PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK OPTIMASI MASALAH MIXED INTEGER PROGRAMMING DENGAN ALGORITMA NON LINIER

Disusun oleh :

FAJAR ASTUTI HERMAWATI

NRP. 2691.100.051

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
1997

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK OPTIMASI MASALAH MIXED INTEGER PROGRAMMING DENGAN ALGORITMA NON LINIER

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan

Untuk Memperoleh Gelar Sarjana Teknik Informatika

Pada

Jurusan Teknik Informatika

Fakultas Teknologi Industri

Institut Teknologi Sepuluh Nopember

S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing

Dr. Ir. SUPENO DJANALI, M.Sc.

NIP. 130 368 610

S U R A B A Y A

Maret, 1997

ABSTRAK

ABSTRAK

Banyak metode yang digunakan untuk menyelesaikan masalah pemrograman linier bilangan bulat. Tetapi selama ini metode-metode tersebut menggunakan pemrograman linier.

Perangkat lunak yang dirancang kali ini digunakan untuk menyelesaikan permasalahan pemrograman bilangan bulat, khusus untuk masalah bilangan bulat biner, yaitu permasalahan dengan kendala variabel bernilai 0 atau 1. Permasalahan ini banyak digunakan untuk menyelesaikan permasalahan pengambilan keputusan ya-atau-tidak, yang aplikasinya banyak dibutuhkan pada dunia industri.

Permasalahan bilangan bulat nol-satu yang diberikan dapat didekati dengan suatu persamaan kuadratik atau non linier. Permasalahan non linier ini diselesaikan dengan salah satu algoritma non linier dengan kendala, yaitu pemrograman Separabel. Prosedur Branch and Bound digabungkan dengan teknik Pemeriksaan digunakan untuk mencari nilai optimum bilangan bulatnya.

PRAKATA

PRAKATA

Ucapan syukur dan terima kasih yang sedalam-dalamnya kami panjatkan kepada Allah Yang Maha Kuasa, yang oleh karena perkenan-Nya sajalah kami dapat menyelesaikan tugas akhir yang berjudul :

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK UNTUK OPTIMASI MASALAH MIXED INTEGER PROGRAMMING DENGAN ALGORITMA NON-LINIER

Harapan kami tugas akhir ini bukan hanya sekedar sebagai persyaratan untuk meraih gelar sarjana Informatika, tetapi kiranya dapat merupakan sumbangan yang bermanfaat bagi kemajuan dan perkembangan pemrograman komputer khususnya, dan bagi para pembaca umumnya.

Akhir kata kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan sumbangan pikiran maupun tenaga dalam membantu menyelesaikan tugas akhir ini.

Surabaya, Januari 1997

Penyusun

UCAPAN TERIMA KASIH

UCAPAN TERIMA KASIH

Dengan terselesaikannya Tugas Akhir ini, maka penulis ingin menyatakan terima kasih kepada :

1. Allah Yang Maha Kuasa atas rahmat dan karunianya, sehingga Tugas Akhir ini dapat terselesaikan dengan baik.
2. Bapak Dr. Ir. Supeno Djanali, M.Sc selaku Dosen Pembimbing atas segala bimbingan, nasihat dan petunjuk mulai dari pembuatan usulan sampai selesainya pembuatan Tugas Akhir ini.
3. Bapak Dr. Drs. Ir. Riyanarto Sarno, M.Sc selaku Dosen Wali atas perhatian dan bimbingan selama tahap perkuliahan.
4. Kedua orang tua dan seluruh anggota keluarga yang telah memberikan perhatian, dorongan dan doa. Tak lupa pula kepada mas Thufael yang tiada putus-putusnya memberikan semangat, dorongan dan dengan rela siap membantu apabila diperlukan.
5. Rekan-rekan mahasiswa Teknik Informatika, khususnya rekan angkatan '91, yang banyak memberi masukan dalam perancangan perangkat lunak, yaitu:
   a. Probo Jatmiko yang banyak membantu mengoreksi kesalahan dalam pembuatan perangkat lunak.
   b. Budi Irmawati yang membantu menyelesaikan tampilan akhir perangkat lunak.

c. Trias Purwitasari yang telah menyediakan tempat dan fasilitas dalam menyelesaikan pembuatan perangkat lunak.

d. Mirna Utari yang telah meminjamkan buku dan membantu pembuatan tampilan perangkat lunak.

e. Primadi yang telah bersedia menjadi moderator pada saat seminar tugas akhir.

f. Lilik, Henny dan rekan-rekan lain yang banyak memberi dukungan dan bantuan ketika ada masalah dengan hard disk penulis.

DAFTAR ISI

DAFTAR ISI

DAFTAR TABEL

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR GAMBAR

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG

Dengan semakin bertambah kompleksnya suatu permasalahan, dibutuhkan suatu sarana penunjang untuk mengatasi dan menyelesaikan masalah yang ada. Pemecahan masalah di bidang industri, membutuhkan penanganan khusus. Selama ini telah banyak disusun metode-metode dan algoritma-algoritma yang digunakan untuk menyelesaikan permasalahan-permasalahan tersebut, terutama permasalahan alokasi sumber daya. Dan antara metode satu dengan yang lainnya saling mendukung dan saling melengkapi. Salah satu pendekatan yang banyak digunakan adalah pemrograman linier.

Pemakaian pemrograman linier dalam memecahkan masalah terutama di bidang industri, banyak menolong. Sebab dengan adanya pemrograman linier, sumber-sumber daya yang ada dapat dialokasikan secara optimum.

Aplikasi pemrograman linier adalah untuk mengalokasikan sumber daya yang terbatas agar memperoleh hasil yang optimum. Misalnya alokasi fasilitas industri mulai dari industri ringan sampai berat, maupun alokasi sumber daya manusia seperti penempatan pekerja yang tepat agar produksi dapat berjalan lebih optimal.

Secara umum, pemrograman linier dapat digambarkan melalui suatu persamaan matematis, dengan satu atau lebih persamaan kendala. Fungsi matematis dalam pemrograman linier harus berupa fungsi linier.

Dalam pemrograman linier, kita diperbolehkan memakai variabel yang tidak bulat atau non integer. Tetapi dalam masalah tertentu, diperlukan suatu variabel yang bulat atau integer, terutama masalah yang menyangkut suatu obyek tertentu. Misalnya, menentukan banyaknya kendaraan yang harus digunakan. Dalam hal ini tidak mungkin kita menggunakan bilangan pecahan, seperti 1,5 buah kendaraan. Untuk itu kita memerlukan suatu pemrograman linier integer yang variabel-variabel fungsi tujuan maupun fungsi kendalanya harus bulat.

Pemrograman linier integer pada intinya berkaitan dengan program-program linier dimana beberapa atau semua variabel memiliki nilai-nilai bulat (integer) atau diskrit.

Walaupun beberapa algoritma telah dikembangkan untuk integer programming, tidak satu pun metode ini yang sepenuhnya andal dari sudut pandang perhitungan, terutama sementara jumlah variabel integer meningkat. Tidak seperti pada linier programming, dimana masalah-masalah dengan ribuan variabel dan ribuan batasan dapat dipecahkan dalam sejumlah waktu yang wajar, pengalaman perhitungan integer programming, setelah dikembangkan selama lebih dari 30 tahun, tetap sulit dilakukan.

Kesulitan perhitungan dengan algoritma integer programming yang tersedia telah mengarahkan para pengguna untuk mencari cara-cara lain untuk memecahkan masalah yang bersangkutan. Salah satu pendekatan seperti ini adalah

memecahkan model tersebut sebagai suatu persamaan linier programming yang kontinyu dan lalu membulatkan pemecahan optimum ke nilai integer terdekat yang layak. Tetapi, tidak ada jaminan dalam kasus ini bahwa pemecahan yang dibulatkan itu akan memenuhi batasan-batasan. Ini selalu berlaku jika integer programming semula memiliki satu batasan atau lebih yang berbentuk persamaan. Dari teori pemrograman linier, sebuah pemecahan yang dibulatkan dalam kasus ini tidak dapat layak, karena pemecahan itu menyiratkan bahwa basis yang sama (dengan semua variabel nondasar berada di tingkat nol) dapat menghasilkan dua pemecahan yang berbeda.

