuk menentukan *predecessor* setiap *trip* dimana *predecessor* tersebut dapat berupa *trip* lain atau *depot* keberangkatan. Prosedur *reverse auction* serupa dengan prosedur *forward auction*, hanya saja penawaran untuk kandidat *sucessor* diganti dengan penawaran untuk kandidat *predecessor*. Untuk menjamin konvergensi algoritma kombinasi *forward auction* dengan *reverse auction*, metode *forward auction* dan *reverse auction* dijalankan secara bergantian minimal setelah satu *trip* berhasil dipasangkan. Namun pada tugas akhir ini tahapan *reverse auction* hanya digunakan untuk melakukan penugasan langsung antara *trip* dengan *depot* keberangkatan.

BAB 3
MASALAH PENJADWALAN KENDARAAN BUS *RAPID TRANSIT* DENGAN MEMPERHATIKAN JADWAL PENGISIAN BAHAN BAKAR BESERTA PENYELESAIANNYA MENGGUNAKAN ALGORITMA *AUCTION*

Sistem transportasi umum bus *rapid transit* merupakan suatu sistem transportasi umum yang menggunakan kendaraan berkapasitas relatif besar, cepat, nyaman, aman, tepat waktu, dan memiliki jalur terpisah dengan kendaraan lainnya (Schwarzenegger, McPeak, dan Kempton, 2006). Sistem ini telah diaplikasikan di beberapa negara seperti Amerika, Australia, Brazil, Kolombia, termasuk Indonesia. Kelebihan sistem *rapid transit* selain kendaraan yang memiliki jalur kendaraan khusus, kendaraan tersebut juga lebih diprioritaskan dibandingkan kendaraan lain ketika berada di jalanan, berkarakteristik *tram* (kendaraan gandeng) sehingga berkapasitas besar, dan halte berkualitas tinggi. Di Indonesia, khususnya Jakarta, sistem *rapid transit* ini diaplikasikan pada kendaraan TransJakarta yang memiliki tempat pengisian bahan bakar khusus.

Menurut Huisman (2004), penjadwalan kendaraan bergantung pada rute yang dilewati kendaraan beserta frekuensi kendaraan melewati rute tersebut. Namun dengan adanya sistem *rapid transit* di Jakarta yang memiliki stasiun pengisian bahan bakar khusus dan ketika pengisian bahan bakar dilakukan tidak diperkenankan adanya penumpang di dalam kendaraan, dalam penjadwalan kendaraannya, jadwal pengisian bahan bakar kendaraan menjadi lebih diperhatikan.

Pada bab ini akan dibuat pemodelan untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar secara umum. Setelah model matematis penjadwalan kendaraan dengan memperhatikan jadwal pengisian bahan bakar diperoleh, model tersebut akan diaplikasikan pada masalah penjadwalan kendaraan bus TransJakarta dalam skala kecil yang kemudian diselesaikan menggunakan algoritma *auction* yang telah disesuaikan.

3.1 Deskripsi Masalah Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Sebelum membuat model matematis penjadwalan kendaraan dengan memperhatikan jadwal pengisian bahan bakar, terlebih dahulu dipaparkan mengenai deskripsi masalah tersebut. Masalah penjadwalan kendaraan yang dibahas pada tugas akhir ini adalah masalah penjadwalan kendaraan satu *depot* karena *depot* yang digunakan hanya satu. Pada penjadwalan kendaraan dengan memperhatikan jadwal pengisian bahan bakar, akan dilakukan pengaturan kendaraan terhadap himpunan *trip* sedemikian sehingga:

- Setiap *trip* dijalankan oleh tepat satu kendaraan dengan memperhatikan jadwal pengisian bahan bakar kendaraan tersebut,
- Setiap kendaraan melakukan barisan *trip* yang layak,
- Diperoleh biaya operasional yang minimum. Biaya operasional yang dihitung ialah biaya yang dikeluarkan karena melakukan *trip*, melakukan *deadhead*, menggunakan kendaraan, dan menggunakan bahan bakar.

Tahapan-tahapan pemodelan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar akan dijelaskan pada subbab selanjutnya.

3.2 Pemodelan Matematis Masalah Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Untuk dapat menyelesaikan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, masalah tersebut terlebih dahulu akan dimodelkan secara matematis. Model dibuat berdasarkan model matematis penjadwalan kendaraan yang sudah ada kemudian disesuaikan untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Selanjutnya akan dibuat model matematis masalah

penjadwalan kendaraan satu *depot* untuk bus *rapid transit* dengan fungsi tujuan meminimumkan biaya operasional dan kendala sebagai berikut:

a) Setiap *trip* ditugaskan kepada tepat satu kendaraan.
b) Setiap kendaraan mengerjakan barisan *trip* yang layak.
c) Setiap kendaraan akan mengisi bahan bakar jika:
   - bahan bakar yang tersisa tidak cukup untuk melakukan *trip* selanjutnya beserta *deadhead* dari lokasi kedatangan *trip* selanjutnya ke lokasi SPBB atau
   - telah selesai melakukan *trip* terakhir dalam satu hari.

Berikut ini dibuat model matematis masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar melalui tahapan sebagai berikut, yaitu penentuan asumsi-asumsi yang digunakan, pendefinisian variabel, pendefinisian fungsi kendala, pendefinisian fungsi tujuan, dan formulasi lengkap model matematis.

3.2.1 Asumsi-Asumsi yang Digunakan

Berikut ini adalah asumsi yang digunakan pada pembuatan model matematis penjadwalan kendaraan dengan memperhatikan jadwal pengisian bahan bakar:

a) Jadwal keberangkatan kendaraan (*timetable*) kendaraan sudah ada.
b) Hanya digunakan satu *depot* dan satu stasiun pengisian bahan bakar. Stasiun pengisian bahan bakar ini selanjutnya disingkat menjadi SPBB
c) Tidak ada biaya yang dikeluarkan ketika kendaraan sedang menganggur (*idle*) di *depot*.
d) Tidak ada kendala waktu, yaitu dalam satu hari tidak ada batasan maksimum lamanya kendaraan boleh digunakan.
e) Dari *depot* kendaraan tidak diperkenankan untuk ke SPBB.
f) Setiap akan melakukan pengisian bahan bakar, kendaraan harus dalam kondisi tidak membawa penumpang sehingga kendaraan hanya diperbolehkan mengisi bahan bakar setelah selesai melakukan suatu perjalanan (*trip*), bukan ketika sedang melakukan *trip*.

g) Setelah selesai melakukan *trip* terakhir dalam satu hari, sebelum kembali ke *depot*, setiap kendaraan menuju SPBB untuk mengisi bahan bakar sehingga pada hari berikutnya kendaraan menjalani *trip* pertama dengan kondisi tangki yang terisi penuh.

3.2.2 Pendefinisian Variabel

Pada Subbab 2.3 telah dijelaskan mengenai tahapan dalam membuat penjadwalan kendaraan yang dimulai dari *timetable* sebagai masukan, kemudian dari *timetable* diperoleh himpunan *trip*, dan dari himpunan *trip* tersebut dibuat penjadwalan kendaraan sehingga diperoleh barisan-barisan *trip* yang dijalankan oleh setiap kendaraan (*vehicle blocks*) sebagai keluaran. Selanjutnya didefinisikan variabel-variabel yang diperlukan dalam model matematis penjadwalan kendaraan bus *rapid transit*:

a) N = himpunan *trip* yang diurutkan meningkat berdasarkan waktu keberangkatan, dimana $N = \{1, 2, ..., n\}$

b) E = himpunan *deadhead* antara dua *trip* atau $E = \{(i,j) \mid i < j \text{ compatible}, i, j \in N\}$

c) r, t = *depot*, r dan t adalah *depot* yang sama, hanya saja r merupakan *depot* keberangkatan kendaraan sedangkan t merupakan *depot* kedatangan kendaraan

d) s = SPBB

e) V = himpunan simpul (*trip*, SPBB, dan *depot*), dimana $V = N \cup \{r, s, t\}$

f) A = himpunan busur, dimana $A = E \cup (r \times N) \cup (N \times s) \cup (s \times N) \cup \{(s, t)\}$

g) c_{ij} = biaya tidak tetap, yaitu biaya yang bergantung pada simpul i dan j. Variabel c_{ij} didefinisikan secara matematis sebagai berikut:

$$c_{ij} = \begin{cases} c + L_{rj} + c_j, & i = r; j \in N \\ L_{is}, & i \in N; j = s \\ L_{ij} + c_j, & i \in N; j \in N \\ L_{sj} + c_j, & i = s; j \in N \\ L_{st}, & i = s; j = t \end{cases}$$

dimana

L_{ij} = biaya untuk melakukan *deadhead* dari simpul i ke simpul j, $i, j \in V$

c = biaya tetap, yaitu biaya yang dikeluarkan karena menggunakan satu unit kendaraan (dalam rupiah)

c_j = biaya untuk melakukan *trip* j, $j \in N$

h) t_{isi} = lamanya kendaraan mengisi bahan bakar (dalam menit)

i) B_k = bahan bakar yang tersisa di tangki kendaraan k

j) $B(r, j)$ = banyaknya bahan bakar yang dibutuhkan kendaraan untuk melakukan *deadhead* dari *depot* keberangkatan ke lokasi keberangkatan *trip j*

k) $B(i, j)$ = banyaknya bahan bakar yang dibutuhkan kendaraan untuk melakukan *deadhead* dari lokasi kedatangan *trip i* ke lokasi keberangkatan *trip j*

l) $B(j, s)$ = banyaknya bahan bakar yang dibutuhkan kendaraan untuk melakukan *deadhead* dari lokasi kedatangan *trip j* ke SPBB

m) BB_j = banyaknya bahan bakar yang dibutuhkan kendaraan untuk melakukan *trip j*

n) kap = kapasitas tangki bahan bakar kendaraan

o) $y_{ij} = \begin{cases} 1, \text{kendaraan melakukan } trip\ j \text{ tepat setelah menyelesaikan } trip\ i \\ 0, \text{lainnya} \end{cases}$ $\forall\ (i, j) \in E$

p) $y_{rj} = \begin{cases} 1, \text{jika dari } depot \text{ kendaraan melakukan } trip\ j \\ 0, \text{lainnya} \end{cases}$ $\forall\ j \in N$

q) $y_{sj} = \begin{cases} 1, \text{jika kendaraan melakukan } trip\ j \text{ tepat setelah mengisi bahan bakar} \\ 0, \text{lainnya} \end{cases}$ $\forall\ j \in N$

r) $y_{is} = \begin{cases} 1, \text{jika kendaraan mengisi bahan bakar tepat setelah melakukan } trip\ i \\ 0, \text{lainnya} \end{cases}$ $\forall\ i \in N$

s) $y_{it} = \begin{cases} 1, \text{jika } trip\ i \text{ adalah } trip \text{ terakhir yang dijalankan sebelum kendaraan mengisi bahan bakar dan kembali ke } depot \\ 0, \text{lainnya} \end{cases}$ $\forall\ i \in N$

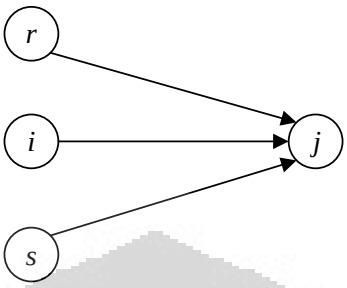
3.2.3 Fungsi-Fungsi Kendala

Pada Subbab 3.2.1 telah didefinisikan variabel-variabel yang akan digunakan dalam pemodelan matematis masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Pada subbab ini akan dijelaskan fungsi-fungsi yang menjadi kendala dalam masalah meminimumkan biaya operasional dalam penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Kendala untuk masalah ini serupa dengan kendala untuk masalah *quasi-assignment* yang telah dijelaskan pada Subbab 2.6, yaitu setiap *trip* memiliki tepat satu *predecessor*, setiap *trip* memiliki tepat satu *successor*, dan nilai setiap variabel keputusan ialah 0 atau 1. Setiap *trip* $j \in N$ memiliki tepat satu *predecessor* artinya sebelum kendaraan melakukan *trip j*, kendaraan berasal dari tepat satu lokasi lain. Dalam penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, lokasi tersebut dapat berupa *depot*, SPBB, atau lokasi kedatangan *trip* lain. Setiap *trip* $i \in N$ memiliki tepat satu *successor* artinya setelah kendaraan melakukan *trip i*, kendaraan tersebut melanjutkan perpindahan ke tepat satu lokasi lain. Dalam penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, karena kendaraan tidak dapat langsung kembali ke *depot* melainkan harus ke SPBB terlebih dahulu, lokasi yang dapat menjadi *trip i* ialah SPBB atau lokasi keberangkatan *trip* lain. Karena pada penjadwalan kendaraan bus *rapid transit* jadwal pengisian bahan bakar juga diperhatikan, maka pada model matematis penjadwalan yang akan dibuat juga ditambahkan kendala pengisian bahan bakar bagi kendaraan.

a) Setiap *trip* memiliki tepat satu *predecessor*

Dari persamaan (2.9) pada Subbab 2.6, diperoleh bahwa *predecessor* untuk setiap *trip* dapat berupa *depot* atau *trip* lainnya. Namun pada tugas akhir ini, karena pengisian bahan bakar juga dipertimbangkan dalam penjadwalan kendaraan bus *rapid transit*, maka SPBB juga mungkin menjadi kandidat *predecessor* suatu *trip*. Simpul kandidat *predecessor* ini harus terhubung dengan

trip tersebut, yaitu melalui busur berarah yang keluar dari simpul kandidat menuju *trip*. Kandidat *predecessor* untuk suatu *trip j* diberikan pada gambar berikut ini:



Gambar 3.1 Kandidat *Predecessor* untuk Suatu *Trip j*

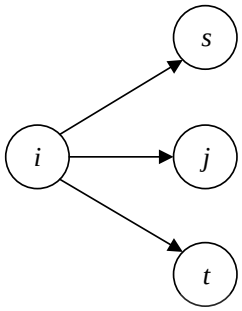
Gambar 3.1 menunjukkan bahwa kandidat *predecessor* suatu *trip j* dapat berupa *depot* keberangkatan, yaitu simpul *r*, atau berupa *trip* lain *i*, dengan $i \in N$, atau berupa SPBB, yaitu simpul *s*. Kandidat *predecessor* suatu *trip* dapat lebih dari satu, namun yang akan dipilih menjadi *predecessor* yang sebenarnya ialah tepat satu. Maka dari itu diperoleh kendala seperti berikut:

$$\sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N, \qquad (3.1)$$

dengan himpunan *A* seperti yang telah didefinisikan pada Subbab 3.2.1.

b) Setiap *trip* memiliki tepat satu *successor*

Dari persamaan (2.10) pada Subbab 2.6, diperoleh bahwa *successor* untuk setiap *trip* dapat berupa *depot* atau *trip* lainnya. Namun karena pengisian bahan bakar juga dipertimbangkan dalam penjadwalan kendaraan bus *rapid transit*, maka SPBB juga mungkin menjadi *successor* suatu *trip*. Berdasarkan asumsi pada Subbab 3.1 jika tidak ada lagi *trip* yang dapat dijalankan oleh suatu kendaraan dalam satu hari, kendaraan tersebut tidak dapat langsung kembali ke *depot* tetapi harus mengisi bahan bakar terlebih dahulu. Oleh karena itu, untuk masalah penjadwalan kendaraan bus *rapid transit*, *depot* tidak dapat menjadi *succcessor* suatu *trip* sehingga kandidat *successor* suatu *trip* ialah SPBB atau *trip* lainnya. Simpul kandidat *successor* ini harus terhubung dengan *trip* tersebut, yaitu melalui busur berarah yang keluar dari *trip* menuju simpul kandidat. Kandidat *predecessor* untuk suatu *trip j* diberikan pada gambar di bawah ini:



Gambar 3.2 Kandidat *Successor* untuk Suatu *Trip i*

Gambar 3.2 menunjukkan bahwa kandidat *successor* suatu *trip i* dapat berupa SPBB, yaitu simpul *s*, atau berupa *trip* lain *j*, dengan $j \in N$, atau berupa *depot* kedatangan, yaitu simpul *t*. Kandidat *successor* ini dapat lebih dari satu simpul, namun yang akan dipilih menjadi *successor* yang sebenarnya dari suatu *trip* hanya satu. Maka dari itu diperoleh kendala seperti berikut:

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N, \qquad (3.2)$$

dengan himpunan A seperti yang telah didefinisikan pada Subbab 3.2.1.

c) Setiap kendaraan akan mengisi bahan bakar jika:

- Bahan bakar yang tersisa di dalam tangki kendaraan tidak cukup untuk melakukan perjalanan selanjutnya dan *deadhead* dari lokasi akhir perjalanan selanjutnya ke lokasi SPBB.

