TUGAS AKHIR – IF184802

PENGEMBANGAN *INCESSANT ALLOCATION METHOD – PRIORITY VALUE* UNTUK MENCARI *INITIAL BASIC FEASIBLE SOLUTION* PADA *TRANSPORTATION PROBLEM*

AISYAH MUSWAR
NRP 05111640000035

Dosen Pembimbing
Victor Hariadi,S.Si,M.Kom.
Bilqis Amaliah, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

PENGEMBANGAN *INCESSANT ALLOCATION METHOD – PRIORITY VALUE* UNTUK MENCARI *INITIAL BASIC FEASIBLE SOLUTION* PADA *TRANSPORTATION PROBLEM*

AISYAH MUSWAR
NRP 05111640000035

Dosen Pembimbing
Victor Hariadi,S.Si,M.Kom.
Bilqis Amaliah, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

DEVELOPMENT OF INCESSANT ALLOCATION METHOD - PRIORITY VALUE TO FINDING INITIAL BASIC FEASIBLE SOLUTION ON TRANSPORTATION PROBLEM

AISYAH MUSWAR
NRP 05111640000035

Supervisors
Victor Hariadi,S.Si,M.Kom.
Bilqis Amaliah, S.Kom., M.Kom.

DEPARTMENT of INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGEMBANGAN *INCESSANT ALLOCATION METHOD – PRIORITY VALUE* UNTUK MENCARI *INITIAL BASIC FEASIBLE SOLUTION* PADA *TRANSPORTATION PROBLEM*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Dasar dan Terapan Komputasi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
AISYAH MUSWAR
NRP 05111640000035

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Victor Hariadi, S.Si., M.Kom. .....
NIP: 19691228 199412 1 001 (Pembimbing 1)

Bilqis Amaliah, S.Kom., M.Kom. .....
NIP: 19751914 200112 2 001 (Pembimbing 2)

SURABAYA
JULI 2020

[Halaman ini sengaja dikosongkan]

PENGEMBANGAN *INCESSANT ALLOCATION METHOD – PRIORITY VALUE* UNTUK MENCARI *INITIAL BASIC FEASIBLE SOLUTION* PADA *TRANSPORTATION PROBLEM*

Nama Mahasiswa : Aisyah Muswar
NRP : 05111640000035
Departemen : Teknik Informatika FTEIC-ITS
Dosen Pembimbing 1 : Victor Hariadi, S.Si, M.Kom.
Dosen Pembimbing 2 : Bilqis Amaliah, S.Kom., M.Kom.

ABSTRAK

Hampir setiap tahun, perusahaan manufaktur mengeluarkan biaya jutaan dolar untuk pendistribusian barang. Kemajuan perusahaan pun berbanding lurus dengan efisiensi metode transportasi perusahaan. Permasalahan ini mendorong para peneliti untuk melakukan riset dalam mencari biaya minimal, yang kemudian dikenal sebagai Transportation Problem (TP). TP merupakan bagian dari Linier Programming, baik pada bidang aplikasi matematika maupun riset operasional. Ada dua proses untuk mencari solusi optimal dari TP. Pertama adalah mencari Initial Basic Feasible Solution (IBFS) dan kedua adalah mencari solusi optimal dari IBFS menggunakan stepping stone. IBFS merupakan solusi awal dengan menggunakan algoritma tertentu sedemikian rupa sehingga biaya pengiriman pada TP mendekati angka paling minimal.

Berbagai riset telah dilakukan oleh peneliti untuk mencari algoritma IBFS, salah satunya adalah Incessant Allocation Method - Priority Value (IAM-PV). IAM-PV merupakan algoritma IBFS yang menggunakan metode pengalokasian terus menerus pada tiap rute dengan memperhatikan nilai prioritas terbesar dari perbandingan hitungan row priority value dan column priority value, hingga semua rute terisi. Pada penelitian ini, algoritma

IAM-PV akan dimodifikasi pada bagian inisiasi matriks awal. Modifikasi ini dinamakan Total Opportunity Cost Matrix – Priority Value (TOCM-PV). Algoritma tersebut diimplementasikan dalam bentuk program berbahasa C dengan format input sejumlah supply, sejumlah demand, dan cost tiap rute. Output yang diharapkan dari program ini adalah nilai alokasi tiap rute dan total biaya pengalokasian tersebut.

Algoritma IAM-PV dan TOCM-PV diujicobakan terhadap 85 data. Pada hasil uji penelitian ini, persentase akurasi algoritma IAM-PV sebesar 16,47%, sedangkan persentase akurasi algoritma TOCM-PV sebesar 18,82%. Selain itu, TOCM-PV lebih baik daripada IAM-PV, hal ini dibuktikan dengan 44 data dari 85 data uji coba dapat mencapai hasil biaya lebih rendah dari IAM-PV. Sehingga pada penelitian ini dapat disimpulkan bahwa algoritma TOCM-PV lebih mendekati hasil optimal daripada algoritma IAM-PV.

Kata kunci: distribusi produk, biaya minimal, transportation problem, total opportunity cost matrix priority value , algoritma incesant allocation method priority value

DEVELOPMENT OF INCESSANT ALLOCATION METHOD - PRIORITY VALUE TO FINDING INITIAL BASIC FEASIBLE SOLUTION ON TRANSPORTATION PROBLEM

Name : **Aisyah Muswar**
NRP : **05111640000035**
Department : **Informatics FTEIC-ITS**
Supervisor I : **Victor Hariadi, S.Si, M.Kom.**
Supervisor II : **Bilqis Amaliah, S.Kom., M.Kom.**

ABSTRACT

Almost every year, manufacturing companies issues millions of dollars in the distribution of goods. The company's progress is directly proportional to the efficiency of the company's transportation methods. This problem encouraged researchers to conduct research in looking for minimal costs, which became known as the Transportation Problem (TP). TP is part of Linear Programming, both in the field of mathematical applications and operational research. There are two processes to find the optimal solution from TP. First is looking for Initial Basic Feasible Solution (IBFS) and second is looking for optimal solutions from IBFS using stepping stones. IBFS is an initial solution that is provided using certain algorithms in such a way that shipping costs on TP approach the minimum amount.

Various studies have been conducted by researchers to look for the IBFS algorithm, one of which is the Incessant Allocation Method - Priority Value (IAM-PV). IAM-PV is an IBFS algorithm that uses a continuous allocation method on each route by taking into account the greatest priority value from the ratio of row priority value and column priority value, until all routes are filled. In this study, the IAM-PV algorithm will be modified in the

initial matrix initiation section. Modification of this method is called the Total Opportunity Cost Matrix - Priority Value (TOCM-PV). The algorithm is implemented in the form of a C-language program with the input format of a number of supplies, a number of demands, and the cost of each route. The expected output from this program is the value of the allocation of each route and the total allocation costs.

IAM-PV and TOCM-PV algorithms were tested on 85 data. In the test results of this study, the percentage accuracy of the IAM-PV algorithm was 16.47%, while the percentage accuracy of the TOCM-PV algorithm was 18.82%. In addition, TOCM-PV is better than IAM-PV is better than IAM-PV, this is proven by 44 data from 85 trial data that can achieve lower cost results than IAM-PV. So in this study it can be concluded that the TOCM-PV algorithm is closer to optimal results than the IAM-PV algorithm.

Keywords: product distribution, minimal costs, transportation problems, total opportunity cost matrix priority value, accession allocation method priority value algorithm

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT karena atas karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

PENGEMBANGAN *INCESSANT ALLOCATION METHOD – PRIORITY VALUE* UNTUK MENCARI *INITIAL BASIC FEASIBLE SOLUTION* PADA *TRANSPORTATION PROBLEM*

Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Bapak, Ibu, adik, sekeluarga tercinta yang selalu memberikan doa serta dukungan kepada penulis untuk menyelesaikan tugas akhir ini.
2. Bapak Victor Hariadi, S.Si, M.Kom selaku dosen pembimbing tugas akhir pertama yang telah membimbing dan memberi banyak masukkan dalam pengerjaan tugas akhir ini.
3. Ibu Bilqis Amaliah, S.Kom., M.Kom. selaku dosen pembimbing tugas akhir kedua yang telah memberikan masukkan serta koreksi dalam pengerjaan tugas akhir.
4. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. selaku dosen wali yang senantiasa memberikan dukungan selama perluliahan.
5. Bapak dan Ibu dosen serta seluruh civitas Departemen Teknik Informatika yang telah memberikan pelajaran dan pengalaman selama menjadi mahasiswa di Departemen Teknik Informatika.
6. Teman-teman TC16, serta seluruh anggota BEMF dan IFLS yang sudah menemani, mendukung dan memberikan pelajaran dalam organisasi maupun perkuliahan.

7. Teman sepejuangan Nada, Ramdan, Mukti, Puguh, Ardin, Ivan, Nafis dan Fadhlan yang sudah membantu penulis selama perkuliahan dan juga memberi rasa kekeluargaan.
8. Henry Haidar Jati Andrian untuk semangat dan dukungan kepada penulis selama pengerjaan tugas akhir.
9. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyelesaikan tugas akhir ini. Namun, penulis mohon maaf apabila terdapat kekurangan ataupun kesalahkan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk ke depannya.

Surabaya, Juli 2020

Aisyah Muswar

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Transportation Problem (TP) adalah pendistribusian barang dari banyak *supply* ke banyak *demand* dengan tujuan untuk mencari total biaya minimal dengan memperhatikan batasan-batasan yang ada [1]. Tujuan TP sendiri adalah pencarian total biaya minimal untuk mencapai solusi optimal [2]. Hampir setiap tahun, perusahaan mengeluarkan biaya jutaan dolar untuk pendistribusian barang [3]. TP merupakan bagian dari *Linier Programming*, baik pada bidang aplikasi matematika maupun riset operasional [4]. Pemanfaatan model transportasi dapat dikembangkan pada bidang operasi lainnya, seperti pengaturan inventaris, penjadwalan kerja, dan penempatan karyawan [1].

Ada dua proses untuk mencari solusi optimal dari TP. Pertama adalah mencari *Initial Basic Feasible Solution* dan kedua adalah mencari solusi optimal dari IBFS menggunakan *stepping stone* [5]. IBFS sangat penting dicari untuk mendapatkan solusi optimal karena IBFS sendiri merupakan dasaran dari percarian solusi TP [6]. Hasil dari IBFS bisa mendekati atau bahkan sama dengan nilai solusi optimal. Angka iterasi proses pencarian akan berkurang apabila menggunakan IBFS dengan hasil yang baik [2].

Pada peneliatian sebelumnya, terdapat beberapa algoritma untuk menyelesaikan IBFS, antara lain *North West Corner Rule* (NWC), *Least Cost Method* (LCM), dan *Vogel's Approximation Method* (VAM). Dari ketiga algorima tersebut, VAM merupakan algortima yang paling efisien dalam pencarian IBFS [4]. Studi literatur berisi metode-metode pengembangan pencarian IBFS yang diusulkan peneliti lain. Kirca dan Satir [7] mengembangkan metode *Total Opportunity-Cost Method* (TOCM) untuk mencari IBFS. Mathirajan dan Meenakshi [8] melakukan pengembangan dengan cara menyatukan konsep *Total Opportunity Cost* (TOC) dengan VAM. Khan [9] menggunakan *pointer cost* yang dihasilkan dari selisih biaya tertinggi dan biaya tertinggi kedua pada setiap

baris dan kolom. Islam et al [10] mengemukakan pendekatan baru yang disebut *Total Opportunity Cost Table* (TOCT). Metode TOC menghitung *Distribution Indicators* (DI) yang didapatkan dari selisih biaya unit terbesar dan biaya unit terbesar kedua. Khan et al [11] mengembangkan sebuah metode baru untuk menemukan *initial solution*, metode ini bernama TOCM-SUM *Approach*.

Metode berbeda telah diusulkan oleh para peneliti lainnya. Babu et al. [12], Kousalya dan P.Malarvizhi [13] dan Moraden [14] memulai IBFS dari *supply* minimal atau *demand* minimal. Babu et al. [4] mempertimbangkan biaya pada setiap *cell* dan mulai dari yang minimal. Uddin dan Khan [5] menambahkan beberapa aturan apabila terdapat nilai yang sama pada biaya minimal. Ahmed et al. [15] mengusulkan *Incessant Allocation Method* (IAM) yang dimulai dari biaya paling sedikit, kemudian dilanjutkan dengan cara mencari biaya lainnya pada baris atau kolom yang sama hingga *satisfy*. Ahmed et al. [16] dan Deshmukh [17] memulai Langkah IBFS dengan pengalokasian dari biaya ganjil paling sedikit. Kaur et al. [18] menambahkan sebuah aturan baru jika terdapat nilai *Maximum Difference Method* yang sama.

Berdasarkan beberapa studi yang telah dilakukan, penelitian ini mengusulkan sebuah metode baru untuk mendapatkan IBFS yang lebih baik. Metode yang diusulkan ini dinamakan *Total Opportunity Cost Matrix – Priority Value* (TOCM-PV) yaitu modifikasi dari IAM-PV [19]. TOCM-PV adalah gabungan dari *Total Opportunity Cost Matrix* (TOCM) [7] dan IAM-PV. TOCM-PV menggunakan TOCM sebagai matriks inisial dan langkah selanjutnya memproses matriks TOCM menggunakan metode IAM-PV. *Priority Value* adalah *cell* paling berharga dibandingkan *cell* lain di baris dan kolom yang sama. PV menghitung *Row Priority Value* (RPV) dan C*olumn Priority Value* (CPV). Langkah selanjutnya adalah memilih *cell* dengan nilai prioritas tertinggi. 86 contoh data numerik digunakan untuk mengevaluasi kinerja TOCM-PV. Data tersebut juga dibandingkan dengan metode yang telah ada seperti IAM [15], VAM [20], dan IAM-PV [19].

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah :

1. Bagaimana memodifikasi algoritma *Incessant Allocation Method Priority Value* (IAM-PV) sehingga dapat menghasilkan biaya yang lebih kecil daripada algoritma IAM-PV yang asli?
2. Berapa besar nilai kenaikan akurasi algoritma IAM-PV yang telah dikembangkan dibandingkan dengan algoritma IAM-PV yang asli?

1.3 Batasan Permasalahan

Batasan masalah pada tugas akhir ini antara lain:

1. ToraSystem 7th Edition merupakan *tools* yang digunakan untuk memvalidasi jawaban optimal. *Tools* ini melakukan langkah inisiasi dan optimalisasi (*Stepping Stone*) sekaligus sehingga hasil yang didapatkan pasti optimal.
2. Dev-C++5.11 adalah *tools* yang akan digunakan untuk mengimplementasikan hasil modifikasi IAM-PV dalam bentuk pemrograman C++ .
3. Algoritma IBFS yang dibahas pada penelitian ini hanya akan ditelaah pada proses inisialisasi saja.
4. Data yang digunakan adalah 36 data real dari perusahaan XYZ pada tahun 2017 sebanyak 18 data dan pada tahun 2018 sebanyak 18 data. Perusahaan XYZ terdiri dari 2 pabrik yang berfungsi sebagai *supply* dan 94 kota tujuan yang berfungsi sebagai *demand*. Data biaya adalah data *cycle time* (waktu yang diperlukan oleh truk dari suatu pabrik ke kota tujuan dan kembali ke pabrik dalam satuan jam).
5. Contoh kasus sebanyak 34 data diambil dari beberapa referensi jurnal.
6. 15 Data sintesis dengan ukuran data yang besar.

7. Variabel biaya *supply* dan *demand* bertipe *integer*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini antara lain:

1. Memodifikasi IAM-PV agar dapat menghasilkan biaya yang lebih kecil dan kenaikan nilai akurasi lebih besar daripada algoritma IAM-PV yang asli.
2. Mengevaluasi kinerja metode modifikasi IAM-PV dengan menggunakan pengukuran *improvement percentage*, *deviation percentage* dan akurasi.

1.5 Manfaat

Tugas akhir diharapakan dapat menghasilkan algoritma baru yang menjadi pembanding algoritma IAM-PV yang asli, sehingga dapat menjadi referensi bagi peneliti lain dalam mengembangkan algoritma IBFS. Program yang dibuat dalam tugas akhir ini dapat digunakan untuk mendapatkan solusi IBFS dari berbagai soal, sehingga pada studi kasus nyata dapat digunakan untuk menyelesaikan masalah *Transportation Problem*.

.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi literatur

Pada studi literatur ini akan dipelajari sejumlah referensi yang relevan terhadap tugas akhir yang akan dikerjakan. Studi literatur ini didapatkan dari buku, internet serta materi-materi kuliah yang berhubungan dengan metode yang akan digunakan. Hal-hal yang akan dipelajari antara lain mengenai *Transportation Problem*, *Initial Base Feasible Soultion* (IBFS) dan algoritma *Incessant Allocation Method – Priority Value* (IAM-PV).

2. Analisis dan desain metode

Metode yang akan digunakan adalah modifikasi dari *Incessant Allocation Method Priority Value* (IAM-PV). Bagian yang dikembangkan pada algoritma IAM-PV adalah pada proses inisiasi matriks awal yang dikerjakan dengan menggunakan metode TOCM yaitu orisinal matriks menjadi inisial matriks dengan cara menjumlahkan antara matriks *row opportunity cost* dan matriks *coloum opportunity cost*. Selanjutnya matriks akan diproses menggunakan metode IAM-PV.

3. Implementasi

Pada tugas akhir ini, algoritma IAM-PV modifikasi yang akan dibuat akan diimplementasikan menjadi program berbahasa C++. Tools yang akan digunakan adalah Dev C++ 5.11 karena telah menyediakan *library* yang dibutuhkan.

4. Pengujian dan evaluasi

Uji coba akan dilakukan terhadap 34 soal studi kasus dari beberapa paper, 36 data real dari perusahaan XYZ dan 15 data sintesis. Algoritma yang akan digunakan sebagai pembanding antara lain Vogel, IAM, TDM1, IAM-PV, JHM, TOCM-MT dan BCE. Langkah pertama dalam pengujian adalah mencoba algoritma IAM, TDM1, IAM-PV, JHM, TOCM-MT dan BCE yang telah dibuat programnya pada keseluruhan studi kasus. Langkah berikutnya adalah mendapatkan hasil dari algoritma Vogel dan jawaban optimal menggunakan *tools* ToraSystem 7th Edition pada setiap studi kasus. Jawaban optimal ini akan berperan sebagai kunci jawaban setiap studi kasus yang nantinya akan menjadi patokan terhadap akurasi masing-masing algoritma.

5. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, tinjauan pustaka, metode, implementasi, proses yang telah dilakukan, pengujian, evaluasi dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini menjelaskan beberapa teori yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan sistem yang akan dibangun. Perancangan sistem meliputi perancangan data dan alur proses dari sistem itu sendiri.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan sistem yang telah ditentukan sebelumnya.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dari metode yang ditawarkan dalam tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Bab ini membahas teori-teori yang menjadi dasar pembuatan tugas akhir.

2.1 *Transportation Problem* (TP)

Salah satu masalah yang sering terjadi di bidang perindustrian adalah masalah distribusi produk. Dalam melakukan proses distribusi produk, diperlukan perencanaan agar biaya distribusi dapat ditekan menjadi seminimal mungkin [21]. Faktor-faktor yang mempengaruhi besarnya biaya distribusi antara lain adalah banyaknya pemasok, banyaknya kebutuhan, dan estimasi biaya tiap rute. Proses perencanaan ini dikenal sebagai *Transportation Problem* (TP).

TP adalah salah satu dari masalah Pemrograman Linier di mana tujuannya adalah mengirimkan sejumlah kuantitas barang yang mulanya tersimpan di beberapa tempat pemasok ke tempat-tempat tujuan dengan biaya sekecil mungkin [5]. TP dapat dipresentasikan sebagai sebuah tabel seperti pada Tabel 2. 1 [15] dan direpresentasikan sebagai diagram jaringan sesuai pada Gambar 2. 1 [21][18]. TP terdiri atas sejumlah *supply m*, sejumlah *demand n*, biaya transportasi $C_{i,j}$ dan jumlah unit yang dialokasikan $X_{i,j.}$ Tujuan dari diagram jaringan dan tabel TP adalah untuk menemukan nilai dari variabel $X_{i,j}$, dimana variabel ini yang akan meminimalkan total biaya dari TP, seperti digambarkan pada persamaan (2.1). Berikut adalah notasi yang digunakan pada formula matematika TP [18].

m	Jumlah total *supply*
n	Jumlah total *demand*
S_i	Jumlah barang yang tersedia di *supply i*
D_j	Jumlah barang yang dibutuhkan *demand j*
C_{ij}	Biaya transportasi dari *supply i* ke *demand j*
X_{ij}	Jumlah unit yang dialokasikan dari *supply i* ke *demand j*

Fungsi objektif dari TP dapat diformulasikan sebagai berikut:

Min
$$Z = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} X_{ij} \qquad (2.1)$$

Batasan

$$\sum_{j=1}^{n} X_{ij} = S_i \qquad \text{untuk } i = 1,2,\ldots.m$$

$$\sum_{i=1}^{m} X_{ij} = D_j \qquad \text{untuk } j = 1,2,\ldots.,n$$

Dimana

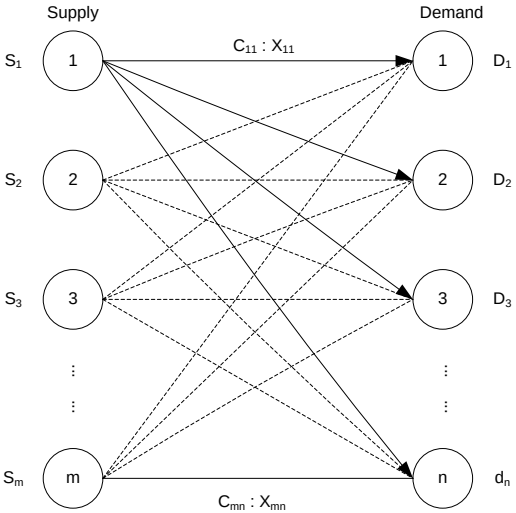
$$X_{ij} \geq 0 \qquad \text{Untuk semua } i,j$$

Sebuah TP dikatakan *balance* jika total *supply* sama dengan total *demand* seperti terlihat pada persamaan (2.2).

$$\sum_{i=1}^{m} S_i = \sum_{j=1}^{m} D_j \qquad (2.2)$$

Tabel 2. 1 Bentuk tabel dari TP

	***Demand* 1**	***Demand* 2**		***Demand* n**	
***Supply* 1**	C_{11} X_{11}	C_{12} X_{12}	…	C_{1n} X_{1n}	S_1
***Supply* 2**	C_{21} X_{21}	C_{22} X_{22}	…	C_{2n} X_{2n}	S_2
	. . .	. . .	…	. . .	
Supply m	C_{m1} X_{m1}	C_{m2} X_{m2}	…	C_{mn} X_{mn}	S_m
	D_1	D_2		D_n	



Gambar 2. 1 Diagram Jaringan dari Transportation Problem

2.2 *Total Opportunity Cost Matrix* (TOCM)

Kirca dan Satir [7] mengembangkan metode bernama *Total Opportunity Cost Matrix* (TOCM). TOCM merubah orisinal matriks menjadi inisial matriks dengan cara menjumlahkan antara matriks *row opportunity cost* dan matriks *coloum opportunity cost*. Tabel 2. 2 adalah orisinal matriks. Tabel 2. 3 adalah matriks *row opportunity cost*, matriks ini didapat dengan cara mengurangi setiap baris dari orisinal matriks dengan biaya terkecil pada baris tersebut. Tabel 2. 4 adalah matriks *column opportunity cost*, matriks ini didapat dengan cara mengurangi setiap kolom dari orisinal matriks dengan biaya terkecil pada kolom tersebut. Tabel 2. 5 adalah TOCM matriks, dimana tabel ini adalah penjumlahan antara *row opportunity cost* dan *column opportunity cost* matriks.

Tabel 2. 2 Orisinal Matriks

	D1	D2	D3	D4	*Supply*
S1	19	30	50	12	7
S2	70	30	40	60	10
S3	40	10	60	20	18
Demand	5	8	7	15	

Tabel 2. 3 Matriks *Row Opportunity Cost*

	D1	D2	D3	D4	*Supply*
S1	7	18	38	0	7
S2	40	0	10	30	10
S3	30	0	50	10	18
Demand	5	8	7	15	

Tabel 2. 4 Matriks *Column Opportunity Cost*

	D1	D2	D3	D4	*Supply*
S1	0	20	10	0	7
S2	51	20	0	48	10
S3	21	0	20	8	18
Demand	5	8	7	15	

Tabel 2. 5 Matriks TOCM

	D1	D2	D3	D4	*Supply*
S1	7	38	48	0	7
S2	91	20	10	78	10
S3	51	0	70	18	18
Demand	5	8	7	15	

2.3 *Initial Base Feasible Solution* (IBFS)

Transportation Problem (TP) digunakan untuk mendapatkan solusi paling optimal. Ada dua proses untuk mencari solusi optimal dari TP. Pertama adalah mencari IBFS dan kedua adalah mencari solusi optimal dari IBFS menggunakan *stepping stone* [5]. IBFS adalah penerapan sebuah algoritma tertentu pada permasalahan TP sehingga mendapatkan solusi awal [1]. Setelah

mendapatkan hasil dari penerapan algoritma IBFS, solusi awal yang dihasilkan tersebut kemudian diproses pada tahap s*tepping stone*.

Secara umum, tahapan *stepping stone* membutuhkan waktu yang lama. Hal ini menyebabkan para peneliti banyak melakukan riset untuk menciptakan algoritma IBFS untuk mendapatkan hasil seoptimal mungkin. Algoritma IBFS yang baik adalah algoritma yang dapat menghasilkan solusi dengan biaya yang paling kecil. Semakin mendekati angka minimal, semakin baik pula algoritma tersebut [15].

2.4 Algoritma *Vogel Approximation Method* (VAM)

VAM merupakan algoritma yang dapat menemukan solusi yang baik untuk pencarian IBFS [1]. Pada algoritma VAM, alokasi diberikan dengan membandingkan *penalty* setiap baris dan kolom. *Flowchart* algoritma VAM ditunjukkan pada Gambar 2. 2. Secara garis besar, langkah-langkah pada algoritma VAM adalah sebagai berikut :

Langkah 1 Membuat matriks orisinal awal dengan ukuran matriks m_x_n. Pastikan bahwa total *supply* dan *demand* sama. Jika berbeda, maka tambahkan *dummy* pada baris atau kolom agar seimbang.

Langkah 2 Setiap baris dan kolom dihitung nilai penaltinya (*P*). Nilai penalty didapatkan dengan cara mengurangkan dua biaya terkecil. Seperti pada persamaan (2.3).

$$P = C_2 - C_1 \tag{2.3}$$

Dimana C_1 adalah biaya terkecil pertama

C_2 adalah biaya terkecil kedua

Langkah 3 Pilih penalti terbesar.

Langkah 4 Alokasikan unit sebanyak mungkin pada *cell* dengan biaya terkecil.

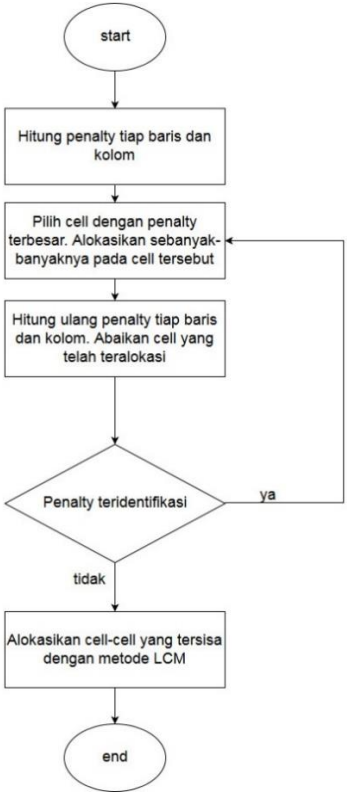
Langkah 5 Hitung kembali dan sesuaikan jumlah unit *supply* serta *demand*, kemudian eliminasi baris atau kolom yang jumlah unit *supply/demand* sudah nol. Eliminasi dengan cara mencoret menandakan bahwa baris atau kolom tersebut sudah terpenuhi

kebutuhannya (*satisfy*). Baris atau kolom yang sudah dicoret, tidak akan diikutkan pada perhitungan berikutnya.

Langkah 6 Hitung kembali penalti.

Langkah 7 Lakukan kembali langkah 3 – 7 hingga jumlah unit semua *supply/demand* sudah nol, maka kebutuhannya telah terpenuhi (*satisfy*).

Langkah 8 Setelah semua telah *satisfy*, kalikan biaya pada sel dengan unit yang dialokasikan pada *cell* tersebut untuk mendapatkan total biaya TP.



Gambar 2. 2 *Flowchart* algoritma VAM

2.5 Algoritma *Total Differences Method 1* (TDM1)

TDM1 adalah metode yang pengerjaannya dengan cara menghitung penalty baris. Penalti sendiri didapatkan dari menghitung total perbedaan antara biaya terkecil dengan semua biaya yang ada pada baris tersebut [6]. Langkah-langkah TDM1 adalah sebagai berikut:

Langkah 1 Buatlah matriks orisinal transportasi awal dengan ukuran matriks *m x n*. Jika total *supply* tidak sama dengan total *demand*, maka tambahkan *dummy* pada baris atau kolom.

Langkah 2 Hitung penalti untuk setiap baris (α_i). Nilai penalti adalah total perbedaan antara biaya terkecil dengan semua biaya yang ada pada baris tersebut, seperti pada persamaan (2.4).

$$\alpha_i = \sum_j (C_{ij} - C_r) \quad (2.4)$$

Dimana C_r = min{C_{ij} dan *i* adalah baris}

Langkah 3 Pilih penalti terbesar.

Langkah 4 Alokasikan unit sebanyak mungkin pada *cell* dengan biaya terkecil.

Langkah 5 Perbarui jumlah unit *supply* dan *demand*, kemudian eliminasi baris atau kolom yang jumlah unitnya sudah nol, berarti baris atau kolom tersebut sudah terpenuhi kebutuhannya (*satisfy*). Baris atau kolom yang sudah dieliminasi, tidak akan diikutkan pada perhitungan berikutnya.

Langkah 6 Hitung kembali penalti.

Langkah 7 Ulangi langkah 3 – 7 hingga jumlah unit semua baris dan kolom sudah nol, maka kebutuhannya telah terpenuhi (*satisfy*).

Langkah 8 Terakhir, hitung total biaya TP dengan cara mengalikan antara biaya pada sel dan unit yang dialokasikan pada *cell* tersebut.