Ketidaklayakan yang tercipta dari pembulatan dapat ditolelir, karena, pada umumnya, parameter (yang diestimasi) dari masalah-masalah tersebut tidak pasti. Tetapi terdapat batasan persamaan tertentu dalam masalah initeger di mana parameter-parameternya pasti. Masalah dimana semua nilai variabelnya bernilai 0 atau 1 adalah salah satu contohnya. Dalam kondisi seperti ini, pembulatan tidak dapat dipergunakan, dan algoritma yang pasti menjadi sangat penting.

Untuk menekankan lebih lanjut ketidak-layakan pembulatan pada umumnya, walaupun variabel-variabel integer umumnya dipandang mewakili sejumlah diskrit objek (misalnya, mesin, orang, kapal), jenis-jenis lain mewakili kuantifikasi dengan arti tertentu. Jadi keputusan untuk menandai atau tidak menandai sebuah proyek dapat diwakili dengan variabel biner $x = 0$ jika proyek tersebut ditolak atau $x = 1$ jika diterima. Dalam kasus ini, tidak masuk akal untuk menangani nilai-nilai pecahan dari x, dan penggunaan pembulatan sebagai aproksimasi secara logis tidak dapat diterima.

Algoritma berikut ini digunakan untuk menekan ketidak layakan pembulatan pada umumnya. Karena dalam algoritma ini, akan digunakan suatu pendekatan non linier untuk mencari penyelesaian suboptimal yang bulat dengan nilai semua variabelnya sama dengan 0 atau 1. Teknik selanjutnya digunakan gabungan antara prosedur Branch and Bound yang sudah biasa dipakai dengan suatu teknik pemeriksaan.

1.2. PERUMUSAN MASALAH[1]

Masalah-masalah pemrograman integer secara luas dapat digunakan untuk mencari suatu keputusan "ya" atau "tidak", yang dapat digambarkan secara matematis sebagai 0 atau 1.

Jika kita mempunyai suatu masalah linier mixed integer programming P sebagai berikut :

Minimumkan : $z_Q = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

Kendala : $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$

.

.

.

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m$

$0 \leq x_i \leq 1$

$x_i \in \{0,1\}$

[1] A. Ciriani Tito and C. Leachman, Robert, Optimization in Industry : Mathematical Programming and Modelling Techniques in Practice, John Wiley & Sons, Inc., 1993, hal. 202-203

Tujuan dari algoritma berikut adalah untuk menyelesaikan masalah suboptimal P*, dimana P* adalah suatu masalah untuk mencari suatu penyelesaian bulat yang fisibel.

Untuk lebih memudahkan pencarian, kita akan menentukan suatu persamaan yang menghubungkan masalah suboptimal P* dengan suatu persamaan pemrograman matematis non linier.

Pada masalah suboptimal P*, kita tentukan bahwa setiap variabel x_i (i=1,2,..., n) adalah merupakan variabel 0-1. Sehingga kita dapat menuliskan suatu persamaan kendala yang baru :

$$0 \leq x_i \leq 1$$

Dan dari persamaan z di atas dapat diperoleh suatu fungsi persamaan kuadrat yang non-konvek untuk diminimumkan, yaitu:

$$z_Q = \Sigma (x_i - x_i^{\,2})$$

Dari persamaan yang baru diatas, dapat dilihat bahwa agar nilai z_Q non-negatif, dalam hal ini bernilai nol, jika dan hanya jika $x_i = 0$ atau $x_i = 1$.

Ini berarti bahwa setiap penyelesaian fisibel dari masalah suboptimal P* dapat diperoleh jika penyelesaian optimal keseluruhan dari masalah kuadratik di atas dapat dicapai, dengan kondisi :

$$z_Q = 0$$

Jadi setiap perhitungan selesai kita harus memeriksa besarnya nilai z_Q. Jika nilai $z_Q = 0$, maka semua variabelnya bernilai 0 atau 1. Dengan demikian kita memperoleh salah satu jawaban bilangan integer, yang selanjutnya kita periksa apakah jawaban itu merupakan jawab layak keseluruhan.

Formulasi permasalahan P* dapat ditulis sebagai berikut :

Minimumkan : $z_Q = c_1(x_1-x_1^2) + c_2(x_2-x_2^2) + \ldots + c_n(x_n-x_n^2)$

Kendala : $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$

.

.

.

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m$

$0 \leq x_i \leq 1$

$x_i \in \{0,1\}$

Dengan formulasi di atas dapat diperoleh suatu penyelesaian yang fisibel untuk P*, yaitu suatu jawaban dengan variabel yang bernilai bulat. Setelah itu, untuk memeriksa apakah jawaban itu optimal untuk masalah P, dicari nilai paling optimum diantara semua jawaban yang diperoleh setelah semua node pada prosedur Branch and Bound ter-*fathomed.*

1.3. TUJUAN STUDI

Dalam pembangunan ataupun perbaikan suatu sistem umumnya membutuhkan berbagai macam disiplin ilmu yang berkaitan dengan sistem yang akan dibangun atau diperbaiki. Disiplin ilmu yang luas dalam bidang komputer memungkinkan berbagai bidang permasalahan dapat didekati termasuk dalam permasalahan pengambilan keputusan yang sederhana.

Masalah pengalokasian sumber daya yang dapat memberi hasil optimal dapat diselesaikan dengan algoritma ini. Hal ini mendorong penulis mencoba

merancang suatu perangkat lunak dengan menggunakan algoritma tersebut. Adapun tujuan dari studi ini adalah

1. Mencari cara baru yang lebih mudah dan cepat untuk menyelesaikan masalah pemrograman bilangan bulat (*integer programming*), khususnya masalah pengambilan keputusan dua variabel ya atau tidak.
2. Merancang suatu perangkat lunak yang dapat membantu menyelesaikan masalah pemrograman linier bilangan bulat dengan pendekatan algoritma non-linier.
3. Mempelajari hubungan antara masalah pemrograman linier integer dengan formulasi pemrograman kuadratik non-linier.

1.4.MANFAAT STUDI

Masalah pengambilan keputusan dalam suatu bidang industri memegang peran cukup penting. Pengambilan keputusan di sini dimaksudkan adalah keputusan dua variabel yaitu ya atau tidak atau 0 atau 1. Maka perancangan perangkat lunak ini diharapkan dapat bermanfaat dalam :

1. Membantu menyelesaikan masalah pengambilan keputusan pada umumnya.
2. Membantu menyelesaikan masalah pemrograman integer dengan variabel 0-1 yang umum terjadi pada bidang industri, yang merupakan bagian dari masalah pemrograman bilangan bulat campuran (*Mixed Integer Programming*).
3. Menjadi bahan pustaka untuk masalah pemrograman integer yang terus berkembang dari waktu ke waktu.

1.5.PEMBATASAN MASALAH

Dalam tugas akhir ini, permasalahannya dibatasi hanya untuk permasalahan pemrograman bilangan bulat biner 0 dan 1.

Model permasalahan yang dipilih dalam tugas akhir ini adalah masalah maksimasi ataupun minimisasi. Tetapi lebih menekankan pada permasalahan minimisasi seperti yang telah dijelaskan pada bab perumusan masalah.

