TUGAS AKHIR – KM184801

PERANCANGAN ALGORITMA UNTUK PEMROGRAMAN LINEAR DENGAN BANYAK KENDALA

FACHRI FIRMANSYAH HUTAGALUNG
06111640000094

Dosen Pembimbing
Drs. Soetrisno, M.I.Komp.
Muhammad Luthfi Shahab, S.Si.,M.Si.

Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2020

FINAL PROJECT – KM184801

DESIGNING AN ALGORITHM FOR LINEAR PROGRAMMING WITH MANY CONSTRAINTS

FACHRI FIRMANSYAH HUTAGALUNG
06111640000094

Supervisors :
Drs. Soetrisno, M.I.Komp.
Muhammad Luthfi Shahab, S.Si.,M.Si.

DEPARTMENT OF MATHEMATICS
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya
2020

PERANCANGAN ALGORITMA UNTUK PEMROGRAMAN LINEAR DENGAN BANYAK KENDALA

DESIGNING AN ALGORITHM FOR LINEAR PROGRAMMING WITH CONSTRAINTS

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

FACHRI FIRMANSYAH HUTAGALUNG
NRP. 06111640000094

Menyetujui,

Dosen Pembimbing II,	Dosen Pembimbing I,
Muhammad Luthfi Shahab, S.Si., M.Si.	Drs. Soetrisno, M.I.Komp.
NIP. 19950331 201803 1 001	NIP. 19571103 198603 1 003

Mengetahui,

Kepala Departemen Matematika
FSAD ITS

Subchan, Ph.D.
NIP. 19710513 199702 1 001

Surabaya, Agustus 2020

PERANCANGAN ALGORITMA UNTUK PEMROGRAMAN LINEAR DENGAN BANYAK KENDALA

Nama Mahasiswa	: Fachri Firmansyah Hutagalung
NRP	: 06111640000094
Departemen	: Matematika
Dosen Pembimbing	: 1. Drs. Soetrisno, M.I.Komp.
	2. Muhammad Luthfi Shahab,S.Si.,M.Si.

Abstrak

Pemrograman linear merupakan suatu metode atau cara untuk mendapatkan nilai optimal dari suatu fungsi yang memenuhi semua kendala yang diberikan. Pemrograman linear merupakan salah satu metode yang sering dibahas dalam mathematical optimization dan riset operasi. Dalam Tugas Akhir ini dirancang sebuah algoritma baru (yang disebut sebagai Algoritma "X") yang dapat digunakan untuk menyelesaikan pemrograman linear, khususnya yang memiliki banyak kendala. Karena belum diketahui perumusan dan kemampuan dari Algoritma "X" maka dimulai dengan sedikit variabel. Sebagai pembanding, digunakan Metode Simplex yang merupakan metode umum dalam penyelesaian pemrograman linear yang dirancang dalam bentuk suatu algoritma. Untuk melakukan perbandingan Algoritma "X" dengan Algoritma Simplex dilakukan pembuatan fungsi pembangkitan kendala. Perbandingan dilihat dari sisi hasil optimal dan waktu komputasi (running time). Hasilnya, Algoritma "X" memiliki waktu yang sama dengan Algoritma Simplex untuk 2 variabel dan memiliki waktu yang lebih lambat dari Algoritma Simplex untuk 3 variabel. Algoritma "X" memiliki hasil yang optimal.

Kata Kunci: Algoritma "X", Pemrograman Linear, Metode Simplex, Hasil Optimal, Waktu Komputasi.

DESIGNING AN ALGORITHM FOR LINEAR PROGRAMMING WITH MANY CONSTRAINTS

Name : Fachri Firmansyah Hutagalung
NRP : 06111640000094
Departement : Mathematics
Supervisors : 1. Drs. Soetrisno, M.I.Komp.
2. Muhammad Luthfi Shahab,S.Si., M.Si.

Abstract

Linear programming is a problem to obtain the optimal value of a function that satisfies all the constraints given. Linear programming is one of the problems that is often discussed in mathematical optimization and operations research. In this study, we design a new algorithm (who called an Algorithm "X") to solve linear programming with many constraints. We start with a few variables since the formulation is not known yet . As a comparison, we use a method that is common in linear programming problems, we know as the Simplex Method. To do a comparisons between "X" Algorithm with Simplex Algorithm, we creat a constraint generator function. The comparison can be seen from optimization results and execution time (running time). In conlusion,"X" Algorithm has the same running time as the Simplex Algorithm for 2 variables and has a slower running time than the Simplex Algorithm for 3 variables. Algorithm "X" has an optimal result.

Keyword: Algorithm X, Linear Programming, Simplex Algorithm, Optimization Results, Computational Time.

KATA PENGANTAR

Puji syukur penulis ucapkan kehadirat Allah SWT atas rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Perancangan Algoritma untuk Pemrograman Linear dengan Sedikit Variabel dan Banyak Kendala" yang merupakan salah satu prasyarat akademis dalam menyelesaikan Program Sarjana Departemen Matematika FSAD Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis ingin mnegucapkan terimakasih dan penghargaan kepada:

1. Bapak, Ibu, Adik dan seluruh keluarga penulis yang tidak hentinya memberi dukungan secara moril dan materil untuk kesuksesan penulis.
2. Subchan, Ph.D. selaku Kepala Departemen Matematika FSAD ITS yang telah memberikan dukungan dan motivasi selama perkuliahan hingga selesainya Tugas Akhir ini.
3. Drs. Soetrisno, M.I.Komp., Muhammad Luthfi Shahab S.Si, M.Si. selaku Dosen Pembimbing yang telah memberikan bimbingan, arahan, dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini hingga dapat selesai dengan baik.
4. Dr. Valeriana Lukitosari, S.Si., MT. ; Moh. Iqbal, S.Si., M.Si. dan Drs. Nurul Hidayat, M.Kom. selaku Dosen Penguji yang telah memberikan bimbingan, arahan, dan saran kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Dr. Budi Setiyono, S.Si.,MT. selaku Dosen Wali yang telah memberikan dukungan dan motivasi selama perkuliahan hingga selesainya Tugas Akhir.
6. Seluruh Bapak dan Ibu dosen Departemen Matematikan ITS atas ilmu dan motivasi yang diberikan kepada penulis selama perkuliahan.
7. Seluruh Staf Departemen Matematika ITS yang telah memberikan pelayanan terbaik kepada penulis selama perkuliahan hingga selesai.

8. Sahabat penulis selama perkuliahan, Irfan Adli, M. Rizal Fanani, Ardyan Chandra, Dimas Ari, Fityan Azizi, Ahmad Ulul Albab, M. Ardi Gunawan dan Praditya Lutvi yang telah memberikan semangat, motivasi dan masukan lainnya saat penulis berada dibawah tekanan selama perkuliahan.
9. Teman-teman Matematika ITS 2016 "LEMNISCATE" yang telah memberikan banyak cerita selama perkuliahan dan banyak pihak yang tidak dapat ditulis satu persatu oleh penulis yang telah membantu selama penulisan Tugas Akhir ini.
10. Teman-teman dari Tim Futsal Matematika ITS dari angkatan 2012 sampai 2018 yang telah berjuang bersama pada IFC 2017, 2018 hingga menjadi Runner-Up di IFC 2019, Semoga Tim Futsal Matematika ITS dapat mengharumkan nama Matematika ITS di IFC berikutnya.

Penulis menyadari sepenuhnya bahwa dalam penyusunan Tugas Akhir ini masih terdapat kekurangan, sehingga penulis mengharapkan kritik dan saran dari semua pihak demi kesempurnaan Tugas Akhir ini. Semoga Tugas Ahir ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, 23 Agustus 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang dalam penulisan Tugas Akhir, rumusan masalah, batasan masalah, tujuan dan manfaat dari penulisan Tugas Akhir. Berikut dibawah ini merupakan sub-bab beserta penjelasannya.

1.1. Latar Belakang

Di era globalisasi seperti saat ini, cukup banyak perusahaan – perusahaan yang mengalami perkembangan yang pesat, sehingga mereka juga dengan mudah dan cepat menjadi perusahaan yang besar. Produktivitas yang tinggi menyebabkan tingkat produksi yang sama dapat dicapai dengan biaya yang lebih rendah. Pada dasarnya, produktivitas dan biaya mempunyai hubungan terbalik. Apabila produktivitas semakin tinggi, maka biaya produksi akan semakin rendah. Begitu pula sebaliknya. Hal tersebut terjadi karena semua faktor produksi merupakan variabel, yang artinya biaya produksi dapat disesuaikan dengan tingkat produksi. Dalam jangka panjang, perusahaan akan lebih mudah meningkatkan produktivitas dibanding dalam jangka pendek [3].

Diperlukan pengolahan yang profesional untuk memperoleh keuntungan yang maksimum dengan meminimumkan biaya produksi. Pengolahan tersebut merupakan persoalan dalam Riset Operasi yaitu Optimalisasi. Dalam proses pengoptimalisasian terdapat metode - metode yang digunakan untuk mendapatkan solusi yang optimal, dengan memperhatikan sumber daya yang menjadi batasan – batasan untuk memperoleh keuntungan yang maksimum dengan biaya produksi yang minimum.

Pada umumnya, metode-metode yang dikembangkan untuk memecahkan suatu masalah pada program linear ditujukan untuk

mencari solusi dari beberapa alternatif solusi yang dibentuk untuk persamaan-persamaan pembatas, sehingga diperoleh nilai fungsi tujuan yang optimal. Terdapat dua cara yang dapat digunakan untuk menyelesaikan persoalan program linear, yaitu dengan cara Grafis dan Metode Simplex. Metode Simplex merupakan teknik yang paling berhasil dikembangkan untuk memecahkan persoalan program linear yang mempunyai jumlah variabel keputusan dan pembatas yang besar [1].

Seiring berkembangnya zaman menjadikan teknologi sebagai hal yang cukup penting dalam penelitian. Salah satunya adalah bahasa pemorgraman. Bahasa pemrograman semakin banyak dipelajari oleh sekumpulan orang. Dalam membuat pemrograman, dibutuhkan algoritma – algoritma yang benar, yang nantinya akan diubah ke dalam suatu bentuk bahasa pemrograman tertentu. Algoritma adalah suatu langkah – langkah yang terurut sehingga mendapatkan hasil yang diinginkan. Masing – masing algortima mempunyai kelebihan dan kekurangannya masing – masing. Efisiensi algoritma menjadi salah satu alasan digunakannya algoritma tersebut. Dalam mencari efisiensi algoritma dibutuhkan kompleksitas algoritma. Kompleksitas algoritma merupakan suatu ukuran banyaknya komputasi yang dibutuhkan algoritma tersebut untuk mendapat hasil yang diinginkan. Algoritma yang mendapatkan hasil yang diinginkan dalam waktu lama memiliki nilai kompleksitas yang tinggi, sementara algoritma yang mendapatkan hasil yang diinginkan dalam waktu cepat maka memiliki nilai kompleksitas yang rendah [2].

Dalam membandingkan dua algoritma untuk mencari algoritma yang efisien, Kusumawati membandingkan Metode Simplex dengan Revised Simplex. Hasilnya, didapatkan kesimpulan bahwa Metode Simplex lebih efisien jika dibandingkan dengan Revised Simplex dengan meninjau lamanya waktu yang dibutuhkan selama proses penyelesaian program linear [4]

Kemudian, A.L. Soyster et. all. melakukan penelitian mengenai *Zero-one Programming* dengan banyak variabel dan sedikit *constraints* serta mengabaikan waktu komputasi. Oleh karena itu, dalam penelitian ini penulis membuat sebuah algoritma baru (sebut saja Algoritma "X") untuk pemrograman linear. Karena belum diketahui perumusan dan kemampuan dari Algoritma "X" maka diawali dengan sedikit variabel terlebih dahulu, dan digunakan banyak kendala yang memungkinkan (dengan hipotesa Algoritma "X" mampu menyelesaikannya dengan cepat) yang dibandingkan dengan metode yang telah umum digunakan dalam pemrograman linear yaitu Metode Simplex yang dirancang dalam bentuk algoritma. Perbandingan dilihat dari hasil optimal dan *running time*.

1.2. Rumusan Masalah

Rumusan masalah pada pembahasan ini adalah :

1. Bagaimana perumusan Algoritma "X" agar dapat digunakan untuk pemrograman linear dengan sedikit variabel dan banyak kendala?
2. Bagaimana perbandingan hasil optimal dan waktu komputasi (*running time*) Algoritma "X" dan Algoritma Simplex untuk menyelesaikan permasalahan dalam pemrogramman linear dengan sedikit variabel dan banyak kendala?

1.3. Batasan Masalah

Batasan masalah pada pembahasan ini adalah :

1. Banyak variabel yang digunakan adalah dua atau tiga variabel.
2. Banyaknya kendala yang diuji memungkinkan untuk diaplikasikan (*feasible*).

1.4. Tujuan Penelitian

Tujuan dalam penelitian ini adalah :

1. Mendapatkan perumusan suatu algoritma baru (Algoritma "X") yang dapat digunakan untuk pemrograman linear dengan sedikit variabel dan banyak kendala.
2. Mendapatkan perbandingan hasil optimal dan waktu komputasi (*running time*) Algoritma "X" dan Algoritma Simplex untuk menyelesaikan permasalahan dalam pemrograman linear dengan sedikit variabel dan banyak kendala.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini :

1. Algoritma yang telah dibuat dapat dimanfaatkan untuk penelitian selanjutnya dalam menyelesaikan permasalahan pemrogaman linear.
2. Hasil perbandingan dapat digunakan peneliti lain sebagai pertimbangan dalam pemilihan algoritma yang tepat untuk menyelesaikan pemrograman linear untuk sedikit variabel dan banyak kendala.

1.6. Sistematika Penulisan Tugas Akhir

Sistematika penulisan dalam laporan Tugas Akhir ini adalah sebagai berikut:

1. BAB I : PENDAHULUAN
   Bab ini menjelaskan latar belakang penyusunan Tugas Akhir, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan laporan Tugas Akhir.
2. BAB II : TINJAUAN PUSTAKA
   Bab ini menjelaskan tentang *linear programming*, Metode Simplex, *sorting* dan kompleksitas algoritma.

3. BAB III : METODOLOGI PENELITIAN
   Bab ini menjelaskan tentang tahap-tahap yang dilakukan dalam penyusunan Tugas Akhir.
4. BAB IV : ANALISIS DAN PEMBAHASAN
   Bab ini menjelaskan tentang analisis dan pembahasan untuk mendapatkan hasil optimal dan waktu komputasi (*running time*) dari Algoritma "X" dan Algoritma Simplex.
5. BAB V : PENUTUP
   Bab ini menjelaskan kesimpulan yang diperoleh dari pembahasan masalah pada bab sebelumnya serta saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dijelaskan mengenai teori-teori yang berkaitan dengan penelitian Tugas Akhir ini. Teori-teori yang dibahas yaitu mengenai penelitian terdahulu, *Linear Programming*, Metode Simplex, *Sorting*, dan Komplesitas Algoritma.