2.6 Algortima *Incessant Allocation Method*

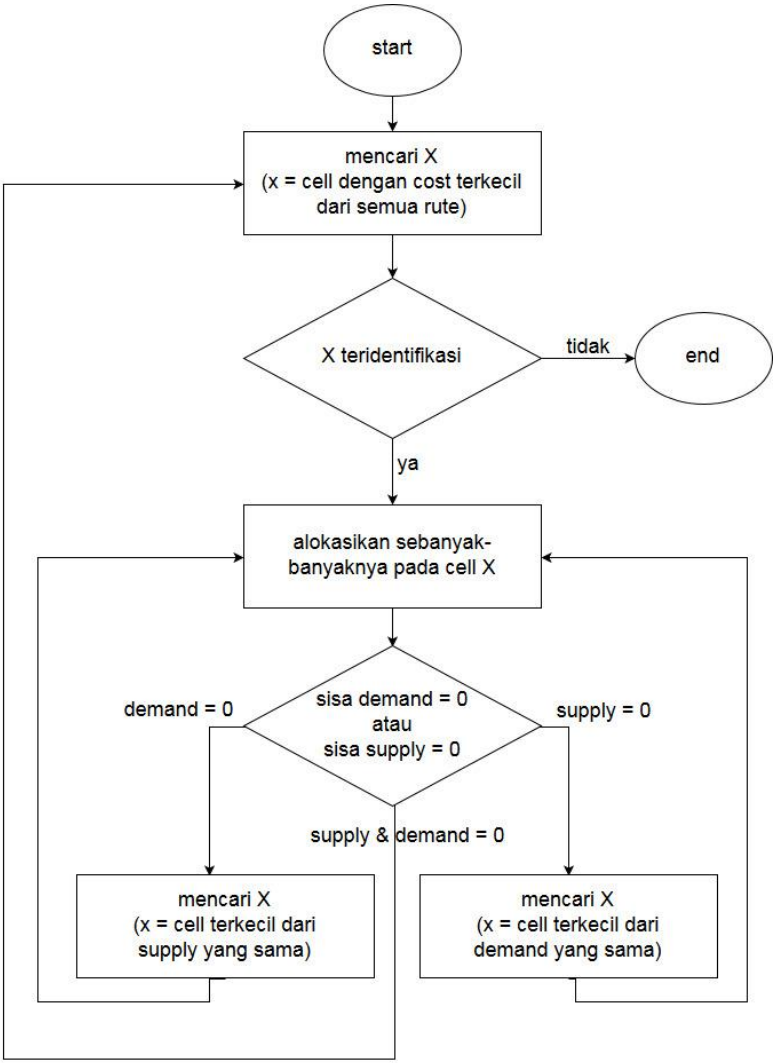
Incessant Allocation Method (IAM) adalah salah satu algoritma IBFS yang ditemukan oleh Mollah Mesbahuddin pada tahun 2016 [21]. *Flowchart* IAM dapat dilhat pada *Gambar 2. 3*.

Secara garis besar, langkah-langkah dalam algoritma IAM adalah sebagai berikut:

Langlah 1 Buatlah matriks orisinal transportasi awal dengan ukuran matriks *m x n*.
Langkah 2 Cari sel dengan biaya terkecil (B_t), seperti pada persamaan (2.5), dan alokasikan sebanyak mungkin pada sel ini.

$$B_t = min(C_{ij}) \quad (2.5)$$

Langkah 3 Perbarui jumlah unit *supply* dan *demand*, dan ikuti langkah berikut :
Kasus 1 : Jika alokasi $x_{ij} = s_i$, baris ke-*i* dieliminasi dan d_j dikurangi menjadi (d_j-s_i). Sekarang dilanjutkan dengan mengalokasikan sisa unit ke biaya terkecil pada kolom ke-*j* secara terus menerus. Jika kolom ke-*j* sudah *satisfy*, lanjutkan dengan mengalokasikan sisa unit ke biaya terkecil pada baris ke-*k* secara terus menerus.
Kasus 2 : Jika alokasi $x_{ij} = d_j$, kolom ke-*j* dieliminasi dan s_i dikurangi menjadi (s_i-d_j). Sekarang dilanjutkan dengan mengalokasikan sisa unit ke biaya terkecil pada baris ke-*i* secara terus menerus. Jika baris ke-*i* sudah *satisfy*, lanjutkan dengan mengalokasikan sisa unit ke biaya terkecil pada kolom ke-k secara terus menerus.
Kasus 3 : Jika alokasi $x_{ij} = s_i = d_j$, maka baris ke-*i* dan kolom ke-*j* eliminasi.
Kasus 4 : Jika baris atau kolom yang akan dialokasikan sudah penuh, kembali ke langkah 2.
Langkah 4 Terakhir, hitung biaya total transportasi dengan cara mengalikan biaya dan alokasi unit.



Gambar 2. 3 *Flowchart* Algoritma IAM

2.7 Algoritma *Incessant Allocation Method – Priority Value*

Incessant Allocation Method – Priority Value (IAM-PV) merupakan algoritma IBFS yang menggunakan metode pengalokasian terus menerus pada tiap rute dengan memperhatikan nilai prioritas terbesar dari perbandingan hitungan *row priority value* dan *column priority value*, hingga semua rute terisi [21]. *Flowchart* algoritma IAM-PV dapat dilihat pada Gambar 2. 4. Secara garis besar, langkah-langkah pada algoritma IAM-PV adalah sebagai berikut :

Langkah 1 Hitung *row priority value* dan *column priority value* semua *cell* yang masih kosong. Apabila semua *cell* telah terisi maka lanjut ke langkah 6

Langkah 2 Cari *cell* dengan priority value terbesar dari semua *cell*. Untuk selanjutnya *cell* ini dinamakan dengan *cell* X.

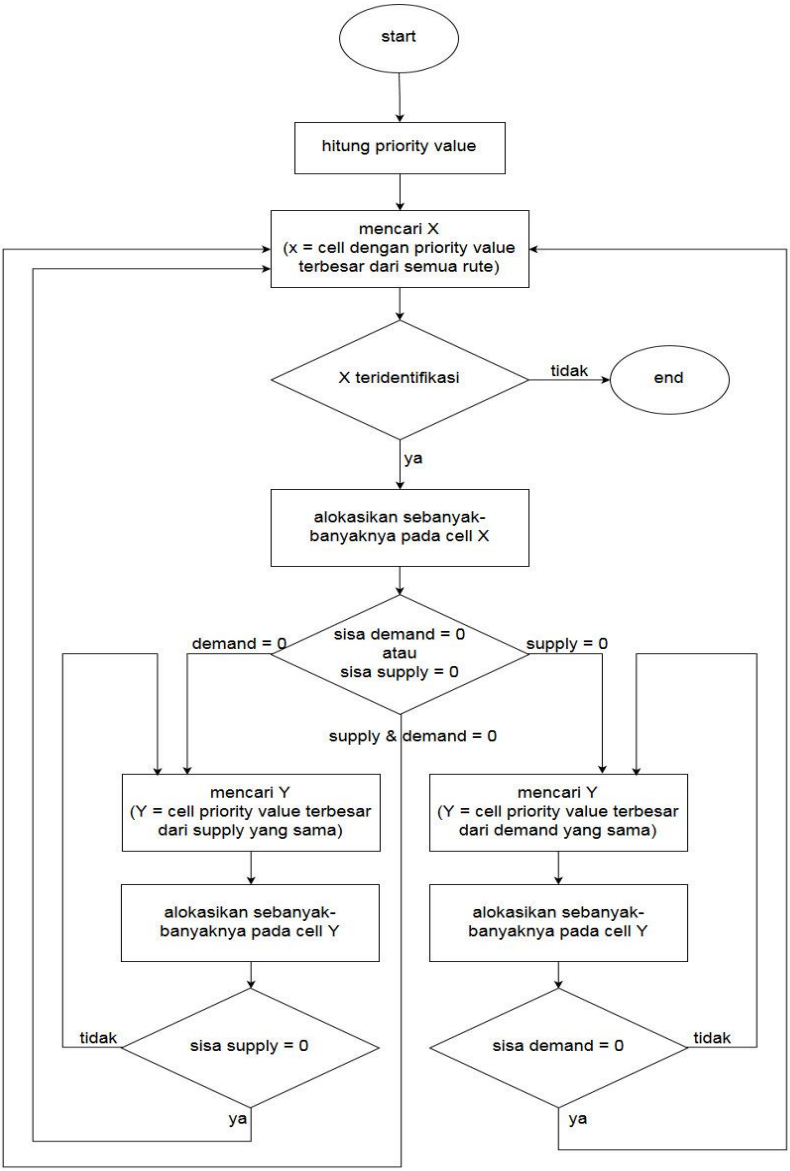
Langkah 3 Alokasikan sebanyak mungkin pada *cell* X

Langkah 4 Perhatikan jumlah *supply* dan *demand* yang tersisa setelah mengalokasikan *cell* X, kemudian lakukan langkah dengan memperhatikan percabangan sebagai berikut

i. Apabila jumlah *demand* dan *supply* keduanya habis, maka kembali ke langkah 1
ii. Apabila jumlah *supply* masih tersisa, maka cari *cell* dengan *priority value* terbesar berikutnya dalam satu baris
iii. Apabila jumlah *demand* masih tersisa, maka cari *cell* dengan *priority value* terbesar berikutnya dalam satu kolom

Langkah 5 Kembali ke langkah 1

Langkah 6 Hitung jumlah biaya pengiriman.



Gambar 2. 4 *Flowchart* Algoritma IAM-PV

2.8 Algoritma *Juman and Hoque Method (JHM)*

Berbeda dengan VAM, JHM tidak memerlukan transportasi yang *balance* dan tidak perlu menghitung penalty [1].

Langkah-langkah JHM adalah sebagai berikut:

Langkah 1 Buatlah matriks orisinal transportasi awal dengan ukuran *m x n*.

Langkah 2 Pada setiap kolom, cari *cell* dengan biaya terkecil (B_t), seperti pada persamaan (2.4) dan alokasikan jumlah unit *demand* pada sel tersebut.

$$B_t = min(C_{ij}) \tag{2.6}$$

Langkah 3 Untuk alokasi pada setiap baris, cek apakah jumlah total unit kurang atau sama dengan jumlah *supply*, jika sama, maka menuju ke langkah 9.

Langkah 4 Untuk tiap alokasi pada *unmet row* (jumlah total unit lebih besar dari *supply*), hitung selisih antara 2 biaya terkecil pada kolom tersebut, dan pilih selisih yang terkecil. Jika hanya ada satu baris yang *unmet row*, menuju ke langkah 7, jika tidak maka menuju langkah 5.

Langkah 5 Cek jika ada sebuah atau beberapa sel pada *unmet row* yang tidak mengandung biaya unit terkecil kedua yang berhubungan dengan perbedaan terkecil antara biaya unit yang terkecil kedua dan biaya unit yang terkecil pada kolom untuk masing-masing alokasi pada *unmet row* yang lain, seperti pada teorema 3. Jika ada baris seperti ini, identifikasi *unmet row* sebelumnya dan kemudian menuju ke langkah 7.

Langkah 6 Pilih dua baris *unmet row*. Untuk setiap baris yang *unmet row*, cari selisih antara 2 biaya terkecil. Asumsi perbedaan terkecil untuk *unmet row* yang berhubungan dengan biaya terkecil c_1 dan perbedaan terkecil untuk *unmet row* yang lain berhubungan dengan biaya terkecil e_1. Asumsi c_1, c_2 dan c_3 adalah biaya terkecil pertama, kedua dan ketiga pada kolom pertama dan e_1, e_2 dan e_3 adalah pertama, kedua dan ketiga pada kolom kedua. Jika $(c_3 - c_1) > (e_3 - e_2)$, maka *unmet row* adalah baris yang terdapat c_1, jika tidak maka *unmet row* adalah baris yang terdapat e_1 seperti pada teorema 4.

Langkah 7 Mempertimbangkan *unmet row* dari langkah 4, 5 atau 6 kemudian transfer maksimal sisa unit dari *cell* dengan biaya terkecil ke sel dengan biaya terkecil kedua pada kolom tersebut seperti pada teorema 1 dan 2. Lanjutkan transfer ini hingga tidak ada sisa unit.
Langkah 8 Eliminasi baris yang jumlah unitnya sudah nol, maka baris sudah terpenuhi kebutuhannya *(satisfy)*. Baris yang sudah dieliminasi, tidak akan diikutkan pada perhitungan berikutnya, kemudian kembali ke langkah 3.
Langkah 9 Terakhir, hitung total biaya TP dengan cara mengalikan antara biaya pada sel dan unit yang dialokasikan pada sel tersebut.

2.9 Algoritma *Total Opportunity Cost Matrix* – Minimal Total (TOCM-MT)

Metode IBFS *Total Opportunity Cost Matrix – Minimal Total* (TOCM-MT) adalah kombinasi dari TOCM dan modifikasi TDM1 [2]. Langkah-langkah dari metode TOCM-MT adalah sebagai berikut:
Langkah 1: Buat matriks orisinal transportasi awal dengan ukuran matriks *m x n*. Jika total *supply* tidak sama dengan total *demand*, maka tambahkan *dummy* baris atau kolom.
Langkah 2: Buat matriks *row opportunity cost* dari matriks orisinal TP dengan cara mengurangi setiap biaya dengan biaya terkecil pada baris tersebut.
Langkah 3: Buat matriks *column opportunity cost* dari matriks orisinal TP dengan cara mengurangi setiap biaya dengan biaya terkecil pada kolom tersebut.
Langkah 4: Buat matriks TOCM dimana entri nya adalah penjumlahan dari matriks *row opportunity cost* dan *column opportunity cost.*
Langkah 5: Hitung penalti untuk setiap baris (P_i). Nilai penalti adalah total perbedaan antara biaya terkecil (LC_i) dengan semua

biaya yang ada pada baris tersebut, seperti pada persamaan (2.7) dan (2.8).

$$LC_i = min\ (C_{ij})\ ,\ \ j = 1. \ .n \tag{2.7}$$

$$P_i = \sum_{j=1}^{n} (C_{ij} - LC_i) \tag{2.8}$$

Langkah 6: Pilih penalti terbesar (HP) seperti pada persamaan (2.9). Jika ada nilai HP yang sama, maka lakukan hal berikut secara berurutan: (i) pilih HP dengan nilai C_{ij} terkecil. (ii) jika ada nilai yang sama pada (i), pilih HP dengan TC_i terbesar, seperti pada persamaan (2.10). (iii) jika ada nilai yang sama pada (ii), pilih HP dengan alokasi maksimal pada $X_{i,j}$.

$$HP = max\ (Pi), \quad i = 1 \ . \ . \ m \tag{2.9}$$

$$TC_i = \sum_{j=1}^{n} C_{ij} \tag{2.10}$$

Langkah 7: Pilih biaya terkecil (LC) dari HP. Jika ada nilai LC yang sama, pilih LC dengan alokasi maksimal pada $X_{i,j}$.

Langkah 8: Periksa nilai LC. Jika LC tidak sama dengan nol, maka ke langkah 9, jika LC sama denga nol, maka pilih HP antara HP pertama (HP_1) atau HP kedua (HP_2). Pilih HP dengan cara membandingkan tiap biaya pada sel HP_1 dan HP_2. C_{1j} adalah biaya pada HP_1 dan C_{2j} adalah biaya pada HP_2. GV_{1j} bernilai 1 jika biaya pada HP_1 lebih besar dari biaya pada HP_2 dan 0 jika biaya pada HP_1 lebih kecil dari biaya pada HP_2. GV_{2j} bernilai 1 jika biaya pada HP_1 lebih kecil dari pada HP_2 dan 0 jika biaya pada HP_1 lebih besar dari pada HP_2. $TotalGV_{1j}$ adalah penjumlahan dari GV_{1j} seperti pada persamaan (2.11) dan $TotalGV_{2j}$ adalah penjumlahan dari GV_{2j} seperti pada persamaan (2.12). HP adalah HP_1 jika $TotalGV_{1j}$ lebih besar dari $TotalGV_{2j}$ dan HP adalah HP_2 jika $TotalGV_{1j}$ lebih kecil $TotalGV_{2j}$ seperti pada persamaan (2.13).

$$TotalGV_{1j} = \sum_{j=1}^{n} GV_{1j} \tag{2.11}$$

Dimana

$$GV_{1j} = \begin{cases} 1 & if\ C_{1j} \geq C_{2j},\ \ j = 1,2,..n \\ 0 & if\ C_{1j} < C_{2j},\ \ j = 1,2,..n \end{cases}$$

$$TotalGV_{2j} = \sum_{j=1}^{n} GV_{2j} \tag{2.12}$$

Dimana

$$GV_{2j} = \begin{cases} 1 & if\ C_{1j} < C_{2j},\ \ j = 1,2,..n \\ 0 & if\ C_{1j} \geq C_{2j},\ \ j = 1,2,..n \end{cases}$$

$$HP = \begin{cases} HP_1 & if\ \ TotalGV_{1j} \geq TotalGV_{2j} \\ HP_2 & if\ \ TotalGV_{1j} < TotalGV_{2j} \end{cases} \tag{2.13}$$

Langkah 9: Alokasikan unit X_{ij} sebanyak mungkin pada sel dengan biaya terkecil dari HP.
Langkah 10: Perbarui jumlah unit *supply* dan *demand*, kemudian eliminasi baris atau kolom yang jumlah unitnya sudah nol, berarti baris atau kolom tersebut sudah terpenuhi kebutuhannya (*satisfy*). Baris atau kolom yang sudah dieliminasi, tidak akan diikutkan pada perhitungan berikutnya.
Langkah 11: Hitung kembali penalti.
Langkah 12: Ulangi langkah 6 – 11 hingga semua baris dan kolom jumlah unitnya sudah nol, sudah terpenuhi kebutuhannya (*satisfy*).
Langkah 13: Terakhir, hitung total biaya TP (TCTP) dengan cara menggabungkan antara hasil TOCM-MT dan orisinal TP, seperti pada persamaan (2.14).

$$TCTP = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} X_{ij} \tag{2.14}$$

2.10 Algoritma Bilqis Chastine Erma (BCE)

Metode IBFS selanjutnya adalah metode Bilqis Chastine Erma (BCE). Langkah-langkah dari metode BCE adalah sebagai berikut:

Langkah 1: Buatlah matriks orisinal transportasi awal dengan ukuran *m x n* dengan biaya C_{ij}, *supply* S_i dimana i = 1..*m* dan *demand* D_j dimana j=1..*n*. Untuk setiap baris, statusnya adalah *Not Satisfied* (NS). Tidak perlu menambahkan *dummy* jika jumlah total *supply* tidak sama dengan jumlah total *demand*.

Langkah 2: Alokasikan *demand* D_j dari kolom *j* ke X_{ij}, dimana C_{ij} adalah *First Least Cost (FLC)*$_j$ untuk setiap kolom, seperti pada persamaan (2.15).

$$FLC_j = min(C_{ij}),\ \ i=1..m \tag{2.15}$$

Langkah 3: Pada tiap baris, jika total alokasi X_{ij} pada baris *i* (TA_i) lebih besar dari *supply* S_i maka ubah status baris menjadi *Excess Row* (ER). ER adalah baris yang total *demand*-nya lebih besar dari *supply*, sehingga kelebihan isi sel harus dipindah dari *First Least Row* (FLR) ke *Second Least Row* (SLR). TA_i dihitung dengan menggunakan persamaan (2.16).

$$TA_i = \sum_{j=1}^{n} X_{ij} \tag{2.16}$$

Langkah 4: Tiap *cell* pada ER, hitung perbedaan (*Diff*) antara FLC_j dengan SLC_j seperti pada persamaan (2.17).

$$Diffj = SLC_j - FLC_j \tag{2.17}$$

Langkah 5: Pilih *cell* dengan *Diff* terkecil (*Smallest Diff* - SD) seperti pada persamaan (2.18). Baris SD menjadi *First Least Row* (FLR) dan baris SLC menjadi *Second Least Row* (SLR). Kelebihan isi *cell* pada FLC akan dipindah ke SLC.

$$\text{SD} = min(Diff_j) \tag{2.18}$$

Langkah 6: Jika TP adalah *balance* maka ke langkah 7, jika tidak menuju langkah 17.

Langkah 7: Jika FLC tidak sama dengan SLCj dan *Least Supply* (LS) tidak sama dengan *Second Supply* (SS) maka menuju langkah 8, jika tidak menuju langkah 15.

Langkah 8: Cek status dari SLR, Jika status SLR adalah ER maka menuju langkah 9, jika tidak menuju langkah 11.

Langkah 9: Alokasikan maksimal unit X_{ij} ke FLC dan pindah semua sisanya ke SLC.

Langkah 10: Jika jumlah total unit pada FLR (TUFLR) pada baris i sama dengan *supply* FLR maka eliminasi FLR dan ganti status

menjadi *satisfy* (berarti baris tersebut sudah terpenuhi kebutuhannya dan tidak akan diikutkan pada perhitungan berikutnya) kemudian ke langkah 18, jika TUFLR tidak sama dengan *supply* FLR maka ke langkah 18. TUFLR dihitung dengan menggunakan persamaan (2.19).

$$TUFLR = \sum_{j=1}^{n} X_{ij} \quad (2.19)$$

Langkah 11: Hitung *modulo* antara *Diff Supply (DS)* and $Diff_j$ seperti pada persamaan (2.20) dan (2.21). Jika *modulo* sama dengan nol, maka ke langkah 12, jika tidak ke langkah 9.

$$DS = |SS\text{-}LS| \quad (2.20)$$

$$Modulo = mod\ (DS, Diff_j) \quad (2.21)$$

Langkah 12: Hitung biaya total FLR (TCFLR) seperti pada persamaan (2.22) and biaya total SLR (TCSLR) seperti pada persamaan (2.23). Jika TCFLR lebih besar daripada TCSLR maka ke langkah 9, jika tidak maka ke langkah 13.

$$TCFLR = \sum_{j=1}^{n} C_{1,j} \quad (2.22)$$

$$TCSLR = \sum_{j=1}^{n} C_{2,j} \quad (2.23)$$

Dimana $C_{1,j}$ adalah biaya pada FLR and $C_{2,j}$ adalah biaya pada SLR.

Langkah 13: Alokasikan maksimal unit X_{ij} dari SD ke SLC dan simpan sisanya di SD.

Langkah 14: Jika jumlah total unit pada SLR (TUSLR) pada baris *i* sama dengan *supply* SLR maka eliminasi SLR dan ganti status menjadi *satisfy* (berarti baris tersebut sudah terpenuhi kebutuhannya dan tidak akan diikutkan pada perhitungan berikutnya) kemudian ke langkah 18, jika TUSLR tidak sama dengan *supply* SLR maka ke langkah 18. TUSLR dihitung dengan menggunakan persamaan (2.24).

$$TUSLR = \sum_{j=1}^{n} X_{ij} \quad (2.24)$$

Langkah 15: Jika FLC tidak sama dengan SLCj dan *Least Supply* (LS) sama dengan *Second Supply* (SS) maka ke langkah 9.

Langkah 16: Jika FLC sama dengan SLC maka hitung TCFLR and TCSLR. Jika TCFLR lebih besar daripada TCSLR maka ke langkah 9, jika tidak maka pindahkan semua unit X_{ij} dari FLC ke SLC. Ganti FLC dengan SLC dan SLC dengan SLC berikutnya.

Langkah 17: Lakukan langkah berikut pada saat TP adalah *unbalanced*.

Jika TP adalah *unbalanced*, maka cek status SLR
Jika status SLR adalah ER maka ke langkah 9, jika tidak
jika TUNFLR lebih besar dari *supply* LR maka ke langkah 13
Jika tidak ke langah 9.

TUNFLR dihitung menggunakan persamaan (2.25).

$$TUNFLR = \sum_{j=1}^{n} X_{1,j} - \sum_{j=1}^{n} X_{2,j} \quad (2.25)$$

Dimana TUNLR adalah total unit saat ini pada FLR
$X_{1,j}$ adalah alokasi X_{ij} pada LR
$X_{2,j}$ adalah alokasi X_{ij} pada SR

Langkah 18: Jika hanya terdapat satu baris yang NS, maka alokasikan unit yang tersisa pada sel yang tepat dan hitung total biaya dari TP (TC) seperti pada persamaan (2.26), jika tidak maka ke langkah 19.

$$TC = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \quad (2.26)$$

Langkah 19: Jika masih ada ER, maka ke langkah 4, jika tidak ke langkah 3.

2.11 Metode Evaluasi

Tiga pengukuran digunakan untuk mengevaluasi kinerja dari metode yang diusulkan, dua pengukuran dari jurnal JHM [1] dan satu pengukuran adalah akurasi. Pengukuran – pengukuran tersebut adalah sebagai berikut:

1. Pengukuran *Improvement percentage* (Ip) adalah ukuran persentase peningkatan kinerja metode yang diusulkan terhadap metode yang telah ada, di mana terdapat 3 nilai angka, yaitu angka positif, angka nol, dan angka negatif. Angka positif berarti ada persentase peningkatan metoda yang diusulkan (MU) terhadap metoda yang telah ada (MA), dengan kata lain bahwa total biaya MU lebih rendah dari pada

total biaya MA. Angka nol berarti tidak ada persentase peningkatan MU terhadap MA, dengan kata lain bahwa total biaya MU sama dengan total biaya MA. Angka negatif berarti tidak ada persentase peningkatan, tapi ada persentase penurunan MU terhadap MA, dengan kata lain bahwa total biaya MU lebih tinggi dari total biaya MA. Pengukuran *improvement percentage* (Ip) dirumuskan seperti pada persamaan 2.27.

$$Ip = \frac{MA - MU}{MA} x100 \qquad (2.27)$$

2. Pengukuran *Deviation percentage* (Dv) adalah ukuran persen penyimpangan antara total biaya sebuah metode (Met) dengan total biaya solusi optimal (Op), di mana terdapat 2 nilai angka, yaitu angka positif dan angka nol. Angka positif berarti ada penyimpangan dari Met terhadap Op, dengan kata lain bahwa total biaya Met lebih tinggi dari Op. Angka nol berarti tidak ada penyimpangan dari metode Met terhadap Op, dengan kata lain bahwa total biaya Met sama dengan Op. Pengukuran *deviation percentage* (Dv) dirumuskan eperti pada persamaan 2.28.

$$Dv = \frac{Met - Op}{Op} x100 \qquad (2.28)$$

3. Pengukuran Akurasi (Ak) adalah persen jumlah total data yang mencapai solusi optimal terhadap jumlah total data yang ada, seperti pada persamaan 2.29.

$$Ak = \frac{Jumlah\_total\_data\_yang\_mencapai\_solusi\_optimal}{Jumlah\_total\_data} x100 \qquad (2.29)$$

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan menjelaskan tentang analisis dan perancangan sistem untuk mencapai tujuan dari tugas akhir. Perancangan ini meliputi hal-hal yang akan dimodifikasi pada algoritma IAM-PV.

3.1 Analisis Algoritma IAM-PV

Untuk menganalisis algortima IAM-PV, akan dijelaskan tahap pengerjaan algoritma tersebut dengan menggunakan 1 data yang diambil dari 34 data berbagai jurnal. Data yang diambil merupakan urutan data ke-15 berasal dari jurnal Uddin dengan ukuran 3 x 4. Berikut adalah tabel solusi optimal data jurnal ke-15 [5], yang akan dijadikan acuan hasil terbaik. Solusi optimal ini diperoleh dari *tools* ToraSystem.

Tabel 3. 1 Tabel Solusi Optimal Data Jurnal ke-15

	D1	**D2**	**D3**	**D4**	***Supply***
S1	19 **5**	30 **0**	50 **0**	12 **2**	**7**
S2	70 **0**	30 **3**	40 **7**	60 **0**	**10**
S3	40 **0**	10 **5**	60 **0**	20 **13**	**18**
Demand	**5**	**8**	**7**	**15**	

Dari tabel diatas, didapatkan total biaya solusi optimal yang perhitungannya adalah sebagai berikut :

Total Biaya = (19 x 5) + (12 x 2) + (30 x 3) + (40 x 7) + (10 x 5) + (20 x 13)
= 799

Selanjutnya, data yang sama akan diproses sesuai dengan langkah-langkah algoritma IAM-PV seperti yang dijelaskan pada

sub-bab 2.7**Error! Reference source not found.**. Langkah-langkah pada algoritma IAM-PV adalah sebagai berikut :

Langkah 1 Hitung *row priority value* (RPV) dan *column priority value* (CPV) semua *cell*, seperti yang ditunjukan pada ***Tabel 3. 2***.

Langkah 2 Cari *cell* dengan *priority value* terbesar (X) dari semua *cell*. X = 90 pada *cell* (3,2).

Langkah 3 Alokasikan sebanyak mungkin pada *cell* X. Alokasi 8 unit pada *cell* (3,2), seperti yang tunjukan pada ***Tabel 3. 3***.

Langkah 4 Hitung kembali jumlah *supply* dan *demand*. Perhatikan kembali jumlah *supply* dan *demand*. *Demand* sudah *satisfy* maka dieliminasi. *Supply* belum *satisfy*, masih tersisa 10 unit. *Cell* selanjutnya dengan *priority value* terbesar (X) di baris sama adalah 50 pada *cell* (3,4).

Langkah 3 Alokasikan sebanyak mungkin pada *cell* X. Alokasi 10 unit pada *cell* (3,4), seperti yang tunjukan pada ***Tabel 3. 3***.

Langkah 4 Hitung kembali jumlah *supply* dan *demand*. Perhatikan kembali jumlah *supply* dan *demand*. *Supply* dan *demand* sudah *satisfy*, maka lanjut ke langkah 1.

Tabel 3. 2 Perhitungan RPV dan CPV

		D1	**D2**	**D3**	**D4**
S1	RPV	35	-9	-89	63
	CPV	72	-20	0	56
S2	RPV	-80	80	40	-40
	CPV	-81	-20	30	-88
S3	RPV	-30	**90**	-110	50
	CPV	9	40	-30	32

Tabel 3. 3 Iterasi Pertama Algortima IAM-PV

	D1	**D2**	**D3**	**D4**	***Supply***
S1	19	30 **0**	50	12	**7**
S2	70	30 **0**	40	60	**10**
S3	40 **0**	10 **8**	60 **0**	20 **10**	**18**
Demand	**5**	**8**	**7**	**15**	

Pada iterasi pertama, hasil pengalokasian unit yang sudah dilakukan terlihat berbeda dengan solusi optimal. Alokasi pada cell (3,2) pada solusi optimal hanya sebesar 5 unit, sedangkan IAM-V melakukan alokasi sebanyak 8 unit. Hal ini tentu akan berpengaruh besar pada total biaya yang dihasilkan, sehingga algoritma IAM-PV belum mencapai hasil optimal. Hasil akhir dari algoritma IAM-PV pada data ini sebesar 859, dapat dilihat pada **Tabel 5. 6**. Oleh karena itu, pada sub-bab ini dapat disimpulkan bahwa algoritma IAM-PV masih belum efektif. Maka untuk memperbaiki ini, matriks awal yang dipakai adalah matrix TOCM, bukan matriks awal biasa.

3.2 Total Opportunity Cost Matrix – Priority Value (TOCM-PV)

TOCM-PV adalah gabungan dari *Total Opportunity Cost Matrix* (TOCM) dan IAM-PV. TOCM-PV mengubah matriks orisinal awal menjadi matriks TOCM. Kemudian matriks TOCM tersebut akan diproses dengan cara menghitung *row priority value* dan *column priority value*, lalu mencari *priority value* terbesar dari seluruh *cell*, sama seperti algoritma IAM-PV. Setelah pengalokasian menggunakan algoritma IAM-PV selesai, maka hitung biaya penalty dengan menggunakan matriks orisinal.

Tujuan diubahnya matriks orisinal menjadi matriks TOCM adalah agar setiap nilai yang ada dalam *cell* akan semakin jelas, dimana *cell* kecil akan terlihat makin kecil dan *cell* besar akan

bernilai lebih besar, sehingga hal ini mendukung proses pengalokasian dengan metode IAM-PV mendapatkan hasil yang optimal.

Semakin besar nilai *priority value* pada matriks tersebut maka semakin penting pula *cell* tersebut dialokasikan terlebih dahulu. Secara garis besar, langkah-langkah pada algoritma TOCM-PV adalah sebagai berikut :

Langkah 1: Buatlah matriks orisinal transportasi awal dengan ukuran matriks *m x n*. Jika total *supply* tidak sama dengan total *demand*, maka tambahkan *dummy* baris atau kolom.

Langkah 2: Buatlah matriks *row opportunity cost* dari matriks orisinal TP dengan cara mengurangi setiap biaya dengan biaya terkecil pada baris tersebut.