  Jika suatu kendaraan telah selesai melakukan suatu *trip i*, sebelum kendaraan tersebut melakukan *trip* lain *j*, banyaknya bahan bakar yang tersedia dalam tangki diperiksa apakah cukup untuk melakukan *deadhead* dari lokasi kedatangan *trip i* ke lokasi keberangkatan *trip j*, kemudian untuk melakukan *trip j* itu sendiri, dan *deadhead* dari lokasi kedatangan *trip j* ke lokasi SPBB. Jika bahan bakar tidak mencukupi untuk melakukan *trip* dan *deadhead* tersebut maka kendaraan diharuskan mengisi bahan bakar.

$$y_{is} = \begin{cases} 1, B_k \le B(i,j) + BB_j + B(j,s) \\ 0, \text{lainnya} \end{cases} \quad \forall\, i \in N, (i,j) \in E, k = 1, 2, \ldots, n \qquad (3.3)$$

- Telah selesai melakukan *trip* terakhir dalam satu hari.

  Ketika tidak ada lagi *trip* yang dapat dijalankan oleh suatu kendaraan, maka sesuai dengan persamaan (3.1), kendaraan akan menuju SPBBG. Kemudian

setelah selesai mengisi bahan bakar dan tidak ada lagi *trip* yang dapat dijalankan, maka sesuai dengan persamaan (3.2), kendaraan akan kembali ke *depot*.

3.2.4 Fungsi Tujuan

Fungsi tujuan yang dibuat adalah fungsi yang meminimumkan biaya operasional yang dikeluarkan oleh suatu perusahaan transportasi umum dengan kendaraan berupa bus *rapid transit*. Biaya operasional yang dihitung pada tugas akhir ini ialah biaya yang dikeluarkan karena menggunakan kendaraan, melakukan barisan *trip*, dan beberapa *deadhead* yang diperlukan sehingga diperoleh fungsi tujuan sebagai berikut:

$$\text{Min} \sum_{(i,j)\in A} c_{ij} y_{ij}. \tag{3.4}$$

3.2.5 Formulasi Lengkap

Sesuai dengan kendala-kendala dan fungsi tujuan yang telah dibuat dari persamaan (3.1) – (3.5), diperoleh formulasi lengkap untuk penjadwalan kendaraan bus *rapid transit* secara umum seperti berikut:

$$\text{Min} \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{3.5}$$

$$\text{d. s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{3.6}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{3.7}$$

$$y_{is} = \begin{cases} 1, B_{ki} \le B(i,j) + B_j + B(j,s) \\ 0, \text{lainnya} \end{cases} \qquad \forall\, i \in N, (i,j) \in E, k = 1, 2, \ldots, n \tag{3.8}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A. \tag{3.9}$$

Model (3.5) – (3.9) merupakan model umum untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Masalah penjadwalan tersebut akan diselesaikan menggunakan algoritma *auction*

dengan terlebih dahulu mengubah masalah LP (3.5) – (3.9) ke dalam bentuk dualnya seperti yang akan dijelaskan pada subbab berikut ini.

3.3 Pembentukan Masalah Dual dari Masalah Primal Penjadwalan Kendaraan Bus *Rapid Transit*

Masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar akan diselesaikan menggunakan algoritma *auction*. Masalah primal penjadwalan kendaraan bus *rapid transit* yang dimodelkan menjadi masalah (3.5) – (3.9) diubah terlebih dahulu menjadi masalah memaksimumkan dan memisalkan $a_{ij} = -c_{ij}$ sehingga diperoleh:

$$\text{Maks} \sum_{(i,j)\in A} a_{ij} y_{ij} \tag{3.10}$$

$$\text{d. s.} \sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, j \in N \tag{3.11}$$

$$\sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall\, i \in N \tag{3.12}$$

$$y_{is} = \begin{cases} 1, B_{ki} \le B(i,j) + B_j + B(j,s) \\ 0, \text{lainnya} \end{cases} \qquad \forall\, i \in N, (i,j) \in E, k = 1, 2, \ldots, n \tag{3.13}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A. \tag{3.14}$$

Karena algoritma *auction* adalah algoritma primal-dual, yang menyelesaikan masalah primal dan masalah dualnya, maka masalah primal (3.10) – (3.14) akan dicari bentuk dualnya. Dengan cara yang sama seperti pada Subbab 2.7, diperoleh bentuk dual untuk masalah primal (3.10) – (3.14) sebagai berikut:

$$\text{Min} \sum_{i=1}^{n} \pi_i + \sum_{j=1}^{n} p_j \tag{3.15}$$

$$\text{d.s. } \pi_i + p_j \geq a_{ij} \qquad \forall\, (i,j) \in E \tag{3.16}$$

$$p_j \geq a_{rj} \qquad \forall\, j \in N \tag{3.17}$$

$$p_j \geq a_{sj} \qquad \forall\, j \in N \tag{3.18}$$

$$\pi_i \geq a_{is} \qquad \forall\, i \in N \tag{3.19}$$

$$\pi_i \geq a_{it} \qquad \forall\, i \in N. \tag{3.20}$$

Setelah diperoleh masalah primal (3.10) – (3.14) dan masalah dual (3.15) – (3.20) adalah masalah primal-dual yang akan diselesaikan menggunakan algoritma *auction* untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Pada subbab selanjutnya, akan dipaparkan ide algoritma *auction* untuk penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Algoritma tersebut merupakan modifikasi dari algoritma *auction* untuk SDVSP yang telah dijelaskan pada Subbab 2.8.6.

3.4 Penerapan Algoritma *Auction* pada Masalah Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Untuk menyelesaikan masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, juga akan digunakan algoritma kombinasi *forward auction* dengan *reverse auction*. Pada Subbab 2.8.6 dijelaskan bahwa konsep algoritma *forward auction* digunakan untuk menentukan *successor* setiap *trip*. Konsep yang sama juga akan digunakan pada masalah penjadwalan kendaraan bus *rapid transit*. Namun karena pada masalah ini jadwal pengisian bahan bakar juga turut diperhatikan, maka kandidat *successor* untuk setiap *trip* tidak hanya dapat berupa *trip* atau *depot* kedatangan, tetapi juga dapat

berupa SPBB. Selanjutnya setiap kendaraan melakukan *trip*, sisa bahan bakar yang ada di dalam tangki akan dihitung.

Berikut ini akan dijelaskan mengenai algoritma *forward auction* untuk penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Kondisi yang ditentukan untuk algoritma tersebut ialah:

a) Jika masih ada *trip* yang dapat dijalankan dan bahan bakar yang tersisa di dalam tangki mencukupi untuk melakukan *trip* tersebut beserta *deadhead* yang diperlukan maka *trip* yang akan menjadi *successor*.
b) Jika masih ada *trip* yang dapat dijalankan namun bahan bakar yang tersisa di dalam tangki tidak mencukupi untuk melakukan *trip* tersebut beserta *deadhead* yang diperlukan maka SPBB yang akan menjadi *successor* dan kembali ke poin a).
c) Jika sudah tidak ada lagi *trip* yang dapat dijalankan maka yang akan menjadi *successor* ialah *depot* kedatangan dengan sebelumnya kendaraan mengisi bahan bakar di SPBB.

Selanjutnya akan dijelaskan mengenai perbedaan antara algoritma *auction* untuk SDVSP dengan algoritma *auction* untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar. Pada algoritma *auction* untuk SDVSP, barisan *trip* yang dieksekusi oleh suatu kendaraan dapat dilihat dari himpunan penugasan *trip-trip S*. Untuk mengetahui *trip* mana saja yang memiliki *predecessor* berupa *depot*, dapat dilihat dari himpunan penugasan *depot* keberangkatan D_r. Untuk mengetahui *trip* mana saja yang memiliki *successor* berupa *depot* kedatangan, dapat dilihat dari himpunan penugasan *depot* kedatangan D_t. Namun pada algoritma *auction* untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, untuk mengetahui *trip* mana yang dilakukan oleh suatu kendaraan, terutama setelah kendaraan tersebut mengisi bahan bakar, didefinisikan *array* S_1, S_2, ..., S_n, n adalah banyaknya *trip*, dimana S_k menunjukkan barisan *trip* dan pengisian bahan bakar, dimulai dari dan berakhir di *depot*, yang dieksekusi oleh kendaraan ke-k, $k = 1, 2, ..., n$. Hal tersebut dikarenakan simpul SPBB dapat dikunjungi lebih dari satu kali oleh suatu kendaraan dan dapat dikunjungi oleh lebih dari satu kendaraan.

Berikut ini akan dijelaskan langkah-langkah dalam algoritma *forward auction* untuk penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar:

Kondisi awal. Untuk setiap $(i, j) \in A$, ditentukan $a_{ij} = -c_{ij}$. Masukkan nilai variabel *kap* dan t_{isi}. Kemudian untuk setiap k, $k = 1, 2, ..., n$, $B_k = kap - (B(r, k) + BB_k)$ dan didefinisikan *array* $S_k = [\,]$. $\pi_1 = \pi_2 = ... = \pi_n = p_1 = p_2 = ... = p_n = 0$, $\varepsilon < 1/n$, $U = [1\ 2\ ...\ n]$, $i = U(1)$.

Langkah 1. Periksa apakah terdapat $1 \leq k \leq n$, dimana $i \in S_k$. Jika tidak, lanjut ke Langkah 2. Jika ya, lanjut ke Langkah 3.

Langkah 2. Cari k, dimana $1 \leq k \leq n$ dan k adalah bilangan terkecil sedemikian sehingga $S_k = [\,]$. Kemudian i menjadi elemen S_k, yaitu $S_k = [i]$.

Langkah 3. Misalkan *kand* adalah himpunan kandidat *successor* berupa *trip* yang diperoleh dari himpunan pasangan *trip* yang *compatible*, yaitu $kand = \{j \mid (i, j) \in E\}$. Jika $kand \neq \emptyset$ maka lanjut ke Langkah 4. Jika tidak, lanjut ke Langkah 8.

Langkah 4. Untuk setiap $j \in kand$, definisikan $sisa_j = B_k - (B(i, j) + BB_j) - B(j, s)$. Variabel tersebut menunjukkan sisa bahan bakar kendaraan k setelah menjalankan *trip* j dengan memperhatikan jadwal pengisian bahan bakar kendaraan. Misalkan *kandi* adalah himpunan kandidat *successor* berupa *trip* yang *compatible* dengan *trip* i dan bahan bakar kendaraan k cukup untuk melakukan *trip* tersebut beserta *deadhead* yang diperlukan, yaitu $kandi = \{j \mid j \in kand \text{ dan } sisa_j \geq 0\}$. Jika $kandi \neq \emptyset$, lanjut ke Langkah 5. Jika tidak, lanjut ke Langkah 8.

Langkah 5. Cari *trip* j_i yang memberikan nilai keuntungan bersih maksimal, yaitu

$$j_i = \arg \underset{j \in kandi}{\text{maks}}\, f_{ij} = \arg \underset{j \in kandi}{\text{maks}} \{a_{ij} - p_j\}, \tag{3.21}$$

dengan nilai

$$\alpha_i = \underset{j \in kandi}{\text{maks}}\, f_{ij}, \tag{3.22}$$

dan jika terdapat lebih dari satu kandidat *successor* maka

$$\beta_i = \underset{j \in kandi, j \neq j_i}{\text{maks}}\, f_{ij}, \tag{3.23}$$

jika tidak, $\beta_i \approx -\infty$.

Langkah 6. Perbaharui *price* dan *profit*:

Jika banyaknya elemen himpunan *kandi* ada lebih dari satu maka

$$p_{j_i} = p_{j_i} + \alpha_i - \beta_i + \varepsilon, \quad (3.24)$$

$$\pi_i = a_{ij_i} - p_{j_i}, \quad (3.25)$$

Jika banyaknya elemen himpunan *kandi* hanya satu maka

$$\pi_i = a_{ij_i}. \quad (3.26)$$

Perbaharui penugasan S_k: $S_k = [S_k\, j_i]$ dan keluarkan i dari U. Perbarui sisa bahan bakar kendaraan k: $B_k = B_k - (B(i, j_i) + BB_{j_i})$. Jika terdapat $1 \le l \le n$, $l \ne k$, dimana $j_i = S_l(m)$, dengan $1 \le m \le length(S_l)$ dan *length* adalah banyaknya elemen dalam S_l, maka lanjut ke Langkah 7. Jika tidak, pilih $i = j_i$. Kembali ke Langkah 1.

Langkah 7. $S_l = S_l(1{:}(m\text{-}1))$. Jika $S_l(m\text{-}1)$ merupakan *trip* maka $S_l(m\text{-}1) \in U$. Kemudian $S_l(m)$, $S_l(m+1)$, ..., $S_l(length(S_l))$ yang merupakan *trip*, menjadi elemen U. Perbaharui sisa bahan bakar kendaraan l: Jika terdapat $z = S_l(q)$, $1 \le q \le length(S_l)$, adalah saat pengisian bahan bakar yang terakhir kali dilakukan kendaraan l maka $B_l = kap - \left(B\left(z, S_l(q+1)\right) - BB_{S_l(q+1)}\right) - \sum_{v=(q+1)}^{length-1}\left(B\left(S_l(v), S_l(v+1)\right) - BB_{S_l(v+1)}\right)$. Jika tidak maka $B_l = kap - \sum_{v=1}^{length-1}\left(B\left(S_l(v), S_l(v+1)\right) - BB_{S_l(v+1)}\right)$. Pilih $i = j_i$ dan kembali ke Langkah 1.

Langkah 8. s menjadi elemen S_k dan $B_k = kap$. Misalkan *kands* adalah himpunan *trip* yang dapat dilakukan oleh kendaraan k setelah mengisi bahan bakar. Jika $kands = \emptyset$ maka lanjut ke Langkah 9. Jika tidak, lanjut ke Langkah 10.

Langkah 9. t menjadi elemen S_k dan perbaharui *profit*: $\pi_i = a_{is} + a_{st}$. Jika $U = [\ ]$, lanjut ke Langkah 12. Jika tidak, pilih $i = U(1)$ dan kembali ke Langkah 1.

Langkah 10. Cari *trip* j_i yang memberikan nilai keuntungan bersih maksimal, yaitu

$$j_i = \arg \underset{j \in kands}{\text{maks}}\, f_{ij} = \arg \underset{j \in kands}{\text{maks}}\, \{a_{ij} - p_j\}, \quad (3.27)$$

dengan nilai

$$\alpha_i = \underset{j \in kands}{\text{maks}}\, f_{ij}, \quad (3.28)$$

dan jika terdapat lebih dari satu kandidat *successor*,

$$\beta_i = \underset{j \in kands, j \ne j_i}{\text{maks}}\, f_{ij}, \quad (3.29)$$

jika tidak, $\beta_i \approx -\infty$.

Langkah 11. Perbaharui *price* dan *profit*:

Jika banyaknya elemen himpunan *kands* ada lebih dari satu maka

$$p_{j_i} = p_{j_i} + \alpha_i - \beta_i + \varepsilon, \quad (3.30)$$

$$\pi_i = a_{is} + a_{sj_i} - p_{j_i}, \quad (3.31)$$

jika banyaknya elemen himpunan *kands* hanya satu maka

$$\pi_i = a_{is} + a_{sj_i}. \quad (3.32)$$

Perbaharui penugasan S_k: $S_k = [S_k\, j_i]$ dan keluarkan i dari U. Perbarui sisa bahan bakar kendaraan k: $B_k = B_k - (B(s, j_i) + BB_{j_i})$. Jika terdapat $1 \le l \le n$, $l \ne k$, dimana $j_i = S_l(m)$, dengan $1 \le m \le length(S_l)$ dan *length* adalah banyaknya elemen dalam S_l, maka kembali ke Langkah 7. Jika tidak, pilih $i = j_i$. Kembali ke Langkah 1.

Langkah 12. Algoritma *forward auction* selesai. Lanjutkan dengan algoritma *reverse auction*.

Setelah dijelaskan mengenai algoritma *forward auction* untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, akan dipaparkan mengenai algoritma *reverse auction* untuk masalah tersebut. Algoritma *reverse auction* pada SDVSP, pada dasarnya merupakan prosedur untuk mencari *predecessor* setiap *trip*. *Predecessor* tersebut dapat berupa *depot* keberangkatan, SPBB, atau *trip* lainnya. Namun pada tugas akhir ini, algoritma *reverse auction* hanya digunakan untuk melakukan pemasangan *trip* pertama yang dilakukan oleh setiap kendaraan dengan *depot* keberangkatan. Oleh karena itu, pada algoritma *reverse auction* hanya dilakukan langkah berikut ini:

Langkah algoritma *reverse auction*. untuk setiap k, $k = 1, 2, ..., n$, $S_k \ne [\,]$. Jika $S_k(1) \ne s$ dan misalkan $j_k = S_k(1)$ maka $S_k = [r\ S_k]$ dan $p_k = p_k + \alpha_{rj_k}$. Algoritma selesai dan diperoleh keluaran berupa *array* $S_1, S_2, ..., S_n$, dan total biaya = $\sum_{i \in N} \pi_i + \sum_{j \in N} p_j$.