2.1. Penelitian Terdahulu

Dalam membandingkan dua algoritma untuk mencari algoritma yang efisien, Kusumawati membandingkan Metode Simplex dengan Revised Simplex. Percobaan dilakukan dengan cara melakukan perulangan sebanyak 5 kali untuk masing masing data. Variabel terbanyak yang digunakan sebanyak 10 variabel dengan fungsi kendalanya sebanyak 4, dan Fungsi Kendala terbanyak yang digunakan sebanyak 7 fungsi dengan variabel sebanyak 9. Hasilnya, didapatkan kesimpulan bahwa Metode Simplex lebih efisien jika dibandingkan dengan Revised Simplex [4]. Pada tahun 1977 A.L. Soyster et. all. Melakukan peneletian mengenai *Zero-one Programming* dengan banyak variabel dan sedikit kendala. Pada penelitian tersebut mereka mengabaikan waktu komputasi sebagai tolok ukur kinerja karena sangat tergantung pada kemampuan *programmer* serta instalasi komputer tertentu [8].

2.2. Linear Programming

Linear programming merupakan suatu teknik yang dapat membantu dalam proses pengambilan keputusan dalam mengalokasikan sumber daya (mesin, tenaga kerja, uang, waktu, kapasitas gudang, dan bahan baku). *Linear programming* merupakan penggunaan secara luas dari teknik model matematika yang dirancang untuk membantu manajer dalam merencanakan dan mengambil keputusan dalam mengalokasikan sumber daya [5].

Dalam memaksimalkan suatu fungsi objektif atau fungsi tujuan, terdapat titik yang menyebabkan nilai dari suatu fungsi tujuan menjadi optimal. Titik tersebut didapatkan dari perpotongan fungsi kendala yang memenuhi *search space*.

2.2.1. Fungsi Tujuan

Fungsi tujuan disini menggambarkan tujuan di dalam permasalahan pemrograman linear yang berkaitan dengan peraturan secara optimal dari sumber-sumber daya untuk memperoleh keuntungan yang maksimal atau biaya yang minimum[4].

Bentuk umum dari fungsi tujuan :

Maksimumkan/Minimumkan :

$$Z = C_1X_1 + C_2X_2 + C_3X_3 + \cdots + C_nX_n \qquad (1)$$

2.2.2. Fungsi Kendala

Fungsi kendala merupakan penyajian secara matematis atas kendala-kendala kapasitas sumber daya yang tersedia, yang akan dialokasikan secara optimal ke berbagai kegiatan.

Kendala :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &\leq b_n \end{aligned} \qquad (2)$$

Keterangan : (1) bentuk umum fungsi objektif
(2) bentuk umum fungsi kendala

2.3. Metode Simplex

Metode simplex merupakan prosedur algoritma yang digunakan untuk menghitung dan menyimpan banyak angka pada iterasi-iterasi yang sekarang dan untuk pengambilan keputusan pada iterasi berikutnya. Metode Simplex merupakan suatu metode yang digunakan untuk menyelesaikan masalah-masalah pada program linear yang meliputi banyak pertidaksamaan dan banyak variabel. Dalam menggunakan Metode Simplex, untuk menyelesaikan masalah-masalah pada program linear, model program linear harus dirubah terlebih dahulu kedalam suatu bentuk umum yang dinamakan ”bentuk baku”. Ciri-ciri dari bentuk baku model program linear adalah semua kendala berupa persamaan dengan sisi kanan non-negatif, fungsi tujuan dapat memaksimumkan [5].

Contoh :

- Fungsi Tujuan :
  Maksimumkan $Z = 30x + 40y$
- Fungsi Kendala :
  1. $3x + 4y \leq 120$
  2. $y \leq 20$
  3. $2x + 2y \leq 40$
- Fungsi Non-Negatif
  1. $x, y, S_1, S_2, S_3 \geq 0$

$S_1, S_2\ dan\ S_3$ merupakan Slack, terdapat 3 Slack karena terdapat 3 fungsi kendala. Selanjutnya akan dibuat dalam Tabel-2.1.

Tabel-2. 1 Tabel dari Metode Simplex

NB	x	y	S_1	S_2	S_3	Nilai Kanan	Indeks
Z	-30	-40	0	0	0	0	
S_1	3	4	1	0	0	120	
S_2	0	1	0	1	0	20	
S_3	2	2	0	0	1	40	

Setelah membuat tabel-2.1, langkah selanjutnya adalah mencari kolom kunci dengan nominal *Z* yang terkecil yang akan dijadikan sebagai kolom kunci.

Tabel-2. 2 Tabel dari Metode Simplex

NB	x	y	S_1	S_2	S_3	Nilai Kanan	Indeks
Z	-30	-40	0	0	0	0	
S_1	3	4	1	0	0	120	
S_2	0	1	0	1	0	20	
S_3	2	2	0	0	1	40	

Kemudian akan didapatkan kolom *y* sebagai kolom kunci karena memiliki nominal z yang paling kecil yakni sebesar -40. Langkah selanjutnya adalah mencari indeks terkecil dari setiap baris pada kolom *y*.

$$Indeks = \frac{Nilai\ Kanan}{Nilai\ Kolom\ Kunci}$$

Tabel-2. 3 Tabel dari Metode Simplex

NB	x	Y	S_1	S_2		S_3	Nilai Kanan	Indeks
Z	-30	-40	0	0		0	0	
S_1	3	4	1	0		0	120	30
S_2	0	1	0	1		0	20	20
S_3	2	2	0	0		1	40	20

Karena kolom kunci S_2 memiliki nilai indeks yang sama dengan S_3 maka lebih baik diambil S_3 sebagai pivot dari kolom kunci, karena nilai S_2 pada kolom y sudah bernilai 1. Kemudian semua nilai pada baris S_3 akan dibagi dengan pivot yaitu 2. Sehingga didapatkan hasil sebagai berikut :

Tabel-2. 4 Tabel dari Metode Simplex

NB	x	y	S_1	S_2	S_3	Nilai Kanan	Indeks
Z							
S_1							
S_2							
$\boldsymbol{y}$	**1**	**1**	**0**	**0**	**1/2**	**20**	

Selanjutnya, nilai pada kolom baris kunci lainnya akan dibuat nol.

Tabel-2. 5 Tabel dari Metode Simplex

NB	x	y	S_1	S_2	S_3	Nilai Kanan	Indeks
Z	10	0	0	0	20	800	
S_1	-1	0	1	0	-2	40	
S_2	-1	0	0	1	-1/2	0	
y	1	1	0	0	1/2	20	

Selanjutnya, tersisa kolom *x*

Tabel-2. 6 Tabel dari Metode Simplex

NB	x	y	S_1	S_2	S_3	Nilai Kanan	Indeks
Z	10	0	0	0	20	800	
S_1	1	0	1	0	-2	40	40
S_2	-1	0	0	1	-1/2	0	0
y	1	1	0	0	1/2	20	20

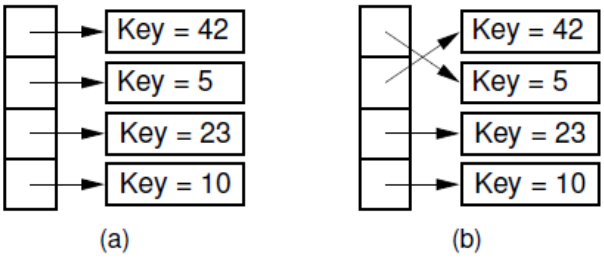
Dapat dilihat pada baris Z (yang berwarna hijau) tidak terdapat nilai yang negatif maka iterasi selesai. Sehingga didapat $x = 0$ dan $y = 20$ dengan $Z_{max} = 800$

2.4. Sorting

Pada umumnya pengurutan atau *sorting* terbagi menjadi dua yaitu *Ascending* (pengurutan dari karakter/angka kecil ke karakter/angka besar) dan *Descending* (pengurutan dari karakter/angka besar ke karakter/angka kecil) [7]. Berikut dibawah ini merupakan beberapa model dari *sorting.*

2.4.1. Selection Sort

Model Sorting :



Gambar 2. 1 Ilustrasi Selection Sort

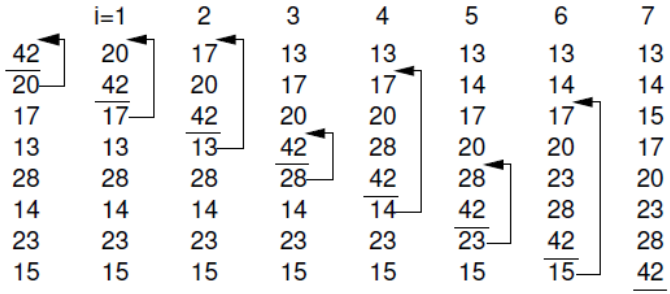
Selection sort merupakan sorting dengan cara menukar posisi seperti pada (a) dan (b), dimana pada (b) 5 ditukar dengan 42 karena lebih kecil dari 42. Berikut adalah implementasi dalam Java:

```
static <E extends Comparable<? super E>>
void selectsort(E[] A) {
  for (int i=0; i<A.length-1; i++) { // Select i'th record
    int lowindex = i;                // Remember its index
    for (int j=A.length-1; j>i; j--) // Find the least value
      if (A[j].compareTo(A[lowindex]) < 0)
        lowindex = j;                // Put it in place
    DSutil.swap(A, i, lowindex);
  }
}
```

Gambar 2. 2 Source Code Selection Sort

2.4.2. Insertion Sort

Model Sorting :



Gambar 2. 3 Ilustrasi Insertion Sort

Gambar 2.3 diatas merupakan sebuah ilustrasi *Insertion Sort*. Setiap kolom menampilkan *array* setelah iterasi dengan mengasumsikan nilai *i* diluar perulangan *for*. Nilai diletakkan diatas garis disetiap kolom yang telah diurutkan.

Berikut adalah implementasi dalam Java. Inputnya berupa sebuah array A dengan elemen sebanyak *n.*

```
static <E extends Comparable<? super E>>
void inssort(E[] A) {
  for (int i=1; i<A.length; i++) // Insert i'th record
    for (int j=i; (j>0) && (A[j].compareTo(A[j-1])<0); j--)
      DSutil.swap(A, j, j-1);
}
```

Gambar 2. 4 Source Code Insertion Sort

2.5. Kompleksitas Algoritma

Algoritma yang efisien adalah algoritma yang meminimukan kebutuhan waktu dan ruang. Kompleksitas Algoritma digunakan untuk menjelaskan model pengukuran waktu dan ruang. Akan tetapi, kebutuhan waktu dan ruang dari suatu algoritma bergantung pada jumlah data yang diproseskan dan algoritma yang digunakan. Jika kompleksitas waktu untuk menjalankan suatu algoritma dinyatakan dengan $T(n)$ dan memenuhi.

$$T(n) \leq C(f(n))$$

Untuk $n \geq n_0$, maka kompleksitas dapat dinyatakan dengan

$$T(n) = O(f(n))$$

BAB III
METODOLOGI PENELITIAN

Pada bab ini dilakukan pembahasan tentang metode penelitian yang digunakan dalam Tugas Akhir agar proses pengerjaan dapat terstruktur dengan baik.

3.1. Langkah – Langkah Penelitian

Adapun langkah – langkah yang dilakukan dalam penelitian ini sampai didapatkan perbandingan hasil optimal dan waktu komputasinya adalah :

1. Studi Literatur

Pencarian referensi yang berkaitan dengan penelitian ini, yaitu meliputi *Linear Programming*, Metode Simplex, *Sorting* dan lainnya yang dibutuhkan untuk pengerjaan penelitian ini.

2. Perumusan Algoritma "X"

Melakukan perumusan Algoritma "X" untuk penyelesaian masalah program linear untuk dua atau tiga variabel dan dengan banyak kendala.

3. Pembuatan program Algoritma "X"

Setelah melakukan perumusan Algoritma "X" untuk pemrogramman linear dengan dua atau tiga variabel, dilakukan perancangan program Algoritma "X" dengan bahasa pemrogramman Java berdasarkan perumusan yang telah dibuat.

4. Pembangkitan Data

Pada tahap ini, akan dilakukan pembangkitan data yang dijadikan sebagai kendala dengan jumlah yang diinginkan, yang akan digunakan untuk penyelesaian dalam program Algoritma "X" dan Algoritma Simplex.

5. *Running* program

Selanjutnya, data yang telah dibuat akan diuji pada program Algoritma "X" sehingga dapat ditinjau hasil optimal yang diperoleh program Algoritma "X".

6. Membandingkan Hasil Optimal dan *Running Time*

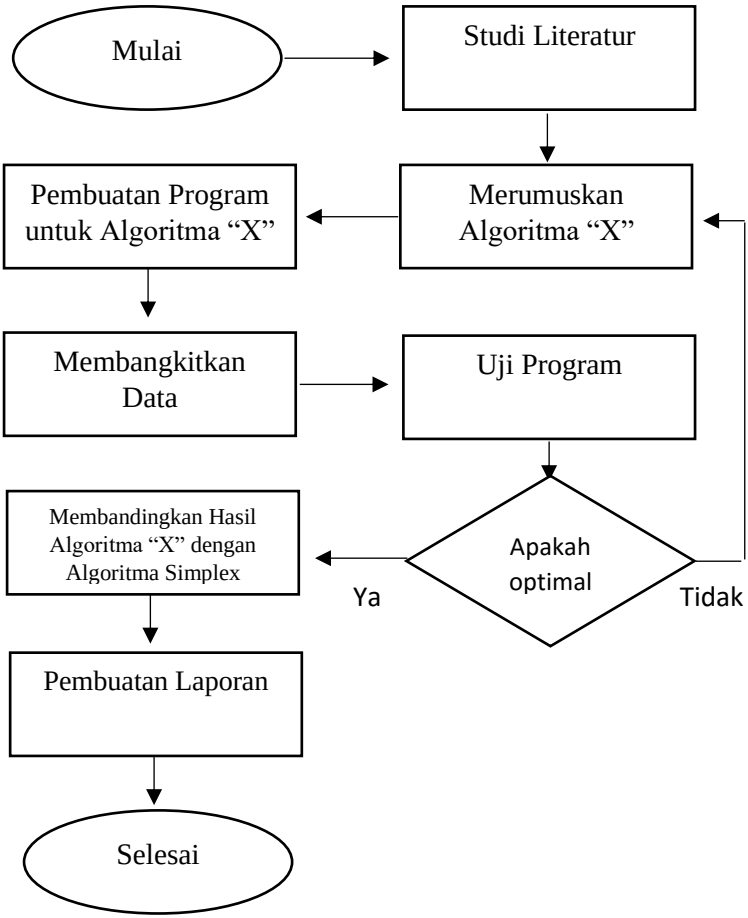
Pada tahap ini, program Algoritma "X" yang sudah diuji, dilakukan perbandingan kinerja program Algoritma "X" dengan Algoritma Simplex dan meninjau hasil optimal yang didapatkan dan *running time* yang dibutuhkan untuk masing-masing program.

