PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK MEMBANTU MENYELESAIKAN METODE LINIER GOAL PROGRAMMING

Oleh :

Nama : BETTY YULISTIOWATI

NIM : 94. 41010. 4083

Program : S1 (Strata Satu)

Jurusan : Manajemen Informatika

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

S U R A B A Y A

2000

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK MEMBANTU MENYELESAIKAN METODE LINIER GOAL PROGRAMMING

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Komputer

Oleh :

Nama : BETTY YULISTIOWATI

NIM : 94. 41010. 4083

Program : S1 (Strata Satu)

Jurusan : Manajemen Informatika

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

S U R A B A Y A

2000

Untuk menghitung suatu realita dibutuhkan suatu

model perhitungan tidak pasti,karena sesuatu yang pasti

tidak berdasarkan realita. (Einstein)

Kupersembahkan kepada :

Ayahanda Ibunda tercinta

dan Kakakku tersayang

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

UNTUK MEMBANTU MENYELESAIKAN

METODE LINIER GOAL PROGRAMMING

Telah diperiksa, diuji dan disetujui

Surabaya, mei 2000

Menyetujui :

Lucy Indrawati, S. Si	Drs. Edi Wilianto
Pembimbing II	Pembimbing I

Mengetahui :

Haryanto Tanuwijaya, S.Kom

Pembantu Ketua I

ABSTRAKSI

Pemrograman linier banyak digunakan dalam penyelesaian masalah pengalokasian sumber-sumber. Tetapi pada umumnya masih terbatas pada permasalahan dengan satu tujuan (single objective). Sedangkan pada permasalahan yang sebenarnya, optimalisasi pengalokasian sumber-sumber melibatkan lebih dari satu fungsi tujuan sekaligus (multiple objective). Untuk itu diperlukan perangkat lunak bantu untuk penyelesaian masalah tersebut.

Dalam Tugas Akhir ini digunakan metode Multiple Objective Linear Goal Programming (MOLGP) dari Multiple Criteria Optimization (MCO), yang dapat digunakan untuk permasalahan single, multiple dari Linier Goal Programming.

Dengan menggunakan tools ini akan sangat membantu pengambilan keputusan dalam pengalokasian sumber-sumber, terutama untuk permasalahan yang memiliki fungsi objective yang berprioritas.

KATA PENGANTAR

Alhambulillah, dengan mengucap rasa syukur dan berkat rakhmat Tuhan Yang Maha Kuasa penyusunan Tugas Akhir ini dapat penulis selesaikan. Tugas Akhir ini merupakan persyaratan dalam menyelesaikan program studi strata satu di Sekolah Tinggi Manajemen Informatika & Teknik Komputer Surabaya.

Untuk memenuhi persyaratan tersebut, penulis membuat tugas akhir dengan judul :

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

UNTUK MEMBANTU MENYELESAIKAN

METODE LINIER GOAL PROGRAMMING

Topik ini diangkat karena metode perhitungan yang berlaku pada banyak kasus dilakukan secara manual, sehingga waktu yang diperlukan untuk membuat sebuah keputusan menjadi lama dan terkadang kurang akurat.

Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa Tugas Akhir ini jauh dari kesempurnaan. Baik yang menyangkut aspek materi, aspek pembahasan materi maupun teknik penulisannya. Hal ini disebabkan keterbatasan pengetahuan dan penalaran dalam diri penulis.

Pada kesempatan ini, penulis menyampaikan rasa penghargaan yang tinggi serta terima kasih sebesar-besarnya kepada yang terhormat :

1. Bapak Drs. Edi Wilianto, selaku Dosen Pembimbing I, yang telah mengarahkan dan membimbing penulis dalam menyusun Tugas Akhir ini. Terima kasih yang setulus-tulusnya penulis sampaikan atas segala bantuan dan kebijaksanaan Bapak selama penyusunan Tugas Akhir ini.
2. Ibu Lucy Indrawati, S. Si, selaku dosen pembimbing II, atas arahan, bimbingan dan bantuan Ibu dalam menyusun Tugas Akhir.
3. Bapak Ketua, Pembantu Ketua, dan Staf Dosen STIKOM Surabaya, yang telah memberikan bekal ilmu kepada penulis selama menempuh kuliah.
4. Ayah, Ibu, Kakak dan Segenap keluarga, yang telah memberikan doa dan dukungan moril dan materil selama penulis menempuh jenjang pendidikan hingga menyelesaikan Tugas Akhir ini.
5. Khusus buat Lulut dan Toni yang telah mendampingi sejak awal penulisan tugas akhir ini.
6. Sahabat-sahabatku angkatan ’94 dan teman-temanku Kendangsari gang III/3, atas bantuan dan dukungan semangat yang selalu diberikan selama ini.
7. Rekan rekan di infotech yang juga mendukung selama tugas akhir ini dibuat.

Semoga Allah SWT membalas keikhlasan, ketulusan hati dan kebaikan yang telah diberikan kepada penulis dalam penyusunan Tugas Akhir. Akhirnya penulis mengharapkan, semoga Tugas Akhir ini bermanfaat bagi semua pihak. Amien.

Surabaya, 25 mei 2000

Penulis

DAFTAR ISI

halaman

ABSTRAKSI vi

KATA PENGANTAR vii

DAFTAR ISI x

DAFTAR GAMBAR xii

BAB I PENDAHULUAN 1

1.1. Latar Belakang Masalah 1

1.2. Tujuan 2

1.3. Perumusan Masalah 3

1.4. Pembatasan Masalah 3

1.5. Metodologi 4

1.5. Sistematika Penulisan 5

BAB II LANDASAN TEORI 7

2.1. Dasar Matematika 7

2.2. Linear Programming 13

2.3. Linear Goal Programming 16

BAB III ANALISIS DAN PERANCANGAN PERANGKAT LUNAK 26

3.1. Analisa Simplex 26

3.2. Analisa Linier Goal Programming 31

3.3. Kebutuhan Sistem 40

3.4. Perancangan Perangkat Lunak 41

3.5. Pembuatan Perangkat Lunak 44

3.6. Desain Input..... 47

3.7. Desain Output dan Editor Output. 52

BAB IV IMPLEMENTASI DAN EVALUASI PERANGKAT LUNAK..... 54

4.1. Implementasi dari Linier Goal Programming 54

4.2. Evaluasi Linier Goal Programming 59

4.3. Evaluasi Single Goal Programming..... 61

4.4. Evaluasi Multiple Goal Programming 65

BAB V KESIMPULAN DAN SARAN..... 78

5.1. Kesimpulan..... 78

5.2. Saran..... 79

DAFTAR PUSTAKA 80

LAMPIRAN 82

DAFTAR GAMBAR

halaman

Gambar 3.1. Komponen Finite Automata (FA) sistem 42
Gambar 3.2. FA utama sistem 43
Gambar 3.3. FA proses 43
Gambar 3.4. FA Optimasi perhitungan 44
Gambar 3.5. FA Inisialisasi dengan parsing 44
Gambar 3.6. FA Visualisasi dari Optimasi 44
Gambar 3.7. Disain Form Problem 50
Gambar 3.8. Disain Form Input 51
Gambar 3.9. Disain Form Output Iterasi 52
Gambar 3.10. Disain form output persamaan 53
Gambar 4.1. Setting variabel, fungsi objective, dan constraint 55
Gambar 4.2. Persamaan fungsi objective 56
Gambar 4.3. Entry data persamaan constraint 56
Gambar 4.4. Entry data bobot 57
Gambar 4.5. Tampilan entry grid 57
Gambar 4.6. Hasil optimasi berupa iterasi 58
Gambar 4.7. Tampilan optimasi berupa persamaan 59
Gambar 4.8. Tampilan input Single LGP 62
Gambar 4.9. Hasil perhitungan iterasi Single goal Programming 63
Gambar 4.10. Hasil dari perhitungan Single goal Programming 64

Gambar 4.11. Tampilan input Prioritas sama 67

Gambar 4.12. Tampilan iterasi dari prioritas sama 68

Gambar 4.13. Tampilan Hasil dari prioritas sama 69

Gambar 4.14. Tampilan Input Banyak prioritas tanpa Bobot 71

Gambar 4.15. Tampilan iterasi banyak prioritas tanpa Bobot 72

Gambar 4.16. Tampilan Hasil Banyak prioritas tanpa Bobot 73

Gambar 4.17. Tampilan Input banyak prioritas dan bobot 75

Gambar 4.18. Tampilan iterasi banyak prioritas dan bobot 76

Gambar 4.19. Tampilan Hasil banyak prioritas dan bobot 77

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Dari metode – metode pengambilan keputusan pada ***Operation Research***, metode Linear Programming adalah salah satu metode yang paling banyak digunakan. Linear Progaramming adalah salah satu cara untuk mengalokasikan sumber daya yang ada.

Persoalan yang dialami para produsen dan pengusaha pada awal abad 20 sangat jauh lebih sederhana dari pada persoalan yang dihadapi mereka pada saat ini. Pengalokasian sumber daya yang terbatas di antara beberapa aktivitas-aktivitas yang bersaing merupakan persoalan yang paling sering ditemui oleh mereka. Persoalan pengalokasian ini muncul manakala seseorang harus memilih tingkat aktivitas – aktivitas tertentu yang bersaing dalam hal penggunaan sumber daya yang terbatas yang dibutuhkan untuk melaksanakan aktiviatas –aktivitas tersebut. Beberapa contoh dari situasi atau persoalan di atas adalah persoalan pengalokasian fasilitas produksi, pemilihan pola pengiriman dan sebagainya.

Permasalahan di atas kebanyakan hanya mempunyai satu fungsi objective saja, oleh karena itu ***Single Objective Linear Programming (SOLP)*** sudah cukup untuk menyelesaikan masalah sederhana tersebut.

Seiring dengan perkembangan jaman, maka permasalahan *Single Objective* Berubah menjadi *Multiple Objective* (mempunyai lebih dari satu fungsi objective). Sebagai contoh, Sebuah perusahaan yang ingin memaksimumkan keuntungan, meminimumkan biaya produksi, meningkatkan pelayanan pada konsumen dan sekaligus menambah inventori. Keempat permasalahan tersebut harus diselesaikan sekaligus.

Kasus diatas dipersulit dengan suatu kondisi dimana pengambil keputusan dituntut tidak hanya memaksimumkan satu fungsi objective saja. Tetapi mempunyai beberapa tujuan dengan tingkat prioritas yang berbeda fungsi-fungsi objective yang ada. Inilah yang dihadapi oleh para produsen dan pengusaha saat ini.

Dengan diiringi perkembangan teknologi komputer, maka para produsen dan pengusaha menuntut agar masalah yang mereka hadapi baik single maupun multiple objective dapat diselesaikan dengan aplikasi yang ada agar hasil yang diperolah lebih cepat, tepat dan akurat.

1.2. Tujuan

Tujuan dari Tugas Akhir ini adalah untuk merancang dan membangun suatu perangkat lunak bantu yang dapat menyelesaikan permasalahan Goal Programming secara cepat, tepat dan akurat. Sehingga dapat digunakan oleh pemakai sebagai salah satu alat bantu untuk menghemat waktu dalam memecahkan masalah yang membutuhkan perhitungan optimasi secara matematis dengan cepat dan akurat, dalam penggunaan beberapa fungsi obyektif (kriteria) yang ada tanpa harus mengesampingkan fungsi obyektif (kriteria) yan lainnya.

1.3. Perumusan Masalah

Pada kasus Multiple Objective sering ditemui para pengambil keputusan hanya menetukan opimal solution untuk satu fungsi objective saja dan mengabaikan fungsi objective yang lain yang harus diselesaikan secara bersamaan. Maka permasalahan yang ada dalam menyelesaikan adalah :

a. Bagaimana dapat membuat Suatu aplikasi yang dapat menyelesaikan Perhitungan perhitungan yang menggunakan metode simplex itu cenderung banyak memakan waktu dan ketelitian. Sehingga menghambat pekerjaan yang akan dilakukan.

b. Apakah aplikasi ini nantinya dapat dikembangkan penggunaannya pada masalah metode metode simplex sederhana, karena simplex menjadi dasar perhitungan pada linier goal programming.

c. Bagaimana membuat sistem yang *user friendly* agar dapat dengan mudah digunakan oleh orang awam.

1.4. Pembatasan Masalah

Dalam Tugas Akhir ini perangkat lunak yang dibuat mempunyai batasan-batasan sebagai berikut :

A. Untuk menyelesaikan kasus fungsi objektif tunggal digunakan metode Simplex

B. Untuk menyelesaikan kasus fungsi objektif jamak dalam hal ini digunakan Goal Programming dengan bantuan modifikasi metode

Simplex. Fungsi obyektif jamak ini dibatasi sampai maximal 7 fungsi obyektif. variabel-variabel utama yang digunakan adalah sebagai berikut: Z=fungsi obyektif (terdiri dari $\mathbf{d_p^+}$= fungsi obyektif deviasi maximize, $\mathbf{d_q^-}$ =fungsi obyektif minimize), k=menunjukkan urutan(1,2,3,4), y =skala prioritas, j=bilangan (1,2,3…), x=variabel dari Z, a=konstanta dari x, P=variabel dari hasil hitung fungsi obyektif yang berprioritas.

C. Input dari user harus mengisikan variabel-variabel pada suatu tabel yang sesuai dengan pengaturan yang ada

D. Output dari perangkat lunak ini adalah tabel Iterasi Simplex.

E. Hasil dari aplikasi ini dapat dilihat pada format persamaan matematika standard.

1.5. Metodologi

Dalam menyusun Tugas Akhir ini dilakukan langkah-langkah sebagai berikut:

A. Menetukan permasalahan apa saja yang akan dibahas mengenai goal programming.

B. Studi literatur mengenai teori Goal Programming dan pemrograman DELPHI berupa studi pustaka, wawancara dan lain sebagainya

C. Perancangan perangkat lunak yang telah dibuat tentang goal programming, bagaimana tampilan dari rancangan input outputnya

D. Implementasi hasil rancangan yang telah dibuat bagaimana suatau masalah dapat diselesaikan dengan perangkat lunak yang tealah dibuat itu.

E. Pengujian serta perbaikan-perbaikan perangkat lunak. Dari hasil pengujian dilakukan perancangan pada beberapa bagian, kemudian diimplementasikan serta dilakukan pengujian lagi. Demikian seterusnya sampai hasil yang diperoleh dirasa cukup memuaskan.

F. Penulisan kesimpulan dari segala permasalahan yang adan dan solusi terbaiknya.

G. Penulisan naskah tugas akhir.

1.5. Sistematika Penulisan

Dalam buku Tugas Akhir ini, pembahasan mengenai perangkat lunak yang dibuat dalam 5 (lima) bab dengan sistematika sebagai berikut :

BAB 1 : Pendahuluan, yang membahas tentang latar belakang yang menjadi dasar pertimbangan timbulnya permasalahan, permasalahan yang disoroti, tujuan pembuatan Tugas Akhir ini dan batasan permasalahan, metodologi penyusunan Tugas Akhir serta sistematika pembahasan dalam buku Tugas Akhir ini.

BAB 2 : Landasan Teori, yang membahas tentang teori yang mendasari perancangan dan pembuatan perangkat lunak teori himpunan menjadi dasar dalam mempelajari linier programming, metode simplex, yang kemudian baru meluas menjadi Goal programming.

BAB 3 : Analisis dan Perancangan Perangkat Lunak, Analisa dari metode simplex yang berisi langkah langkah metode ini dan ketentuannya, linier

goal programming berupa jenisnya dan metode yang digunakan sekaligus juga ketentuannya.

BAB 4 : Implementasi dan Evaluasinya Perangkat Lunak Bantu, yang membahas tentang langkah-langkah dalam melakukan Implementasi pada aplikasi Linier Goal Programming. Baru kemudian bagaimana evaluasi dari implementasinya pada yang terdiri dari evaluasi linier goal programming, evaluasi single goal programming, dan evaluasi multiple goal programming.

BAB 5 : Kesimpulan dan Saran, yang merupakan bab terakhir dari laporan Tugas Akhir ini adalah kesimpulan yang berisi apakah tugas akhir ini dapat memenuhi tujuannya dalam mengaplikasikan Linier Goal Programming yang diperoleh dari pembuatan Tugas Akhir ini, serta saran untuk pengembangan Tugas Akhir ini.

BAB II

LANDASAN TEORI

Pada bab ini akan dibahas tentang teori yang melandasi metode Multiple Objective Linier Goal Programming.

2.1. Dasar Matematika

Sub bab ini mengulas tentang dasar matematika yang digunakan untuk mempelajari Multiple Criteria Optimization.

2.1.1. Teori himpunan

Himpunan adalah sekumpulan bilangan atau lainnya yang mempunyai jenis atau sifat yang sama. Suatu himpunan ditandai dengan { dan }.

Contoh :

A = {a,g,h,b}

B = {8,10,12,14,...}

C = { x | x = 2^n, n adalah bilangan genap, 1 £ n £ 5 }

D = {y | y adalah konsonan}

Ada dua cara untuk menulis bentuk matematika dari himpunan, yaitu :

Roster, yaitu dengan mendaftar semua isi dari himpunan.

Defining-property, yaitu dengan cara menentukan langsung properti dari himpunan tersebut.

Dari contoh penulisan himpunan di atas, A dan B merupakan Roster, dan C dan D menggunakan cara Defining-property. Î merupakan simbol dari anggota himpunan. Jika X merupakan anggota dari himpunan X maka ditulis sebagai berikut :

x Î X

Jika x bukan anggota dari himpunan X, maka ditulis sebagai berikut : xÏ X.

Himpunan kosong adalah himpunan yang tidak mempunyai anggota sama sekali. Suatu himpunan kosong ditulis sebagai berikut :

A = {} atau A = f

Jangan sampai terjadi penggandaan penggunaan seperti {f}, karena himpunan tersebut berarti beranggotakan suatu himpunan kosong.

Ada 3 macam jenis operasi himpunan, yaitu :

A. Union (gabungan)

B. Intersection (irisan)

C. Difference

Jika A dan B adalah suatu himpunan, maka A gabungan B (AÈB) adalah himpunan yang semua elemennya adalah gabungan elemen himpunan A dan B.

A irisan B (AÇB) adalah himpunan yang elemennya ada di himpunan A dan B.

A difference B (A-B) adalah himpunan yang elemennya ada di A tapi tidak di B.

Contoh :

A = {1,2,3}

B = {2,5,1}

I = {a,b,c,d}

A È B = {1,2,3,5}

A Ç B = {1,2}

(I È {e})-{c} = {a,b,d,e}

A-B = {3}

2.1.2. Subset, superset dan set equality

Himpunan F merupakan *subset* himpunan G jika dan hanya jika x Î F merupakan anggota dari G (dengan kata lain x Î G). Artinya F merupakan subset G jika semua anggota F merupakan anggota dari G, ditulis sebagai berikut :

F Ì G

Superset merupakan kebalikan dari subset, ditulis sebagai berikut

G É F , G merupakan superset dari F.

Himpunan kosong merupakan subset dari semua himpunan.

Dua himpunan dikatakan *equal* jika dan hanya jika anggota kedua himpunan tersebut sama. Dengan kata lain, A=B jika A Ì B dan B Ì A.

Contoh : A = {5,2,5,7}

B = {7,2,5}

C = {5,2,7}

Maka A = B = C karena posisi dan duplikasi tidak diperhitungkan.

2.1.3. Disjoint set dan family

Dua himpunan dikatakan disjoint jika tidak ada satu anggota pun yang menjadi anggota di himpunan yang lain.

A = {7}

B = {10,11}

C = {7,8,9,10}

Sebuah *family* adalah himpunan yang anggotanya adalah himpunan anggotanya itu sendiri. Himpunan family dari {11,12} adalah

Y = { {11,12},{11},{12},f }

2.1.4. Index set dan cardinality

Operasi Union dan Intersection dapat digeneralisasikan menjadi himpunan lain.

Contoh : A_1 = {1,2,5,7} A_2 = {3,4,5} A_3 = {1,5,7,8}

A_4 = {2,5,7} A_5 = {1,4,5}

Operasi Union dan intersection secara berturut-turut dari contoh di atas dapat ditulis sebagai berikut :

$$\bigcup_{i=1}^{5} Ai = \{1,2,3,4,5,7,8\}$$

$$\bigcap_{i=1}^{5} Ai = \{5\}$$

Jika digunakan *index set* I = {1,2,3,4,5}, operasi union dan intersection secara berturut-turut dari 5 himpunan di atas dapat ditulis sebagai berikut :

$$\bigcup_{i \in I} Ai = \{1,2,3,4,5,7,8\}$$

$$\bigcup_{i \in I} Ai = \{5\}$$

Jika J adalah sebuah himpunan, maka cardinality dari J (ditulis $|J|$) adalah jumlah elemen yang menjadi anggota J.

Misal :

Union A_1, A_4, A_5 adalah

$$\bigcup_{i \in I} Ai = \{1,2,4,5,7\}$$

I = {1,4,5}

maka $|I|$ = 3.

2.1.5. Partisi

Jika A adalah sebuah himpunan, maka subset $\{A_1, A_2, A_3, ..., A_q\}$ dikatakan sebagai partisi dari A jika dan hanya jika

(i) $A = \bigcup_{i=q}^{q} Ai$

dan

(ii) $Ai \cap Aj = \emptyset$ semua $i \neq j$.

2.1.6. Pemetaan dan fungsi

Jika $f : A \rightarrow B$ berarti f memetakan setiap elemen dari A menjadi setiap elemen dari B.

A disebut sebagai *domain* (daerah asal) dan B disebut sebagai *co-domain* (daerah tujuan).

Setiap elemen $a \in A$ dari daerah domain, elemen $b \in B$ yang bersekutu dengan a disebut *image* dari a dan ditulis f(a).

Jika f memetakan $a \in B$ ke $b \in B$, maka a adalah sebuah *inverse image* dari b.

2.1.7. Ordered n-tuple dan cartesian product

Jika n adalah sebuah bilangan positif, maka tuple-n-terorde *(ordered-n-tuple*) adalah sebuah urutan n bilangan riil $(a_1, a_2, ..., a_n)$. Himpunan semua tuple-n-terorde dinamakan ruang-n dan dinyatakan dengan R^n.

Jika A dan B adalah dua buah himpunan, *Cartesian Product* A dan B (ditulis : A x B) adalah himpunan semua *ordered pair* (a,b) dimana a Î A dan b Î B.

2.1.8. Matrix

Matriks adalah suatu susunan segi empat siku-siku dari bilangan-bilangan. Bilangan-bilangan dalam susunan tersebut dinamakan entri dalam matriks. Matriks ditulis di antara tanda [dan].

Ukuran matriks dijelaskan dengan menyatakan banyaknya baris (garis horisontal) dan banyaknya kolom (garis vertikal) yang terdapat dalam matriks tersebut. Sebuah matriks dengan n baris dan n kolom dinamakan matriks kuadrat berode *n* (*square matrix of order n*), dan entri-entri a_{11}, a_{22}, ..., a_{nn} dikatakan berada pada diagonal utama.

2.2. Linear Programming

Linear Programming merupakan suatu model yang dapat digunakan dalam pemecahan masalah pengalokasian sumber-sumber yang terbatas secara optimal. Masalah tersebut timbul jika kita diharuskan untuk memilih dan menentukan tingkat setiap kegiatan yang dilakukannya, dimana masing-masing kegiatan membutuhkan sumber daya yang sama sedangkan jumlahnya terbatas.

Dalam memecahkan masalah diatas linear programming menggunakan model matematis. Sebutan “linear” berarti bahwa semua fungsi-fungsi matematis yang disajikan dalam model ini haruslah fungsi-fungsi linear. Kata “programming” jangan dikacaukan dengan “computer progaramming”, seperti yang sering kita dengar,

walaupun secara mendasar keduanya sering digunakn untuk perencanaan. Jadi, linear programming menyangkut perencanaan-perencanaan untuk mencapai hasil yang "optimal", yaitu suatu hasil yang mencerminkan tercapainya sasaran tertentu yang paling baik (menurut model matematis) di antara alternatif-alternatif yang mungkin, dengan menggunakan fungsi linear.