Untuk memperjelas tahapan penyelesaian masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma kombinasi *forward auction* dengan *reverse auction*, pada subbab selanjutnya akan diberikan contoh masalah tersebut beserta

penyelesaiannya menggunakan algoritma kombinasi *forward auction* dengan *reverse auction*.

3.5 Contoh Masalah Penjadwalan Kendaraan Bus *Rapid Transit* dengan Memperhatikan Jadwal Pengisian Bahan Bakar dan Penyelesaiannya Menggunakan Algoritma *Auction*

Misalkan diketahui *timetable* sebagai berikut:

Tabel 3.1 Contoh *Timetable* sebagai Masukan untuk Masalah Penjadwalan Kendaraan Bus *Rapid Transit*

Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan	Waktu Kedatangan
A	B	11.00	11.50
A	B	11.05	11.55
B	A	11.57	12.47
B	A	12.02	12.52
A	B	13.05	13.55
A	B	13.12	14.02

Dari *timetable* di atas, dapat diperoleh *trip* sebagai berikut:

Trip 1: berangkat dari lokasi A pukul 11.00 dan tiba di lokasi B pukul 11.50

Trip 2: berangkat dari lokasi A pukul 11.05 dan tiba di lokasi B pukul 11.55

Trip 3: berangkat dari lokasi B pukul 11.57 dan tiba di lokasi A pukul 12.47

Trip 4: berangkat dari lokasi B pukul 12.02 dan tiba di lokasi A pukul 12.52

Trip 5: berangkat dari lokasi A pukul 13.05 dan tiba di lokasi B pukul 13.55

Trip 6: berangkat dari lokasi A pukul 13.12 dan tiba di lokasi B pukul 14.02

Selanjutnya akan dipaparkan asumsi-asumsi tambahan yang akan digunakan selain dari asumsi yang telah disebutkan pada Subbab 3.1 beserta nilai dari variabel-variabel yang terdapat pada Subbab 3.2.1. Asumsi-asumsi tambahan tersebut ialah:

a) SPBB terletak di lokasi A.

b) Bahan bakar yang dibutuhkan oleh kendaraan untuk melakukan perpindahan dari lokasi A ke lokasi B atau sebaliknya adalah 8 liter (L).

c) Bahan bakar yang dibutuhkan oleh kendaraan untuk melakukan perpindahan dari *depot* ke lokasi A atau lokasi B adalah 3 L.

d) Kapasitas tangki setiap kendaraan adalah 22 L.

e) Harga bahan bakar yang digunakan ialah Rp. 3.100 per liter.

Berikut ini diberikan nilai untuk variabel-variabel yang telah disebutkan pada Subbab 3.2.1:

a) $N = \{1, 2, ..., n\}$

b) $E = \{(1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 6)\}$

c) r, t = *depot*, r dan t adalah *depot* yang sama, hanya saja r merupakan *depot* keberangkatan kendaraan sedangkan t merupakan *depot* kedatangan kendaraan

d) s = SPBB

e) $V = \{1, 2, ..., n, r, s, t\}$

f) $A = \{(1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 6), (r, 1), (r, 2), (r, 3), (r, 4), (r, 5), (r, 6), (1, s), (2, s), (3, s), (4, s), (5, s), (6, s), (s, 1), (s, 2), (s, 3), (s, 4), (s, 5), (s, 6), (s, t)\}$

g) c = 1.000.000

h) untuk setiap $(i, j) \in A$, nilai c_{ij} ialah sebagai berikut:

- $c_{13} = c_{14} = c_{23} = c_{24} = c_{35} = c_{36} = c_{46} = 8 \times 3.100 = 24.800$,
- $c_{15} = c_{16} = c_{25} = c_{26} = 16 \times 3.100 = 49.600$,
- $c_{r1} = c_{r2} = c_{r3} = c_{r4} = c_{r5} = c_{r6} = 1.000.000 + (11 \times 3.100) = 1.034.100$,
- $c_{1s} = c_{2s} = c_{5s} = c_{6s} = 8 \times 3.100 = 24.800$,
- $c_{3s} = c_{4s} = 0 \times 3.100 = 0$,
- $c_{s1} = c_{s2} = c_{s5} = c_{s6} = 0 \times 3.100 = 0$,
- $c_{s3} = c_{s4} = 8 \times 3.100 = 24.800$,
- $c_{st} = 3 \times 3.100 = 9.300$.

i) t_{isi} = 15

Dari *timetable* pada Tabel 3.1 akan dibuat penjadwalan kendaraan dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma kombinasi *forward auction* dengan *reverse auction* yang telah dijelaskan pada Gambar 3.1 dan Gambar 3.2.

Kondisi awal. Untuk setiap $(i, j) \in A$, $a_{ij} = -c_{ij}$, $kap = 22$, $t_{isi} = 15$, $k = 1, 2, 3, 4, 5, 6$, $B_1 = 22 - (B(r, 1) + BB_1) = 22 - (3 + 8) = 11$. Dengan cara yang sama diperoleh $B_2 = B_3 = B_4 = B_5 = B_6 = B_1 = 11$. Selanjutnya $S_1 = S_2 = S_3 = S_4 = S_5 = S_6 = [\,]$, $\pi_1 = \pi_2 = \pi_3 = \pi_4 = \pi_5 = \pi_6 = p_1 = p_2 = p_3 = p_4 = p_5 = p_6 = 0$, $\varepsilon = \frac{1}{7}$, $U = [1\ 2\ 3\ 4\ 5\ 6]$, $i = U(1) = 1$

Algoritma *forward auction*.

Iterasi 1.

- Tidak terdapat k, $1 \leq k \leq 6$, dimana $1 \in S_k$. Karena S_1 hingga S_6 masih belum memiliki elemen, maka $k = 1$. Kemudian $1 \in S_1$ sehingga $S_1 = [1]$.
- Akan dicari himpunan *trip* yang akan menjadi elemen himpunan *kand*. Berdasarkan himpunan busur A, diperoleh $kand = \{3, 4, 5, 6\}$. Selanjutnya akan dihitung nilai $sisa_j$ untuk setiap $j \in kand$ yaitu:
  $sisa_3 = B_1 - (B(1, 3) + BB_3) - B(3, s) = 11 - (0 + 8) - 0 = 3$
  $sisa_4 = B_1 - (B(1, 4) + BB_4) - B(4, s) = 11 - (0 + 8) - 0 = 3$
  $sisa_5 = B_1 - (B(1, 5) + BB_5) - B(5, s) = 11 - (8 + 8) - 0 = -5$
  $sisa_6 = B_1 - (B(1, 6) + BB_6) - B(6, s) = 11 - (8 + 8) - 0 = -5$
  Dari penghitungan $sisa_j$ untuk setiap $j \in kand$ diperoleh $kandi = \{3, 4\}$.
- $kandi \neq \emptyset$ maka akan dicari $j_1 \in kandi$ yang memberikan keuntungan bersih maksimal untuk *trip* 1, dengan mula-mula menghitung nilai f_{1j} untuk setiap $j \in kandi$.
  $f_{13} = a_{13} - p_3 = -c_{13} - p_3 = -24.800 - 0 = -24.800$
  $f_{14} = a_{14} - p_4 = -c_{14} - p_4 = -24.800 - 0 = -24.800$
  maka $\alpha_1 = \text{maks}\{f_{13}, f_{14}\} = \text{maks}\{-24.800, -24.800\} = -24.800$
  dan diperoleh $j_1 = 3$. Diperoleh $\beta_1 = maks\{f_{14}\} = maks\{-24.800\} = -24.800$.
- Nilai *price* dan *profit* akan diperbaharui:
  $p_3 = p_3 + \alpha_1 - \beta_1 + \varepsilon = 0 + (-24.800) - (-24.800) + \frac{1}{7} = \frac{1}{7}$,
  $\pi_1 = a_{13} - p_3 = -24.800 - 1/7 = -24.800\frac{1}{7}$,

$3 \in S_1$ dan diperoleh $S_1 = [1, 3]$ dan keluarkan 1 dari U sehingga diperoleh $U = [2\ 3\ 4\ 5\ 6]$. Sisa bahan bakar kendaraan 1 adalah: $B_1 = B_1 - (B(1, 3) + BB_3) = 11 - (0 + 8) = 3$.

$\nexists\ 1 \leq l \leq 6,\ l \neq 1$, dimana $3 = S_l(m)$, dengan $1 \leq m \leq length(S_l)$ dan *length* adalah banyaknya elemen dalam S_l, maka pilih $i = 3$.

Iterasi 2.

- Terdapat $k = 1$ dimana $3 \in S_1$. Selanjutnya $kand = \{5, 6\}$ dan hitung nilai $sisa_j$ untuk setiap $j \in kand$:
  $sisa_5 = B_1 - (B(3, 5) + BB_5) - B(5, s) = 3 - (0 + 8) - 8 = -13$ dan dengan cara yang sama diperoleh $sisa_6 = -13$. Diperoleh $kandi = \emptyset$ sehingga s menjadi elemen S_1 dan $S_1 = [1\ 3\ s]$.
- *Trip* terakhir yang dilakukan oleh kendaraan 1 adalah *trip* 3. Artinya kendaraan 1 tiba di lokasi A pukul 12.47. Setelah itu, kendaraan mengisi bahan bakar. Diketahui bahwa lamanya pengisian bahan bakar ialah 15 menit. Oleh karena itu kendaraan 1 akan selesai mengisi bahan bakar pada pukul 13.02. Maka $kands = \{5, 6\}$ dan karena $kands \neq \emptyset$, langkah berikut dilakukan:
  $f_{35} = a_{3s} + a_{s5} - p_5 = 0 + (-24.800) - 0 = -24.800$ dan dengan cara yang serupa diperoleh $f_{36} = -24.800$. Maka $\alpha_3 = -24.800$ dan $j_3 = 5$. Kemudian nilai $\beta_3 = -24.800$.
- Perbaharui *price* dan *profit*:
  $p_5 = p_5 + \alpha_3 - \beta_3 + \varepsilon = 0 + (-24.800) - (-24.800) + \frac{1}{7} = \frac{1}{7}$,
  $\pi_3 = a_{3s} + a_{s5} - p_5 = 0 + (-24.800) - \frac{1}{7} = -24.800\frac{1}{7}$,
  dan $S_1 = [1\ 3\ s\ 5]$ dan keluarkan 3 dari U sehingga diperoleh $U = [2\ 4\ 5\ 6]$. Pilih $i = 5$.

Dengan langkah-langkah yang serupa seperti pada Iterasi 1 dan Iterasi 2, akan dilakukan penghitungan untuk Iterasi 3 hingga algoritma *forward auction* mencapai kondisi berhenti.

Iterasi 3.

- $k = 1$ dan $kand = \emptyset$ sehingga dan s menjadi elemen S_1, diperoleh $S_1 = [1\ 3\ s\ 5\ s]$. Selanjutnya $kands = \emptyset$, maka t menjadi elemen S_1 dan perbaharui *profit*: $\pi_5 = a_{5s} + a_{st} = -24.800 + (-9.300) = -34.100$. Perbaharui penugasan S_1:

S_1 = [1 3 *s* 5 *s t*] dan keluarkan 5 dari U sehingga U = [2 4 6]. Pilih $i = U(1)$ = 2.

Iterasi 4.

- Tidak terdapat k, $1 \leq k \leq 6$, dimana $2 \in S_k$. Karena S_2 hingga S_6 masih belum memiliki elemen, maka $k = 2$. Selanjutnya $2 \in S_2$ sehingga S_2 = [2].
- *kand* = {3, 4, 5, 6}, artinya *kand* ≠ ∅. Dengan cara yang serupa pada iterasi-iterasi sebelumnya, akan diperoleh $sisa_j$ untuk setiap $j \in$ *kand* sebagai berikut: $sisa_3 = sisa_4 = 3$ dan $sisa_5 = sisa_6 = -13$.
  Maka *kandi* = {3, 4}. Karena *kandi* ≠ ∅, akan cari *trip* j_i yang memberikan keuntungan maksimum dan dengan cara yang serupa dengan iterasi-iterasi sebelumya diperoleh $f_{23} = -24.800\frac{1}{7}$ dan $f_{24} = -24.800$. Diperoleh $\alpha_2 = -24.800$, $j_2 = 4$, dan $\beta_2 = -24.800\frac{1}{7}$.
- perbaharui *price* dan *profit*:
  $p_4 = 0 + (-24.800) - (-24.800\frac{1}{7}) + \frac{1}{7} = \frac{2}{7}$,
  $\pi_2 = -24.800 - \frac{2}{7} = -24.800\frac{2}{7}$,
  dan perbaharui penugasan S_2: S_2 = [2 4] dan keluarkan 2 dari U sehingga U = [4 6]. Pilih $i = 4$.

Iterasi 5.

- Terdapat $k = 2$ dimana $4 \in S_2$. Selanjutnya *kand* = {6} dan hitung nilai $sisa_j$ untuk setiap $j \in$ *kand*: $sisa_6 = B_2 - (B(4, 6) + BB_6) - B(6, s) = 3 - (0 + 8) - 8 = -13$. Diperoleh *kandi* = ∅ sehingga *s* menjadi elemen S_2 dan S_2 = [2 4 *s*].
- *Trip* terakhir yang dilakukan oleh kendaraan 2 adalah *trip* 4. Artinya kendaraan 2 tiba di lokasi A pukul 12.52. Setelah itu, kendaraan mengisi bahan bakar. Diketahui bahwa lamanya pengisian bahan bakar ialah 15 menit. Oleh karena itu kendaraan 2 akan selesai mengisi bahan bakar pada pukul 13.07. Maka *kands* = {6} dan karena *kands* ≠ ∅, langkah berikut dilakukan: $f_{46} = a_{4s} + a_{s6} - p_6 = 0 + (-24.800) - 0 = -24.800$. Diperoleh $\alpha_4 = -24.800$ dan $j_4 = 6$. Kemudian nilai $\beta_4 \approx -\infty$.
- Perbaharui *profit*: $\pi_4 = 0 + (-24.800) = -24.800$,
  dan S_2 = [2 4 *s* 6] dan keluarkan 4 dari U sehingga U = [6]. Pilih $i = 6$.

Iterasi 6.

- $k = 2$ dan $kand = \emptyset$ sehingga dan s menjadi elemen S_2, diperoleh $S_2 = [2\ 4\ s\ 6\ s]$. Selanjutnya $kands = \emptyset$, maka t menjadi elemen S_2 dan perbaharui *profit*: $\pi_6 = a_{6s} + a_{st} = -24.800 + (-9.300) = -34.100$. Perbaharui penugasan S_2: $S_2 = [2\ 4\ s\ 6\ s\ t]$ dan keluarkan 6 dari U sehingga $U = [\ ]$. Karena $U = [\ ]$, algoritma *forward auction* berhenti dan akan dilanjutkan dengan algoritma *reverse auction*.

Tahapan algoritma *auction* akan dilakukan untuk S_1 dan S_2 karena kedua penugasan tersebut bukan merupakan *array* kosong.

Algoritma *reverse auction*.

Iterasi 1. $k = 1$, maka $j_1 = S_1(1) = 1$. Perbaharui penugasan S_1: $S_1 = [r\ 1\ 3\ s\ 5\ s\ t]$ dan perbaharui *price*: $p_1 = p_1 + a_{r1} = 0 + (-1.034.100) = -1.034.100$.

Iterasi 2. k = 2, maka $j_2 = S_2(1) = 2$. Perbaharui penugasan S_2: $S_2 = [r\ 2\ 4\ s\ 6\ s\ t]$ dan perbaharui *price*: $p_2 = p_2 + a_{r2} = 0 + (-1.034.100) = -1.034.100$.

Algoritma *reverse auction* selesai dan diperoleh keluaran:

$S_1 = [r\ 1\ 3\ s\ 5\ s\ t]$, $S_2 = [r\ 2\ 4\ s\ 6\ s\ t]$, dan total biaya:

$\sum_{i \in N} \pi_i + \sum_{j \in N} p_j = \pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 + \pi_6 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 =$

$(-24.800\frac{1}{7}) + (-24.800\frac{2}{7}) + (-24.800\frac{1}{7}) + (-24.800) + (-34.100) + (-34.100) +$

$(-1.034.100) + (-1.034.100) + \frac{1}{7} + \frac{2}{7} + \frac{1}{7} + 0 = -2.235.600$.

Kesimpulan yang dapat diperoleh dari hasil di atas ialah, untuk memenuhi *timetable* yang pada Tabel 3.1, diperlukan 2 kendaraan untuk melakukan perjalanan yang ada dengan total biaya yang dikeluarkan ialah Rp. 2.235.600. Kendaraan pertama berpindah dari *depot*, kemudian melakukan *trip* 1, *trip* 3, mengisi bahan bakar, melakukan *trip* 5, mengisi bahan bakar, dan kembali ke *depot*. Kemudian kendaraan ke dua berpindah dari *depot*, kemudian melakukan *trip* 2, *trip* 4, mengisi bahan bakar, melakukan *trip* 6, dan kembali ke *depot*.