Langkah 3: Buatlah matriks *column opportunity cost* dari matriks orisinal TP dengan cara mengurangi setiap biaya dengan biaya terkecil pada kolom tersebut.

Langkah 4: Buatlah matriks TOCM dimana entri nya adalah penjumlahan dari matriks *row opportunity cost* dan *column opportunity cost.*

Langkah 5: Hitung *Row Priority Value* (RPV) *dan Column Priority Value* (CPV) semua *cell* yang masih kosong, seperti yang ditujukan pada persamaan (3. 1) dan (3. 2). Apabila semua *cell* telah terisi maka lanjut ke langkah 9.

$$RPV(x,y) = \sum_{n}^{j=1} cost(x,j) - n * cost(x,y) \qquad (3.\ 1)$$

$$CPV(x,y) = \sum_{m}^{i=1} cost(i,y) - m * cost(x,y) \qquad (3.\ 2)$$

x,y = koordinat *cell*

n = banyaknya jumlah *demand*

m = banyaknya jumlah *supply*

Langkah 6: Cari *cell* dengan *priority value* terbesar dari semua *cell*. Untuk selanjutnya *cell* ini dinamakan dengan *cell* X.

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X

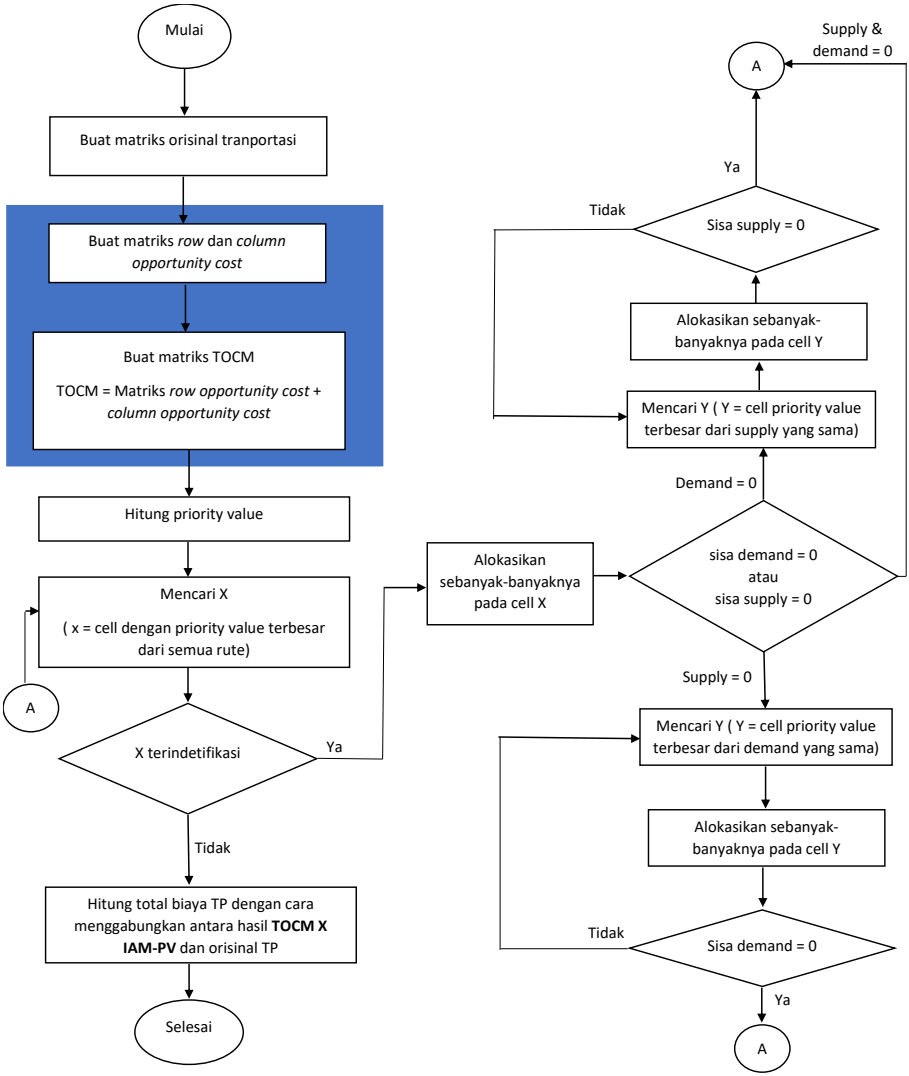
Langkah 8: Perbarui jumlah *supply* dan *demand* yang tersisa setelah mengalokasikan *cell* X. Eliminasi *supply* atau *demand* yang sudah *satisfy*, kemudian lakukan langkah dengan memperhatikan percabangan sebagai berikut:

i. Apabila jumlah *demand* dan *supply* keduanya habis atau *satisfy*, maka kembali ke langkah 5
ii. Apabila *supply* belum *satisfy*, cari *cell* selanjutnya dengan *priority value* terbesar (X) pada baris yang sama, kemudian kembali ke langkah 7
iii. Apabila semua *supply* dan *demand* sudah satisfy, lanjutkan ke langkah 9

Langkah 9: Terakhir, hitung total biaya TP (TCTP) dengan cara menggabungkan antara hasil TOCM-PV dan orisinal TP, seperti pada persamaan (3.3).

$$TCTP = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \tag{3.3}$$

Flowchart algoritma TOCM-PV dapat dilihat pada *Gambar 3. 1*. Berikut ini adalah contoh penerapan algoritma TOCM-PV terhadap data jurnal ke-15 [5].



Gambar 3. 1 *Flowchart* Algoritma TOCM-PV

Tabel 3. 4 Matriks Orisinal Data Jurnal ke-15

	D1	D2	D3	D4	*Supply*
S1	**19**	**30**	**50**	**12**	**7**
S2	**70**	**30**	**40**	**60**	**10**
S3	**40**	**10**	**60**	**20**	**18**
Demand	**5**	**8**	**7**	**15**	

Tabel 3. 5 Matriks TOCM Data Jurnal ke-15

	D1	D2	D3	D4	*Supply*
S1	**7**	**38**	**48**	**0**	**7**
S2	**91**	**20**	**10**	**78**	**10**
S3	**51**	**0**	**70**	**28**	**18**
Demand	**5**	**8**	**7**	**15**	

Iterasi Pertama

Langkah 5: Hitung RPV dan CPV semua *cell*, seperti yang ditunjukan pada ***Tabel 3. 6***.

Contoh RPV(2,3) = (91 + 20 + 10 + 78) – 4 * 10 = 159

CPV(1,2) =(38 + 20 + 0) – 3 *38 = -56

Langkah 6: Cari *cell* dengan *priority value* terbesar (X) dari semua *cell*. X = 159 pada *cell* (2,3).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 7 unit pada *cell* (2,3), seperti yang ditunjukan pada ***Tabel 3. 7***

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Apabila *supply* belum satisfy, cari *cell* selanjutnya dengan *priority value* terbesar (X) pada baris yang sama, kemudian kembali ke Langkah 7.

Supply belum *satisfy*, masih tersisa 3 unit. *Cell* selanjutnya dengan *priority value* terbesar (X) di baris yang sama adalah 119 pada *cell* (2,2).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 3 unit pada *cell* (2,2) seperti yang ditunjukan pada ***Tabel 3. 7***.

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Supply dan *demand* sudah *satisfy* keduanya, lanjut ke langkah 5.

Tabel 3. 6 Perhitungan RPV dan CPV pada Semua *Cell*

		D1	**D2**	**D3**	**D4**
S1	RPV	65	-59	-99	93
	CPV	128	-56	-16	96
S2	RPV	-165	119	**159**	-113
	CPV	-124	-2	98	-138
S3	RPV	-65	139	-141	67
	CPV	-4	58	-82	42

Tabel 3. 7 Iterasi Pertama

	D1		D2		D3		D4		*Supply*
S1	7		38		48	0	0		7
S2	91	0	20	**3**	10	**7**	78	0	10
S3	51		0		70	0	28		18
Demand	5		8		7		15		

Iterasi Kedua

Langkah 5: Hitung RPV dan CPV semua *cell,* seperti yang tunjukkan pada ***Tabel 3. 8***

Contoh RPV(1,2) = (7 + 38 + 0) – 3 * 38 = -69
CPV(3,2) =(38 + 0) – 2 *0 = 38.

Langkah 6: Cari *cell* dengan *priority value* terbesar (X) dari semua *cell*. X= 69 pada *cell* (3,2).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 5 unit pada *cell* (3,2), seperti yang ditunjukan pada ***Tabel 3. 9***.

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Apabila *supply* belum satisfy, cari *cell* selanjutnya dengan *priority value* terbesar (X) pada baris yang sama, kemudian kembali ke Langkah 7.

Supply belum *satisfy*, masih tersisa 13 unit. *Cell* selanjutnya dengan *priority value* terbesar (X) di baris yang sama adalah 15 pada *cell* (3,4).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 13 unit pada *cell* (3,4) seperti yang ditunjukan pada ***Tabel 3. 9***.

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Supply dan *demand* keduanya sudah *satisfy*, lanjut ke langkah 5.

Tabel 3. 8 Perhitungan RPV dan CPV pada Iterasi Kedua

		D1	**D2**	**D3**	**D4**
S1	RPV	24	-69		45
	CPV	44	-38		18
S2	RPV				
	CPV				
S3	RPV	-84	**69**		15
	CPV	-44	38		-18

Tabel 3. 9 Iterasi Kedua.

	D1		D2		D3		D4		*Supply*
S1	7		38	0	48	0	0		7
S2	91	0	20	**3**	10	**7**	78	0	10
S3	51	0	0	**5**	70	0	28	**13**	18
Demand	5		8		7		15		

Iterasi ketiga

Langkah 5: *Hitung RPV dan CPV semua cell, seperti yang ditunjukan pada* ***Tabel 3. 10***.

Contoh RPV(1,1) = (7 + 0) – 2 * 7 = -7

RPV(1,4) =(7 + 0) – 2 *0 = 7.

Langkah 6: Cari *cell* dengan *priority value* terbesar (X) dari semua *cell*. X= 7 pada *cell* (1,4).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 2 unit pada *cell* (1,4), seperti yang ditunjukan pada ***Tabel 3. 11***.

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Apabila *supply* belum satisfy, cari *cell* selanjutnya dengan *priority value* terbesar (X) pada baris yang sama, kemudian kembali ke Langkah 7.

Supply belum *satisfy*, masih tersisa 5 unit. *Cell* selanjutnya dengan *priority value* terbesar (X) di baris yang sama adalah 0 pada *cell* (1,1).

Langkah 7: Alokasikan sebanyak mungkin pada *cell* X. Alokasi 5 unit pada *cell* (1,1) seperti yang ditunjukan pada ***Tabel 3. 11***.

Langkah 8: Hitung kembali jumlah *supply* dan *demand.* Perhatikan kembali jumlah *supply* dan *demand*.

Jika semua supply dan *demand* sudah *satisfy*, lanjut ke langkah 9.

Tabel 3. 10 Perhitungan RPV dan CPV pada Iterasi Ketiga

		D1	**D2**	**D3**	**D4**
S1	RPV	-7			**7**
	CPV	0			0

Tabel 3. 11 Iterasi Ketiga

	D1		D2		D3		D4		Supply
S1	7	**5**	38	0	48	0	0	**2**	7
S2	91	0	20	**3**	10	**7**	78	0	10
S3	51	0	0	**5**	70	0	28	**13**	18
Demand	5		8		7		15		

Langkah 9: Hitung total biaya TP (TCTP) dengan cara menggabungkan antara hasil penalti TOCM-PV dan orisinal TP, seperti yang ditunjukan pada ***Tabel 3. 12***. Total biaya adalah 19 x 5) + (12 x 2) + (30 x 3) + (40 x 7) + (10 x 5) + (20 x 13) = 799, dan ini merupakan solusi optimal.

Tabel 3. 12 Hasil Akhir TOCM-PV pada Data Jurnal ke-15

	D1		D2		D3		D4		Supply
S1	19	**5**	30	0	50	0	12	**2**	7
S2	70	0	30	**3**	40	**7**	60	0	10
S3	40	0	10	**5**	60	0	20	**13**	18
Demand	5		8		7		15		

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman C.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan tugas akhir ini meliputi perangkat keras dan perangkat lunak yang memiliki spesifikasi seperti yang ditunjukkan oleh Tabel 4. 1.

Tabel 4. 1 Spesifikasi Perangkat

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i7-4720 CPU @ 2.60GHz (8 CPUs), ~2.6GHz • Memori: 4096MB
Perangkat Lunak	• Sistem Operasi Microsoft Windows 10 Pro 64-bit • Perangkat Pengembang Dev C++ 5.11 • Perangkat Pembantu Notepad, Microsoft Office 365, Snipping Tools

4.2 Implementasi Algoritma TOCM-PV

Implementasi proses dilakukan berdasarkan perancangan proses yang dijelaskan pada bab analisis dan perancangan. Algoritma yang akan diimplementasikan adalah algoritma TOCM-PV. Algoritma TOCM-PV akan diimplementasikan dalam bentuk program berbahasa C dengan menggunakan *tools* Dev-C++ 5.11.

4.2.1. Implementasi Tahap Mengubah Matriks Orisinal

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan mengubah matriks orisinal menjadi matriks matriks TOCM dengan cara menjumlahkan antara matriks *row opportunity cost* dan matriks *coloum opportunity cost*. Implementasi pada Kode Sumber 4. 1 menunjukkan matriks *row opportunity cost* didapat dengan cara mengurangi setiap baris dari orisinal matriks dengan biaya terkecil pada baris tersebut. Sedangkan matriks *column opportunity cost* didapat dengan cara mengurangi setiap kolom dari orisinal matriks dengan biaya terkecil pada kolom.

```
1.  printf("\n\n(Tabel Awal)");
2.  printf("\n----------------------------------------
    ---\n");
3.  for(p=0;p<jumlah_supply;p++)
4.  {
5.      for(q=0;q<jumlah_demand;q++)
6.      {
7.          if(allocation[p][q]==-1)
8.              printf("%d\t",cost[p][q]);
9.          else printf("%d\t",allocation[p][q]);
10.     }
11.     printf("| %d\n",supply[p]);
12. }
13.
14.
15. // mencari cost terkecil tiap baris
16. int h_min[jumlah_supply];
17. for(i=0;i<jumlah_supply;i++)
18. {
19.     h_min[i]=99999;
20.     for(j=0;j<jumlah_demand;j++)
21.     {
22.         if(cost[i][j]<h_min[i]) h_min[i]=cost[i][j]
    ;
23.     }
24. }
25.
26. // mencari cost terkecil tiap kolom
27. int v_min[jumlah_demand];
```

```
for(j=0;j<jumlah_demand;j++)
{
    v_min[j]=99999;
    for(i=0;i<jumlah_supply;i++)
    {
        if(cost[i][j]<v_min[j]) v_min[j]=cost[i][j]
;
    }
}


printf("\n\n(Tabel Pengurangan Baris)");
printf("\n-----------------------------------------
---\n");
// menentukan h_cost
for(i=0;i<jumlah_supply;i++)
{
    for(j=0;j<jumlah_demand;j++)
    {
        h_cost[i][j]=cost[i][j]-h_min[i];
        printf("%d\t",h_cost[i][j]);
    }
    printf("| %d\n",supply[i]);
}


printf("\n\n(Tabel Pengurangan Kolom)");
printf("\n-----------------------------------------
---\n");
// menentukan v_cost
for(i=0;i<jumlah_supply;i++)
{
    for(j=0;j<jumlah_demand;j++)
    {
        v_cost[i][j]=cost[i][j]-v_min[j];
        printf("%d\t",v_cost[i][j]);
    }
    printf("| %d\n",supply[i]);
}


// menghitung matrix TOCM
for(i=0;i<jumlah_supply;i++)
```

```
{
    for(j=0;j<jumlah_demand;j++)
    {
        tocm_cost[i][j]=v_cost[i][j]+h_cost[i][j];

    }
}

// print matrix TOCM
printf("\n\n\n");
printf("Matrix Akhir TOCM");
printf("\n----------------------------------------
---\n");
for(i=0;i<jumlah_supply;i++)
{
    for(j=0;j<jumlah_demand;j++)
    {
        printf("%d\t",tocm_cost[i][j]);
    }
    printf("\n");
}
```

Kode Sumber 4. 1 Implementasi Tahap Mengubah Matriks Orisinal

4.2.2. Implementasi Tahap Menghitung *Priority Value*

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan menghitung *Row Priority Value* (RPV) dan *Column Priority Value* (CPV). Implementasi pada Kode Sumber 4. 2 menunjukkan pengimplementasian dari algoritma TOCM-PV pada tahap menghitung *Priority Value.*

```
for(j=0;j<jumlah_demand;j++)
{
    int sum=0;
    for(i=0;i<jumlah_supply;i++)
```

```
5.          {if(allocation[i][j]==-1)
    sum+=cost[i][j]}
6.      for(i=0;i<jumlah_supply;i++)
7.          {if(allocation[i][j]==-1)
8.              {v_cost[i][j]=sum-
    cost[i][j]*(jumlah_supply - supply_satisfy)}}
9.  }
10.
11. for(i=0;i<jumlah_supply;i++)
12. {
13.     int sum=0;
14.     for(j=0;j<jumlah_demand;j++)
15.         {if(allocation[i][j]==-
    1)  sum+=cost[i][j];}
16.     for(j=0;j<jumlah_demand;j++)
17.         {if(allocation[i][j]==-1)
18.             h_cost[i][j]=sum-
    cost[i][j]*(jumlah_demand - demand_satisfy)}}
19. }
```

Kode Sumber 4. 2 Implementasi Tahap Menghitung *Priority Value*

4.2.3. Implementasi Tahap Memilih *Priority Value*

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan memilih *Priority Value* terbesar. Implementasi pada Kode Sumber 4. 3 menunjukkan implementasi dari algoritma TOCM-PV pada tahap memilih *Priority Value* terbesar.

```
1.  int max_penalty=-100000;
2.  int max_penalty_i, max_penalty_j=-1;
3.  for(i=0;i<jumlah_supply;i++)
4.  {
5.      for(j=0;j<jumlah_demand;j++)
6.      {
7.          if(v_cost[i][j]>max_penalty && allocation
    [i][j]==-1)
8.          {
```

```
9.              max_penalty=v_cost[i][j];
10.             max_penalty_i=i;
11.             max_penalty_j=j;
12.         }
13.
14.         if(h_cost[i][j]>max_penalty && allocation
    [i][j]==-1)
15.         {
16.             max_penalty=h_cost[i][j];
17.             max_penalty_i=i;
18.             max_penalty_j=j;
19.         }
20.     }
21. }
```

Kode Sumber 4. 3 Implementasi Tahap Mencari *Priority Value* Terbesar

4.2.4. Implementasi Tahap Alokasi *Supply Demand* Habis

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan alokasi. Implementasi pada Kode Sumber 4. 4. menunjukkan implementasi dari algoritma TOCM-PV pada tahap pengalokasian pada kondisi *supply* dan *demand* habis.

```
1.  if(supply[max_penalty_i]==demand[max_penalty_j])
2.  {
3.      allocation[max_penalty_i][max_penalty_j]=
    demand[max_penalty_j];
4.      supply[max_penalty_i]=0;
5.      demand[max_penalty_j]= 0;
6.      for(i=0;i<jumlah_supply;i++)
7.          if(allocation[i][max_penalty_j]==-1)
8.              allocation[i][max_penalty_j]=0;
9.      for(j=0;j<jumlah_demand;j++)
10.         if(allocation[max_penalty_i][j]==-1)
11.             allocation[max_penalty_i][j]=0;
12.     supply_satisfy++;
13.     demand_satisfy++;
```

```
14. }
```

Kode Sumber 4. 4 Implementasi Tahap Alokasi *Supply Demand* Habis

4.2.5. Implementasi Tahap Alokasi *Supply* Habis

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan alokasi. Implementasi pada Kode Sumber 4. 5 menunjukkan pengimplementasian dari algoritma TOCM-PV pada tahap pengalokasian dalam kondisi *supply* habis.

```
1.  if(supply[max_penalty_i]<demand[max_penalty_j])
2.  {
3.      allocation[max_penalty_i][max_penalty_j]= sup
    ply[max_penalty_i];
4.      demand[max_penalty_j]-=
    supply[max_penalty_i];
5.      supply[max_penalty_i]= 0;
6.      for(j=0;j<jumlah_demand;j++)
7.          if(allocation[max_penalty_i][j]==-1)
8.              allocation[max_penalty_i][j]=0;
9.      supply_satisfy++;
10.
11.     while(demand[max_penalty_j]>0)
12.     {
13.         int next_penalty=-100000;
14.         int next_penalty_i=-1;
15.         for(i=0;i<jumlah_supply;i++)
16.         {
17.         if(v_cost[i][max_penalty_j]>next_penalty
    && allocation[i][max_penalty_j]==-1)
18.           next_penalty= v_cost[i][max_penalty_j];
19.           next_penalty_i=i;
20.
21.         if(h_cost[i][max_penalty_j]>next_penalty
    && allocation[i][max_penalty_j]==-1)
22.           next_penalty= h_cost[i][max_penalty_j];
23.           next_penalty_i=i;
```

```
            }
        }

 if(supply[next_penalty_i]>demand[max_penalty_j]
   {
       allocation[next_penalty_i][max_penalty_j]=
    demand[max_penalty_j];
       supply[next_penalty_i]-
   = demand[max_penalty_j];
       demand[max_penalty_j]= 0;
       for(i=0;i<jumlah_supply;i++)
          if(allocation[i][max_penalty_j]==-1)
                allocation[i][max_penalty_j]=0;
                demand_satisfy++;

          else if(supply[next_penalty_i]<
   demand[max_penalty_j])
        allocation[next_penalty_i][max_penalty_j]
   = supply[next_penalty_i];
        demand[max_penalty_j]-
   = supply[next_penalty_i];
        supply[next_penalty_i]= 0;
              for(j=0;j<jumlah_demand;j++)
        if(allocation[next_penalty_i][j]==-1)
             allocation[next_penalty_i][j] =0;
                  supply_satisfy++;

        else if(supply[next_penalty_i]==
   demand[max_penalty_j])
        {
        allocation[next_penalty_i][max_penalty_j]
   = supply[next_penalty_i];
            demand[max_penalty_j]=0;
            supply[next_penalty_i]= 0;
            for(i=0;i<jumlah_supply;i++)
          if(allocation[i][max_penalty_j]==-1)
                  allocation[i][max_penalty_j]=0;
```

```
54.             for(j=0;j<jumlah_demand;j++)
55.          if(allocation[next_penalty_i][j]==-1)
56.                  allocation[next_penalty_i][j]=0;

57.             supply_satisfy++;
58.             demand_satisfy++;
59.         }
60.     }
61. }
```

Kode Sumber 4. 5 Implementasi Tahap Alokasi Supply Habis

4.2.6. Implementasi Tahap Alokasi *Demand* Habis

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan alokasi. Implementasi pada Kode Sumber 4. 6 menunjukkan pengimplementasian dari algoritma TOCM-PV pada tahap pengalokasian dalam kondisi *demand* habis.

```
1.  if(supply[max_penalty_i]>demand[max_penalty_j])
2.  {
3.      allocation[max_penalty_i][max_penalty_j]=
    demand[max_penalty_j];
4.      supply[max_penalty_i]-=
    demand[max_penalty_j];
5.      demand[max_penalty_j]= 0;
6.      for(i=0;i<jumlah_supply;i++)
7.         if(allocation[i][max_penalty_j]==1)
8.              allocation[i][max_penalty_j]=0;
9.      demand_satisfy++;
10.
11.     while(supply[max_penalty_i]>0)
12.     {
13.         int next_penalty=-100000;
14.         int next_penalty_j=-1;
15.         for(j=0;j<jumlah_demand;j++)
16.         {
17.             if(v_cost[max_penalty_i][j]>
    next_penalty&&allocation[max_penalty_i][j]==-1)
```

```
18.            next_penalty=v_cost[max_penalty_i][j];
19.            next_penalty_j=j;
20.
21.             if(h_cost[max_penalty_i][j]>
   next_penalty&&allocation[max_penalty_i][j]==-1)
22.            next_penalty=h_cost[max_penalty_i][j];
23.            next_penalty_j=j;
24.         }
25.
26.         if(supply[max_penalty_i]>
   demand[next_penalty_j])
27.         {
28.         allocation[max_penalty_i][next_penalty_j]
   =demand[next_penalty_j];
29.             supply[max_penalty_i]-
   =demand[next_penalty_j];
30.             demand[next_penalty_j]= 0;
31.          for(i=0;i<jumlah_supply;i++)
32.          if(allocation[i][next_penalty_j]==-1)
33.          allocation[i][next_penalty_j]=0;
34.          demand_satisfy++;
35.         }
36.         else if(supply[max_penalty_i]<
   demand[next_penalty_j])
37.         {
38.         allocation[max_penalty_i][next_penalty_j]
   =supply[max_penalty_i];
39.             demand[next_penalty_j]-
   =supply[max_penalty_i];
40.           supply[max_penalty_i]= 0;
41.           for(j=0;j<jumlah_demand;j++)
42.           if(allocation[max_penalty_i][j]==-1)
43.              allocation[max_penalty_i][j]=0;
44.           supply_satisfy++;
45.         }
46.      else if(supply[max_penalty_i]==demand
   [next_penalty_j])
47.         {
```

```
48.         allocation[max_penalty_i][next_penalty_j]
   =supply[max_penalty_i];
49.         demand[next_penalty_j]=0;
50.         supply[max_penalty_i]= 0;
51.         for(i=0;i<jumlah_supply;i++)
52.         if(allocation[i][next_penalty_j]==-1)
53.           allocation[i][next_penalty_j]=0;
54.         for(j=0;j<jumlah_demand;j++)
55.           if(allocation[max_penalty_i][j]==-1)
56.           allocation[max_penalty_i][j]=0;
57.         supply_satisfy++;
58.         demand_satisfy++;
59.         }
60.     }
61. }
```

Kode Sumber 4. 6 Implementasi Tahap Alokasi *Demand* Habis

4.2.7. Implementasi Tahap Menghitung Total Cost

Bagian ini membahas implementasi algoritma TOCM-PV pada tahapan akhir yaitu menghitung *total cost. Total cost* sendiri dengan cara menggabungkan antara hasil penalti TOCM-PV dan orisinal TP. Implementasi pada Kode Sumber 4. 7 menunjukkan pengimplementasian dari algoritma TOCM-PV pada tahap menghitung *total cost.*

```
1.  int result=0;
2.  for(i=0;i<jumlah_supply;i++)
3.  {
4.      for(j=0;j<jumlah_demand;j++)
5.          result+=(allocation[i][j]*cost[i][j]);
6.  }
```

Kode Sumber 4. 7 Implementasi Tahap Menghitung *Total Cost*

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas uji coba dan evaluasi terhadap algoritma TOCM-PV yang telah dikembangkan.

5.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada tugas akhir ini adalah sebagai berikut:

Prosesor	: Intel® Core™ i7-4720 CPU @ 2.60GHz (8 CPUs), ~2.6GHz
RAM	: 4096 MB
Jenis *Device*	: Laptop
Sistem Operasi	: Microsoft Windows 10 Pro 64-bit

5.2 Data Uji Coba

Data yang digunakan untuk uji coba algoritma TOCM-PV adalah 85 *Problem Set Transportation Problem* yang terdiri dari :

1. Data sintesis sebanyak 15 set data. Detail dari *cost*, *supply*, dan *demand* dapat dilihat pada Lampiran A.
2. Data dari beberapa jurnal sebanyak 34 set data dapat dilihat pada Tabel 5. 1.
3. Data real dari perusahaan XYZ sebanyak 36 set data. Perusahaan XYZ terletak di pulau jawa dan mendistribusikan produknya ke kota-kota yang ada di pulau Jawa. Perusahaan XYZ terdiri dari 2 pabrik yang berfungsi sebagai *supply* dan 94 kota yang berfungsi sebagai *demand*. Data biaya adalah data *cycle time* (waktu yang diperlukan oleh truk berangkat dari pabrik ke kota tujuan dan kembali ke pabrik lagi dalam satuan jam). Detail dari *supply* dan *demand* untuk 36 set data dapat dilihat pada Lampiran B. Data biaya untuk semua data adalah sama yaitu :

- *Cycle time* dari pabrik pertama ke 94 kota adalah 22 38 32 17 31 15 23 25 25 20 33 30 18 12 23 18 26 20 15 18 15 25 28 23 16 18 21 28 20 20 25 18 18 18 25 27 21 25 23 26 33 38 29 17

38 17 39 39 22 22 35 25 28 31 33 20 36 31 23 27 25 35 30 31 28 27 28 30 27 54 43 52 56 46 59 53 40 58 44 51 40 43 45 49 48 57 46 46 60 61 66 59 55 56.

- *Cycle time* dari pabrik kedua ke 94 kota adalah 26 41 36 21 35 20 27 30 28 25 38 34 23 15 20 15 23 18 14 17 13 23 25 25 17 23 23 33 18 19 28 23 20 18 25 31 20 25 25 23 39 36 28 12 33 14 38 38 20 28 33 28 26 28 31 17 38 30 30 25 22 33 26 29 31 34 28 33 31 49 38 49 52 43 56 49 36 54 40 46 36 39 41 46 44 54 41 43 57 57 62 55 52 51.

Tabel 5. 1 Tiga Puluh Empat Data Dari Beberapa Referensi.