2.2.1. Model linear programming

Model matematis perumusan masalah umum pengalokasian sumber daya untuk berbagai kegiatan, disebut sebagai model *linear programming* (LP). Model LP ini merupakan bentuk dan susunan dalam menyajikan masalah-masalah yang akan dipecahkan dengan teknik LP. Dalam model LP dikenal 2 macam "fungsi", fungsi tujuan (*objective function*) dan fungsi batasan (*constraint*). Objective adalah fungsi yang menggambarkan sasaran di dalam permasalahan LP yang berkaitan dengan pengaturan secara optimal sumber daya, untuk memperoleh keuntungan maksimal atau biaya minimal. Pada umumnya nilai yang akan dioptimalkan dinyatakan sebagai Z. Sedangkan constrain merupakan bentuk penyajian secara matematis, batasan-batasan kapasitas yang tersedia yang akan dialokasikan secara optimal ke berbagai kegiatan.

Model matematis yang digunakan untuk mengemukakan suatu masalah LP adalah sebagai berikut :

Max $Z = c_1x_1 + c_2x_2 + c_3x_3 + \ldots. + c_nx_n$

Constraint $a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots. + a_{mn}x_n \; £ \; b_m$

Di mana :

m = macam batasan-batasan sumber atau fasilitas yang tersedia

n = macam kegiatan-kegiatan yang menggunakan sumber atau fasilitas tersebut.

i = nomor setiap macam sumber fasilitas yang tersedia

j = nomor setiap macam kegiatan yang menggunakan sumber atau fasilitas yang tersedia

x_j = tingkat kegiatan ke j

a_{ij} = banyaknya sumber I yang diperlukan untuk menghasilkan setiap unit keluaran (output) kegiatan

Constraint dapat dikelompokkan menjadi dua macam, yaitu :

Fungsi batasan fungsional, yaitu fungsi-fungsi batasan sebanyak m (yaitu $a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n$).

Fungsi batasan non-negatif (*non-negative constraints*), yaitu fungsi-fungsi batasan yang dinyatakan dengan $x_I \geq 0$.

2.2.2. Metode simplex

Metode Simplex ialah suatu metode yang secara sistematis dimulai dari suatu pemecahan dasar yang fisibel ke pemecahan dasar yang fisible (feasible) lainnya dan ini dilakukan berulang-ulang (dengan jumlah ulangan yang terbatas) sehingga akhirnya tercapai suatu titik pemecahan dasar yang optimum dan pada setiap step

menghasilkan suatu nilai dari fungsi tujuan yang selalu lebih besar (lebih kecil) atau sama dari step sebelumnya. Apabila suatu masalah LP hanya mengandung 2 variabel saja, maka akan dapat diselesaikan dengan metode grafik. Tetapi bila melibatkan lebih dari dua variabel maka metode grafik tidak dapat digunakan lagi, sehingga diperlukan Metode Simplex. Metode Simplex merupakan suatu cara yang lazim dipakai untuk menentukan kombinasi optimal dari tiga variabel atau lebih.

2.3. Linear Goal Programming

LGP merupakan pengembangan Linear Programming (LP). Perbedaan utama antara LGP dan LP terletak pada struktur penggunaan fungsi tujuan. dalam LP fungsi tujuannya hanya mengandung satu tujuan, sementara dalam LGP semua tujuan apakah satu atau beberapa digabungakan dalam sebuah fungasi tujuan. Ini dapat dilakukan dengan mengekspresikan tujuan itu dalam bentuk sebuah kendala (*goal constraint*), memasukkan suatu variabel simpangan (*deviational variable*) dalam kendala itu untuk mencermiankan seberapa jauh tujuan itu dicapai, dan menggabungkan variabel simpangan dalam fungsi tujuan. Dalam LP tujuannya bisa maksimisasi atau minimisasi, sementara dalam LGP tujuannya adalah meminimumkan penyimpangan- penyimpangan dari tujuan-tujuan tertentu. Ini berarti semua masalah LGP adalah masalah minimisasi.

Karena penyimpangan-penyimpangan dari tujuan-tujuan itu diminimumkan, sebuah model LGP dapat menangani beraneka ragam tujuan dengan dimensi atau satuan ukuran yang berbeda. Tujuan-tujuan yang saling bertabrakan juga dapat diselesaikan. Jika terdapat banyak tujuan, prioritas atau urutan ordinalnya dapat

ditentukan, dan proses penyelesaian LGP akan berjalan sedemikian rupa sehingga tujuan dengan prioritas tertinggi dipenuhi sedekat mungkin sebelum memikirkan tujuan-tujuan dengan prioritas yang lebih rendah. Jika LP berusaha mengidentifikasi solusi optimum (*optimal solution*) dari suatu himpunan solusi layak, LGP mencari titik yang paling memuaskan dari sebuah persoalan dengan beberapa tujuan sekali lagi LGP ingin meminimumkan penyiampangan-penyimpangan dari tujuan-tujuan dengan mempertimbangkan hierarki prioritas.

2.3.1. Terminologi LGP

Berikut ini adalah beberapa definisi dari beberapa istilah dan lambang yang biasa digunakan dalam LGP.

Decision variables : seperangkat variabel yang tak diketahui (dalam model LGP dilambangkan dengan x_j , di mana $_j$ =1,2,…,*n*) yang akan dicari nilainya.(variabel keputusan.

Right hand side values (RHS) : nilai-nilai yang biasanya menunjukkan ketersediaan sumber daya (dilambangkan dengan b_i) yang akan ditentukan kekurangan atau kelebihan penggunaannya. (Nilai sisi kanan).

Goal : keinginan untuk meminimumkan angka penyimpangan dari suatu nialai RHS pada suatu Goal consraint tertentu. (Tujuan).

Preemtive priority factor : suatu sistem urutan (yang dilambangkan dengan P_k, dimana k = 1,2,…,K dan K menunjukkan banyaknya tujuan dalam model) yang memungkinkan tujuan-tujuan di susun secara ordinal dalam model LGP. Sistem

urutan itu menempatkan tujuan-tujuan dalam susunan dengan hubungan seperti berikut :

$P_1 > P_2 >>> P_k$

P_1 merupakan tujuan yang paling penting

P_2 merupakan tujuan yang kurang penting dan seterusnya.

Deviational variables : variabel-variabel yang menunjukkan kemungkinan penyimpangan negatif dari suatu nilai RHS kendala tujuan (dalam model LGP dilambangkan dengan d_I^- dimana I = 1,2,…,m dan m adalah banyaknya kendala tujuan dalam model) atau penyimpangan positif dari suatu nilai RHS (dilambangkan dengan d_I^+**,** variabel-variabel ini serupa dengan *slack variable* dalam LP (variabel simpangan).

Differential weight : timbangan matematik yang diekspresikan dengan angka c*ardinal* (dilambangkan dengan w_{ki} dimana k = 1,2,…,K; *i* = 1,2,…,m) dan digunakan untuk membedakan variabel simpangan *i* di dalam suatu tingkat prioritas *k* (bobot)

Technological coefficient : nilai-nilai numerik (dilambangkan dengan a_{ij}) yang menunjukkan penggunaan nilai b_i perunit untuk menciptakan x_j (koefisien teknologi).

2.3.2. Unsur – unsur LGP

Setiap model LGP paling sedikit terdiri atas tiga komponen, yaitu sebuah fungsi tujuan, kendala-kendala tujuan, dan kendala non negatif.

A. Fungsi tujuan

Ada tiga jenis fungsi tujuan dalam Linier Goal Programming yang digunakan dalam tugas akhir ini, yaitu :

Minimumkan $Z = \sum_{i=1}^{m} d_I^- + d_i^+$

Minimumkan $Z = \sum_{i=1}^{m} P_k (d_I^- + d_I^+)$ untuk k = 1,2,…, K

Minimumkan $Z = \sum_{i=1}^{m} W_{ki} P_k (d_I^- + d_I)$ untuk k = 1,2,…, K

Fungsi tujuan yang pertama digunakan jika variabel simpangan dalam suatu masalah tidak dibedakan menurut prioritas atau bobot. Fungsi tujuan kedua digunakan dalam suatu masalah diamana urutan tujuan-tujuan diperlukan, tetapi variabel simpangan didalam setiap tingkat prioritas memiliki kepentingan yang sama.

Perlu diperhatikan bahwa dalam model LGP tidak ditemukan variabel keputusan pada fungsi tujuan. Kita masih mencari, seperti yang dilakukan model LP, nilai x_j yang tidak diketahui, tetapi akan melakukannya secara tidak langsung melaui minimisasi simpangan negatif dan positif dari nilai RHS kendala tujuan. LP mencari nilai solusi x_j secara langsung melalui minimisasi penyimapangan-penyimpangan dari nilai RHSnya.

B. Kendala tujuan

Ada enam jenis kendala tujuan yang berlainan. Maksud dari setiap jenis kendala itu ditentukan oleh hubungannya dengan fungsi tujuan.

Persamaan No	Kendala Tujuan	Variabel Simpangan Dalam Fungsi Tujuan	Kemungkinan Simpangan	Penggunaan Nilai RHS yang Diinginkan
1	$a_{ij}x_j + d_i^- = b_i$	d_i^-	Negatif	$= b_I$
2	$a_{ij} - d_i^+ = b_i$	d_i^+	Positif	$= b_i$
3	$a_{ij}x_j + d_i^- - d_i^+ = b_i$	d_i^-	Neg. dan pos.	b_i atau lebih
4	$a_{ij}x_j + d_i^- - d_i^+ = b_i$	d_i^-	Neg. dan pos.	b_i atau kurang
5	$a_{ij}x_j + d_i^- - d_i^+ = b_i$	d_i^- *dan* d_i^+	Neg. dan pos.	$= b_i$
6	$a_{ij}\, x_j - d_i^+ = b_i$	d_i^+ *(artf.)*	Tidak ada	Pas = b_i

Terlihat pada tabel diatas bahwa setiap jenis kendala tujuan harus punya satu atau dua variabel simpangan yang ditempatkan pada fungsi tujuan. Dimungkinkan adanya kendala-kendala yang tidak memiliki variabel simpangan. Kendala ini sama seperti kendala-kendala persamaan linier.

B.1. Persamaan pertama pada tabel diatas maknanya serupa dengan kendala pertidaksamaan £ dalam masalah program linier maksimasisasi.

B.2. Persamaan kedua, maknanya serupa dengan kendala pertidaksamaan ³ pada masalah program linier minimisasi.

B.3. Persamaan ketiga, **keempat dan kelima** semuanya memperbolehkan penyimpangan dua arah, tetapi persamaan kelima mencari penggunaan sumber daya yang diinginkan sama dengan b_i. Ini serupa dengan kendala persamaan dalam LP, tetapi tidak menempel pada solusi karena dimungkinkan dengan adanya penyimpangan negatif dan positif.

B.4. Persamaan keenam, dipakai jika kendala persamaan dianggap perlu dalam perumusan model LGP dengan menempatkan sebuah artificial variabel.

C. Kendala non negatif

Seperti dalam LP, variabel variabel model LGP biasanya bernilai lebih besar atau sama dengan nol. Semua model LGP terdiri dari variabel simpangan dan variabel keputusan, sehingga pernyaaan non negatif dilambangkan sebagai : x_i, d_I^-, di^+ ³ 0.

D. Kendala struktural

Disamping ketiga komponen yang telah disebutkan itu, dalam model LGP kadang-kadang terdapat komponen yang lain yaitu kendala struktural artinya kendala-kendala lingkungan yang tidak berhubungan dengan langsung dengan tujuan-tujuan masalah yang dipelajari. Variabel simpangan tidak dimasukkan dalam kendala ini, karena itu, kendala ini tidak disertakan dalam fungsi tujuan.

2.3.3. Asumsi model LGP

Sebelum merumuskan model. Perlu diketahui bahwa model LGP memerlukan sejumlah asumsi. Jika dalam membuat model dari suatu masalah tertentu asumsi-asumsi tersebut tidak dipenuhi, maka LGP bukan merupakan model yang cocok untuk masalah yang sedang dipelajari. Jadi asumsi model membatasi penerapan LGP. Asumsi-asumsi berikut harus diingat agar penerapan model LGP bermanfaat.

A. Addivitas dan linieritas

Diasumsikan bahwa penggunaan b_i yang ditentukan oleh a_{ij} harus tetap benar tanpa memperhatikan nilai solusi x_j yang dihasilkan. Artinya, Left Hand Side values (LHS) dari kendala tujuan harus sama dengan nilai nilai RHS. Dalam kehidupan sehari hari, Hubungan *synergistik* dapat menyebabkan penyimpangan asumsi ini. Suatu contoh ketika seseorang ditempatkan dalam suatu lingkungan yang kompetitif akan lebih produktif dibanding jika prestasi seseorang diukur dari lingkungan yang tidak kompetitif. Prosedur model lain, seperti Stochastic Goal programing, cocok untuk model jenis persoalan ini.

B. Divisibilitas

Diasumsikan bahwa nilai-nilai x_j, d_i

C. Terbatas

Diasumsikan bahwa nilai x_j, d_i dan d_i^+ yang dihasilkan harus terbatas. Artinya, tidak dapat memiliki nilai variabel keputusan, sumber daya, atau penyimpangan tujuan yang tak terbatas. Segalanya dalam dunia ini terbatas.

D. Kepastian atau periode waktu statis

Diasumsikan bahwa parameter model LGP seperti a_{ij}, b_i, P_k, dan W_{ki} diketahui dengan pasti mereka akan tetap statis selama periode perencanaan dimana hasil model digunakan.

2.3.4. Perumusan masalah LGP

Agar mengerti bagaimana merumuskan suatu maslah LGP, marilah kita melihat prosedur perumusan. Kemudian kita akan menerapkan prosedur itu pada bebarapa situasi persoalan yang berlainan.

Prosedur perumusan

Perumusan suatu maslah LGP sangat mirip dengan perumusan sebuah masalah LP. Penjelasan variabel keputusan x_J, Koefisien teknologi a_{ij}, dan nilai sisi kanan b_i, diperlukan baik pada LP maupun LGP. Langkah langkah perumusan LGP meliputi beberapa tahap :

1. **Tentukan variabel keputusan**. Disini kuncinya adalah menyatakan dengan jelas variabel keputusan yang tak diketahui. Makin tepat definisi akan makin mudah pekerjaan permodelan yang lain.

2. **Nyatakan sistem kendala.** Kuncinya pertama adalah menetukan nilai-nilai sisi kanan dan kemudian menentukan koefisien teknologi yang cocok dan variabel keputusan yang diikut sertakan dalam kendala. Juga perhatikan jenis penyimpangan yang diperbolehkan dari nilai RHS. Jika penyimpangan diperrbolahkan dalam dua arah, tempatkan kedua variabel simpangan pada kendala itu. Jika penyimpangan hanya diperbolehkan pada satu arah, tempatkan satu variabel simpangan yang tepat pada kendala yang bersangkutan.

3. **Tentukan prioritas utama.** Kuncinya disini adalah membuat urutan tujuan tujuan. Biasanya urutan tujuan merupakan pernyataan preferensi individu. Jika persoalannya tidak memiliki urutan tujuan, lewati langkah ini dan kemudian ke langkah berikutnya.

4. **Menentukan bobot.** Di sini kuncinyan adalah membuat urutan di dalam suatu tujuan tertentu. Jika tidak diperlukan lewati langkah ini.

5. **Nyatakan fungsi tujuan .** Di sini kuncinya adalah memilih variabel simpangan yang benar untuk dimasukkan dalam fungsi tujuan. Untuk menyakinkan penggunaan nilai RHS yang diinginkan sesuai adalah konsisten dengan keperluan persoalan. Kedua, tambahkan prioritas dan bobot yang tepat jika diperlukan.

6. **Nyatakan keperluan non negatif.** Langkah ini merupakan bagian resmi dari perumusan masalah LGP.

Prosedur formulasi ini merupakan salah satu pendekatan yang mungkin bermanfaat dalam perumusan model LGP. Kita dapat memperbaiki atau mengubah prosedur itu sesuai dengan kebutuhan masing-masing. Untuk membantu bagaimana masalah LGP berikut ini disajikan persolan-persoalan serta formulasinya.

BAB III

ANALISIS DAN PERANCANGAN

PERANGKAT LUNAK

3.1. Analisa Simplex

Seperti yang telah dijelaskan pada landasan teori apa yang dimaksud dengan metode simplex, maka pada bab ini kita akan membahas lebih lanjut tentang bagaimana langkah langkah penggunaan metode simplex.

3.1.1. Langkah-langkah metode simplex

Mula-mula akan dilakukan penjelasan langkah-langkah penyelesaian persoalan yang dapat di formulasikan dalam bentuk standar, karena bentuk ini yang paling mudah diselesaikan. Selanjutnya akan dibicarakan penyelesaian masalah yang formulasinya menyimpang dari bentuk standar.

Langkah 1 : Merubah fungsi objektif dan constraint

Fungsi objektif diubah menjadi fungsi implisit, artinya semua $c_j x_{ij}$ digeser ke kiri

Pada bentuk standar, semua constraint mempunyai tanda £ . Pertidaksamaan ini harus dirubah menjadi persamaan. Caranya dengan menambahkan *slack variable*.

Langkah 2 : Menyusun persamaan-persamaan di dalam tabel.

Langkah 3 : Memilih kolom kunci

Kolom kunci adalah kolom yang merupakan dasar untuk merubah tabel. Pilihlah kolom yang mempunyai nilai pada garis fungsi objektif yang bernilai negatif dengan angka terbesar.

Langkah 4 : Memilih baris kunci

Baris kunci adalah baris yang merupakan dasar untuk merubah tabel. Untuk itu terlebih dahulu dicari index tiap-tiap baris dengan cara membagi nilai pada kolom RHS (Right Hand Side) dengan nilai yang sebaris dengan kolom kunci.

Pilihlah baris yang mempunyai index positif dengan angka terkecil.

Langkah 5 : Merubah nilai-nilai baris kunci

Nilai baris kunci dirubah dengan cara membaginya dengan angka kunci.

Ganti variabel dasar pada baris itu dengan variabel yang terdapat di bagian atas kolom kunci.

Langkah 6 : Merubah nilai-nilai selain pada baris kunci

Baris baru = baris lama - {(koefisien pada kolom kunci) x nilai baru baris kunci}

Langkah 7 : Melanjutkan perbaikan-perbaikan/perubahan-perubahan

Ulangi langkah-langkah perbaikan mulai langkah 3 sampai langkah ke 6 untuk memperbaiki tabel-tabel yang telah dirubah/diperbaiki nilainya. Perubahan baru berhenti setelah pada baris pertama (fungsi objektif) tidak ada yang bernilai negatif.

3.1.2. Ketentuan-ketentuan tambahan

Masalah yang dihadapi kadang-kadang dapat menghasilkan dua kolom kunci, dua baris kunci dan *multiple solution*. Hal ini akan dibicarakan satu persatu.

Terdapat lebih dari satu kolom kunci negatif dengan angka terbesar.

Kalau pada baris fungsi objektif terdapat lebih dari satu kolom yangmempunyai nilai negatif yang angkanya terbesar, maka ada dua kolom yang bisa dipilih menjadi kolom kunci. Untuk mengatasi hal ini kita pilih salah satu diantara dua pilihan itu secara sembarang, akan menghasilkan keputusan yang sama.

Dua baris atau lebih mempunyai indeks positif terkecil

Kalau ada dua baris atau lebih yang mempunyai nilai positif terkecil, maka ada beberapa baris yang dapat terpilih di sebagai kunci. Untuk mengatasi hal ini dapat dipilih baris kunci secara bebas diantara keduanya.

Kenaikan nilai Z tidak terbatas

Nilai Z (fungsi objektif) suatu permasalahan dapat ditambah terus bila paling tidak ada satu kegiatan yang tidak ada constraintnya. Kalau di dalam linear programming dilihat hal-hal semacam ini, maka tidak perlu dilanjutkan, cukup disebutkan bahwa kenaikan nilai Z dapat tidak terbatas. Disamping itu ada baiknya

pula apabila diteliti lagi formulasi masalahnya, sebab hal ini dapat pula terjadi karena kesalahan dalam formulasi.

Multiple Optimal Solution

Di dalam metode Simplex tabel, untuk mengetahui apabila suatu masalah linear programming bersifat multiple solution atau tidak, dilihat dari baris fungsi objektif pada tabel terakhir. Apabila di dalam baris itu terdapat paling tidak satu kolom variabel dasar yang mempunyai nilai 0 maka masalah itu bersifat multiple solution.dengan kata lain masalah itu akan menghasilkan paling tidak 2 alternatif yang mempunyai nilai Z sama.

3.1.3. Penyimpangan-penyimpangan dari bentuk standar

Di depan telah diuraikan bahwa masalah-masalah yang dihadapi tidak selalu dapat diformulasikan menjadi bentuk standar. Berikut ini akan dibahas cara-cara mengatasi penyimpangan-penyimpangan dari bentuk standar agar bisa diselesaikan dengan metode Simplex.

A. Constraint dengan tanda "="

Kalau suatu batasan memakai tanda kesamaan, maka cara mengatasinya dengan menambahkan variabel buatan (artificial variable). Karena adanya variabel buatan, maka fungsi objektif harus disesuaikan dengan menambahkan bilangan M. Bilangan M mempunyai nilai sangat besar tetapi tidak tak terhingga, sehingga nilai Z maksimum bisa diperoleh jika variabel buatan tadi bernilai 0. Persamaan baru yang telah ditambah dengan bilangan *M* tersebut tidak memungkinkan penggunaan

metode Simplex, sebab seperti telah diuraikan di depan, nilai setiap variabel dasar pada persamaan ini harus sebesar 0, padahal variabel dasar merupakan variabel dasar pada tabel permukaan. Oleh karena itu persamaan tersebut dirubah dengan cara menguranginya dengan *M* dikalikan dengan baris batasan yang bersangkutan. Kemudian fungsi tujuan yang telah dirubah ini dimasukkan ke dalam tabel kemudian baru diselesaikan.

B. Minimisasi

Fungsi objektif dari permasalahan linier programming yang bersifat minimisasi, harus ditambah menjadi maksimasi, agar sesuai dengan bentuk standar, yaitu maksimisasi. Caranya adalah dengan mengganti tanda positif dan negatif pada fungsi tujuan,sebagai berikut :

min { $c_n x_n = Z$ }

sama dengan

max { $- c_n x_n = -Z$ }

C. Constraint bertanda "³ "

Bila suatu constrain bertanda "³ ", maka harus dirubah menjadi "£" dan akhirnya menjadi "=" agar dapat diselesaikan dengan metode Simplex, dengan cara merubah tanda tiap-tiap koefisien dari positif menjadi negatif (atau sebaliknya). Akibatnya RHS menjadi negatif. Ini harus dirubah lagi supaya dapat diselesaikan, dan akan dibicarakan pada bagian berikutnya. Slack Variable bertanda positif karena

semua koefisien bertanda negatif (dalam hal ini slack variable sering disebut dengan surplus variable).

D. RHS bertanda negatif

Bila di bagian kanan persamaan bertanda negatif maka harus dirubah menjadi positif. Caranya dengan merubah tanda positif negatif dari tiap-tiap koefisien, kemudian ditambah dengan variabel buatan. Karena perubahan ini slack variabel menjadi negatif, hal ini tidak memungkinkan penggunaan metode Simplex. Oleh karena itu harus ditambahkan satu variabel buatan (artificial variabel) yang akan menjadi variabel dasar dalam tabel permulaan. Sesuai dengan penjelasan sebelumnya maka kalau ada variabel buatan harus ditambahkan nilai M pada fungsi tujuan, dan merubahnya agar nilai variabel dasar pada fungsi tujuan bernilai 0.

3.2. Analisa Linier Goal Programming

3.2.1. Model tujuan tunggal (Single Goal Programming)

Untuk memperjelas hubungan antara LP dan LGP maka akan dibahas model tujuan tunggal, dengan memakai contoh sebuah perusahaan menghasilkan dua barang, yaitu barang 1 dan barang 2. Masing-masing barang memerlukan waktu untuk ditangani dalam dua bagian, yaitu bagian 1 dan bagian 2. Barang 1 membutuhkan 20 jam dibagian 1 dan barang 2 memerlukan 10 jam di bagian 2. Barang 2 membutuhkan 10 jam di bagian 1 dan membutuhkan 10 jam di bagian 2. Bagian 1 memiliki keterbatasan waktu sampai 60 jam dan bagian 2 sampai 40 jam. Sumbangan keuntungan barang 1 sebesar 40 dan barang 2 sebanyak 80. Tujuan pemilik adalah memaksimumkan keuntungan. Perumusan LP masalah itu adalah :

Maximize :

$Z = 40x_1 + 80x_2$

Subject to :

$20x_1 + 10x_2$

$10x_1 + 10x_2$

x_1, x_2

dimana

x_1 : banyaknya barang 1 yang diproduksi

x_2 : banyaknya barang 2 yang diproduksi.