Penyelesaian masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction* yang telah dijelaskan pada Subbab 3.4 akan diaplikasikan untuk menyelesaikan masalah penjadwalan kendaraan bus TransJakarta pada Bab 4.

BAB 4
APLIKASI PENDEKATAN ALGORITMA *AUCTION* PADA PENJADWALAN KENDARAAN BUS *RAPID TRANSIT* DENGAN MEMPERHATIKAN JADWAL PENGISIAN BAHAN BAKAR PADA PENJADWALAN KENDARAAN BUS TRANSJAKARTA

Pada Bab 3 telah dijelaskan mengenai penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar beserta algoritma *auction* untuk menyelesaikan masalah tersebut. Pada bab ini, pendekatan algoritma *auction* pada masalah penjadwalan kendaraan bus *rapid transit* akan diaplikasikan pada masalah penjadwalan kendaraan bus TransJakarta rute Dukuh Atas – Pulogadung dan rute Dukuh Atas – Ragunan. Dalam mengaplikasikan masalah penjadwalan kendaraan bus *rapid transit* pada bus TransJakarta, akan dilakukan tahapan penjadwalan kendaraan seperti pada Gambar 2.3, yaitu:

a) menentukan jadwal keberangkatan kendaraan (*timetable*) sebagai masukan dalam masalah penjadwalan,
b) melakukan penjadwalan kendaraan (*vehicle scheduling*),
c) diperoleh keluaran berupa barisan *trip* dan *deadhead* yang harus dilakukan oleh setiap kendaraan (*vehicle blocks*).

Pada tugas akhir ini, tahapan pada poin b) dilakukan menggunakan algoritma auction yang telah dijelaskan pada Subbab 3.4. Sebelum *timetable* dibuat sebagai masukan, akan dipaparkan mengenai asumsi-asumsi yang digunakan dalam penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar untuk rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan.

4.1 Asumsi-Asumsi yang Digunakan untuk Penjadwalan Kendaraan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Untuk menyelesaikan masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar, digunakan asumsi yang

sama seperti pada masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan bahan bakar seperti yang tertera pada Subbab 3.1 dengan tambahan asumsi sebagai berikut:

- Bahan bakar yang digunakan oleh bus TransJakarta berupa gas dan lokasi stasiun pengisian bahan bakar gas, yang selanjutnya disingkat menjadi SPBBG terletak di Pulogadung.
- Harga bahan bakar yang digunakan ialah Rp. 3.100 per liter skala premium (LSP).
- Bahan bakar yang diperlukan bus TransJakarta untuk berpindah dari Dukuh Atas ke Pulogadung atau sebaliknya ialah 7 LSP.
- Bahan bakar yang diperlukan bus TransJakarta untuk berpindah dari Dukuh Atas ke Ragunan atau sebaliknya ialah 8 LSP.
- Bahan bakar yang diperlukan bus TransJakarta untuk berpindah dari *depot* ke lokasi awal suatu *trip* ialah 3 LSP.
- Kapasitas tangki bahan bakar setiap bus TransJakarta ialah 120 LSP.

Pada subbab berikutnya akan dibahas mengenai pembuatan *timetable* untuk masalah penjadwalan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar. *Timetable* yang akan dibuat adalah jadwal keberangkatan kendaraan untuk rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan.

4.2 Pembuatan *Timetable* Bus TransJakarta Rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan

Pada penjadwalan kendaraan, suatu *timetable* seharusnya diketahui dan akan menjadi masukan untuk penjadwalan tersebut. Namun pada tugas akhir ini, data mengenai jadwal keberangkatan (*timetable*) kendaraan bus TransJakarta sulit diperoleh sehingga *timetable* akan dibuat dengan asumsi tertentu. *Timetable* untuk bus TransJakarta rute Dukuh Atas – Pulogadung dan rute Dukuh Atas – Ragunan dalam satu hari dibuat berdasarkan pertimbangan jam sibuk dan jam lengang. Jam sibuk maksudnya waktu ketika relatif banyak orang yang berpindah dari satu lokasi ke lokasi lain menggunakan kendaraan. Jam lengang maksudnya waktu

ketika relatif sedikit orang yang berpindah dari satu lokasi ke lokasi lainnya menggunakan kendaraan. *Timetable* dibuat berdasarkan selisih waktu kedatangan antara dua kendaraan (*headway*). Semakin kecil *headway* yang ditentukan, semakin banyak perjalanan (*trip*) yang harus jalankan. Hal ini berarti semakin banyak *trip* yang harus dievaluasi ketika menjalankan algoritma *auction*. Banyaknya *trip* mempengaruhi lamanya algoritma *auction* mencapai kondisi berhenti. Semakin banyak *trip* yang harus dievaluasi, semakin lama algoritma *auction* mencapai kondisi berhenti. Oleh karena itu pada tugas akhir ini, semakin mendekati jam sibuk, *headway* bus TransJakarta dibuat semakin kecil dengan nilai minimal 5 menit. Selanjutnya semakin mendekati jam lengang, *headway* bus TransJakarta dibuat semakin besar dengan maksimum *headway* 13 menit. Berikut ini ditentukan rincian *headway* bus TransJakarta dalam satu hari:

- Pukul 05.00 – 06.03, *headway* bus adalah 7 menit,
- Pukul 06.03 – 08.03, *headway* bus adalah 5 menit,
- Pukul 08.03 – 09.06, *headway* bus adalah 7 menit,
- Pukul 09.06 – 11.06, *headway* bus adalah 10 menit,
- Pukul 11.06 – 12.11, *headway* bus adalah 13 menit,
- Pukul 12.11 – 13.01, *headway* bus adalah 5 menit,
- Pukul 13.01 – 16.01, *headway* bus adalah 10 menit,
- Pukul 16.01 – 17.04, *headway* bus adalah 7 menit,
- Pukul 17.04 – 19.04, *headway* bus adalah 5 menit,
- Pukul 19.04 – 20.00, *headway* bus adalah 7 menit,
- Pukul 20.00 – 22.00, *headway* bus adalah 10 menit,
- Pukul 22.00 – 23.05, *headway* bus adalah 13 menit.

Dengan asumsi *headway* seperti di atas, kemudian dimisalkan waktu tempuh rute Dukuh Atas – Pulogadung adalah 45 menit dan waktu tempuh rute Dukuh Atas – Ragunan adalah 50 menit, dapat diperoleh *timetable* untuk jalur Dukuh Atas – Pulogadung, jalur Pulogadung – Dukuh Atas, jalur Dukuh Atas – Ragunan, dan jalur Ragunan – Dukuh Atas seperti yang tercantum dalam Lampiran 1. *Timetable* tersebut disusun berdasarkan waktu keberangkatan

kendaraan yang meningkat. Jika ada waktu keberangkatan yang sama dari jalur yang berbeda, maka urutan prioritasnya dibuat sebagai berikut:

- Jalur Dukuh Atas – Pulogadung
- Jalur Pulogadung – Dukuh Atas
- Jalur Dukuh Atas – Ragunan
- Jalur Ragunan – Dukuh Atas

Penetapan prioritas di atas berdasarkan waktu tempuh masing-masing jalur. Hal tersebut dilakukan agar kendaraan terlebih dahulu melakukan *trip* yang waktu tempuhnya lebih cepat meskipun pada akhirnya setiap *trip* tetap akan dilakukan.

Dari *timetable* pada Lampiran 1 baris pertama, diketahui bahwa kendaraan berangkat dari Dukuh Atas pukul 05.00 WIB dan tiba di Pulogadung pukul 05.45 WIB. Karena baris pertama *timetable* tersebut menyimpan data perpindahan kendaraan dari lokasi awal yang spesifik ke lokasi akhir yang spesifik pada waktu keberangkatan dan waktu tiba yang juga spesifik, maka baris tersebut merupakan *trip* 1. Secara keseluruhan, dari *timetable* pada Lampiran 1 diperoleh 584 *trip*.

Setelah diperoleh *timetable*, akan diberikan nilai untuk variabel-variabel yang telah didefinisikan pada masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar, yaitu pada Subbab 3.2.1. Nilai variabel-variabel ini disesuaikan dengan kondisi penjadwalan kendaraan bus TransJakarta. Masalah penjadwalan kendaraan bus TransJakarta akan diselesaikan menggunakan algoritma *auction* untuk masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan bantuan perangkat lunak MATLAB R2009.

4.3 Penentuan Nilai Variabel

Berikut ini adalah variabel-variabel yang digunakan dalam penjadwalan bus TransJakarta rute Dukuh Atas – Pulogadung dan rute Dukuh Atas – Ragunan dengan memperhatikan jadwal pengisian bahan bakar beserta nilainya:

a) $N = \{1, 2, ..., 584\}$ merupakan himpunan *trip* yang diperoleh dari *timetable* pada Lampiran 1,

b) $E = \{(i, j) \mid i, j \text{ compatible}, i, j \in N\} = \{(1, 30), (1, 32), (1, 34)...\}$

c) r = simpul untuk *depot* keberangkatan, t = simpul untuk *depot* kedatangan
Simpul r maupun t, pada program akan direpresentasikan sebagai simpul 585

d) s = simpul untuk SPBBG, yang dalam program direpresentasikan sebagai simpul 586

e) $V = \{1, 2, ..., 584, r, s, t\}$

f) $A = E \cup (\{r\} \times N) \cup (N \times \{s\}) \cup (\{s\} \times N) \cup \{s, t\}$

g) c_{ij} (dalam rupiah) dihitung seperti yang telah didefinisikan pada Subbab 3.2.1 poin h) dengan

- $c = 1.000.000$
- $c_j = \begin{cases} 21.700, & \text{jika lokasi keberangkatan } trip\ j \text{ alah Dukuh Atas dan lokasi kedatangannya ialah Pulogadung atau sebaliknya} \\ 24.800, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas dan lokasi kedatangannya ialah Ragunan atau sebaliknya} \end{cases}$ $\forall\, j \in N$
- $L_{rj} = 9.300$ $\forall\, j \in N$
- $L_{is} = \begin{cases} 21.700, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Dukuh Atas} \\ 46.500, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Ragunan} \\ 0, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Pulogadung} \end{cases}$ $\forall\, i \in N$
- $L_{sj} = \begin{cases} 21.700, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas} \\ 46.500, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Ragunan} \\ 0, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Pulogadung} \end{cases}$ $\forall\, j \in N$
- $L_{ij} = \begin{cases} 21.700, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Pulogadung dan lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas atau sebaliknya} \\ 24.800, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Ragunan dan lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas atau sebaliknya} \\ 46.500, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Pulogadung dan lokasi keberangkatan } trip\ j \text{ ialah Ragunan atau sebaliknya} \end{cases}$ $\forall\, i,j \in N$
- $L_{st} = 9.300$

h) $t_k = 15$ menit

i) $B_k = 120$ $k = 1, 2, ..., n$

j) $B(r, j) = 3$ $\forall\, j \in N$

k) $B(i,j) = \begin{cases} 7, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Pulogadung dan lokasi keberangkatan } trip\ j \text{ ialah ialah Dukuh Atas atau sebaliknya} \\ 8, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Ragunan dan lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas atau sebaliknya} \\ 15, & \text{jika lokasi kedatangan } trip\ i \text{ ialah Pulogadung dan lokasi keberangkatan } trip\ j \text{ ialah Ragunan atau sebaliknya} \end{cases}$ $\forall i, j \in N$

l) $B_j = \begin{cases} 7, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas dan lokasi kedatangan } trip\ j \text{ ialah Pulogadung atau sebaliknya} \\ 8, & \text{jika lokasi keberangkatan } trip\ j \text{ ialah Dukuh Atas dan lokasi kedatangan } trip\ j \text{ ialah Ragunan atau sebaliknya} \end{cases}$ $\forall j \in N$

m) $B(j,s) = \begin{cases} 7, & \text{jika lokasi kedatangan } trip\ j \text{ ialah Dukuh Atas} \\ 15, & \text{jika lokasi kedatangan } trip\ j \text{ ialah Ragunan} \\ 0, & \text{jika lokasi kedatangan } trip\ j \text{ ialah Pulogadung} \end{cases}$ $\forall j \in N$

n) kap = 120 LSP

Tahapan selanjutnya, *timetable* dan nilai-nilai variabel di atas menjadi masukan dalam program yang dijalankan menggunakan bantuan perangkat lunaks MATLAB R2009. *Source code* untuk penyelesaian masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction* ditampilkan dalam Lampiran 3 hingga Lampiran 6. Berikut ini dipaparkan mengenai keluaran yang diperoleh dari penyelesaian masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan bahan bakar.

4.4 Keluaran dari Penjadwalan Kendaraan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar Menggunakan Bantuan Perangkat Lunak MATLAB R2009

Keluaran yang diperoleh sebagai hasil penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction* ditampilkan pada Lampiran 2. Dari keluaran tersebut, akan

dilakukan analisa hasil penyelesaian masalah yang dipaparkan pada subbab selanjutnya.

4.5 Analisa Hasil Penjadwalan Kendaraan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Setiap baris dari tabel yang terdapat pada Lampiran 2 menunjukkan himpunan barisan perjalanan yang dijalankan oleh setiap kendaraan. Misal akan dijelaskan mengenai baris pertama dari tabel pada Lampiran 2. Dari baris pertama diperoleh S_1 = {585, 1, 30, 67, 112, 436, 586, 585}. S_1 artinya himpunan barisan perjalanan tersebut dijalankan oleh kendaraan 1. Elemen-elemen dalam S_1 adalah 585, 1, 30, 67, 112, 436, 586, 585 artinya kendaraan 1 memulai perjalanannya dari simpul 585, yaitu *depot* keberangkatan. Selanjutnya dari *depot* keberangkatan, kendaraan melakukan *trip* 1, yang berdasarkan Lampiran 1 berarti kendaraan berpindah dari Dukuh Atas pada pukul 05.00 WIB dan tiba Pulogadung pada pukul 05.45. Setelah melakukan *trip* 1, kendaraan menjalankan *trip* 30, yaitu berpindah Pulogadung pukul 05.49 dan tiba di Dukuh Atas pukul 06.34. Sebelum melakukan *trip* 30, kendaraan terlebih dahulu melakukan *deadhead* dari Pulogadung pada pukul 05.45 dan tiba di lokasi yang sama, yaitu Pulogadung pada pukul 05.49. Kemudian barisan perjalanan kendaraan 1 dilanjutkan dengan melakukan *trip* 67, *trip* 112, dan *trip* 436. Setelah selesai melakukan *trip* 436, tidak ada lagi *trip* yang dapat dijalankan oleh kendaraan 1 sehingga kendaraan akan kembali ke *depot*, yaitu simpul 585. Namun sesuai asumsi pada Subbab 3.1, sebelum kembali ke *depot* setelah melakukan *trip* terakhir, kendaraan diharuskan mengisi bahan bakar sehingga setelah melakukan *trip* 436, kendaraan mengunjungi simpul 586 terlebih dahulu kemudian ke *depot* kedatangan, yaitu simpul 585.

Penghitungan biaya yang dilakukan dalam proses penyelesaian masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction* ialah menghitung total biaya yang dikeluarkan karena menggunakan kendaraan, untuk melakukan barisan *trip*, dan untuk melakukan barisan *deadhead*. Sebagai contoh, akan dihitung biaya untuk

melakukan barisan perjalanan yang dijalankan kendaraan 1, yaitu S_1. Ketika berpindah dari *depot* keberangkatan ke *trip* 1, seperti definisi biaya pada Subbab 3.2.1, biaya yang dikeluarkan ialah biaya karena menggunakan kendaraan ditambah biaya untuk melakukan *deadhead* dari *depot* ke lokasi keberangkatan *trip* 1 ditambah biaya untuk melakukan *trip* 1. Sesuai pendefinisian biaya pada Subbab 4.3, biaya karena menggunakan kendaraan ialah Rp. 1.000.000, biaya untuk melakukan *deadhead* dari *depot* keberangkatan ke lokasi keberangkatan *trip* 1, yaitu Dukuh Atas, ialah Rp. 9.300, dan biaya untuk melakukan *trip* 1 ialah Rp. 21.700 karena lokasi keberangkatannya ialah Dukuh Atas dan lokasi kedatangannya ialah Pulogadung. Diperoleh total biaya untuk berpindah dari simpul 585 ke simpul 1 ialah

Rp. 1.000.000 + Rp. 9.300 + Rp. 21.700 = Rp. 1.031.000.

Dengan cara penghitungan tersebut, diperoleh biaya total sebesar Rp. 272.069.387,982980. Hasil penghitungan total biaya yang bukan merupakan bilangan bulat ialah karena ada faktor ε yang besarannya bukan merupakan bilangan bulat.