Data 1 : [22] ➔ nilai optimal ➔ 880 $[C_{ij}]_{3x4}$ = [3 6 3 4; 6 5 11 15; 1 3 10 5] $[S_i]_{3x1}$ = [80, 90, 55] $[D_j]_{1x4}$ = [70, 60, 35, 60]	Data 2 : [23] ➔ 1.650 $[C_{ij}]_{3x3}$ = [6 10 14; 12 19 21; 15 14 17] $[S_i]_{3x1}$ = [50, 50, 50] $[D_j]_{1x3}$ = [30, 40, 55]
Data 3 : [17] ➔ 743 $[C_{ij}]_{3x4}$ = [19 30 50 10; 70 30 40 60; 40 8 70 20] $[S_i]_{3x1}$ = [7, 9, 18] $[D_j]_{1x4}$ = [5, 8, 7, 14]	Data 4 : [24] ➔ 5.600 $[C_{ij}]_{3x3}$ = [32 40 120; 60 68 104; 200 80 60] $[S_i]_{3x1}$ = [20, 30, 45] $[D_j]_{1x3}$ = [30, 35, 30]
Data 5 : [25] ➔ 840 $[C_{ij}]_{4x3}$ = [3 4 6; 7 3 8; 6 4 5; 7 5 2] $[S_i]_{4x1}$ = [100, 80, 90, 120] $[D_j]_{1x3}$ = [110, 110, 60]	Data 6 : [26] ➔ 59 [Cij]3x4 = [3 6 1 5; 7 9 2 7; 2 4 2 1] [Si]3x1 = [6, 6, 6] [Dj]1x4 = [4, 5, 4, 5]
Data 7 : [27] ➔ 28 [Cij]3x4 = [1 2 3 4; 4 3 2 0; 0 2 2 1] [Si]3x1 = [6, 8, 10] [Dj]1x4 = [4, 6, 8, 6]	Data 8 : [28] ➔ 435 [Cij]3x4 = [10 2 20 11; 12 7 9 20; 4 14 16 18] [Si]3x1 = [15, 25, 10] [Dj]1x4 = [5, 15, 15, 15]
Data 9 : [29] ➔ 390 [Cij]4x5 = [2 1 3 2 2; 3 2 1 1 1; 5 4 2 1 3; 7 5 5 3 1] [Si]4x1 = [20, 70, 30, 60]	Data 10 : [18] ➔ 1.580 [Cij]3x5 = [6 4 4 7 5; 5 6 7 4 8; 3 4 6 3 4] [Si]3x1 = [100, 125, 175] [Dj]1x5 = [60, 80, 85, 105, 70]

[Dj]1x5 = [50, 30, 30, 50, 20]	
Data 11 : [30] ➔ 49 [Cij]4x4 = [4 3 0 5; 9 7 3 2; 1 5 4 3; 6 8 24 16] [Si]4x1 = [8, 2, 9, 2] [Dj]1x4 = [5, 7, 6, 3]	Data 12 : [15] ➔ 410 [Cij]4x4 = [7 5 9 11; 4 3 8 6; 3 8 10 5; 2 6 7 3] [Si]4x1 = [30, 25, 20, 15] [Dj]1x4 = [30, 30, 20, 10]
Data 13 : [21] ➔ 2.850 [Cij]3x4 = [3 1 7 4; 2 6 5 9; 8 3 3 2] [Si]3x1 = [300, 400, 500] [Dj]1x4 = [250, 350, 400, 200]	Data 14 : [21] ➔ 183 [Cij]3x5 = [5 7 10 5 3; 8 6 9 12 14; 10 9 8 10 15] [Si]3x1 = [5, 10, 10] [Dj]1x5 = [3, 3, 10, 5, 4]
Data 15 : [5]➔ 799 [Cij]3x4 = [19 30 50 12; 70 30 40 60; 40 10 60 20] [Si]3x1 = [7, 10, 18] [Dj]1x4 = [5, 8, 7, 15]	Data 16 : [5]➔ 273 [Cij]3x5 = [4 1 2 4 4; 2 3 2 2 3; 3 5 2 4 4] [Si]3x1 = [60, 35, 40] [Dj]1x5 = [22, 45, 20, 18, 30]
Data 17 : [4] ➔ 1.160 $[C_{ij}]_{3x4}$ = [6 1 9 3; 11 5 2 8; 10 12 4 7] $[S_i]_{3x1}$ = [70, 55, 90] $[D_j]_{1x4}$ = [85, 35, 50, 45]	Data 18 : [11] ➔ 200 $[C_{ij}]_{3x4}$ = [3 6 8 4; 6 1 2 5; 7 8 3 9] $[S_i]_{3x1}$ = [20, 28, 17] $[D_j]_{1x4}$ = [15, 19, 13, 18]
Data 19 : [31] ➔ 240 [Cij]3x4 = [9 8 5 7; 4 6 8 7; 5 8 9 5] [Si]3x1 = [12, 14, 16] [Dj]1x4 = [8, 18, 13, 3]	Data 20 : [14] ➔ 820 [Cij]3x3 = [50 30 220; 90 45 170; 250 200 50] [Si]3x1 = [1, 3, 4] [Dj]1x3 = [4, 2, 2]
Data 21 : [32] ➔ 190 [Cij]3x4 = [5 7 9 6; 6 7 10 5; 7 6 8 1] [Si]3x1 = [12, 14, 10] [Dj]1x4 = [10, 6, 8, 12]	Data 22 : [32] ➔ 83 [Cij]4x4 = [2 5 6 3; 9 6 2 1; 5 2 3 6; 7 7 2 4] [Si]4x1 = [6, 9, 7, 12] [Dj]1x4 = [10, 4, 6, 14]
Data 23 : [6] ➔ 3.460 [Cij]3x4 = [20 22 17 4; 24 37 9 7; 32 37 20 15]	Data 24 : [2] ➔ 910 [Cij]3x4 = [20 2 20 11; 24 7 9 20; 8 14 16 18] [Si]3x1 = [30, 50, 20]

[Si]3x1 = [120, 70, 50] [Dj]1x4 = [60, 40, 30, 110]	[Dj]1x4 = [10, 30, 30, 30]
Data 25 : [2] ➔ 1.670 [Cij]4x4 = [7 5 27 22; 4 3 24 12; 6 16 60 20; 2 6 21 6] [Si]4x1 = [60, 50, 40, 30] [Dj]1x4 = [60, 60, 40, 20]	Data 26 : [2] ➔ 2.280 [Cij]4x4 = [21 5 9 11; 12 3 86;98105;6673] [Si]4x1 = [120, 100, 80, 60] [Dj]1x4 = [120, 120, 80, 40]
Data 27 : [2] ➔ 2.460 [Cij]3x4 = [10 2 20 22; 12 7 9 40; 4 14 16 32] [Si]3x1 = [60, 100, 40] [Dj]1x4 = [20, 60, 60, 60]	Data 28 : [2] ➔ 291 [Cij]3x3 = [7 8 7; 18 8 12; 8 12 12] [Si]3x1 = [13, 14, 8] [Dj]1x3 = [14, 8, 13]
Data 29 : [1] ➔ 4.525 [Cij]3×3 = [6 8 10; 7 11 11; 4 5 12] [Si]3×1 = [150, 175, 275] [Dj]1×3 = [200, 100, 300]	Data 30 : [1] ➔ 920 [Cj]3×4 = [4 6 8 8; 6 8 6 7; 5 7 6 8] [Si]3×1 = [40, 60, 50] [Dj]1×4 = [20, 30, 50, 50]
Data 31 : [1] ➔ 809 [Cij]3×4 = [19 30 50 12; 70 30 40 60; 40 10 60 20] [Si]3×1 = [7, 10, 18] [Dj]1×4 = [5, 7, 8, 15]	Data 32 : [1] ➔ 417 [Cij]3×4 = [13 18 30 8; 55 20 25 40; 30 6 50 10] [Si]3×1 = [8, 10, 11] [Dj]1×4 = [4, 6, 7, 12]
Data 33 : [1] ➔ 3.458 [Cij]4×5 = [25 14 34 46 45; 10 47 14 20 41; 22 42 38 21 46; 36 20 41 38 44] [Si]4×1 = [27, 35, 37, 45] [Dj]1×5 = [22, 27, 28, 33, 34]	Data 34 : [1] ➔ 109 [Cij]4×6 = [9 12 9 6 9 10; 7 3 7 7 5 5; 6 5 9 11 3 11; 6 8 11 2 2 10] [Si]4×1 = [2, 5, 6, 9] [Dj]1×6 = [2, 2, 4, 4, 4, 6]

5.3 Hasil Uji Coba TOCM-PV Terhadap 15 Data Sintesis

Data yang digunakan untuk uji coba algoritma TOCM-PV pada *sub-bab* ini adalah 15 data sintesis. Pada pengujian ini, metode TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV diimplemetasikan menggunakan program bahasa C++,

sedangkan metode VAM dan solusi optimal didapatkan dari aplikasi TORA.

Tabel 5. 2 menujukkan *improvement percentage* (persentase peningkatan) metode TOCM-PV terhadap VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE. Di dalam tabel tersebut terdapat 3 jenis angka, yaitu angka positif, angka nol dan angka negatif. Angka positif menunjukkan adanya peningkatan persentase TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, yang berarti bahwa total biaya TOCM-PV lebih rendah dari total biaya metode lainnya. Angka nol berarti bahwa total biaya TOCM-PV sama dengan metode lain yang diuji, dengan kata lain total biaya TOCM-PV sama dengan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, sehingga tidak ada persentase peningkatan. Sedangkan angka negatif menunjukkan persentase penurunan TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, karena total biaya TOCM-PV lebih tinggi dibandingkan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE.

Jumlah perbandingan data *improvement percentage* TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE dapat dilihat pada Tabel 5. 2 dan Tabel 5. 4. Dari 15 data, terdapat 1 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari VAM, tidak ada total biaya yang sama dari hasil metode TOCM-PV dengan VAM, dan 14 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari VAM. Jika dibandingkan dengan TDM1, terdapat 12 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari TDM1, 1 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TDM1 dan 2 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TDM1. Untuk metode TOCM-MT, total biaya yang dihasilkan oleh TOCM-PV lebih rendah 9 data dari TOCM-MT, 2 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TOCM-MT dan 4 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TOCM-

MT. Jika dibandingkan dengan JHM, terdapat 1 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari JHM, tidak ada data dimana total biaya yang dihasilkan oleh TOCM-PV yang sama dengan JHM, dan 14 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari JHM. Selanjutnya dari 15 data, total biaya yang dihasilkan oleh TOCM-PV selalu lebih besar dari metode BCE. Sedangkan untuk metode IAM dan IAM-PV hasilnya sama, dimana hasil total biaya dari 15 data yang diuji, 14 data dari metode IAM dan IAM-PV memiliki total biaya yang lebih besar dari TOCM-PV, tidak ada total biaya yang sama dengan TOCM-PV dan 1 yang dihasilkan oleh TOCM-PV lebih besar dari IAM dan IAM-PV.

Tabel 5. 3 menujukkan *deviation percentage* (persen penyimpangan) dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal. Terdapat 2 nilai angka, yaitu angka positif dan angka nol. Angka nol berarti tidak ada penyimpangan dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal atau berarti bahwa metode tersebut total biayanya sama dengan solusi optimal. Sedangkan angka positif berarti hasil total biaya dari metode metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV lebih besar dari solusi optimal, dengan kata lain ada penyimpangan terhadap solusi optimal. Persentase deviasi dihitung dengan menggunakan persamaan (2.28). Pada Tabel 5. 3 dapat dillihat bahwa hanya 3 metode antara lain VAM, JHM dan BCE yang mencapai nilai optimal sebanyak 12 dari 15 data, sedangkan metode lain tidak ada yang hasilnya mencapai optimal.

Jumlah data *deviation percentage* dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal terdapat pada. VAM, JHM dan BCE mencapai nilai optimal yang sama yaitu 12 dari 15 data, sehinnga ketiga metode ini mencapai nilai akurasi sebesar 80%. Sedangkan metode lain yaitu TDM1, IAM, IAM-PV, JHM, TOCM-MT dan

TOCM-PV, tidak ada yang mencapai nilai optimal dan memperoleh nilai akurasi 0%.

Hasil uji coba pada penelitian ini menunjukkan bahwa dari 15 data sintesis, metode VAM mencapai nilai optimal paling besar yaitu 12 dari 15 data, dengan akurasi 80%. Sedangkan metode TOCM-PV, IAM dan IAM-PV tidak mencapai hasil yang maksimal.

Pada 15 data sintesis, persentase kinerja (improvement percentage) TOCM-PV terhadap VAM sebesar 6,67%, karena 1 dari 15 contoh kasus mempunyai total biaya TOCM-PV lebih rendah dari pada VAM, diikuti oleh metode IAM sebesar 93,3%, dan IAM-PV sebesar 93,3%.

Tabel 5. 2 *Improvement Percentage* Terhadap 15 Data Sintesis

Nama	*Inisial Basic Feasible Solution* (IBFS)								Optimal	*Improvement Percentage (%)*						
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV
S01	214394	217000	214828	214394	214394	218289	214974	214449	214394	-0,03	1,18	0,18	-0,03	-0,03	1,76	0,24
S02	216570	218786	217051	216570	216570	220744	217230	216671	216570	-0,05	0,97	0,18	-0,05	-0,05	1,85	0,26
S03	409076	415913	415180	409076	409076	443506	418008	412834	409076	-0,92	0,74	0,57	-0,92	-0,92	6,92	1,24
S04	403140	411304	408778	403140	403140	435026	409150	406941	403140	-0,94	1,06	0,45	-0,94	-0,94	6,46	0,54
S05	1649732	1731106	1679307	1649732	1649732	1791152	1825143	1797381	1649732	-8,95	-3,83	-7,03	-8,95	-8,95	-0,35	1,52
S06	1767639	1966498	1807828	1767639	1767639	1978354	1978354	1966498	1767639	-11,25	0	-8,78	-11,25	-11,25	0,6	0,6
S07	1917304	2005105	1971126	1917304	1917304	2094790	1951496	1936506	1917304	-1	3,42	1,76	-1	-1	7,56	0,77
S08	1750798	1820359	1819816	1750798	1750798	1995849	1831027	1819816	1750798	-3,94	0,03	0	-3,94	-3,94	8,82	0,61
S09	1665063	1783944	1767153	1665063	1665063	1783944	1783944	1767153	1665063	-6,13	0,94	0	-6,13	-6,13	0,94	0,94
S10	1374650	1458304	1403690	1374650	1374650	1490976	1396112	1397278	1374650	-1,65	4,18	0,46	-1,65	-1,65	6,28	-0,08
S11	1509230	1596141	1530430	1509230	1509230	1697345	1544710	1515751	1509230	-0,43	5,04	0,96	-0,43	-0,43	10,7	1,87
S12	1738720	1813469	1772090	1738720	1738720	1893231	1891055	1859734	1738720	-6,96	-2,55	-4,95	-6,96	-6,96	1,77	1,66
S13	522493	531777	518615	515995	499117	997993	635470	523724	457716	-0,24	1,51	-0,99	-1,5	-4,93	47,52	17,58
S14	800103	905189	820059	708066	708066	993123	884184	786174	636525	1,74	13,15	4,13	-11,03	-11,03	20,84	11,08
S15	526605	551187	535818	540756	506474	728922	640257	532173	466173	-1,06	3,45	0,68	1,59	-5,07	26,99	16,88

Tabel 5. 3 *Deviation Percentage* Terhadap 15 Data Sintesis

Nama	*Inisial Basic Feasible Solution* (IBFS)								Optimal	*Deviation Percentage* (%)							
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV
S01	214394	217000	214828	214394	214394	218289	214974	214449	214394	0	1,22	0,2	0	0	1,82	0,27	0,03
S02	216570	218786	217051	216570	216570	220744	217230	216671	216570	0	0,97	0,18	-0,05	-0,05	1,85	0,26	0,05
S03	409076	415913	415180	409076	409076	443506	418008	412834	409076	0	0,74	0,57	-0,92	-0,92	6,92	1,24	0,92
S04	403140	411304	408778	403140	403140	435026	409150	406941	403140	0	1,06	0,45	-0,94	-0,94	6,46	0,54	0,94
S05	1649732	1731106	1679307	1649732	1649732	1791152	1825143	1797381	1649732	0	-3,83	-7,03	-8,95	-8,95	-0,35	1,52	8,95
S06	1767639	1966498	1807828	1767639	1767639	1978354	1978354	1966498	1767639	0	0	-8,78	-11,3	-11,3	0,6	0,6	11,25
S07	1917304	2005105	1971126	1917304	1917304	2094790	1951496	1936506	1917304	0	3,42	1,76	-1	-1	7,56	0,77	1
S08	1750798	1820359	1819816	1750798	1750798	1995849	1831027	1819816	1750798	0	0,03	0	-3,94	-3,94	8,82	0,61	3,94
S09	1665063	1783944	1767153	1665063	1665063	1783944	1783944	1767153	1665063	0	0,94	0	-6,13	-6,13	0,94	0,94	6,13
S10	1374650	1458304	1403690	1374650	1374650	1490976	1396112	1397278	1374650	0	4,18	0,46	-1,65	-1,65	6,28	-0,08	1,65
S11	1509230	1596141	1530430	1509230	1509230	1697345	1544710	1515751	1509230	0	5,04	0,96	-0,43	-0,43	10,7	1,87	0,43
S12	1738720	1813469	1772090	1738720	1738720	1893231	1891055	1859734	1738720	0	-2,55	-4,95	-6,96	-6,96	1,77	1,66	6,96
S13	522493	531777	518615	515995	499117	997993	635470	523724	457716	14,2	1,51	-0,99	-1,5	-4,93	47,52	17,58	14,42
S14	800103	905189	820059	708066	708066	993123	884184	786174	636525	25,7	13,15	4,13	-11	-11	20,84	11,08	23,51
S15	526605	551187	535818	540756	506474	728922	640257	532173	466173	13	3,45	0,68	1,59	-5,07	26,99	16,88	14,16

Tabel 5. 4 Jumlah Data *Improvement Percentage* Untuk 15 Data Sintesis

Metode	Angka Positif	Angka Nol	Angka Negatif
VAM	1	0	14
TDM1	12	1	2
TOCM-MT	9	2	4
JHM	1	0	14
BCE	0	0	15
IAM	14	0	1
IAM-PV	14	0	1

Tabel 5. 5 Jumlah Data Optimal dan Akurasi Untuk 15 Data Sintesis

Metode	Optimal	Akurasi (%)
VAM	12	80
TDM1	0	0
TOCM-MT	0	0
JHM	12	80
BCE	12	80
IAM	0	0
IAM-PV	0	0
IAM-PV	0	0

5.4 Hasil Uji Coba TOCM-PV Terhadap 34 Data Dari Beberapa Jurnal

Data yang digunakan untuk uji coba algoritma TOCM-PV pada *sub-bab* ini adalah 34 data dari berbagai jurnal. Pada pengujian ini, hasil dari metode TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV diperoleh dari implemetasi

program bahasa C++, sedangkan metode VAM dan solusi optimal didapatkan dengan pengaplikasian TORA.

Tabel 5. 6 menujukkan *improvement percentage* (persentase peningkatan) metode TOCM-PV terhadap VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE. Di dalam tabel tersebut terdapat 3 jenis angka, yaitu angka positif, angka nol dan angka negatif. Angka positif menunjukkan adanya peningkatan persentase TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, yang berarti bahwa total biaya TOCM-PV lebih rendah dari total biaya metode lainnya. Angka nol berarti bahwa total biaya TOCM-PV sama dengan metode lain yang duji, dengan kata lain total biaya TOCM-PV sama dengan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, sehingga tidak ada persentase peningkatan. Sedangkan angka negatif menunjukkan persentase penurunan TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, karena total biaya TOCM-PV lebih tinggi dibandingkan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE.

Jumlah perbandingan data *improvement percentage* TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE pada Tabel 5. 6 dan Tabel 5. 8. Dari 34 data, terdapat 15 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari VAM, 9 data dimana total biaya yang sama dari hasil metode TOCM-PV dengan VAM, dan 10 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari VAM. Jika dibandingkan dengan TDM1, terdapat 9 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari TDM1, 16 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TDM1 dan 9 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TDM1. Untuk metode TOCM-MT, total biaya yang dihasilkan oleh TOCM-PV lebih rendah 2 data dari TOCM-MT, 21 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TOCM-MT dan 11 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TOCM-

MT. Jika dibandingkan dengan JHM, terdapat 3 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari JHM, 15 data dimana total biaya yang dihasilkan oleh TOCM-PV yang sama dengan JHM, dan 16 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari JHM. Selanjutnya metode BCE, total biaya yang dihasilkan oleh TOCM-PV lebih rendah 3 data dari BCE, 13 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan BCE dan 18 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari BCE. Sedangkan untuk metode IAM, terdapat 23 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari IAM, 5 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan IAM dan 6 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari IAM. Terakhir adalah metode IAM-PV, terdapat 8 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari IAM-PV, 20 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan IAM-PV dan 6 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari IAM-PV.

Tabel 5. 7 menujukkan *deviation percentage* (persen penyimpangan) dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal. Terdapat 2 nilai angka, yaitu angka positif dan angka nol. Angka nol berarti tidak ada penyimpangan dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal atau berarti bahwa metode tersebut total biayanya sama dengan solusi optimal. Sedangkan angka positif berarti hasil total biaya dari metode metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV lebih besar dari solusi optimal, dengan kata lain ada penyimpangan terhadap solusi optimal. Persentase deviasi dihitung dengan menggunakan persamaan (2.28). Pada Tabel 5. 7 dapat dillihat bahwa metode BCE mempunyai jumlah optimal terbanyak dari metode lain, yaitu 30 dari 34 data uji coba.

Jumlah data *deviation percentage* dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV

terhadap solusi optimal terdapat pada Tabel 5. 9. Terdapat 9 data yang dihasilkan oleh VAM dapat mencapai solusi optimal, dan memiliki akurasi sebesar 26,47%. Metode TDM1 dan IAM-PV mempunyai hasil yang sama yaitu 14 data yang mencapai optimal dengan akurasi 41,18%. Terdapat 24 data yang dihasilkan oleh TOCM-MT dapat mencapai solusi optimal, dan memiliki akurasi sebesar 70,59%. Terdapat 23 data yang dihasilkan oleh JHM mencapai solusi optimal dengan akurasi sebesar 67,65%. Terdapat 30 data yang dihasilkan oleh BCE dapat mencapai solusi optimal dengan akurasi sebesar 88,24%. Terdapat 6 data yang di hasilkan oleh IAM dapat mencapai solusi optimal, dan memiliki akurasi sebesar 17,65%. Terakhir TOCM-PV mempunyai 16 data yang mencapai optimal dengan akurasi 47,06%.

Hasil uji coba pada penelitian ini menunjukkan bahwa 34 data dari berbagai jurnal, metode yang memperoleh hasil optimal tertinggi adalah metode TOCM-PV yaitu 16 dari 34, dengan nilai akurasi 47,06%. Disusul oleh IAM-PV dengan akurasi 41,18%, selanjutnya metode VAM yang mencapai 9% akurasi dan terakhir adalah IAM dengan akurasi 6%.

Pada 34 data dari berbagai jurnal, persentase kinerja (*improvement percentage*) TOCM-PV terhadap VAM sebesar 44,12%, karena 15 dari 34 contoh kasus, TOCM-PV mempunyai total biaya lebih rendah dari pada VAM, diikuti oleh metode IAM sebesar 67,65%, dan IAM-PV sebesar 23,53%.

Tabel 5. 6 *Improvement Percentage* Terhadap 34 Data dari Berbagai Jurnal

Jurnal	*Inisial Basic Feasible Solution* (IBFS)								Optimal	*Improvement Percentage (%)*						
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-SC		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV
Srinivasan [43]	955	880	880	880	880	1075	880	880	880	7,85	0	0	0	0	18,14	0
Goyal [25]	1745	1650	1650	1650	1650	1695	1650	1650	1650	5,44	0	0	0	0	2,65	0
Deshmukh [19]	779	779	743	743	743	781	779	779	743	0	0	-4,85	-4,85	-4,85	0,26	0
Ramadan[44]	5600	5600	5600	5600	5600	5600	5600	5600	5600	0	0	0	0	0	0	0
Kulkarni [45]	880	980	980	840	840	1020	980	980	840	-11,36	0	0	-16,67	-16,67	3,92	0
Schrenk [46]	59	59	61	59	59	66	69	61	59	-3,39	-3,39	0	-3,39	-3,39	7,58	11,59
Samuel [41]	28	28	28	28	28	28	28	28	28	0	0	0	0	0	0	0
Imam [36]	475	475	435	460	435	520	520	520	435	-9,47	-9,47	-19,54	-13,04	-19,54	0	0
Adlakha [42]	390	400	390	390	390	570	390	390	390	0	2,5	0	0	0	31,58	0
kaur [55]	1600	1595	1580	1580	1580	1880	1595	1710	1580	-6,88	-7,21	-8,23	-8,23	-8,23	9,04	-7,21
Patel [56]	49	53	53	49	49	61	49	53	49	-8,16	0	0	-8,16	-8,16	13,11	-8,16
Ahmed [18]	470	435	435	420	410	420	415	435	410	7,45	0	0	-3,57	-6,1	-3,57	-4,82
Ahmed [18]	2850	2850	2850	2850	2850	2850	2850	2850	2850	0	0	0	0	0	0	0
Ahmed [11]	187	186	187	183	187	183	183	192	183	-2,67	-3,23	-2,67	-4,92	-2,67	-4,92	-4,92
Uddin [12]	859	859	799	799	799	868	859	799	799	6,98	6,98	0	0	0	7,95	6,98

Uddin [12]	273	273	273	273	273	273	273	275	273	-0,73	-0,73	-0,73	-0,73	-0,73	-0,73	-0,73
Das [26]	1220	1160	1160	1170	1170	1315	1160	1160	1160	4,92	0	0	0,85	0,85	11,79	0
Khan [31]	204	200	200	218	204	218	200	200	200	1,96	0	0	8,26	1,96	8,26	0
Azad [20]	248	248	240	240	240	248	249	249	240	-0,4	-0,4	-3,75	-3,75	-3,75	-0,4	0
Morade [47]	820	820	820	820	820	855	820	820	820	0	0	0	0	0	4,09	0
Jude [48]	190	190	190	190	192	194	190	190	190	0	0	0	0	1,04	2,06	0
Jude [48]	92	83	83	83	83	115	88	88	83	4,35	-6,02	-6,02	-6,02	-6,02	23,48	0
Hosseini [3]	3520	3570	3460	3460	3460	3790	3570	3460	3460	1,7	3,08	0	0	0	8,71	3,08
Amaliah [54]	990	990	910	960	910	1080	990	990	910	0	0	-8,79	-3,13	-8,79	8,33	0
Amaliah [54]	1680	1670	1670	1690	1670	3280	1690	1690	1670	-0,6	-1,2	-1,2	0	-1,2	48,48	0
Amaliah [54]	2400	2400	2400	2340	2280	2360	2400	2360	2280	1,67	1,67	1,67	-0,85	-3,51	0	1,67
Amaliah [54]	2980	2980	2500	2500	2460	3280	3020	2500	2460	16,11	16,11	0	0	-1,63	23,78	17,22
Amaliah [54]	327	291	291	327	291	319	291	291	291	11,01	0	0	11,01	0	8,78	0
Juman [1]	5125	4550	5225	4525	4525	4550	4550	4525	4525	11,71	0,55	13,4	0	0	0,55	0,55
Juman [1]	960	960	930	920	920	920	960	960	920	0	0	-3,23	-4,35	-4,35	-4,35	0
Juman [1]	859	849	809	809	809	855	849	809	809	5,82	4,71	0	0	0	5,38	4,71
Juman [1]	476	465	417	417	417	495	465	417	417	12,39	10,32	0	0	0	15,76	10,32
Juman [1]	3778	3572	3513	3487	3458	4147	3513	3513	3458	7,01	1,65	0	-0,75	-1,59	15,29	0
Juman [1]	112	117	109	112	109	112	119	125	109	-11,61	-6,84	-14,68	-11,61	-14,68	-11,61	-5,04

Tabel 5. 7 *Deviation Percentage* Terhadap 34 Data dari Berbagai Jurnal

Jurnal	*Inisial Basic Feasible Solution* (IBFS)								Opti Mal	*Deviation Percentage* (%)							
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV
Srinivasan [22]	955	880	880	880	880	1075	880	880	880	8,52	0	0	0	0	22,2	0	0
Goyal [23]	1745	1650	1650	1650	1650	1695	1650	1650	1650	5,76	0	0	0	0	2,73	0	0
Deshmukh [17]	779	779	743	743	743	781	779	779	743	4,85	4,85	0	0	0	5,11	4,85	4,85
Ramadan [24]	5600	5600	5600	5600	5600	5600	5600	5600	5600	0	0	0	0	0	0	0	0
Kulkarni [25]	880	980	980	840	840	1020	980	980	840	4,76	16,7	16,67	0	0	21,4	16,7	16,67
Schrenk [26]	59	59	61	59	59	66	69	61	59	0	0	3,39	0	0	11,9	17	3,39
Samuel [27]	28	28	28	28	28	28	28	28	28	0	0	0	0	0	0	0	0
Imam [28]	475	475	435	460	435	520	520	520	435	9,2	9,2	0	5,75	0	19,5	19,5	19,54
Adlakha [29]	390	400	390	390	390	570	390	390	390	0	2,56	0	0	0	46,2	0	0
kaur [18]	1600	1595	1580	1580	1580	1880	1595	1710	1580	1,27	0,95	0	0	0	19	0,95	8,23
Patel [30]	49	53	53	49	49	61	49	53	49	0	8,16	8,16	0	0	24,5	0	8,16
Ahmed [15]	470	435	435	420	410	420	415	435	410	14,6	6,1	6,1	2,44	0	2,44	1,22	6,1
Ahmed [21]	2850	2850	2850	2850	2850	2850	2850	2850	2850	0	0	0	0	0	0	0	0
Ahmed [21]	187	186	187	183	187	183	183	192	183	2,19	1,64	2,19	0	2,19	0	0	4,92
Uddin [5]	859	859	799	799	799	868	859	799	799	7,51	7,51	0	0	0	8,64	7,51	0

Uddin [5]	273	273	273	273	273	273	273	275	273	0	0	0	0	0	0	0	0,73
Das [4]	1220	1160	1160	1170	1170	1315	1160	1160	1160	5,17	0	0	0,86	0,86	13,4	0	0
Khan [11]	204	200	200	218	204	218	200	200	200	2	0	0	9	2	9	0	0
Azad [31]	248	248	240	240	240	248	249	249	240	3,33	3,33	0	0	0	3,33	3,75	3,75
Morade [14]	820	820	820	820	820	855	820	820	820	0	0	0	0	0	4,27	0	0
Jude [32]	190	190	190	190	192	194	190	190	190	0	0	0	0	1,05	2,11	0	0
Jude [32]	92	83	83	83	83	115	88	88	83	10,8	0	0	0	0	38,6	6,02	6,02
Hosseini [6]	3520	3570	3460	3460	3460	3790	3570	3460	3460	1,73	3,18	0	0	0	9,54	3,18	0
Amaliah [2]	990	990	910	960	910	1080	990	990	910	8,79	8,79	0	5,49	0	18,7	8,79	8,79
Amaliah [2]	1680	1670	1670	1690	1670	3280	1690	1690	1670	0,6	0	0	1,2	0	96,4	1,2	1,2
Amaliah [2]	2400	2400	2400	2340	2280	2360	2400	2360	2280	5,26	5,26	5,26	2,63	0	3,51	5,26	3,51
Amaliah [2]	2980	2980	2500	2500	2460	3280	3020	2500	2460	21,1	21,1	1,63	1,63	0	33,3	22,8	1,63
Amaliah [2]	327	291	291	327	291	319	291	291	291	12,4	0	0	12,4	0	9,62	0	0
Juman [1]	5125	4550	5225	4525	4525	4550	4550	4525	4525	13,3	0,55	15,47	0	0	0,55	0,55	0
Juman [1]	960	960	930	920	920	920	960	960	920	4,35	4,35	1,09	0	0	0	4,35	4,35
Juman [1]	859	849	809	809	809	855	849	809	809	6,18	4,94	0	0	0	5,69	4,94	0
Juman [1]	476	465	417	417	417	495	465	417	417	14,2	11,5	0	0	0	18,7	11,5	0
Juman [1]	3778	3572	3513	3487	3458	4147	3513	3513	3458	9,25	3,3	1,59	0,84	0	19,9	1,59	1,59
Juman [1]	112	117	109	112	109	112	119	125	109	2,75	7,34	0	2,75	0	2,75	9,17	14,68

Tabel 5. 8 Jumlah Data *Improvement Percentage* Untuk 34 Data Dari Berbagai Jurnal

Metode	Angka Positif	Angka Nol	Angka Negatif
VAM	15	9	10
TDM1	9	16	9
TOCM-MT	2	21	11
JHM	3	15	16
BCE	3	13	18
IAM	23	5	6
IAM-PV	8	20	6

Tabel 5. 9 Jumlah Data Optimal dan Akurasi Untuk 34 Data Dari Berbagai Jurnal

Metode	Optimal	Akurasi (%)
VAM	9	26,47
TDM1	14	41,18
TOCM-MT	24	70,59
JHM	23	67,65
BCE	30	88,24
IAM	6	17,65
IAM-PV	14	41,18
TOCM-PV	16	47,06

5.5 Hasil Uji Coba TOCM-PV Terhadap 36 Data *Real*

Data yang digunakan untuk uji coba algoritma TOCM-PV pada *sub-bab* ini adalah 36 data *real* dari perusahaan XYZ. Pada pengujian ini, hasil dari metode TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV diperoleh dari implemetasi

program bahasa C++, sedangkan metode VAM dan solusi optimal didapatkan dengan pengaplikasian TORA.