Bagian 1 masih menyisakan waktu $s_1 = 20$ jam bagian 2 bekerja penuh, $s_2 = 0$.

Perumusan LGP masalah itu (karena tujuannya maksimisasi keuntungan, kita tetapkan secara sembarang target keuntungan, misalnya 1000) adalah :

Maximize : $Z = d^+$

Subject to :

$20x_1 + 10x_2 \quad £ \quad 60$

$10x_1 + 10x_2 \quad £ \quad 40$

$40x_1 + 80x_2 + d^- - d^+ = 1000$

$x_1, \quad x_2, \quad d^-, \quad d^+ \quad ³ \quad 0$

dimana *x* menunjukkan variabel keputusan dan d merupkan variabel simpangan (*deviational variable*).

Perbedaan antara model LP dengan LGP terletak pada fungsi tujuannya, dimana dalam LGP digunakan variabel simpangan dan ditambahkan persamaan :

$40x_1 + 80x_2 + d^- - d^+ = 1000$. Persamaan itu tampak seperti kendala, tetapi sesungguhnya merupakan persamaan tujuan untuk model itu, dalam kasus ini tujuan keuntungan. Dua variabel non negatif d^- dan d^+ adalah variabel simpangan untuk tujuan. Mereka menunjukkan berapa banyak kekurangan (d^-) atau kelebihan (d^+) dari target keuntungan sebesar 1000.

Meskipun pada sebagian besar aplikasi d^- dan d^+ akan tampak pada sebuah persamaan tujuan, paling banyak hanya satu dari dua variabel itu yang memiliki nilai positif dalam setiap solusi. Dengan kata lain, adalah tidak mungkin pada saat yang sama terjadi kekurangan atau kelebihan target keuntungan. Jika nilai target dicapai sacara pas, kedua variabel simpangan akan bernilai nol. Jika tujuan tidak dapat dicapai, salah satu variabel simpan akan bernilai nol.

Karena tujuan masalah ini adalah maksimisasi keuntungan, fungsi tujuan masalah LGP hanya mengandung sebuah variabel simpangan. Fungsi tujuan yang tampak pada model LP ditulis sebagai sebuah kendala tujuan. Hanya variabel simpangan yang berkaitan dengan tujuan tampak pada funsi tujuan. Untuk kasus ini hanya variabel simpangan d^- yang dimasukkan. Ini berdasar pada tujuan dalam bentuk LGP, yaitu meminimumkan kekurangan target keutungan. Dan karena kekurangan itu tak diinginkan, kita akan menekan d^- mendekati nol. Jika kelebihan

nilai target merupakan hal yang tak diinginkan (seperti waktu lembur, polusi, dan lain-lain), maka hanya d^+ yang dimasukkan dalam fungsi tujuan. Jika kita menghendaki pencapaian target secara pas, maka kedua variabel simpangan akan dimasukkan dalam fungsi tujuan.

Karena variabel simpangan dapat diperlakukan seperti variabel-variabel yang lain, perumusan LGP persoalan ini dapat diselesaikan dengan algoritma simplex. Solusi optimalnya adalah x_1=0, x_2=4, s_1=0, dan d^-=680. Solusi ini identik dengan solusi formulasi LP, kecuali pada nilai Z, untuk perumusan LP diperoleh Z=320 dan untuk LGP diperoleh Z=680 (nilai z sesungguhnya 680, tetapi perhatikan bahwa kita menyelesaikan masalah minimisasi dengan logika maksimisasi). Nilai Z pada masalah LGP menunjukkan seberapa besar kekurangan pencapaian target keuntungan, sehingga keuntungan maksimum yang dapat dicapai adalah 320, yaitu 1000-680. Yang identik dengan solusi model LP.

3.2.2. Model banyak tujuan (Multiple Goal Programming)

Ada tiga jenis model banyak tujuan, yaitu tujuan banyak tanpa prioritas, tujuan banyan dengan prioritas, dan tujuan banyak dengan prioritas dan bobot.

Dalam praktek model yang terakhir adalah yang paling berguna, namun kita akan membahas satu persatu model-model itu untuk memahami dengan baik konsep prioritas dan bobot.

A. Multiple Goal Programming tanpa Prioritas (Prioritas sama)

Meskipun model banyak tujuan tanpa prioritas mudah penanganannya, ia kurang memiliki arti praktis. Pikirkan persoalan berikut. Misalnya masalah produksi tujuan tunggal pada sesi sebelumnya dimodifikasi sedemikian rupa sehingga disamping tujuan kentungan, paling sedikit dua unit dari setiap jenis barang harus diproduksi. Pemilik memandang tujuan terakhir ini sama penting dengan tujuan pertama. Dalam keadaan ini, pemilik menganggap bahwa penyimpangan satu rupiah dari target keutungan sama penting dengan penyimpangan satu unit target produksi. Perumusan LGP untuk masalah itu adalah :

Minimize :

$$Z = d_1^- + d_2^+ + d_3^-$$

Subject to :

$$20x_1 + 10x_2 \pounds 60$$

$$10x_1 + 10x_2 \pounds 40$$

$$40x_1 + 80x_2 + d_1^- - d_2^+ = 1000$$

$$x_1 + + d_2^- - d_2^+ = 2$$

$$x_2 + d_3^- - d_3^+ = 2$$

$$x_1, \quad x_2, \quad d_1^-, \quad d_1^+, \quad d_2^-, \quad d_2^+, \quad d_3^-, \quad d_3^+, \; ^3 \; 0.$$

Karena variabel simpangan pada fungsi tujuan tidak memiliki prioritas, persoalan ini dapat diselesaikan dengan metode simplex. Solusinya adalah $x_1=0$, $x_2=4$, $s_1=20$,

s_2=0,d_1^-= 680, d_1^+=0, d_2^-=0, d_2^+=0, d_3^-=0, d_3^+=2, dan Z=682. Solusi model ini tidak mencapai tujuan produksi (paling tidak dua unit x_1 dan dua unit x_2) karena tidak ada prioritas untuk tujuan-tujuan itu. Model itu hanya meminimkan jumlah simpangan untuk semua tujuan.

B. Multiple Goal Programming Dengan Prioritas

Jika pemilik mempertimbangkan banyak tujuan, biasanya memiliki skala prioritas untuk tujuan-tujuan itu. LGP memberikan urutan preferensi tujuan melalui koefisien prioritas (P). Tujuan (variabel simpangan) yang memiliki prioritas pertama di beri nilai fungsi tujuan P_1, tujuan dengan prioritas kedua di beri nilai P_2, proses ini diteruskan sampai semua tujuan telah diturutkan. Koefisien prioritas P_1, P_2, dan seterusnya bukan merupakan parameter atau variabel. Pada umumnya mereka bukan suatu nilai angka, mereka hanya menunjukkan tingkat prioritas. Karena koefisien prioritas muncul pada fungsi tujuan, algoritma simplex biasa tidak dapat digunakan untuk menyelesaikan masalah itu. Namum,metode simplex dapat di modifikasi untuk menangani koefisen prioritas dan menjamin bahwa penyimpangan bagi tujuan dengan prioritas pertama diminimumkan sebelum prioritas yang lebih rendah diperhatikan.

Penggunaan prioritas dapat ditunjukkan melalui masalah produksi yang ditunjukkan dengan menetapkan skala prioritas sebagai berikut :

P_1 (prioritas 1) : capai tujuan produksi dua unit untuk setiap jenis barang.

P_2 (prioritas 2) : maksimumkan keuntungan.

Model untuk masalah ini serupa dengan model tanpa prioritas sebelumnya dengan kekecualian penambahan koefisien prioritas pada fungsi tujuan atau Model yang baru itu adalah :

Minimize :

$Z = P_1d_2^- + P_1d_3^- + P_2d_1^+$

Subject to :

$20x_1 + 10x_2 \leq 60$

$10x_1 + 10x_2 \leq 40$

$40x_1 + 80x_2 + d_1^- - d_1^+ = 1000$

$x_1 + d_2^- - d_2^+ = 20$

$x_2 + d_3^- - d_3^+ = 2$

$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+ \geq 0$

Penafsiran solusi ini adalah :

Tujuan 1 (P_1) dicapai : dua unit x_1 dan x_2 diproduksi

Tujuan 2 (P_2) tak tercapai : sumbangan total terhadap keuntungan adalah 240 (=1000–760), kekurangan target keuntungan sebesar 760 (d_1^-=760).

Membandingkan hasil ini dengan hasil sebelumnya, terlihat bahwa pemilik harus mengorbankan keuntungan (240 lawan 320) untuk memenuhi tujuan produksi paling tidak 2 unit tiap barang.

C. Multiple Goal Programming Dengan Prioritas dan Bobot

Dua masalah tujuan banyak sebelumnya memiliki indeks prioritas yang sama. Kadang-kadang kita dihadapkan pada beberapa tujuan dengan urutan yang sama adalah lebih penting dibanding tujuan-tujuan lain. Jika demikian perlu digunakan bobot yang berlainan untuk mencerminkan beda kepentingan dalam tingkat prioritas yang sama. Misalkan, keuntungan dan waktu lembur dari suatu persoalan tadi memiliki urutan prioritas yang sama.. jika tidak ada bobot, pemilik menganggap bahwa pwnyimpangan keuntungan satu rupiah sama pentingnya satu jam lembur. Jika tidak demikian, kemudian dapat diberikan bobot yang . Sebagai contohnya kita hadapi masalah berikut ini. Sebuah perusahaan memproduksi

2 jenis produk. Dalam proses memproduksi dibutuhkan dua departemen produksi. Produk 1 memerlukan 2 jam produksi di departemen satu dan 3 jam produksi di departemen dua. Dan untuk produk 2 diperlukan 4 jam produksi di departemen satu dan 3 jam produksi di departemen dua. Waktu kerja tiap minggu biasanya adalah 80 jam kerja pada masing-masing departemen yang ada. Dan sudah mejadi kebijaksanaan perusahaan sejak perusahaan ini berdiri menghindari terjadinya **lembur**, jika itu mungkin. Keuntungan perusahaan dari produk 1 adalah 30, dan keuntungan yang didapat dari produk 2 adalah 40 tiap camera. Pihak manajemen mempunyai kebijaksanaan dengan prioritas tujuan sebagai berikut :

P1 (prioritas 1) : menghindari waktu lembur.

P2 (prioritas 2) : catatan penjualan menunjukkan bahwa rata - rata penjualan tiap minggunya adalah minimum 10 buah camera standard dan 10 buah untuk

produk 2. Sejak terjadinya batasan waktu dalam memproduksi produk dan sejak produk 2 mempunyai margin keuntungan lebih besar, goal yang harus dicapai oleh pihak penjualan berdasarkan bobot keuntungan yaitu 30 untuk produk 1 dan 40 untuk produk2 (dalam rasio kita dapat menggunakan bobot 3 dan 4 sama dengan rasio keuntungan yang dihasilkan).

P3 (prioritas 3) : maximalkan keuntungan.

Dengan nilai keuntungan maximal (RHS) yang dicapai 1200.

Bentuk persamaan Goal programming dari kasus diatas :

Minimize :

$$Z = P_1 d_1^+ + P_2 d_2^+ + 30P_2 d_3^- + 40P_3 d_3^-$$

Subject to :

$$2x_1 + 40x_1 + d_1^- + d_1^+ = 80$$

$$3x_1 + 4x_2 + d_2^- - d_2^+ = 80$$

$$x_1 + d_3^- - d_3^+ = 10$$

$$x_2 + d_4^- - d_4^+ = 10$$

$$30x_1 + 40x_2 + d_4^- - d_5^+ = 1200$$

$$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+, d_5^-, d_5^+$$

Variabel simpangan d_4^+ dan d_5^- meunujukkan waktu lembur yang diperlukan kegiatan 1 dan 2. Variabel d_6^+dimasukkan dalam kendala tujuan untuk mencerminkan kemungkinan melebihi 50 jam lembur. Perhatikan bahwa :

Tujuan 1 (P_1) dicapai, waktu lembur pas 50 jam, 20 jam lembur (d_4^+) dalam kegiatan 1 dan 30 jam (d_5^+) dalam kegiatan 2.

Sebagian tujuan 2 (P_2) dipenuhi, enam barang 2 diproduksi tujuan ini memiliki bobot terbesar, tetapi hanya satu unit barang 1 dihasilkan sebelum batasan jam lembur dilewati.

Tujuan 3 (P_3) tak tercapai, sumbangan total terhadap keuntungan adalah 520 (yaitu 1000- 480), kekurangan target keuntungan adalah 480 (d_1^- = 480).

3.3. Kebutuhan Sistem

Pembuatan program komputer pada dewasa ini menjadi semakin mudah saja. Banyak sekali compiler visual yang menjanjikan kemudahan pembuatan program. Salah satunya adalah produk dari Borland International yakni compiler yang bernama Borland Delphi. Compiler ini punya basis bahasa Pascal sehingga sangat memudahkan programmer-programmer pemula yang kebanyakan menguasai bahasa Pascal terlebih dahulu.

Kemudahan yang menonjol dari Borland Delphi adalah pada cara penggunaannya. Delphi memiliki banyak komponen visual yang sudah spesifik, sehingga kita tinggal memilih dan menempatkannya pada form kita. Untuk memasukkan komponen-komponen yang kita kehendaki, kita tinggal mengklik pada gambar komponen yang kita kehendaki dan menggambarnya pada form. Propertiesnya bisa kita atur lewat kotak properties atau lewat program pada waktu run time. Prosedur program terbentuk secara otomatis jika kita double klik suatu nama event pada kotak event. Selanjutnya kita bisa menuliskan kode program di sini.

Untuk pembuatan perangkat lunak dalam tugas akhir ini penulis menggunakan compiler Borland Delphi versi 3.0. Compiler ini membutuhkan perangkat keras dengan spesifikasi sebagai berikut :

- CPU PC/AT 486 atau yang lebih baru
- Memory 16 MB atau lebih
- Monitor SVGA (atau minimum VGA)
- Harddisk dengan ruang kosong beberapa MB
- Mouse

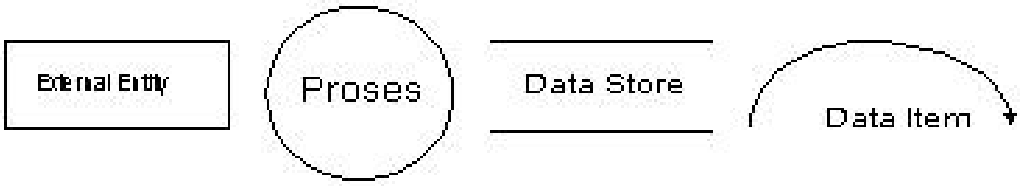
Selain itu dibutuhkan juga sistem operasi dari Microsoft yakni Microsoft Windows 95 atau yang lebih baru. Bisa juga digunakan sistem operasi Microsoft Windows NT.

3.4. Perancangan Perangkat Lunak

Sebelum dilakukan pembuatan perangkat lunak, terlebih dahulu dilakukan perancangan sistem. Dalam perancangan sistem ini, untuk analisa kebutuhan, digunakan diagram aliran data (data flow diagram).

3.4.1. Flowchart dari sistem

Flowchart merupakan teknik grafis yang menggambarkan aliran informasi dan transformasi data dari input sampai ke output. Disebut juga data flow graph atau bubble chart. Komponen dari Flowchart ini adalah :



Gambar 3.1. Komponen Finite Automata (FA) sistem

A. External Entity

Entitas yang berperan sebagai penghasil atau pemakai informasi yang berada di luar sistem yang dimodelkan.

B. Process

Proses transformasi informasi yang berada dalam sistem.

C. Data Store

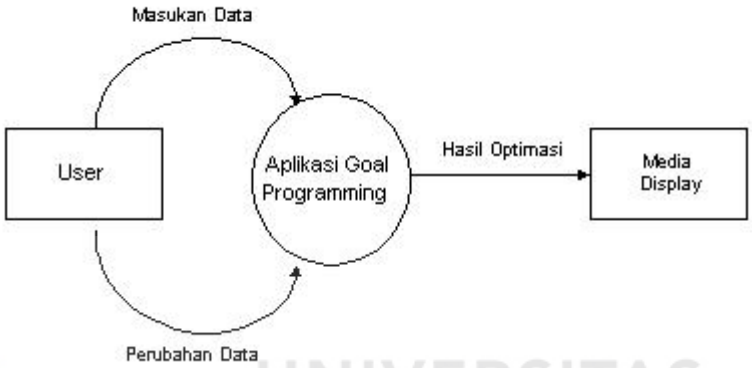
Penyimpanan data dalam sistem yang dapat diakses oleh satu atau beberapa proses. Biasanya berupa file data dalam media penyimpanan.

D. Data Item

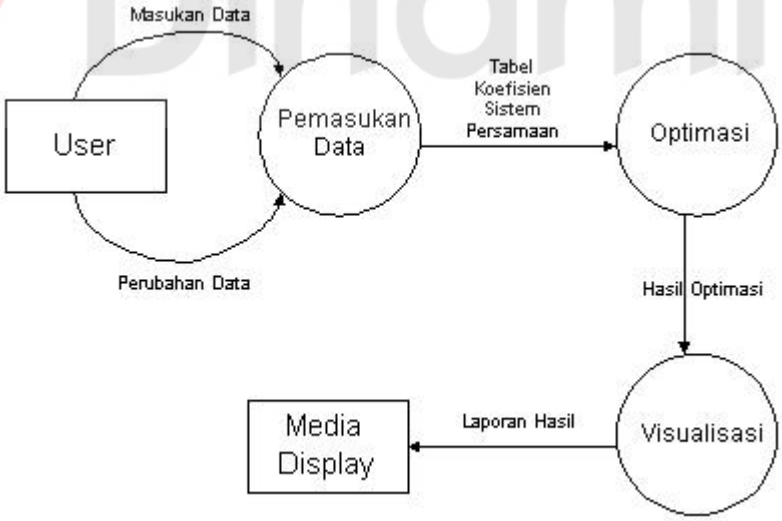
Satu atau sekumpulan data atau informasi yang mengalir dari satu proses ke proses yang lainnya. Tanda panah menunjukkan arah aliran data.

3.4.2. Flowchart optimasi

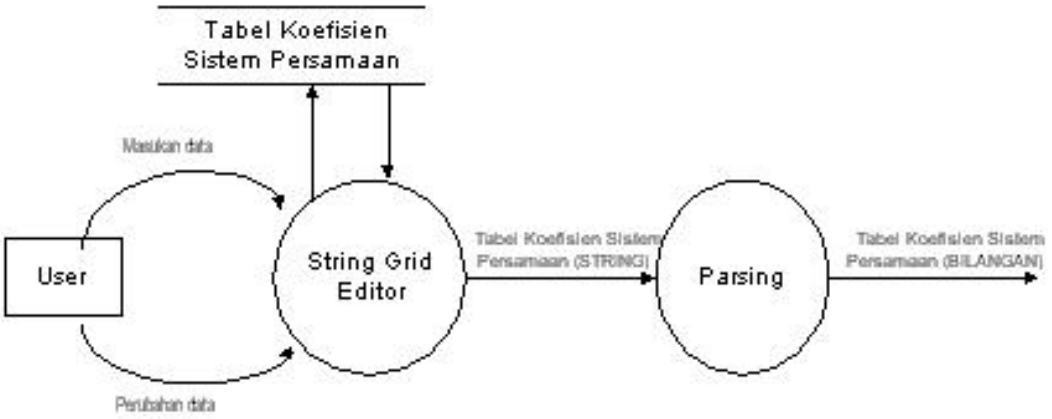
Susunan Flowchart perangkat lunak untuk Goal Programming dibuat seperti gambar-gambar flow di bawah ini mulai dari sistem sampai sub sistem yang ada.



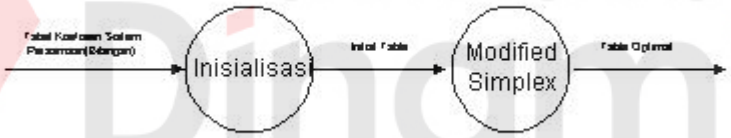
Gambar 3.2. FA utama sistem



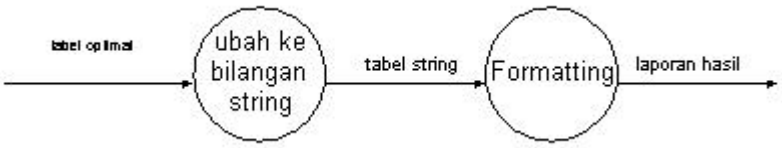
Gambar 3.3. FA proses



Gambar 3.4. FA Optimasi perhitungan



Gambar 3.5. FA Inisialisasi dengan parsing



Gambar 3.6. FA Visualisasi dari Optimasi

3.5. Pembuatan Perangkat Lunak

Perangkat lunak ini dibangun dari beberapa macam modul yang masing-masing mempunyai kegunaan sendiri-sendiri. Setiap modul merupakan implementasi

dari satu atau beberapa proses dalam diagram di atas. Modul-modul tersebut diuraikan lebih lanjut pada bagian-bagian berikut ini.

3.5.1. Editor Input

Editor input di sini berupa string grid yang dapat diisi user dengan koefisien-koefisien sistem persamaan yang akan diselesaikan dengan aplikasi ini. String grid adalah komponen visual bawaan Delphi. String grid tak lain adalah sebuah tabel yang masing-masing selnya dapat berisi sebuah string. Dalam program ini sel diisi dengan string berupa angka saja, baik bulat maupun pecahan desimal.

Input yang sudah dimasukkan dapat disimpan ke dalam sebuah file agar jika ingin memasukkan input yang sama pada lain waktu, user tinggal membuka file tersebut. File input disimpan sebagai file teks biasa dengan ekstensi LGP. Saat membuka file ini, teks yang diambil dari file akan dipotong-potong sesuai dengan formatnya sehingga dapat kembali tersusun ke dalam string grid tanpa terjadi kesalahan.

3.5.2. Parser

Modul ini membaca string grid sel demi sel dan menerjemahkannya ke dalam array bilangan real. Isi string grid tidak bisa langsung dioperasikan sebagai bilangan karena isinya masih berupa string. Untuk dapat dioperasikan masing-masing isi sel string grid diubah ke dalam bilangan real dan di copykan ke array dengan tipe real sesuai dengan posisinya di dalam string grid.

Setelah melakukan tugasnya yang pertama Parser akan melakukan tugas berikutnya yakni membuat Initial Table yakni tabel awal yang bisa dioperasikan dengan metode Modified Simplex. Dalam pembuatan Initial Table ini dilakukan penambahan deviational variable pada tabel yang sudah terbentuk dari parsing dan juga klasifikasi prioritas goal yang akan dioptimalkan dengan program ini.

3.5.3. Optimasi Goal Programming

Metode Simplex sudah umum digunakan untuk mengoptimasi Linear Programming. Tetapi metode Simplex standar tidak bisa menyelesaikan Linear Programming yang mempunyai banyak tujuan. Goal Programming merupakan salah satu bentuk Linear Programming yang mempunyai banyak tujuan. Jadi untuk menyelesaikan Goal Programming kita membutuhkan suatu metode baru yang bisa menyelesaikannya. Metode yang digunakan di sini tidak jauh berbeda dengan metode Simplex standar yakni metode Modified Simplex.

Modul ini merupakan jantung dari program aplikasi ini, di mana optimasi dari Goal Programming dilakukan di sini. Program akan melakukan iterasi sampai tercapai hasil yang paling optimal dari sistem persamaan Goal Programming yang dimasukkan. Dalam banyak kasus, langkah-langkah penyelesaian Goal Programming ini diperlukan untuk analisa. Untuk itu di dalam program ini dilakukan penyimpanan isi tabel Simplex tiap kali melakukan satu iterasi sampai dicapai penyelesaian optimal. Penyimpanan ini dilakukan oleh modul lain yang dioperasikan tiap kali iterasi.

3.5.4. Visualisasi

Tujuan akhir dari program ini adalah memberikan laporan hasil dari proses yang dilakukan oleh program. Laporan yang dimaksud di atas dilakukan oleh modul ini.

Modul ini melakukan tiga langkah sebelum akhirnya bisa memberikan laporan yang dapat dimengerti oleh user. Pertama modul ini akan merubah tabel Simplex yang masih berupa bilangan menjadi string sehingga dapat dirangkaikan dengan string yang lain. Proses ini merupakan kebalikan dari proses parsing pada pemasukan data.