7. Pembuatan Laporan

Pembuatan Laporan Tugas Akhir dilakukan dengan memperhatikan pembahasan yang telah diselesaikan pada tahap-tahap sebelumnya. Penulisan dilakukan dengan memperhatikan kaidah penulisan yang berlaku.

3.2. Diagram Alir

Langkah – langkah untuk melakukan penelitian ini secara umum dapat dilihat pada diagram alir Gambar 3.1.



Gambar 3. 1 Diagram Alir

BAB IV
ANALISIS DAN PERANCANGAN

Pada bab ini dilakukan analisis dan pembahasan secara menyeluruh mengenai perancangan Algoritma "X" untuk mendapatkan hasil optimal pada permasahalan pemrograman linear dengan banyak kendala.

4.1. Perumusan Algoritma "X"

Pada sub-bab ini, dijelaskan perumusan untuk Algoritma "X" dalam mencari nilai optimal untuk permasalahan pemrograman linear dengan meninjau fungsi yang ingin dioptimalkan dan fungsi kendala yang diberikan.

4.1.1. Fungsi Linear 2 Variabel

Dalam mencari hasil optimal pada fungsi linear dua variabel, diperlukan titik yang maksimum sehingga didapatkan nilai yang maksimal. Dalam pemrograman linear dua variabel, titik maksimum didapatkan dari perpotongan dua fungsi kendala (*constraint*). Dua fungsi kendala yang dipilih adalah fungsi yang memiliki kemiripan karakteristik (gradien) dengan fungsi objektif yang dimiliki. Fungsi kendala atau *constraint* $ax + by \leq c$ dengan $a, b, c \geq 0$, dalam dua dimensi, suatu batas mempunyai fitur yaitu $\left(\frac{a}{b}\right)$. Dengan fitur tersebut, dapat dicari dua kendala dengan fitur semirip mungkin dengan fitur dari fungsi yang ingin dimaksimalkan yaitu dengan mencari nilai terkecil dari :

$$E = \left|\frac{a_1}{b_1} - \frac{a_2}{b_2}\right|$$

Kemiripan gradien dapat diartikan dalam bentuk interval yaitu ($m_i \leq m_{f_{obektif}} \leq m_{i+1}$). Untuk lebih jelasnya, berikut merupakan langkah-langkah dalam pencarian nilai maksimum menggunakan Algoritma "X" :

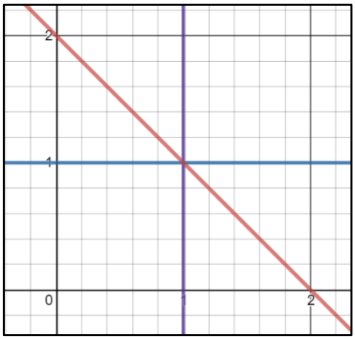
Misal, diberikan 3 fungsi kendala yaitu :

- $x \leq 1$
- $y \leq 1$
- $x + y \leq 2$

1. Analisis fungsi kendala

Pada langkah pertama, dilakukan analisis fungsi kendala, tujuannya adalah untuk menghilangkan fungsi yang berada pada luar domain. Karena fungsi yang berada diluar domain dapat menyebabkan hasil dari Algoritma "X" tidak optimal, berikut penjelasan dalam bentuk grafiknya.

Grafik :



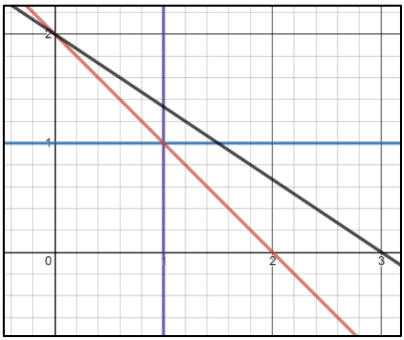
Gambar 4. 1 Grafik Fungsi Kendala berada pada domain

Berdasarkan **Gambar 4.1**, semua fungsi kendala sudah berada di daerah domain, maka semua fungsi digunakan. Misal kendala tersebut dirubah menjadi 4 kendala yaitu :

- $x \leq 1$
- $y \leq 1$
- $x + y \leq 2$
- $2x + 3y \leq 6$

Maka didapatkan grafik sebagai berikut :

Grafik :



Gambar 4. 2 Terdapat Fungsi Kendala diluar domain

Berdasarkan **Gambar 4.2** terdapat fungsi yang berada diluar domain (garis berwarnah hitam) oleh karena itu, garis tersebut harus dihapus terlebih dahulu.

2. Mengurutkan semua fungsi.

Pada tahap ini, dilakukan pengurutan semua fungsi kendala yang diberikan berdasarkan gradien yang dimiliki. Hal ini bertujuan untuk mempermudah mendapatkan dua fungsi yang memiliki kemiripan gradien dengan fungsi objektif yang dimiliki. Fungsi diurutkan berdasarkan gradien terkecil ke besar. Berikut adalah fungsi kendala yang diberikan :

1) $y \leq 1$
2) $x + y \leq 2$
3) $x \leq 1$

Selanjutnya, dilakukan pengurutan fungsi-fungsi berdasarkan gradien yang terkecil ke gradien terbesar, sehingga didapatkan urutan menjadi :

1) $x \leq 1$ $\quad m_1 = -\infty$
2) $x + y \leq 2$ $\quad m_2 = -1$
3) $y \leq 1$ $\quad m_3 = 0$

Diberikan fungsi yang ingin dimaksimalkan adalah $Z = 30x + 20y$ dengan gradiennya ($m_z = -1.5$), maka berlaku, $m_1 \leq m_z \leq m_2$ dimana $m_1 = -\infty$, $m_2 = -1$, $m_z = -1.5$.

3. Menghitung hasil optimal

Berdasarkan interval yang didapatkan pada langkah sebelumnya, terpilih fungsi kendala dengan gradien m_1 dan m_2 yaitu $x \leq 1$ dan $x + y \leq 2$. Tahap selanjutnya dapat dilakukan pencarian titik optimal yaitu dengan melakukan perpotongan antara dua fungsi kendala tersebut yang dapat dilakukan dengan cara eliminasi atau subtitusi. Sehingga didapatkan nilai $x = 1$ dan $y = 1$. Maka nilai maksimalnya adalah : $30(1) + 20(1) = 50$.

4.1.2. Fungsi Linear 3 Variabel

Dalam mencari nilai optimal pada pemrograman linear tiga variabel dilakukan pembagian bidang yaitu (XoY, XoZ, YoZ dan XYZ). Hal tersebut dilakukan karena pada pemrograman linear 3 variabel memiliki 4 kemungkinan titik optimal yaitu pada bidang yang telah disebutkan sebelumnya. Kemudian dilakukan perbandingan nilai optimal yang dihasilkan dari masing – masing bidang tersebut. Dalam mencari nilai optimal pada bidang XoZ, YoZ, dan XoY dapat digunakan Algoritma "X" yang telah dibuat untuk penyelesaian fungsi linear dua variabel, sehingga didapatkan nilai x, y dan z yang maksimal pada masing-masing bidang tersebut. Pada bidang XYZ dilakukan cara yang hampir serupa dengan Algoritma "X" pada pemrograman linear dua variabel. Untuk lebih lanjut, dalam mendapatkan nilai optimal pada fungsi linear tiga variabel, dilakukan langkah-langkah sebagai berikut :

1. Mencari nilai optimal pada bidang XoY, XoZ dan YoZ

Diberikan fungsi kendala sebagai berikut :

1) $2x + y + z \leq 4$
2) $x + 2y + z \leq 4$
3) $x + y + 2z \leq 4$
4) $x + y + z \leq 3$

Dengan fungsi yang ingin dioptimalkan adalah $Q = 3000x + 4000y + 3000z$.

- **Bidang XoY :**

Pada bidang XoY, variabel z akan diabaikan terlebih dahulu atau $z = 0$, sehingga fungsi kendala yang diberikan sebelumnya menjadi :

1) $2x + y \leq 4$
2) $x + 2y \leq 4$
3) $x + y \leq 4$
4) $x + y \leq 3$

Dengan fungsi yang ingin dioptimalkan menjadi $Q_1 = 3000x + 4000y$. Didapatkan fungsi tidak mempengaruhi domain yaitu $x + y \leq 4$ dan $x + y \leq 3$ maka fungsi dikeluarkan dari urutan. Kemudian, dilakukan pengurutan fungsi berdasarkan gradien terkecil ke terbesar dari masing-masing fungsi tersebut sehingga menjadi urutan sebagai berikut:

1) $2x + y \leq 4$ $\quad m_1 = -2$
2) $x + 2y \leq 4$ $\quad m_2 = 0$

Dengan fungsi objektif $Q_1 = 3000x + 4000y$, dengan gradiennya $m_q = -0.75$, sehingga didapatkan interval $m_1 \leq m_q \leq m_2$, maka titik optimal merupakan perpotongan dari fungsi $2x + y \leq 4$ dan $x + 2y \leq 4$, yaitu $x = 1.33$ dan $y = 1.33$ dengan $Q_1 = 3000(1.33) + 4000(1.33) = 9310$

- **Bidang XoZ**

Pada bidang XoZ, nilai y diabaikan atau $y = 0$, sehingga fungsi menjadi

1) $2x + z \leq 4$
2) $x + 2z \leq 4$
3) $x + z \leq 4$
4) $x + z \leq 3$

Dengan fungsi yang dioptimalkan $Q_2 = 3000x + 3000z$, berdasarkan urutan diatas $x + z \leq 3$ dan $x + z \leq 4$ tidak mempengaruhi domain. Setelah itu, dilakukan pengurutan fungsi berdasarkan gradien terkecil ke besar :

1) $2x + z \leq 4 \qquad m_1 = -2$
2) $x + 2z \leq 4 \qquad m_2 = -0.5$

Dengan $Q_2 = 3000x + 3000z$ yang memiliki gradien $m_q = -1$, sehingga didapatkan interval $m_1 \leq m_q \leq m_2$. Maka, titik optimal didapatkan dari perpotongan fungsi $2x + z \leq 4$ dengan $x + 2z < 4$ yaitu $x = 1.33$ dan $z = 1.33$ dengan $Q_2 = 3000(1.33) + 3000(1.33) = 7980$

- **Bidang YoZ**

Pada bidang YoZ, nilai x diabaikan atau $x = 0$, maka fungsi menjadi :

1) $2y + z \leq 4$
2) $y + 2z \leq 4$
3) $y + z \leq 4$
4) $y + z \leq 3$

Dengan fungsi yang ingin dioptimalkan menjadi $Q_3 = 4000y + 3000z$. Pada bidang YoZ memiliki fungsi kendala yang sama dengan bidang XoY dan XoZ. Sehingga dapat disimpulkan titik optimal merupakan perpotongan dari $2y + z \leq 4$ dengan $y +$

$2z \leq 4$ yaitu dengan $y = 1.33$ dan $z = 1.33$ maka $Q_3 = 4000(1.33) + 3000(1.33) = 9310$

2. Mencari nilai optimal pada bidang *XYZ*

Selanjutnya pencarian titik maksimal pada bidang XYZ sehingga didapatkan nilai yang maksimum. Misal, fungsi kendala atau *constraint* $ax + by + cz \leq d$ dengan $a, b, c, d \geq 0$, dalam tiga dimensi XYZ, suatu batas mempunyai fitur yaitu $\frac{a}{b}, \frac{a}{c}, \frac{b}{c}$. Dengan fitur tersebut dapat dicari 3 fungsi kendala yang memiliki fitur semirip mungkin dengan fitur dari fungsi yang ingin dimaksimalkan atau fungsi tujuan. Untuk mencari kemiripan antara fitur dari fungsi kendala dengan fungsi tujuan maka dilakukan dengan menselisihkan masing-masing fitur fungsi kendala dengan fitur fungsi tujuan.

$$E = \left|\frac{a_1}{b_1} - \frac{a_2}{b_2}\right| + \left|\frac{a_1}{c_1} - \frac{a_2}{c_2}\right| + \left|\frac{b_1}{c_1} - \frac{b_2}{c_2}\right|$$

Tiga fungsi kendala yang memiliki nilai E terkecil maka fungsi tersebut yang akan dipilih dan dilakukan perpotongan. Selanjutnya, dilakukan pencarian nilai Q_4 dengan cara seperti diatas. $Q_4 = 3000x + 4000y + 3000z$, dengan *constraint* :

1) $2x + y + z \leq 4$
2) $x + 2y + z \leq 4$
3) $x + y + 2z \leq 4$
4) $x + y + z \leq 3$

Dilakukan penghitungan fitur Q_4 , didapatkan

- $\frac{a_1}{b_1} = 0.75$
- $\frac{a_1}{c_1} = 1$
- $\frac{b_1}{c_1} = 1.33$

Sehingga, rumus E menjadi seperti berikut :

$$E = \left|0.75 - \frac{a_2}{b_2}\right| + \left|1 - \frac{a_2}{c_2}\right| + \left|1.33 - \frac{b_2}{c_2}\right|$$

Untuk $\boldsymbol{2x + y + z \leq 4}$:

$$E = |0.75 - 2| + |1 - 2| + |1.33 - 1| = 2.58$$

Untuk $\boldsymbol{x + 2y + z \leq 4}$:

$$E = |0.75 - 0.5| + |1 - 1| + |1.33 - 2| = 0.92$$

Untuk $\boldsymbol{x + y + 2z \leq 4}$:

$$E = |0.75 - 1| + |1 - 0.5| + |1.33 - 0.5| = 1.58$$

Untuk $\boldsymbol{x + y + z \leq 3}$:

$$E = |0.75 - 1| + |1 - 1| + |1.33 - 1| = 0.58$$

Sehingga terpilih fungsi yang memiliki nilai E terkecil adalah $\boldsymbol{x + 2y + z \leq 4}$, $\boldsymbol{x + y + 2z \leq 4}$ dan $\boldsymbol{x + y + z \leq 3}$. Selanjutnya dilakukan perpotongan antara 3 fungsi tersebut dan didapatkan nilai $x = 1$, $y = 1$, $z = 1$. Maka titik optimal yang didapat pada bidang XYZ adalah (1,1,1). Sehingga$Q_4 = 3000(1) + 4000(1) + 3000(1) = 10000$.