1.6. ASUMSI

Dalam algortima ini digunakan asumsi-asumsi sebagai berikut:

- Dengan asumsi bahwa suatu permasalahan pemrograman linier integer untuk variabel 0-1 dapat didekati dengan pendekatan non linier, seperti yang tertulis dalam pustaka Optimization in Industry, "Non Linier Optimization Algorithm for Mixed Integer Programming Problems: Applications and Results, halaman 203.
- Diasumsikan bahwa dalam pemrograman separabel kita dapat hanya menggunakan salah satu diantara pasangan separabel dalam proses, sedangkan yang lain diset ke batas atas atau batas bawahnya. Misalnya kita mempunyai variabel x, dimana $x = x_1 + x_2 + \ldots + x_k$. Kita dapat memilih salah satu x_i (i=1,2,. . ., k) untuk masuk dalam proses. Sisanya, sebut saja variabel x_r, diset pada batas atas jika $r < i$, dan pada batas bawah jika $r > i$.

1.7. METODOLOGI PENELITIAN

Metodologi yang digunakan dalam menyusun dan menyelesaikan tugas akhir ini adalah :

1. Merumuskan permasalahan
2. Mengumpulkan referensi yang menunjang.
3. Merancang sistem, menentukan batasan masalah, dan menentukan metode yang akan digunakan.
4. Melakukan uji coba dan mengevaluasi hasilnya.
5. Mengembangkan dan mengoptimumkan perangkat lunak yang telah dibuat.
6. Menyusun laporan.

1.8. SISTEMATIKA PENULISAN

Sistematika penulisan laporan tugas akhir ini diuraikan sebagai berikut :

BAB I PENDAHULUAN

Berisi latar belakang pokok permasalahan, tujuan studi penelitian, manfaat studi, pembatasan masalah, asumsi-asumsi yang digunakan, metodologi penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Berisi uraian teori-teori yang relevan dengan penelitian yang dilakukan. Teori-teori yang diuraikan disini meliputi uaraian tentang metode simplek, masalah upper-bounded variabel, masalah pemrograman integer termasuk masalah pemrograman bilangan biner

dan algoritma Branch and Bound, masalah pemrograman non-linier terutama pemrograman separabel, dan algoritma Q yang digunakan untuk menyelesaikan masalah mixed integer programming dalam tugas akhir ini, serta penjelasan algoritma gabungan antara teknik pemeriksaan dengan teknik Branch dang Bound.

BAB III PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Berisi penjelasan mengenai perancangan dan pembuatan perangkat lunak, yang berupa uraian struktur data dan prosedur-prosedur yang digunakan dalam pemrograman.

BAB IV ANALISA HASIL

Bab ini menganalisa hasil dari proses dengan menggunakan perangkat lunak yang telah dirancang, serta waktu yang diperlukan dalam proses.

BAB V PENUTUP

Bab ini adalah bagian terakhir laporan yang berisi kesimpulan pembahasan yang telah dilakukan pada bab-bab sebelumnya, serta saran yang mungkin dapat lebih menyempurnakan perangkat lunak yang telah dibuat.

BAB II

TINJAUAN PUSTAKA

BAB II

TINJAUAN PUSTAKA

2.1. PEMROGRAMAN LINIER

Pemrograman linier[2] adalah suatu cara untuk menyelesaikan persoalan pengalokasian sumber-sumber yang terbatas di antara beberapa aktifitas yang bersaing, dengan cara yang terbaik yang mungkin dilakukan.

Pemrograman linier menggunakan model matematika untuk menggambarkan suatu permasalahan. Di dalam model ini semua fungsi matematikanya adalah linier.

2.1.1. MODEL PEMROGRAMAN LINIER[3]

Misalkan saja kita memiliki m sumber daya yang masing-masing dalam jumlah yang terbatas, yang akan dibagi-bagikan dalam n aktifitas.

	Penggunaan sumber daya per unit aktifitas				Banyaknya
	1	2	. . .	n	sumber daya
1	a_{11}	a_{12}	. . .	a_{1n}	b_1
2	a_{21}	a_{22}	. . .	a_{2n}	b_2
. . .	. . .	. . .	. . .	. . .	. . .
m	a_{m1}	a_{m2}	. . .	a_{mn}	b_m

Tabel 2.1
Tabel Sumber Daya dan Penggunaaannya

---

[2] Dimyati, Tjutju Tarliah dan Dimyati, Ahmad, Operations Research : Model-Model Pengambilan Keputusan, Sinar Baru, Bandung, 1992, halaman 17.

[3] Ibid, halaman 24.

Jika laba per unit aktifitas adalah c_i, laba dari aktifitas adalah x_i dan laba keseluruhan adalah z, maka model matematika dari permasalahan tersebut :

Maksimum (atau minimum) : $z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

Kendala fungsi : $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_n$

Kendala nonnegatif : $x_1, x_2, \ldots, x_n \geq 0$

Model ini kita sebut sebagai bentuk standar dari pemrograman linier.

2.1.2. ASUMSI DALAM PEMROGRAMAN LINIER[4]

Ada beberapa asumsi yang diberikan untuk pemrograman linier ini. Asumsi-asumsi itu adalah sebagai berikut :

1. **Proporsionalitas** (*Proportionality*)

   Bila tiap aktifitas ditinjau secara terpisah, maka naiknya z adalah proporsional terhadap naiknya x_k (c_kx_k) dan naiknya penggunaan sumber daya i juga proporsional terhadap naiknya x_k (a_kx_k).

2. **Additifitas** (*Additivity*)

   Asumsi ini menjamin bahwa total harga sama dengan jumlah total harga masing-masing aktifitas. Demikian juga untuk total pemakaian sumber daya adalah jumlah pemakaian sumber daya masing-masing aktifitas

[4] Bazaraa, Mokhtar S. and Jarvis, John J., Linier Programming and Network Flows, John Wiley & Sons Inc., 1977, halaman 3-4

3. **Divisibelitas** (*Divisibility*)

Semua variabel keputusan (*decision variable*) dapat dibagi dalam beberapa bilangan pecahan atau dengan kata lain semua variabel dapat memiliki harga berapapun asalkan bernilai real non negatif.

2.2. METODE SIMPLEKS

2.2.1. DEFINISI[5]

Metode Simpleks merupakan prosedur aljabar yang bersifat iteratif, yang bergerak selangkah demi selangkah, dimulai dari suatu titik ekstrem pada daerah fisibel menuju ke titik ekstrem yang optimum.

Dalam Metode Simpleks terdapat beberapa istilah, yaitu

- **Solusi basis** (*Basis Solution*)

  Solusi basis untuk AX = b adalah solusi di mana terdapat paling sedikit (n-m) variabel bernilai nol. Untuk mendapatkan solusi basis dari AX = b maka sebanyak (n-m) variabel harus bernilai nol. Variabel-variabel yang bernilai nol tersebut dinamakan **variabel nonbasis** (*Non Basis Variable*). Sedangkan variabel lainnya sebanyak m buah yang tidak bernilai nol dinamakan variabel basis (*Basis Variable*).

- **Solusi basis fisibel** (*Basis Feasible Solution*)

  Solusi basis fisibel adalah suatu solusi basis yang seluruh variabelnya bernilai nonnegatif.

---

[5] Dimyati, Tjutju Tarliah dan Dimyati, Ahmad. Operations Research: Model-model Pengambilan Keputusan, Sinar Baru Bandung, 1992, halaman 48-49.

- **Solusi fisibel titik ekstrem**

  Solusi fisibel titik ektrem adalah suatu solusi fisibel yang tidak terletak pada suatu segmen garis yang menghubungkan dua solusi fisibel lainnya.

- **Variabel yang masuk basis** (*Entering Variable*)

  Variabel yang masuk basis adalah suatu variabel non basis yang dipilih menurut bentuk permasalahannya, maksimasi atau minimisasi, yang akan masuk ke dalam variabel basis.

- **Variabel yang meninggalkan basis** (*Leaving Variable*)

  Variabel yang meninggalkan basis ditentukan dengan memilih nilai positif terkecil dari nilai perbandingan tiap baris antara nilai ruas kanan dari variabel kendala dengan konstanta dari kolom yang dipilih untuk masuk menjadi basis.

2.2.2. ALGORITMA METODE SIMPLEKS

Langkah-langkah algoritma simplex adalah sebagai berikut:[6]

Step 0 : Konversikan formulasi persoalan ke dalam bentuk standar. Dengan menggunakan bentuk standar, ditentukan solusi basis fisibel (BFS) awal dengan jumlah variabel nonbasis n-m pada level 0.