Setelah melakukan perubahan dari bilangan ke string, modul ini akan memformat tabel dengan menambahkan header dan label-label yang diperlukan. Kemudian dilakukan pengaturan tabulasi sehingga tabel yang akan ditampilkan tampak bagus dan mudah dimengerti.

Terakhir modul ini akan membuat laporan nilai optimal dari masing-masing variabel.

3.6. Desain Input

Disain aplikasi input ini terdiri dari tabel – tabel yang berisi dari fungsi objective dan constraint yang datanya diisi manual atau dapat di ambil dari data yang sudah disimpan.

3.6.1. Komponen program

Sistem secara keseluruhan diintegrasikan oleh sebuah pop-up menu yang akan sangat memudahkan pemakaian perangkat lunak ini. Untuk mengintegrasikan sistem secara keseluruhan maka dibuat sebuah menu *Pop-Up*. Disamping aplikasi dibawah ini tersedia dalam bentuk menu juga terdapat pada *toolbar* berupa *Icon* dari aplikasi software Linier Goal Programming.

Menu yang mempunyai menu dan toolbar sebagai berikut :

A. Menu

File

New : Untuk menginputkan persamaan baru,

Open : Untuk membuka persamaan LP yang tersimpan dalam disk

Save : Untuk menyimpan persamaan yang sedang aktif

Save **A**s : Untuk menyimpan persamaan yang aktif dengan nama baru

Exit **:** Untuk keluar dari perangkat lunak ini.

Action

Run : Merubah persamaan matematis menjadi array bilangan yang akan dioperasikan dengan metode simplex.

Add **W**eight : Menyediakan fasilitas Input weight (bobot) yang mungkin ada pada persamaan.

Over Value Limitation : Menyediakan fasilitas input Grid.

Help

Content : Berisi panduan untuk menggunakan software ini

A**b**out LGP : Keterangan tentang pembuat program

B. Form

Pada garis besarnya aplikasi linier Goal Programming ini menggunakan 4 form utama dengan menggunakan penekanan **tab** yaitu : **tab *problem*, tab *Input*, tab *iterasi*, tab *result (hasil)*.**

B.1. **Tab** problem

Berisi form kosong dimana anda dapat membuat keterangan dari setiap permasalahan yang akan diselesaikan.

B.2. **Tab** Input

Berupa tabel persamaan dan tabel constraint, dan control dari tabel tabel yang akan diselesaikan yaitu : control dari fungsi Objective, fungsi Contraint, dan variabel yang akan digunakan.

B.3. **Tab** Iteration

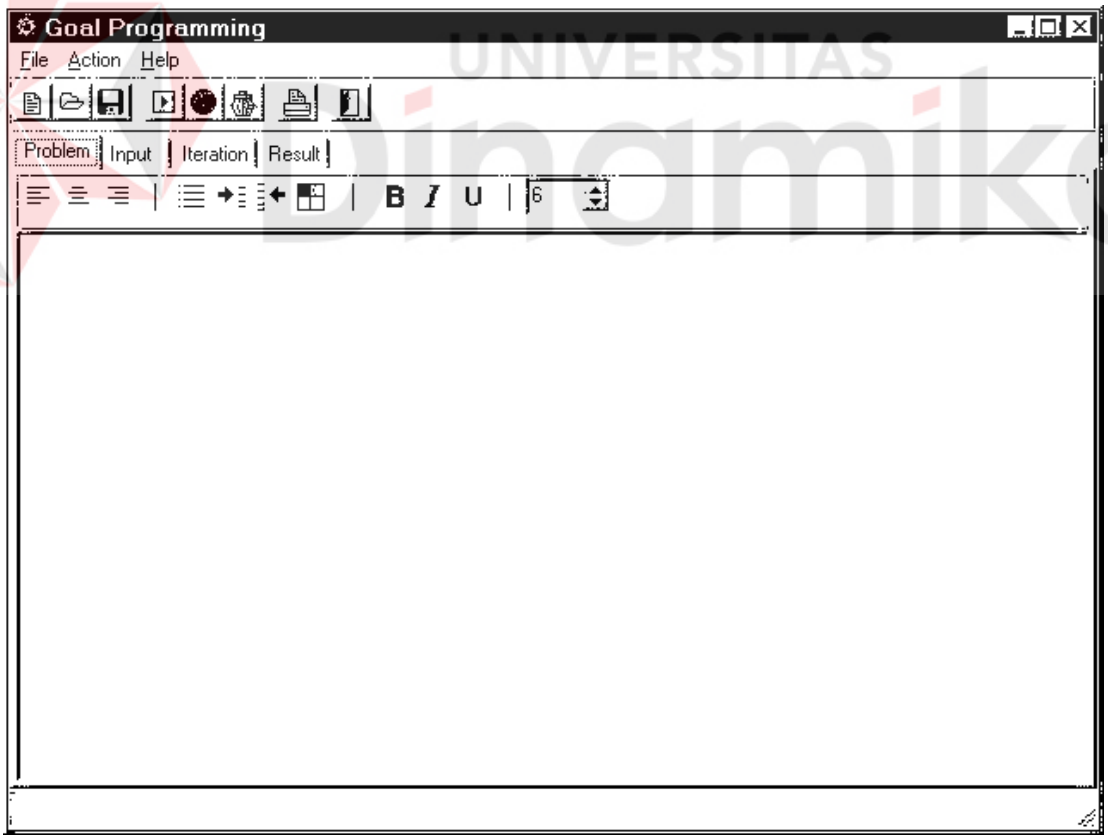
Berisi form kosong yang nantinya akan menampilkan tabel-tabel iterasi dari persamaan yang sudah di - ***Run.***

B.4. **Tab** Result

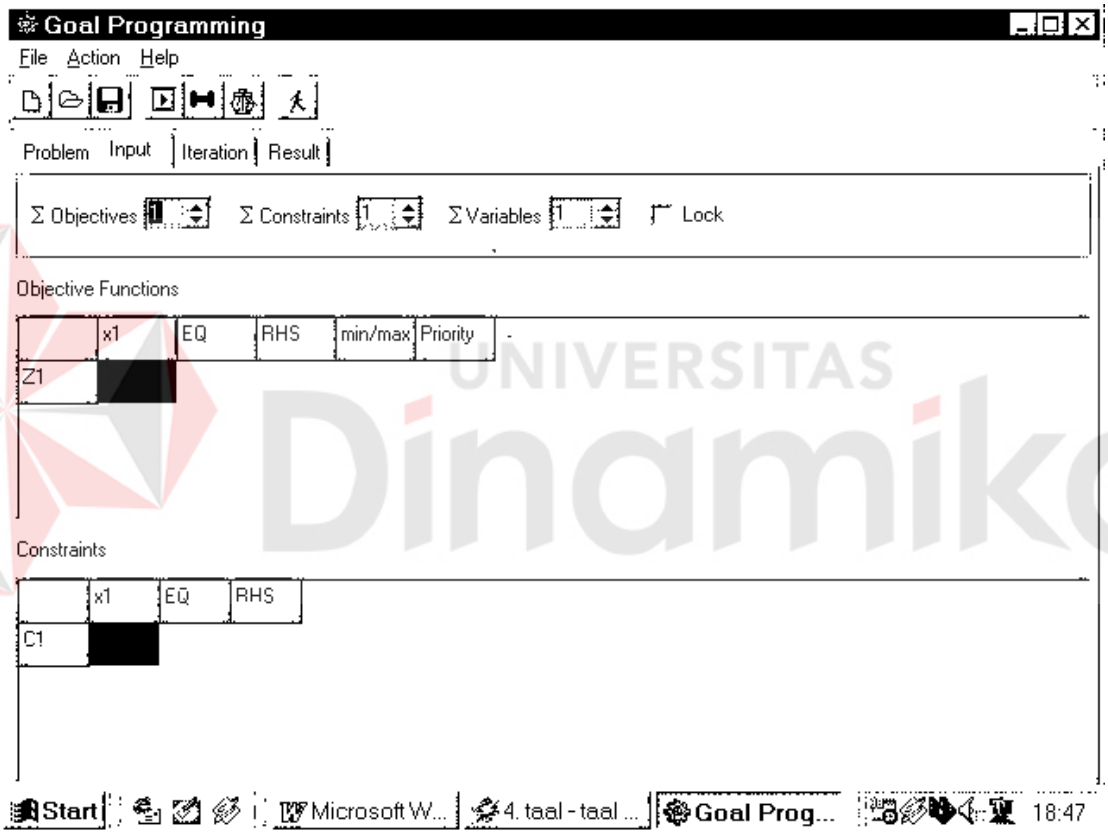
Menampilkan form kosong yang nantinya akan berisi kesimpulan dari hasil akhir yang telah dicapai.

3.6.2. Input dan editor input

Editor input berupa tabel tabel yang sudah terbagi menjadi beberapa bagian yang dapat di setting sesuai dengan kebutuhan untuk melakukan perhitungan (sesuai dengan persamaan yang ada). Dan data yang dinputkan juga dapat di simpan dalam suatu file, yang dapat anda buka sewaktu waktu hingga dapat menghemat waktu anda unutk melakukan entry data. Dan anda juga dapat meng inputkan keterangan pada form problem.



Gambar 3.7. Disain Form Problem



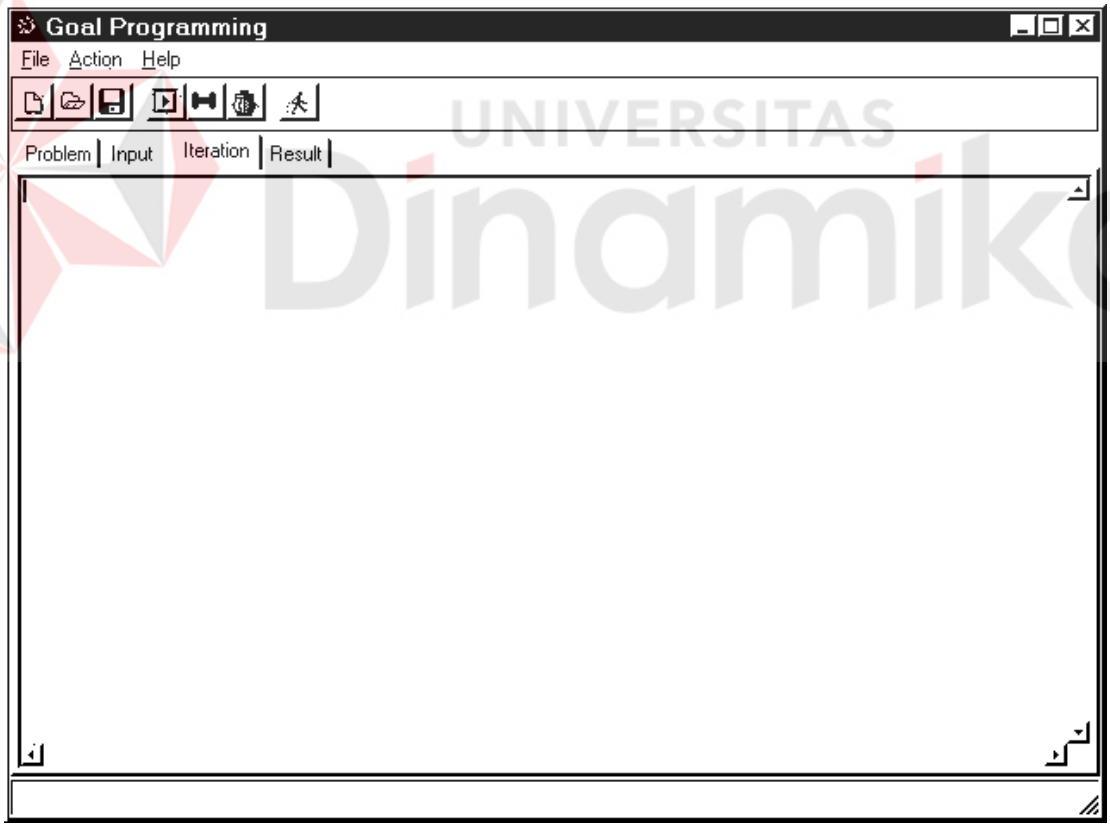
Gambar 3.8. Disain Form Input

3.7. Desain Output dan Editor Output.

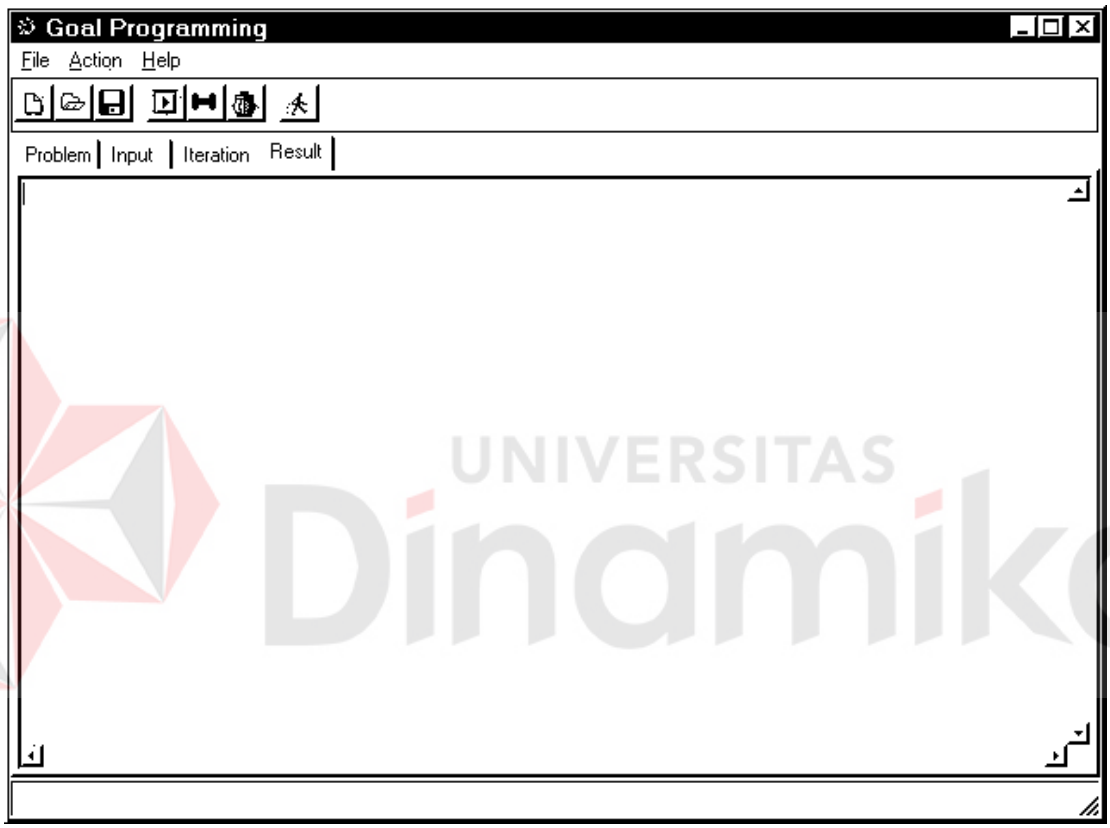
Output dari Aplikasi ini berupa tabel iterasi ke-1 sampai iterasi ke-n dan juga hasil akhir dari perhitungan yang diminta dalam bentu persamaan yang diambil dari perhitungan iterasi dari soal yang ada.

Output Editor

Yang terdiri dari dua bentuk tampilan yaitu form iterasi dan form hasil yang berupa persamaan dari perhitungan iterasi.



Gambar 3.9. Disain Form Output Iterasi



Gambar 3.10. Disain form output persamaan

BAB IV

IMPLEMENTASI DAN EVALUASI

PERANGKAT LUNAK

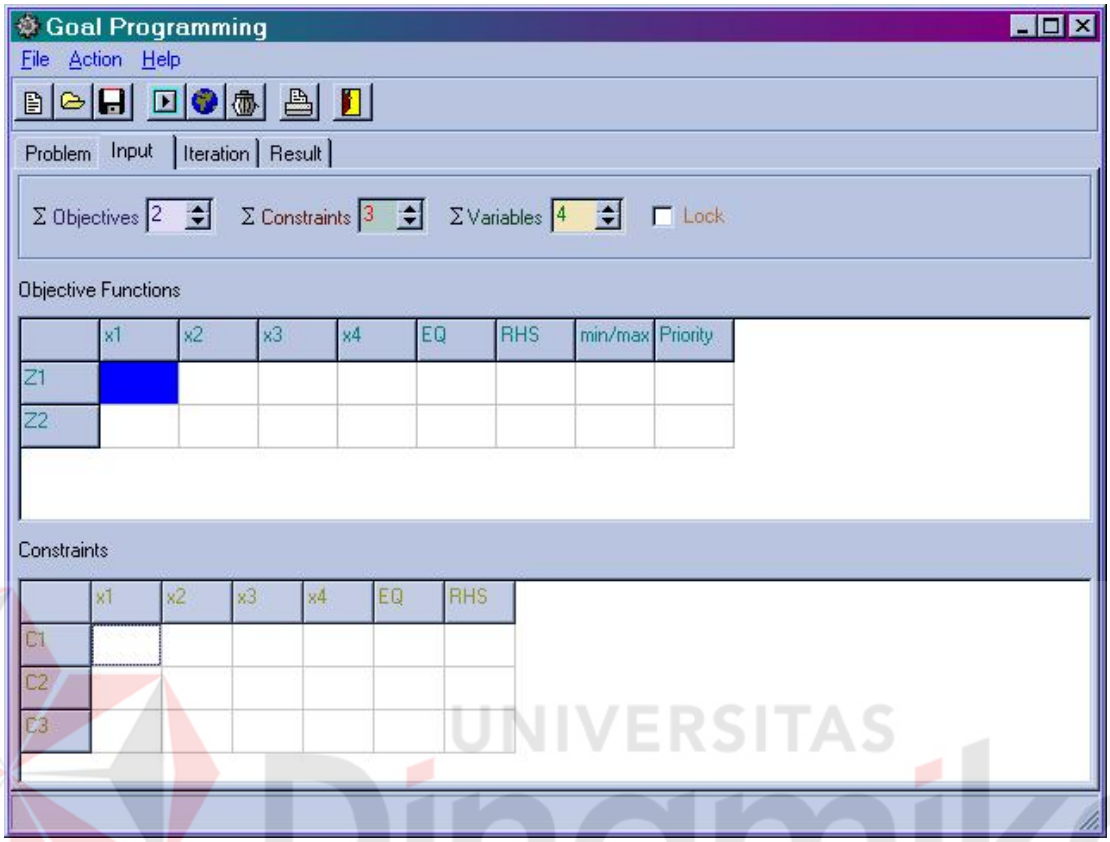
Dalam bab ini akan diberikan beberapa contoh permasalahan yang ada pada perangkat lunak bantu ini. Inputan yang dimasukkan berupa persamaan matematis sedangkan hasil optimasi tersebut berupa iterasi dengan menggunakan Metode Simplex.

4.1. Implementasi dari Linier Goal Programming

Dalam belajar menggunakan aplikasi Linier Goal Programming harus terlebih dahulu melalui beberapa tahap, yaitu :

4.1.1. Setting variabel, fungsi objective, dan contraint

Sebelum anda melakukan entry data, anda harus dapat mengevaluasi terlebih dahulu permasalahan yang ada, Agar dapat diketahui berapa sebenarnya jumlah variabel, fungsi objective dan contraint yang akan anda entry. Karena aplikasi ini menyediakan setting dari fungsi diatas, untuk mempermudah pemakai dalam menggunakan aplikasi Linier Goal Programming ini. Implementasi dalam sistem tampak pada gambar dibawah ini.



Gambar 4.1. Setting variabel, fungsi objective, dan constraint

4.1.2. Entry data persamaan

Setelah mengevaluasi permasalahan yang ada dan merubah bentuknya menjadi persamaan linier goal programming. Maka entry data pada aplikasi ini sangat mudah dengan adanya fasilitas fasilitas yang disediakan sebagai berikut :

A. Fungsi objective

Setelah setting dilakukan entrykan data persamaan sesuai dengan variabel – variabel yang ada. Seperti tampak pada gambar dibawah ini.

Objective Functions

	x1	x2	x3	EQ	RHS	min/max	Priority
Z1	1	1	2	=	12	max	P1
Z2	2	6	1	=	1	1	P2

Gambar 4.2. Persamaan fungsi objective

B. Contraint

Entry constraint hanya dilakukan jika permasalahan memang mempunyai constraint kerena biasanya pada Linier Goal Programming semua contraint dapat menjadi Goal. Akan tetapi jika anda membutuhkan constraint maka entry datanya akan tampak seperti dibawah ini.

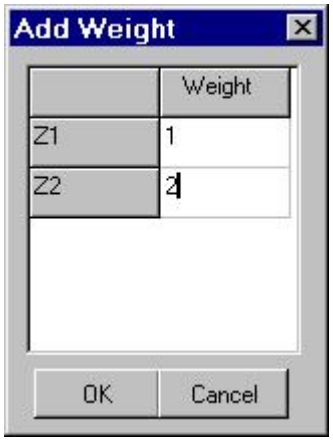
Constraints

	x1	x2	x3	EQ	RHS
C1	1	2	3	<=	12

Gambar 4.3. Entry data persamaan constraint

C. Bobot

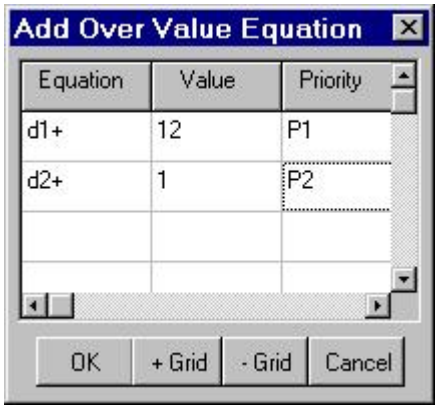
Sama halnya seperti constraint entry data bobot dilakukan apabila diperlukan. Dan jika anda melakukan entry pada data ini anda harus terlebih dahulu mengaktifkan *lock* yang ada pada toolbar, baru anda jalankan aplikasi Add Weight pada sistem ini. Maka tampilan entry data tampak seperti dibawah ini.



Gambar 4.4. Entry data bobot

D. Over value limitation

Sama halnya seperti constraint entry data Grid dilakukan apabila diperlukan. Anda jalankan aplikasi Over value limitation pada sistem ini. Maka tampilan entry data tampak seperti dibawah ini.



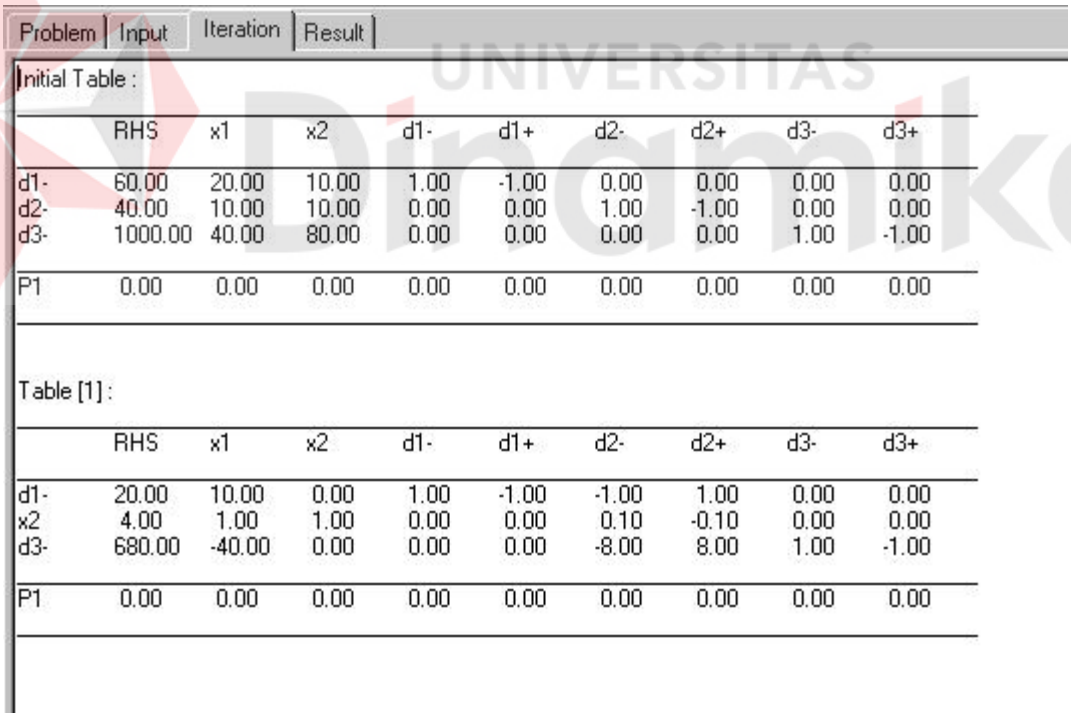
Gambar 4.5. Tampilan entry grid

4.1.3. Optimasi persamaan (*Run* Persamaan)

Setelah program di *Run* maka yang pertama kali dilakukan sistem adalah melakukan perhitungan iterasi simplex, lalu hasilnya akan di format kembali menjadi persamaan yang tampak pada tampilan hasil. Yang kemudian sehingga menghasilkan 2 macam ouput yaitu:

A. Tampilan iterasi.

Tampilan ini tidak tampak langsung setelah program di *Run* tetapi sudah dapat dilihat pada **tab** iteration. Seperti pada gambar di bawah ini.