Salah satu contoh kendaraan yang melakukan *interlining* ialah kendaraan 2 yang melakukan *trip* 107 setelah melakukan *trip* 66. Hal tersebut dikarenakan *trip* 66 ialah perpindahan dari Pulogadung ke Dukuh Atas. Kemudian *trip* 107 ialah perpindahan dari Dukuh Atas ke Ragunan. Oleh karena itu kendaraan dikatakan melakukan *interlining* dari rute Dukuh Atas – Pulogadung menjadi rute Dukuh Atas – Ragunan.

Dari keluaran yang ditampilkan pada Lampiran 2 secara keseluruhan, akan disimpulkan hasil penjadwalan kendaraan bus TransJakarta untuk rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan dengan memperhatikan jadwal pengisian bahan bakar pada subbab berikut.

4.6 Kesimpulan Hasil Penjadwalan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar

Pendekatan algoritma *auction* untuk penjadwalan bus TransJakarta rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan dengan memperhatikan jadwal pengisian bahan bakar memberikan keluaran berupa barisan-barisan *trip* yang masing-masing harus dijalankan oleh suatu kendaraan, kapan suatu kendaraan harus mengisi bahan bakar, berapa banyak kendaraan yang diperlukan untuk memenuhi *timetable*, dan berapa biaya operasional yang dikeluarkan per hari. Dari tabel pada Lampiran 2 diperoleh bahwa untuk menjalankan 584 *trip* pada satu hari diperlukan 47 kendaraan bus TransJakarta. Bus TransJakarta ini dapat melakukan *interlining*, yaitu berubah rute dari rute Dukuh Atas – Pulogadung menjadi rute Dukuh Atas – Ragunan atau sebaliknya. Total biaya yang dikeluarkan untuk melakukan semua *trip* pada satu hari rute Dukuh Atas – Pulogadung dan Dukuh Atas – Ragunan dengan *timetable* yang tertera pada Lampiran 1 ialah Rp. 272.069.387,982980.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pada pembahasan mengenai penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar dan aplikasinya pada penjadwalan kendaraan bus TransJakarta, diperoleh kesimpulan bahwa masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar dapat dimodelkan menjadi masalah *quasi-assignment* dan dapat diselesaikan menggunakan algoritma *auction*. Algoritma *auction* yang digunakan untuk menyelesaikan masalah tersebut adalah algoritma *auction* yang telah dimodifikasi dengan menambahkan penghitungan sisa bahan bakar kendaraan dalam langkah-langkahnya. Kemudian penyelesaian masalah penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction* untuk masalah tersebut dapat diaplikasikan untuk menyelesaikan masalah penjadwalan kendaraan bus TransJakarta.

Dari hasil penyelesaian masalah penjadwalan kendaraan bus TransJakarta dengan memperhatikan jadwal pengisian bahan bakar menggunakan algoritma *auction*, diperoleh kesimpulan bahwa hanya diperlukan 47 kendaraan untuk menjalankan seluruh *trip* per hari dari rute Dukuh Atas – Pulogadung dan rute Pulogadung Dukuh Atas dengan total biaya operasional sebesar Rp. 272.069.387,982980.

5.2 Saran

Berikut ini saran bagi penelitian lebih lanjut mengenai pendekatan algoritma *auction* untuk penjadwalan kendaraan bus *rapid transit* dengan memperhatikan jadwal pengisian bahan bakar:

- Pada tugas akhir ini, tahapan *reverse auction* hanya digunakan untuk memasangkan setiap *trip* pertama suatu kendaraan dengan *depot* keberangkatan. Untuk penelitian selanjutnya, *reverse auction* dapat dimodifikasi sedemikian sehingga tahapan *reverse auction* juga dapat digunakan untuk mencari *predecessor* suatu *trip* yang berupa *trip* lain, bukan hanya *depot* keberangkatan dengan tetap memperhatikan jadwal pengisian bahan bakar. Dengan tahapan *reverse auction* tersebut, keluaran yang diperoleh mungkin berbeda dengan keluaran pada tugas akhir ini. Namun keluaran dari program dengan tahapan *reverse auction* yang dimodifikasi mungkin lebih baik dari keluaran pada tugas akhir ini karena *predecessor* setiap *trip* diperiksa dengan memperhatikan jadwal pengisian bahan bakar kendaraan.
- Masalah penjadwalan kendaraan yang diselesaikan pada tugas akhir ini merupakan masalah penjadwalan kendaraan dengan satu *depot*. Untuk masalah yang lebih kompleks, pada penelitian selanjutnya dapat dilakukan penjadwalan kendaraan dengan lebih dari satu *depot*.

DAFTAR PUSTAKA

Akbar, M. *Sambutan Kepala Unit Pengelola TransJakarta Busway*, 1 (3). Maret 4, 2012. pk. 20.44 WIB. http://transjakarta.co.id/page.php

Bertsekas, D. P. (1992). *Auction Algorithms for Network Flow Problems: A Tutorial Introduction*. Cambridge: Massachusetts Institute of Technology.

Dantzig, G. B. dan Thapa, M. N. (1997). *Linear Programming* 1 : *Introduction*. New York: Springer-Verlag.

Freling, R., Wagelmans, A. P. M., dan Paixão, J. M. P. (2001). Model and Algorithms for Vehicle Scheduling. *Transportation Science*, 35, 165 – 180.

Hillier, F. S. dan Lieberman, G. J. (1980). *Introduction to Operations Research*. San Francisco: Holden-Day.

Huisman, D. (2004). *Integrated and Dynamic Vehicle and Crew Scheduling* (Vol. 325). Netherlands: Tinbergen Institute Research Series.

Schwarzenegger, McPeak, and Kempton. (2006). *Bus Rapid Transit*, *A Handbook for Partners*. California: Caltrans.

Tarjuki, M. T. dan Nurachman. *Banjir & Kemacetan Lalu Lintas*. Juli 23, 2012. pk. 12.27 WIB. www.jakarta.go.id/web/news/2011/10/banjir-kemacetan-lalu-lintas

Wolsey, L. A. (1998). *Integer Programming*. New Jersey: John Wiley & Sons.

LAMPIRAN

Lampiran 1. *Timetable* untuk jalur Dukuh Atas – Pulogadung, jalur Pulogadung – Dukuh Atas, jalur Dukuh Atas – Ragunan, dan jalur Ragunan – Dukuh Atas

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
1.	Dukuh Atas	Pulogadung	05.00	05.45
2.	Pulogadung	Dukuh Atas	05.00	05.45
3.	Dukuh Atas	Ragunan	05.00	05.50
4.	Ragunan	Dukuh Atas	05.00	05.50
5.	Dukuh Atas	Pulogadung	05.07	05.52
6.	Pulogadung	Dukuh Atas	05.07	05.52
7.	Dukuh Atas	Ragunan	05.07	05.57
8.	Ragunan	Dukuh Atas	05.07	05.57
9.	Dukuh Atas	Pulogadung	05.14	05.59
10.	Pulogadung	Dukuh Atas	05.14	05.59
11.	Dukuh Atas	Ragunan	05.14	06.04
12.	Ragunan	Dukuh Atas	05.14	06.04
13.	Dukuh Atas	Pulogadung	05.21	06.06
14.	Pulogadung	Dukuh Atas	05.21	06.06
15.	Dukuh Atas	Ragunan	05.21	06.11
16.	Ragunan	Dukuh Atas	05.21	06.11
17.	Dukuh Atas	Pulogadung	05.28	06.13
18.	Pulogadung	Dukuh Atas	05.28	06.13
19.	Dukuh Atas	Ragunan	05.28	06.18
20.	Ragunan	Dukuh Atas	05.28	06.18
21.	Dukuh Atas	Pulogadung	05.35	06.20
22.	Pulogadung	Dukuh Atas	05.35	06.20
23.	Dukuh Atas	Ragunan	05.35	06.25
24.	Ragunan	Dukuh Atas	05.35	06.25

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
25.	Dukuh Atas	Pulogadung	05.42	06.27
26.	Pulogadung	Dukuh Atas	05.42	06.27
27.	Dukuh Atas	Ragunan	05.42	06.32
28.	Ragunan	Dukuh Atas	05.42	06.32
29.	Dukuh Atas	Pulogadung	05.49	06.34
30.	Pulogadung	Dukuh Atas	05.49	06.34
31.	Dukuh Atas	Ragunan	05.49	06.39
32.	Ragunan	Dukuh Atas	05.49	06.39
33.	Dukuh Atas	Pulogadung	05.56	06.41
34.	Pulogadung	Dukuh Atas	05.56	06.41
35.	Dukuh Atas	Ragunan	05.56	06.46
36.	Ragunan	Dukuh Atas	05.56	06.46
37.	Dukuh Atas	Pulogadung	06.03	06.48
38.	Pulogadung	Dukuh Atas	06.03	06.48
39.	Dukuh Atas	Ragunan	06.03	06.53
40.	Ragunan	Dukuh Atas	06.03	06.53
41.	Dukuh Atas	Pulogadung	06.08	06.53
42.	Pulogadung	Dukuh Atas	06.08	06.53
43.	Dukuh Atas	Ragunan	06.08	06.58
44.	Ragunan	Dukuh Atas	06.08	06.58
45.	Dukuh Atas	Pulogadung	06.13	06.58
46.	Pulogadung	Dukuh Atas	06.13	06.58
47.	Dukuh Atas	Ragunan	06.13	07.03
48.	Ragunan	Dukuh Atas	06.13	07.03
49.	Dukuh Atas	Pulogadung	06.18	07.03
50.	Pulogadung	Dukuh Atas	06.18	07.03
51.	Dukuh Atas	Ragunan	06.18	07.08

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
52.	Ragunan	Dukuh Atas	06.18	07.08
53.	Dukuh Atas	Pulogadung	06.23	07.08
54.	Pulogadung	Dukuh Atas	06.23	07.08
55.	Dukuh Atas	Ragunan	06.23	07.13
56.	Ragunan	Dukuh Atas	06.23	07.13
57.	Dukuh Atas	Pulogadung	06.28	07.13
58.	Pulogadung	Dukuh Atas	06.28	07.13
59.	Dukuh Atas	Ragunan	06.28	07.18
60.	Ragunan	Dukuh Atas	06.28	07.18
61.	Dukuh Atas	Pulogadung	06.33	07.18
62.	Pulogadung	Dukuh Atas	06.33	07.18
63.	Dukuh Atas	Ragunan	06.33	07.23
64.	Ragunan	Dukuh Atas	06.33	07.23
65.	Dukuh Atas	Pulogadung	06.38	07.23
66.	Pulogadung	Dukuh Atas	06.38	07.23
67.	Dukuh Atas	Ragunan	06.38	07.28
68.	Ragunan	Dukuh Atas	06.38	07.28
69.	Dukuh Atas	Pulogadung	06.43	07.28
70.	Pulogadung	Dukuh Atas	06.43	07.28
71.	Dukuh Atas	Ragunan	06.43	07.33
72.	Ragunan	Dukuh Atas	06.43	07.33
73.	Dukuh Atas	Pulogadung	06.48	07.33
74.	Pulogadung	Dukuh Atas	06.48	07.33
75.	Dukuh Atas	Ragunan	06.48	07.38
76.	Ragunan	Dukuh Atas	06.48	07.38
77.	Dukuh Atas	Pulogadung	06.53	07.38
78.	Pulogadung	Dukuh Atas	06.53	07.38

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
79.	Dukuh Atas	Ragunan	06.53	07.43
80.	Ragunan	Dukuh Atas	06.53	07.43
81.	Dukuh Atas	Pulogadung	06.58	07.43
82.	Pulogadung	Dukuh Atas	06.58	07.43
83.	Dukuh Atas	Ragunan	06.58	07.48
84.	Ragunan	Dukuh Atas	06.58	07.48
85.	Dukuh Atas	Pulogadung	07.03	07.48
86.	Pulogadung	Dukuh Atas	07.03	07.48
87.	Dukuh Atas	Ragunan	07.03	07.53
88.	Ragunan	Dukuh Atas	07.03	07.53
89.	Dukuh Atas	Pulogadung	07.08	07.53
90.	Pulogadung	Dukuh Atas	07.08	07.53
91.	Dukuh Atas	Ragunan	07.08	07.58
92.	Ragunan	Dukuh Atas	07.08	07.58
93.	Dukuh Atas	Pulogadung	07.13	07.58
94.	Pulogadung	Dukuh Atas	07.13	07.58
95.	Dukuh Atas	Ragunan	07.13	08.03
96.	Ragunan	Dukuh Atas	07.13	08.03
97.	Dukuh Atas	Pulogadung	07.18	08.03
98.	Pulogadung	Dukuh Atas	07.18	08.03
99.	Dukuh Atas	Ragunan	07.18	08.08
100.	Ragunan	Dukuh Atas	07.18	08.08
101.	Dukuh Atas	Pulogadung	07.23	08.08
102.	Pulogadung	Dukuh Atas	07.23	08.08
103.	Dukuh Atas	Ragunan	07.23	08.13
104.	Ragunan	Dukuh Atas	07.23	08.13
105.	Dukuh Atas	Pulogadung	07.28	08.13
106.	Pulogadung	Dukuh Atas	07.28	08.13

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
107.	Dukuh Atas	Ragunan	07.28	08.18
108.	Ragunan	Dukuh Atas	07.28	08.18
109.	Dukuh Atas	Pulogadung	07.33	08.18
110.	Pulogadung	Dukuh Atas	07.33	08.18
111.	Dukuh Atas	Ragunan	07.33	08.23
112.	Ragunan	Dukuh Atas	07.33	08.23
113.	Dukuh Atas	Pulogadung	07.38	08.23
114.	Pulogadung	Dukuh Atas	07.38	08.23
115.	Dukuh Atas	Ragunan	07.38	08.28
116.	Ragunan	Dukuh Atas	07.38	08.28
117.	Dukuh Atas	Pulogadung	07.43	08.28
118.	Pulogadung	Dukuh Atas	07.43	08.28
119.	Dukuh Atas	Ragunan	07.43	08.33
120.	Ragunan	Dukuh Atas	07.43	08.33
121.	Dukuh Atas	Pulogadung	07.48	08.33
122.	Pulogadung	Dukuh Atas	07.48	08.33
123.	Dukuh Atas	Ragunan	07.48	08.38
124.	Ragunan	Dukuh Atas	07.48	08.38
125.	Dukuh Atas	Pulogadung	07.53	08.38
126.	Pulogadung	Dukuh Atas	07.53	08.38
127.	Dukuh Atas	Ragunan	07.53	08.43
128.	Ragunan	Dukuh Atas	07.53	08.43
129.	Dukuh Atas	Pulogadung	07.58	08.43
130.	Pulogadung	Dukuh Atas	07.58	08.43
131.	Dukuh Atas	Ragunan	07.58	08.48
132.	Ragunan	Dukuh Atas	07.58	08.48
133.	Dukuh Atas	Pulogadung	08.03	08.48

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
134.	Pulogadung	Dukuh Atas	08.03	08.48
135.	Dukuh Atas	Ragunan	08.03	08.53
136.	Ragunan	Dukuh Atas	08.03	08.53
137.	Dukuh Atas	Pulogadung	08.10	08.55
138.	Pulogadung	Dukuh Atas	08.10	08.55
139.	Dukuh Atas	Ragunan	08.10	09.00
140.	Ragunan	Dukuh Atas	08.10	09.00
141.	Dukuh Atas	Pulogadung	08.17	09.02
142.	Pulogadung	Dukuh Atas	08.17	09.02
143.	Dukuh Atas	Ragunan	08.17	09.07
144.	Ragunan	Dukuh Atas	08.17	09.07
145.	Dukuh Atas	Pulogadung	08.24	09.09
146.	Pulogadung	Dukuh Atas	08.24	09.09
147.	Dukuh Atas	Ragunan	08.24	09.14
148.	Ragunan	Dukuh Atas	08.24	09.14
149.	Dukuh Atas	Pulogadung	08.31	09.16
150.	Pulogadung	Dukuh Atas	08.31	09.16
151.	Dukuh Atas	Ragunan	08.31	09.21
152.	Ragunan	Dukuh Atas	08.31	09.21
153.	Dukuh Atas	Pulogadung	08.38	09.23
154.	Pulogadung	Dukuh Atas	08.38	09.23
155.	Dukuh Atas	Ragunan	08.38	09.28
156.	Ragunan	Dukuh Atas	08.38	09.28
157.	Dukuh Atas	Pulogadung	08.45	09.30
158.	Pulogadung	Dukuh Atas	08.45	09.30
159.	Dukuh Atas	Ragunan	08.45	09.35
160.	Ragunan	Dukuh Atas	08.45	09.35
161.	Dukuh Atas	Pulogadung	08.52	09.37