Step 1: Menentukan **variabel yang masuk basis** (*entering variable)* pada baris ke-0. Untuk persoalan:

Maksimasi: Pilih nilai negatif terbesar. Jika semua nilainya

---

[6] Dimyati, Tjutju Tarliah dan Dimyati, Ahmad. Operations Research: Model-model Pengambilan Keputusan, Sinar Baru Bandung, 1992, halaman 53-54 dan halaman 58.

nonnegatif, BFS optimal tercapai, ke *step 4*.

Minimisasi: Pilih nilai positif terbesar. Jika sudah nonpositif semua, ke *step 4*, BFS optimalnya sudah tercapai.

Step 2: Menentukan **variabel yang meninggalkan basis** (*leaving variable*).

Step 3: Melakukan operasi baris elementer dengan metode Eliminasi Gauss-Jordan untuk mencari solusi basis fisibel yang baru.

Step 4: Selesai.

Baris yang terhubung ke leaving variable dinamakan persamaan pivot (*pivot equation*). Sedangkan perpotongan antara kolom dari variabel yang masuk basis (*entering column*) dengan *pivot equation* disebut elemen pivot (*pivot element*). Operasi baris elementer akan melakukan perubahan pada basis dengan menggunakan dua tipe perhitungan[7]

1. Tipe 1 (untuk persamaan pivot)

   persamaan pivot baru = persamaan pivot lama / elemen pivot.

2. Tipe 2 (untuk semua persamaan, termasuk z)

   persamaan baru = persamaan lama - (koefisien *entering column*) X

   (persamaan pivot baru)

[7] Taha, Hamdi A., Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 73.

2.2.3. BENTUK STANDAR PEMROGRAMAN LINIER[8]

Dalam menyelesaikan persoalan pemrograman linier dengan menggunakan metode Simpleks, bentuk pemrograman linier yang digunakan haruslah dalam bentuk standar, yaitu bentuk formulasi yang memiliki sifat-sifat sebagai berikut :

1. Fungsi tujuannya dapat berupa maksimasi atau minimasi.
2. Seluruh variabel keputusan haruslah berupa variabel nonnegatif.
3. Seluruh kendala harus berbentuk persamaan dengan ruas kanan nonnegatif.

Secara matematis bentuk standar pemrograman linier dapat dinyatakan sebagai berikut :

Maksimum atau Minimum : $z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

Kendala :

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$

$$x_j \geq 0 , j = 1,2, \ldots , n$$

Dalam persoalan yang sebenarnya, ternyata banyak diketemukan bentuk dari pemrograman linier yang tidak sesuai dengan bentuk standar di atas. Bentuk pemrograman linier yang tidak standar memiliki sifat-sifat sebagai berikut :

a. Paling sedikit satu kendala berbentuk ketidaksamaan.
b. Paling sedikit satu ruas kanan bernilai negatif.

---

[8] Herniwan, Dani, Makalah Tugas Akhir : Perancangan dan Pembuatan Perangkat Lunak Mixed-Strategy Two-Person Zero-Sum Game dengan Metode Simpleks, Jurusan Teknik Komputer, Institut Teknologi Sepuluh Nopember, Surabaya, 1996, halaman 3-4

c. Paling sedikit satu variabel keputusan berupa variabel yang tidak terbatas dalam tanda (*unrestricted in sign*).

Untuk merubah suatu bentuk pemrograman linier yang tidak standar ke dalam bentuk standar dapat dilakukan dengan cara-cara sebagai berikut:

- Tidak standar dalam kendala

  Kendala yang berbentuk ketidaksamaan bertanda kurang atau sama dengan (≤), dapat dijadikan bentuk standar, dengan menambahkan *slack variable* (variabel tambahan) pada ruas kiri dari persamaan tersebut.

  Kendala yang berbentuk ketidaksamaan bertanda lebih besar atau sama dengan (≥), dapat dijadikan bentuk standar, dengan menambahkan *artificial variable* (variabel tiruan) ke ruas kiri persamaan dan mengurangkannya dengan variabel tambahan. Khusus untuk masalah ini, dapat diselesaikan dengan metode Big-M (*penalty method*) atau teknik dua fase.

- Ruas kanan dari kendala yang tidak standar atau bernilai negatif dapat dirubah ke bentuk nonnegatif dengan mengalikan tiap suku dalam persamaan dengan -1, baik itu di ruas kanan maupun di ruas kiri.

- Variabel keputusan yang tidak terbatas dalam tanda dapat dijadikan bentuk yang standar, yaitu variabel keputusan yang nonnegatif, dengan cara mensubstitusikan variabel keputusan tersebut menjadi dua buah variabel yang nonnegatif pada seluruh kendala dan fungsi tujuannya. Substitusi tersebut berbentuk : $x_i = x_i' - x_i''$, dimana x_i berupa variabel keputusan yang tidak terbatas dalam tanda dan $x_i', x_i'' \geq 0$.

2.3. MASALAH VARIABEL DENGAN BATAS ATAS[9]

Algoritma untuk menyelesaikan masalah variabel yang dibatasi (*bounded variable*) merupakan suatu perluasan dari metode Simpleks dasar. Kenyataannya banyak variabel dalam suatu penyelesaian pemrograman linier seringkali dibatasi dengan suatu batas bawah dan atau batas atas. Sebagai contoh, kapasitas mesin pabrik yang sering dibatasi maksimum 720 jam tiap bulannya. Hal ini termasuk suatu batas atas.

Dalam algoritma pemrograman linier konvensional, suatu kendala harus ditunjukkan sebagai suatu bagian persamaan dalam model matematikanya. Dalam algoritma *bounded variable*, informasi ini dapat dinyatakan ke dalam suatu kode solusi pemrograman linier tanpa harus menempatkan baris atau persamaan khusus dalam matriks. Pada waktu yang sama, jika ada kendala penggunaan mesin sedikitnya 400 jam tiap bulannya, dapat diperlakukan dengan cara yang sama, sebagai batas bawah. Kalau kita amati, sepertinya algoritma *bounded variabel* memperbolehkan pemakaian, dan kenyataannya, spesifikasi dari batas atas dan bawah pada variabel-variabel tertentu tidak digabungkan dalam matrik tetapi digunakan secara langsung dalam perhitungan.

Hasilnya, dengan kode-kode *bounded variable*, matriks penyelesaiannya menjadi lebih kecil, dan tentu saja jumlah input-output yang ditransfer selama penyelesaian pemrograman linier menjadi lebih kecil. Jumlah iterasi untuk mencapai

---

[9] Taha, Hamdi A., Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 260-263.

optimal solution tidak terlalu berpengaruh, dan peningkatan efisiensi penyelesaian hanya untuk mereduksi waktu input-output yang disediakan.

Secara umum masalah *Upper-Bounded* dapat ditulis sebagai berikut :

Maksimum atau minimum : $z = \sum c_i x_i$

Kendala : $(A,I)\ x_i = b$

$x_i + x_i' = u$

$x_i, x_i' \geq 0$

Diasumsikan b adalah suatu vektor bilangan nonnegatif, sehingga inisialisasi awal dari masalah tersebut layak (*feasible*).

Adanya tambahan kendala : $x_i + x_i' = u$ pada tabel Simpleks berpengaruh pada kondisi feasibilitas dari metode Simpleks tersebut. Sedangkan kondisi optimalitasnya masih sama dengan metode Simpleks biasa.