Problem | Input | Iteration | Result

Initial Table :

	RHS	x1	x2	d1-	d1+	d2-	d2+	d3-	d3+
d1-	60.00	20.00	10.00	1.00	-1.00	0.00	0.00	0.00	0.00
d2-	40.00	10.00	10.00	0.00	0.00	1.00	-1.00	0.00	0.00
d3-	1000.00	40.00	80.00	0.00	0.00	0.00	0.00	1.00	-1.00
P1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

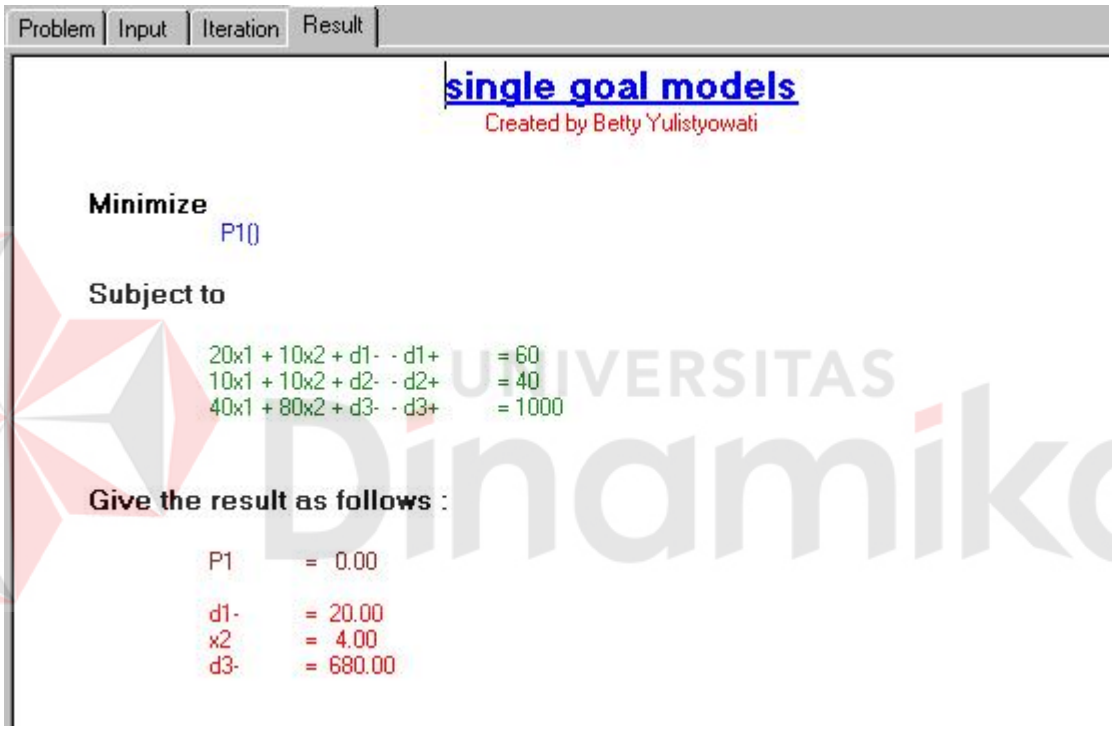
Table [1] :

	RHS	x1	x2	d1-	d1+	d2-	d2+	d3-	d3+
d1-	20.00	10.00	0.00	1.00	-1.00	-1.00	1.00	0.00	0.00
x2	4.00	1.00	1.00	0.00	0.00	0.10	-0.10	0.00	0.00
d3-	680.00	-40.00	0.00	0.00	0.00	-8.00	8.00	1.00	-1.00
P1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Gambar 4.6. Hasil optimasi berupa iterasi

B. Tampilan hasil.

Ketika *Run* dari permasalahan telah selesai maka yang tampil pertama kali adalah tampilan Result yang merupakan hasil akhir dengan format persamaan linier goal programming. Seperti yang ada pada gambar dibawah ini.



Gambar 4.7. Tampilan optimasi berupa persamaan linier goal programming

4.2. Evaluasi Linier Goal Programming

Sebagai contoh pemakaian perangkat lunak bantu untuk permasalahan LGP di bawah ini akan diberikan contoh sebuah perusahaan menghasilkan dua barang, yaitu barang 1 dan barang 2. Masing-masing barang memerlukan waktu untuk ditangani dalam dua bagian, yaitu bagian 1 dan bagian 2. Barang 1 membutuhkan 20 jam

dibagian 1 dan barang 2 memerlukan 10 jam di bagian 2. Barang 2 membutuhkan 10 jam di bagian 1 dan membutuhkan 10 jam di bagian 2. Bagian 1 memiliki keterbatasan waktu sampai 60 jam dan bagian 2 sampai 40 jam. Sumbangan keuntungan barang 1 sebesar 40 dan barang 2 sebanyak 80. Tujuan pemilik adalah memaksimumkan keuntungan. Perumusan LP masalah itu adalah :

Maximize :

$Z = 40x_1 + 80\ x_1$

Subject To :

$20x_1 + 10x_1$

$10x_1 + 10x_2$

x_1, x_2

Formulasi persamaan Goal Programming

Minimize :

$Z = d^-$

Subject To :

$20x_1 + 10x_2$

$10x_1 + 10x_2$

$40x_1 + 80x_2 + d^- - d^+ = 1000$ (contoh pendapatan maksimum yang pernah dicapai)

x_1, x_2, d^-, d^+

4.3. Evaluasi Single Goal Programming

Pada permasalahan yang sudah ada pada evaluasi Linier Goal Programming. Sudah memenuhi syarat untuk persoalan Goal Programming yang akan menjadi persamaan dibawah ini untuk diterapkan pada software LGP.

Maximize :

$Z = 40x_1 + 80\ x_1$

Subject To :

$20x_1 + 10x_1$

$10x_1 + 10x_2$

x_1, x_2

Formulasi persamaan Goal Programming

Minimize :

$Z = d^-$

Subject To :

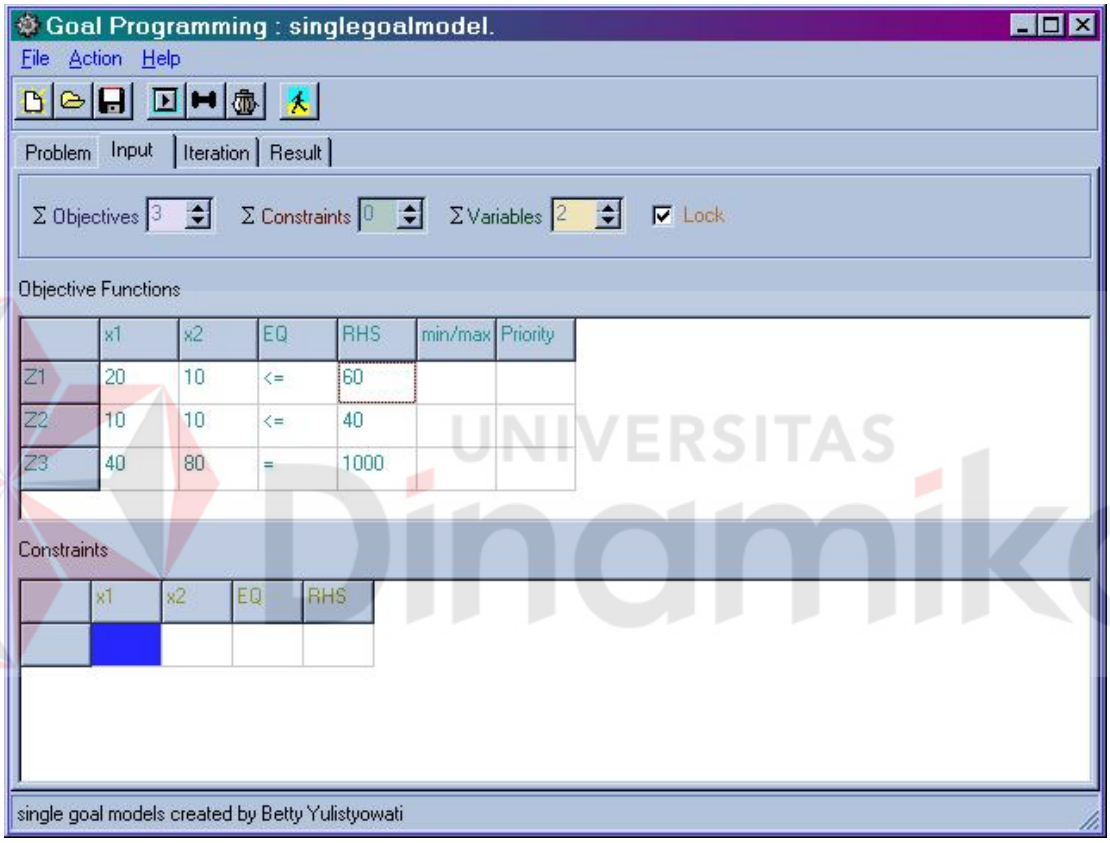
$20x_1 + 10x_2$

$10x_1 + 10x_2$

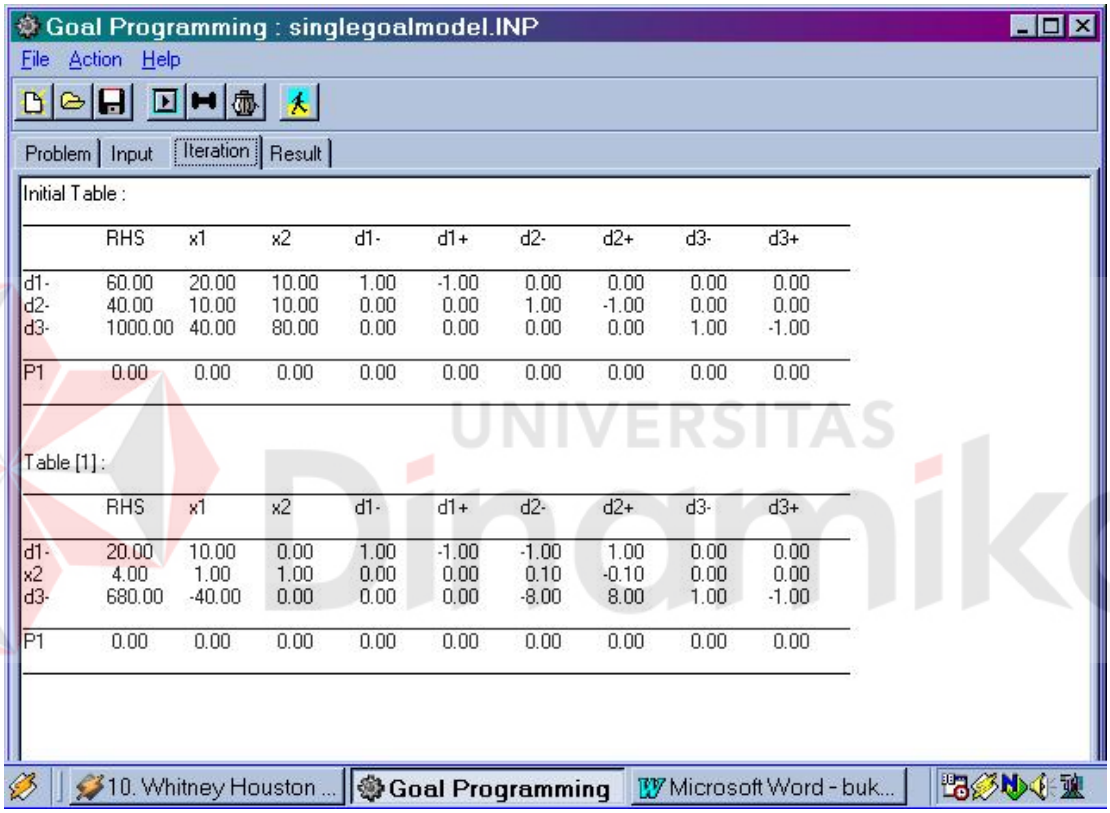
$40x_1 + 80x_2 + d^- - d^+ = 1000$ (contoh pendapatan maksimum yang pernah dicapai)

x_1, x_2, d^-, d^+

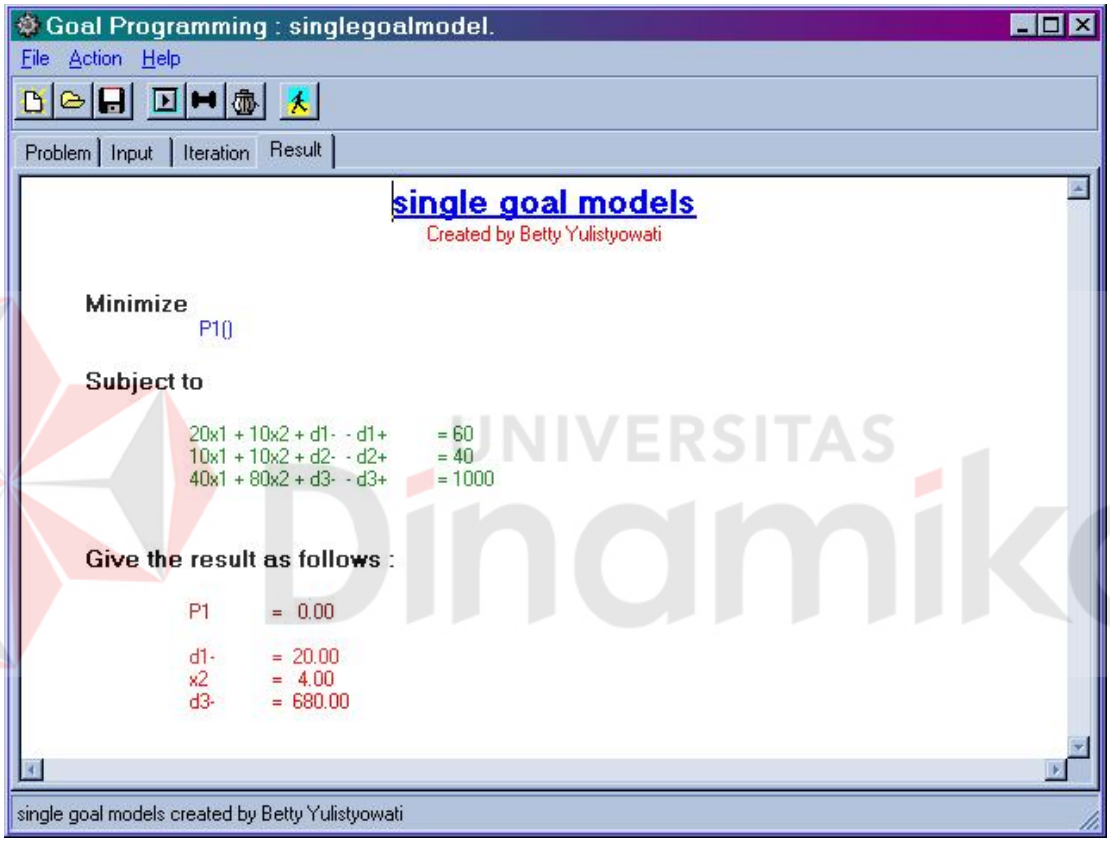
hasil yang diperoleh dari perhitungan diatas adalah $x_1 = 0$, $x_2 = 4$, dan $d_1^- = 20$ $d_3^+ = 680$.



Gambar 4.8. Tampilan input Single LGP



Gambar 4.9. Hasil perhitungan iterasi
Single goal Programming



Gambar 4.10. Hasil dari perhitungan Single goal Programming

4.4. Evaluasi Multiple Goal Programming

Penggunaan **LGP** dapat digunakan untuk memecahkan masalah yang mempunyai goal yang jamak yang dapat kita sebut juga dengan multiple Goal Programming. Aplikasi dari Multiple Goal Programming sendiri masih dibagi menjadi 3 lagi :

A. Multiple Goal Programming tanpa Prioritas (Prioritas sama)

B. Multiple Goal Programming dengan Prioritas

C. Multiple Goal Programming dengan Proritas dan Bobot

4.4.1. Multiple Goal Programming tanpa prioritas (prioritas sama)

Pikirkan persoalan berikut. Misalnya masalah produksi tujuan tunggal pada sesi sebelumnya dimodifikasi sedemikian rupa sehingga disamping tujuan keuntungan, paling sedikit dua unit dari setiap jenis barang harus diproduksi. Pemilik memandang tujuan terakhir ini sama penting dengan tujuan pertama. Dalam keadaan ini, pemilik menganggap bahwa penyimpangan satu rupiah dari target keutungan sama penting dengan penyimpangan satu unit target produksi. Perumusan LGP untuk masalah itu adalah :

Formulasi persamaan Goal Programming

Minimize :

$$Z = d_1^- + d_2^- + d_3^-$$

Subject To :

$$20x_1 + 10x_2 \quad £ \; 60$$

$$10x_1 + 10x_2 \quad £ 40$$

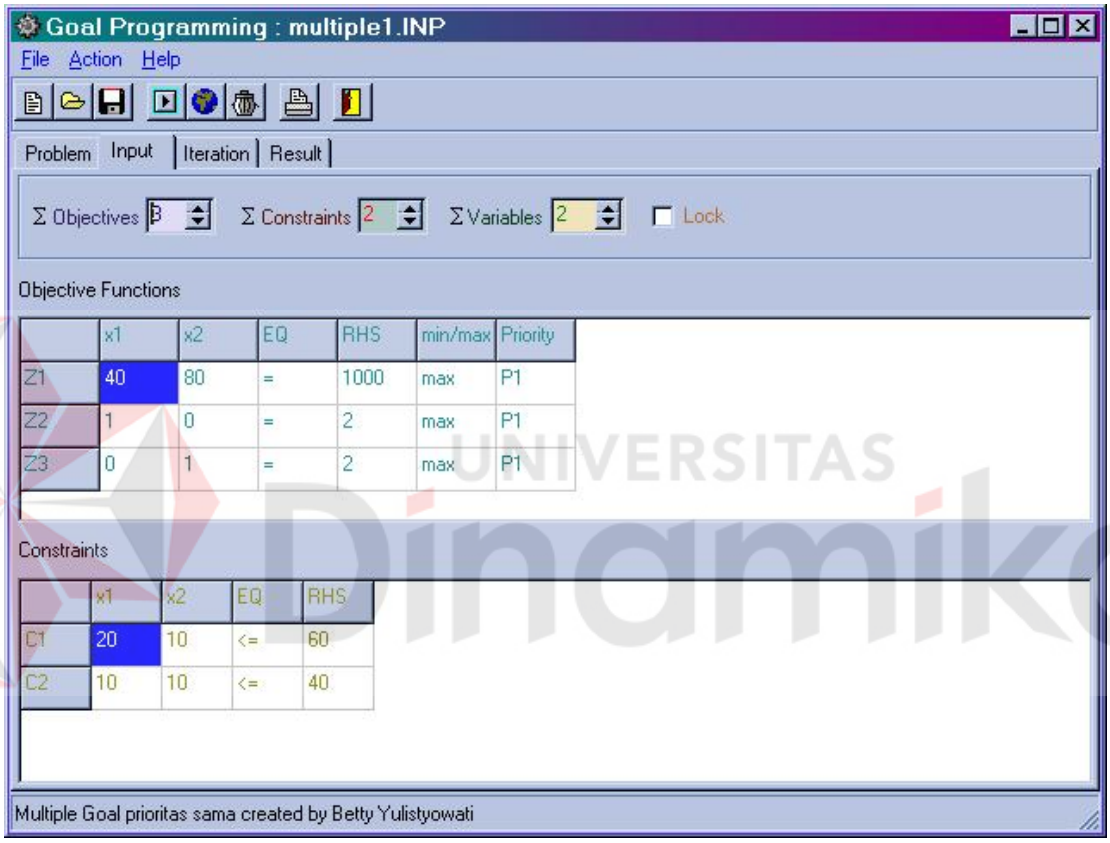
$$40x_1 + 80x_2 + d_1^- - d_2^+ = 1000$$

$$x_1 + + d_2^- - d_2^+ = 2$$

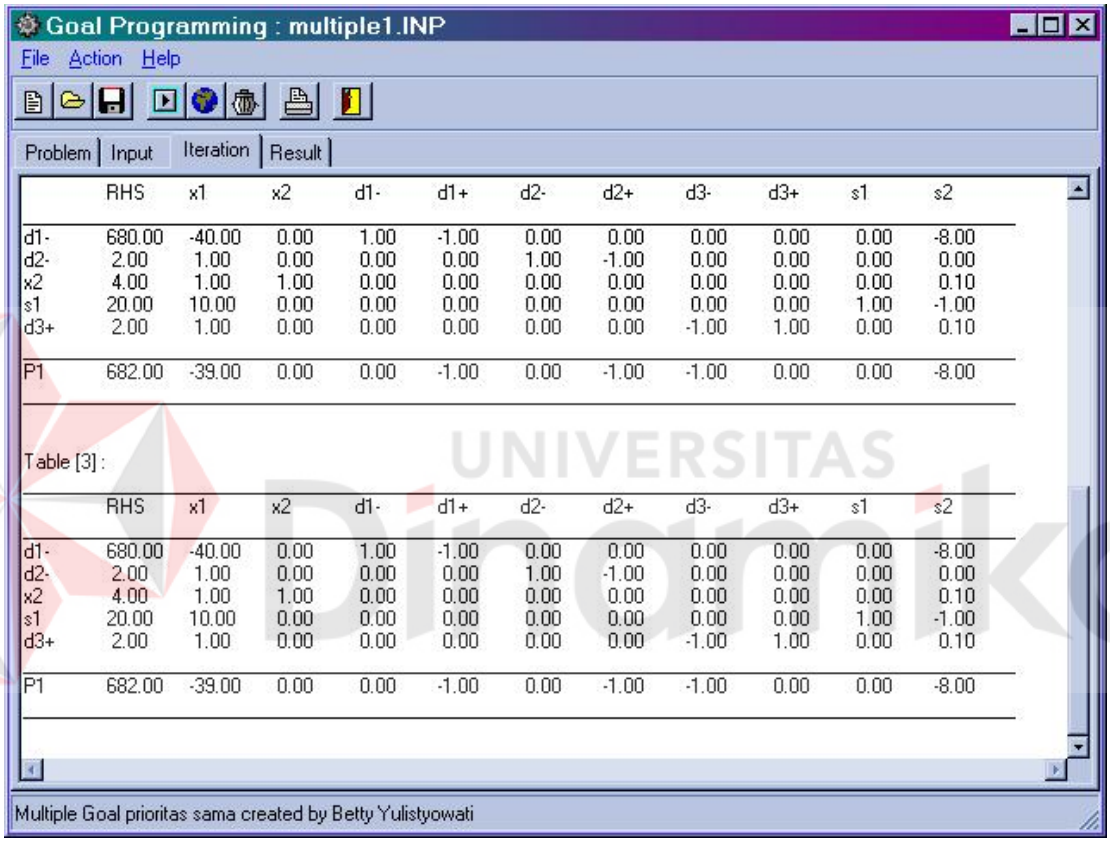
$$x_2 + d_3^- - d_3^+ = 2$$

$$x_1, \quad x_2, \quad d_1^-, \quad d_1^+, \quad d_2^-, \quad d_2^+, \quad d_3^-, \quad d_3^+, \; ³ \quad 0.$$

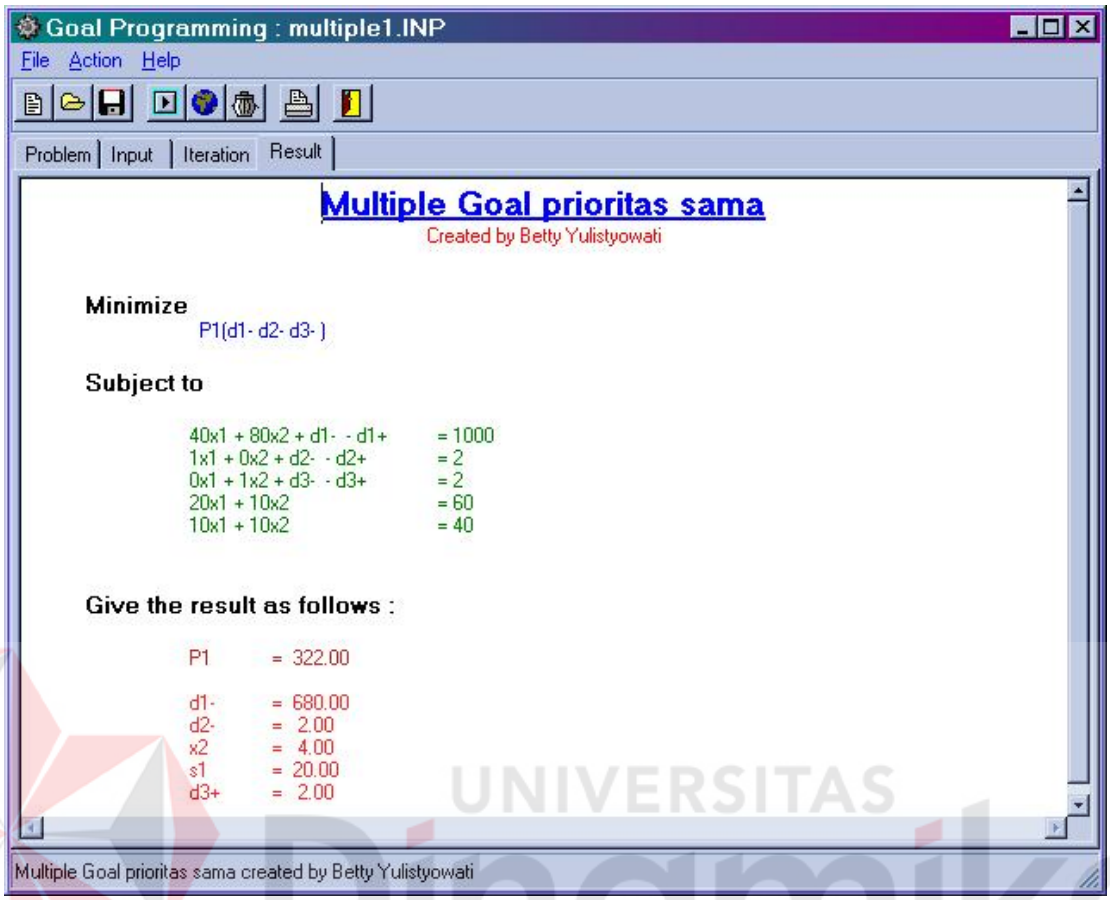
Karena variabel simpangan pada fungsi tujuan tidak memiliki prioritas, persoalan ini dapat diselesaikan dengan metode simplex. Solusinyan adalah $x_1=0$, $x_2=4$, $s_1=20$, $s_2=0$,$d_1^-= 680$, $d_1^+=0$, $d_2^-=0$, $d_2^+=0$, $d_3^-=0$, $d_3^+=2$, dan $Z=682$. Solusi model ini tidak mencapai tujuan produksi (paling tidak dua unit x_1 dan dua unit x_2) karena tidak ada prioritas untuk tujuan-tujuan itu. Model itu hanya meminimmkan jumlah simpangan untuk semua tujuan.