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
162.	Pulogadung	Dukuh Atas	08.52	09.37
163.	Dukuh Atas	Ragunan	08.52	09.42
164.	Ragunan	Dukuh Atas	08.52	09.42
165.	Dukuh Atas	Pulogadung	08.59	09.44
166.	Pulogadung	Dukuh Atas	08.59	09.44
167.	Dukuh Atas	Ragunan	08.59	09.49
168.	Ragunan	Dukuh Atas	08.59	09.49
169.	Dukuh Atas	Pulogadung	09.06	09.51
170.	Pulogadung	Dukuh Atas	09.06	09.51
171.	Dukuh Atas	Ragunan	09.06	09.56
172.	Ragunan	Dukuh Atas	09.06	09.56
173.	Dukuh Atas	Pulogadung	09.16	10.01
174.	Pulogadung	Dukuh Atas	09.16	10.01
175.	Dukuh Atas	Ragunan	09.16	10.06
176.	Ragunan	Dukuh Atas	09.16	10.06
177.	Dukuh Atas	Pulogadung	09.26	10.11
178.	Pulogadung	Dukuh Atas	09.26	10.11
179.	Dukuh Atas	Ragunan	09.26	10.16
180.	Ragunan	Dukuh Atas	09.26	10.16
181.	Dukuh Atas	Pulogadung	09.36	10.21
182.	Pulogadung	Dukuh Atas	09.36	10.21
183.	Dukuh Atas	Ragunan	09.36	10.26
184.	Ragunan	Dukuh Atas	09.36	10.26
185.	Dukuh Atas	Pulogadung	09.46	10.31
186.	Pulogadung	Dukuh Atas	09.46	10.31
187.	Dukuh Atas	Ragunan	09.46	10.36
188.	Ragunan	Dukuh Atas	09.46	10.36

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
189.	Dukuh Atas	Pulogadung	09.56	10.41
190.	Pulogadung	Dukuh Atas	09.56	10.41
191.	Dukuh Atas	Ragunan	09.56	10.46
192.	Ragunan	Dukuh Atas	09.56	10.46
193.	Dukuh Atas	Pulogadung	10.06	10.51
194.	Pulogadung	Dukuh Atas	10.06	10.51
195.	Dukuh Atas	Ragunan	10.06	10.56
196.	Ragunan	Dukuh Atas	10.06	10.56
197.	Dukuh Atas	Pulogadung	10.16	11.01
198.	Pulogadung	Dukuh Atas	10.16	11.01
199.	Dukuh Atas	Ragunan	10.16	11.06
200.	Ragunan	Dukuh Atas	10.16	11.06
201.	Dukuh Atas	Pulogadung	10.26	11.11
202.	Pulogadung	Dukuh Atas	10.26	11.11
203.	Dukuh Atas	Ragunan	10.26	11.16
204.	Ragunan	Dukuh Atas	10.26	11.16
205.	Dukuh Atas	Pulogadung	10.36	11.21
206.	Pulogadung	Dukuh Atas	10.36	11.21
207.	Dukuh Atas	Ragunan	10.36	11.26
208.	Ragunan	Dukuh Atas	10.36	11.26
209.	Dukuh Atas	Pulogadung	10.46	11.31
210.	Pulogadung	Dukuh Atas	10.46	11.31
211.	Dukuh Atas	Ragunan	10.46	11.36
212.	Ragunan	Dukuh Atas	10.46	11.36
213.	Dukuh Atas	Pulogadung	10.56	11.41
214.	Pulogadung	Dukuh Atas	10.56	11.41
215.	Dukuh Atas	Ragunan	10.56	11.46

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
216.	Ragunan	Dukuh Atas	10.56	11.46
217.	Dukuh Atas	Pulogadung	11.06	11.51
218.	Pulogadung	Dukuh Atas	11.06	11.51
219.	Dukuh Atas	Ragunan	11.06	11.56
220.	Ragunan	Dukuh Atas	11.06	11.56
221.	Dukuh Atas	Pulogadung	11.19	12.04
222.	Pulogadung	Dukuh Atas	11.19	12.04
223.	Dukuh Atas	Ragunan	11.19	12.09
224.	Ragunan	Dukuh Atas	11.19	12.09
225.	Dukuh Atas	Pulogadung	11.32	12.17
226.	Pulogadung	Dukuh Atas	11.32	12.17
227.	Dukuh Atas	Ragunan	11.32	12.22
228.	Ragunan	Dukuh Atas	11.32	12.22
229.	Dukuh Atas	Pulogadung	11.45	12.30
230.	Pulogadung	Dukuh Atas	11.45	12.30
231.	Dukuh Atas	Ragunan	11.45	12.35
232.	Ragunan	Dukuh Atas	11.45	12.35
233.	Dukuh Atas	Pulogadung	11.58	12.43
234.	Pulogadung	Dukuh Atas	11.58	12.43
235.	Dukuh Atas	Ragunan	11.58	12.48
236.	Ragunan	Dukuh Atas	11.58	12.48
237.	Dukuh Atas	Pulogadung	12.11	12.56
238.	Pulogadung	Dukuh Atas	12.11	12.56
239.	Dukuh Atas	Ragunan	12.11	13.01
240.	Ragunan	Dukuh Atas	12.11	13.01
241.	Dukuh Atas	Pulogadung	12.16	13.01
242.	Pulogadung	Dukuh Atas	12.16	13.01

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
243.	Dukuh Atas	Ragunan	12.16	13.06
244.	Ragunan	Dukuh Atas	12.16	13.06
245.	Dukuh Atas	Pulogadung	12.21	13.06
246.	Pulogadung	Dukuh Atas	12.21	13.06
247.	Dukuh Atas	Ragunan	12.21	13.11
248.	Ragunan	Dukuh Atas	12.21	13.11
249.	Dukuh Atas	Pulogadung	12.26	13.11
250.	Pulogadung	Dukuh Atas	12.26	13.11
251.	Dukuh Atas	Ragunan	12.26	13.16
252.	Ragunan	Dukuh Atas	12.26	13.16
253.	Dukuh Atas	Pulogadung	12.31	13.16
254.	Pulogadung	Dukuh Atas	12.31	13.16
255.	Dukuh Atas	Ragunan	12.31	13.21
256.	Ragunan	Dukuh Atas	12.31	13.21
257.	Dukuh Atas	Pulogadung	12.36	13.21
258.	Pulogadung	Dukuh Atas	12.36	13.21
259.	Dukuh Atas	Ragunan	12.36	13.26
260.	Ragunan	Dukuh Atas	12.36	13.26
261.	Dukuh Atas	Pulogadung	12.41	13.26
262.	Pulogadung	Dukuh Atas	12.41	13.26
263.	Dukuh Atas	Ragunan	12.41	13.31
264.	Ragunan	Dukuh Atas	12.41	13.31
265.	Dukuh Atas	Pulogadung	12.46	13.31
266.	Pulogadung	Dukuh Atas	12.46	13.31
267.	Dukuh Atas	Ragunan	12.46	13.36
268.	Ragunan	Dukuh Atas	12.46	13.36
269.	Dukuh Atas	Pulogadung	12.51	13.36

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
270.	Pulogadung	Dukuh Atas	12.51	13.36
271.	Dukuh Atas	Ragunan	12.51	13.41
272.	Ragunan	Dukuh Atas	12.51	13.41
273.	Dukuh Atas	Pulogadung	12.56	13.41
274.	Pulogadung	Dukuh Atas	12.56	13.41
275.	Dukuh Atas	Ragunan	12.56	13.46
276.	Ragunan	Dukuh Atas	12.56	13.46
277.	Dukuh Atas	Pulogadung	13.01	13.46
278.	Pulogadung	Dukuh Atas	13.01	13.46
279.	Dukuh Atas	Ragunan	13.01	13.51
280.	Ragunan	Dukuh Atas	13.01	13.51
281.	Dukuh Atas	Pulogadung	13.11	13.56
282.	Pulogadung	Dukuh Atas	13.11	13.56
283.	Dukuh Atas	Ragunan	13.11	14.01
284.	Ragunan	Dukuh Atas	13.11	14.01
285.	Dukuh Atas	Pulogadung	13.21	14.06
286.	Pulogadung	Dukuh Atas	13.21	14.06
287.	Dukuh Atas	Ragunan	13.21	14.11
288.	Ragunan	Dukuh Atas	13.21	14.11
289.	Dukuh Atas	Pulogadung	13.31	14.16
290.	Pulogadung	Dukuh Atas	13.31	14.16
291.	Dukuh Atas	Ragunan	13.31	14.21
292.	Ragunan	Dukuh Atas	13.31	14.21
293.	Dukuh Atas	Pulogadung	13.41	14.26
294.	Pulogadung	Dukuh Atas	13.41	14.26
295.	Dukuh Atas	Ragunan	13.41	14.31
296.	Ragunan	Dukuh Atas	13.41	14.31

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
297.	Dukuh Atas	Pulogadung	13.51	14.36
298.	Pulogadung	Dukuh Atas	13.51	14.36
299.	Dukuh Atas	Ragunan	13.51	14.41
300.	Ragunan	Dukuh Atas	13.51	14.41
301.	Dukuh Atas	Pulogadung	14.01	14.46
302.	Pulogadung	Dukuh Atas	14.01	14.46
303.	Dukuh Atas	Ragunan	14.01	14.51
304.	Ragunan	Dukuh Atas	14.01	14.51
305.	Dukuh Atas	Pulogadung	14.11	14.56
306.	Pulogadung	Dukuh Atas	14.11	14.56
307.	Dukuh Atas	Ragunan	14.11	15.01
308.	Ragunan	Dukuh Atas	14.11	15.01
309.	Dukuh Atas	Pulogadung	14.21	15.06
310.	Pulogadung	Dukuh Atas	14.21	15.06
311.	Dukuh Atas	Ragunan	14.21	15.11
312.	Ragunan	Dukuh Atas	14.21	15.11
313.	Dukuh Atas	Pulogadung	14.31	15.16
314.	Pulogadung	Dukuh Atas	14.31	15.16
315.	Dukuh Atas	Ragunan	14.31	15.21
316.	Ragunan	Dukuh Atas	14.31	15.21
317.	Dukuh Atas	Pulogadung	14.41	15.26
318.	Pulogadung	Dukuh Atas	14.41	15.26
319.	Dukuh Atas	Ragunan	14.41	15.31
320.	Ragunan	Dukuh Atas	14.41	15.31
321.	Dukuh Atas	Pulogadung	14.51	15.36
322.	Pulogadung	Dukuh Atas	14.51	15.36
323.	Dukuh Atas	Ragunan	14.51	15.41

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
324.	Ragunan	Dukuh Atas	14.51	15.41
325.	Dukuh Atas	Pulogadung	15.01	15.46
326.	Pulogadung	Dukuh Atas	15.01	15.46
327.	Dukuh Atas	Ragunan	15.01	15.51
328.	Ragunan	Dukuh Atas	15.01	15.51
329.	Dukuh Atas	Pulogadung	15.11	15.56
330.	Pulogadung	Dukuh Atas	15.11	15.56
331.	Dukuh Atas	Ragunan	15.11	16.01
332.	Ragunan	Dukuh Atas	15.11	16.01
333.	Dukuh Atas	Pulogadung	15.21	16.06
334.	Pulogadung	Dukuh Atas	15.21	16.06
335.	Dukuh Atas	Ragunan	15.21	16.11
336.	Ragunan	Dukuh Atas	15.21	16.11
337.	Dukuh Atas	Pulogadung	15.31	16.16
338.	Pulogadung	Dukuh Atas	15.31	16.16
339.	Dukuh Atas	Ragunan	15.31	16.21
340.	Ragunan	Dukuh Atas	15.31	16.21
341.	Dukuh Atas	Pulogadung	15.41	16.26
342.	Pulogadung	Dukuh Atas	15.41	16.26
343.	Dukuh Atas	Ragunan	15.41	16.31
344.	Ragunan	Dukuh Atas	15.41	16.31
345.	Dukuh Atas	Pulogadung	15.51	16.36
346.	Pulogadung	Dukuh Atas	15.51	16.36
347.	Dukuh Atas	Ragunan	15.51	16.41
348.	Ragunan	Dukuh Atas	15.51	16.41
349.	Dukuh Atas	Pulogadung	16.01	16.46
350.	Pulogadung	Dukuh Atas	16.01	16.46

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
351.	Dukuh Atas	Ragunan	16.01	16.51
352.	Ragunan	Dukuh Atas	16.01	16.51
353.	Dukuh Atas	Pulogadung	16.08	16.53
354.	Pulogadung	Dukuh Atas	16.08	16.53
355.	Dukuh Atas	Ragunan	16.08	16.58
356.	Ragunan	Dukuh Atas	16.08	16.58
357.	Dukuh Atas	Pulogadung	16.15	17.00
358.	Pulogadung	Dukuh Atas	16.15	17.00
359.	Dukuh Atas	Ragunan	16.15	17.05
360.	Ragunan	Dukuh Atas	16.15	17.05
361.	Dukuh Atas	Pulogadung	16.22	17.07
362.	Pulogadung	Dukuh Atas	16.22	17.07
363.	Dukuh Atas	Ragunan	16.22	17.12
364.	Ragunan	Dukuh Atas	16.22	17.12
365.	Dukuh Atas	Pulogadung	16.29	17.14
366.	Pulogadung	Dukuh Atas	16.29	17.14
367.	Dukuh Atas	Ragunan	16.29	17.19
368.	Ragunan	Dukuh Atas	16.29	17.19
369.	Dukuh Atas	Pulogadung	16.36	17.21
370.	Pulogadung	Dukuh Atas	16.36	17.21
371.	Dukuh Atas	Ragunan	16.36	17.26
372.	Ragunan	Dukuh Atas	16.36	17.26
373.	Dukuh Atas	Pulogadung	16.43	17.28
374.	Pulogadung	Dukuh Atas	16.43	17.28
375.	Dukuh Atas	Ragunan	16.43	17.33
376.	Ragunan	Dukuh Atas	16.43	17.33
377.	Dukuh Atas	Pulogadung	16.50	17.35

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
378.	Pulogadung	Dukuh Atas	16.50	17.35
379.	Dukuh Atas	Ragunan	16.50	17.40
380.	Ragunan	Dukuh Atas	16.50	17.40
381.	Dukuh Atas	Pulogadung	16.57	17.42
382.	Pulogadung	Dukuh Atas	16.57	17.42
383.	Dukuh Atas	Ragunan	16.57	17.47
384.	Ragunan	Dukuh Atas	16.57	17.47
385.	Dukuh Atas	Pulogadung	17.04	17.49
386.	Pulogadung	Dukuh Atas	17.04	17.49
387.	Dukuh Atas	Ragunan	17.04	17.54
388.	Ragunan	Dukuh Atas	17.04	17.54
389.	Dukuh Atas	Pulogadung	17.09	17.54
390.	Pulogadung	Dukuh Atas	17.09	17.54
391.	Dukuh Atas	Ragunan	17.09	17.59
392.	Ragunan	Dukuh Atas	17.09	17.59
393.	Dukuh Atas	Pulogadung	17.14	17.59
394.	Pulogadung	Dukuh Atas	17.14	17.59
395.	Dukuh Atas	Ragunan	17.14	18.04
396.	Ragunan	Dukuh Atas	17.14	18.04
397.	Dukuh Atas	Pulogadung	17.19	18.04
398.	Pulogadung	Dukuh Atas	17.19	18.04
399.	Dukuh Atas	Ragunan	17.19	18.09
400.	Ragunan	Dukuh Atas	17.19	18.09
401.	Dukuh Atas	Pulogadung	17.24	18.09
402.	Pulogadung	Dukuh Atas	17.24	18.09
403.	Dukuh Atas	Ragunan	17.24	18.14
404.	Ragunan	Dukuh Atas	17.24	18.14

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
405.	Dukuh Atas	Pulogadung	17.29	18.14
406.	Pulogadung	Dukuh Atas	17.29	18.14
407.	Dukuh Atas	Ragunan	17.29	18.19
408.	Ragunan	Dukuh Atas	17.29	18.19
409.	Dukuh Atas	Pulogadung	17.34	18.19
410.	Pulogadung	Dukuh Atas	17.34	18.19
411.	Dukuh Atas	Ragunan	17.34	18.24
412.	Ragunan	Dukuh Atas	17.34	18.24
413.	Dukuh Atas	Pulogadung	17.39	18.24
414.	Pulogadung	Dukuh Atas	17.39	18.24
415.	Dukuh Atas	Ragunan	17.39	18.29
416.	Ragunan	Dukuh Atas	17.39	18.29
417.	Dukuh Atas	Pulogadung	17.44	18.29
418.	Pulogadung	Dukuh Atas	17.44	18.29
419.	Dukuh Atas	Ragunan	17.44	18.34
420.	Ragunan	Dukuh Atas	17.44	18.34
421.	Dukuh Atas	Pulogadung	17.49	18.34
422.	Pulogadung	Dukuh Atas	17.49	18.34
423.	Dukuh Atas	Ragunan	17.49	18.39
424.	Ragunan	Dukuh Atas	17.49	18.39
425.	Dukuh Atas	Pulogadung	17.54	18.39
426.	Pulogadung	Dukuh Atas	17.54	18.39
427.	Dukuh Atas	Ragunan	17.54	18.44
428.	Ragunan	Dukuh Atas	17.54	18.44
429.	Dukuh Atas	Pulogadung	17.59	18.44
430.	Pulogadung	Dukuh Atas	17.59	18.44
431.	Dukuh Atas	Ragunan	17.59	18.49