Ide dasar untuk memodifikasi kondisi fisibelitas tersebut adalah bahwa sebuah variabel menjadi infisibel jika variabel tersebut menjadi negatif atau melebihi batas atasnya (*upper-bound*). Untuk mengatasi kondisi nonnegatifitas dapat dipergunakan metode Simpleks biasa. Sedangkan untuk mengatasi kondisi upper-bound digunakan cara khusus yaitu dengan mengijinkan suatu variabel basis menjadi nonbasis pada batas atasnya. Pada metode Simpleks biasa, semua variabel nonbasis sama dengan nol. Demikian juga jika variabel nonbasis dipilih, nilai yang dimasukkan tidak boleh melebihi batas atasnya. Jadi untuk mengatasi kondisi fisibelitas yang baru, ada dua point yang harus dipertimbangkan, yaitu:

1. Kendala nonnegatif dan batas atas untuk entering variabel.
2. Kendala nonnegatif dan batas atas untuk variabel-variabel basis yang mungkin dipengaruhi oleh *entering variable*.

Untuk mengembangkan ide tersebut secara matematis, dapat digunakan masalah linier programming tanpa batas atas. Dengan perincian sebagai berikut :

x_j : variabel nonbasis pada level nol yang dipilih sebagai entering variabel.

$(x_B)_i = (x_B^*)_i$: variabel penyelesaian basis x_B ke i .

Sehingga jika x_j dimasukkan dalam penyelesaian, persamaan akan menjadi:

$$(x_B)_i = (x_B^*)_i - \alpha_{ij}x_j$$

Dimana α_{ij} : elemen ke i dari $\alpha_{ij} = B^{-1}P_j$ dan P_j adalah vektor (A,I) yang terhubung ke x_j . x_j tetap fisibel jika :

$$0 \le x_j \le u_j$$

Dan $(x_B)_i$ menjadi fisibel jika

$$0 \le (x_B^*)_i - \alpha_{ij}x_j \le u_j , \quad i = 1,2, ..., m$$

Kondisi nonnegatif dapat dilihat dari:

$$(x_B)_i = (x_B^*)_i - \alpha_{ij}x_j \ge 0$$

Dari persamaan tersebut dapat dilihat bahwa semakin besar nilai α_{ij} akan menyebabkan nilai $(x_B)_i$ menjadi negatif. Sehingga untuk memaksimumkan nilainya, maka dipilih nilai α_{ij} yang paling minimum.

$$\theta_1 = \min i \{ (x_B^*)_i / \alpha_{ij} , \alpha_{ij} > 0 \}$$

Proses untuk formulasi diatas sama seperti pada metode Simpleks biasa.

Untuk mencegah $(x_B)i$ nilainya akan melebihi batas atasnya, maka :

$$(x_B)_i = (x_B^*)_i + (-\alpha_{ij})\, x_j \leq (u_B)_i$$

dimana u_B = vektor batas atas untuk basis.

Juga untuk mencegah hasil yang negatif dan memaksimumkan nilai x_j dipilih kondisi:

$$\theta_2 = \min i\{(u_B)_i - (x_B^*)_i \;/\; -\alpha_{ij}\,,\ \alpha_{ij} < 0\}$$

Dari persamaan-persamaan diatas dipilih salah satu untuk mendapatkan nilai x_j yang maksimum. Maka:

$$\theta = \min \{ \theta_1, \theta_2, u_j \}$$

Setelah dipilih nilai θ, maka dilakukan operasi untuk merubah solusi basis awalnya. Ada 3 aturan untuk merubah solusi basis yaitu :

a. Jika $\theta = \theta_1$, sama seperti aturan pada metode Simpleks, yaitu $(x_B)_r$ menjadi nonbasis dan x_j masuk. Kemudian diselesaikan dengan Operasi Baris Elementer Gauss-Jordan.

b. Jika $\theta = \theta_2$, $(x_B)_r$ keluar dan x_j masuk; $(x_B)_r$ menjadi nonbasis pada batas atasnya, sehingga harus disubstisusikan sebagai berikut : $(x_B)_r = u_r - (x_B)_r'$.

c. Jika $\theta = u_j$, x_j disubstitusikan pada batas atasnya sebagai : $u_j - x_j'$ tetapi masih tetap menjadi variabel nonbasis.

2.4. PEMROGRAMAN LINIER BILANGAN BULAT[10]

Salah satu asumsi dasar dari prosedur pemrograman linier yang konvensional adalah bahwa semua variabelnya kontinyu, atau dengan kata lain, variabel bilangan pecahan juga diperbolehkan dalam perhitungan. Dalam situasi industri nyata, bilangan-bilangan pecahan tersebut masih memungkinkan dan berguna. Sebagai contoh, masih mungkin menggunakan mesin P untuk memproduksi A dalam 1,3 hari. Dalam contoh-contoh lain juga masih memungkinkan menggunakan bilangan pecahan.

Tetapi tidak menutup kemungkinan kita harus menggunakan suatu bilangan yang bulat. Misalnya, dalam penggunaan suatu objek seperti mesin, mobil dan sebagainya. Tidak mungkin kita menggunakan kendaraan sebanyak 1,2. Masalah ini biasa disebut masalah pemrograman linier bilangan bulat campuran (*Mixed Integer Linier Programming*), karena variabel-variabel yang digunakan bernilai bulat dan kontinyu.

Masalah pemrograman bilangan bulat kelihatannya relatif mudah untuk diselesaikan. Karena model matematisnya sama dengan pemrograman linier, jadi kita bisa menganggap bahwa masalah pemrograman bilangan bulat sebagai masalah pemrograman linier biasa. Dengan demikian akan dapat dengan cepat dicari penyelesaiannya dengan menggunakan metode Simpleks. Penyelesaian yang didapat dari pemrograman linier, apabila variabel yang dikehendaki bernilai bilangan bulat belum mencapai nilai bilangan bulat, maka nilai pecahan yang

[10] Pala Windu, I Ketut, Makalah Tugas Akhir : Perancangan dan Pembuatan Perangkat Lunak Mixed Integer Programming dengan Metode Branch and Bound, Jurusan Teknik Komputer, ITS, 1995, halaman 11-16.

didapat dibulatkan. Cara ini dalam beberapa kasus mungkin cukup memuaskan, tetapi secara garis besar masih banyak kekurangannya. Ada dua masalah yang paling mungkin terjadi bila dilakukan pembulatan terhadap penyelesaian pemrograman linier.

- Hasil yang didapat dari pembulatan tidak layak terhadap masalah awal.

  Untuk kasus ini dapat digambarkan dengan contoh berikut. Misalkan setelah dilakukan iterasi dengan metode Simpleks didapat penyelesaian layak yang optimal untuk suatu masalah pemrograman linier sebagai berikut :

  $x_1 = 6.5$ dan $x_2 = 10$,

  dimana masalah pemrograman bilangan bulat-nya mempunyai dua fungsi kendala sebagai berikut :

  $-x_1 + x_2 \leq 3.5$

  $x_1 + x_2 \leq 16.5$

  x_1, x_2 adalah bilangan bulat (bilangan bulat)

  Jika x_1 kita bulatkan ke bawah, menjadi $x_1 = 6$, maka apabila nilai tersebut kita substitusikan ke dalam fungsi kendalanya untuk mengetahui layak atau tidak layaknya suatu hasil akan diperoleh nilai sebagai berikut :

  $-6 + 10 \leq 3.5 \Rightarrow 4 \leq 3.5$ (tidak layak)

  $6 + 10 \leq 16.5 \Rightarrow 16 \leq 16.5$ (layak)

  Karena kendala pertama tidak layak maka, jawaban untuk $x_1 = 6$ menjadi tidak layak. Selanjutnya kita bulatkan ke atas menjadi $x_1 = 7$. Dengan cara yang sama kita peroleh nilai fungsi kendalanya menjadi :

$-7 + 10 \leq 3.5 \Rightarrow 3 \leq 3.5$ (layak)

$7 + 10 \leq 16.5 \Rightarrow 16 \leq 16.5$ (tidak layak)

Ini berarti untuk jawaban $x_1 = 7$ juga merupakan jawaban yang tidak layak.

- Hasil yang didapat dari pembulatan sangat jauh dari harga optimal yang sebenarnya.