Tabel 5. 10 menujukkan *improvement percentage* (persentase peningkatan) metode TOCM-PV terhadap VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE. Di dalam tabel tersebut terdapat 3 jenis angka, yaitu angka positif, angka nol dan angka negatif. Angka positif menunjukkan adanya peningkatan persentase TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, yang berarti bahwa total biaya TOCM-PV lebih rendah dari total biaya metode lainnya. Angka nol berarti bahwa total biaya TOCM-PV sama dengan metode lain yang duji, dengan kata lain total biaya TOCM-PV sama dengan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, sehingga tidak ada persentase peningkatan. Sedangkan angka negatif menunjukkan persentase penurunan TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, karena total biaya TOCM-PV lebih tinggi dibandingkan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE.

Jumlah perbandingan data *improvement percentage* TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE pada Tabel 5. 10 dan Tabel 5. 12. Dari 36 data, metode VAM, JHM dan BCE memperoleh hasil yang sama, yaitu hanya terdapat 1 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan VAM, JHM dan BCE, serta 35 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari ketiga metode tersebut. Metode TDM1 dan TOCM-MT juga memiliki kesamaan jumlah hasil, dimana total biaya yang dihasilkan oleh TOCM-PV lebih rendah 23 data dari TDM1 dan TOCM-MT, tidak ada data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TDM1 dan TOCM-MT, serta 13 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TDM1 dan TOCM-MT. Selanjutnya metode IAM, seluruh total biaya yang dihasilkan oleh TOCM-PV lebih rendah 36 data dari IAM, sehingga tidak ada total biaya yang dihasilkan oleh TOCM-

PV sama yang ataupun lebih besar dari IAM. Terakhir adalah metode IAM-PV, terdapat 22 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari IAM-PV, 12 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan IAM-PV dan tidak data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari IAM-PV.

Tabel 5. 11 menujukkan *deviation percentage* (persen penyimpangan) dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal. Terdapat 2 nilai angka, yaitu angka positif dan angka nol. Angka nol berarti tidak ada penyimpangan dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal atau berarti bahwa metode tersebut total biayanya sama dengan solusi optimal. Sedangkan angka positif berarti hasil total biaya dari metode metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV lebih besar dari solusi optimal, dengan kata lain ada penyimpangan terhadap solusi optimal. Persentase deviasi dihitung dengan menggunakan persamaan (2.28). Pada Tabel 5. 11 dapat dillihat bahwa metode JHM dan BCE mempunyai jumlah optimal tertinggi dari metode lain.

Jumlah data *deviation percentage* dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal terdapat pada Tabel 5. 13. Terdapat 35 data yang dihasilkan oleh VAM dapat mencapai solusi optimal, dan memiliki akurasi sebesar 97,22%. Metode JHM dan BCE mempunyai hasil yang sama yaitu akurasi 100% dengan 36 data mencapai nilai optimal. Sedangkan metode lain yaitu TDM1, TOCM-MT, IAM, IAM-PV dan TOCM-PV tidak memiliki hasil yang mencapai nilai optimal, dengan kata lain mempunya akurasi 0%.

Hasil uji coba pada penelitian ini menunjukkan bahwa dari 36 data *real*, metode yang memperoleh hasil optimal tertinggi adalah metode VAM dengan jumlah optimal 35 dari 36 data,

sehingga mencapai akurasi 97,22%. Sedangkan metode IAM, IAM-PV dan TOCM-PV tidak selalu menunjukkan hasil optimal.

Pada 36 data *real*, persentase kinerja (*improvement percentage*) TOCM-PV terhadap metode IAM sebesar 100%, karena 36 contoh kasus mempunyai total biaya TOCM-PV lebih rendah dari pada IAM, sedangkan metode IAM-PV sebesar 61,11%. TOCM-PV terhadap VAM sendiri tidak selalu menunjukkan persentase kinerja yang baik.

Tabel 5. 10 *Improvement Percentage* Terhadap 36 Data *Real*

Nama	*Inisial Basic Feasible Solution* (IBFS)								Optimal	*Improvement Percentage (%)*						
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV
R01	12175097	12208071	12210362	12175097	12175097	12869231	12346537	12322042	12175097	-1,21	-0,93	-0,91	-1,21	-1,21	4,25	0,2
R02	12793609	12869920	12826647	12793609	12793609	13435114	12962364	12939800	12793609	-1,14	-0,54	-0,88	-1,14	-1,14	3,69	0,17
R03	13243021	13331185	13277784	13243021	13243021	13857828	13411490	13402909	13243021	-1,21	-0,54	-0,94	-1,21	-1,21	3,28	0,06
R04	12928429	13006080	12965785	12928429	12928429	13495738	13098507	13092722	12928429	-1,27	-0,67	-0,98	-1,27	-1,27	2,99	0,04
R05	13657053	13775787	13700907	13657053	13657053	14172575	13836003	13832598	13657053	-1,29	-0,41	-0,96	-1,29	-1,29	2,4	0,02
R06	12296587	12324912	12322885	12296587	12296587	12460774	12460774	12455418	12296587	-1,29	-1,06	-1,08	-1,29	-1,29	0,04	0,04
R07	14117586	14236337	14166940	14117586	14117586	14674414	14282395	14253252	14117586	-0,96	-0,12	-0,61	-0,96	-0,96	2,87	0,2
R08	14017419	14148941	14074887	14017419	14017419	14457185	14193481	14188878	14017419	-1,22	-0,28	-0,81	-1,22	-1,22	1,86	0,03
R09	13748896	13871425	13821858	13748896	13748896	14219125	13915646	13913035	13748896	-1,19	-0,3	-0,66	-1,19	-1,19	2,15	0,02
R10	13691814	13820314	13747189	13691814	13691814	14128948	13874252	13869649	13691814	-1,3	-0,36	-0,89	-1,3	-1,3	1,84	0,03
R11	13715927	13839206	13770792	13715927	13715927	14179234	13894606	13890003	13715927	-1,27	-0,37	-0,87	-1,27	-1,27	2,04	0,03
R12	13922982	14061909	14010673	13922982	13922982	14394586	14061061	14058450	13922982	-0,97	0,02	-0,34	-0,97	-0,97	2,34	0,02
R13	5408695	5416127	5461418	5408695	5408695	5775529	5493693	5464813	5408695	-1,04	-0,9	-0,06	-1,04	-1,04	5,38	0,53
R14	5438725	5462780	5511842	5438725	5438725	5550692	5550692	5492135	5438725	-0,98	-0,54	0,36	-0,98	-0,98	1,05	1,05
R15	5663908	5671616	5742635	5663908	5663908	5794206	5769540	5702654	5663908	-0,68	-0,55	0,7	-0,68	-0,68	1,58	1,16
R16	5793809	5802139	5883715	5793809	5793809	6165186	5890110	5831703	5793809	-0,65	-0,51	0,88	-0,65	-0,65	5,41	0,99
R17	6025531	6033695	6093834	6025531	6025531	6406917	6122804	6085634	6025531	-1	-0,86	0,13	-1	-1	5,01	0,61

R18	5365603	5400594	5432628	5363699	5363699	5474312	5474312	5426793	5363699	-1,14	-0,49	0,11	-1,18	-1,18	0,87	0,87
R19	5952349	5960405	6024821	5952349	5952349	6329907	6049536	6012071	5952349	-1	-0,87	0,21	-1	-1	5,02	0,62
R20	6128530	6139109	6197774	6128530	6128530	6503933	6219462	6194698	6128530	-1,08	-0,91	0,05	-1,08	-1,08	4,75	0,4
R21	6183085	6191745	6258172	6183085	6183085	6553937	6271193	6246364	6183085	-1,02	-0,88	0,19	-1,02	-1,02	4,69	0,4
R22	6192863	6203251	6265426	6192863	6192863	6559997	6282042	6258140	6192863	-1,05	-0,88	0,12	-1,05	-1,05	4,6	0,38
R23	6067533	6076125	6145072	6067533	6067533	6435148	6152249	6131289	6067533	-1,05	-0,91	0,22	-1,05	-1,05	4,72	0,34
R24	6235927	6247377	6309471	6235927	6235927	6611957	6328317	6302172	6235927	-1,06	-0,88	0,12	-1,06	-1,06	4,69	0,41
R25	5097639	5165922	5141792	5097639	5097639	5105978	5098012	5098012	5097639	-0,01	1,31	0,85	-0,01	-0,01	0,16	0
R26	5343427	5420814	5393193	5343427	5343427	5351913	5343800	5343800	5343427	-0,01	1,42	0,92	-0,01	-0,01	0,15	0
R27	5566579	5648171	5621027	5566579	5566579	5574307	5567193	5567193	5566579	-0,01	1,43	0,96	-0,01	-0,01	0,13	0
R28	5705771	5793028	5766648	5705771	5705771	5714360	5706144	5706144	5705771	-0,01	1,5	1,05	-0,01	-0,01	0,14	0
R29	6046057	6141013	6114502	6046057	6046057	6054287	6046555	6046555	6046057	-0,01	1,54	1,11	-0,01	-0,01	0,13	0
R30	5147474	5202849	5182501	5147474	5147474	5155068	5147723	5147723	5147474	0	1,06	0,67	0	0	0,14	0
R31	5934380	6028936	6002186	5934380	5934380	5943274	5934878	5934878	5934380	-0,01	1,56	1,12	-0,01	-0,01	0,14	0
R32	6139606	6237512	6217992	6139606	6139606	6149129	6140104	6140104	6139606	-0,01	1,56	1,25	-0,01	-0,01	0,15	0
R33	5980728	6078483	6056181	5980728	5980728	5988697	5981226	5981226	5980728	-0,01	1,6	1,24	-0,01	-0,01	0,12	0
R34	6207890	6304620	6285817	6207890	6207890	6217533	6208388	6208388	6207890	-0,01	1,53	1,23	-0,01	-0,01	0,15	0
R35	5975209	6070899	6050441	5975209	5975209	5982919	5975707	5975707	5975209	-0,01	1,57	1,24	-0,01	-0,01	0,12	0
R36	6255273	6352857	6332291	6255273	6255273	6263838	6255771	6255771	6255273	-0,01	1,53	1,21	-0,01	-0,01	0,13	0

Tabel 5. 11 *Deviation Percentage* Terhadap 36 Data *Real*

Nama	*Inisial Basic Feasible Solution* (IBFS)								Optimal	*Deviation Percentage* (%)							
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV
R01	12175097	12208071	12210362	12175097	12175097	12869231	12346537	12322042	12175097	0	0,27	0,29	0	0	5,7	1,41	1,21
R02	12793609	12869920	12826647	12793609	12793609	13435114	12962364	12939800	12793609	0	0,6	0,26	0	0	5,01	1,32	1,14
R03	13243021	13331185	13277784	13243021	13243021	13857828	13411490	13402909	13243021	0	0,67	0,26	0	0	4,64	1,27	1,21
R04	12928429	13006080	12965785	12928429	12928429	13495738	13098507	13092722	12928429	0	0,6	0,29	0	0	4,39	1,32	1,27
R05	13657053	13775787	13700907	13657053	13657053	14172575	13836003	13832598	13657053	0	0,87	0,32	0	0	3,77	1,31	1,29
R06	12296587	12324912	12322885	12296587	12296587	12460774	12460774	12455418	12296587	0	0,23	0,21	0	0	1,34	1,34	1,29
R07	14117586	14236337	14166940	14117586	14117586	14674414	14282395	14253252	14117586	0	0,84	0,35	0	0	3,94	1,17	0,96
R08	14017419	14148941	14074887	14017419	14017419	14457185	14193481	14188878	14017419	0	0,94	0,41	0	0	3,14	1,26	1,22
R09	13748896	13871425	13821858	13748896	13748896	14219125	13915646	13913035	13748896	0	0,89	0,53	0	0	3,42	1,21	1,19
R10	13691814	13820314	13747189	13691814	13691814	14128948	13874252	13869649	13691814	0	0,94	0,4	0	0	3,19	1,33	1,3
R11	13715927	13839206	13770792	13715927	13715927	14179234	13894606	13890003	13715927	0	0,9	0,4	0	0	3,38	1,3	1,27
R12	13922982	14061909	14010673	13922982	13922982	14394586	14061061	14058450	13922982	0	1	0,63	0	0	3,39	0,99	0,97
R13	5408695	5416127	5461418	5408695	5408695	5775529	5493693	5464813	5408695	0	0,14	0,97	0	0	6,78	1,57	1,04
R14	5438725	5462780	5511842	5438725	5438725	5550692	5550692	5492135	5438725	0	0,44	1,34	0	0	2,06	2,06	0,98
R15	5663908	5671616	5742635	5663908	5663908	5794206	5769540	5702654	5663908	0	0,14	1,39	0	0	2,3	1,87	0,68
R16	5793809	5802139	5883715	5793809	5793809	6165186	5890110	5831703	5793809	0	0,14	1,55	0	0	6,41	1,66	0,65
R17	6025531	6033695	6093834	6025531	6025531	6406917	6122804	6085634	6025531	0	0,14	1,13	0	0	6,33	1,61	1

R18	5365603	5400594	5432628	5363699	5363699	5474312	5474312	5426793	5363699	0,04	0,69	1,29	0	0	2,06	2,06	1,18
R19	5952349	5960405	6024821	5952349	5952349	6329907	6049536	6012071	5952349	0	0,14	1,22	0	0	6,34	1,63	1
R20	6128530	6139109	6197774	6128530	6128530	6503933	6219462	6194698	6128530	0	0,17	1,13	0	0	6,13	1,48	1,08
R21	6183085	6191745	6258172	6183085	6183085	6553937	6271193	6246364	6183085	0	0,14	1,21	0	0	6	1,42	1,02
R22	6192863	6203251	6265426	6192863	6192863	6559997	6282042	6258140	6192863	0	0,17	1,17	0	0	5,93	1,44	1,05
R23	6067533	6076125	6145072	6067533	6067533	6435148	6152249	6131289	6067533	0	0,14	1,28	0	0	6,06	1,4	1,05
R24	6235927	6247377	6309471	6235927	6235927	6611957	6328317	6302172	6235927	0	0,18	1,18	0	0	6,03	1,48	1,06
R25	5097639	5165922	5141792	5097639	5097639	5105978	5098012	5098012	5097639	0	1,34	0,87	0	0	0,16	0,01	0,01
R26	5343427	5420814	5393193	5343427	5343427	5351913	5343800	5343800	5343427	0	1,45	0,93	0	0	0,16	0,01	0,01
R27	5566579	5648171	5621027	5566579	5566579	5574307	5567193	5567193	5566579	0	1,47	0,98	0	0	0,14	0,01	0,01
R28	5705771	5793028	5766648	5705771	5705771	5714360	5706144	5706144	5705771	0	1,53	1,07	0	0	0,15	0,01	0,01
R29	6046057	6141013	6114502	6046057	6046057	6054287	6046555	6046555	6046057	0	1,57	1,13	0	0	0,14	0,01	0,01
R30	5147474	5202849	5182501	5147474	5147474	5155068	5147723	5147723	5147474	0	1,08	0,68	0	0	0,15	0	0
R31	5934380	6028936	6002186	5934380	5934380	5943274	5934878	5934878	5934380	0	1,59	1,14	0	0	0,15	0,01	0,01
R32	6139606	6237512	6217992	6139606	6139606	6149129	6140104	6140104	6139606	0	1,59	1,28	0	0	0,16	0,01	0,01
R33	5980728	6078483	6056181	5980728	5980728	5988697	5981226	5981226	5980728	0	1,63	1,26	0	0	0,13	0,01	0,01
R34	6207890	6304620	6285817	6207890	6207890	6217533	6208388	6208388	6207890	0	1,56	1,26	0	0	0,16	0,01	0,01
R35	5975209	6070899	6050441	5975209	5975209	5982919	5975707	5975707	5975209	0	1,6	1,26	0	0	0,13	0,01	0,01
R36	6255273	6352857	6332291	6255273	6255273	6263838	6255771	6255771	6255273	0	1,56	1,23	0	0	0,14	0,01	0,01

Tabel 5. 12 Jumlah Data *Improvement Percentage* Untuk 36 Data *Real*

Metode	Angka Positif	Angka Nol	Angka Negatif
VAM	0	1	35
TDM1	23	0	13
TOCM-MT	23	0	13
JHM	0	1	35
BCE	0	1	35
IAM	36	0	0
IAM-PV	22	12	0

Tabel 5. 13 Jumlah Data Optimal dan Akurasi Untuk 36 Data *Real*

Metode	Optimal	Akurasi (%)
VAM	35	97,22
TDM1	0	0
TOCM-MT	0	0
JHM	36	100
BCE	36	100
IAM	0	0
IAM-PV	0	0
TOCM-PV	0	0

5.6 Hasil Uji Coba TOCM-PV Terhadap 85 Data Campuran

Uji coba dilakukan terhadap 85 *Problem Set Transportation Problem* yang merupakan gabungan dari 15 data sintesis, 34 data dari berbagai jurnal dan 36 data *real*. Untuk membandingkan hasil antara metode VAM, TDM1, IAM, IAM-PV, JHM, TOCMT dan BCE dengan metode yang dikembangkan yaitu IAM-PV. Pada pengujian ini, metode TDM1, IAM, IAM-PV, JHM, TOCMT, BCE dan TOCM-PV diimplementasikan menggunakan program Bahasa C++, sedangkan hasil VAM dan solulsi optimal didapatkan melalui aplikasi TORA.

Tabel 5. 14 menujukkan *improvement percentage* (persentase peningkatan) metode TOCM-PV terhadap VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE. Di dalam tabel tersebut terdapat 3 jenis angka, yaitu angka positif, angka nol dan angka negatif. Angka positif menunjukkan adanya peningkatan persentase TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, yang berarti bahwa total biaya TOCM-PV lebih rendah dari total biaya metode lainnya. Angka nol berarti bahwa total biaya TOCM-PV sama dengan metode lain yang duji, dengan kata lain total biaya TOCM-PV sama dengan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, sehingga tidak ada persentase peningkatan. Sedangkan angka negatif menunjukkan persentase penurunan TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE, karena total biaya TOCM-PV lebih tinggi dibandingkan metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE.

Jumlah perbandingan data *improvement percentage* TOCM-PV terhadap metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, dan BCE pada Tabel 5. 14 dan Tabel 5. 16. Dari 85 data, terdapat 16 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari VAM, 10 data dimana total biaya yang dihasilkan metode TOCM-PV sama dengan VAM, dan 59 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih

besar dari VAM. Jika dibandingkan dengan TDM1, terdapat 44 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari TDM1, 17 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TDM1 dan 24 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TDM1. Untuk metode TOCM-MT, total biaya yang dihasilkan oleh TOCM-PV lebih rendah 34 data dari TOCM-MT, 23 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan TOCM-MT dan 28 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari TOCM-MT. Jika dibandingkan dengan JHM, terdapat 4 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari JHM, 16 data dimana total biaya yang dihasilkan oleh TOCM-PV yang sama dengan JHM, dan 65 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari JHM. Selanjutnya metode BCE, total biaya yang dihasilkan oleh TOCM-PV lebih rendah 3 data dari BCE, 14 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan BCE dan 68 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari BCE. Sedangkan untuk metode IAM, terdapat 73 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari IAM, 5 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan IAM dan 7 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari IAM. Terakhir adalah metode IAM-PV, terdapat 44 data dimana total biaya yang di hasilkan oleh TOCM-PV lebih rendah dari IAM-PV, 32 data dimana total biaya yang dihasilkan oleh TOCM-PV sama dengan IAM-PV dan 7 data dimana total biaya yang dihasilkan oleh TOCM-PV lebih besar dari IAM-PV.

Tabel 5. 15 menujukkan *deviation percentage* (persen penyimpangan) dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal. Terdapat 2 nilai angka, yaitu angka positif dan angka nol. Angka nol berarti tidak ada penyimpangan dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal atau berarti bahwa metode tersebut total biayanya sama dengan solusi optimal. Sedangkan angka positif berarti hasil

total biaya dari metode metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV lebih besar dari solusi optimal, dengan kata lain ada penyimpangan terhadap solusi optimal. Persentase deviasi dihitung dengan menggunakan persamaan (2.28). Pada Tabel 5. 15 dapat dillihat bahwa metode BCE mempunyai jumlah optimal terbanyak dari metode lain, yaitu 78 dari 84 data uji coba.

Jumlah data *deviation percentage* dari metode VAM, TDM1, IAM, IAM-PV, JHM, TOCM-MT, BCE, dan TOCM-PV terhadap solusi optimal terdapat pada Tabel 5. 17. Terdapat 56 data yang dihasilkan oleh VAM dapat mencapai solusi optimal, dan memiliki akurasi sebesar 65,88%. Metode TDM1 menghasilkan 14 data yang mencapai optimal dengan akurasi 16,47%. Terdapat 24 data yang dihasilkan oleh TOCM-MT dapat mencapai solusi optimal, dan memiliki akurasi sebesar 28,24%. Terdapat 71 data yang dihasilkan oleh JHM mencapai solusi optimal dengan akurasi sebesar 83,53%. Terdapat 78 data yang dihasilkan oleh BCE dapat mencapai solusi optimal dengan akurasi sebesar 91,76%. Terdapat 6 data yang di hasilkan oleh IAM dapat mencapai solusi optimal, dan memiliki akurasi sebesar 7,06%. Metode IAM-PV menghasilkan 14 data yang mencapai optimal dengan akurasi 16,47%. Terakhir TOCM-PV mempunyai 16 data yang mencapai optimal dengan akurasi 18,82%.

Hasil uji coba pada penelitian ini menunjukkan bahwa dari 85 data campuran, metode yang memperoleh hasil optimal tertinggi adalah metode VAM yaitu 56 dari 85 data, dengan nilai akurasi 65,88%. Disusul oleh TOCM-PV dengan akurasi 18,82%, selanjutnya metode VAM yang mencapai 16,47% akurasi dan terakhir adalah IAM dengan akurasi 7,06%.

Pada 85 data campuran, persentase kinerja (*improvement percentage*) TOCM-PV terhadap VAM sebesar 18,82%, karena 16 dari 34 contoh kasus mempunyai total biaya TOCM-PV lebih rendah dari pada VAM, diikuti oleh metode IAM sebesar 85,8%, dan IAM-PV sebesar 51,76%.

Tabel 5. 14 *Improvement Percentage* Terhadap 85 Data Campuran

Nama	Inisial Basic Feasible Solution (IBFS)								Optimal	Improvement Percentage (%)						
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV
S01	214394	217000	214828	214394	214394	218289	214974	214449	214394	-0,03	1,18	0,18	-0,03	-0,03	1,76	0,24
S02	216570	218786	217051	216570	216570	220744	217230	216671	216570	-0,05	0,97	0,18	-0,05	-0,05	1,85	0,26
S03	409076	415913	415180	409076	409076	443506	418008	412834	409076	-0,92	0,74	0,57	-0,92	-0,92	6,92	1,24
S04	403140	411304	408778	403140	403140	435026	409150	406941	403140	-0,94	1,06	0,45	-0,94	-0,94	6,46	0,54
S05	1649732	1731106	1679307	1649732	1649732	1791152	1825143	1797381	1649732	-8,95	-3,83	-7,03	-8,95	-8,95	-0,4	1,52
S06	1767639	1966498	1807828	1767639	1767639	1978354	1978354	1966498	1767639	-11,3	0	-8,78	-11,3	-11,3	0,6	0,6
S07	1917304	2005105	1971126	1917304	1917304	2094790	1951496	1936506	1917304	-1	3,42	1,76	-1	-1	7,56	0,77
S08	1750798	1820359	1819816	1750798	1750798	1995849	1831027	1819816	1750798	-3,94	0,03	0	-3,94	-3,94	8,82	0,61
S09	1665063	1783944	1767153	1665063	1665063	1783944	1783944	1767153	1665063	-6,13	0,94	0	-6,13	-6,13	0,94	0,94
S10	1374650	1458304	1403690	1374650	1374650	1490976	1396112	1397278	1374650	-1,65	4,18	0,46	-1,65	-1,65	6,28	-0,08
S11	1509230	1596141	1530430	1509230	1509230	1697345	1544710	1515751	1509230	-0,43	5,04	0,96	-0,43	-0,43	10,7	1,87
S12	1738720	1813469	1772090	1738720	1738720	1893231	1891055	1859734	1738720	-6,96	-2,55	-4,95	-6,96	-6,96	1,77	1,66
S13	522493	531777	518615	515995	499117	997993	635470	523724	457716	-0,24	1,51	-0,99	-1,5	-4,93	47,5	17,58
S14	800103	905189	820059	708066	708066	993123	884184	786174	636525	1,74	13,2	4,13	-11	-11	20,8	11,08
S15	526605	551187	535818	540756	506474	728922	640257	532173	466173	-1,06	3,45	0,68	1,59	-5,07	27	16,88

J01	955	880	880	880	880	1075	880	880	880	7,85	0	0	0	0	18,1	0
J02	1745	1650	1650	1650	1650	1695	1650	1650	1650	5,44	0	0	0	0	2,65	0
J03	779	779	743	743	743	781	779	779	743	0	0	-4,85	-4,85	-4,85	0,26	0
J04	5600	5600	5600	5600	5600	5600	5600	5600	5600	0	0	0	0	0	0	0
J05	880	980	980	840	840	1020	980	980	840	-11,4	0	0	-16,7	-16,7	3,92	0
J06	59	59	61	59	59	66	69	61	59	-3,39	-3,39	0	-3,39	-3,39	7,58	11,59
J07	28	28	28	28	28	28	28	28	28	0	0	0	0	0	0	0
J08	475	475	435	460	435	520	520	520	435	-9,47	-9,47	-19,54	-13	-19,5	0	0
J09	390	400	390	390	390	570	390	390	390	0	2,5	0	0	0	31,6	0
J10	1600	1595	1580	1580	1580	1880	1595	1710	1580	-6,88	-7,21	-8,23	-8,23	-8,23	9,04	-7,21
J11	49	53	53	49	49	61	49	53	49	-8,16	0	0	-8,16	-8,16	13,1	-8,16
J12	470	435	435	420	410	420	415	435	410	7,45	0	0	-3,57	-6,1	-3,6	-4,82
J13	2850	2850	2850	2850	2850	2850	2850	2850	2850	0	0	0	0	0	0	0
J14	187	186	187	183	187	183	183	192	183	-2,67	-3,23	-2,67	-4,92	-2,67	-4,9	-4,92
J15	859	859	799	799	799	868	859	799	799	6,98	6,98	0	0	0	7,95	6,98
J16	273	273	273	273	273	273	273	275	273	-0,73	-0,73	-0,73	-0,73	-0,73	-0,7	-0,73
J17	1220	1160	1160	1170	1170	1315	1160	1160	1160	4,92	0	0	0,85	0,85	11,8	0
J18	204	200	200	218	204	218	200	200	200	1,96	0	0	8,26	1,96	8,26	0

J19	248	248	240	240	240	248	249	249	240	-0,4	-0,4	-3,75	-3,75	-3,75	-0,4	0
J20	820	820	820	820	820	855	820	820	820	0	0	0	0	0	4,09	0
J21	190	190	190	190	192	194	190	190	190	0	0	0	0	1,04	2,06	0
J22	92	83	83	83	83	115	88	88	83	4,35	-6,02	-6,02	-6,02	-6,02	23,5	0
J23	3520	3570	3460	3460	3460	3790	3570	3460	3460	1,7	3,08	0	0	0	8,71	3,08
J24	990	990	910	960	910	1080	990	990	910	0	0	-8,79	-3,13	-8,79	8,33	0
J25	1680	1670	1670	1690	1670	3280	1690	1690	1670	-0,6	-1,2	-1,2	0	-1,2	48,5	0
J26	2400	2400	2400	2340	2280	2360	2400	2360	2280	1,67	1,67	1,67	-0,85	-3,51	0	1,67
J27	2980	2980	2500	2500	2460	3280	3020	2500	2460	16,1	16,1	0	0	-1,63	23,8	17,22
J28	327	291	291	327	291	319	291	291	291	11	0	0	11,01	0	8,78	0
J29	5125	4550	5225	4525	4525	4550	4550	4525	4525	11,7	0,55	13,4	0	0	0,55	0,55
J30	960	960	930	920	920	920	960	960	920	0	0	-3,23	-4,35	-4,35	-4,4	0
J31	859	849	809	809	809	855	849	809	809	5,82	4,71	0	0	0	5,38	4,71
J32	476	465	417	417	417	495	465	417	417	12,4	10,3	0	0	0	15,8	10,32
J33	3778	3572	3513	3487	3458	4147	3513	3513	3458	7,01	1,65	0	-0,75	-1,59	15,3	0
J34	112	117	109	112	109	112	119	125	109	-11,6	-6,84	-14,68	-11,6	-14,7	-12	-5,04
R01	12175097	12208071	12210362	12175097	12175097	12869231	12346537	12322042	12175097	-1,21	-0,93	-0,91	-1,21	-1,21	4,25	0,2
R02	12793609	12869920	12826647	12793609	12793609	13435114	12962364	12939800	12793609	-1,14	-0,54	-0,88	-1,14	-1,14	3,69	0,17

R03	13243021	13331185	13277784	13243021	13243021	13857828	13411490	13402909	13243021	-1,21	-0,54	-0,94	-1,21	-1,21	3,28	0,06
R04	12928429	13006080	12965785	12928429	12928429	13495738	13098507	13092722	12928429	-1,27	-0,67	-0,98	-1,27	-1,27	2,99	0,04
R05	13657053	13775787	13700907	13657053	13657053	14172575	13836003	13832598	13657053	-1,29	-0,41	-0,96	-1,29	-1,29	2,4	0,02
R06	12296587	12324912	12322885	12296587	12296587	12460774	12460774	12455418	12296587	-1,29	-1,06	-1,08	-1,29	-1,29	0,04	0,04
R07	14117586	14236337	14166940	14117586	14117586	14674414	14282395	14253252	14117586	-0,96	-0,12	-0,61	-0,96	-0,96	2,87	0,2
R08	14017419	14148941	14074887	14017419	14017419	14457185	14193481	14188878	14017419	-1,22	-0,28	-0,81	-1,22	-1,22	1,86	0,03
R09	13748896	13871425	13821858	13748896	13748896	14219125	13915646	13913035	13748896	-1,19	-0,3	-0,66	-1,19	-1,19	2,15	0,02
R10	13691814	13820314	13747189	13691814	13691814	14128948	13874252	13869649	13691814	-1,3	-0,36	-0,89	-1,3	-1,3	1,84	0,03
R11	13715927	13839206	13770792	13715927	13715927	14179234	13894606	13890003	13715927	-1,27	-0,37	-0,87	-1,27	-1,27	2,04	0,03
R12	13922982	14061909	14010673	13922982	13922982	14394586	14061061	14058450	13922982	-0,97	0,02	-0,34	-0,97	-0,97	2,34	0,02
R13	5408695	5416127	5461418	5408695	5408695	5775529	5493693	5464813	5408695	-1,04	-0,9	-0,06	-1,04	-1,04	5,38	0,53
R14	5438725	5462780	5511842	5438725	5438725	5550692	5550692	5492135	5438725	-0,98	-0,54	0,36	-0,98	-0,98	1,05	1,05
R15	5663908	5671616	5742635	5663908	5663908	5794206	5769540	5702654	5663908	-0,68	-0,55	0,7	-0,68	-0,68	1,58	1,16
R16	5793809	5802139	5883715	5793809	5793809	6165186	5890110	5831703	5793809	-0,65	-0,51	0,88	-0,65	-0,65	5,41	0,99
R17	6025531	6033695	6093834	6025531	6025531	6406917	6122804	6085634	6025531	-1	-0,86	0,13	-1	-1	5,01	0,61
R18	5365603	5400594	5432628	5363699	5363699	5474312	5474312	5426793	5363699	-1,14	-0,49	0,11	-1,18	-1,18	0,87	0,87
R19	5952349	5960405	6024821	5952349	5952349	6329907	6049536	6012071	5952349	-1	-0,87	0,21	-1	-1	5,02	0,62
R20	6128530	6139109	6197774	6128530	6128530	6503933	6219462	6194698	6128530	-1,08	-0,91	0,05	-1,08	-1,08	4,75	0,4