Dari bidang XoY, XoZ, YoZ dan XYZ didapatkan nilai maksimal adalah $Q = 10000$ dengan $x = 1$, $y = 1$ dan $z = 1$.

Secara umum, langkah-langkah Algoritma “X” adalah seperti pada Gambar 4.3. :

Algoritma "X"

Fungsi tujuan $z = c_{z1}x_1 + c_{z2}x_2 + \cdots + c_{zn}x_n$
Fungsi kendala :
$c_{11}x_1 + c_{21}x_2 + \cdots + c_{n1}x_n \leq k_1 , \ldots , c_{1m}x_1 + c_{2m}x_2 + \cdots + c_{nm}x_n \leq k_m$ dengan $n = 2 ,3$ dimana n merepresentasikan banyaknya variabel yang digunakan dan m merepresentasikan banyaknya kendala.
Dilakukan Input pada Algoritma "X" berupa suatu array 2D yang bertipe double yang berisikan fungsi kendala dan fungsi objektif dimana fungsi objektif diletakkan pada posisi akhir dari array.
dilakukan pengurutan fungsi kendala berdasarkan nilai karakteristik (E) yang didapat dengan menggunakan *selection sort*.

$$E = \left(\sum_{i=1}^{n-1} \left(\sum_{j=i+1}^{n} \left| \frac{c_{im}}{c_{jm}} - \frac{c_{z_{im}}}{c_{z_{jm}}} \right| \right) \right)$$

Didapatkan n kendala terkecil yang kemudian dilakukan perpotongan n kendala tersebut
Didapatkan titik dan hasil maksimalnya

Gambar 4. 3 Pseudo-code Algoritma "X"

4.2. Pembuatan Program

Pada sub-bab ini, perumusan dari Algoritma "X" akan dibuat kedalam pemrograman dengan bahasa JAVA.

4.2.1. Fungsi Linear 2 variabel

Dalam program ini digunakan *array* 2 dimensi dengan tipe 'double' sebagai wadah dari koefisien fungsi kendala dan fungsi tujuan, dimana fungsi tujuan diletakkan di posisi akhir dalam *array*. Langkah-langkah pembuatan program yang diperlukan berdasarkan perumusan Algoritma "X" adalah sebagai berikut :

1. Analisis fungsi kendala

Pada sub-bab perumusan algoritma sebelumnya, kita dapat menentukan fungsi kendala yang memenuhi dengan melihat grafiknya. Untuk dapat mengimplementasikan tahap ini kedalam

program, maka ditemukan cara untuk menentukan fungsi kendala yang memenuhi, yaitu sebagai berikut :

1) Mencari fungsi kendala dengan nilai y terkecil saat $x = 0$
2) Mengurutkan fungsi kendala berdasarkan gradien (besar ke kecil)
3) Melakukan pertukaran posisi fungsi kendala dengan nilai y terkecil menjadi urutan pertama. Jika sudah di urutan pertama tidak perlu dilakukan pertukaran.
4) Mencari fungsi perpotongan terdekat dengan syarat $y \geq 0$.

Untuk keseluruhan *Source Code* dapat dilihat pada Lampiran A

2. Algoritma "X"

Pada tahap sebelumnya didapatkan *array* baru yang berisi fungsi kendala yang sudah memenuhi serta fungsi tujuan yang diletakkan di posisi akhir dalam *array*. Kemudian langkah selanjutnya dilakukan penyelesaian menggunakan program Algoritma "X". Dibuat program Algoritma "X" dengan tahap-tahap seperti berikut.

Tahap – tahap pada Algoritma "X" antara lain :

1) Mengurutkan fungsi kendala dan fungsi tujuan berdasarkan gradiennya (kecil ke besar).
2) Setelah mengurutkan dan mendapatkan urutan, maka didapatkan fungsi-fungsi kendala dengan gradien yang memenuhi $m_i \leq m_{f-tujuan} \leq m_{i+1}$, dengan m_i dan m_{i+1} merupakan gradien dari fungsi kendala ke-i dan i+1 , dengan i = 1,2,….,n (n merupakan banyaknya fungsi kendala dan i merupakan indeks dari fungsi kendala yang terurut).
3) Selanjutnya akan didapatkan nilai x dan y yang optimal dari perpotongan 2 fungsi kendala yang memenuhi.
4) Terakhir, akan didapatkan hasil optimalnya.

Untuk keseluruhan *Source Code* dapat dilihat pada Lampiran B

4.2.2. Fungsi Linear 3 variabel

Pada fungsi linear tiga variabel juga digunakan array seperti pada pembahasan fungsi linear dua variabel pada sub-sub bab 4.2.1, pada fungsi linear 3 variabel, dibuat beberapa kelas yang dibutuhkan untuk menyelesaikan langkah-langkah yang diperlukan oleh Algoritma "X" 3 Variabel dalam mencari nilai optimalisasinya. Berikut adalah kelas yang telah dibuat :

1. AlgoritmaX

AlgoritmaX adalah kelas utama yang digunakan untuk menjalankan Algoritma "X" untuk fungsi linear tiga variabel dengan inputnya berupa array 2 dimensi dengan tipe data array adalah 'double'. Sebagai contoh salah satu input pada Algoritma "X" adalah sebagai berikut:

Fungsi objektif $f = a_k x + b_k y + c_k z$

Fungsi Kendala :

$$a_1 x + b_1 y + c_1 z \leq d_1$$
$$a_2 x + b_2 y + c_2 z \leq d_2$$
$$\vdots$$
$$a_n x + b_n y + c_n z \leq d_n$$

Maka, input dari Algoritma "X" adalah

double [][] A = {{a_1,b_1,c_1},{a_2,b_2,c_2},...,{a_n,b_n,c_n}}

Untuk keselurahan *source code* dapat dilihat pada Lampiran C

2. AlgoritmaX_help

AlgoritmaX_help adalah kelas yang digunakan untuk mencari nilai optimal dari fungsi linear dua variabel, kelas ini dapat digunakan untuk membantu dalam mencari nilai optimal pada bidang (XoY, XoZ, YoZ, XYZ). Untuk keseluruhan *source code* dapat dilihat pada Lampiran D

3. Domain

Domain adalah kelas dengan output berupa boolean, yang digunakan untuk mengecek titik yang didapatkan apakah berada pada domain atau diluar domain

Source Code :

```
public static boolean domain(double x,double y,double
z,double p[][]){
     boolean cekdomain = true;
     int n = p.length;
     for (int i = 0; i < n-1; i++) {
     double hitung = (x*p[i][0])+(y*p[i][1])+(z*p[i][2]);
         if(hitung>p[i][3]){
                 cekdomain = false;
                 break;
         }
     }
return cekdomain;  }
```

4. Urutkan (*Sort)*

Urutkan adalah kelas yang digunakan untuk mengurutkan fungsi kendala berdasarkan gradiennya dari yang terkecil ke terbesar, kelas ini memiliki output array yang berisi gradien dan posisi fungsi tersebut.

Source Code :

```
public static double [][] urutkan(double c[][]){
        int n = c.length;
        for (int i = 0; i < n-1 ; i++) {
            for (int j = i+1; j < n; j++) {
                if(c[i][0]>c[j][0]){
                    double save[] = new double[2];
                    save[0]=c[i][0];save[1]=c[i][1];
                    c[i][0]=c[j][0];c[j][0]=save[0];
                    c[i][1]=c[j][1];c[j][1]=save[1];
                }
            }
        }
        return c;
    }
```

5. TitikPotong

TitikPotong adalah kelas untuk mencari titik potong fungsi linear 2 variabel dengan inputnya adalah array dan posisi fungsi dalam array yang ingin dicari perpotongannya.

Source Code :

```
public static double [] TitikPotong(double C[][], int i,
int j) {
```

```
        double a = C[i][0];
        double b = C[i][1];
        double c = C[i][2];
        double d = C[j][0];
        double e = C[j][1];
        double f = C[j][2];

        double x = (c*e-b*f)/(a*e-b*d);
        double y = (a*f-c*d)/(a*e-b*d);
        //System.out.println(x+"\t"+y);
        double []r = {x,y};
        return r;}
```

6. TitikPotong3

Tipot_3 adalah class yang digunakan untuk menyelesaikan permaslahan optimalisasi pada bidang *XYZ* dengan input array yang berisi fungsi kendala dan fungsi tujuan serta posisi-posisi fungsi yang ingin dihitung. Untuk *Source Code* dapat dilihat pada Lampiran E.

7. XoY, XoZ, YoZ

Terdapat kelas XoY, XoZ, YoZ, kelas tersebut digunakan untuk mengubah fungsi 3 variabel ke bentuk fungsi 2 variabel pada masing-masing bidang tersebut, selain itu dengan adanya kelas ini mempermudah pencarian titik optimal pada masing-masing bidang (XoY, XoZ, YoZ).

Source Code (XoY):

```
public static double [][] xoy(double c[][]){
        int n = c.length;
        double xy [][] = new double [n][3];

        for (int i = 0; i < n; i++) {
            if(c[i][0]==0&&c[i][1]==0){

            }
            else{
            xy[i][0]=c[i][0];
            xy[i][1]=c[i][1];
            xy[i][2]=c[i][3];

            }
        }
```

```
        return xy;
}
```

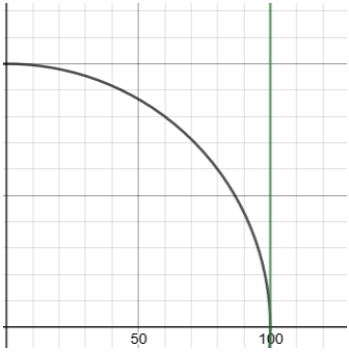
Untuk *Source Code* lebih lengkapnya dapat dilihat pada Lampiran F

4.3. Pembangkitan Data

Dalam penelitian ini, Algoritma "X" akan dibandingkan dengan Algoritma Simplex. Untuk membandingkan Algoritma "X" dengan Algoritma Simplex dengan banyak kendala dibutuhkan fungsi pembangkit data untuk mempermudah dalam pembuatan fungsi kendala dan meninjau hasil perbandingannya berdasarkan waktu komputasi dari masing-masing algoritma.

4.3.1. Fungsi Linear 2 Variabel

Pembangkitan fungsi kendala dilakukan dengan membuat persamaan lingkaran dengan pusat (0,0) yaitu $x^2 + y^2 = 10000$ dengan $r = 100$ dimana fungsi kendala merupakan garis-garis yang menyinggung persamaan lingkaran tersebut dan setiap fungsi saling berpotongan. Jari-jari lingkaran merupakan sebuah garis yang menghubungkan titik pusat dengan titik terluar dari lingkaran, suatu garis yang tegak lurus dengan jari-jari lingkaran merupakan garis singgung dari lingkaran tersebut. Jari-jari lingkaran juga memiliki besar sudut yang berbeda-beda.



Gambar 4. 4 Satu garis singgung lingkaran

Dari Gambar 4.4 dapat dilihat garis singgung tegak lurus dengan jari-jari lingkaran yang memiliki besar sudut (θ) = 0°. Dengan menggunakan jari-jari lingkaran dan besar sudut lingkaran yang berbeda-beda didapatkan persamaan garis singgung lingkaran yaitu : $r.\cos(\theta)\,x + r.\sin(\theta)\,y \leq r^2$ dengan (θ) merupakan besar sudut dari jari-jari lingkaran. Jadi, jari-jari lingkaran $r = 100$ dengan $\theta = 0°$ didapatkan persamaan garis singgungnya adalah :

$$100.\cos(0)\,x + 100.\sin(0)y \leq 100^2$$

$$100x \leq 10000 \text{ atau } x \leq 100.$$

Karena pada pemrograman linear memiliki syarat $x \geq 0$ dan $y \geq 0$ dalam artian perpotongan fungsi kendala harus berada di kuadran yang positif, maka besar sudut (θ) dari jari-jari berada pada selang $0° \leq \theta \leq 90°$.

Misal, dibuat fungsi kendala yang menyinggung lingkaran berjari-jari $r = 100$ sebanyak 4 fungsi, maka diperlukan $t = \frac{90}{n-1}$, dimana t merupakan pergeseran sudut sampai kurang dari sama dengan $90°$ dan n adalah banyaknya fungsi kendala yang dibuat yaitu 4. Maka didapatkan $t = \frac{90}{3} = 30$. Dilakukan $n = 4$ iterasi, dengan θ_i, dengan $i = 0,1,2 \ldots, n-1$ dengan $\theta_i = i.t$. Maka didapatkan :

$$\theta_0 = 0.t = 0°$$

$$\theta_1 = 1.t = 30°$$

$$\theta_2 = 2.t = 60°$$

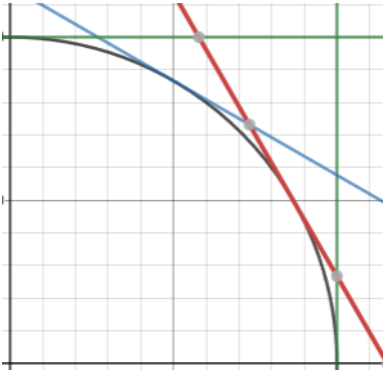
$$\theta_3 = 3.t = 90°$$

Maka persamaan garis singgung adalah :

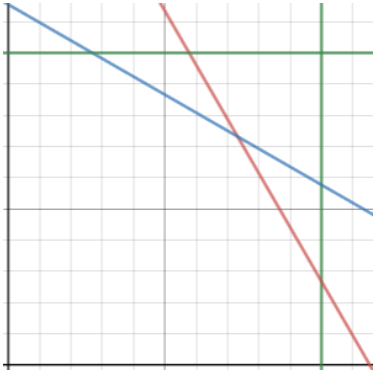
1. $r.\cos(\theta_0)\,x + r.\sin(\theta_0)\,y \leq r^2$
2. $r.\cos(\theta_1)\,x + r.\sin(\theta_1)\,y \leq r^2$
3. $r.\cos(\theta_2)\,x + r.\sin(\theta_2)\,y \leq r^2$
4. $r.\cos(\theta_3)\,x + r.\sin(\theta_3)\,y \leq r^2$

Substitusi nilai r dan θ, sehingga persamaan menjadi :

1. $100x \leq 10000$
2. $86{,}6x + 50y \leq 10000$
3. $50x + 86{,}6y \leq 10000$
4. $100y \leq 10000$



Gambar 4. 5 Fungsi yang menyinggung lingkaran



Gambar 4. 6 Fungsi kendala yang digunakan

Berdasarkan Gambar 4.6, didapatkan 4 fungsi kendala yang akan dicari nilai optimalisasinya. Dengan proses pembangkitan data seperti ini, maka semua fungsi kendala pasti berada dalam

domain sehingga tidak perlu lagi untuk melakukan proses pengecekan kendala.