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
432.	Ragunan	Dukuh Atas	17.59	18.49
433.	Dukuh Atas	Pulogadung	18.04	18.49
434.	Pulogadung	Dukuh Atas	18.04	18.49
435.	Dukuh Atas	Ragunan	18.04	18.54
436.	Ragunan	Dukuh Atas	18.04	18.54
437.	Dukuh Atas	Pulogadung	18.09	18.54
438.	Pulogadung	Dukuh Atas	18.09	18.54
439.	Dukuh Atas	Ragunan	18.09	18.59
440.	Ragunan	Dukuh Atas	18.09	18.59
441.	Dukuh Atas	Pulogadung	18.14	18.59
442.	Pulogadung	Dukuh Atas	18.14	18.59
443.	Dukuh Atas	Ragunan	18.14	19.04
444.	Ragunan	Dukuh Atas	18.14	19.04
445.	Dukuh Atas	Pulogadung	18.19	19.04
446.	Pulogadung	Dukuh Atas	18.19	19.04
447.	Dukuh Atas	Ragunan	18.19	19.09
448.	Ragunan	Dukuh Atas	18.19	19.09
449.	Dukuh Atas	Pulogadung	18.24	19.09
450.	Pulogadung	Dukuh Atas	18.24	19.09
451.	Dukuh Atas	Ragunan	18.24	19.14
452.	Ragunan	Dukuh Atas	18.24	19.14
453.	Dukuh Atas	Pulogadung	18.29	19.14
454.	Pulogadung	Dukuh Atas	18.29	19.14
455.	Dukuh Atas	Ragunan	18.29	19.19
456.	Ragunan	Dukuh Atas	18.29	19.19
457.	Dukuh Atas	Pulogadung	18.34	19.19
458.	Pulogadung	Dukuh Atas	18.34	19.19

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
459.	Dukuh Atas	Ragunan	18.34	19.24
460.	Ragunan	Dukuh Atas	18.34	19.24
461.	Dukuh Atas	Pulogadung	18.39	19.24
462.	Pulogadung	Dukuh Atas	18.39	19.24
463.	Dukuh Atas	Ragunan	18.39	19.29
464.	Ragunan	Dukuh Atas	18.39	19.29
465.	Dukuh Atas	Pulogadung	18.44	19.29
466.	Pulogadung	Dukuh Atas	18.44	19.29
467.	Dukuh Atas	Ragunan	18.44	19.34
468.	Ragunan	Dukuh Atas	18.44	19.34
469.	Dukuh Atas	Pulogadung	18.49	19.34
470.	Pulogadung	Dukuh Atas	18.49	19.34
471.	Dukuh Atas	Ragunan	18.49	19.39
472.	Ragunan	Dukuh Atas	18.49	19.39
473.	Dukuh Atas	Pulogadung	18.54	19.39
474.	Pulogadung	Dukuh Atas	18.54	19.39
475.	Dukuh Atas	Ragunan	18.54	19.44
476.	Ragunan	Dukuh Atas	18.54	19.44
477.	Dukuh Atas	Pulogadung	18.59	19.44
478.	Pulogadung	Dukuh Atas	18.59	19.44
479.	Dukuh Atas	Ragunan	18.59	19.49
480.	Ragunan	Dukuh Atas	18.59	19.49
481.	Dukuh Atas	Pulogadung	19.04	19.49
482.	Pulogadung	Dukuh Atas	19.04	19.49
483.	Dukuh Atas	Ragunan	19.04	19.54
484.	Ragunan	Dukuh Atas	19.04	19.54
485.	Dukuh Atas	Pulogadung	19.11	19.56

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
486.	Pulogadung	Dukuh Atas	19.11	19.56
487.	Dukuh Atas	Ragunan	19.11	20.01
488.	Ragunan	Dukuh Atas	19.11	20.01
489.	Dukuh Atas	Pulogadung	19.18	20.03
490.	Pulogadung	Dukuh Atas	19.18	20.03
491.	Dukuh Atas	Ragunan	19.18	20.08
492.	Ragunan	Dukuh Atas	19.18	20.08
493.	Dukuh Atas	Pulogadung	19.25	20.10
494.	Pulogadung	Dukuh Atas	19.25	20.10
495.	Dukuh Atas	Ragunan	19.25	20.15
496.	Ragunan	Dukuh Atas	19.25	20.15
497.	Dukuh Atas	Pulogadung	19.32	20.17
498.	Pulogadung	Dukuh Atas	19.32	20.17
499.	Dukuh Atas	Ragunan	19.32	20.22
500.	Ragunan	Dukuh Atas	19.32	20.22
501.	Dukuh Atas	Pulogadung	19.39	20.24
502.	Pulogadung	Dukuh Atas	19.39	20.24
503.	Dukuh Atas	Ragunan	19.39	20.29
504.	Ragunan	Dukuh Atas	19.39	20.29
505.	Dukuh Atas	Pulogadung	19.46	20.31
506.	Pulogadung	Dukuh Atas	19.46	20.31
507.	Dukuh Atas	Ragunan	19.46	20.36
508.	Ragunan	Dukuh Atas	19.46	20.36
509.	Dukuh Atas	Pulogadung	19.53	20.38
510.	Pulogadung	Dukuh Atas	19.53	20.38
511.	Dukuh Atas	Ragunan	19.53	20.43
512.	Ragunan	Dukuh Atas	19.53	20.43

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
513.	Dukuh Atas	Pulogadung	20.00	20.45
514.	Pulogadung	Dukuh Atas	20.00	20.45
515.	Dukuh Atas	Ragunan	20.00	20.50
516.	Ragunan	Dukuh Atas	20.00	20.50
517.	Dukuh Atas	Pulogadung	20.10	20.55
518.	Pulogadung	Dukuh Atas	20.10	20.55
519.	Dukuh Atas	Ragunan	20.10	21.00
520.	Ragunan	Dukuh Atas	20.10	21.00
521.	Dukuh Atas	Pulogadung	20.20	21.05
522.	Pulogadung	Dukuh Atas	20.20	21.05
523.	Dukuh Atas	Ragunan	20.20	21.10
524.	Ragunan	Dukuh Atas	20.20	21.10
525.	Dukuh Atas	Pulogadung	20.30	21.15
526.	Pulogadung	Dukuh Atas	20.30	21.15
527.	Dukuh Atas	Ragunan	20.30	21.20
528.	Ragunan	Dukuh Atas	20.30	21.20
529.	Dukuh Atas	Pulogadung	20.40	21.25
530.	Pulogadung	Dukuh Atas	20.40	21.25
531.	Dukuh Atas	Ragunan	20.40	21.30
532.	Ragunan	Dukuh Atas	20.40	21.30
533.	Dukuh Atas	Pulogadung	20.50	21.35
534.	Pulogadung	Dukuh Atas	20.50	21.35
535.	Dukuh Atas	Ragunan	20.50	21.40
536.	Ragunan	Dukuh Atas	20.50	21.40
537.	Dukuh Atas	Pulogadung	21.00	21.45
538.	Pulogadung	Dukuh Atas	21.00	21.45
539.	Dukuh Atas	Ragunan	21.00	21.50
540.	Ragunan	Dukuh Atas	21.00	21.50

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
541.	Dukuh Atas	Pulogadung	21.10	21.55
542.	Pulogadung	Dukuh Atas	21.10	21.55
543.	Dukuh Atas	Ragunan	21.10	22.00
544.	Ragunan	Dukuh Atas	21.10	22.00
545.	Dukuh Atas	Pulogadung	21.20	22.05
546.	Pulogadung	Dukuh Atas	21.20	22.05
547.	Dukuh Atas	Ragunan	21.20	22.10
548.	Ragunan	Dukuh Atas	21.20	22.10
549.	Dukuh Atas	Pulogadung	21.30	22.15
550.	Pulogadung	Dukuh Atas	21.30	22.15
551.	Dukuh Atas	Ragunan	21.30	22.20
552.	Ragunan	Dukuh Atas	21.30	22.20
553.	Dukuh Atas	Pulogadung	21.40	22.25
554.	Pulogadung	Dukuh Atas	21.40	22.25
555.	Dukuh Atas	Ragunan	21.40	22.30
556.	Ragunan	Dukuh Atas	21.40	22.30
557.	Dukuh Atas	Pulogadung	21.50	22.35
558.	Pulogadung	Dukuh Atas	21.50	22.35
559.	Dukuh Atas	Ragunan	21.50	22.40
560.	Ragunan	Dukuh Atas	21.50	22.40
561.	Dukuh Atas	Pulogadung	22.00	22.45
562.	Pulogadung	Dukuh Atas	22.00	22.45
563.	Dukuh Atas	Ragunan	22.00	22.50
564.	Ragunan	Dukuh Atas	22.00	22.50
565.	Dukuh Atas	Pulogadung	22.13	23.58
566.	Pulogadung	Dukuh Atas	22.13	23.58
567.	Dukuh Atas	Ragunan	22.13	23.03
568.	Ragunan	Dukuh Atas	22.13	23.03

No.	Lokasi Keberangkatan	Lokasi Kedatangan	Waktu Keberangkatan (WIB)	Waktu Kedatangan (WIB)
569.	Dukuh Atas	Pulogadung	22.26	23.11
570.	Pulogadung	Dukuh Atas	22.26	23.11
571.	Dukuh Atas	Ragunan	22.26	23.16
572.	Ragunan	Dukuh Atas	22.26	23.16
573.	Dukuh Atas	Pulogadung	22.39	23.24
574.	Pulogadung	Dukuh Atas	22.39	23.24
575.	Dukuh Atas	Ragunan	22.39	23.29
576.	Ragunan	Dukuh Atas	22.39	23.29
577.	Dukuh Atas	Pulogadung	22.52	23.37
578.	Pulogadung	Dukuh Atas	22.52	23.37
579.	Dukuh Atas	Ragunan	22.52	23.42
580.	Ragunan	Dukuh Atas	22.52	23.42
581.	Dukuh Atas	Pulogadung	23.05	23.50
582.	Pulogadung	Dukuh Atas	23.05	23.50
583.	Dukuh Atas	Ragunan	23.05	23.55
584.	Ragunan	Dukuh Atas	23.05	23.55

Lampiran 2. Keluaran dari Penyelesaian Masalah Penjadwalan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar Menggunakan Algoritma *Auction*

S_1	585 – 1 – 30 – 67 – 112 – 436 – 586 – 585
S_2	585 – 2 – 29 – 66 – 107 – 288 – 427 – 476 – 586 – 585
S_3	585 – 3 – 36 – 75 – 120 – 279 – 407 – 472 – 586 – 585
S_4	585 – 4 – 35 – 76 – 119 – 276 – 403 – 452 – 531 – 586 – 585
S_5	585 – 5 – 34 – 71 – 116 – 272 – 401 – 442 – 483 – 586 – 585
S_6	585 – 6 – 33 – 74 – 115 – 156 – 266 – 347 – 380 – 425 – 474 – 515 – 548 – 571 – 586 – 585
S_7	585 – 7 – 40 – 83 – 128 – 270 – 389 – 446 – 491 – 540 – 586 – 585
S_8	585 – 8 – 73 – 114 – 249 – 374 – 411 – 456 – 495 – 528 – 586 – 585
S_9	585 – 9 – 38 – 79 – 124 – 271 – 392 – 435 – 480 – 511 – 544 – 586 – 585
S_{10}	585 – 10 – 37 – 78 – 117 – 170 – 275 – 368 – 405 – 450 – 485 – 586 – 585
S_{11}	585 – 11 – 44 – 87 – 132 – 185 – 269 – 378 – 419 – 464 – 507 – 532 – 567 – 586 – 585
S_{12}	585 – 12 – 77 – 118 – 159 – 255 – 312 – 364 – 393 – 434 – 479 – 524 – 586 – 585
S_{13}	585 – 13 – 42 – 81 – 122 – 155 – 184 – 225 – 246 – 301 – 350 – 381 – 422 – 467 – 536 – 586 – 585
S_{14}	585 – 14 – 41 – 82 – 123 – 164 – 210 – 257 – 310 – 339 – 408 – 451 – 492 – 527 – 586 – 585
S_{15}	585 – 15 – 48 – 105 – 150 – 179 – 248 – 287 – 348 – 383 – 428 – 471 – 512 – 560 – 586 – 585
S_{16}	585 – 16 – 45 – 86 – 127 – 160 – 212 – 252 – 293 – 322 – 379 – 420 – 463 – 586 – 585
S_{17}	585 – 17 – 50 – 91 – 136 – 165 – 190 – 209 – 243 – 284 – 341 – 370 – 413 – 454 – 586 – 580 – 586 – 585
S_{18}	585 – 18 – 49 – 90 – 151 – 201 – 242 – 281 – 306 – 331 – 356 – 385 – 426 – 465 – 502 – 586 – 585

S_{19}	585 – 19 – 56 – 99 – 140 – 191 – 224 – 239 – 292 – 330 – 369 – 406 – 449 – 486 – 519 – 586 – 585
S_{20}	585 – 20 – 61 – 102 – 153 – 186 – 222 – 263 – 328 – 359 – 396 – 443 – 516 – 556 – 586 – 585
S_{21}	585 – 21 – 54 – 97 – 142 – 171 – 204 – 241 – 286 – 326 – 376 – 415 – 460 – 586 – 543 – 572 – 586 – 585
S_{22}	585 – 22 – 53 – 94 – 135 – 172 – 195 – 232 – 265 – 294 – 329 – 358 – 387 – 432 – 473 – 550 – 586 – 585
S_{23}	585 – 23 – 60 – 103 – 144 – 173 – 202 – 233 – 274 – 297 – 318 – 349 – 382 – 417 – 458 – 493 – 586 – 555 – 576 – 586 – 585
S_{24}	585 – 24 – 57 – 98 – 147 – 176 – 205 – 250 – 299 – 324 – 351 – 384 – 423 – 468 – 503 – 586 – 585
S_{25}	585 – 25 – 62 – 111 – 152 – 181 – 206 – 259 – 296 – 335 – 360 – 421 – 462 – 499 – 586 – 585
S_{26}	585 – 26 – 85 – 126 – 167 – 196 – 234 – 267 – 300 – 323 – 375 – 424 – 469 – 506 – 552 – 586 – 585
S_{27}	585 – 27 – 68 – 109 – 146 – 175 – 200 – 223 – 256 – 291 – 316 – 337 – 362 – 395 – 440 – 481 – 535 – 586 – 585
S_{28}	585 – 28 – 65 – 106 – 143 – 180 – 218 – 235 – 280 – 317 – 366 – 397 – 438 – 477 – 514 – 561 – 586 – 585
S_{29}	585 – 31 – 72 – 121 – 154 – 183 – 220 – 245 – 282 – 313 – 334 – 357 – 390 – 439 – 500 – 565 – 586 – 585
S_{30}	585 – 32 – 69 – 110 – 149 – 193 – 230 – 261 – 290 – 309 – 338 – 363 – 400 – 441 – 498 – 533 – 586 – 585
S_{31}	585 – 39 – 84 – 157 – 221 – 254 – 285 – 327 – 372 – 409 – 466 – 497 – 586 – 534 – 553 – 586 – 585
S_{32}	585 – 43 – 88 – 131 – 168 – 189 – 214 – 231 – 268 – 307 – 344 – 377 – 414 – 453 – 494 – 521 – 586 – 570 – 586 – 585
S_{33}	585 – 46 – 89 – 130 – 163 – 188 – 211 – 244 – 319 – 352 – 399 – 444 – 487 – 520 – 557 – 586 – 582 – 586 – 585

S_{34}	585 – 47 – 92 – 133 – 162 – 208 – 229 – 262 – 311 – 336 – 367 – 412 – 455 – 496 – 545 – 566 – 586 – 585
S_{35}	585 – 51 – 96 – 139 – 192 – 227 – 264 – 295 – 320 – 355 – 404 – 447 – 488 – 525 – 546 – 573 – 586 – 585
S_{36}	585 – 52 – 93 – 134 – 161 – 207 – 260 – 305 – 343 – 388 – 431 – 484 – 523 – 586 – 585
S_{37}	585 – 55 – 100 – 137 – 187 – 216 – 258 – 289 – 342 – 371 – 416 – 461 – 510 – 529 – 586 – 558 – 575 – 586 – 585
S_{38}	585 – 58 – 101 – 138 – 169 – 194 – 217 – 238 – 277 – 302 – 325 – 346 – 373 – 410 – 459 – 504 – 537 – 586 – 578 – 586 – 585
S_{39}	585 – 59 – 104 – 141 – 182 – 203 – 237 – 278 – 303 – 332 – 353 – 386 – 429 – 470 – 509 – 586 – 538 – 559 – 584 – 586 – 585
S_{40}	585 – 63 – 108 – 145 – 178 – 215 – 240 – 283 – 308 – 361 – 394 – 433 – 478 – 505 – 549 – 586 – 574 – 586 – 585
S_{41}	585 – 64 – 125 – 166 – 199 – 228 – 253 – 298 – 321 – 391 – 448 – 489 – 530 – 563 – 586 – 585
S_{42}	585 – 70 – 113 – 174 – 197 – 226 – 247 – 315 – 340 – 365 – 402 – 445 – 490 – 517 – 542 – 569 – 586 – 585
S_{43}	585 – 80 – 129 – 158 – 213 – 251 – 304 – 333 – 354 – 430 – 501 – 526 – 551 – 586 – 585
S_{44}	585 – 95 – 148 – 177 – 198 – 219 – 236 – 273 – 314 – 345 – 398 – 437 – 586 – 518 – 539 – 564 – 583 – 586 – 585
S_{45}	585 – 418 – 457 – 522 – 541 – 562 – 577 – 586 – 585
S_{46}	585 – 475 – 508 – 547 – 568 – 581 – 586 – 585
S_{47}	585 – 482 – 513 – 554 – 579 – 586 – 585

Dengan total biaya yang dikeluarkan sebesar Rp. 272.069.387,982980.