R21	6183085	6191745	6258172	6183085	6183085	6553937	6271193	6246364	6183085	-1,02	-0,88	0,19	-1,02	-1,02	4,69	0,4
R22	6192863	6203251	6265426	6192863	6192863	6559997	6282042	6258140	6192863	-1,05	-0,88	0,12	-1,05	-1,05	4,6	0,38
R23	6067533	6076125	6145072	6067533	6067533	6435148	6152249	6131289	6067533	-1,05	-0,91	0,22	-1,05	-1,05	4,72	0,34
R24	6235927	6247377	6309471	6235927	6235927	6611957	6328317	6302172	6235927	-1,06	-0,88	0,12	-1,06	-1,06	4,69	0,41
R25	5097639	5165922	5141792	5097639	5097639	5105978	5098012	5098012	5097639	-0,01	1,31	0,85	-0,01	-0,01	0,16	0
R26	5343427	5420814	5393193	5343427	5343427	5351913	5343800	5343800	5343427	-0,01	1,42	0,92	-0,01	-0,01	0,15	0
R27	5566579	5648171	5621027	5566579	5566579	5574307	5567193	5567193	5566579	-0,01	1,43	0,96	-0,01	-0,01	0,13	0
R28	5705771	5793028	5766648	5705771	5705771	5714360	5706144	5706144	5705771	-0,01	1,5	1,05	-0,01	-0,01	0,14	0
R29	6046057	6141013	6114502	6046057	6046057	6054287	6046555	6046555	6046057	-0,01	1,54	1,11	-0,01	-0,01	0,13	0
R30	5147474	5202849	5182501	5147474	5147474	5155068	5147723	5147723	5147474	0	1,06	0,67	0	0	0,14	0
R31	5934380	6028936	6002186	5934380	5934380	5943274	5934878	5934878	5934380	-0,01	1,56	1,12	-0,01	-0,01	0,14	0
R32	6139606	6237512	6217992	6139606	6139606	6149129	6140104	6140104	6139606	-0,01	1,56	1,25	-0,01	-0,01	0,15	0
R33	5980728	6078483	6056181	5980728	5980728	5988697	5981226	5981226	5980728	-0,01	1,6	1,24	-0,01	-0,01	0,12	0
R34	6207890	6304620	6285817	6207890	6207890	6217533	6208388	6208388	6207890	-0,01	1,53	1,23	-0,01	-0,01	0,15	0
R35	5975209	6070899	6050441	5975209	5975209	5982919	5975707	5975707	5975209	-0,01	1,57	1,24	-0,01	-0,01	0,12	0
R36	6255273	6352857	6332291	6255273	6255273	6263838	6255771	6255771	6255273	-0,01	1,53	1,21	-0,01	-0,01	0,13	0

Tabel 5. 15 *Deviation Percentage* Terhadap 85 Data Campuran

Nama	Inisial Basic Feasible Solution (IBFS)								Optimal	*Deviation Percentage* (%)							
	VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV		VAM	TDM1	TOCM-MT	JHM	BCE	IAM	IAM-PV	TOCM-PV
S01	214394	217000	214828	214394	214394	218289	214974	214449	214394	0	1,22	0,2	0	0	1,82	0,27	0,03
S02	216570	218786	217051	216570	216570	220744	217230	216671	216570	0	0,97	0,18	-0,05	-0,1	1,85	0,26	0,05
S03	409076	415913	415180	409076	409076	443506	418008	412834	409076	0	0,74	0,57	-0,92	-0,9	6,92	1,24	0,92
S04	403140	411304	408778	403140	403140	435026	409150	406941	403140	0	1,06	0,45	-0,94	-0,9	6,46	0,54	0,94
S05	1649732	1731106	1679307	1649732	1649732	1791152	1825143	1797381	1649732	0	-3,8	-7,03	-8,95	-9	-0,4	1,52	8,95
S06	1767639	1966498	1807828	1767639	1767639	1978354	1978354	1966498	1767639	0	0	-8,78	-11,3	-11	0,6	0,6	11,25
S07	1917304	2005105	1971126	1917304	1917304	2094790	1951496	1936506	1917304	0	3,42	1,76	-1	-1	7,56	0,77	1
S08	1750798	1820359	1819816	1750798	1750798	1995849	1831027	1819816	1750798	0	0,03	0	-3,94	-3,9	8,82	0,61	3,94
S09	1665063	1783944	1767153	1665063	1665063	1783944	1783944	1767153	1665063	0	0,94	0	-6,13	-6,1	0,94	0,94	6,13
S10	1374650	1458304	1403690	1374650	1374650	1490976	1396112	1397278	1374650	0	4,18	0,46	-1,65	-1,7	6,28	-0,08	1,65
S11	1509230	1596141	1530430	1509230	1509230	1697345	1544710	1515751	1509230	0	5,04	0,96	-0,43	-0,4	10,7	1,87	0,43
S12	1738720	1813469	1772090	1738720	1738720	1893231	1891055	1859734	1738720	0	-2,6	-4,95	-6,96	-7	1,77	1,66	6,96
S13	522493	531777	518615	515995	499117	997993	635470	523724	457716	14,2	1,51	-0,99	-1,5	-4,9	47,5	17,58	14,42
S14	800103	905189	820059	708066	708066	993123	884184	786174	636525	25,7	13,2	4,13	-11	-11	20,8	11,08	23,51

S15	526605	551187	535818	540756	506474	728922	640257	532173	466173	13	3,45	0,68	1,59	-5,1	27	16,88	14,16
J01	955	880	880	880	880	1075	880	880	880	8,52	0	0	0	0	22,2	0	0
J02	1745	1650	1650	1650	1650	1695	1650	1650	1650	5,76	0	0	0	0	2,73	0	0
J03	779	779	743	743	743	781	779	779	743	4,85	4,85	0	0	0	5,11	4,85	4,85
J04	5600	5600	5600	5600	5600	5600	5600	5600	5600	0	0	0	0	0	0	0	0
J05	880	980	980	840	840	1020	980	980	840	4,76	16,7	16,67	0	0	21,4	16,67	16,67
J06	59	59	61	59	59	66	69	61	59	0	0	3,39	0	0	11,9	16,95	3,39
J07	28	28	28	28	28	28	28	28	28	0	0	0	0	0	0	0	0
J08	475	475	435	460	435	520	520	520	435	9,2	9,2	0	5,75	0	19,5	19,54	19,54
J09	390	400	390	390	390	570	390	390	390	0	2,56	0	0	0	46,2	0	0
J10	1600	1595	1580	1580	1580	1880	1595	1710	1580	1,27	0,95	0	0	0	19	0,95	8,23
J11	49	53	53	49	49	61	49	53	49	0	8,16	8,16	0	0	24,5	0	8,16
J12	470	435	435	420	410	420	415	435	410	14,6	6,1	6,1	2,44	0	2,44	1,22	6,1
J13	2850	2850	2850	2850	2850	2850	2850	2850	2850	0	0	0	0	0	0	0	0
J14	187	186	187	183	187	183	183	192	183	2,19	1,64	2,19	0	2,19	0	0	4,92
J15	859	859	799	799	799	868	859	799	799	7,51	7,51	0	0	0	8,64	7,51	0
J16	273	273	273	273	273	273	273	275	273	0	0	0	0	0	0	0	0,73
J17	1220	1160	1160	1170	1170	1315	1160	1160	1160	5,17	0	0	0,86	0,86	13,4	0	0

J18	204	200	200	218	204	218	200	200	200	2	0	0	9	2	9	0	0
J19	248	248	240	240	240	248	249	249	240	3,33	3,33	0	0	0	3,33	3,75	3,75
J20	820	820	820	820	820	855	820	820	820	0	0	0	0	0	4,27	0	0
J21	190	190	190	190	192	194	190	190	190	0	0	0	0	1,05	2,11	0	0
J22	92	83	83	83	83	115	88	88	83	10,8	0	0	0	0	38,6	6,02	6,02
J23	3520	3570	3460	3460	3460	3790	3570	3460	3460	1,73	3,18	0	0	0	9,54	3,18	0
J24	990	990	910	960	910	1080	990	990	910	8,79	8,79	0	5,49	0	18,7	8,79	8,79
J25	1680	1670	1670	1690	1670	3280	1690	1690	1670	0,6	0	0	1,2	0	96,4	1,2	1,2
J26	2400	2400	2400	2340	2280	2360	2400	2360	2280	5,26	5,26	5,26	2,63	0	3,51	5,26	3,51
J27	2980	2980	2500	2500	2460	3280	3020	2500	2460	21,1	21,1	1,63	1,63	0	33,3	22,76	1,63
J28	327	291	291	327	291	319	291	291	291	12,4	0	0	12,37	0	9,62	0	0
J29	5125	4550	5225	4525	4525	4550	4550	4525	4525	13,3	0,55	15,47	0	0	0,55	0,55	0
J30	960	960	930	920	920	920	960	960	920	4,35	4,35	1,09	0	0	0	4,35	4,35
J31	859	849	809	809	809	855	849	809	809	6,18	4,94	0	0	0	5,69	4,94	0
J32	476	465	417	417	417	495	465	417	417	14,2	11,5	0	0	0	18,7	11,51	0
J33	3778	3572	3513	3487	3458	4147	3513	3513	3458	9,25	3,3	1,59	0,84	0	19,9	1,59	1,59
J34	112	117	109	112	109	112	119	125	109	2,75	7,34	0	2,75	0	2,75	9,17	14,68
R01	12175097	12208071	12210362	12175097	12175097	12869231	12346537	12322042	12175097	0	0,27	0,29	0	0	5,7	1,41	1,21

R02	12793609	12869920	12826647	12793609	12793609	13435114	12962364	12939800	12793609	0	0,6	0,26	0	0	5,01	1,32	1,14
R03	13243021	13331185	13277784	13243021	13243021	13857828	13411490	13402909	13243021	0	0,67	0,26	0	0	4,64	1,27	1,21
R04	12928429	13006080	12965785	12928429	12928429	13495738	13098507	13092722	12928429	0	0,6	0,29	0	0	4,39	1,32	1,27
R05	13657053	13775787	13700907	13657053	13657053	14172575	13836003	13832598	13657053	0	0,87	0,32	0	0	3,77	1,31	1,29
R06	12296587	12324912	12322885	12296587	12296587	12460774	12460774	12455418	12296587	0	0,23	0,21	0	0	1,34	1,34	1,29
R07	14117586	14236337	14166940	14117586	14117586	14674414	14282395	14253252	14117586	0	0,84	0,35	0	0	3,94	1,17	0,96
R08	14017419	14148941	14074887	14017419	14017419	14457185	14193481	14188878	14017419	0	0,94	0,41	0	0	3,14	1,26	1,22
R09	13748896	13871425	13821858	13748896	13748896	14219125	13915646	13913035	13748896	0	0,89	0,53	0	0	3,42	1,21	1,19
R10	13691814	13820314	13747189	13691814	13691814	14128948	13874252	13869649	13691814	0	0,94	0,4	0	0	3,19	1,33	1,3
R11	13715927	13839206	13770792	13715927	13715927	14179234	13894606	13890003	13715927	0	0,9	0,4	0	0	3,38	1,3	1,27
R12	13922982	14061909	14010673	13922982	13922982	14394586	14061061	14058450	13922982	0	1	0,63	0	0	3,39	0,99	0,97
R13	5408695	5416127	5461418	5408695	5408695	5775529	5493693	5464813	5408695	0	0,14	0,97	0	0	6,78	1,57	1,04
R14	5438725	5462780	5511842	5438725	5438725	5550692	5550692	5492135	5438725	0	0,44	1,34	0	0	2,06	2,06	0,98
R15	5663908	5671616	5742635	5663908	5663908	5794206	5769540	5702654	5663908	0	0,14	1,39	0	0	2,3	1,87	0,68
R16	5793809	5802139	5883715	5793809	5793809	6165186	5890110	5831703	5793809	0	0,14	1,55	0	0	6,41	1,66	0,65
R17	6025531	6033695	6093834	6025531	6025531	6406917	6122804	6085634	6025531	0	0,14	1,13	0	0	6,33	1,61	1
R18	5365603	5400594	5432628	5363699	5363699	5474312	5474312	5426793	5363699	0,04	0,69	1,29	0	0	2,06	2,06	1,18
R19	5952349	5960405	6024821	5952349	5952349	6329907	6049536	6012071	5952349	0	0,14	1,22	0	0	6,34	1,63	1

R20	6128530	6139109	6197774	6128530	6128530	6503933	6219462	6194698	6128530	0	0,17	1,13	0	0	6,13	1,48	1,08
R21	6183085	6191745	6258172	6183085	6183085	6553937	6271193	6246364	6183085	0	0,14	1,21	0	0	6	1,42	1,02
R22	6192863	6203251	6265426	6192863	6192863	6559997	6282042	6258140	6192863	0	0,17	1,17	0	0	5,93	1,44	1,05
R23	6067533	6076125	6145072	6067533	6067533	6435148	6152249	6131289	6067533	0	0,14	1,28	0	0	6,06	1,4	1,05
R24	6235927	6247377	6309471	6235927	6235927	6611957	6328317	6302172	6235927	0	0,18	1,18	0	0	6,03	1,48	1,06
R25	5097639	5165922	5141792	5097639	5097639	5105978	5098012	5098012	5097639	0	1,34	0,87	0	0	0,16	0,01	0,01
R26	5343427	5420814	5393193	5343427	5343427	5351913	5343800	5343800	5343427	0	1,45	0,93	0	0	0,16	0,01	0,01
R27	5566579	5648171	5621027	5566579	5566579	5574307	5567193	5567193	5566579	0	1,47	0,98	0	0	0,14	0,01	0,01
R28	5705771	5793028	5766648	5705771	5705771	5714360	5706144	5706144	5705771	0	1,53	1,07	0	0	0,15	0,01	0,01
R29	6046057	6141013	6114502	6046057	6046057	6054287	6046555	6046555	6046057	0	1,57	1,13	0	0	0,14	0,01	0,01
R30	5147474	5202849	5182501	5147474	5147474	5155068	5147723	5147723	5147474	0	1,08	0,68	0	0	0,15	0	0
R31	5934380	6028936	6002186	5934380	5934380	5943274	5934878	5934878	5934380	0	1,59	1,14	0	0	0,15	0,01	0,01
R32	6139606	6237512	6217992	6139606	6139606	6149129	6140104	6140104	6139606	0	1,59	1,28	0	0	0,16	0,01	0,01
R33	5980728	6078483	6056181	5980728	5980728	5988697	5981226	5981226	5980728	0	1,63	1,26	0	0	0,13	0,01	0,01
R34	6207890	6304620	6285817	6207890	6207890	6217533	6208388	6208388	6207890	0	1,56	1,26	0	0	0,16	0,01	0,01
R35	5975209	6070899	6050441	5975209	5975209	5982919	5975707	5975707	5975209	0	1,6	1,26	0	0	0,13	0,01	0,01
R36	6255273	6352857	6332291	6255273	6255273	6263838	6255771	6255771	6255273	0	1,56	1,23	0	0	0,14	0,01	0,01

Tabel 5. 16 Jumlah Data *Improvement Percentage* Untuk 85 Data Campuran

Metode	Angka Positif	Angka Nol	Angka Negatif
VAM	16	10	59
TDM1	44	17	24
TOCM-MT	34	23	28
JHM	4	16	65
BCE	3	14	68
IAM	73	5	7
IAM-PV	44	32	7

Tabel 5. 17 Jumlah Data Optimal dan Akurasi Untuk 85 Data Campuran

Metode	Optimal	Akurasi (%)
VAM	56	65,88
TDM1	14	16,47
TOCM-MT	24	28,24
JHM	71	83,53
BCE	78	91,76
IAM	6	7,06
IAM-PV	14	16,47
TOCM-PV	16	18,82

5.7 Hasil Uji Waktu

Hasil uji waktu diperlukan untuk membuktikan bahwa metode IBFS lebih unggul waktu eksekusinya dibandingkan dengan TORA. Seluruh metode dihitung waktu eksekusinya dan tuliskan dalam satuan detik. Berikut adalah tabel hasil uji waktu terhadap seluruh metode.

Tabel 5. 18 Waktu Eksekusi Seluruh Metode

Nama	IBFS								Optimal
	IAM	IAM-PV	VAM	TDM1	TOCM-MT	JHM	BCE	TOCM-PV	(TORA)
S1	2,955	0,2609	0,992	2,769	2,734	0,843	1,108	0,2251	1,414
S2	3,364	0,31	1,45	2,844	3	0,7701	1,262	0,1659	1,512
S3	3,133	0,2863	1,373	2,733	2,972	0,6857	1,041	0,1471	1,125
S4	3,094	0,2456	0,911	2,723	2,886	0,8319	1,279	0,1615	1,237
S5	3,077	0,228	0,853	2,808	2,832	0,708	1,244	0,1268	1,133
S6	2,996	0,2823	0,83	2,746	2,997	1,039	1,558	0,1878	1,323
S7	3,14	0,2863	0,927	2,721	3,009	0,8947	1,228	0,2024	1,439
S8	3,113	0,2745	0,854	2,835	2,754	0,9222	1,253	0,2258	1,471
S9	2,954	0,2515	0,852	2,779	2,831	1,249	1,626	0,1574	1,114
S10	2,916	0,3005	0,861	2,767	2,902	0,7876	1,216	0,1827	1,228
S11	2,922	0,3015	0,799	2,754	2,875	1,009	1,241	0,1689	1,232
S12	2,886	0,253	0,7912	2,818	2,981	0,9244	1,326	0,1786	1,462
S13	11,31	2,926	0,782	9,709	10	6,155	5,94	1,412	1,348
S14	11,08	2,82	0,799	9,767	9,339	7,564	8,532	1,378	1,425
S15	10,87	2,813	0,85	9,699	10,65	5,667	6,827	1,29	1,283
Srinivasan [22]	0,2549	0,1067	0,6332	0,152	21.280	0,169	0,0667	0,05324	0,814
Goyal [23]	0,2486	0,09785	0,8345	16.570	11.390	0,7577	0,1333	0,07549	0,611
Deshmukh [17]	0,2543	0,09603	0,6923	0,119	0,5623	25.020	0,0939	0,05709	0,667
Ramadan [24]	0,435	0,09239	13.615	0,088	0,0692	0,1472	0,0353	0,06375	0,638

Kulkarni [25]	0,2485	0,144	0,7524	0,575	0,5719	0,1374	0,1169	0,06629	0,632
Schrenk [26]	0,2691	0,1047	0,7135	0,995	0,0653	31.190	0,061	0,09492	0,762
Samuel [27]	0,2153	0,09279	0,9231	0,322	0,0915	0,2839	0,0633	0,05431	0,593
Imam [28]	0,2192	0,102	0,852	20.640	15.820	0,1208	0,0434	0,1298	0,514
Adlakha [29]	0,3063	0,1392	0,7634	17.580	14	0,4754	0,1122	0,06742	0,689
kaur [18]	0,2421	0,1033	0,9341	0,091	0,0871	0,1294	0,0725	0,0537	0,662
Patel [30]	0,2683	0,1231	0,7243	0,094	0,0699	0,1405	0,0631	0,1316	0,561
Ahmed [15]	0,258	0,1337	0,7545	0,762	13.740	0,1455	0,0933	0,07127	0,656
Ahmed [21]	0,2502	0,09756	0,5343	0,134	0,0605	0,155	0,0564	0,0513	0,633
Ahmed [21]	0,2737	0,1198	0,8735	16.330	18.780	0,1282	0,0855	0,123	0,752
Uddin [5]	0,2304	0,1038	0,6835	0,333	0,209	0,6506	0,093	0,07462	0,677
Uddin [5]	0,2738	0,1307	0,6324	0,073	0,1125	0,4907	0,0602	0,05548	0,591
Das [4]	0,2318	0,1377	0,7943	0,207	0,1236	0,4826	0,0618	0,06266	0,603
Khan [11]	0,2583	0,112	0,9124	0,08	0,0784	0,1193	0,0583	0,0642	0,666
Azad [31]	0,2692	0,1026	0,7842	0,101	0,4745	0,8759	0,0851	0,1065	0,623
Morade [14]	0,2159	0,1106	0,7598	0,191	0,6761	0,7804	0,045	0,05383	0,599
Jude [32]	0,245	0,1048	0,7564	0,257	0,7062	0,1427	0,0496	0,05438	0,625
Jude [32]	0,2675	0,1367	0,753	0,113	0,0687	15.620	0,0409	0,07749	0,776
Hosseini [6]	0,2373	0,1227	0,7497	0,481	15.190	0,5666	0,1664	0,04961	0,576
Amaliah [2]	0,2095	0,1337	0,7463	0,27	0,1206	0,1198	0,0745	0,05523	0,668
Amaliah [2]	0,2364	0,1443	0,7429	0,259	11.360	0,9035	0,0612	0,1012	0,833
Amaliah [2]	0,2496	0,1322	0,7395	12.010	0,3719	18.530	0,0794	0,05889	0,629
Amaliah [2]	0,2034	0,1458	0,7361	0,07	0,4832	10.920	0,055	0,07497	0,561
Amaliah [2]	0,2107	0,141	0,8423	28.360	0,6593	0,1228	0,052	0,1076	0,691
Juman [1]	0,2672	0,09599	0,7823	11.790	0,0528	0,1638	0,0477	0,05025	0,735
Juman [1]	0,24	0,1002	0,6621	0,773	0,0557	0,1228	0,0427	0,08296	0,633
Juman [1]	0,2366	0,098	0,7824	0,071	0,4776	11.630	0,0499	0,05956	0,755
Juman [1]	0,246	0,112	0,6642	0,289	0,0624	12.720	0,0437	0,05694	0,667
Juman [1]	0,3063	0,1267	0,7141	10.320	0,0991	0,1534	0,0963	0,1024	0,593

Juman [1]	0,319	0,1368	0,8431	0,092	12.490	16.300	0,0923	0,08033	0,655
R1	2,314	0,3148	1	2,237	2,173	0,6445	0,878	0,1939	1,566
R2	2,559	0,2416	1	2,464	2,534	0,771	1,305	0,3095	1,537
R3	2,779	0,3054	1	2,623	2,739	0,964	1,410	0,1543	1,757
R4	3,022	0,2781	1	2,563	2,760	0,909	1,564	0,2625	1,711
R5	2,838	0,355	1	2,665	2,815	1,029	1,539	0,2099	1,965
R6	2,93	0,3144	1	2,652	2,737	0,972	1,424	0,1756	1,623
R7	2,925	0,2769	1	2,617	2,644	1,005	1,473	0,2369	1,884
R8	2,778	0,3382	1	2,562	2,684	0,975	1,539	0,2583	1,941
R9	2,836	0,2831	1	2,741	2,791	1,044	1,546	0,1995	1,754
R10	2,829	0,2473	1	2,664	2,801	0,971	1,541	0,1801	1,946
R11	2,637	0,2482	1	2,599	2,744	0,902	1,439	0,1465	2,081
R12	2,785	0,2921	1	2,617	2,690	0,971	1,620	0,146	1,872
R13	2,31	0,2481	1	2,330	2,243	0,348	0,710	0,133	1,314
R14	2,802	0,2261	1	2,417	2,427	0,328	0,572	0,1322	1,201
R15	2,956	0,2916	1	2,343	2,354	0,363	0,786	0,1393	1,322
R16	2,672	0,2597	1	2,328	2,411	0,381	0,832	0,1574	1,227
R17	2,637	0,2938	1	2,304	2,443	0,538	1,052	0,141	1,145
R18	2,811	0,3637	1	2,330	2,424	0,247	0,454	0,1625	0,912
R19	2,665	0,2843	1	2,325	2,382	0,549	0,917	0,1433	1,554
R20	2,766	0,3063	1	2,476	2,418	0,557	1,099	0,143	1,754
R21	2,575	0,3301	1	2,377	2,496	0,585	0,889	0,2043	1,562
R22	2,663	0,3023	1	2,371	2,470	0,525	1,014	0,1255	1,822
R23	2,685	0,2937	1	2,301	2,493	0,542	0,965	0,1687	1,685
R24	2,708	0,278	1	2,326	2,500	0,588	1,060	0,1505	1,756
R25	2,966	0,2941	1	1,697	1,885	0,782	1,181	0,1355	2,033
R26	2,784	0,2811	1	1,752	1,972	0,780	1,312	0,134	2,099
R27	2,72	0,2651	1	1,694	1,831	0,867	1,360	0,1571	1,975
R28	2,765	0,2487	1	1,697	1,933	0,826	1,312	0,1422	1,722

R29	2,814	0,2802	1	1,747	1,925	0,885	1,245	0,1263	1,836
R30	2,769	0,3295	1	1,721	1,950	0,867	1,315	0,1226	2,037
R31	2,814	0,2773	1	1,703	2,432	0,793	1,320	0,1967	1,944
R32	3,099	0,2842	1	1,679	1,928	0,854	1,266	0,1435	1,681
R33	2,843	0,2674	1	1,773	1,847	0,848	1,334	0,1472	1,947
R34	2,858	0,277	1	1,708	1,879	0,842	1,322	0,1632	1,748
R35	2,822	0,2773	1	1,746	1,932	0,883	1,354	0,1365	1,755
R36	2,748	0,2904	1	1,682	1,887	0,858	1,299	0,1512	1,692

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. TOCM-PV terbukti dapat menghasilkan biaya lebih sedikit dari metode IAM-PV pada hasil uji coba tugas akhir ini. Hal ini karena metode TOCM-PV merubah matriks awal menjadi matriks TOCM.
2. Metode IAM-PV masih belum optimal, ini ditandai dengan nilai akurasi yang kecil yaitu 16,47% untuk 85 data uji coba. TOCM-PV mempunyai nilai akurasi yang lebih tinggi yaitu 18,82%. Sehingga kenaikan nilai akurasi TOCM-PV terhadap IAM-PV sebesar 2,35%. TOCM-PV lebih baik daripada IAM-PV lebih baik daripada IAM-PV, hal ini dibuktikan dengan 44 data dari 85 data uji coba dapat mencapai hasil biaya lebih rendah dari IAM-PV.
3. Waktu eksekusi metode TOCM-PV terbukti lebih cepat dibandingkan waktu eksekusi TORA.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan pengujian yang telah dilakukan.

1. Algoritma TOCM-PV mempunyai nilai error yang cukup rendah, namun pada beberapa *problem set* masih ditemui

kegagalan. sehingga harapannya di masa depan algoritma ini akan dikembangkan pada tahap-tahp tertentu.

2. Pengaplikasian algoritma IBFS dengan menggunakan Dev-C++ cukup memakan waktu karena harus mengubah matriks dan beberapa fungsi tidak tersedia dalam *library* seperti fungsi mencari nilai minmal, fungsi men-*sorting*, dan lain-lain. Disarankan untuk menggunakan *tools* lebih baik atau bahaasa pemrograman yang lebih modern.
3. Pada soal bertipe *Unbalance Problem*, program masih belum bisa menyesuaikan otomatis sehingga diperlukan penambahan *dummy*. Kedepannya diharapkan program yang dibuat dapat meng-*generate* sendiri penambahan *dummy* secara otomatis.

DAFTAR PUSTAKA

[1] Z. A. M. S. Juman and M. A. Hoque, "An efficient heuristic to obtain a better initial feasible solution to the transportation problem," *Appl. Soft Comput. J.*, vol. 34, pp. 813–826, 2015.

[2] B. Amaliah, C. Fatichah, and E. Suryani, "Total opportunity cost matrix – Minimal total: A new approach to determine initial basic feasible solution of a transportation problem," *Egypt. Informatics J.*, vol. 20, no. 2, pp. 131–141, 2019.

[3] Z. A. M. S. Juman and N. G. S. A. Nawarathne, "An efficient alternative approach to solve a transportation problem," *Ceylon J. Sci.*, vol. 48, no. 1, p. 19, 2019.

[4] U. K. Das, R. Ashraful Babu, A. Khan, and U. Helal, "Logical Development Of Vogel's Approximation Method (LD-VAM): An Approach To Find Basic Feasible Solution Of Transportation Problem," *Int. J. Sci. Technol. Res.*, vol. 3, no. 2, 2014.

[5] S. Uddin and A. R. Khan, "Improved Least Cost Method to Obtain a Better IBFS to the Transportation Problem," *J. Appl. Math. Bioinforma.*, vol. 6, no. 2, pp. 1–20, 2016.

[6] E. Hosseini, "Three new methods to find initial basic feasible solution of transportation problems," *Appl. Math. Sci.*, vol. 11, no. 37, pp. 1803–1814, 2017.

[7] Ö. Kirca and A. Şatir, "A Heuristic for Obtaining and Initial Solution for the Transportation Problem," *J. Oper. Res. Soc.*, vol. 41, no. 9, pp. 865–871, Sep. 1990.

[8] M. Mathirajan and B. Meenakshi, "Experimental analysis of some variants of Vogel's approximation method," *Asia-Pacific J. Oper. Res.*, vol. 21, no. 4, pp. 447–462, 2004.

[9] A. Khan, "A Re-solution of the Transportation Problem: An Algorithmic Approach," *Jahangirnagar Univ. J. Sci.*, vol. 34, pp. 49–62, Dec. 2011.

[10] A. Islam, A. Khan, M. Uddin, and M. Malek, "Determination of Basic Feasible Solution of Transportation Problem: A New Approach,"

Jahangirnagar Univ. J. Sci., vol. 35, pp. 101–108, Jun. 2012.

[11] A. R. Khan, A. Vilcu, N. Sultana, and S. S. Ahmed, "Determination of initial basic feasible solution of a transportation problem: a tocm-sum approach," *Bul. Institutului Politeh. Din Iasi*, vol. 61, no. 1, pp. 39–49, 2015.

[12] M. Ashraful Babu, M. Abu Helal, M. Sazzad Hasan, and U. Kanti Das, "Implied Cost Method (ICM): An Alternative Approach to Find the Feasible Solution of Transportation Problem," *Type Double Blind Peer Rev. Int. Res. J. Publ. Glob. Journals Inc*, vol. 14, no. 1, 2014.

[13] P. Kousalya and P.Malarvizhi, "A New Technique for Finding Initial Basic Feasible Solution to Transportation Problem," *Int. J. Eng. Manag. Res.*, vol. 6, no. 3, pp. 26–32, 2016.

[14] N. M. Morade, "New Method to find initial basic feasible solution of Transportation Problem using MSDM," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 12, pp. 223–226, 2017.

[15] M. M. Ahmed, A. R. Khan, F. Ahmed, and M. S. Uddin, "Incessant Allocation Method for Solving Transportation Problems," *Am. J. Oper. Res.*, vol. 06, no. 03, pp. 236–244, 2016.

[16] M. M. Ahmed, M. A. Islam, M. Katun, S. Yesmin, and M. S. Uddin, "New Procedure of Finding an Initial Basic Feasible Solution of the Time Minimizing Transportation Problems," *Open J. Appl. Sci.*, vol. 05, no. 10, pp. 634–640, 2015.

[17] N. M. Deshmukh, "an Innovative Method for Solving Transportation," *Int. J. Phys. Math. Sci.*, vol. 2, no. 3, pp. 86–91, 2012.

[18] L. Kaur, M. Rakshit, and S. Singh, "An Improvement in Maximum Difference Method to Find Initial Basic Feasible Solution for Transportation Problem," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 9, pp. 533–535, 2018.