Gambar 4.11. Tampilan input Multiple Goal Programming Prioritas sama



Gambar 4.12. Tampilan iterasi dari prioritas sama



Gambar 4.13. Tampilan Hasil dari prioritas sama

4.4.2. Multiple Goal Programming dengan prioritas

Model untuk masalah ini serupa dengan model tanpa prioritas sebelumnya kecuali penambahan koefisisen prioritas pada fungsi tujuan/ Model yang baru itu adalah :

Formulasi persamaan Goal Programming

Minimize :

$$Z = P_1 d_2^- + P_1 d_3^- + P_2 d_1^+$$

Subject To :

$$20x_1 + 10x_2 \quad £ \; 60$$

$$10x_1 + 10x_2 \quad £ \; 40$$

$$40x_1 + 80x_2 + d_1^- - d_1^+ = 1000$$

$$x_1 + d_2^- - d_2^+ = 20$$

$$x_2 + d_3^- - d_3^+ = 2$$

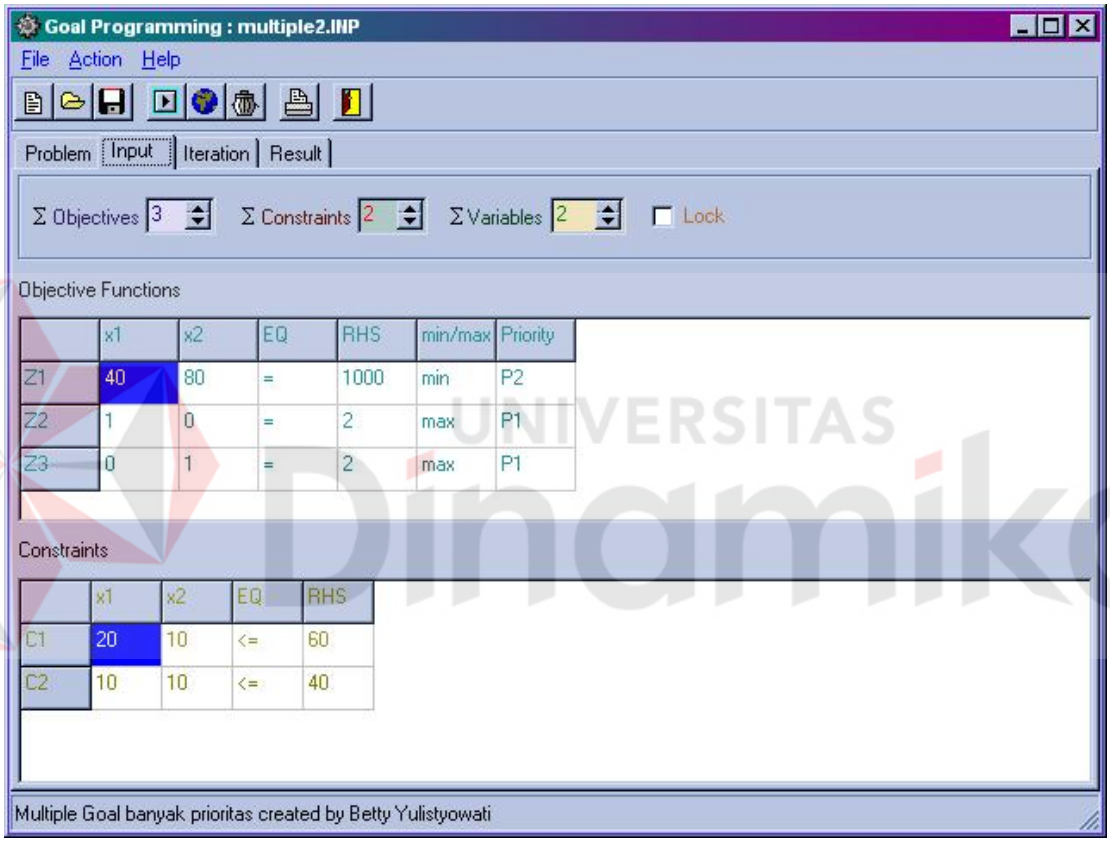
$$x_1, \; x_2, d_1^-, \; d_1^+, \; d_2^-, \; d_2^+, \; d_3^-, \; d_3^+ \; ³ \; 0$$

Penerapan algoritma simplex yang dimodifikasi pada masalah ini menghasilkan x_1 = 2, x_2 = 2, s_1 =0, d_1^-= 760, d_1^+=0, d_1^+=0, d_2^-=0, d_2^+=0, d_3^-=0, d_3^+=0, dan Z = 760P_2. Penafsiran solusi ini adalah :

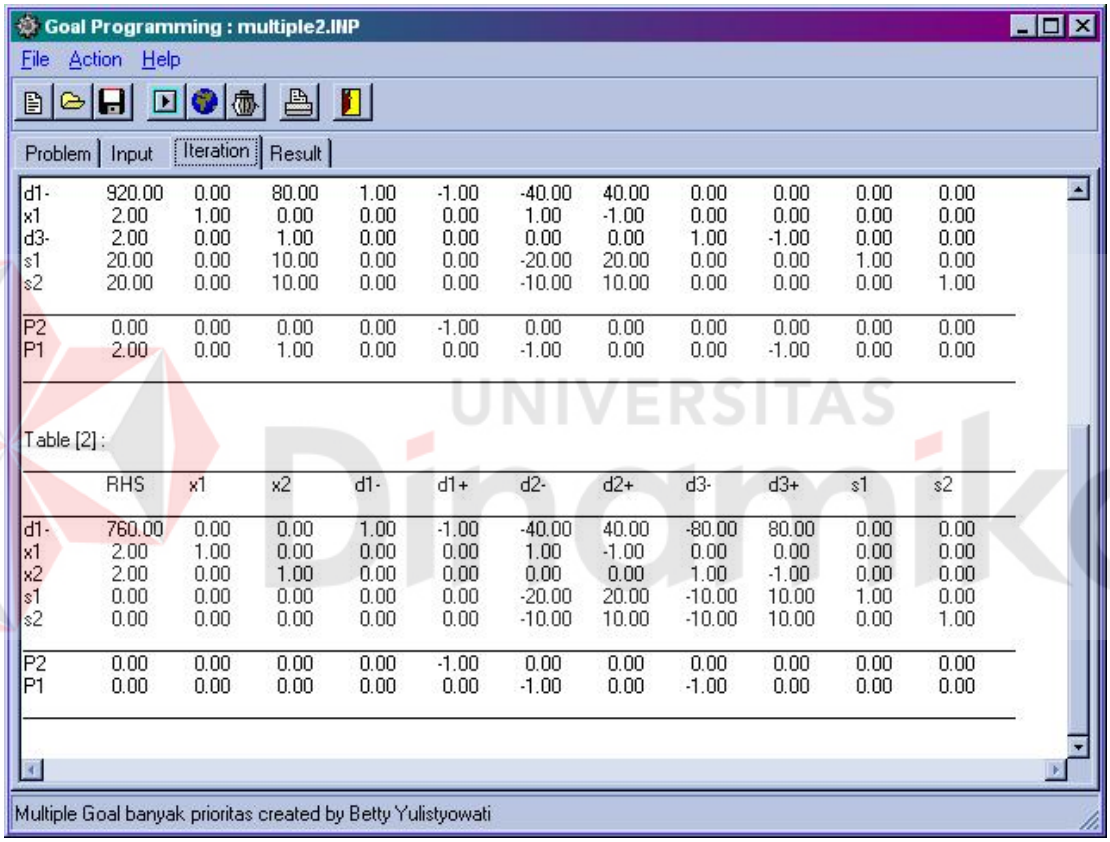
Tujuan 1 (P_1) dicapai : dua unit x_1 dan x_2 diproduksi

Tujuan 2 (P_2) tak tercapai : sumbangan total terhadap keuntungan adalah 240 (=1000–760), kekurangan target keuntungan sebesar 760 (d_1^-=760).

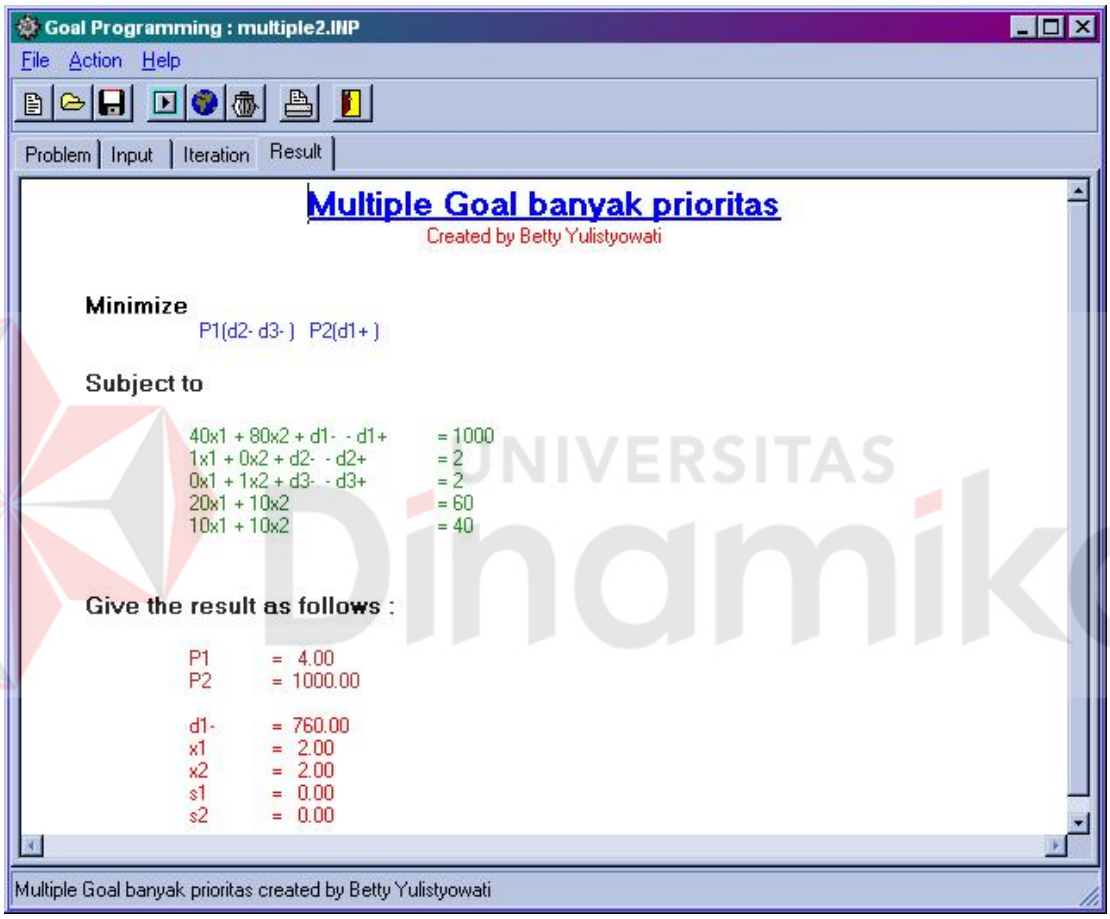
Membandingkan hasil ini dengan hasil sebelumnya, terlihat bahwa pemilik harus mengorbankan keuntungan (240 lawan 320) untuk memenuhi tujuan produksi paling tidak 2 unit tiap barang. Dan Perhitunngan pada aplikasinya menghasilkan seperti dibawah ini.



Gambar 4.14. Tampilan Input banyak prioritas tanpa bobot



Gambar 4.15. Tampilan iterasi banyak prioritas tanpa Bobot



Gambar 4.16. Tampilan Hasil Banyak prioritas tanpa Bobot

4.4.3. Multiple Goal Programming dengan prioritas dan bobot

Dengan simbol – simbol seperti pada persoalan sebelumnya, model yang telah dimodifikasikan itu adalah :

Formulasi persamaan Goal Programming

Minimize :

$$Z = P_1 d_1^+ + P_2 d_2^+ + 30P_2 d_3^- + 40P_3 d_3^-$$

Subject to :

$$2x_1 + 40x_1 + d_1^- + d_1^+ = 80$$

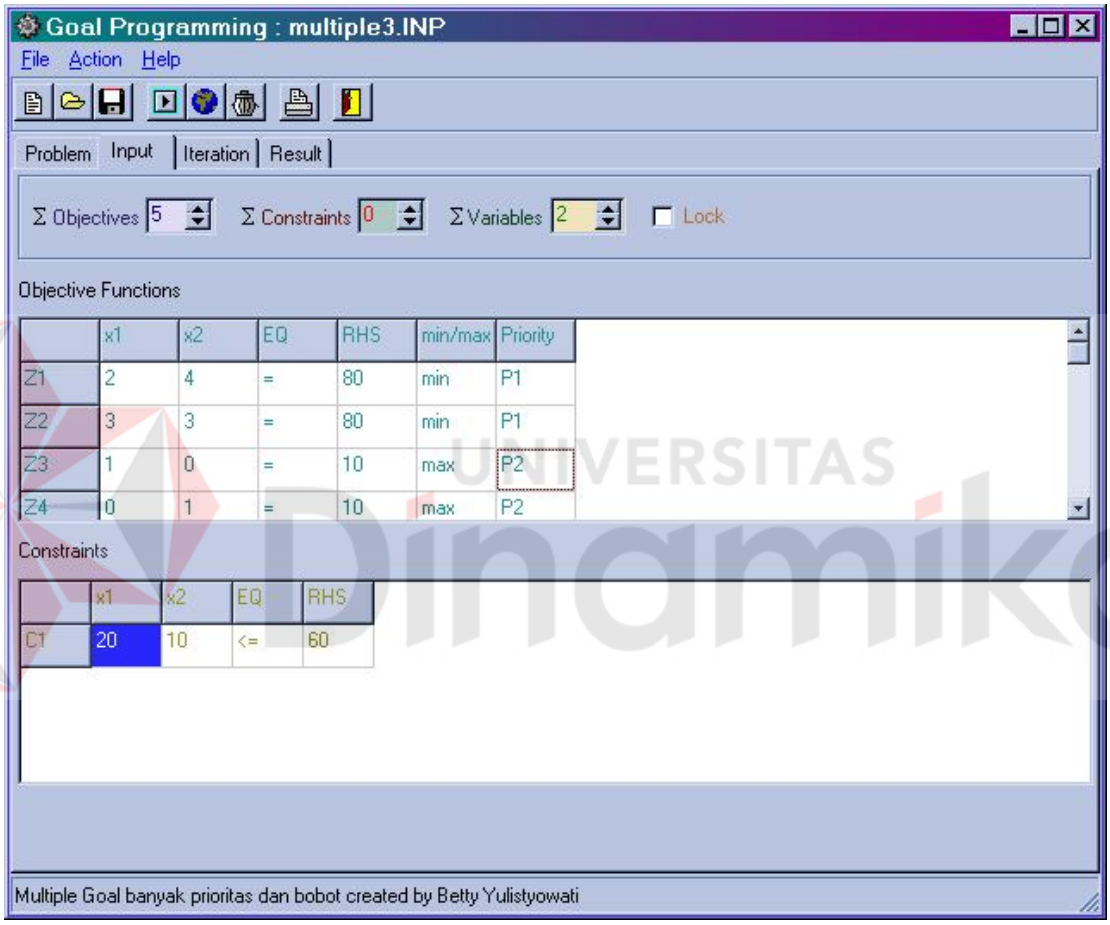
$$3x_1 + 4x_2 + d_2^- - d_2^+ = 80$$

$$x_1 + d_3^- - d_3^+ = 10$$

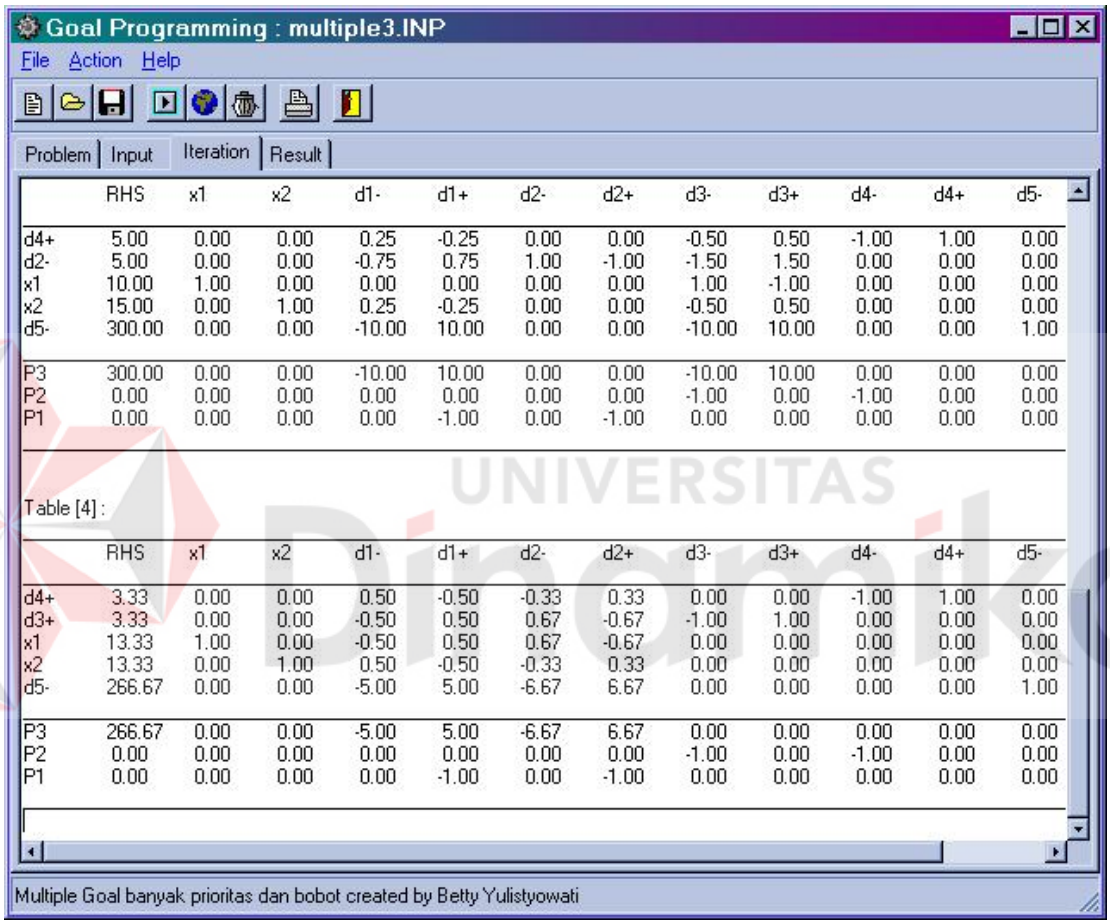
$$x_2 + d_4^- - d_4^+ = 10$$

$$30x_1 + 40x_2 + d_4^- - d_5^+ = 1200$$

$$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+, d_5^-, d_5^+$$



Gambar 4.17. Tampilan Input banyak prioritas dan bobot

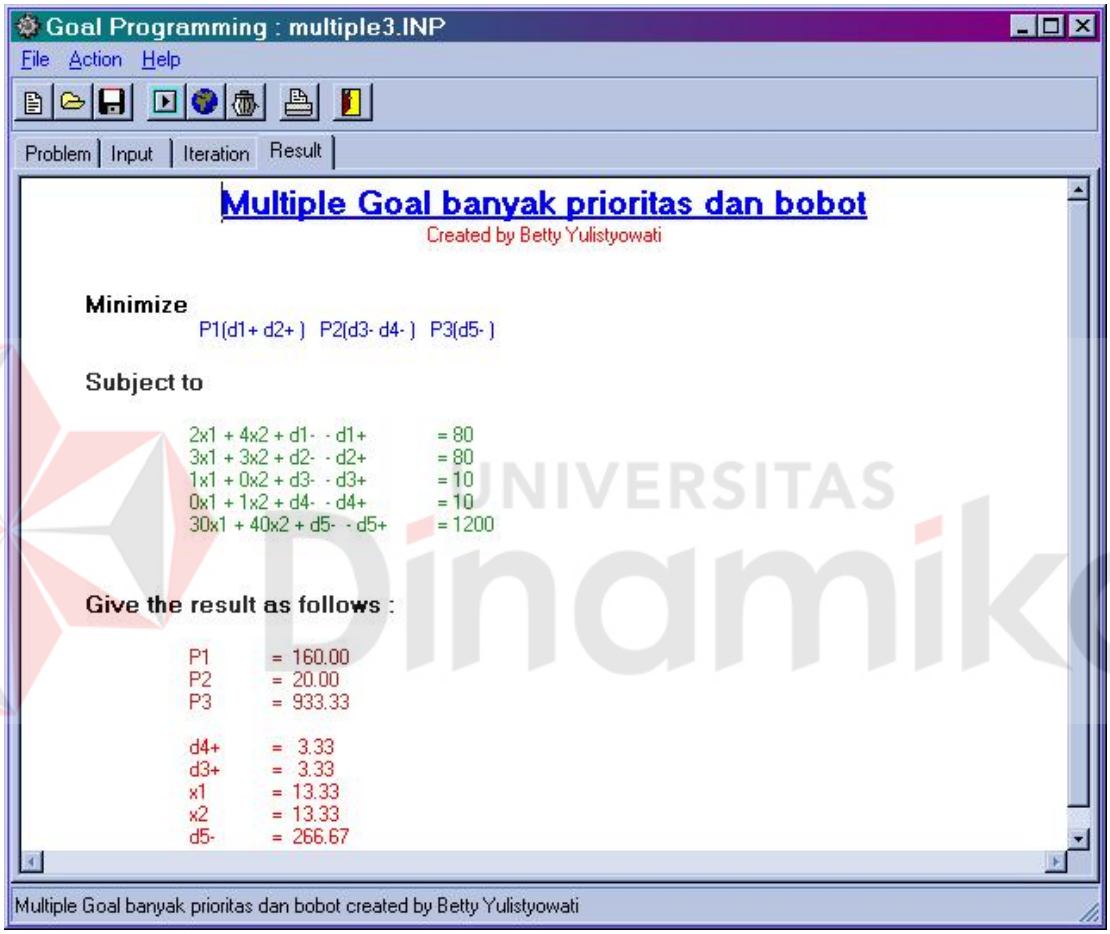


	RHS	x1	x2	d1-	d1+	d2-	d2+	d3-	d3+	d4-	d4+	d5-
d4+	5.00	0.00	0.00	0.25	-0.25	0.00	0.00	-0.50	0.50	-1.00	1.00	0.00
d2-	5.00	0.00	0.00	-0.75	0.75	1.00	-1.00	-1.50	1.50	0.00	0.00	0.00
x1	10.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	-1.00	0.00	0.00	0.00
x2	15.00	0.00	1.00	0.25	-0.25	0.00	0.00	-0.50	0.50	0.00	0.00	0.00
d5-	300.00	0.00	0.00	-10.00	10.00	0.00	0.00	-10.00	10.00	0.00	0.00	1.00
P3	300.00	0.00	0.00	-10.00	10.00	0.00	0.00	-10.00	10.00	0.00	0.00	0.00
P2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00	0.00	-1.00	0.00	0.00
P1	0.00	0.00	0.00	0.00	-1.00	0.00	-1.00	0.00	0.00	0.00	0.00	0.00