Source Code :

```
public static double[][] BuatData(int n) {

        double pi = 90;
        double t = Math.toRadians(pi/(n-1));
        double r = 100;

        double c[][] = new double[n+1][3];
        for(int i = 0; i < n; i++) {
            c[i][0] = r*Math.cos(i*t);
            c[i][1] = r*Math.sin(i*t);
            c[i][2] = r*r;

        return c;
    }
```

Pada sub-bab ini dapat kita ketahui bahwa semakin banyak fungsi kendala tidak selalu menyebabkan daerah *search space* semakin mengecil. Untuk memahaminya lebih lanjut, simulasi fungsi pembangkit kendala dapat dilihat pada Lampiran H.

4.3.2. Fungsi Linear 3 Variabel

Untuk menguji dalam banyak kendala dibutuhkan sebuah fungsi yang bertujuan untuk membuat banyak kendala yang saling berpotongan, untuk mempermudah dalam membuat banyak data digunakan persamaan yang menyinggung bola, dengan bola berpusat pada $P(a,b,c)$, dimana persamaan yang menyinggung bola adalah

$$(x_1 - a)(x - a) + (y_1 - b)(y - b) + (z_1 - c)(z - c) = r^2$$

Karena bola yang digunakan berpusat pada $P(0,0,0)$ maka didapatkan persamaan singgungnya adalah:

$$x_1 x + y_1 y + z_1 z = r^2$$

Dengan x_1, y_1 dan z_1 merupakan titik yang berada pada kulit terluar bola. Untuk membuat kendala seperti itu, dibuat fungsi BuatData dengan inputnya berupa integer untuk menginisialisasikan banyaknya kendala yang ingin dibuat. Pada fungsi 'BuatData' persamaan yang menyinggung bola dibuat pada kondisi titik z tertentu hal ini ditujukan untuk mempermudah dalam pembuatan fungsi kendala. Pada fungsi ini, bola yang dibuat memiliki jari-jari atau $r = 100$. Untuk *Source Code* dapat dilihat pada Lampiran G.

4.4. Uji Program

Pada tahap ini dilakukan pengujian program Algoritma "X" yang telah dibuat pada Fungsi Linear 2 Variabel dan Fungsi Linear 3 Variabel. Uji coba dilakukan dengan menggunakan program komputer dengan bahasa pemrograman JAVA.

4.4.1. Fungsi Linear 2 Variabel

Sebelumnya pada sub-sub-bab 4.2.1, input dari Algoritma "X" berupa *array*, karena sudah dibuat sebuah fungsi *array* pembangkitan data dengan inputnya sebuah *integer* yang mendefinisikan banyaknya data yang ingin dibuat. Oleh karena hal tersebut, maka dilakukan sedikit perubahan pada fungsi Algoritma "X" menjadi :

Source Code :

```
public static void AlgoritmaX(int data){
        double c[][] = BuatData(data);
        int n = c.length;
```

Terdapat 2 kelas berbeda yaitu fungsi Algoritma "X" untuk banyak data dengan menggunakan fungsi 'BuatData' dan fungsi Algoritma "X" untuk kendala yang diinginkan (diinput secara manual), yang akan dilakukan pengujian untuk masing-masing fungsi tersebut.

1. Algoritma "X" dengan input kendala manual

Pada sub-sub-bab 4.1.1 diberikan fungsi kendala $x \leq 1, x + y \leq 2$ dan $y \leq 1$ dengan fungsi objektifnya adalah $z = 30x + 20y$. Berdasarkan perhitungan secara matematis, didapatkan titik maksimalnya $(1,1)$ dengan nilai maksimalnya adalah $Z = 30(1) + 20(1) = 50$. Selanjutnya akan dilakukan uji program dengan fungsi objektif dan fungsi kendala tersebut.

Input merupakan koefisien dari masing-masing fungsi, yang diletakkan didalam *array*. Fungsi objektif diletakkan diposisi akhir pada *array*. Diperlukan pengecekan fungsi kendala dengan menggunakan fungsi yang telah dibuat sebelumnya yaitu fungsi 'cek' sehingga untuk menjalankan program seperti berikut.

Input :

```
double c[][] =
{{1,0,1},{0,1,1},{1,1,2},{30,20,0}};
```

Running Program :

```
AlgoritmaX(cek(c));
```

Output :

```
run:
=====Algoritma X=====
Dengan hasil maksimalnya adalah 50.0
dengan X = 1.0 dan Y = 1.0
Waktu yang dibutuhkan adalah = 0.0 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 7 Hasil Optimal Algoritma "X" dua variabel

Kemudian didapatkan hasil optimalnya benar dan sesuai dengan penghitungan pada sub-sub-bab 4.1.1 yaitu sebesar $z = 30x + 20y = 50$ pada titik $(1,1)$. Untuk mengetahui hasil optimal dari Algoritma "X", akan dilakukan uji coba pada program

Algoritma Simplex yang telah dibuat oleh penulis dan didapatkan hasil :

Output :

```
=====Simpleks=====
Hasil Maksimalnya adalah 50.0
dengan nilai X = 1.0 dan Y = 1.0
Waktu yang dibutuhkan adalah = 0.0 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 8 Hasil Optimal Algoritma Simplex dua variabel

Didapatkan hasil yang diperoleh Algoritma “X” adalah optimal.

2. Algoritma “X” dengan input banyak data

Seperti yang dijelaskan pada awal bagian pada sub-sub-bab ini, untuk input banyak data digunakan fungsi ‘BuatData’, maka untuk dapat menjalankan programmnya, dilakukan tahapan seperti berikut :

Input :

```
int jumlah = 10;
```

Running Program :

```
AlgoritmaX(jumlah);
```

Output :

```
run:
== ALGORITMA X ==
dengan X = 57.57 dan Y = 82.22
Dengan hasil maksimalnya adalah 320723.74
Waktu yang dibutuhkan adalah = 0.005 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 9 Hasil Optimal Algoritma "X" dua variabel

Dengan menginputkan data fungsi kendala sebanyak 10 dan fungsi objektifnya adalah $Z = 2000x + 2500y$, didapatkan hasil *output* seperti diatas. Untuk mengetahui hasil optimal Algoritma "X" dapat dilihat pada sub-bab berikutnya.

4.4.2. Fungsi Linear 3 Variabel

Pada sub-sub bab ini, akan dilakukan uji program AlgoritmaX untuk fungsi linear 3 variabel, Algoritma "X" akan diuji dengan 2 input yang berbeda, yaitu dengan cara manual dan dengan cara menggunakan fungsi BuatData yang telah buat. Berikut dilakukan uji programnya :

1. Algoritma "X" dengan input manual

Pada sub-sub bab 4.1.2 diberikan fungsi yang ingin dioptimalkan adalah $Q = 3000x + 4000y + 3000z$ dengan kendala sebagai berikut :

1) $2x + y + z \leq 4$
2) $x + 2y + z \leq 4$
3) $x + y + 2z \leq 4$
4) $x + y + z \leq 3$

Didapatkan titik maksimalnya adalah $x = 1, y = 1$ dan $z = 1$. Dengan nilai maksimalnya adalah $Q = 3000(1) + 4000(1) + 3000(1) = 10.000$.

Untuk menggunakan fungsi Algoritma “X”, dilakukan pembuatan array untuk menampung fungsi kendala dan fungsi tujuan yang ingin diselesaikan seperti berikut :

Input :

```
double c[][] =
{{2,1,1,4},{1,2,1,4},{1,1,2,4},{1,1,1,3},{3000,4000,30
00,0}};
```

Kemudian, dilakukan uji program Algoritma “X” seperti berikut :

Running Program :

```
AlgoritmaX(c);
```

Output :

```
====ALGORITMA X====

dengan x = 1.0 y = 1.0 dan z = 1.0
Maka hasil maksimalnya adalah = 10.0
TOTAL WAKTU = 0.009 detik

BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 10 Hasil Optimal Algoritma “X” tiga variabel

Didapatkan hasil yang sama dengan hasil pada sub-sub bab 4.1.2 dengan $Q = 10.000$ serta titik optimalnya $x = 1$, $y = 1$, $z = 1$. Untuk mengetahui hasil optimal dari Algoritma “X” akan dilakukan uji coba pada program Algoritma Simplex yang telah dibuat oleh penulis. Didapatkan hasilnya :

Output :

```
====METODE SIMPLEKS====

Dengan x = 1.0 y = 1.0 dan z = 1.0
Maka hasil Maksimalnya adalah = 10.0
TOTAL WAKTU = 0.001 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 11 Hasil Optimal Algoritma Simplex tiga variabel

Dari hasil yang didapatkan oleh Algoritma Simplex diatas. Dapat dikatakan bahwa hasil yang diperoleh (*Output*) Algoritma "X" adalah optimal.

2. AlgoritmaX dengan input fungsi 'BuatData'

Dilakukan uji program dengan input yang telah dibuat dengan fungsi 'BuatData' dengan menggunakan fungsi ini tidak diperlukan lagi proses penginputan array, karena 'BuatData' memiliki output berupa array, selanjutnya dilakukan pengujian program seperti berikut :

Input :

```
Int jumlah = 10;
```

Running Program :

```
AlgoritmaX(BuatData(jumlah));
```

Output :

```
====ALGORITMA X====

dengan x = 0.33 y = 0.47 dan z = 19.0
Maka hasil maksimalnya adalah = 97896.23
TOTAL WAKTU = 0.023 detik

BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 12 Hasil Optimal Algoritma "X" tiga variabel

Dengan fungsi objektif adalah $Z = 3000x + 4000y + 5000z$. Untuk mengetahui lebih lanjut apakah hasil yang didapat sudah optimal, dilakukan perbandingan dengan Algoritma Simplex yang telah dibuat programnya pada sub-bab berikutnya.

4.5. Membandingkan Hasil Optimal

Pada sub-bab ini akan dilakukan perbandingan antara Algoritma "X" dan Algoritma Simplex untuk pemrograman linear dua variabel dan tiga variabel. Dalam membandingkan Algoritma "X" dengan Algoritma Simplex objek yang akan diperhatikan adalah waktu komputasi (*running time*), banyak data (fungsi kendala) dan hasil optimalnya.

4.5.1. Fungsi Linear 2 Variabel

Pada sub-sub-bab ini akan dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex yang ditampilkan dalam sebuah tabel yang berisi banyaknya data, hasil optimal dan waktu yang dibutuhkan. Kemudian dihasilkan tabel seperti berikut :

Tabel-4. 1 Tabel Perbandingan 2 Variabel

Banyak Kendala	Hasil Optimal		*Running Time (sekon)*	
	AlgoritmaX	Simplex	AlgoritmaX	Simplex
5	367618.61	367618.61	0.005	0.005
10	723675.60	723675.60	0.004	0.004
15	861583.68	861583.68	0.005	0.005
20	875109.30	875109.30	0.005	0.005

- Untuk 5 fungsi kendala :

$$P = 3000x + 2000y$$

- Untuk 10 fungsi kendala :

$$P = 4000x + 6000y$$

- Untuk 15 fungsi kendala :

$$P = 7000x + 5000y$$

- Untuk 20 fungsi kendala :

$$P = 5500x + 6800y$$

Dari tabel-4.1 diatas, Algoritma "X" mendapatkan hasil yang optimal dan Algoritma "X" membutuhkan waktu yang sama dengan Algoritma Simplex dalam mencari nilai optimal. *Running time* diambil yang tercepat dari 10 kali percobaan.

Selanjutnya, dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex dengan data yang diambil dari Toko Bakpia Pathuk 714. Berikut adalah bahan-bahan yang dibutuhkan untuk membuat setiap jenis bakpia per kotak [10].

***Tabel-4. 2 Data Kendala Bakpia Pathuk 714* [10]**

Bahan Baku (Kg)	Kacang Hijau (x)	Cokelat (y)	Tersedia
Tepung terigu	0.2	0.5	30 kg
Gula pasir	0.0025	0.25	10 kg
Minyak goreng	0.015	0.1	6 kg
Margarin	0.01	0.075	3 kg
Kacang hijau	0.125	0	10 kg
Cokelat	0	0.0625	0.5 kg
Keuntungan	Rp. 13.045	Rp. 6.275	

Sehingga, didapatkan persamaan fungsi objektif nya $Z = 13045x + 6275y$. Dengan fungsi kendala sebagai berikut :

1) $2x + 3y \leq 300$
2) $25x + 2500y \leq 100000$
3) $15x + 100y \leq 6000$
4) $10x + 75y \leq 3000$
5) $125x \leq 10000$
6) $625y \leq 5000$

Selanjutnya fungsi akan diuji pada program Algoritma “X” dan Algoritma Simplex. Didapatkan hasil sebagai berikut.

```
run:
=====Algoritma X=====
Dengan hasil maksimalnya adalah 1093800.0
dengan X = 80.0 dan Y = 8.0
Waktu yang dibutuhkan adalah = 0.002 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 13 Hasil dari Algoritma “X”

```
=====Simpleks=====
Hasil Maksimalnya adalah 1093800.0
dengan nilai X = 80.0 dan Y = 8.0
Waktu yang dibutuhkan adalah = 0.002 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 14 Hasil dari Algoritma Simplex

Dari hasil yang diperoleh, maka Toko Bakpia Pathuk 714 dapat memperoleh keuntungan yang maksimal sebesar Rp. 1.093.800,00 dengan menjual Bakpia Kacang Hijau sebanyak 80 kotak dan Bakpia Cokelat sebanyak 8 Kotak. Berdasarkan hasil diatas Algoritma "X" memperoleh hasil yang sama dengan Algoritma Simplex sehingga Algoritma "X" memiliki hasil yang optimal.

4.5.2. Fungsi Linear 3 Variabel

Pada sub-sub-bab ini akan dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex yang ditampilkan dalam sebuah tabel yang berisi banyaknya data, hasil optimal dan waktu yang dibutuhkan. Kemudian dihasilkan tabel seperti berikut :

Tabel-4. 3 Tabel Perbandingan 3 Variabel

Banyak Kendala	Hasil Optimal		*Running Time (sekon)*	
	AlgoritmaX	Simplex	AlgoritmaX	Simplex
5	77305.95	77305.95	0.011	0.001
10	49176.92	49176.92	0.026	0.002
15	138330.15	138330.15	0.031	0.001
20	59451.58	59451.58	0.037	0.002

- Untuk 5 fungsi kendala :

$$Q = 2000x + 1000y + 4000z$$

- Untuk 10 fungsi kendala :

$$Q = 3000x + 3000y + 500z$$

- Untuk 15 fungsi kendala :

$$Q = 7000x + 1000y + 7000z$$

- Untuk 20 fungsi kendala :

$$Q = 3000x + 3000y + 3000z$$

Dari Tabel-4.3 diatas, Algoritma “X” mendapatkan hasil yang optimal dan Algoritma “X” membutuhkan waktu yang lebih lambat dari pada Algoritma Simplex. *Running time* diambil yang tercepat dari 10 kali percobaan.