Kita ambil contoh sebagai berikut :

Maksimum : $z = x_1 + 5x_2$,

Kendala : $x_1 + 10x_2 \leq 20$

$x_1 \leq 2$,

$x_1, \quad x_2 \geq 0$ dan bilangan bulat.

Nilai optimal dari masalah di atas adalah $z = 11$ dengan $x_1 = 2$ dan $x_2 = 9/5$. Jika x_2 dibulatkan ke bawah menjadi $x_2 = 1$, maka harga optimal untuk masalah tersebut adalah $z = 7$. Padahal hasil optimal yang sebenarnya adalah $z = 10$ dengan $x_1 = 0$ dan $x_2 = 2$.

Apabila kita kembali pada masalah pemrograman linier, maka dengan metode Simpleks penyelesaian optimalnya dapat dicari dengan sangat efisien. Tetapi pemrograman bilangan bulat menyebabkan hilangnya sejumlah kemungkinan penyelesaian yang layak untuk masalah pemrograman linier, sebagai akibat adanya kendala beberapa variabelnya bernilai bilangan bulat. Ini menyebabkan masalah pemrograman linier tidak layak lagi terhadap masalah pemrograman bilangan bulat, sehingga metode Simpleks tidak dapat menentukan secara langsung harga optimal dari masalah pemrograman bilangan bulat.

Dengan memperhatikan uraian di atas dapat ditarik kesimpulan, walaupun masalah pemrograman bilangan bulat mempunyai jawab layak yang lebih sedikit dari pemrograman linier, akan tetapi jawab layak yang berhingga ini akan menjadi tak berhingga banyaknya apabila algoritma yang digunakan kurang baik.

Sebagian besar algoritma yang dapat menyelesaikan pemrograman bilangan bulat, pada umumnya sedapat mungkin menyertakan metode Simpleks didalamnya, dengan menghubungkan ketentuan-ketentuan yang ada pada pemrograman bilangan bulat dengan ketentuan yang bersesuaian dengan pemrograman linier, atau untuk menyelesaikan masalah pemrograman bilangan bulat pertama kali diselesaikan dengan metode Simpleks dengan menghilangkan kendala variabel bilangan bulat. Pada dasarnya penyelesaian masalah pemrograman bilangan bulat masih berada dalam ruang jawab layak pemrograman linier. Untuk suatu masalah pemrograman bilangan bulat yang diberikan, masalah pemrograman linier yang bersesuaian ini disebut *LP-relaxation.* Pada uraian berikutnya diperlihatkan bagaimana serangkaian *LP-relaxation* untuk bagian-bagian pemrograman bilangan bulat dapat digunakan untuk menyelesaikan masalah tersebut secara keseluruhan dengan efisien.

Suatu hal khusus dapat terjadi pada saat kita menyelesaikan *LP-relaxation* dari masalah pemrograman bilangan bulat yang sudah memenuhi kendala bilangan bulat. Penyelesaian layak optimal yang didapat untuk *LP-relaxation* adalah juga merupakan penyelesaian layak dari masalah pemrograman bilangan bulat. Untuk itu menyelesaikan masalah pemrograman bilangan bulat, biasanya dimulai dengan menyelesaikan *LP-relaxation* dengan metode Simpleks untuk memeriksa bilamana

kondisi ini terjadi. Jadi metode Simpleks memegang peranan penting dalam menyelesaikan masalah pemrograman bilangan bulat.

Ada dua faktor yang menyebabkan kesulitan perhitungan pada masalah pemrograman bilangan bulat, yaitu :

- Banyaknya variabel bilangan bulat.
- Struktur permasalahan pemrograman bilangan bulat.

Untuk masalah pemrograman bilangan bulat campuran (MIP) yang mempunyai jumlah variabel bilangan bulat yang kecil, akan lebih mudah dicari penyelesaiannya, walaupun jumlah variabel secara keseluruhan besar.

2.4.1. METODE BRANCH AND BOUND UNTUK PEMROGRAMAN BILANGAN BULAT[11]

Pemrograman linier bilangan bulat campuran (*Mixed Integer Linier Programming*) adalah kasus khusus dari masalah pemrograman linier yang mempunyai variabel bilangan bulat dan kontinyu. Bentuk umum dari permasalahan ini adalah sebagai berikut :

---

[11] Pala Windu, I Ketut, Makalah Tugas Akhir : Perancangan dan Pembuatan Perangkat Lunak Mixed Integer Programming dengan Metode Branch and Bound, Jurusan Teknik Komputer, ITS, 1995, halaman 20-27

Maksimum atau minimum : $z = \sum c_j x_j$

Kendala : $\sum a_{ij} x_j \leq b_i$ untuk $i = 1,2, \ldots, m$

$x_j \geq 0$ untuk $j = 1,2, \ldots, n$

x_j integer untuk $j = 1,2, \ldots, I (I \leq n)$

dari formulasi di atas dapat dilihat bahwa :

1. Jika I = n, maka permasalahan tersebut menjadi permasalahan pemrograman bilangan bulat murni.
2. Jika I < n, maka permasalahan tersebut menjadi permasalahan *Mixed Integer Programming* .

Metode Branch and Bound yang dipergunakan untuk menyelesaikan masalah pemrograman linier bilangan bulat campuran berdasarkan pada algoritma yang dikembangkan A.H. Land dan A.G. Doig.

Kendala utama dalam menyelesaikan masalah tersebut adalah adanya kendala variabel yang berupa bilangan bulat. Untuk itu kendala tersebut terlebih dahulu harus dihilangkan dari permasalahan, sehingga dapat dianggap sebagai permasalahan pemrograman linier biasa. Menghilangkan kendala yang memerlukan suatu variabel bernilai bulat dari masalah pemrograman bilangan bulat campuran disebut relaksasi (*relaxation*). Masalah pemrograman liniernya disebut *Linier Programming-relaxation (LP-relaxation).* Formulasi permasalahan submasalah *LP-relaxation* menjadi submasalah 1:

Maksimum atau minimumkan : $z = \sum c_j x_j$

Fungsi kendala : $\sum a_{ij} x_j \leq b_i$ untuk $i = 1,2, \ldots, m$

$x_j \geq 0$, untuk $j = 1,2, \ldots, n$

Dengan menyelesaikan submasalah di atas kita akan dapat menentukan penyelesaian optimal dari permasalahan pemrograman bilangan bulat secara keseluruhan. Ada dua hubungan antara submasalah diatas dengan masalah pemrograman bilangan bulat campuran, yaitu:

1. Hubungan batas

   Misalkan z_0^* dan z_1^* masing-masing merupakan harga fungsi tujuan untuk jawab layak optimum masalah MIP dan submasalah 1, maka berlaku :

   $$z_0^* \leq z_1^*$$

   Ini berarti harga jawab layak optimum fungsi tujuan untuk masalah MIP tidak boleh melebihi harga jawab layak optimum fungsi tujuan untuk masalah 1. Jadi z_1^* adalah batas atas untuk z_0^*. Hubungan ini disebut *bounding*.

2. Hubungan Optimalitas

   Jika jawab layak untuk submasalah 1 adalah juga layak untuk masalah MIP, maka harga jawab optimal ini juga optimal untuk masalah MIP. Yang berarti bahwa jika suatu jawab optimal layak untuk submasalah 1, dimana $z_0^* = z_1^*$, maka jawaban tersebut juga layak untuk masalah MIP. Karena z_0^* nilainya tidak boleh melebihi z_1^*. Jika jawab optimal untuk submasalah 1 tidak layak untuk masalah MIP, hal itu dikarenakan adanya variabel bilangan bulat yang masih bernilai pecahan. Maka kita harus memecah submasalah 1 menjadi beberapa

submasalah baru. Submasalah baru ini harus memenuhi ketentuan sebagai berikut :

- Setiap submasalah baru dibentuk dengan menambahkan suatu fungsi kendala baru, yang akan mengakibatkan jawab layak untuk submasalah 1 tidak layak untuk submasalah baru tersebut.
- Sekurang-kurangnya satu jawab optimal masalah MIP layak untuk sekurang-kurangnya satu dari submasalah-submasalah yang ada.