Table [4] :

	RHS	x1	x2	d1-	d1+	d2-	d2+	d3-	d3+	d4-	d4+	d5-
d4+	3.33	0.00	0.00	0.50	-0.50	-0.33	0.33	0.00	0.00	-1.00	1.00	0.00
d3+	3.33	0.00	0.00	-0.50	0.50	0.67	-0.67	-1.00	1.00	0.00	0.00	0.00
x1	13.33	1.00	0.00	-0.50	0.50	0.67	-0.67	0.00	0.00	0.00	0.00	0.00
x2	13.33	0.00	1.00	0.50	-0.50	-0.33	0.33	0.00	0.00	0.00	0.00	0.00
d5-	266.67	0.00	0.00	-5.00	5.00	-6.67	6.67	0.00	0.00	0.00	0.00	1.00
P3	266.67	0.00	0.00	-5.00	5.00	-6.67	6.67	0.00	0.00	0.00	0.00	0.00
P2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00	0.00	-1.00	0.00	0.00
P1	0.00	0.00	0.00	0.00	-1.00	0.00	-1.00	0.00	0.00	0.00	0.00	0.00

Gambar 4.18. Tampilan iterasi banyak prioritas dan bobot



Gambar 4.19. Tampilan Hasil banyak prioritas dan bobot

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari pembuatan Tugas Akhir ini dapat diambil beberapa kesimpulan sebagai berikut :

1. Bahwa sesuai tujuannya aplikasi ini telah memenuhi tujuannya sebagai alat perhitungan bagi user untuk mengerjakan perhitungan-perhitungan yang terkait dengan pencarian optimasi dan validasi dari suatu permasalahan sangat diperlukan untuk menghemat waktu, karena telah terbukti validasi, kecepatannya yang dihasilkan akan lebih terjamin dibandingkan dengan menggunakan aplikasi manual.
2. Dan dengan aplikasi sistem ini dapat dengan mudah digunakan oleh user. Dengan penggunaan desain input output yang mudah dimengerti sehingga bisa menjadi *user friendly*..

Disamping dapat membantu perhitungan linier goal programming aplikasi ini juga dapat digunakan dalam melakukan perhitungan simplex standard.

5.2. Saran

Perangkat lunak bantu yang dihasilkan masih berupa prototipe dan masih memiliki kekurangan dan memerlukan banyak tambahan kelengkapan. Berikut ini berisi saran yang berguna untuk pemikiran maupun implementasi lebih lanjut :

1. Dalam perangkat lunak bantu ini didesain dan dikembangkan dengan menggunakan algoritma dan metode tertentu (selected algorithm), sehingga masih memungkinkan untuk mengembangkan perangkat lunak bantu ini dengan menggunakan alternatif algoritma yang digunakan. Misalnya untuk proses iterasi linier Goal programming menggunakan *Metode Update Fungsi Objektif*.
2. Belum adanya visualisasi grafis dalam perangkat lunak bantu ini, sehingga diharapkan ada pembaca yang mau mengembangkan perangkat lunak bantu ini.
3. Segi kemudahan interaksi perangkat lunak yang dibuat perlu ditingkatkan dengan penyediaan beberapa obyek template dari Editor Input. Atau memberikan fasilitas untuk memasukkan fungsi objektif dan constraint berupa ekspresi matematiknya sehingga mempermudahkan pemakai dalam melakukan input persamaan.

DAFTAR PUSTAKA

Bernard W. Taylor III,1982, ***Introduction to Management Science***, 2nd Edition, Wm.C.Brown Publisher.

Djoko Purnomo, 1998, ***Mudah Menguasai Delphi 3.0 jilid 1 dan jilid 2***, Elex Media Computindo Gramedia Group, Jakarta.

Hillier and Lieberman, 1986, ***Introduction to Operations research***, McGraw-Hill Publishing company, New York.

K. Roscoe Davis, Petrick G. McKeown, Terry R. Rakes, 1982, ***Management Science An Introduction***, Kent Publishing Company A Division of Wadsworth, Inc

Pangestu Subagyo, Marwan Asri, T. Hani Handoko, 1989, ***Dasar-dasar Operations Research***, BPFE Yogyakarta

Sri Mulyono, 1991, ***Operations Research,*** lembaga penerbit Fakultas Ekonomi Universitas Indonesia

Steur E. Ralp**,** 1998, ***Multiple Criteria Optimization*** *: Theory, Computation and Aplication*, ohn Wiley & Son, Inc.

Taha Hamdy, A, 1987, ***Operation Research : An Introduction***, 4th Edition, MacMillan Publishing Company,