Selanjutnya, akan dilakukan perbandingan antara Algoritma “X” dengan Algoritma Simplex dengan data yang diambil dari PT. KBDTI yang merupakan perusahaan yang bergerak dalam bidang manufaktur yang memproduksi ekstrak buah dengan berbagai macam varians. Varians rasa yang dihasilkan yaitu ekstrak buah jambu biji merah, ekstrak buah sirsak dan ekstrak buah lemon. Ketersediaan buah perminggu yaitu buah jambu biji merah sebanyak 80.000 kg, buah sirsak sebanyak 50.000 kg dan buah lemon sebanyak 25.000 kg. sedangkan ketersediaan bahan baku penunjang yaitu asam nitrat sebanyak 300 kg, fruktosa 600 kg, tenaga kerja yang dipekerjakan di PT. KBDTI yaitu sebanyak 20 orang pada bagian produksi, dalam satu minggu PT. KBDTI membutuhkan waktu produksi selama 90 jam[9], dengan waktu yang dibutuhkan dalam produksi adalah 3 jam untuk setiap varians.

***Tabel-4. 4 Data Kendala PT. KBDTI* [9]**

Uraian	Ekstrak Jambu biji ($\boldsymbol{x}$)	Eksetrak Sirsak ($\boldsymbol{y}$)	Ekstrak Lemon ($\boldsymbol{z}$)	Tersedia
Buah Jambu	13	0	0	80.000 kg
Buah Sirsak	0	12	0	50.000 kg
Buah Lemon	0	0	20	25.000 kg
Fruktosa	23	21	25	600 kg
Asam Sitrat	10	5	9	300 kg
Tenaga Kerja	1	1	1	20
Waktu	3	3	3	90 Jam
Keuntungan	Rp 32.500	Rp 60.000	Rp 87.500	

Sehingga fungsi objektif dapat ditulis $Z = 32500x + 60000y + 87500z$. Dengan fungsi kendala yaitu :

1) $13x \leq 80000$
2) $12y \leq 50000$
3) $20z \leq 25000$
4) $23x + 21y + 25z \leq 600$
5) $10x + 5y + 9z \leq 300$
6) $x + y + z \leq 20$
7) $3x + 3y + 3z \leq 90$

Selanjutnya fungsi akan diuji pada program Algoritma “X” dan Algoritma Simplex. Didapatkan hasil seperti berikut :

```
====ALGORITMA X====

dengan x = 0.0 y = 0.0 dan z = 20.0
Maka hasil maksimalnya adalah = 1750000.0
TOTAL WAKTU = 0.006 detik

BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 15 Hasil dari Algoritma "X"

```
====METODE SIMPLEKS====

Dengan x = 0.0 y = 0.0 dan z = 20.0
Maka hasil Maksimalnya adalah = 1750000.0
TOTAL WAKTU = 0.003 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 16 Hasil dari Algoritma Simplex

Dari hasil yang diperoleh, maka PT. KBDTI dapat memperoleh keuntungan yang maksimal sebesar Rp. 1.750.000,00 dengan memproduksi Ekstrak Lemon sebanyak 20 dalam satu minggu. Berdasarkan hasil diatas Algoritma "X" memperoleh *output* yang sama dengan Algoritma Simplex sehingga *output* Algoritma "X" dapat dikatan optimal.

4.6. Perbandingan yang mencapai 1000 Kendala

Pada sub-bab ini dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex yang mencapai 1000 kendala. Dilakukannya perbandingan ini adalah untuk mengetahui hasil optimal yang didapatkan dan *running time* yang dibutuhkan apabila mencapai 1000 kendala. Selain itu, dilakukannya perbandingan ini untuk mengetahui banyaknya jumlah kendala yang menyebabkan Algoritma "X" lebih cepat dari Algoritma Simplex

4.6.1. Fungsi Linear 2 Variabel

Pada sub-sub-bab ini akan dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex yang ditampilkan dalam sebuah tabel yang berisi banyaknya kendala, hasil optimal dan waktu yang dibutuhkan. Kemudian dihasilkan tabel seperti berikut :

Tabel-4. 5 Tabel Perbandingan 2 Variabel

Banyak Kendala	Hasil Optimal		*Running Time (sekon)*	
	AlgoritmaX	Simplex	AlgoritmaX	Simplex
100	360557.65	360057.65	0.0015	0.0062
250	721112.62	721112.62	0.0016	0.0281
500	860232.68	860232.68	0.0031	0.1719
1000	874585.88	874585.88	0.0062	1.1779

- Untuk 100 fungsi kendala :

$$P = 3000x + 2000y$$

- Untuk 250 fungsi kendala :

$$P = 4000x + 6000y$$

- Untuk 500 fungsi kendala :

$$P = 7000x + 5000y$$

- Untuk 1000 fungsi kendala :

$$P = 5500x + 6800y$$

Dari Tabel-4.5 diatas, Algoritma "X" mendapatkan hasil yang optimal dan Algoritma "X" membutuhkan waktu yang lebih singkat dari pada Algoritma Simplex dalam mencari nilai optimal sampai 1000 kendala. *Running time* diambil yang tercepat dari 5 kali percobaan. Saat dilakukan uji coba dengan 60 kendala, didapatkan Algoritma "X" lebih cepat dari Algoritma Simplex. Dalam hal ini, jika kendala yang digunakan lebih dari sama dengan 60 kendala maka Algoritma "X" terbukti lebih cepat dari Algoritma Simplex.

```
== METODE SIMPLEKS ==
Dengan nilai X = 62.79 dan Y = 77.83
Hasil Maksimalnya adalah 874662.48

TOTAL WAKTU ADALAH = 0.0031
```

Gambar 4. 17 Running time Algoritma Simplex saat 60 kendala

```
== ALGORITMA X ==
dengan X = 62.79 dan Y = 77.83
Dengan hasil maksimalnya adalah 874662.48

TOTAL WAKTU ADALAH = 0.0015
```

Gambar 4. 18 Running time Algoritma "X" saat 60 kendala

Gambar 4. 19 Kurva Perbandingan Waktu Komputasi Algoritma "X" dengan Algoritma Simplex

4.6.2. Fungsi Linear 3 Variabel

Pada sub-sub-bab ini akan dilakukan perbandingan antara Algoritma "X" dengan Algoritma Simplex yang ditampilkan dalam sebuah tabel yang berisi banyaknya kendala, hasil optimal dan waktu yang dibutuhkan. Kemudian dihasilkan tabel seperti berikut :

Tabel-4. 6 Tabel Perbandingan 3 Variabel

Banyak Kendala	Hasil Optimal		*Running Time (sekon)*	
	AlgoritmaX	Simplex	AlgoritmaX	Simplex
100	460466.54	460466.54	0.051	0.011
250	427508.24	427508.24	0.109	0.031
500	995408.76	995408.76	0.125	0.094
1000	520296.25	520296.25	0.281	0.484

- Untuk 100 fungsi kendala :
$$Q = 2000x + 1000y + 4000z$$
- Untuk 250 fungsi kendala :
$$Q = 3000x + 3000y + 500z$$
- Untuk 500 fungsi kendala :
$$Q = 7000x + 1000y + 7000z$$
- Untuk 1000 fungsi kendala :
$$Q = 3000x + 3000y + 3000z$$

Dari Tabel-4.6 diatas, Algoritma "X" mendapatkan hasil yang optimal dan Algoritma "X" membutuhkan waktu yang lebih singkat dari pada Algoritma Simplex pada 1000 kendala, dalam mencari nilai optimal. *Running time* diambil yang tercepat dari 5 kali percobaan. Saat dilakukan uji coba dengan 800 kendala, didapatkan Algoritma "X" lebih cepat dari Algoritma Simplex.

Dengan demikian, ketika kendala mencapai lebih dari sama dengan 800 kendala, Algoritma “X” lebih cepat dari Algoritma Simplex

```
====METODE SIMPLEKS====

Dengan x = 59.11 y = 59.11 dan z = 55.19
Maka hasil Maksimalnya adalah = 520302.62
TOTAL WAKTU = 0.344 detik
BUILD SUCCESSFUL (total time: 0 seconds)
```