Pembentukan submasalah-submasalah baru ini disebut *branching*.

Karena variabel yang dibatasi oleh bilangan bulat mempunyai penyelesaian yang tak berhingga banyaknya, dan akan menjadi tidak efisien untuk membuat dan menganalisa banyak submasalah dengan memberikan nilai bulat tertentu pada variabel tersebut. Oleh karena itu, yang dilakukan selanjutnya adalah memecah submasalah menjadi dua submasalah baru dengan menentukan dua daerah nilai untuk variabel percabangan (*branching*). Metode percabangan ini dikembangkan pertama kali oleh R.J. Dakin.

Misalkan kita mempunyai variabel bilangan bulat x_j yang mempunyai harga jawab x_j^* yang belum bernilai bulat, maka variabel x_j kita tetapkan sebagai variabel percabangan untuk saat ini. Lalu kita tentukan dua nilai pembulatan untuk x_j, dengan daerah nilai untuk submasalah baru :

$$x_j \leq [x_j^*] \quad \text{dan} \quad x_j \geq [x_j^*] + 1$$

Setiap pertidaksamaan ini menjadi fungsi kendala tambahan untuk masing-masing submasalah baru. Karena adanya tambahan fungsi kendala ini, maka jawab optimal dari submasalah sebelumnya menjadi tidak layak untuk submasalah yang

baru, yang berarti syarat pertama dari hubungan optimalitas terpenuhi. Ini berarti jawab layak untuk masalah MIP berada pada salah satu atau kedua ruang penyelesaian submasalah baru, dan syarat kedua dari hubungan optimalitas terpenuhi.

Dari kedua submasalah baru tersebut, diselesaikan *LP-relaxation*nya, lalu dilihat kembali apakah variabel bilangan bulatnya sudah mencapai nilai yang bulat. Jika belum, kembali dilakukan percabangan. Demikian seterusnya. Yang menjadi masalah sekarang kapan percabangan harus dihentikan. Untuk itu diperlukan suatu pengujian untuk setiap penyelesaian *LP-relaxation* , yang disebut uji penghentian (*fathoming test*). Tujuan dari pengujian ini adalah untuk menentukan submasalah mana yang mengalami percabangan lebih lanjut, sehingga submasalah yang mempunyai jawab tidak layak dapat dibuang dari perhitungan.

Ada tiga pengujian yang dilakukan untuk submasalah yang baru. Tiga pengujian itu adalah sebagai berikut :

1. *Fathoming Test* 1

   Dilakukan jika nilai optimum dari submasalah baru lebih kecil atau sama dengan nilai batas bawah dari persoalan MIP. Atau $z \leq z^*$, dimana z^* merupakan batas bawah MIP untuk saaat ini.

2. *Fathoming Test* 2

   Untuk menguji apakah *LP-relaxation* dari submasalah ini mempunyai jawab yang layak. Jika tidak, maka percabangan untuk submasalah ini dihentikan (di-*fathom*).

3. *Fathoming Test* 3

Jika jawab layak untuk submasalah ini juga layak untuk masalah MIP, yang berarti semua variabel bilangan bulat sudah mempunyai nilai bulat. Jika nilai jawab layak optimal lebih baik dari batas bawah penyelesaian sebelumnya, maka batas bawah saat ini diganti dengan z dari submasalah ini sehingga $z^* = z$.

Secara garis besar, langkah-langkah metode Branch and Bound dapat dirumuskan sebagai berikut :

1. Tahap Inisialisasi

   Misalkan z^* adalah batas bawah dari persoalan MIP, maka z^* pertama kali diset dengan nilai negatif yang kecil sekali ($z^* = -\infty$).

2. Tahap Iterasi

   a. Percabangan (*Branching*)

      Algoritma yang digunakan untuk memilih submasalah yang mengalami percabangan berikutnya adalah algoritma *newest bound rule.* Algoritma ini bertujuan untuk memilih submasalah yang tidak ter-*fathom* yang dibuat dari iterasi paling akhir. Jika jumlah submasalah yang tidak ter-*fathom* lebih dari satu maka dipilih submasalah yang memiliki harga jawab layak yang paling baik. Jika ada beberapa variabel bilangan bulat yang belum mempunyai nilai yang bulat, maka dipilih variabel bilangan bulat pertama untuk dijadikan variabel percabangan (*branching*).

b. Pembatasan (*Bounding*)

Selesaikan *LP-relaxation* masing-masing submasalah baru, yang mana penyelesaian layak optimal setiap submasalah merupakan penyelesaian yang paling baik dari submasalah ini.

c. Penghentian (*Fathoming*)

Untuk setiap submasalah baru, dilakukan pengujian untuk menentukan apakah submasalah baru tersebut mempunyai jawab layak atau tidak, atau apakah jawab submasalah tersebut merupakan jawab optimal untuk masalah MIP, dan apakah submasalah ini masih mungkin mengandung nilai optimal untuk MIP yang lebih baik dari penyelesaian optimal yang sekarang.

3. Test Optimalitas

Test ini dilakukan untuk menentukan apakah masih ada submasalah yang tidak ter-*fathomed.* Jika masih ada, maka kembali dilakukan iterasi, yang menyatakan masih mungkin mencari nilai optimal yang lebih baik dari harga optimal saat ini. Jika semua submasalah sudah ter-*fathomed,* maka nilai optimal untuk saat ini adalah merupakan penyelesaian yang paling baik untuk masalah MIP. Iterasi dihentikan.

Metode ini telah banyak dipakai, dan banyak diteliti keakuratannya. Tetapi masih banyak kekurangannya.

2.4.2. ALGORITMA BRANCH-AND-BOUND UNTUK PEMROGRAMAN BILANGAN BINER[12]

Pemrograman linier biner merupakan masalah pemrograman linier bilangan bulat yang variabelnya dibatasi dua nilai, yaitu nol dan satu. Pemrograman ini banyak digunakan untuk menyelesaikan masalah pengambilan keputusan yang membutuhkan jawaban "ya"-atau-tidak. Jika batas dari variabel bilangan bulat x adalah :

$$0 \le x \le u, \text{ untuk } 2^{N-1} < u \le 2^N,$$

maka tiapnilai yang mungkin untuk x dapat dinyatakan sebagai berikut :

$$x = \sum_{i=0}^{N} 2^i y_i$$

dimana $y_i \in \{0,1\}$. Jadi variabel x dapat dinyatakan sebagai penjumlahan (N-1) variabel biner. Secara lengkap, formulasi permasalahan pemrograman biner dapat ditulis sebagai berikut :

Maksimasi atau minimasi : $Z = \sum_{j=1}^{n} c_j x_j$

Kendala : $\sum_{j=1}^{n} a_{ij} x_j \ge b_i$, untuk i = 1,2, . . . , m

$x_j = 0$ atau $x_j = 1$ untuk j = 1,2,..., n

[12] Hillier, Frederick S. and Jarvis, John J., Introduction to Operations Research , Third Edition, Holden-Day Inc., Oakland, California, 1980, halaman 725-729.

dimana $0 \leq c_1 \leq c_2 \leq \ldots \leq c_n$. Kendala untuk koefisien c_j ini tidak diharuskan karena jika $c_j \leq 0$, maka x_j dapat diganti dengan $(1 - x'_j)$, dimana $x'_j = 0$ atau 1, sehingga dalam fungsi obyektif x'_j akan mempunyai koefisien positif.

Algoritma yang digunakan untuk menyelesaikan masalah pemrograman linier bilangan biner ini, disebut algoritma *Additive* Balas[1)], yang dapat dijelaskan sebagai berikut :

- Langkah Percabangan (*Branching*)

  Algoritma ini mendefinisikan subset penyelesaian dengan nilai-nilai yang dimasukkan ke beberapa variabel, $(x_1, x_2, \ldots, x_N)$. Jika langkah ini memilih solusi parsial $(x_1, x_2, \ldots, x_N)$, untuk dipartisi, maka solusi ini dibagi ke dalam dua subset baru yang masing-masing ditambah variabel $x_{N+1} = 0$ untuk subset 1 dan $x_{N+1} = 0$ untuk subset 2. Sehingga jumlah variabel untuk subset yang baru adalah N+1.