Lampiran 3. *Source Code* Algoritma *Auction* untuk Menyelesaikan Masalah Penjadwalan Kendaraan Bus TransJakarta dengan Memperhatikan Jadwal Pengisian Bahan Bakar

```
%Fungsi penghitungan sisa bahan bakar kendaraan:
function BBG = HitungBBG(Sk, def_BBG, cBBG, startloc, endloc)

n = length(startloc);
k = length(Sk);
i = find(Sk==n+2, 1, 'last');

if ~isempty(i)
    if i == k
        BBG = def_BBG;
    else
        BBG = def_BBG - cBBG(2, startloc(Sk(i+1))) -
cBBG(startloc(Sk(i+1)), endloc(Sk(i+1)));
    end
else
    BBG = def_BBG - cBBG(4, startloc(Sk(1))) -
cBBG(startloc(Sk(1)), endloc(Sk(1)));
    i = 1;
end

for j = i+1 : k-1
    BBG = BBG - cBBG(endloc(Sk(j)), startloc(Sk(j+1))) -
cBBG(startloc(Sk(j+1)), endloc(Sk(j+1)));
end

%Fungsi penentuan kandidat successor:
function fKandidat = FilterKandidat(kandidat, S, k)

fKandidat = [];
for i = 1 : length(kandidat)
    flag = 0;
    for j = 1 : length(S)
        if ~isempty(find(S{j} == kandidat(i), 1))
            flag = flag +1;
            break;
        end
    end
    if k > j
        flag = 0;
    end
    if flag == 0
        fKandidat = [fKandidat kandidat(i)];
    end
end

%Fungsi penghapusan elemen array:
function A = del(A,i)
```

```
if isempty(A) || i>length(A)
    fprintf('\nMatrix kosong atau index melebihi batas\n');
elseif i == 1
    A = A(i+1:length(A));
elseif i == length(A)
    A = A(1:length(A)-1);
else
    A= [A(1:i-1) A(i+1:length(A))];
end

%Algoritma auction:
clc;
clear;
timetable=xlsread('timetable.xlsx',8);
startloc=timetable(:,2);
endloc=timetable(:,3);
traveltime=[2 45 50;45 2 95;50 95 2];
starttime=timetable(:,6);
n=numel(startloc);
for i=1:n
    endtime(i)=starttime(i)+traveltime(startloc(i),endloc(i));
end

%Pembentukan himpunan busur:
for i=1:n
    arc(n+1,i)=1;
    for j=1:n
        if
endtime(i)+traveltime(endloc(i),startloc(j))<=starttime(j)
            arc(i,j)=1;
        end
    end
end

%Pembentukan matriks biaya
for i=1:n
    if (startloc(i)==1 && endloc(i)==2)
        cost(n+1,i)=1031000;
        cost(n+2,i)=43400;
    elseif startloc(i)==2
        cost(n+1,i)=1031000;
        cost(n+2,i)=21700;
    elseif (startloc(i)==1 && endloc(i)==3)
        cost(n+1,i)=1034100;
        cost(n+2,i)=46500;
    elseif (startloc(i)==3 && endloc(i)==1)
        cost(n+1,i)=1034100;
        cost(n+2,i)=71300;
    end
    if endloc(i)==1
        cost(i,n+2)=21700;
    elseif endloc(i)==2
```

```
        cost(i,n+2)=0;
    elseif endloc(i)==3
        cost(i,n+2)=46500;
    end
    for j=1:n
        if endloc(i)==startloc(j)
            if endloc(i)==1
                if endloc(j)==2
                    cost(i,j)=21700;
                elseif endloc(j)==3
                    cost(i,j)=24800;
                end
            elseif endloc(i)==2
                cost(i,j)=21700;
            elseif endloc(i)==3
                cost(i,j)=24800;
            end
        else
            if endloc(i)==1 && startloc(j)==2
                cost(i,j)=43400;
            elseif endloc(i)==1 && startloc(j)==3
                cost(i,j)=49600;
            elseif endloc(i)==2
                if startloc(j)==1 && endloc(j)==2
                    cost(i,j)=43400;
                elseif startloc(j)==1 && endloc(j)==3
                    cost(i,j)=46500;
                elseif startloc(j)==3
                    cost(i,j)=71300;
                end
            elseif endloc(i)==3
                if startloc(j)==1 && endloc(j)==2
                    cost(i,j)=46500;
                elseif startloc(j)==1 && endloc(j)==3
                    cost(i,j)=49600;
                elseif startloc(j)==2
                    cost(i,j)=68200;
                end
            end
        end
    end
    cost(i,n+1)=inf;
end
cost(n+1,n+1)=inf;
cost(n+1,n+2)=inf;
cost(n+2,n+2)=inf;
cost(n+2,n+1)=9300;
cBBG=[0 7 8;7 0 15;8 15 0;3 3 3];

%Kondisi awal:
for k=1:n+2
    if k<=n+1
        price(k)=0;
        profit(k)=0;
```

```
    end
    for j=1:n+2
        a(k,j)=-cost(k,j);
    end
end
S={};
S_ujung=[];
epsilon=1/(n+3);

%Dilakukan forward assignment untuk setiap trip yang belum
memiliki succcessor
fwd = zeros(1,n); i = 1; def_BBG = 120;
BBG=def_BBG*ones(1,n);
for b=1:n
    BBG(b) = def_BBG - cBBG(4,startloc(b)) -
cBBG(startloc(b),endloc(b));
end
while sum(fwd) < numel(fwd)

    if i == n+1
        next = S_ujung(find(S_ujung~=n+1, 1, 'first'));
        if isempty(next)
            next = find(fwd==0);
        end
        if ~isempty(next)
            i = next(1);
        end
    end

    while i == n+2
        k = find(S_ujung == i);
        if isempty(k)
            next = S_ujung(find(S_ujung~=n+1, 1, 'first'));
            if isempty(next)
                next = find(fwd==0);
            end
            if ~isempty(next)
                i = next(1);
            end
            break;
        end
        end_k = length(S{k});
        i = S{k}(end_k - 1);
        ind_kandidat = 1;
        kandidat=[];
        BBG(k)=def_BBG;
        for j=1:n
            if
endtime(i)+traveltime(endloc(i),2)+15+traveltime(2,startloc(j))<=s
tarttime(j)
                kandidat(ind_kandidat) = j;
                ind_kandidat = ind_kandidat + 1;
            end
```

```
        end
        kandidat = FilterKandidat(kandidat, S, k);
        if ~isempty(kandidat)
            f(i,:)=-inf*ones(1,n);
            f(i,kandidat)=a(i,n+2)+a(n+2,kandidat)-
price(kandidat);
            [beta(i) j]=max(f(i,:));
            f(i,j)=-inf;
            gamma(i)=max(f(i,:));
            if gamma(i) == -inf
                profit(i)=a(i,n+2)+a(n+2,j);
            else
                price(j)=price(j)+beta(i)-gamma(i)+epsilon;
                profit(i)=a(i,n+2)+a(n+2,j)-price(j);
            end

            i = j;
            disp(['terpilih ' num2str(i)]);
            for l=1:length(S)
                z=find(S{l}==j);
                if ~isempty(z) && j~=n+1
                    break;
                end
            end

            if ~isempty(z) && j~=n+2
                if z > 1
                    Sl = S{l};
                    Sl2= Sl(z-1:length(Sl));
                    Sl2= Sl2(find(Sl2<=n));
                    fwd(Sl2) = zeros(size(Sl2));
                    S{l}=[Sl(1:z-1)];
                    S_ujung(l) = S{l}(end);
                    BBG(l) = HitungBBG(S{l}, def_BBG, cBBG,
startloc, endloc);

                    S{k} = [S{k} j];
                    fwd(k) = 1;
                else
                    S{l}=[];
                end
                S_ujung(k)=S{k}(end);
                i = S_ujung(find(S_ujung~=n+1, 1, 'first'));

                if isempty(S{l})
                    disp(['panjang S' num2str(l) ' = '
num2str(length(S{l}))]);
                    k = k-1;
                    S = del(S,l);
                    S_ujung = del(S_ujung,l);
                else
                    if S{l}(end) == n+2
                        S{l}(end) = [];
```

```
                    end
                    S_ujung(l)=S{l}(end);
                    x = fwd(S_ujung(l));
                    disp(['l = ' num2str(l) '; fwd '
num2str(S_ujung(l)) ' = ' num2str(x)]);
                    disp(S{l});
                end
            elseif i == n+2
                S{k}=[S{k} j n+1];
                S_ujung(k)=n+1;
            else
                S{k}=[S{k} j];
                S_ujung(k)=j;
            end
        else
            S{k} = [S{k} n+1];
            S_ujung(k) = n+1;
            next = S_ujung(find(S_ujung~=n+1, 1, 'first'));
            if isempty(next)
                next = find(fwd==0);
            end
            if ~isempty(next)
                i = next(1);
            end
        end
    end

    k = find(S_ujung == i);
    if isempty(k)
        k = length(S)+1;
        S{k} = i;
        S_ujung(k) = i;
    end
    fwd(i)=1;
    disp(['k = ' num2str(k) '/' num2str(length(S)) '; i = '
num2str(i) '; fwd = ' num2str(fwd(i)) '; sum = '
num2str(sum(fwd))]);
    kandidat=find(arc(i,:)==1);
    kandidat = FilterKandidat(kandidat, S, k);
    BBGk=BBG(k)*ones(size(kandidat));
    for h=1:length(kandidat)
        BBGk(h)=BBGk(h)-cBBG(endloc(i),startloc(kandidat(h)))-
cBBG(startloc(kandidat(h)),endloc(kandidat(h)))-
cBBG(endloc(kandidat(h)),2);
    end
    kandidat=kandidat(find(BBGk>=0));
    if ~isempty(kandidat)
        f(i,:) = -inf*ones(1,n);
        j=kandidat;
        if ~isempty(j)
            f(i,j)=a(i,j)-price(j);
            [beta(i) j]=max(f(i,:));
            f(i,j)=-inf;
```

```
            BBG(k) = BBG(k)-cBBG(endloc(i),startloc(j))-
cBBG(startloc(j),endloc(j))-cBBG(endloc(j),1);
            gamma(i)=max(f(i,:));

            if gamma(i) == -inf
                profit(i)=a(i,j);
            else
                price(j)=price(j)+beta(i)-gamma(i)+epsilon;
                profit(i)=a(i,j)-price(j);
            end
            i = j;
            disp(['terpilih ' num2str(i)]);
            for l=1:length(S)
                z=find(S{l}==j);
                if ~isempty(z) && j~=n+2
                    break
                end
            end
            if ~isempty(z) && j~=n+2
                if z > 1
                    Sl = S{l};
                    Sl2= Sl(z-1:length(Sl));
                    Sl2= Sl2(find(Sl2<=n));
                    fwd(Sl2) = zeros(size(Sl2));
                    S{l}=[Sl(1:z-1)];
                    S_ujung(l) = S{l}(end);
                    BBG(l) = HitungBBG(S{l}, def_BBG, cBBG,
startloc, endloc);

                    S{k} = [S{k} j];
                    fwd(k) = 1;
                else
                    S{l}=[];
                end
                S_ujung(k)=S{k}(end);
                i = S_ujung(find(S_ujung~=n+1, 1, 'first'));
                if isempty(S{l})
                    disp(['panjang S' num2str(l) ' = '
num2str(length(S{l}))]);
                    k = k-1;
                    S = del1(S,l);
                    S_ujung = del(S_ujung,l);
                else
                    disp(S{l})
                end
            elseif i == n+2
                S{k}=[S{k} j n+1];
                S_ujung(k)=n+1;
            else
                S{k}=[S{k} j];
                S_ujung(k)=j;
            end
        else
```

```
            S{k} = [S{k} n+2 n+1];
            S_ujung(k) = n+1;
            next = S_ujung(find(S_ujung~=n+1, 1, 'first'));
            if isempty(next)
                next = find(fwd==0);
            end
            if ~isempty(next)
                i = next(1);
            end
        end
    else
        S{k}=[S{k} n+2];
        disp('terpilih SPBBG');
        ind_kandidat = 1;
        kandidat=[];
        BBG(k)=def_BBG;
        for j=1:n
            if
endtime(i)+traveltime(endloc(i),2)+15+traveltime(2,startloc(j))<=s
tarttime(j)
                kandidat(ind_kandidat) = j;
                ind_kandidat = ind_kandidat + 1;
            end
        end
        kandidat = FilterKandidat(kandidat, S, k);
        if ~isempty(kandidat)
            f(i,:)=-inf*ones(1,n);
            f(i,kandidat)=a(i,n+2)+a(n+2,kandidat)-
price(kandidat);
            [beta(i) j]=max(f(i,:));
            f(i,j)=-inf;
            gamma(i)=max(f(i,:));
            if gamma(i) == -inf
                profit(i)=a(i,n+2)+a(n+2,j);
            else
                price(j)=price(j)+beta(i)-gamma(i)+epsilon;
                profit(i)=a(i,n+2)+a(n+2,j)-price(j);
            end

            i = j;
            disp(['terpilih ' num2str(i)]);
            for l=1:length(S)
                z=find(S{l}==j);
                if ~isempty(z) && j~=n+1
                    break;
                end
            end

            if ~isempty(z) && j~=n+2
                if z > 1
                    Sl = S{l};
                    Sl2= Sl(z-1:length(Sl));
                    Sl2= Sl2(find(Sl2<=n));
```

```
                    fwd(Sl2) = zeros(size(Sl2));
                    S{l}=[Sl(1:z-1)];
                    S_ujung(l) = S{l}(end);
                    BBG(l) = HitungBBG(S{l}, def_BBG, cBBG,
startloc, endloc);

                    S{k} = [S{k} j];
                    fwd(k) = 1;
                else
                    S{l}=[];
                end
                S_ujung(k)=S{k}(end);
                i = S_ujung(find(S_ujung~=n+1, 1, 'first'));

                if isempty(S{l})
                    disp(['panjang S' num2str(l) ' = '
num2str(length(S{l}))]);
                    k = k-1;
                    S = del(S,l);
                    S_ujung = del(S_ujung,l);
                else
                    disp(S{l});
                end
            elseif i == n+2
                S{k}=[S{k} j n+1];
                S_ujung(k)=n+1;
            else
                S{k}=[S{k} j];
                S_ujung(k)=j;
            end
        else
            S{k} = [S{k} n+1];
            S_ujung(k) = n+1;
            next = S_ujung(find(S_ujung~=n+1, 1, 'first'));
            if isempty(next)
                next = find(fwd==0);
            end
            if ~isempty(next)
                i = next(1);
            end
        end
    end
    fprintf('\n End While \n')
end

for i=length(S):-1:1
    if isempty(S)
        S = del(S,i);
    end
end

%Proses backward assignment trip ke depot keberangkatan
for i = 1:length(S)
```

```
    if S{i}(1)~=n+1
        S{i}=[n+1 S{i}];
    end
end

for i=1:length(S)
    for j=1:length(S{i})
        if j==1
            fprintf('%d ',S{i}(j));
        else
            fprintf('- %d ',S{i}(j));
        end
        if j==length(S{i})
            fprintf('\n');
        end
    end
end

%Penghitungan total biaya
total_cost=sum(price)+sum(profit);
for i=1:length(S)
    total_cost=total_cost+a(n+1,S{i}(2));
end
fprintf('%f\n',total_cost);
```