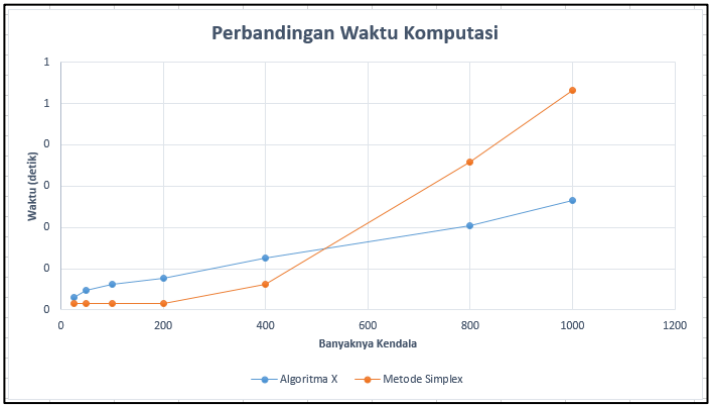
Gambar 4. 20 Running Time Algoritma Simplex saat 800 kendala

```
====ALGORITMA X====

dengan x = 59.11 y = 59.11 dan z = 55.19
Maka hasil maksimalnya adalah = 520302.62
TOTAL WAKTU = 0.234 detik

BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 21 Running Time Algoritma “X” saat 800 kendala



Gambar 4. 22 Kurva Perbandingan Waktu Komputasi Algoritma “X” dengan Algoritma Simplex

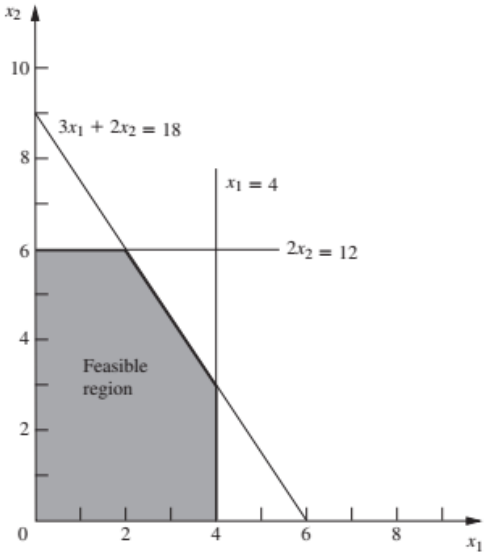
4.7. Kompleksitas Waktu

Pada sub-bab ini dilakukan pencarian kompleksitas waktu dari masing-masing algoritma yang dibuat yaitu Algoritma "X" dan Algoritma Simplex. Pada Algoritma "X" terdapat beberapa tahapan yang memiliki pengaruh terhadap kompleksitas waktu dari Algoritma "X" yaitu seperti *selection sort* yang memiliki kompleksitas waktu $\left(O(n^2)\right)$ dan *sequential search* dengan kompleksitas waktunya $\left(O(n)\right)$. Dalam menentukan kompleksitas waktu dari suatu Algoritma hanya diperlukan mengambil derajat tertinggi dari banyaknya kerja yang dilakukan oleh algoritma tersebut. Misal, $O(m) + O(n) = \left(O(\max(m,n))\right)$ dimana m, n adalah derajat tertinggi yang didapat dari suatu tahapan. Sehingga untuk Algoritma "X" didapatkan kompleksitas waktunya adalah $O(\max(n^2, n)) = O(n^2)$. Pada Algoritma Simplex sendiri salah satu faktor yang mempengaruhi kompleksitas waktunya adalah pada tahapan Operasi Baris Elementer (OBE) dengan kompleksitas waktunya adalah $O(n^2)$, maka kompleksitas waktu dari Algoritma Simplex adalah $O(n^2)$.

Dari segi kompleksitas waktu, Algoritma "X" memiliki kompleksitas yang sama dengan Algoritma Simplex yaitu $O(n^2)$. Apabila kendala semakin banyak, pertumbuhan waktu dari Algoritma "X" lebih baik dari Algoritma Simplex. Hal tersebut memungkinkan karena proses iterasi terhadap n kendala pada Algoritma Simplex lebih banyak daripada Algoritma "X".

4.8. Contoh Aplikasi

Sebagaimana kita ketahui, batasan-batasan dalam permasalahan linear programming mempengaruhi domain (*feasible region*) dari variabel-variabel yang ada dalam permasalahan tersebut.



Gambar 4. 23 Grafik Fungsi

Dalam bagian ini, diberikan contoh aplikasi pada permasalahan optimasi fungsi yang memiliki dua variabel.

Kebanyakan masalah optimasi memiliki bentuk

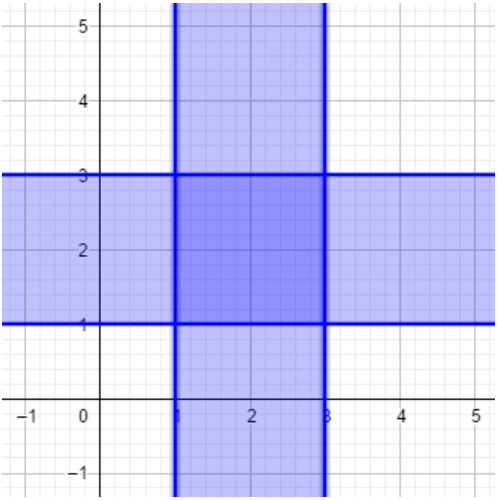
$$\max f(x, y)$$

dengan

$$a \leq x \leq b$$

$$c \leq y \leq d$$

Misalkan $a = 1, b = 3, c = 1, d = 3$, maka domain dari permasalahan tersebut adalah sebagaimana pada Gambar 4.24.



Gambar 4. 24 Grafik dari Domain

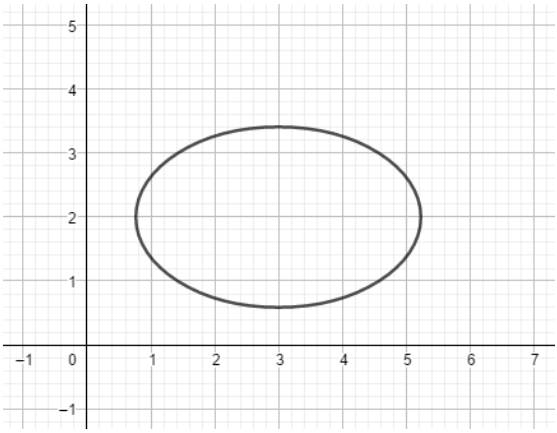
Salah satu aplikasi dari apa yang dikerjakan dalam Tugas Akhir ini adalah untuk menyelesaikan masalah optimasi dengan domain yang lebih kompleks, sebagai contoh

$$\max f(x, y)$$

dengan

$$\frac{(x-a)^2}{b} + \frac{(y-c)^2}{d} = 1.$$

Misalkan $a = 3, b = 5, c = 2, d = 2$, maka domain dari permasalahan tersebut adalah sebagaimana pada Gambar 4.25.



Gambar 4. 25 Bentuk Domain dari Persamaan Ellips

Hal ini dimungkinkan karena kita bisa menghampiri bentuk tersebut dengan beberapa garis lurus yang menghubungkan titik-titik pada bentuk tersebut.

BAB V
PENUTUP

5.1. Kesimpulan

Kesimpulan dari penelitian ini adalah sebagai berikut :

1. Pada pemrogramman linear dua variabel, Algoritma "X" dapat menentukan dua fungsi kendala (yang mempengaruhi domain) yang memiliki perpotongan titik yang optimal secara langsung dengan melihat karakteristik (gradien) dari semua fungsi kendala dan juga fungsi objektif yang diberikan. Dengan mudah, gradien dapat dibuat dalam bentuk interval seperti berikut ($m_i \leq m_{f_{objektif}} \leq m_{i+1}$). Sehingga didapatkan titik maksimalnya merupakan perpotongan dari fungsi yang gradiennya memenuhi interval tersebut. Pada pemrograman linear tiga variabel, titik optimal didapatkan dari perpotongan tiga fungsi kendala tertentu (yang mempengaruhi domain). Ketiga fungsi kendala tersebut dapat dicari dengan melihat karakteristik $\left(E = \left|\frac{a_1}{b_1} - \frac{a_2}{b_2}\right| + \left|\frac{a_1}{c_1} - \frac{a_2}{c_2}\right| + \left|\frac{b_1}{c_1} - \frac{b_2}{c_2}\right|\right)$ dari semua fungsi kendala dan juga fungsi objektif yang diberikan. Sehingga, didapatkan tiga fungsi yang terpilih yaitu fungsi yang memiliki nilai karakteristik (E) terkecil. Tiga fungsi terpilih merupakan fungsi yang memiliki titik potong yang menyebabkan fungsi objektif mendapatkan nilai yang maksimal.
2. Waktu (*Running Time*) yang dibutuhkan Algoritma "X" pada pemrograman linear 2 variabel sama cepat dengan Algoritma Simplex dan mendapatkan hasil yang optimal. Waktu yang dibutuhkan Algoritma "X" pada pemrograman linear 3 variabel lebih lambat dari pada Algoritma Simplex dan mendapatkan hasil yang optimal.

3. Apabila kendala yang digunakan mencapai lebih dari sama dengan 60 kendala, untuk pemrograman linear 2 variabel Algoritma "X" terbukti lebih cepat dari Algoritma Simplex. Sedangkan, untuk pemrograman linear 3 variabel Algoritma "X" lebih cepat apabila kendala yang digunakan mencapai lebih dari sama dengan 800 kendala.
4. Berdasarkan kompleksitas waktunya, Algoritma "X" memiliki kompleksitas yang sama dengan Algoritma Simplex yaitu $O(n^2)$. Namun, bila banyaknya kendala mencapai titik optimalnya (kesimpulan nomor 3) Algoritma "X" terbukti lebih cepat dari Algoritma Simplex hal tersebut dikarenakan iterasi pada Algoritma Simplex lebih banyak daripada Algoritma "X".
5. Berdasarkan simulasi dengan fungsi pembangkit kendala dan kasus permasalahan nyata dari pemrograman linear didapatkan bahwa Algoritma "X" memperoleh hasil yang valid.

5.2. Saran

Saran dari penulis untuk penelitian selanjutnya adalah :

Algoritma "X" dapat dikembangkan agar dapat digunakan untuk menyelesaikan pemrograman linear dengan banyak variabel.

DAFTAR PUSTAKA

[1] Chandra, T. (2015). Penerapan Algoritma Simplex dalam Aplikasi Penyelesaian Masalah Program Linear. *Jurnal Times*, *4*(1), 18-21.

[2] Wisudawan, W. F. (2007). Kompleksitas Algoritma Sorting yang Populer Dipakai. *Bandung: Teknik Informatika, Institut Teknologi Bandung*.

[3] Sundary, B. (2014). Penerapan Program Linear dalam Optimasi Biaya Pakan Ikan dengan Metode Simplex (Studi Kasus PT. Indojaya Agrinusa Medan). *Informasi dan Teknologi Ilmiah*.

[4] Kusumawati, P. Y. (2007). Perbandingan Metode Simplex dan Revised Simplex pada Penyelesaian Program Linear. *Yogyakarta : Teknik Informatika, Universitas Sanata Dharma.*

[5] Sriwidadi, T., & Agustina, E. (2013). Analisis Optimalisasi Produksi dengan Linear Programming Melalui Metode Simplex. *Binus Business Review*, *4*(2), 725-741.

[6] Hillier, F. S. (2012). *Introduction to operations research*. Tata McGraw-Hill Education.

[7] Rahayuningsih, P. A. (2016). Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (Sorting). *EVOLUSI: Jurnal Sains dan Manajemen*, *4*(2).

[8] Soyster, A. L., Lev, B., & Slivka, W. (1978). Zero-one programming with many variables and few constraints. *European Journal of Operational Research*, *2*(3), 195-201.

[9] AlVonda, Q. R., Dinni, F., Saputra, D. D., Puspita, I., Falani, I., & Wiratmani, E. (2019). Implementasi Metode Simpleks dalam Penentuan Jumlah Produksi untuk Memaksimasi Keuntungan. *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*,*4*(1),57-64.

[10] Putri, R. O., Retnoningsih, S., Suryawan, A., Affianti, A., & Yanty, R. (2017). Implementasi Linear Programming untk Memaksimumkan Keuntungan Produksi Bakpia Dengan Menggunakan Metode Simpleks. *Yogyakarta : Manajemen Keuangan Syari'ah, Universitas Islam Negri Sunan Kalijaga*

LAMPIRAN

LAMPIRAN A

```
public static double [][] cek (double c [][]){
        int n = c.length;
//mencari fungsi kendala dengan nilai y terkecil
saat x=0
        double cek_y=Double.POSITIVE_INFINITY;
        int curr=0;
        for (int i = 0; i < n-1; i++) {
            double small = c[i][2]/c[i][1];
            if(small<cek_y){
                cek_y=small;
                curr=i;
            }    }
//melakukan pertukaran posisi fungsi kendala
        double [] save2 = new double[3];
        save2[0]=c[0][0];
        save2[1]=c[0][1];
        save2[2]=c[0][2];
        c[0][0]=c[curr][0];c[curr][0]=save2[0];
        c[0][1]=c[curr][1];c[curr][1]=save2[1];
        c[0][2]=c[curr][2];c[curr][2]=save2[2];
//mengurutkan fungsi kendala berdasarkan gradien
        for(int i=1;i<n-2;i++){
            for(int j=i+1;j<n-1;j++){
                double gradien=c[i][0]/c[i][1]*-1;
                double gradien2=c[j][0]/c[j][1]*-1;
                if(gradien2>gradien){
                    double [] save = new double[3];
                    save[0]=c[i][0];
                    save[1]=c[i][1];
                    save[2]=c[i][2];

                 c[i][0]=c[j][0];c[j][0]=save[0];
                 c[i][1]=c[j][1];c[j][1]=save[1];
                 c[i][2]=c[j][2];c[j][2]=save[2];
                }
            }
         }
```

```
//memasukkan fungsi constrain terbaru kedalam array
baru
        double baru [][]=new double[n+2][3];
        baru[0][0]=c[0][0];
        baru[0][1]=c[0][1];
        baru[0][2]=c[0][2];

        boolean loop = true;
        int j=0;
        int set=-1;
        curr=0;
        while (loop==true){
            loop = false;
            double xx = Double.POSITIVE_INFINITY ;
            for (int i = j+1; i < n-1; i++) {
                double x = TitikPotong(c,i,j)[0];
                double y = TitikPotong(c,i,j)[1];
                if(x<xx&&y>=0&&x>=0){
                    xx=x;
                    set=i;
                    loop =true;
                }
            }
            j=set;
                        if(loop==true){
                curr++;
                baru[curr][0]=c[j][0];
                baru[curr][1]=c[j][1];
                baru[curr][2]=c[j][2];
            }
        }
```

```
 //melakukan penambahan fungsi constraint dan
fungsi tujuan
      double cek_x = Double.POSITIVE_INFINITY;
        for (int i = 0; i < curr+1; i++) {
            double small = baru[i][2]/baru[i][0];
            if(small<cek_x){
                cek_x=small;
            }
        }
        baru[curr+1][0]=1;
        baru[curr+1][1]=0;
        baru[curr+1][2]=cek_x;
        baru[curr+2][0]=0;
        baru[curr+2][1]=1;
        baru[curr+2][2]=cek_y;
        baru[curr+3][0]=c[n-1][0];
        baru[curr+3][1]=c[n-1][1];
        baru[curr+3][2]=c[n-1][2];
        n = curr+4; // n array terbaru berdasarkan
kendala terpilih
        double [][] save = new double [n][3];
        for (int i = 0; i < n; i++) {
            save[i][0]=baru[i][0];
            save[i][1]=baru[i][1];
            save[i][2]=baru[i][2];
        }
       baru = new double[n][3];
       int nn = baru.length;
       System.out.println("AFTER RENEW...");
        for (int i = 0; i < nn; i++) {
            baru[i][0]=save[i][0];
            baru[i][1]=save[i][1];
            baru[i][2]=save[i][2];
            System.out.println(baru[i][0]+"x
+"+baru[i][1]+"y <= "+baru[i][2]);
        }
        return baru;
 }
```

LAMPIRAN B

```
public static void AlgoritmaX(double c[][]){
int n = c.length;
double gradien,gradien2;
int pos=0;
if(c[n-1][0]!=0&&c[n-1][1]!=0){
        for(int i=0;i<n-1;i++){
            for(int j=i+1;j<n;j++){
                gradien=c[i][0]/c[i][1]*-1;
                gradien2=c[j][0]/c[j][1]*-1;
                if(gradien2<gradien){
                    double [] save = new double[3];
                    save[0]=c[i][0];
                    save[1]=c[i][1];
                    save[2]=c[i][2];
                    c[i][0]=c[j][0];c[j][0]=save[0];
         c[i][1]=c[j][1];c[j][1]=save[1];
         c[i][2]=c[j][2];c[j][2]=save[2];
                }
                else continue;
            }
        }
            System.out.println("=====setelah diurutkan
berdasarkan gradien======");
        for (int i = 0; i < n; i++) {
             System.out.println((i+1)+". "+c[i][0]+"x +
"+c[i][1]+"y <= "+c[i][2]);
         }
        for (int i = 0; i < n; i++) {//mencari posisi
dari array fungsi tujuan
            if(c[i][2]==0){
                pos=i;
            }
        }
int m_1 = pos-1; //dimana m1 < pos < m2
int m_2 = pos+1;

double x = TitikPotong(c,m_1,m_2)[0];
double y = TitikPotong(c,m_1,m_2)[1];
double hasil = (x*c[pos][0])+(y*c[pos][1]);
System.out.println("Dengan hasil maksimalnya adalah
"+hasil);
System.out.println("dengan X = "+x+ " dan Y = "+y);
}
```

```
else{
System.out.println("Fungsi tujuan ada yg nol");
    if(c[n-1][0]==0){
       double y = c[0][2]/c[0][1];
       double hasil =(y*c[n-1][1]);
       System.out.println("Dengan hasil maksimalnya
adalah "+hasil);
       System.out.println("dengan X = "+0+" dan Y =
"+y);
    }
     else if(c[n-1][1]==0){
       double x = c[n-2][2]/c[n-2][0];
       double hasil =(x*c[n-1][0]);
       System.out.println("Dengan hasil maksimalnya
adalah "+hasil);
      System.out.println("dengan X = "+x+" dan Y =
"+0);
```

LAMPIRAN C

```
public static void AlgoritmaX (double p[][]){
        int n = p.length;
        // ini untuk bidang XOY,XOZ,YOZ
        int end=3;
        if(n>3){
            end=4;
        }
        int loop =0;
        int f=0;
        double hasil_bid [][] = new double [4][4];
        while (loop<end){
            switch(loop){
                case 0:
                {
                    double x =
AlgoritmaX_help(cek(xoy(p)))[0];
                    double y =
AlgoritmaX_help(cek(xoy(p)))[1];
                    double z = 0;
                    double hasil =
AlgoritmaX_help(cek(xoy(p)))[2];

                    int hit = 0;
                    while(hit<n-1){
                        double zz=(p[hit][3]-
(p[hit][0]*x+p[hit][1]*y))/p[hit][2];
                        double hitung = (x*p[n-
1][0])+(y*p[n-1][1])+(zz*p[n-1][2]);

if(domain(x,y,zz,p)==true&&isNaN(zz)==false&&hitung>hasil){
                            z=zz;
                            hasil = hitung;
                        }
                        hit++;
                    }
                    hasil_bid[loop][0]=x;
                    hasil_bid[loop][1]=y;
                    hasil_bid[loop][2]=z;
                    hasil_bid[loop][3]=hasil;
                    f++;
                    break;
```

```
                }
                case 1:
                {
                    double x =
AlgoritmaX_help(cek(xoz(p)))[0];
                    double y=0;
                    double z =
AlgoritmaX_help(cek(xoz(p)))[1];
                    double hasil =
AlgoritmaX_help(cek(xoz(p)))[2];

                    int hit = 0;
                    while(hit<n-1){
                        double yy=(p[hit][3]-
(p[hit][0]*x+p[hit][2]*z))/p[hit][1];
                        double hitung = (x*p[n-
1][0])+(yy*p[n-1][1])+(z*p[n-1][2]);

if((domain(x,yy,z,p)==true&&isNaN(yy)==false)&&hitung>hasil
){
                            y=yy;
                            //System.out.println("x = "+x+"
y = "+y+" z = "+z);
                            hasil=hitung;

                        }
                        hit++;
                    }

if(domain(round(x,2),round(y,2),round(z,2),p)==true){
                        hasil_bid[loop][0]=x;
                        hasil_bid[loop][1]=y;
                        hasil_bid[loop][2]=z;
                        hasil_bid[loop][3]=hasil;
                    }

                    f++;
                    break;
                }
                case 2:
                {
                    double x = 0;
```