Lampiran. Listing Program.

```
unit GoalProgramming;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, Menus, ToolWin, StdCtrls, Grids, Spin, ExtCtrls;

type
    TForm1 = class(TForm)
    PageControl1: TPageControl;
    StatusBar: TStatusBar;
    ToolBar1: TToolBar;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    SaveAs: TMenuItem;
    Exit1: TMenuItem;
    N2: TMenuItem;
    Action1: TMenuItem;
    Run1: TMenuItem;
    N3: TMenuItem;
    Help1: TMenuItem;
    Content1: TMenuItem;
    N4: TMenuItem;
    About1: TMenuItem;
    Input: TTabSheet;
    Iterasi: TTabSheet;
    Result: TTabSheet;
    ButtonNew: TToolButton;
    ButtonOpen: TToolButton;
    ButtonSave: TToolButton;
    ButtonRun: TToolButton;
    ButtonWeight: TToolButton;
    ButtonOverValue: TToolButton;
    ImageList1: TImageList;
    ToolButton4: TToolButton;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    SpinObj: TSpinEdit;
    SpinConst: TSpinEdit;
    SpinVar: TSpinEdit;
    Label7: TLabel;
    Label8: TLabel;
    GridObj: TStringGrid;
    Label9: TLabel;
    Label10: TLabel;
    GridConst: TStringGrid;
    Lock: TCheckBox;
    New1: TMenuItem;
    N1: TMenuItem;
    Save1: TMenuItem;
    ToolButton9: TToolButton;
```

```
ButtonExit: TToolButton;
AddWeight1: TMenuItem;
AddOverValue1: TMenuItem;
OpenDialog1: TOpenDialog;
SaveDialog1: TSaveDialog;
Memo2: TMemo;
RCResult: TRichEdit;
Problem: TTabSheet;
ToolBar2: TToolBar;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
REProblem: TRichEdit;
ImageList2: TImageList;
ColorDialog1: TColorDialog;
ToolButton5: TToolButton;
ToolButton3: TToolButton;
ToolButton6: TToolButton;
ToolButton7: TToolButton;
SpinEdit1: TSpinEdit;
Bullet: TToolButton;
LeftIndent: TToolButton;
ReduceLeftIndent: TToolButton;
BoldButton: TToolButton;
ItalicButton: TToolButton;
UnderlineButton: TToolButton;
ToolButton15: TToolButton;
Printsetup: TPrinterSetupDialog;
PrintDialog: TPrintDialog;
N5: TMenuItem;
Print1: TMenuItem;
Memo1: TRichEdit;
procedure LockClick(Sender: TObject);
procedure SpinObjChange(Sender: TObject);
procedure SpinConstChange(Sender: TObject);
procedure SpinVarChange(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure GridObjKeyPress(Sender: TObject; var Key: Char);
procedure GridConstKeyPress(Sender: TObject; var Key: Char);
procedure About1Click(Sender: TObject);
procedure Run1Click(Sender: TObject);
procedure Parsing;
procedure ShowTable;
procedure ArrayClear;
procedure GridClear;
function  CheckLock : boolean;
procedure BuildInitialTable;
procedure ButtonRunClick(Sender: TObject);
procedure ButtonNewClick(Sender: TObject);
procedure ButtonExitClick(Sender: TObject);
procedure AddWeight1Click(Sender: TObject);
procedure AddOverValue1Click(Sender: TObject);
procedure ButtonWeightClick(Sender: TObject);
procedure ButtonOverValueClick(Sender: TObject);
procedure Save1Click(Sender: TObject);
procedure New1Click(Sender: TObject);
procedure SaveData(FileName : string);
procedure ButtonSaveClick(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure ButtonOpenClick(Sender: TObject);
procedure SaveAsClick(Sender: TObject);
```

```
    procedure FormCreate(Sender: TObject);
    procedure Modified_Simplex;
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure Content1Click(Sender: TObject);
    procedure ToolButton1Click(Sender: TObject);
    procedure ToolButton2Click(Sender: TObject);
    procedure ToolButton5Click(Sender: TObject);
    procedure ToolButton6Click(Sender: TObject);
    procedure REProblemSelectionChange(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure BoldButtonClick(Sender: TObject);
    procedure ItalicButtonClick(Sender: TObject);
    procedure UnderlineButtonClick(Sender: TObject);
    procedure BulletClick(Sender: TObject);
    procedure LeftIndentClick(Sender: TObject);
    procedure ReduceLeftIndentClick(Sender: TObject);
    procedure REProblemChange(Sender: TObject);
    procedure Print1Click(Sender: TObject);

  private
    { Private declarations }

  public
    { Public declarations }
  end;

const garis = '_________';

var
  Form1: TForm1;
  ObjArray, ConstArray   : array[1..30,1..10] of real;
  TempGoal               : array[1..30,1..10] of real;
  ProsesArray            : array[1..30,1..20] of real;
  RHS                  : array[1..20] of real;
  fucker               : array[1..20] of real;
  RHSConst               : array[1..10] of real;
  weights              : array[1..20] of real;
  PTy                  : array[1..20] of string[5];
  EQ, EQi              : array[1..10] of string[5];
  objective            : array[1..10] of string;
  minmax               : array[1..10] of string[5];
  UpperLabel             : array[1..30] of string[5];
  LeftLabel            : array[1..20] of string[5];
  fname                : string;
  dev, devIndex, slack   : integer;
  count                : integer;
  holderI,holderJ        : integer;
  GoalPointer            : integer;
  ZP                   : integer;
  c, code, cnt           : integer;
  JmlBaris, JmlKolom, k  : integer;
  useWeight,  saved,
  useOverValue           : boolean;
  fs                   : array[1..3] of boolean;


implementation

uses About, Weight, OverValue, Title, ShellApi, Splash;
```

```
{$R *.DFM}



{Untuk menginisialisasi array agar bernilai nol semua}
procedure TForm1.ArrayClear;
var i,j : integer;
begin
  for j := 1 to 10 do
  begin
    for i := 1 to 20 do
    begin
      ObjArray[i,j]   := 0;
      ConstArray[i,j] := 0;
    end;
    EQ[j]     := '';
    EQi[j]    := '';
    minmax[j] := '';
  end;
end;

procedure Heading1;
begin
  with form1.RCResult do
  begin
    lines.Clear;
    Paragraph.Alignment := taCenter;
    SelAttributes.Size := 14;
    SelAttributes.Color := clBlue;
    SelAttributes.Style := [fsbold,fsunderline];
    lines.Add(FormTitle.edit1.Text);
    SelAttributes.Color := clred;
    SelAttributes.Size := 4;
    SelAttributes.Style := [];
    lines.Add('Created by '+FormTitle.edit2.Text);
    lines.Add('');lines.Add('');
  end;
end;

procedure Heading2;
var st : string;
    i  : integer;
begin
  st := '';
  for i := 1 to goalPointer do
  begin
    st := st +LeftLabel[holderJ+GoalPointer-i+1]+ '('+objective[i]+')   ';
  end;

  with Form1.RCResult do
  begin
    Paragraph.Alignment := taLeftJustify;
    Paragraph.LeftIndent := 10;
    SelAttributes.Color := clblack;
    SelAttributes.Size := 10;
    SelAttributes.Style := [fsbold];
    lines.Add('         Minimize ');
    //lines.Add('');
    SelAttributes.Color := clblue;
```

```
    SelAttributes.Size := 8;
    SelAttributes.Style := [];
    lines.Add(#9+#9+#9+'  '+st);
  end;
end;

procedure Heading3;
var st,s   : string;
   i,j,c  : integer;
begin
  c := 0;
  with Form1.RCResult do
  begin
    Paragraph.Alignment := taLeftJustify;
    Paragraph.LeftIndent := 10;
    SelAttributes.Color := clblack;
    SelAttributes.Size := 10;
    SelAttributes.Style := [fsbold];
    lines.add('');
    lines.Add('         Subject to ');
    lines.Add('');
    SelAttributes.Color := clgreen;
    SelAttributes.Size := 8;
    SelAttributes.Style := [];
    for j := 1 to Form1.SpinObj.Value do
    begin
      st := '';
      for i := 1 to Form1.SpinVar.Value do
      begin
        s := inttostr(j);
        st := st + form1.GridObj.Cells[i,j] + UpperLabel[i]+' + ';

      end;
      st := st + 'd'+s+'- '+' - '+'d'+s+'+';
      st := st + #9 + '= '+ form1.GridObj.Cells[form1.SpinVar.value+2,j];
      lines.Add(#9+#9+#9+st);
    end;
    if useOverValue then
    begin
      for j := 1 to FormOverValue.GridOverValue.RowCount-1 do
      begin
        st := '';
        for i := form1.SpinVar.Value+1 to devIndex-1 do
        begin
          if prosesArray[i,form1.spinobj.value+J] <> 0 then
          begin
            inc(c);
            case c of
            1 : s := '';
            2 : s := #9+'+ ';
            3 : s := ' - ';
            end;
            st := st+s+UpperLabel[i-1];
          end;
        end;
        st := st + #9+'= '+ FormOverValue.GridOverValue.Cells[1,j];
        lines.Add(#9+#9+#9+st);
      end;
    end;
    for j := 1 to Form1.SpinConst.Value do
```

```
    begin
      st := '';
      for i := 1 to Form1.SpinVar.Value do
      begin
        s := inttostr(j);
        st := st + form1.GridConst.Cells[i,j] + UpperLabel[i]+' + ';

      end;
//      st := st + 'd'+s+'- '+' - '+'d'+s+'+';
      delete(st,length(st)-2,2);
      st := st + #9+#9 + '= '+ form1.GridConst.Cells[form1.SpinVar.value+2,j];
      lines.Add(#9+#9+#9+st);
    end;
  end;

end;

procedure Result;
var  i  : integer;
    temp : real;
    st,s : string;
begin
  with   Form1.RCResult do
  begin
    Paragraph.Alignment := taLeftJustify;
    Paragraph.LeftIndent := 10;
    SelAttributes.Color := clblack;
    SelAttributes.Size := 10;
    SelAttributes.Style := [fsbold];
    Lines.Add('');
    Lines.Add('');
    Lines.Add('        Give the result as follows :');
    lines.Add('');
    SelAttributes.Color := clred+180;
    SelAttributes.Size := 8;
    SelAttributes.Style := [];

   st := '';
   for i := 1 to goalPointer do
   begin
     temp  := fucker[i] - prosesArray[1,holderJ+GoalPointer-i+1];
     str(temp:5:2,s);
     st := LeftLabel[holderJ+GoalPointer-i+1]+ #9+'=  '+ s;
     Lines.Add(#9+#9+#9+st);
   end;
   lines.Add('');
   SelAttributes.Color := clred+250;
   for i := 1 to holderJ do
   begin
     str(prosesArray[1,i]:5:2,s);
     st := LeftLabel[i]+#9+ '=  '+ s;
     Lines.Add(#9+#9+#9+st);
   end;
  end;
end;

procedure ShowResult;
begin
  Heading1;
  Heading2;
```

```
  Heading3;
  Result;
  Form1.RCResult.SelStart := 0;
end;


{Untuk membersihkan grid untuk isian fungsi objective dan constrain}
procedure TForm1.GridClear;
var i,j : integer;
begin
   for j := 0 to gridObj.RowCount-1 do
      for i := 0 to gridObj.ColCount-1 do
         gridObj.Cells[i,j] := '';

   for j := 0 to gridConst.RowCount-1 do
      for i := 0 to gridConst.ColCount-1 do
         gridConst.Cells[i,j] := '';
end;


{Mengecek apakah jumlah variabel dan jumlah fungsi sudah dilock}
function TForm1.checkLock : boolean;
begin
  if not(lock.Checked) then
  begin
    messagedlg('Lock the Grid first!',mtWarning, [mbOK], 0);
    checkLock := false
  end
  else checkLock := true;
end;


{Mengaktifkan atau menon-aktifkan lock thd jml variabel dan jml fungsi}
procedure TForm1.LockClick(Sender: TObject);
begin
   if Lock.Checked then
   begin
     spinObj.Enabled := false;
     spinConst.Enabled := false;
     spinVar.Enabled := false;
   end
   else begin
     spinObj.Enabled := true;
     spinConst.Enabled := true;
     spinVar.Enabled := true;
   end;
end;


{Menambah atau mengurangi jumlah fungsi objective}
procedure TForm1.SpinObjChange(Sender: TObject);
var i : integer;
begin
   GridObj.RowCount := spinObj.Value+1;
   for i := 1 to spinObj.Value+1 do
      GridObj.Cells[0,i] := 'Z'+inttostr(i);
end;


{Menambah atau mengurangi jumlah constraint}
```

```
procedure TForm1.SpinConstChange(Sender: TObject);
var i : integer;
begin
  if spinconst.Value > 0 then
  begin
    GridConst.RowCount := spinConst.Value+1;
    for i := 1 to spinConst.Value+1 do
      GridConst.Cells[0,i] := 'C'+inttostr(i);
    GridConst.Enabled := true;
  end
  else GridConst.Enabled := false;
end;

{Menambah atau mengurangi jumlah variabel}
procedure TForm1.SpinVarChange(Sender: TObject);
var i : integer;
begin
  GridObj.ColCount := SpinVar.Value+5;
  GridConst.ColCount := SpinVar.Value+3;
  for i := 1 to spinVar.Value do
  begin
    GridObj.Cells[i,0] := 'x'+inttostr(i);
    GridConst.Cells[i,0] := 'x'+inttostr(i);
  end;
  GridObj.Cells[Spinvar.Value+1,0] := 'EQ';
  GridConst.Cells[Spinvar.Value+1,0] := 'EQ';
  GridObj.Cells[Spinvar.Value+2,0] := 'RHS';
  GridConst.Cells[Spinvar.Value+2,0] := 'RHS';
  GridObj.Cells[Spinvar.Value+3,0] := 'min/max';
  GridObj.Cells[Spinvar.Value+4,0] := 'Priority';
end;

{Keluar dari aplikasi}
procedure TForm1.Exit1Click(Sender: TObject);
begin
  close;
end;

{Mengisi atau mengganti tanda kesamaan dan tanda minimasi atau maksimasi dengan
 penekanan <ENTER> pada cell yang aktif dan sesuai dengan kolom yang disediakan}
procedure TForm1.GridObjKeyPress(Sender: TObject; var Key: Char);
var x,y : longint;
    temp: integer;
begin
  saved := false;
  x := GridObj.Col;
  y := GridObj.row;
  if useovervalue then temp := FormOverValue.GridOverValue.RowCount
  else temp := 0;
  if key = #13 then
  begin
    if x = spinVar.Value+1 then
    begin
      if GridObj.Cells[x,y] = '' then
        GridObj.Cells[x,y] := '<='
      else if GridObj.Cells[x,y] = '<=' then
        GridObj.Cells[x,y] := '>='
      else if GridObj.Cells[x,y] = '>=' then
        GridObj.Cells[x,y] := '='
      else if GridObj.Cells[x,y] = '=' then
```

```
          GridObj.Cells[x,y] := '<'
        else if GridObj.Cells[x,y] = '<' then
          GridObj.Cells[x,y] := '>'
        else if GridObj.Cells[x,y] = '>' then
          GridObj.Cells[x,y] := '<='
        else GridObj.Cells[x,y] := '<=';
      end
      else if x = spinVar.Value+3 then
      begin
        if GridObj.Cells[x,y] = '' then
          GridObj.Cells[x,y] := 'min'
        else if GridObj.Cells[x,y] = 'min' then
          GridObj.Cells[x,y] := 'max'
        else if GridObj.Cells[x,y] = 'max' then
          GridObj.Cells[x,y] := 'min'
        else GridObj.Cells[x,y] := 'min';
      end
      else if x=spinvar.Value+4 then
      begin
        if GridObj.Cells[x,y] = '' then count := 1
        else if count = SpinObj.Value+temp then count := 1
            else inc(count);
        GridObj.Cells[x,y] := 'P'+inttostr(count);
      end;
    end;
end;

{Mengisi atau mengganti tanda kesamaan dengan penekanan <ENTER> pada cell yang
 aktif dan sesuai dengan kolom yang disediakan}
procedure TForm1.GridConstKeyPress(Sender: TObject; var Key: Char);
var x,y : longint;
begin
   saved := false;
   x := GridConst.Col;
   y := GridConst.row;
   if  x = spinVar.Value+1 then
     if key = #13 then
     begin
       if GridConst.Cells[x,y] = '' then
         GridConst.Cells[x,y] := '<='
       else if GridConst.Cells[x,y] = '<=' then
         GridConst.Cells[x,y] := '>='
       else if GridConst.Cells[x,y] = '>=' then
         GridConst.Cells[x,y] := '='
       else if GridConst.Cells[x,y] = '=' then
         GridConst.Cells[x,y] := '<'
       else if GridConst.Cells[x,y] = '<' then
         GridConst.Cells[x,y] := '>'
       else if GridConst.Cells[x,y] = '>' then
         GridConst.Cells[x,y] := '<='
       else GridConst.Cells[x,y] := '<=';
     end;
end;

{Mengeluarkan aboutbox}
procedure TForm1.About1Click(Sender: TObject);
begin
  aboutbox.ShowModal;
end;
```

```
procedure TForm1.Run1Click(Sender: TObject);
begin
  Parsing;
  BuildInitialTable;
  memo1.Clear;
  memo1.Lines.Add('Initial Table :');
  ShowTable;
  if MessageDlg('It"s about to run your project.'+#13+#13+' Are you sure?',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    cursor := crHourGlass;
    Modified_Simplex;
    showResult;
    pagecontrol1.ActivePage := result;
    cursor := crDefault;
  end;
end;

procedure TForm1.ButtonRunClick(Sender: TObject);
begin
  Run1Click(Sender);
end;

procedure TForm1.ButtonNewClick(Sender: TObject);
begin
  ArrayClear;
  GridClear;
  memo1.Clear;
  REProblem.Clear;
  RCResult.Clear;
  caption := 'Goal Programming : New Project';
  statusbar.SimpleText := 'New Project';
  SpinObjChange(sender);
  SpinConstChange(sender);
  SpinVarChange(sender);
  saved := false;
end;

procedure TForm1.ButtonExitClick(Sender: TObject);
begin
  close;
end;

{Menampilkan kotak isian untuk Weight}
procedure TForm1.AddWeight1Click(Sender: TObject);
begin
 if checkLock then
    FormWeight.showmodal;
end;

{Menampilkan kotak isian untuk Overvalue Equation}
procedure TForm1.AddOverValue1Click(Sender: TObject);
begin
 if checkLock then
    FormOverValue.showmodal;
end;

procedure TForm1.ButtonWeightClick(Sender: TObject);
```

```
begin
  if checkLock then
     FormWeight.showmodal;
end;

procedure TForm1.ButtonOverValueClick(Sender: TObject);
begin
  if checkLock then
     FormOverValue.showmodal;
end;

{     if useWeight then lines.Add('Weight='+'yes')
        else lines.Add('Weight='+'no');
     if useOverValue then lines.Add('OverValue='+'yes')
        else lines.Add('OverValue='+'no');
}

{_____Begin Save Data_____}

procedure Header;
begin
   with Form1.memo2 do
   begin
     lines.add('Title='+FormTitle.edit1.Text);
     lines.add('Creator='+FormTitle.Edit2.Text);
     lines.Add('Var=' + inttostr(Form1.Spinvar.Value));
     lines.Add('Obj=' + inttostr(Form1.SpinObj.Value));
     lines.Add('Con=' + inttostr(Form1.SpinConst.Value));
   end;
end;

procedure ObjLines;
var i,j : integer;
    st  : string;
begin
   st := '';
   for j := 0 to Form1.gridObj.RowCount-1 do
   begin
      for i := 0 to Form1.gridObj.ColCount-1 do
      begin
         st := st + Form1.gridObj.Cells[i,j] + ' ';
      end;
      Form1.memo2.Lines.Add(st);
      st := '';
   end;
end;

procedure ConstLines;
var i,j : integer;
    st  : string;
begin
   st := '';
   for j := 0 to Form1.gridConst.RowCount-1 do
   begin
      for i := 0 to Form1.gridConst.ColCount-1 do
      begin
         st := st + Form1.gridConst.Cells[i,j] + ' ';
      end;
      Form1.memo2.Lines.Add(st);
      st := '';
```

```
  end;
end;

procedure TForm1.SaveData(FileName : string);
begin
  memo2.Clear;
  Header;
  ObjLines;
  ConstLines;
  memo2.Lines.SaveToFile(filename);
  delete(FName,length(FName)-2,3);
  REProblem.Lines.SaveToFile(FName+'PRB');
  RCResult.Lines.SaveToFile(FName+'RST');
  Memo1.Lines.SaveToFile(FName+'ITR');
end;


procedure TForm1.Save1Click(Sender: TObject);
begin
  if not saved then
  begin
    formTitle.ShowModal;
    if savedialog1.Execute then
    begin
      FName := savedialog1.filename;
      SaveData(FName);
      caption := 'Goal Programming : '+ExtractFileName(FName);
      saved := true;
    end;
  end
  else SaveData(FName);
end;

{_____________________End Save Data_____________________}


{Begin Open Data}

function OpenFile : boolean;
begin
 with Form1 do
 begin
   if opendialog1.Execute then
   begin
     FName    := Opendialog1.FileName;
     memo2.Lines.LoadFromFile(FName);
     Caption  := 'Goal Programming : '+extractFileName(FName);
     OpenFile := true;
   end
   else OpenFile := false;
 end;
end;

procedure InitHeader;
var st1,st2  : string;
begin
 with Form1 do
 begin
   st1 := memo2.Lines.values['Title'];
   st2 := memo2.Lines.Values['Creator'];
```

```
    StatusBar.SimpleText := st1+' created by '+st2;
    formTitle.Edit1.Text := st1;
    formTitle.Edit2.Text := st2;
    SpinVar.Value      := strtoint(memo2.Lines.Values['Var']);
    SpinObj.Value      := strtoint(memo2.Lines.Values['Obj']);
    SpinConst.Value    := strtoint(memo2.Lines.Values['Con']);
    GridObj.ColCount   := SpinVar.Value+5;
    GridObj.RowCount   := SpinObj.Value+1;
    GridConst.ColCount := SpinVar.Value+3;
    if SpinConst.Value < 1 then
      GridConst.RowCount := SpinConst.Value+2
    else
      GridConst.RowCount := SpinConst.Value+1;
  end;
end;

procedure FillGrid(beginning,the_end : integer; grid : TStringgrid);
var f,g,i,j : integer;
    st      : string;
begin
  st := '';
  f  := 0;
  g  := 0;
  for i := beginning to the_end do
  begin
    for j := 0 to length(form1.memo2.Lines[i]) do
    begin
      case form1.memo2.Lines[i][j] of
      'A'..'Z',
      'a'..'z',
      '/'      : begin
                   if (f=0) or (g=0) or (f=grid.ColCount-1) or
                   (f=grid.ColCount-2)then
                      st := st + form1.memo2.Lines[i][j];
                 end;

      '0'..'9',
      '-',',','  : st := st + form1.memo2.Lines[i][j];

      '<','>',
      '='      : st := st + form1.memo2.Lines[i][j];

      ' '      : begin
                   grid.Cells[f,g] := st;
                   st := '';
                   inc(f);
                 end;
      end;
    end;
    f := 0;
    inc(g);
  end;
end;

procedure TForm1.Open1Click(Sender: TObject);
begin
  if OpenFile then
  begin
    pagecontrol1.ActivePage := Problem;
    memo1.Clear;
```

```
    rcresult.Clear;
    InitHeader;
    saved := true;
    FillGrid(5,SpinObj.Value+5,GridObj);
    FillGrid(SpinObj.Value+6,SpinObj.Value+SpinConst.Value+6,GridConst);
    delete(FName,length(FName)-2,3);
    if fileexists(FName+'PRB') then
       REProblem.Lines.LoadFromFile(FName+'PRB')
    else REProblem.Clear;
    if fileexists(FName+'ITR') then
       Memo1.Lines.LoadFromFile(FName+'ITR')
    else Memo1.Clear;
    if fileexists(FName+'RST') then
       RCResult.Lines.LoadFromFile(FName+'RST')
    else RCResult.Clear;
  end;
end;


{End Open Data}

procedure TForm1.New1Click(Sender: TObject);
begin
   saved := false;
   ButtonNewClick(Sender);
end;


procedure TForm1.ButtonSaveClick(Sender: TObject);
begin
   Save1Click(Sender);
end;

procedure TForm1.ButtonOpenClick(Sender: TObject);
begin
   Open1Click(sender);
end;

procedure TForm1.SaveAsClick(Sender: TObject);
begin
  saved := false;
  save1click(sender);
end;

procedure TForm1.FormCreate(Sender: TObject);
var i : integer;
begin
  useWeight    := false;
  useOverValue := false;
  spinObjchange(sender);
  spinConstChange(sender);
  spinVarChange(sender);
  for i := 1 to 3 do
     fs[i] := false;
  toolbutton3.Down := false;
  OpenDialog1.InitialDir := ExtractFilePath(ParamStr(0));
  SaveDialog1.InitialDir := OpenDialog1.InitialDir;
  HelpFile := OpenDialog1.InitialDir + '\GPHelp.hlp';
end;
```

```
{Menterjemahkan isi sel-sel grid menjadi bilangan dan memasukkannya ke
 dalam array}
procedure ParsingObj;
var i,j,c : integer;
begin
  with Form1 do
  begin
    for j := 1 to GridObj.RowCount - 1 do
    begin
      holderJ   := j;
      for i := 1 to GridObj.ColCount - 5 do
      begin
        if GridObj.cells[i,j] = '' then
          val('0',ObjArray[i,j],c)
        else
          val(GridObj.cells[i,j],ObjArray[i,j],c);
        holderI := i;
      end;
      EQ[holderJ] := GridObj.cells[holderI+1,holderJ];
      if GridObj.cells[holderI+2,holderJ] = '' then
        val('0',RHS[j],c)
      else
        val(GridObj.cells[holderI+2,holderJ],RHS[j],c);
      minmax[j] := GridObj.Cells[holderI+3,holderJ];
      PTy[j]    := GridObj.Cells[holderI+4,holderJ];
    end;
  end;
end;

procedure ParsingOverValue;
begin
 { }
end;

procedure ParsingConst;
var i, j, temp, c : integer;
begin
  with form1 do
  begin
    for j := 1 to GridConst.RowCount - 1 do
    begin
      for i := 1 to GridConst.ColCount - 3 do
      begin
        if GridConst.cells[i,j] = '' then
          val('0',ConstArray[i,j],c)
        else
          val(GridConst.cells[i,j],ConstArray[i,j],c);
        temp := i;
      end;
      EQi[j] := GridConst.cells[temp,j];
      if GridConst.cells[temp+2,j] = '' then
        val('0',RHSConst[j],c)
      else
        val(GridConst.cells[temp+2,j],RHSConst[j],c);
    end;
  end;
end;

procedure TForm1.Parsing;
begin
```

```
  ParsingObj;
  ParsingOverValue;
  ParsingConst;
end;

{Membangun Initial Tableau untuk iterasi}

procedure CopyObj;
var i,j : integer;
begin
  for j := 1 to Form1.SpinObj.Value do
  begin
    prosesArray[1,j] := RHS[j];
    for i := 2 to Form1.SpinVar.Value+1 do
       prosesArray[i,j]    := ObjArray[i-1,j];
    LeftLabel[j]           := 'd'+ inttostr(dev)+'-';
    UpperLabel[devIndex-1]  := 'd'+inttostr(dev)+'-';
    prosesArray[devIndex,j] := 1;
    inc(devIndex);
    UpperLabel[devIndex-1]  := 'd'+inttostr(dev)+'+';
    prosesArray[devIndex,j] := -1;
    inc(devIndex);
    inc(dev);
    holderJ                := j;
  end;
end;


procedure CopyConst;
var i,j,temp : integer;
begin
  temp := holderJ;
  with Form1 do
  begin
    if SpinConst.Value > 0 then
    begin
      for j := temp+1 to temp+SpinConst.Value do
      begin
        for i := 2 to SpinVar.Value+1 do
          prosesArray[i,j] := ConstArray[i-1,j-temp];

        prosesArray[1,j]       := RHSConst[j-temp];
        LeftLabel[j]           := 's'+inttostr(slack);
        UpperLabel[devIndex-1]  := 's'+inttostr(slack);
        prosesArray[devIndex,j] := 1;
        inc(slack);
        inc(devIndex);
        holderJ := j;
      end;
    end;
  end;
end;

function ExtractPTy(st : string) : integer;
var int, c : integer;
begin
  delete(st,1,1);
  val(st,int,c);
  result := int;
end;
```

```
function sign(st : string) : string;
begin
  delete(st,1,length(st)-1);
  result := st;
end;

procedure CopyOverValue;
var i,j,temp,temp2,c : integer;
    found            : boolean;
begin
   for j := 1 to FormOverValue.GridOverValue.RowCount-1 do
   begin
      i := 1;
      found := false;
      while (not found) and (i <= devIndex) do
      begin
        if FormOverValue.GridOverValue.Cells[0,j] = UpperLabel[i] then
        begin
          found := true;
          temp  := i;
        end;
        inc(i);
      end;
      inc(holderJ);
      val(FormOverValue.GridOverValue.Cells[1,j],prosesArray[1,holderJ],c);
      temp2 := ExtractPTy(FormOverValue.gridOverValue.cells[2,j]);
      val(FormOverValue.GridOverValue.cells[1,j],fucker[temp2],c);
      prosesArray[temp+1,holderJ] := 1;
      UpperLabel[devIndex-1] := 'd'+inttostr(dev)+'-';
      prosesArray[devIndex,holderJ] := 1;
      LeftLabel[holderJ] := UpperLabel[devIndex-1];
      inc(devIndex);
      UpperLabel[devIndex-1] := 'd'+inttostr(dev)+'+';
      prosesArray[devIndex,holderJ] := -1;
      inc(devIndex);
      inc(dev);
   end;
end;


procedure ApplyWeight(arg : integer);
var j : integer;
begin
   if arg = 1 then
      for j := 1 to FormWeight.GridWeight.RowCount-1 do
          val(FormWeight.GridWeight.cells[1,j],weights[j],cnt)
   else
      for j := 1 to Form1.GridObj.RowCount-1 do
          weights[j] := 1;
end;



procedure ClearTempGoal;
var i,j : integer;
begin
   for j := 1 to 10 do
      for i := 1 to 30 do
          TempGoal[i,j] := 0;
```

```
end;

procedure ClearObjective;
var i : integer;
begin
  for i := 1 to 10 do
     Objective[i] := '';
end;

procedure ClearFucker;
var i : integer;
begin
  for i := 1 to 20 do
    fucker[i] := 0;
end;

procedure CopyGoal;
var i,j,temp : integer;
    st       : char;
begin
  ClearTempGoal;
  ClearObjective;
  ClearFucker;
  for j := 1 to Form1.GridObj.RowCount-1 do
  begin
     temp := ExtractPTy(PTy[j]);
     if minmax[j] = 'min' then st := '+'
     else st := '-';
     objective[temp] := objective[temp] + 'd'+inttostr(j)+st+' ';
     if temp > GoalPointer then GoalPointer := temp;
     for i := 1 to devIndex-1 do
     begin
       if sign(UpperLabel[i]) = '+' then TempGoal[i,temp] := 0
       else
         TempGoal[i,temp] := prosesArray[i,j]*weights[j]+TempGoal[i,temp];
     end;
     fucker[temp] := fucker[temp] + prosesArray[1,j]*weights[j];
     if minmax[j] = 'min' then
     for i := 1 to form1.spinvar.value+1 do
        TempGoal[i,temp] := 0;
  end;
  if useOverValue then
  for j := Form1.GridObj.RowCount to Form1.GridObj.RowCount+
           FormOverValue.GridOverValue.RowCount do
  begin
     temp := ExtractPTy(FormOverValue.GridOverValue.Cells[2,j-(Form1.GridObj.RowCount-1)]);
     fucker[temp] := fucker[temp] + prosesArray[1,j];
     if temp > GoalPointer then GoalPointer := temp;
     for i := 2 to devIndex-1 do
     begin
       if sign(UpperLabel[i+1]) = '+' then
            TempGoal[i,temp] := -1*prosesArray[i,j]+TempGoal[i,temp]
       else TempGoal[i,temp] := prosesArray[i,j]+TempGoal[i,temp];
     end;
     TempGoal[1,temp] := 0;
  end;
end;

procedure ApplyGoal;
var i, j, cnt : integer;
```

```
begin
  cnt := GoalPointer;
  for j := 1+holderJ to GoalPointer+holderJ do
  begin
    for i := 1 to devIndex-1 do
    begin
      prosesArray[i,j]:= TempGoal[i,cnt];
      LeftLabel[j] := 'P'+inttostr(cnt);
    end;
    dec(cnt);
  end;
end;

procedure MakeGoals;
begin
 GoalPointer := 1;
 if useWeight then
   ApplyWeight(1)
 else ApplyWeight(0);
 CopyGoal;
 ApplyGoal;
end;

procedure ClearProsesArray;
var i,j : integer;
begin
  for j := 1 to 20 do
    for i := 1 to 30 do
      Prosesarray[i,j] := 0;
end;

procedure TForm1.BuildInitialTable;
var i : integer;
begin
  dev     := 1;
  slack   := 1;
  ClearProsesArray;
  devIndex := SpinVar.Value+2;
  for i := 1 to SpinVar.Value do
    UpperLabel[i] := 'x'+inttostr(i);
  CopyObj;
  if useOverValue then
    CopyOverValue;
  CopyConst;
  MakeGoals;
end;

procedure MakeLine(arg : integer; var argst : string);
var i : integer;
begin
  argst := '';
  for i := 1 to arg do
    argst := argst+garis;
end;


{Menuliskan Initial Table dari hasil parsing}
procedure TForm1.ShowTable;
var i,j    : integer;
   st,s,Ln : string;
```

```
begin
  st := #9+'RHS';
  for i := 1 to devIndex-1 do
     st := st +#9+ UpperLabel[i];
  makeLine(devIndex,Ln);
  memo1.Lines.Add(Ln);
  memo1.Lines.Add(st);
  memo1.Lines.Add(Ln);
  for j := 1 to holderJ do
  begin
    st := LeftLabel[j]+#9;
    for i := 1 to devIndex-1 do
    begin
      str(ProsesArray[i,j]:5:2,s);
      st := st + s + #9;
    end;
    memo1.Lines.Add(st);
  end;

  memo1.Lines.Add(Ln);
  for j := holderJ+1 to holderJ+GoalPointer do
  begin
    st := LeftLabel[j]+#9;
    for i := 1 to devIndex-1 do
    begin
      str(ProsesArray[i,j]:5:2,s);
      st := st + s + #9;
    end;
    memo1.Lines.Add(st);
  end;
  memo1.Lines.Add(Ln);
end;


{_____________________________________________________________________________}

procedure TForm1.Modified_Simplex;
var temp, ratio              : real;
    ColumnKey, RowKey,
    i,j,k,l,m ,ct,h          : integer;
    StillGoOn, fail          : boolean;
    Key                      : real;

begin
  ct := 0;
  k := 0;                  //berperan sebagai k dalam buku
  h := holderJ+GoalPointer;
  repeat
    while (prosesArray[1,h-k] <= 0) and (k < GoalPointer) do inc(k);
    if prosesArray[1,h-k] > 0 then      //.....7
    begin
       Key       := 0;
       fail      := false;
       StillGoOn  := false;
       fail      := false;
       for i := 2 to devIndex-1 do          //pada P[k] cari nilai terbesar-->ColumnKey
       begin
         if (prosesArray[i,h-k] > Key) or ((prosesArray[i,h-k] = Key)and fail) then   //P[k] =
prosesArray[i,rowholder-ct]
```

```
            begin
                //fail := false;
                for j  := 0 to k-1 do

begin
                    if prosesArray[i,h-j] < 0 then
                    fail := true;
                end;
                if fail then StillGoOn   := false
                else StillGoOn   := true;
              Key        := prosesArray[i,h-k];
              ColumnKey   := i;
            end;
          end;


{mencari z yang paling negatif ==> mencari Column Key}

            ratio := maxint;
            for j := 1 to holderJ do
            begin
               if prosesArray[ColumnKey,j] <> 0 then
               begin
                  temp := prosesArray[1,j]/prosesArray[ColumnKey,j];
                  if temp > 0 then
                  if ratio > temp then
                  begin
                     ratio := temp;
                     RowKey := j;
                  end
               end;
               //else fail := true;
            end;
//.....................................9



{mencari Row Key}
            Leftlabel[RowKey] := Upperlabel[ColumnKey-1];
            temp := prosesArray[ColumnKey, RowKey];
            for i := 1 to devIndex do
              if temp <> 0 then
                prosesArray[i, RowKey] := prosesArray[i, RowKey]/temp;
  //.....................................10
                                    {mengoperasikan Row Key}

            for l := 1 to h do
               if l <> RowKey then
               begin
                  temp := prosesArray[ColumnKey,l];
                  for m := 1 to devIndex do
                     prosesArray[m,l] := prosesArray[m,l] -
                           (temp*prosesArray[m,RowKey]);
               end;
  //.....................................................................11
```

```
{mengoperasikan Row selain Row Key}

      if prosesArray[1,h-k] <= 0 then inc(k);
      inc(ct);
      form1.memo1.Lines.Add('');
      form1.memo1.Lines.Add('');
      form1.memo1.Lines.Add('Table ['+inttostr(ct)+'] :');
      ShowTable;
      if ct = 50 then
      begin
        showmessage('Masalah tidak terselesaikan atau loop terlalu banyak');
      end;
    end;

  until (StillGoOn = false) or (k >= GoalPointer) or (ct>=50);

end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if MessageDlg('Thank you for using this program.'+#13+#13+
    'Do you really want to quit now?',
    mtConfirmation, [mbYes, mbNo], 0) = mrNo then  CanClose := false
  else if not saved then
  if MessageDlg('Your project is not saved yet.'+#13+#13+
    'Save it now?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    Save1Click(sender);
end;

procedure TForm1.Content1Click(Sender: TObject);
begin
  Application.HelpCommand(HELP_CONTENTS, 0);
end;

procedure TForm1.ToolButton1Click(Sender: TObject);
begin
 toolbutton1.Down := true;
 toolbutton2.Down := false;
 toolbutton5.Down := false;
 REProblem.Paragraph.Alignment := taLeftJustify;
end;

procedure TForm1.ToolButton2Click(Sender: TObject);
begin
 toolbutton1.Down := false;
 toolbutton2.Down := true;
 toolbutton5.Down := false;
 REProblem.Paragraph.Alignment := taCenter;
end;

procedure TForm1.ToolButton5Click(Sender: TObject);
begin
 toolbutton1.Down := false;
 toolbutton2.Down := false;
 toolbutton5.Down := true;
 REProblem.Paragraph.Alignment := taRightJustify;
end;
```

```
procedure TForm1.ToolButton6Click(Sender: TObject);
begin
  if colordialog1.Execute then
     REProblem.SelAttributes.Color := colordialog1.Color;
end;

procedure TForm1.REProblemSelectionChange(Sender: TObject);
begin
    BoldButton.Down := fsBold in REProblem.SelAttributes.Style;
    ItalicButton.Down := fsItalic in REProblem.SelAttributes.Style;
    UnderlineButton.Down := fsUnderline in REProblem.SelAttributes.Style;
    case Ord(REProblem.Paragraph.Alignment) of
      0: toolbutton1Click(sender);
      1: toolbutton2Click(sender);
      2: toolbutton5Click(sender);
    end;
end;

procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  REProblem.SelAttributes.Size := spinEdit1.Value;
  REProblem.SetFocus;
end;

procedure TForm1.BoldButtonClick(Sender: TObject);
begin
if BoldButton.Down then
     REProblem.SelAttributes.Style := REProblem.SelAttributes.Style + [fsBold]
else   REProblem.SelAttributes.Style := REProblem.SelAttributes.Style - [fsBold];
  REProblem.SetFocus;
end;

procedure TForm1.ItalicButtonClick(Sender: TObject);
begin
  if ItalicButton.Down then
      REProblem.SelAttributes.Style := REProblem.SelAttributes.Style + [fsItalic]
  else   REProblem.SelAttributes.Style := REProblem.SelAttributes.Style - [fsItalic];
  REProblem.SetFocus;
end;

procedure TForm1.UnderlineButtonClick(Sender: TObject);
begin
  if UnderlineButton.Down then
    REProblem.SelAttributes.Style := REProblem.SelAttributes.Style + [fsUnderline]
  else   REProblem.SelAttributes.Style := REProblem.SelAttributes.Style - [fsUnderline];
  REProblem.SetFocus;
end;

procedure TForm1.BulletClick(Sender: TObject);
begin
   if Bullet.Down then
      REProblem.Paragraph.Numbering := nsBullet
   else REProblem.Paragraph.Numbering := nsNone;
end;

procedure TForm1.LeftIndentClick(Sender: TObject);
begin
   REProblem.Paragraph.FirstIndent :=  REProblem.Paragraph.FirstIndent + 5;
//   REProblem.Paragraph.LeftIndent :=  REProblem.Paragraph.LeftIndent + 5;
end;
```

```
procedure TForm1.ReduceLeftIndentClick(Sender: TObject);
begin
   REProblem.Paragraph.FirstIndent :=  REProblem.Paragraph.FirstIndent - 5;
//   REProblem.Paragraph.LeftIndent :=  REProblem.Paragraph.LeftIndent - 5;
end;

procedure TForm1.REProblemChange(Sender: TObject);
begin
  saved := false;
end;

procedure TForm1.Print1Click(Sender: TObject);
begin
  if PrintSetup.Execute then
  if Printdialog.Execute then
  begin
    if PageControl1.ActivePage = Problem then
       REProblem.Print(formtitle.Edit1.Text+' Created by '+formtitle.Edit2.Text);
    if PageControl1.ActivePage = Iterasi then
       Memo1.Print(formtitle.Edit1.Text+' Created by '+formtitle.Edit2.Text);
    if PageControl1.ActivePage = Result then
       RCResult.Print(formtitle.Edit1.Text+' Created by '+formtitle.Edit2.Text);
    if PageControl1.ActivePage = Input then
       Messagedlg('Select another sheet : Problem, Iteration, or Result',mtWarning,[mbOK],0);
  end;
end;

end.
```