[19] D. Informatika, *PENCARIAN INITIAL BASIC FEASIBLE SOLUTION PADA TRANSPORTATION PROBLEM DENGAN MENGGUNAKAN INCESSANT*. 2018.

[20] H. A. Taha, "Taha - Operation Research 8Ed.pdf." p. 838, 2007.

[21] M. M. Ahmed, A. R. Khan, M. S. Uddin, and F. Ahmed, "A New Approach to Solve Transportation Problems," *Open J. Optim.*, vol. 05, no. 01, pp. 22–30, 2016.

[22] V. Srinivasan and G. L. Thompson, "Cost operator algorithms for the transportation problem," *Math. Program.*, vol. 12, no. 1, pp. 372–391, 1977.

[23] S. K. Goyal, "Improving VAM for Unbalanced Transportation Problems," *J. Oper. Res. Soc.*, vol. 35, no. 12, pp. 1113–1114, 1984.

[24] S. Z. Ramadan and I. Z. Ramadan, "Hybrid two-stage algorithm for solving transportation problem," *Mod. Appl. Sci.*, vol. 6, no. 4, pp. 12–22, 2012.

[25] S. S. Kulkarni, "on Solution To Modified Unbalanced Transportation Problem," *Bull. Marathwada Math. Soc.*, vol. 11, no. 2, pp. 20–26, 2010.

[26] S. Schrenk, G. Finke, and V.-D. Cung, "Two classical transportation problems revisited: Pure constant fixed charges and the paradox," *Math. Comput. Model.*, vol. 54, pp. 2306–2315, Nov. 2011.

[27] A. E. Samuel, "Improved Zero Point Method (IZPM) for the Transportation Problems," *Appl. Math. Sci.*, vol. 6, no. 109, pp. 5421–5426, 2012.

[28] T. Imam, G. Elsharawy, M. Gomah, and I. Samy, "Solving Transportation Problem Using Object-Oriented Model," *Comput. Sci. Netw. Secur.*, vol. 9, Jan. 2009.

[29] V. Adlakha and K. Kowalski, "Alternate Solutions Analysis For Transportation Problems," *J. Bus. Econ. Res.*, vol. 7, no. 11, pp. 41–50, Feb. 2009.

[30] R. . G.Patel, D. P. H. Bhathawala, and D. B. . S.Patel, "'An Alternate Approach to Find an Optimal Solution of a

Transportation Problem.,'" *IOSR J. Math.*, vol. 13, no. 01, pp. 01–05, 2017.

[31] S. M. A. K. Azad and M. B. Hossain, "A New Method for Solving Transportation Problems Considering Average Penalty," *IOSR J. Math.*, vol. 13, no. 01, pp. 40–43, 2017.

[32] O. Jude, "Comparison of Existing Methods of Solving Linear Transportation Problems with a New Approach," *Int. J. Innov. Sci. Math.*, vol. 4, no. 5, pp. 158–163, 2016.

LAMPIRAN A

Lampiran A. Lima Belas Data Sintesis

S1:

$[C_{ij}]_{2x94}$ = [45 33 48 51 42 41 35 45 44 44 42 42 50 40
46 53 39 45 38 53 44 55 56 44 39 59 47 44 47
41 42 49 56 35 46 37 46 46 44 45 39 32 48 60
45 53 48 36 39 49 52 52 46 47 46 47 42 46 45
48 50 45 45 36 34 47 43 45 43 49 42 37 59 46
45 45 36 46 54 49 46 48 44 65 45 54 48 45 45
59 33 49 52 52; 55 40 42 48 43 50 51 52 39
56 44 45 58 46 35 39 36 44 44 44 51 44 51 57
45 40 49 41 49 45 59 47 38 38 51 48 39 39 40
46 47 49 50 46 42 47 53 39 48 39 43 38 41 47
41 37 44 58 46 44 43 40 50 45 44 45 39 41 51
41 52 45 51 47 42 43 53 34 50 46 44 46 43 52
33 52 38 45 51 51 56 49 44 49]

$[S_i]_{2x1}$ = [3407, 1605]

$[D_j]_{1x94}$ = [46 51 50 65 61 47 54 58 55 57 48 53 48 47
58 53 55 55 43 60 64 50 52 49 57 46 45 50 58
61 51 54 51 53 59 52 60 63 49 56 64 56 52 54
52 64 57 51 56 50 55 48 53 54 60 58 52 51 41
50 55 52 59 54 57 47 57 39 63 50 49 45 54 51
50 54 42 53 54 55 57 50 60 55 47 49 52 57 51
49 58 63 51 51]

S2:

$[C_{ij}]_{2x94}$ = [45 33 48 51 42 41 35 45 44 44 42 42 50 40
46 53 39 45 38 53 44 55 56 44 39 59 47 44 47

		41 42 49 56 35 46 37 46 46 44 45 39 32 48 60 45 53 48 36 39 49 52 52 46 47 46 47 42 46 45 48 50 45 45 36 34 47 43 45 43 49 42 37 59 46 45 45 36 46 54 49 46 48 44 65 45 54 48 45 45 59 33 49 52 52; 55 40 42 48 43 50 51 52 39 56 44 45 58 46 35 39 36 44 44 44 51 44 51 57 45 40 49 41 49 45 59 47 38 38 51 48 39 39 40 46 47 49 50 46 42 47 53 39 48 39 43 38 41 47 41 37 44 58 46 44 43 40 50 45 44 45 39 41 51 41 52 45 51 47 42 43 53 34 50 46 44 46 43 52 33 52 38 45 51 51 56 49 44 49]
$[S_i]_{2x1}$	=	[3407, 1658]
$[D_j]_{1x94}$	=	[50 55 60 53 58 54 51 58 46 63 49 54 49 62 50 61 55 63 56 47 56 48 56 61 47 60 58 46 60 52 54 59 51 48 53 48 51 50 50 51 59 59 52 52 45 50 48 47 53 58 56 48 59 49 49 56 55 50 52 59 51 50 53 61 57 56 49 48 53 46 60 48 54 53 53 60 54 56 55 57 58 52 63 56 51 55 53 54 57 58 52 64 52 57]
S3:		
$[C_{ij}]_{2x94}$	=	[68 21 55 36 65 65 74 38 56 67 68 54 95 20 42 77 87 96 82 80 93 64 43 89 53 78 86 36 25 19 46 9 30 67 10 81 84 82 88 26 44 8 66 92 56 75 70 3 33 5 49 31 42 73 36 9 65 92 30 89 70 12 82 51 69 40 90 66 27 54 83 48 48 39 31 85 94 25 73 47 70 17 57 57 26 68 82 52 66 21 50 11 95 91; 40 1 43 10 84 39 65 62 25 72 33 14 61 92 18 92 21 15 70 20 52 52 85 54 38 24 21 45 59 65 19 8 33 48 5 14 31 4 68 30 85 18

72 8 32 78 88 33 74 86 98 73 20 94 80 41 1
67 29 58 52 26 39 86 27 48 8 2 51 35 38 18
40 48 97 55 19 15 94 75 5 11 72 56 71 33 71
96 96 62 2 19 71 26]

$[S_i]_{2x1}$ = [5220 4331]

$[D_j]_{1x94}$ = [46 51 50 65 61 47 54 58 55 57 48 53 48 47
58 53 55 55 43 60 64 50 52 49 57 46 45 50 58
61 51 54 51 53 59 52 60 63 49 56 64 56 52 54
52 64 57 51 56 50 55 48 53 54 60 58 52 51 41
50 55 52 59 54 57 47 57 39 63 50 49 45 54 51
50 54 42 53 54 55 57 50 60 55 47 49 52 57 51
49 58 63 51 51]

S4:

$[C_{ij}]_{2x94}$ = [28 32 28 53 44 93 51 92 10 61 52 24 8 9 68
50 43 7 38 88 53 39 81 73 33 41 51 85 87 49
51 90 85 99 74 23 20 52 16 34 100 49 46 16
88 33 32 64 22 20 11 15 27 75 54 30 4 27 33
81 97 4 84 37 31 36 11 19 21 35 24 62 35 90
97 1 6 83 16 9 56 24 11 77 64 56 43 54 8 23
73 97 11 24; 58 69 71 60 22 95 25 48 78 74
65 99 9 25 34 53 46 52 90 10 46 51 56 79 39
70 57 47 96 100 45 68 100 36 51 80 47 86 85
20 78 76 47 97 55 8 39 91 78 91 9 59 81 36
77 71 9 33 23 40 29 10 93 63 65 1 17 74 73
96 70 68 32 40 72 22 95 57 92 73 11 14 23 36
67 58 93 25 39 41 12 34 19 68]

$[S_i]_{2x1}$ = [4261, 5122]

$[D_j]_{1x94}$	=	[86 101 99 113 66 188 76 140 88 135 117 123 17 34 102 103 89 59 128 98 99 90 137 152 72 111 108 132 183 149 96 158 185 135 125 103 67 138 101 54 178 125 93 113 143 41 71 155 100 111 20 74 108 111 131 101 13 60 56 121 126 14 177 100 96 37 28 93 94 131 94 130 67 130 169 23 101 140 108 82 67 38 34 113 131 114 136 79 47 64 85 131 30 92]
S5:		
$[C_{ij}]_{2x94}$	=	[68 21 18 92 31 72 33 19 49 66 33 63 43 91 96 97 92 53 56 74 98 73 93 79 48 20 35 70 8 31 31 15 64 57 36 58 79 25 92 85 86 72 77 40 21 40 89 14 39 5 8 84 65 41 91 78 66 23 98 75 18 61 48 10 57 72 55 72 22 84 76 20 17 33 31 2 57 10 31 36 75 63 73 37 84 51 69 37 77 9 12 43 22 26; 73 9 40 43 93 95 43 34 96 59 71 46 62 29 28 1 48 61 6 97 13 17 70 18 44 63 54 63 56 78 46 79 7 22 83 6 42 89 23 86 50 76 58 24 49 83 25 59 6 88 28 98 2 45 94 39 61 72 9 47 27 40 4 92 48 60 94 55 15 93 29 34 85 89 28 50 90 100 10 60 80 19 68 82 47 10 99 6 52 7 49 44 44 52]
$[S_i]_{2x1}$	=	[16599, 31138]
$[D_j]_{1x94}$	=	[69 893 62 597 505 315 273 961 823 445 985 968 505 948 878 141 312 75 64 467 507 988 414 863 537 443 109 690 884 802 476 331 880 844 714 661 354 618 117 18 866 912 762 138 346 108 600 485 432 882 321 868 305 995 544 51 287 334 519 51 356 506 86 957

689 525 553 85 571 302 601 848 590 287 90
355 450 153 574 787 450 967 118 731 720
192 692 867 405 648 441 410 33 356]

S6:

$[C_{ij}]_{2x94}$ = [10 38 67 66 43 88 89 92 15 94 71 96 19 61
93 95 48 43 80 95 10 81 18 19 54 61 68 6 50
67 72 90 72 85 49 9 8 81 91 93 99 63 91 27 9
84 11 84 28 12 56 43 4 74 43 83 40 84 66 36
79 31 77 28 89 24 20 88 86 32 56 15 53 44 3
74 27 3 88 9 33 46 10 36 40 64 94 5 48 57 84
75 6 70; 43 12 86 74 97 25 35 40 73 51 34 32
99 96 99 75 97 22 55 46 12 43 26 25 73 34 42
98 69 8 16 30 4 10 80 21 11 96 36 12 29 91
22 66 15 46 37 43 48 17 31 98 20 66 25 23 13
44 57 49 66 34 43 10 86 27 65 24 23 90 18 69
100 53 21 67 66 35 54 24 74 19 85 89 46 63
44 49 9 60 93 43 99 88]

$[S_i]_{2x1}$ = [11480, 34266]

$[D_j]_{1x94}$ = [970 958 154 965 803 283 674 10 78 552 456
979 25 459 973 954 523 746 405 648 206 169
902 693 726 90 591 386 378 277 455 874 153
875 181 462 60 875 183 440 968 104 707 565
864 151 211 473 594 524 904 605 91 137 483
666 400 198 477 345 789 369 560 156 993 91
710 25 945 893 31 757 716 520 5 888 742
522 142 399 349 379 347 463 725 676 383
555 356 667 122 296 103 14]

S7:

$[C_{ij}]_{2x94}$ = [70 50 12 13 18 68 96 98 49 45 4 8 47 56 39 51 42 16 11 37 33 42 80 52 56 78 77 60 48 46 39 73 99 39 34 45 45 43 45 30 36 5 7 32 85 99 14 90 95 34 43 95 100 13 58 17 43 30 77 3 66 46 94 82 99 35 75 40 43 46 7 34 9 30 97 12 34 12 44 59 74 83 74 50 90 16 51 35 31 50 98 85 44 50; 43 43 8 85 80 84 49 59 82 96 32 92 57 4 59 63 49 44 19 46 1 63 52 70 49 44 84 36 38 83 93 62 48 52 51 3 97 40 77 8 32 96 4 59 73 73 42 65 73 44 57 98 90 4 44 31 40 99 99 36 100 32 5 4 62 88 48 44 68 45 85 34 72 36 91 67 86 71 79 46 58 12 5 42 75 46 37 37 97 53 91 5 87 5]

$[S_i]_{2x1}$ = [13781, 35594]

$[D_j]_{1x94}$ = [915 647 415 552 48 64 849 470 750 171 810 330 288 504 948 717 310 318 986 260 76 611 610 158 322 716 109 860 341 711 949 440 800 620 918 490 402 522 106 995 889 347 136 216 859 363 687 744 782 377 917 473 321 819 746 817 388 469 3 182 240 395 744 287 698 311 506 739 763 268 558 806 530 164 675 414 708 238 736 744 257 446 311 379 94 652 832 664 939 658 626 796 3 551]

S8:

$[C_{ij}]_{2x94}$ = [28 32 28 53 44 93 51 92 10 61 52 24 8 9 68 50 43 7 38 88 53 39 81 73 33 41 51 85 87 49 51 90 85 99 74 23 20 52 16 34 100 49 46 16

88 33 32 64 22 20 11 15 27 75 54 30 4 27 33
81 97 4 84 37 31 36 11 19 21 35 24 62 35 90
97 1 6 83 16 9 56 24 11 77 64 56 43 54 8 23
73 97 11 24; 58 69 71 60 22 95 25 48 78 74
65 99 9 25 34 53 46 52 90 10 46 51 56 79 39
70 57 47 96 100 45 68 100 36 51 80 47 86 85
20 78 76 47 97 55 8 39 91 78 91 9 59 81 36
77 71 9 33 23 40 29 10 93 63 65 1 17 74 73
96 70 68 32 40 72 22 95 57 92 73 11 14 23 36
67 58 93 25 39 41 12 34 19 68]

$[S_i]_{2x1}$ = [20865, 26877]

$[D_j]_{1x94}$ = [510 283 94 606 98 89 267 49 494 541 667
982 961 310 161 255 951 649 918 414 825
471 710 754 592 532 948 362 514 649 904
892 327 957 253 909 592 963 257 896 529
425 695 450 240 301 55 935 614 247 681 44
181 749 592 346 315 257 509 325 330 740
306 4 340 309 740 126 983 640 15 805 625
984 955 530 176 295 643 332 742 276 774
931 623 7 464 443 612 612 175 413 225 471]

S9:

$[C_{ij}]_{2x94}$ = [69 33 95 27 19 27 8 29 4 78 42 86 61 93 44
26 31 64 32 80 84 86 61 23 36 60 87 48 12 74
52 46 11 62 77 61 20 100 59 30 52 83 15 38
14 79 54 57 36 26 71 4 3 65 21 72 70 20 64
93 23 9 86 22 39 43 40 16 89 27 16 31 18 60
78 23 80 67 23 71 71 75 1 100 2 82 52 92 91
97 13 6 2 65; 61 1 26 63 33 83 64 51 70 35 22
4 60 63 18 88 12 97 74 14 16 98 41 92 41 60

		24 3 78 74 78 33 51 49 69 16 86 29 30 78 28 81 62 52 72 17 39 24 14 49 9 77 86 69 44 23 30 1 82 45 49 62 21 57 59 14 42 76 5 61 36 34 13 85 51 34 11 72 26 43 91 95 54 6 23 27 91 75 42 45 53 51 93 8]
$[S_i]_{2x1}$	=	[11042, 35765]
$[D_j]_{1x94}$	=	[523 818 818 314 770 517 503 285 740 143 332 905 961 494 800 31 784 963 636 59 5 460 245 451 299 297 157 937 704 903 148 287 723 338 711 603 970 265 429 763 522 387 667 369 430 43 186 584 437 721 189 958 735 282 681 423 291 76 49 494 621 606 271 974 830 937 240 607 170 385 886 483 777 607 699 177 981 141 22 11 733 854 848 7 565 709 238 71 749 512 609 179 473 220]
S10:		
$[C_{ij}]_{2x94}$	=	[82 77 81 42 78 77 37 40 46 66 48 7 12 8 59 72 89 67 31 48 67 97 27 39 96 30 37 18 88 74 11 26 27 42 56 3 79 1 97 62 4 50 71 19 8 52 89 23 39 5 43 27 1 88 71 60 85 56 71 81 80 35 80 1 88 37 69 100 87 13 1 9 86 18 65 82 40 51 60 91 10 1 92 15 36 56 23 4 2 100 68 2 13 9; 20 84 85 49 63 38 36 82 52 65 55 99 18 75 57 40 78 53 65 3 77 50 79 70 7 43 50 88 58 49 78 70 75 35 95 5 48 67 74 26 6 76 1 87 69 54 88 37 31 19 40 60 72 20 8 13 23 87 29 21 3 27 74 4 68 16 48 47 92 9 86 85 6 37 31 9

		47 54 56 39 90 23 73 55 83 44 1 56 29 68 14 79 64 33]
$[S_i]_{2x1}$	=	[11572, 28755]
$[D_j]_{1x94}$	=	[718 141 998 138 13 791 335 220 668 531 220 53 874 464 398 223 255 705 601 524 997 188 39 267 105 908 29 130 291 662 587 148 992 464 14 63 183 176 341 414 513 195 55 764 50 714 325 233 675 763 222 344 481 920 357 305 364 869 408 753 613 187 151 268 771 26 194 416 569 813 706 319 447 822 612 916 674 320 416 689 17 291 70 29 872 537 158 850 138 722 602 75 516 343]
S11:		
$[C_{ij}]_{2x94}$	=	[18 36 57 43 33 85 31 72 43 13 7 7 83 68 40 42 39 14 52 94 36 99 82 20 93 6 73 13 60 10 39 90 59 69 90 36 71 17 5 31 40 10 22 9 54 15 16 9 59 68 21 38 67 71 52 28 48 47 44 42 12 63 20 100 39 43 56 71 67 40 62 21 3 80 98 28 33 28 4 57 52 79 52 77 35 14 89 33 80 11 28 22 48 74; 61 60 69 15 60 73 73 25 47 38 4 99 75 97 38 3 93 23 96 21 61 66 62 89 49 45 20 90 76 16 43 1 22 56 68 27 38 2 51 74 39 21 25 48 2 40 25 28 2 42 44 46 68 31 97 5 84 47 99 64 55 32 24 3 55 86 82 37 40 51 16 60 83 26 95 82 13 8 42 83 54 54 37 58 100 59 69 87 57 39 60 64 36 99]
$[S_i]_{2x1}$	=	[17027, 28173]

$[D_j]_{1x94}$	=	[662 759 867 789 625 264 703 765 20 599 380 836 364 659 843 429 86 21 554 576 374 574 289 245 90 659 337 257 168 477 333 631 674 525 512 344 690 720 811 12 675 222 882 154 788 274 607 136 938 889 917 383 543 59 188 182 700 555 808 862 285 174 322 52 984 205 88 805 819 109 429 957 597 218 249 589 295 346 410 226 460 560 327 374 946 201 60 228 461 503 472 956 534 674]
S12:		
$[C_{ij}]_{2x94}$	=	[68 21 18 92 31 72 33 19 49 66 33 63 43 91 96 97 92 53 56 74 98 73 93 79 48 20 35 70 8 31 31 15 64 57 36 58 79 25 92 85 86 72 77 40 21 40 89 14 39 5 8 84 65 41 91 78 66 23 98 75 18 61 48 10 57 72 55 72 22 84 76 20 17 33 31 2 57 10 31 36 75 63 73 37 84 51 69 37 77 9 12 43 22 26; 73 9 40 43 93 95 43 34 96 59 71 46 62 29 28 1 48 61 6 97 13 17 70 18 44 63 54 63 56 78 46 79 7 22 83 6 42 89 23 86 50 76 58 24 49 83 25 59 6 88 28 98 2 45 94 39 61 72 9 47 27 40 4 92 48 60 94 55 15 93 29 34 85 89 28 50 90 100 10 60 80 19 68 82 47 10 99 6 52 7 49 44 44 52]
$[S_i]_{2x1}$	=	[14893, 31967]
$[D_j]_{1x94}$	=	[62 190 789 60 811 1 615 146 163 456 404 283 651 651 405 41 376 647 517 710 205 227 7 682 996 404 996 156 693 490 628 939 367

	109 583 665 892 829 29 897 358 866 921 904 48 233 111 804 859 55 148 957 883 54 310 403 339 508 134 698 434 408 49 703 364 646 202 447 161 147 996 517 395 618 486 936 952 629 755 46 924 533 92 227 982 379 947 803 900 591 702 152 897 475]
S13: $[C_{ij}]_{10x94}$ =	[13 93 61 63 30 96 77 26 93 99 65 52 47 46 28 82 80 39 20 94 14 71 51 72 51 92 59 13 23 74 97 15 18 85 80 79 58 17 47 77 64 89 61 80 78 44 23 26 88 10 15 84 78 68 93 58 90 35 78 40 98 63 90 30 22 59 97 97 71 64 85 27 40 20 36 7 59 60 78 96 77 97 26 73 68 57 65 3 49 88 21 80 29 30; 96 35 79 30 43 64 9 26 16 4 20 36 71 77 95 44 30 32 37 27 92 52 49 80 97 55 16 50 98 98 69 68 43 99 85 94 72 10 18 27 66 50 58 96 84 95 84 57 37 47 98 88 12 27 70 96 27 85 66 20 43 38 4 21 58 46 67 27 33 60 39 38 95 21 76 65 78 48 12 63 20 1 98 24 38 43 14 1 1 24 74 13 66 24; 66 44 86 24 62 99 98 83 10 65 75 72 69 5 50 87 34 92 11 33 37 20 46 5 31 78 97 66 8 25 96 84 24 52 90 15 77 92 51 32 27 67 31 92 1 31 55 5 86 58 11 16 46 19 68 80 61 54 58 13 21 69 75 28 42 1 72 98 18 29 83 3 74 69 86 75 44 63 40 48 40 97 26 11 10 9 16 40 97 96 69 27 62 76; 60 79 49 48 57 45 58 71 29 47 83 4 3 75 51 32 35 50 74 11 25 58 100 30 46 19 39 14 4 88 29 74 29 39 80 87 90 72 67 57 35 1 86 53 40 86 47 9 46 10 4 97 33 13 30 90 97 87 90 28 62 81

54 33 71 6 35 85 20 16 85 43 20 2 66 19 85
68 30 81 74 51 92 66 84 52 31 33 82 33 56
100 92 42; 24 65 70 28 3 79 72 65 13 75 62 7
6 51 40 7 8 42 93 63 87 29 90 68 27 92 45 4
53 96 19 18 74 94 43 30 35 95 6 64 17 95 66
4 71 89 39 29 24 98 71 32 32 52 52 76 54 22
58 46 41 43 46 72 7 6 45 8 43 59 64 81 61 1
91 34 98 84 62 69 9 36 93 73 46 14 45 83 65
40 80 78 20 25; 44 80 81 96 93 86 91 15 17
32 26 97 3 60 91 97 84 11 7 67 28 52 43 76
42 73 9 43 79 31 74 24 4 46 35 35 16 12 71 3
52 32 72 26 48 67 87 61 86 34 38 36 76 61 80
12 80 18 35 37 10 38 38 90 55 30 11 40 69 94
52 52 98 13 4 27 100 100 73 92 42 33 48 32 9
4 20 66 78 4 35 81 6 84; 13 97 10 2 33 4 11
35 77 89 100 50 82 78 65 47 97 89 3 20 99 19
95 14 68 99 44 99 26 89 22 10 98 61 28 12 94
14 23 27 22 54 59 84 14 46 51 83 8 53 53 77
56 43 68 3 44 6 32 6 80 12 60 1 97 87 37 37
97 99 70 78 50 96 81 39 75 31 87 7 71 32 10
52 8 12 51 16 23 54 74 44 36 98; 47 45 77 63
43 34 54 59 81 59 66 52 7 74 23 65 29 97 98
70 56 42 86 82 83 14 16 19 78 95 31 31 17 67
22 95 34 82 24 11 48 60 55 25 90 43 31 65 45
55 31 43 34 96 51 19 1 59 84 68 49 28 86 57
93 45 10 24 6 81 82 7 20 96 83 29 17 10 8 12
40 2 46 16 61 5 56 84 47 11 83 1 16 36; 46 4
18 85 86 30 97 1 84 15 39 13 73 86 100 80 76
1 69 32 92 92 54 63 85 90 55 58 36 65 63 7
37 53 71 66 86 70 95 78 10 94 100 82 43 67
49 92 52 30 55 49 93 39 44 94 36 33 44 88 90
84 69 74 81 36 3 65 79 40 41 17 75 16 24 41
93 77 32 51 50 83 72 51 50 24 24 38 47 81 76

		56 51 29; 75 16 79 6 81 36 100 54 90 80 45 41 55 32 68 47 4 10 89 31 43 94 23 55 12 48 43 23 20 56 60 15 84 29 78 75 90 34 96 2 39 41 22 99 86 72 33 47 13 13 82 80 13 18 11 94 55 76 82 35 61 46 75 7 38 29 42 95 13 5 4 86 92 83 67 83 75 53 19 100 79 92 19 11 73 40 13 41 72 6 20 32 94 74]
$[S_i]_{10x1}$	=	[4486, 4876, 4297, 4086, 4413, 4557, 4089, 4602, 4839, 6562]
$[D_j]_{1x94}$	=	[523 818 818 314 770 517 503 285 740 143 332 905 961 494 800 31 784 963 636 59 5 460 245 451 299 297 157 937 704 903 148 287 723 338 711 603 970 265 429 763 522 387 667 369 430 43 186 584 437 721 189 958 735 282 681 423 291 76 49 494 621 606 271 974 830 937 240 607 170 385 886 483 777 607 699 177 981 141 22 11 733 854 848 7 565 709 238 71 749 512 609 179 473 220]
S14:		
$[C_{ij}]_{10x94}$	=	[81 32 22 15 64 46 36 33 17 15 23 19 100 70 95 83 91 29 70 6 90 99 68 3 4 36 88 98 42 63 85 62 34 59 70 80 85 88 63 8 58 68 82 51 99 83 89 27 82 93 59 48 8 62 66 89 49 98 18 57 4 81 83 91 10 59 38 90 36 85 25 13 34 68 10 29 16 65 65 82 88 73 1 50 99 72 14 5 78 51 47 36 31 13; 42 52 80 55 25 27 71 33 45 83 97 82 9 36 64 97 29 70 77 80 27 2 11 3 94 70 72 21 89 25 89 17 52 39 69 12 1 76 22 11 35 6 84 12 35 61 82 82 68 92 83 64 84 43 85 67

3 52 52 19 31 82 63 35 52 28 17 38 21 63 56
91 87 71 16 20 82 65 91 18 15 24 53 53 23 87
87 36 72 18 55 66 72 11; 84 70 69 62 11 3 52
74 93 41 51 40 10 71 85 1 20 45 69 9 57 1 74
29 57 54 99 35 38 85 86 98 81 14 48 60 82 76
32 54 8 41 35 44 71 17 35 86 74 16 60 100 77
3 55 41 79 90 15 83 91 100 16 20 7 79 58 48
53 2 62 88 5 30 76 92 40 41 2 16 19 77 73 63
11 71 54 7 45 30 52 91 13 93; 43 8 63 9 45 6
52 91 66 37 38 98 91 35 61 70 62 6 78 79 49
21 16 7 30 95 45 54 96 58 60 97 54 41 49 74
52 33 76 80 22 64 95 75 41 65 1 24 17 53 85
31 35 97 98 49 13 37 24 3 17 58 97 71 35 51
53 59 79 97 33 75 97 73 76 79 20 95 70 58 28
42 25 93 45 38 44 7 17 87 44 81 43 68; 100
20 74 78 94 80 97 62 84 4 94 25 19 77 56 63
44 52 15 17 30 70 94 3 30 6 62 92 89 14 27
11 90 95 87 12 64 9 46 6 78 54 7 88 56 67 91
85 96 89 46 35 71 23 17 91 30 69 60 21 40 30
36 31 97 19 98 54 76 81 95 73 40 27 42 30 45
52 55 32 45 88 56 40 49 76 62 91 84 95 37 6
23 33; 73 43 34 60 64 20 33 98 49 14 60 39
91 15 94 99 91 86 41 47 5 16 89 21 55 77 96
11 86 87 45 92 6 64 6 32 30 78 90 74 77 70
35 77 81 88 49 48 88 74 52 14 64 14 25 16 85
21 57 16 22 60 82 87 68 86 48 91 28 22 64 77
19 37 95 93 88 75 6 61 10 24 77 91 80 92 53
5 41 55 72 70 4 87; 98 92 71 17 98 28 28 55
18 79 5 35 82 94 54 61 98 51 29 31 97 25 58
90 58 92 86 18 42 33 18 97 10 1 2 5 39 29 55
7 20 33 73 87 66 33 88 86 37 62 90 48 19 61
52 87 85 18 45 27 13 90 89 94 82 43 87 92 74
75 68 30 12 10 26 4 32 77 79 94 76 54 100 78

11 83 84 87 2 46 59 62 36 6; 26 7 20 41 16 78 18 20 57 5 62 23 40 5 75 18 12 2 55 48 14 19 71 92 10 99 50 11 91 81 61 35 23 54 10 16 55 58 75 79 89 17 85 9 30 36 60 12 52 63 90 44 48 25 53 52 30 13 4 12 75 59 71 56 74 69 79 51 67 14 59 95 21 52 39 94 4 41 62 1 56 14 67 45 27 30 57 86 40 80 82 2 42 85; 69 82 55 96 18 45 19 94 57 46 83 95 95 80 6 67 68 2 60 88 31 38 83 41 69 64 64 17 36 29 98 90 10 63 90 82 42 97 68 76 31 85 69 39 19 96 88 31 97 59 60 45 26 76 46 79 33 72 30 86 6 59 73 31 33 29 33 25 31 60 16 17 12 42 52 74 68 5 62 97 98 9 75 33 27 43 40 75 13 37 35 68 93 11; 67 35 89 11 58 76 52 66 97 18 55 55 53 12 3 34 26 30 75 41 32 88 57 77 45 81 47 4 25 59 59 5 36 31 21 91 2 25 35 97 24 44 89 28 90 74 29 65 34 37 69 42 57 60 27 63 12 92 23 37 17 75 100 71 42 68 32 11 48 77 46 37 53 18 97 93 86 56 87 93 80 87 75 66 51 47 85 54 11 42 37 73 36 33]

$[S_i]_{10x1}$ = [4270, 4748, 4294, 4691, 4208, 4877, 4818, 4334, 4053, 8234]

$[D_j]_{1x94}$ = [260 914 284 618 452 408 17 362 552 123 370 998 776 658 389 511 8 441 171 203 621 751 397 394 521 999 137 909 874 380 831 572 789 566 628 569 183 928 137 788 732 73 379 321 377 811 141 870 1000 847 683 16 389 194 733 828 900 874 177 207 205 881 497 822 896 168 895 357 109 17 732 278 489 950 789 51 977 602 133 281 8 251 831 936 90 521 799 799 783 823 478 385 537 116]