```
                    double y =
AlgoritmaX_help(cek(yoz(p)))[0];
                    double z =
AlgoritmaX_help(cek(yoz(p)))[1];
                    double hasil =
AlgoritmaX_help(cek(yoz(p)))[2];

                    int hit = 0;
                    while(hit<n-1){
                        double xx =(p[hit][3]-
(p[hit][1]*y+p[hit][2]*z))/p[hit][0];
                        double hitung = (xx*p[n-
1][0])+(y*p[n-1][1])+(z*p[n-1][2]);

if(domain(xx,y,z,p)==true&&isNaN(xx)==false&&hitung>hasil){
                            x=xx;
                            //System.out.println("x = "+x+"
y = "+y+" z = "+z);
                            hasil=hitung;
                        }
                        hit++;
                    }
                    hasil_bid[loop][0]=x;
                    hasil_bid[loop][1]=y;
                    hasil_bid[loop][2]=z;
                    hasil_bid[loop][3]=hasil;
                    f++;
                    break;
                }

                case 3 :
                {
                    if(p.length>0){
                        double pp[] = AlgoritmaX2(p);
                        double qq[] =
AlgoritmaX3.case_1(p);
                        double ww[] =
AlgoritmaX3.case_2(p);
                        double ee[] =
AlgoritmaX3.case_3(p);
                        double rr[] =
AlgoritmaX3.case_4(p);
```

```
                        double tt[] =
AlgoritmaX3.case_5(p);
                        double yy[] =
AlgoritmaX3.case_6(p);
                        n=p.length;
                        double uji[][] = new double [9][4];

uji[0][3]=pp[3];uji[0][0]=pp[0];uji[0][1]=pp[1];uji[0][2]=p
p[2];

uji[1][3]=qq[3];uji[1][0]=qq[0];uji[1][1]=qq[1];uji[1][2]=q
q[2];

uji[2][3]=ww[3];uji[2][0]=ww[0];uji[2][1]=ww[1];uji[2][2]=w
w[2];

uji[3][3]=ee[3];uji[3][0]=ee[0];uji[3][1]=ee[1];uji[3][2]=e
e[2];

uji[4][3]=rr[3];uji[4][0]=rr[0];uji[4][1]=rr[1];uji[4][2]=r
r[2];

uji[5][3]=tt[3];uji[5][0]=tt[0];uji[5][1]=tt[1];uji[5][2]=t
t[2];

uji[6][3]=yy[3];uji[6][0]=yy[0];uji[6][1]=yy[1];uji[6][2]=y
y[2];
                        if(n>3){
                         double uu[] =
AlgoritmaX3.case_7(p);
                         double ii[] =
AlgoritmaX3.case_8(p);

uji[7][3]=uu[3];uji[7][0]=uu[0];uji[7][1]=uu[1];uji[7][2]=u
u[2];

uji[8][3]=ii[3];uji[8][0]=ii[0];uji[8][1]=ii[1];uji[8][2]=i
i[2];
                        }

                        double hasil [] = hasil_akhir(uji);
                        hasil_bid[loop][0]=hasil[0];
                        hasil_bid[loop][1]=hasil[1];
```

```
                        hasil_bid[loop][2]=hasil[2];
                        hasil_bid[loop][3]=hasil[3];
                    }
                     else{
                        hasil_bid[loop][0]=0;
                        hasil_bid[loop][1]=0;
                        hasil_bid[loop][2]=0;
                        hasil_bid[loop][3]=0;
                     }
                f++;
                break;
                }

            }
            loop++;
        }

            double hasil_akhir [] = hasil_akhir(hasil_bid);

        System.out.println("dengan x =
"+round(hasil_akhir[0],2)+" y = "+round(hasil_akhir[1],2)+"
dan z = "+round(hasil_akhir[2],2));
        System.out.println("Maka hasil maksimalnya adalah =
"+round(hasil_akhir[3],2));
    }
```

LAMPIRAN D

```
public static double[] AlgoritmaX_help(double c[][]){

    double start=System.currentTimeMillis();

        int n = c.length;

        double gradien,gradien2;

        int pos=0;

        double hasil_return [] = new double [3];

        //========Mengurutkan susunan array berdasarkan
gradiennya=============

        if(c[n-1][0]!=0&&c[n-1][1]!=0){

        for(int i=0;i<n-1;i++){

            for(int j=i+1;j<n;j++){

                gradien=c[i][0]/c[i][1]*-1;

                gradien2=c[j][0]/c[j][1]*-1;

                if(gradien2<gradien){

                    double [] save = new double[3];

                    save[0]=c[i][0];

                    save[1]=c[i][1];

                    save[2]=c[i][2];

                    c[i][0]=c[j][0];c[j][0]=save[0];

                    c[i][1]=c[j][1];c[j][1]=save[1];

                    c[i][2]=c[j][2];c[j][2]=save[2];

                }

                else continue;
```

```
            }
        }

        for (int i = 0; i < n; i++) {
            if(c[i][2]==0){
                pos=i;
            }
        }
        int m_1 = pos-1; //dimana m1 < pos < m2
        int m_2 = pos+1;
        //TitikPotong(c,m_1,m_2);
        double x = TitikPotong(c,m_1,m_2)[0];
        double y = TitikPotong(c,m_1,m_2)[1];
        double hasil = (x*c[pos][0])+(y*c[pos][1]);
        /*System.out.println("Dengan hasil maksimalisasinya
adalah "+hasil);
         System.out.println("dengan X = "+x+ " dan Y =
"+y);*/
        hasil_return[0]=x;
        hasil_return[1]=y;
        hasil_return[2]=hasil;
     }
        else if(c[n-1][0]==0&&c[n-1][1]==0){
                hasil_return[0]=0;
```

```
                hasil_return[1]=0;

                hasil_return[2]=0;

        }

        else{

            //System.out.println("Fungsi tujuan ada yg
nol");

            if(c[n-1][0]==0){

                double y = c[0][2]/c[0][1];

                double x = 0;

                double hasil =(y*c[n-1][1]);

                //System.out.println("Dengan hasil
maksimalisasinya adalah "+hasil);

                //System.out.println("dengan X = "+x+ " dan
Y = "+y);

                hasil_return[0]=x;

                hasil_return[1]=y;

                hasil_return[2]=hasil;

            }

            else if(c[n-1][1]==0){

                double x = c[n-2][2]/c[n-2][0];

                double y=0;

                double hasil =(x*c[n-1][0]);

                //System.out.println("Dengan hasil
maksimalnya adalah "+hasil);
```

```
                //System.out.println("dengan X = "+x+ " dan
Y = "+y);
                hasil_return[0]=x;
                hasil_return[1]=y;
                hasil_return[2]=hasil;
            }
            else if(c[n-1][0]==0&&c[n-1][1]==0){
                hasil_return[0]=0;
                hasil_return[1]=0;
                hasil_return[2]=0;
            }
        }
        return hasil_return;
    }
```

LAMPIRAN E

```
public static double [] TitikPotong3(double [][] p, int i,
int j, int k){
        int n2 = p.length;

        double a = p[i][0];
        double b = p[i][1];
        double c = p[i][2];
        double d = p[i][3];

        double e = p[j][0];
        double f = p[j][1];
        double g = p[j][2];
        double h = p[j][3];

        double l = p[k][0];
        double m = p[k][1];
        double n = p[k][2];
        double o = p[k][3];
        double hs[]= new double [3];
        if(a>0){
            double t[][] = new double [2][3];
            t[0][0]=(f*a-e*b);
            t[0][1]=(g*a-c*e);
            t[0][2]=(h*a-e*d);

            t[1][0]=(m*a-l*b);
            t[1][1]=(n*a-c*l);
            t[1][2]=(o*a-l*d);


            double y =TitikPotong(t,0,1)[0];
            double z =TitikPotong(t,0,1)[1];
            double x =(d-(b*y)-(c*z))/a;
            hs[0]=round(x,10);
            hs[1]=round(y,10);
            hs[2]=round(z,10);

        }
        else if(b>0){
            double t[][] = new double [2][3];
            t[0][0]=(b*e-f*a);
```

```
            t[0][1]=(g*b-c*f);
            t[0][2]=(h*b-f*d);

            t[1][0]=(b*l-m*a);
            t[1][1]=(n*b-c*m);
            t[1][2]=(o*b-m*d);


            double x = TitikPotong(t,0,1)[0];
            double z = TitikPotong(t,0,1)[1];
            double y =(d-(a*x)-(c*z))/b;

            hs[0]=round(x,10);
            hs[1]=round(y,10);
            hs[2]=round(z,10);

        }
        else if(c>0){
            double t[][] = new double [2][3];
            t[0][0]=(c*e-g*a);
            t[0][1]=(c*f-g*b);
            t[0][2]=(h*c-g*d);

            t[1][0]=(c*l-n*a);
            t[1][1]=(c*m-n*b);
            t[1][2]=(o*c-n*d);

            double x = TitikPotong(t,0,1)[0];
            double y = TitikPotong(t,0,1)[1];
            double z = (d-(a*x)-(b*y))/c;

            hs[0]=round(x,10);
            hs[1]=round(y,10);
            hs[2]=round(z,10);
        }
       return hs;
    }
```

LAMPIRAN F

```
public static double [][] xoy(double c[][]){
        int n = c.length;
        double xy [][] = new double [n][3];

        for (int i = 0; i < n; i++) {
            if(c[i][0]==0&&c[i][1]==0){

            }
            else{
            xy[i][0]=c[i][0];
            xy[i][1]=c[i][1];
            xy[i][2]=c[i][3];

            }
        }

        return xy;
    }
    public static double [][] xoz(double c[][]){
        int n = c.length;

        double xz [][] = new double [n][3];
        for (int i = 0; i < n; i++) {
            if(c[i][0]==0&&c[i][2]==0){

            }
            else{
            xz[i][0]=c[i][0];
            xz[i][1]=c[i][2];
            xz[i][2]=c[i][3];
            }

            }

        return xz;
    }
    public static double [][] yoz(double c[][]){
        int n = c.length;

        double yz [][] = new double [n][3];
        for (int i = 0; i < n; i++) {
```

```
        if(c[i][1]==0&&c[i][2]==0){

        }
        else{
        yz[i][0]=c[i][1];
        yz[i][1]=c[i][2];
        yz[i][2]=c[i][3];

        }
    }
    return yz;
}
```

LAMPIRAN G

```
public static double [][] BuatData(int n){
        double pi=90;
       double t = Math.toRadians(pi/(n-1));
       n=n-1;
       double data[][] = new double [n+1][4];
       double r=10;


        for (int i = 0; i < n; i++) {
            double teta=i*t;
            double z=5;//ketinggian dari z
             double r2 = Math.sqrt((r*r)-(z*z));
                data[i][0]=r2*Math.cos(teta);
                data[i][1]=r2*Math.sin(teta);
                data[i][2]=z;
                data[i][3]=r*r;
        }
        data[n][0]=3000;
        data[n][1]=3000;
        data[n][2]=3000;
        data[n][3]=0;
        data[n-1][0]=0;
        data[n-1][1]=0;
        data[n-1][2]=1;
        data[n-1][3]=10;

        return data;
    }
```

LAMPIRAN H

Input

```
Int jumlah = 10;
```

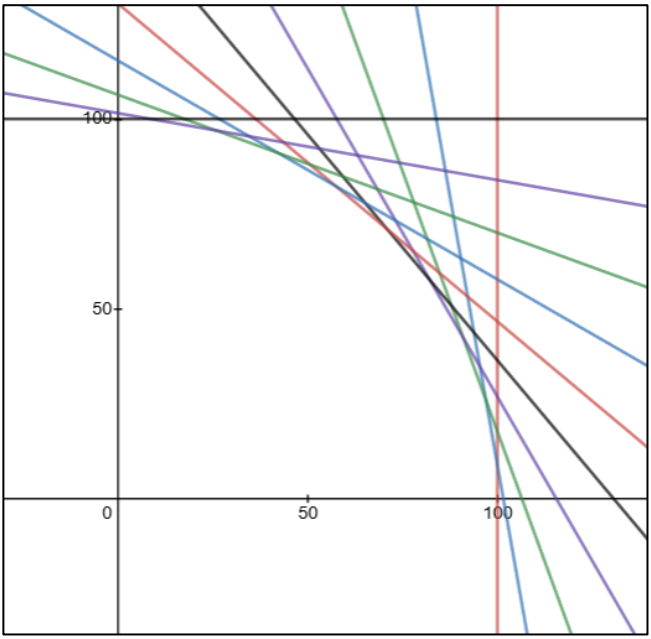
Running Program :

```
BuatData(jumlah);
```

Output :

```
run:
1. 100.0x + 0.0y = 10000.0
2. 98.4807753012208x + 17.364817766693033y = 10000.0
3. 93.96926207859084x + 34.20201433256687y = 10000.0
4. 86.60254037844388x + 49.99999999999999y = 10000.0
5. 76.60444431189781x + 64.27876096865393y = 10000.0
6. 64.27876096865394x + 76.60444431189781y = 10000.0
7. 50.000000000000014x + 86.60254037844386y = 10000.0
8. 34.20201433256688x + 93.96926207859083y = 10000.0
9. 17.364817766693034x + 98.4807753012208y = 10000.0
10. 6.123233995736766E-15x + 100.0y = 10000.0
11. 5500.0x + 6800.0y = 0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Maka, persamaan pada grafik menjadi :



Dengan menginputkan data berupa *integer* sebanyak 10 maka fungsi akan membangkitkan kendala sebanyak 10, jika menginputkan *integer* sebanyak *n* maka fungsi akan membangkitkan kendala sebanyak *n*.

BIODATA PENULIS

Penulis bernama lengkap Fachri Firmansyah Hutagalung, lahir di Jakarta, 18 Juni 1998. Merupakan putra pertama dari pasangan Zainal Hanif Hutagalung dan Lusi Enggarjanti. Jenjang pendidikan yang ditempuh oleh penulis dimulai dari SDIT PERMATA HATI (2004-2010), SMPIT AL-FIDAA (2010-2013), SMAN 02 TAMBUN SELATAN (2013-2016). Setelah lulus dari jenjang SMA, penulis melanjutkan studi ke jenjang S1 di Departemen Matematika ITS Surabaya melalui jalur kemitraan/mandiri. Selama mengikuti perkuliahan di ITS, penulis aktif dalam organisasi Himpunan Mahasiswa Matematika ITS sebagai Staff Departemen *Sport and Art Development* (2017-2018) dan Kepala Divisi *Sport and Art Development* (2018-2019). Selain organisasi, penulis juga aktif dalam kepanitiaan Olimpiade Matematika ITS (OMITS) 2018 dan 2019 sebagai panitia.