S15:

$[C_{ij}]_{10x94}$ = [73 93 61 63 30 96 77 26 93 99 65 52 47 46
28 82 80 39 20 94 14 71 51 72 51 92 59 13 23
74 97 15 18 85 80 79 58 17 47 77 64 89 61 80
78 44 23 26 88 10 15 84 78 68 93 58 90 35 78
40 98 63 90 30 22 59 97 97 71 64 85 27 40 20
36 7 59 60 78 96 77 97 26 73 68 57 65 3 49
88 21 80 29 30; 96 35 79 30 43 64 9 26 16 4
20 36 71 77 95 44 30 32 37 27 92 52 49 80 97
55 16 50 98 98 69 68 43 99 85 94 72 10 18 27
66 50 58 96 84 95 84 57 37 47 98 88 12 27 70
96 27 85 66 20 43 38 4 21 58 46 67 27 33 60
39 38 95 21 76 65 78 48 12 63 20 1 98 24 38
43 14 1 1 24 74 13 66 24; 66 44 86 24 62 99
98 83 10 65 75 72 69 5 50 87 34 92 11 33 37
20 46 5 31 78 97 66 8 25 96 84 24 52 90 15
77 92 51 32 27 67 31 92 1 31 55 5 86 58 11
16 46 19 68 80 61 54 58 13 21 69 75 28 42 1
72 98 18 29 83 3 74 69 86 75 44 63 40 48 40
97 26 11 10 9 16 40 97 96 69 27 62 76; 60 79
49 48 57 45 58 71 29 47 83 4 3 75 51 32 35
50 74 11 25 58 100 30 46 19 39 14 4 88 29 74
29 39 80 87 90 72 67 57 35 1 86 53 40 86 47
9 46 10 4 97 33 13 30 90 97 87 90 28 62 81
54 33 71 6 35 85 20 16 85 43 20 2 66 19 85
68 30 81 74 51 92 66 84 52 31 33 82 33 56
100 92 42; 24 65 70 28 3 79 72 65 13 75 62 7
6 51 40 7 8 42 93 63 87 29 90 68 27 92 45 4
53 96 19 18 74 94 43 30 35 95 6 64 17 95 66
4 71 89 39 29 24 98 71 32 32 52 52 76 54 22

58 46 41 43 46 72 7 6 45 8 43 59 64 81 61 1
91 34 98 84 62 69 9 36 93 73 46 14 45 83 65
40 80 78 20 25; 44 80 81 96 93 86 91 15 17
32 26 97 3 60 91 97 84 11 7 67 28 52 43 76
42 73 9 43 79 31 74 24 4 46 35 35 16 12 71 3
52 32 72 26 48 67 87 61 86 34 38 36 76 61 80
12 80 18 35 37 10 38 38 90 55 30 11 40 69 94
52 52 98 13 4 27 100 100 73 92 42 33 48 32 9
4 20 66 78 4 35 81 6 84; 13 97 10 2 33 4 11
35 77 89 100 50 82 78 65 47 97 89 3 20 99 19
95 14 68 99 44 99 26 89 22 10 98 61 28 12 94
14 23 27 22 54 59 84 14 46 51 83 8 53 53 77
56 43 68 3 44 6 32 6 80 12 60 1 97 87 37 37
97 99 70 78 50 96 81 39 75 31 87 7 71 32 10
52 8 12 51 16 23 54 74 44 36 98; 47 45 77 63
43 34 54 59 81 59 66 52 7 74 23 65 29 97 98
70 56 42 86 82 83 14 16 19 78 95 31 31 17 67
22 95 34 82 24 11 48 60 55 25 90 43 31 65 45
55 31 43 34 96 51 19 1 59 84 68 49 28 86 57
93 45 10 24 6 81 82 7 20 96 83 29 17 10 8 12
40 2 46 16 61 5 56 84 47 11 83 1 16 36; 46 4
18 85 86 30 97 1 84 15 39 13 73 86 100 80 76
1 69 32 92 92 54 63 85 90 55 58 36 65 63 7
37 53 71 66 86 70 95 78 10 94 100 82 43 67
49 92 52 30 55 49 93 39 44 94 36 33 44 88 90
84 69 74 81 36 3 65 79 40 41 17 75 16 24 41
93 77 32 51 50 83 72 51 50 24 24 38 47 81 76
56 51 29; 75 16 79 6 81 36 100 54 90 80 45
41 55 32 68 47 4 10 89 31 43 94 23 55 12 48
43 23 20 56 60 15 84 29 78 75 90 34 96 2 39
41 22 99 86 72 33 47 13 13 82 80 13 18 11 94
55 76 82 35 61 46 75 7 38 29 42 95 13 5 4 86

92 83 67 83 75 53 19 100 79 92 19 11 73 40
13 41 72 6 20 32 94 74]

$[S_i]_{10x1}$ = [4486, 4876, 4297, 4086, 4413, 4557, 4089, 4602, 4839, 6562]

$[D_j]_{1x94}$ = [523 818 818 314 770 517 503 285 740 143
332 905 961 494 800 31 784 963 636 59 5
460 245 451 299 297 157 937 704 903 148
287 723 338 711 603 970 265 429 763 522
387 667 369 430 43 186 584 437 721 189 958
735 282 681 423 291 76 49 494 621 606 271
974 830 937 240 607 170 385 886 483 777
607 699 177 981 141 22 11 733 854 848 7
565 709 238 71 749 512 609 179 473 220]

LAMPIRAN B

Lampiran B. Tiga Puluh Enam Data *Real*

Data 1 :

$[S_i]_{2x1}$ = [281820, 136731]

$[D_j]_{1x94}$ = [3071 4187 1155 3438 6851 3441 5351 20838
2752 6363 2838 2695 37543 6595 2611 0
0 2859 622 2113 4599 6339 249 4727
17334 4391 25700 1519 3029 2207 46729 3084
3922 1381 961 3098 1078 2910 1700 4848
0 0 0 10441 0 4724 0 0
0 0 0 0 0 0 0 9820
0 0 0 0 6215 0 0 0
14221 1503 2137 501 562 6626 1494 5878
5338 611 1117 472 1814 1227 1229 1724
3743 7922 281 689 7822 1307 1012 2395
1150 456 4123 528 21067 33274]

Data 2 :

$[S_i]_{2x1}$ = [302423, 138801]

$[D_j]_{1x94}$ = [2867 3947 1239 3756 8087 3158 5224 17772
2148 5892 2193 1776 35872 7108 2238 0
125 2362 498 1741 3854 5221 373 4321
18436 4279 27869 1469 2959 2539 51084 3028
4303 1334 935 3455 1428 3010 1594 4102
0 0 0 8701 0 3729 0 0
2238 0 1119 0 0 249 0 7955
0 0 0 9447 22497 622 0 0
13045 1314 2018 472 774 6300 1414 6118
4883 556 1080 473 1782 1185 1347 1750

	3616	7777	368	725	8781	1287	1031	2480
	1126	490	4317	469	18950	35143]		
Data 3 :								
$[S_i]_{2x1}$ = [321396, 138801]								
$[D_j]_{1x94}$ = [3503	4328	1365	3049	7179	3435	5100	18589	
	2271	5614	2726	2745	44061	7420	2362	0
	249	2486	498	1865	4102	5718	373	5973
	17038	4142	24243	1477	2877	2577	47969	3236
	3620	1376	863	3409	1338	2523	1697	4351
	125	0	0	9322	0	4102	3232	0
	2362	1989	1243	0	995	249	0	8701
	2611	2238	125	10690	24610	622	498	0
	13654	1515	2171	578	695	6351	1844	5611
	5462	588	1002	547	1551	1281	1434	1710
	3371	7698	331	682	7483	1324	883	2551
	1128	480	4474	543	19452	32342]		
Data 4 :								
$[S_i]_{2x1}$ = [308252, 138801]								
$[D_j]_{1x94}$ = [3852	4759	1312	3451	7537	2827	3878	15224	
	2530	5690	2494	2383	32901	7314	2486	0
	249	2735	498	1989	4475	6091	373	5492
	17978	4305	23799	1597	2773	2556	48509	3279
	4183	1303	795	4024	1381	2597	1666	4848
	125	0	0	10192	0	4475	3356	0
	2611	2984	1368	0	1119	249	0	9447
	2984	2362	125	2984	26599	746	249	0
	14055	1437	2283	556	812	5786	2034	5911
	5026	524	1133	464	1792	1330	1457	1668

	3827	7947	297	759	9074	1436	864	2217
	1160	519	4166	614	19405	30392]		

Data 5 :

$[S_i]_{2x1}$ = [341015, 138738]

$[D_j]_{1x94}$ = [3332	3976	1252	3494	9871	3027	5214	20029
	2437	5584	2615	2184	39154	6641	2735	0
	249	2984	746	2238	4848	6837	373	4677
	17447	4487	24968	1804	3137	2538	50087	3445
	4375	1444	919	4024	1389	2960	1660	5221
	125	0	0	11063	0	4972	3729	125
	2859	5096	1492	0	871	249	125	10129
	3232	2735	249	12554	28961	746	125	0
	12664	1382	2405	479	724	6258	1924	5195
	5217	536	924	385	2053	1111	1340	1712
	3630	8300	321	679	8954	1385	862	2372
	1225	480	4369	556	20814	27658]		

Data 6 :

$[S_i]_{2x1}$ = [280110, 138801]

$[D_j]_{1x94}$ = [3300	4628	1250	2852	8202	2916	3912	16157
	2695	5526	2699	2271	41855	7074	1616	0
	125	1741	373	1243	2859	3854	249	5151
	17117	4368	24291	1560	2679	2952	42223	3585
	4182	1423	838	3282	1491	2324	1553	3108
	125	0	125	6464	0	2859	2238	0
	1616	2921	871	0	498	125	0	6091
	1865	1616	125	2921	17029	498	249	0
	13885	1516	2480	410	799	6616	1980	5900

	4721	584	1032	419	1702	1290	1536	1725
	3455	7792	341	632	9100	1269	897	2333
	1296	452	3957	487	20317	32228]		

Data 7 :

$[S_i]_{2x1}$ = [350910, 142165]

$[D_j]_{1x94}$ = [3512 4300 1291 3477 9394 3031 4597 22500
1776 5384 2824 2530 42638 7341 2735 0
249 2984 746 2238 4848 6837 3737 5858
16806 4210 24717 1715 2549 2520 50440 3185
4359 1437 914 3120 1551 2109 1578 5345
125 0 0 11187 0 4972 3729 125
2859 8452 1492 0 871 249 125 10317
3232 2735 249 8452 29085 746 373 0
13607 1421 2068 554 877 6207 1844 6259
4824 610 1175 403 1840 1110 1389 1739
3546 8113 348 753 8366 1329 970 2121
1181 530 3778 497 19912 34947]

Data 8 :

$[S_i]_{2x1}$ = [359746, 138801]

$[D_j]_{1x94}$ = [3269 4456 1277 2783 8888 3679 4621 18426
2745 5940 2408 2437 48268 5585 3108 0
249 3356 746 2486 5594 7707 373 4944
19086 3758 27005 1884 3139 2275 48013 3411
4479 1386 911 3405 1402 3198 1657 5967
249 0 0 12678 0 5594 4226 125
3232 7955 1616 0 995 373 125 11560

	3605	3108	249	7955	32938	871	249	0
	13500	1377	2189	589	752	5974	1588	6085
	4578	512	1056	452	2240	1138	1268	1818
	3634	8274	300	703	8162	1308	901	2438
	1088	454	4199	613	19521	29812]		

Data 9 :

$[S_i]_{2x1}$ = [342920, 138801]

$[D_j]_{1x94}$ = [2925 4347 1335 3175 7824 3172 4948 23197
2383 5467 2469 2752 39059 6619 3108 0
249 3356 746 2362 5345 7458 373 5506
14992 3845 24351 1794 2936 2539 46157 3356
4609 1455 955 3398 1590 3044 1697 5718
249 0 125 12181 0 5345 4102 125
3108 6588 1616 0 995 373 125 11435
3481 3108 249 6588 32068 871 0 0
13591 1657 2344 487 882 6226 1604 5485
5157 575 1140 492 2117 1075 1231 1751
3340 7952 308 613 8140 1194 809 2573
1121 441 4037 523 20791 30722]

Data 10 :

$[S_i]_{2x1}$ = [341152, 138801]

$[D_j]_{1x94}$ = [3241 4160 1166 3365 9373 3262 4463 19921
2543 6128 2722 2745 31893 7173 3232 0
249 3481 746 2486 5718 7831 373 5434
14628 4201 24296 1562 3014 2643 48925 3661
4064 1300 898 3583 1463 2659 1722 6091
249 0 0 12927 0 5594 4351 0

	3232	6837	1616	0	1119	373	125	11933
	3729	3108	249	6837	33559	871	249	0
	14048	1467	2451	579	707	6182	1680	6488
	5428	554	1167	460	1732	1156	1409	1839
	3380	7633	244	670	8001	1394	1004	2184
	1096	443	4305	576	20644	27659]		

Data 11 :

$[S_i]_{2x1}$ = [342650, 138801]

$[D_j]_{1x94}$ = [3345	4376	1200	3449	9943	3013	5472	19127
	2461	5592	2870	2383	34883	6884	3108	0
	249	3356	746	2486	5594	7707	373	4617
	13879	4019	28972	1895	3048	2349	51999	3726
	4185	1286	842	3282	1470	2515	1631	5842
	249	0	0	12430	0	5594	4226	0
	3232	5345	1616	0	1119	373	125	11560
	3605	3108	249	5345	32689	871	249	0
	12663	1482	2066	537	706	6570	1878	5523
	4546	540	1023	553	1875	1145	1302	1718
	3625	8131	294	762	7981	1439	918	2211
	1191	498	4421	577	19509	29608]		

Data 12 :

$[S_i]_{2x1}$ = [345429, 138803]

$[D_j]_{1x94}$ = [3051	4400	1232	3234	8448	3402	3949	18071
	2483	5441	2088	2543	37374	6698	3108	0
	249	3356	746	2486	5594	7582	373	3245
	13670	3723	25743	1793	3104	2515	49978	3573

	3966	1476	820	3341	1380	2408	1862	5842
	249	0	0	12430	0	5594	4226	125
	3108	8577	1616	0	995	373	125	11560
	3605	3108	249	8577	32567	871	0	0
	13631	1538	2043	475	843	6400	2087	5971
	5627	579	1251	450	2393	1262	1306	1868
	3646	7945	352	690	8525	1378	1028	2216
	1155	477	3972	515	20557	31750]		

Data 13 :

$[S_i]_{2x1}$ = [129248, 61185]

$[D_j]_{1x94}$ = [1535	9221	1912	7522	4198	7248	12387	1622
	6581	13029	2408	4514	3320	379	746	0
	1616	2238	0	2486	125	373	1865	4459
	1369	2163	2276	2455	1595	2338	4434	8503
	3362	2065	1524	3043	2504	2085	2654	0
	0	0	0	498	622	0	0	0
	0	0	0	0	0	8577	4848	995
	0	0	0	0	0	2735	0	0
	539	633	2822	372	927	6876	0	1422
	6177	376	853	555	2391	864	1421	2042
	94	0	129	786	410	926	404	577
	50	533	0	615	6234	976]		

Data 14 :

$[S_i]_{2x1}$ = [129520, 61185]

$[D_j]_{1x94}$ = [1714	10057	1976	7613	4879	6958	10886	1523
	5092	12628	2256	5266	3247	379	622	0
	1243	1865	0	1989	125	249	1368	4504

	1400	1896	2124	2547	1537	1703	4008	8468
	3184	1960	1282	2938	1940	2166	1887	0
	1492	0	3232	373	1989	0	1119	2611
	0	0	871	0	0	8080	3978	871
	4226	0	0	0	0	2238	0	0
	551	565	2446	353	972	6416	0	1524
	6146	428	905	536	2401	943	1518	2006
	103	0	131	744	373	946	393	583
	51	585	0	682	917	928]		

Data 15 :

$[S_i]_{2x1}$ = [137888, 60685]

$[D_j]_{1x94}$ = [1702 9869 1818 7878 3882 7952 9082 1628
5208 13137 2089 4660 3255 394 622 0
1368 1989 0 2238 125 373 1616 4320
1345 2450 2043 2254 1542 2060 4025 7511
3253 2123 1387 2909 2244 2227 2419 0
1741 0 3481 373 2238 0 1368 2735
0 0 871 0 0 8825 4351 746
7085 0 0 0 0 2486 3232 0
581 636 2491 385 824 6856 0 1435
6396 468 855 528 2163 927 1662 2066
101 0 129 783 391 985 384 67
52 616 0 624 761 878]

Data 16 :

$[S_i]_{2x1}$ = [141561, 61185]

$[D_j]_{1x94}$ = [1530 11018 1894 8120 4033 7193 12252 1522
4514 13743 2020 5538 3325 387 622 0
1492 2113 0 2238 125 373 1741 3806
1414 2294 2162 2887 1583 1916 3745 6599
3953 2080 1383 2647 2339 2452 2494 0
1865 0 3729 498 2362 0 1368 2984
0 0 995 0 0 9571 4724 871
7334 0 0 0 0 2611 249 0
596 553 2467 393 915 7353 0 1502
5087 448 860 597 2100 1005 1520 1910
90 0 135 930 362 1156 392 579
48 501 0 920 830 789]

Data 17 :

$[S_i]_{2x1}$ = [149360, 61185]

$[D_j]_{1x94}$ = [1783 8268 1949 7619 4300 7369 10524 1502
5266 13282 2660 6025 3668 415 746 0
1616 2362 0 2486 125 373 1989 4821
1481 1843 2474 2446 1231 2230 3822 7465
3260 2248 1165 2667 1888 2339 2336 0
1989 0 4102 498 2486 0 1492 3232
0 0 1119 0 0 10317 5096 995
9944 0 0 0 0 2859 3978 0
546 608 2595 351 870 7061 0 1519
6202 471 799 441 2069 895 1571 2308
97 0 116 809 340 978 382 581
50 508 0 624 783 821]

Data 18 :

$[S_i]_{2x1}$ = [125861, 61185]

$[D_j]_{1x94}$ = [1756 9831 1780 8545 4083 7247 10811 1528
4660 13286 2575 6267 3413 376 373 0
995 1368 0 1368 125 249 1243 4553
1486 1894 1766 2229 1456 1904 4411 7142
3554 2111 1375 2692 2232 2136 2415 0
1119 0 2486 249 1492 0 871 1989
0 0 622 0 0 6091 2984 498
5718 0 0 0 0 1741 125 0
590 564 2752 389 879 7174 0 1469
7057 409 890 578 1929 1014 1520 1843
95 0 146 875 379 1231 432 569
47 563 0 620 910 902]

Data 19 :

$[S_i]_{2x1}$ = [143718, 61185]

$[D_j]_{1x94}$ = [1542 9673 1878 1051 4028 7799 10965 1524
5538 11922 2453 6175 3457 348 746 0
1616 2362 0 2486 125 373 1989 4521
1171 2066 2033 2581 1650 2159 4059 7890
3743 2020 1348 2642 2010 2342 2079 0
1989 0 4226 498 2611 0 1492 3356
0 0 1119 0 0 10441 5096 995
9944 0 0 0 0 2859 3481 0
600 689 2811 370 918 7128 0 1417
6390 401 760 523 1968 874 1464 1962
95 0 123 850 351 1087 343 630
47 532 0 609 721 769]

Data 20 :

$[S_i]_{2x1}$ = [151684, 61185]

$[D_j]_{1x94}$ = [1501 9980 1750 7823 4029 7132 11428 1626
6025 12791 1991 5013 343 423 746 0
1865 2611 0 2859 125 373 2238 4221
1478 2095 1623 2766 1326 2280 4385 7552
3005 2428 1300 2840 1954 2241 2595 0
2238 0 4724 498 2859 0 1492 3854
0 0 1243 0 0 11808 5842 1119
10441 0 0 0 0 3232 3356 125
541 517 2288 418 859 6905 0 1470
5493 386 841 425 2378 814 1781 2058
90 0 134 700 386 1011 358 628
49 490 0 688 768 801]

Data 21 :

$[S_i]_{2x1}$ = [153950, 61185]

$[D_j]_{1x94}$ = [1635 10229 1887 8534 4323 7292 10486 1619
6267 13118 2427 6581 3087 353 746 0
1865 2611 0 2735 125 373 2238 3842
1491 2263 2121 2360 1654 2256 4318 6296
3473 2068 1346 2645 1802 2343 2321 0
2238 0 4475 498 2859 0 1616 3605
0 0 1243 0 0 11435 5594 1119
10317 0 0 0 0 3108 1741 125
540 580 2632 429 856 7270 0 1387
5321 378 818 565 2199 1042 1531 2092
98 0 128 749 375 1038 423 555
53 597 0 647 840 889]

Data 22 :

$[S_i]_{2x1}$ = [155788, 61185]

$[D_j]_{1x94}$ = [1627 9610 1954 7899 4539 7615 10235 1633
6175 13174 1815 5092 3539 369 746 0
1865 2735 0 2984 125 498 2238 3772
1368 2207 2554 2789 1713 2247 4414 7827
3688 1767 1439 2892 1956 2147 2303 0
2362 0 4848 498 2984 0 1492 3854
0 0 1243 0 0 11933 5842 1119
10441 0 0 0 0 3356 2362 125
572 580 2207 353 869 6977 0 1314
5111 430 862 386 2208 884 1648 2098
91 0 128 894 341 1137 385 600
41 563 0 624 866 795]

Data 23 :

$[S_i]_{2x1}$ = [149635, 61185]

$[D_j]_{1x94}$ = [1802 9748 1967 8129 4418 7132 10291 1494
5013 11953 1893 5208 3168 394 746 0
1865 2611 0 2735 125 373 2238 4092
1565 2173 2132 2445 1577 2079 3997 6962
3591 1978 1448 2844 2179 1930 2536 0
2238 0 4599 498 2859 0 1616 3605
0 0 1243 0 0 11808 5718 1243
10441 0 0 0 0 3232 746 125
596 565 2531 382 842 7113 0 1452
5170 395 907 508 2152 1013 1445 2305

	108	0	117	885	361	1096	375	681
	52	546	0	687	799	935]		

Data 24 :

$[S_i]_{2x1}$ = [154856, 61185]

$[D_j]_{1x94}$ = [1700	11029	1964	6700	3928	7280	10653	1631
5979	12142	2819	5538	2677	391	746	0
1865	2611	0	2735	125	373	2238	4157
1405	2164	2248	2504	1435	2104	4126	7504
3133	2227	1431	2859	2221	2335	2467	0
2238	0	4599	498	2859	0	1616	3605
0	0	1243	0	0	11684	5718	1243
10317	0	0	0	0	3232	3978	125
625	600	2220	387	950	6913	0	1378
6151	405	784	587	2105	841	1503	2116
106	0	149	726	338	1012	354	482
47	544	0	676	810	833]		

Data 25 :

$[S_i]_{2x1}$ = [101397, 50000]

$[D_j]_{1x94}$ = [1052	4082	0	3198	2619	2695	4544	890
2203	5681	0	2656	1931	1089	746	0
995	2611	0	1741	622	1741	0	742
1350	1064	0	944	0	2288	3086	0
11171	0	0	0	0	0	0	0
0	0	0	373	0	1368	0	0
10814	0	0	0	0	0	0	2859
0	0	0	0	9447	0	0	0
910	0	2758	0	971	6627	1461	2487

	5696	0	0	492	2353	0	1501	1843
	720	6306	0	742	439	0	0	971
	222	1605	10442	1980	2767	11502]		

Data 26 :

$[S_i]_{2x1}$ = [111342, 50000]

$[D_j]_{1x94}$ = [1036	4311	0	2775	2853	2897	4072	876
	2303	5068	0	2245	1966	1114	622	0
	1119	2113	0	1368	622	3481	746	844
	1408	1287	0	914	0	2212	2975	0
	10481	0	0	0	0	0	0	0
	0	0	0	373	0	1243	0	0
	11435	0	0	0	0	0	0	2238
	0	0	0	9074	11808	0	0	0
	758	0	2373	0	917	6861	1380	2345
	6032	0	0	477	2059	0	1476	1931
	824	6248	0	752	468	0	0	924
	193	1651	8454	1935	3615	11790]		

Data 27 :

$[S_i]_{2x1}$ = [122554, 50000]

$[D_j]_{1x94}$ = [1067	4285	0	2729	2793	2769	4491	879
	2628	5357	0	2246	1978	1143	622	0
	1368	2238	0	1492	622	3854	871	808
	1283	1134	0	881	0	2398	3474	0
	10945	0	0	0	0	0	0	0
	0	0	0	373	0	1368	0	0
	12678	0	0	0	0	0	0	2486

0	0	0	16407	13051	0	0	0
897	0	2936	0	889	6352	1348	1899
5911	0	0	436	2248	0	1346	1829
686	6361	0	825	432	0	0	939
183	1758	8734	1933	3509	10385]		

Data 28 :

$[S_i]_{2x1}$ = [125581, 50000]

$[D_j]_{1x94}$ = [1031 4231 0 3228 2620 3075 4360 854
2267 5716 0 2244 1952 1236 746 0
1368 2362 0 1616 622 4226 871 664
1562 1326 0 1182 0 2289 3361 0
10978 0 0 0 0 0 0 0
0 0 0 373 0 1243 0 0
13548 0 0 0 0 0 0 2735
0 0 0 13673 14046 0 0 0
807 0 2713 0 902 7731 1209 1925
5406 0 0 533 2174 0 1618 2223
694 6052 0 734 439 0 0 983
157 1644 10304 2015 3783 9930]

Data 29 :

$[S_i]_{2x1}$ = [137824, 50000]

$[D_j]_{1x94}$ = [1113 4680 0 2966 2737 2928 5080 891
2627 4968 0 2265 1869 1218 746 0
1492 2611 0 1741 746 4475 995 882
1535 1247 0 844 0 2233 2930 0
11025 0 0 0 0 0 0 0
0 0 0 498 0 1492 0 0

14916	0	0	0	0	0	0	2859
0	0	0	21627	15164	0	0	0
863	0	2520	0	1112	7022	1158	1966
5700	0	0	476	1921	0	1577	1824
756	6473	0	772	461	0	0	980
201	1719	10603	1801	3034	11485]		

Data 30 :

$[S_i]_{2x1}$ = [104256, 50000]

$[D_j]_{1x94}$ = [949 3693 0 3530 2745 2937 4593 856
2656 5520 0 1818 1785 1178 498 0
871 1616 0 995 373 2735 622 812
1321 1169 0 940 0 2219 2838 0
11123 0 0 0 0 0 0 0
0 0 0 249 0 746 0 0
8825 0 0 0 0 0 0 1741
0 0 0 9571 9198 0 0 0
788 0 3090 0 972 6735 1358 2158
5425 0 0 540 2177 0 1337 2149
685 6461 0 792 426 0 0 907
189 1680 8489 2058 3299 11819]

Data 31 :

$[S_i]_{2x1}$ = [135272, 50000]

$[D_j]_{1x94}$ = [1138 4467 0 3256 2757 3241 4391 867
2245 4755 0 2644 1801 1161 746 0
1492 2611 0 1741 746 4475 995 700
1299 945 0 1058 0 2141 3041 0

11228	0	0	0	0	0	0	0
0	0	0	498	0	1492	0	0
15040	0	0	0	0	0	0	2984
0	0	0	21255	15288	0	0	0
762	0	2543	0	773	6702	1213	2942
5586	0	0	497	2281	0	1300	2054
653	5862	0	741	480	0	0	916
213	1666	9403	1980	3584	10623]		

Data 32 :

$[S_i]_{2x1}$ = [143345, 50000]

$[D_j]_{1x94}$ = [1037 4269 0 3350 2651 3329 4734 871
2246 4876 0 2203 1751 1096 871 0
1741 3108 0 1989 746 5221 1119 862
1620 1293 0 862 0 2434 3035 0
10604 0 0 0 0 0 0 0
0 0 0 498 0 1616 0 0
17029 0 0 0 0 0 0 3356
0 0 0 21752 17277 0 0 0
744 0 2800 0 812 7404 1189 2385
5419 0 0 481 1936 0 1400 2514
672 6083 0 836 500 0 0 977
209 1753 9495 1857 3766 10667]

Data 33 :

$[S_i]_{2x1}$ = [138899, 50000]

$[D_j]_{1x94}$ = [973 4133 0 3472 2759 3126 4393 908
2244 5684 0 2303 1921 1211 871 0
1741 2984 0 1989 746 4972 1119 837

1525	985	0	927	0	2368	3506	0
11034	0	0	0	0	0	0	0
0	0	0	498	0	1243	0	0
16407	0	0	0	0	0	0	3232
0	0	0	20136	16904	0	0	0
703	0	2406	0	995	6630	1360	2439
5920	0	0	505	1908	0	1485	1847
654	5417	0	690	454	0	0	973
188	1715	9570	1792	3181	10916]		

Data 34 :

$[S_i]_{2x1}$ = [143909, 50000]

$[D_j]_{1x94}$ = [1077 4124 0 3047 2550 3222 4681 888
2265 5237 0 2628 1735 1197 871 0
1741 3108 0 2113 746 5221 1119 830
1405 1222 0 931 0 2126 2819 0
10201 0 0 0 0 0 0 0
0 0 0 498 0 1492 0 0
17153 0 0 0 0 0 0 3481
0 0 0 21130 17650 0 0 0
892 0 2915 0 758 6705 1468 2305
5847 0 0 569 2418 0 1542 2023
704 6159 0 750 475 0 0 851
195 1605 11166 1826 3576 10652]

Data 35 :

$[S_i]_{2x1}$ = [138211, 50000]

$[D_j]_{1x94}$ = [1063 4083 0 3221 2781 3488 4573 902
1818 5392 0 2267 1717 1090 871 0
1741 3108 0 1989 746 5221 1119 833
1554 1168 0 927 0 2406 3045 0
10474 0 0 0 0 0 0 0
0 0 0 498 0 1616 0 0
16904 0 0 0 0 0 0 3232
0 0 0 18644 17153 0 0 0
902 0 2668 0 932 6763 1381 2229
6227 0 0 551 2077 0 1303 1561
670 6062 0 788 482 0 0 925
208 1750 10247 2174 3062 9605]

Data 36 :

$[S_i]_{2x1}$ = [146450, 50000]

$[D_j]_{1x94}$ = [984 4223 0 3298 2807 3055 4645 892
2644 5228 0 2627 1980 1105 871 0
1741 3108 0 1989 746 5096 1119 851
1396 1287 0 990 0 2318 3640 0
11412 0 0 0 0 0 0 0
0 0 0 498 0 1616 0 0
16904 0 0 0 0 0 0 3232
0 0 0 22249 17153 0 0 0
741 0 2621 0 1029 7314 1489 1680
6520 0 0 524 2097 0 1272 1695
671 6391 0 831 514 0 0 955
205 1736 10682 1873 3200 10706]

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Aisyah Muswar, lahir di Ujung Pandang pada tanggal 6 Agustus 1998. Penulis menempuh pendidikan di SD Negeri 1 Palatiga (2004-2010), SMP Negeri 1 Baubau (2010-2013), SMA Negeri Baubau (2013-2016) dan saat ini sedang menempuh pendidikan S1 di Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Aktif di berbagai kegiatan di kampus baik akademik maupun non akademik seperti Schematics, INOCHI, KFEST, UKM IFLS, hingga Badan Eksekutif Mahasiswa Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas. Komunikasi dengan penulis dapat melalui email: aisyahmuswar6@gmail.com.