- Langkah Pembatasan (*Bounding*)

  Untuk langkah Pembatasan, batas bawah Z_L untuk solusi parsial $(x_1, x_2, \ldots, x_N)$ adalah :

$$Z_L = \sum_{j=1}^{N} c_j x_j, \qquad \text{jika } x_N = 1 \text{ (atau jika N = 0 atau n)}$$

$$Z_L = \sum_{j=1}^{N-1} c_j x_j + c_{N+1}, \text{ jika } x_N = 0.$$

  Alasan penambahan c_{N+1} jika $x_N = 0$ adalah bahwa algoritma ini akan menghitung kendala ini hanya jika sebelumnya diketemukan bahwa

$(x_1, x_2, \ldots, x_{N-1}, x_N = 0, \ldots, x_n = 0)$ infisibel. Jika $c_N \leq c_{N+1}$, kendala untuk $x_N=1$ selalu lebih kecil atau sama dengan ($\leq$) daripada $x_N = 0$, sehingga aturan Pembatasan yang baru pertama kali selalu memilih $x_N = 1$ untuk dipartisi jika yang lainnya telah di*fathomed* .

- Langkah Penghentian (*Fathoming*)

  Langkah penghentian yang dilakukan sama dengan langkah-langkah yang telah dijelaskan pada subbab 2.4.1. sebelumnya.

  - *Fathoming Test 1*

    Jika $Z_L \geq Z_U$, dimana Z_U adalah nilai fungsi tujuan yang merupakan jawab layak untuk masalah tersebut pada saat ini.

  - *Fathoming Test 2*

    Jika jawaban dari tiap variabel tidak layak untuk masalah tersebut.

  - *Fathoming Test 3*

    Dilakukan jika ditemukan jawab layak yang paling baik untuk masalah tersebut. Dan jika $Z_L < Z_U$ maka untuk iterasi berikutnya $Z_U = Z_L$.

2.4.3.MODEL PEMROGRAMAN BILANGAN BULAT UNTUK MASALAH KEPUTUSAN YA-ATAU-TIDAK[13]

Permasalahan Keputusan Ya-atau-Tidak (*Yes-or-No Decision Problems*) merupakan salah satu masalah pemrograman bilangan bulat biner (*Binary Integer*

[13] Hillier, Frederick S. and Jarvis, John J., Introduction to Operations Research , Third Edition, Holden-Day Inc., Oakland, California, 1980, halaman 732-735.

Programming), yang merupakan masalah pemrograman bilangan bulat yang hanya mempunyai penyelesaian 0 dan 1 untuk setiap variabelnya.

Dalam kehidupan sehari-hari kita selalu dihadapkan pada pengambilan keputusan dimana hanya dua pilihan yang harus diambil, "ya" atau "tidak". Sebagai contoh, apakah kita harus menjalankan suatu proyek? Apakah kita harus melakukan investasi baru? Dan lain sebagainya.

Dengan hanya dua pilihan, keputusan-keputusan dapat diwakili dengan suatu variabel keputusan yang dibatasi hanya mempunyai dua nilai, yaitu nol dan satu. Sehingga jika kita memiliki sejumlah *i* keputusan ya-atau-tidak, yang dapat ditulis sebagai y_i, maka :

$$y_i = \begin{cases} 1, \text{ jika keputusan ke i adalah "ya"} \\ 0, \text{ jika keputusan ke i adalah "tidak"} \end{cases}$$

Dalam suatu format pemrograman linier, tiap-tiap variabel membutuhkan kendala,

$y_i \leq 1$

$y_i \leq 0$

y_i adalah bilangan bulat.

Ada beberapa situasi yang bisa terjadi pada masalah pengambilan keputusan ya-atau-tidak ini, diantaranya

- Alternatif yang saling berdiri sendiri (*mutually exclusive alternatives*)

Dalam suatu kelompok variabel keputusan ya-atau-tidak terdapat kelompok yang hanya mempunyai satu keputusan "ya". Pada kasus ini, tiap kelompok akan memerlukan salah satu kendala dibawah ini :

$$\sum y_i = 1$$

(jika tepat ada satu variabel keputusan yang mempunyai jawaban "ya")

$$\sum y_i \leq 1$$

(jika sedikitnya ada satu variabel keputusan yang mempunyai jawab "ya")

- Keputusan yang saling bergantung (*Contingent decisions*)

Dimana keputusan yang sekarang bergantung pada keputusan sebelumnya. Secara umum, keputusan k dapat dikatakan bergantung pada keputusan j jika keputusan k sama dengan "ya" hanya jika keputusan j adalah "ya". Ini terjadi jika keputusan k merupakan aksi lanjutan yang akan menjadi tidak relevan, atau tidak mungkin, jika keputusan j adalah "tidak". Dalam model matematika, dua keputusan tersebut akan membutuhkan kendala :

$$y_k \leq y_j$$

Nilai $y_j = 1$ akan memberikan kebebasan pilihan pada y_k, sedangkan untuk $y_j = 0$ menyebabkan nilai $y_k = 0$. Model matematika diatas dapat juga ditulis sebagai berikut :

$$y_k - y_j \leq 0$$

Contoh permasalahan :

- Suatu perusahaan mengambil keputusan untuk melakukan ekspansi usaha dengan membangun sebuah pabrik baru di salah satu tempat Los Angeles atau San Fransisco. Perusahaan itu juga mempertimbangkan untuk membangun sebuah gudang di kota yang dipilih untuk dibangun pabrik baru tersebut. *Net Present Value* (total keuntungan terhadap nilai uang) dari tiap-tiap alternatif,

ditunjukkan pada tabel 2.1. dibawah. Sedangkan total dana yang tersedia sebesar 25 juta dollar. Dari tabel nanti kita dapat menganalisa data dan memutuskan pabrik serta gudang akan didirikan di kota apa.

Nomor Pilihan	Pertanyaan Ya atau Tidak	Variabel Keputusan	Net Present Value	Jumlah biaya
1	Pabrik di LA	y_1	7 juta dollar	20 juta dollar
2	Pabrik di SF	y_2	5 juta dollar	15 juta dollar
3	Gudang di LA	y_3	4 juta dollar	12 juta dollar
4	Gudang di SF	y_4	3 juta dollar	10 juta dollar

Tabel 2.2

Contoh Data Pembangunan Pabrik Baru

Dari tabel di atas dapat kita lihat bahwa antara pilihan nomor 1 dan 2 terdapat hubungan *mutually exclusive alternatives* , yang berarti jika pilihan nomor 1 jawabnya "ya" maka yang nomor 2 jawabnya "tidak". Ini dapat dirumuskan dengan :

$$y_1 + y_2 = 1$$

Sedangkan pilihan nomor 3 dan 4 jawabannya bergantung pada jawaban nomor 1 dan 2. Jika jawab nomor 1 adalah "ya" maka jawab nomor 3 "ya" dan jika jawab nomor 2 "ya" maka jawab nomor 4 adalah "ya". Karena keputusan tempat untuk membangun gudang diambil setelah keputusan tempat untuk membangun pabrik ditetapkan. Disini terjadi *decisions contingent* , dan formulasi fungsi kendalanya dapat ditulis sebagai berikut :

$$y_3 - y_1 \leq 0$$

$$y_4 - y_2 \leq 0$$

Model matematikanya selengkapnya adalah :

$$\text{Maksimumkan} : \quad Z = 7y_1 + 5y_2 + 4y_3 + 3y_4$$

$$\text{dengan kendala} : \quad 20y_1 + 15y_2 + 12y_3 + 10y_4 \leq 25$$

$$y_1 + y_2 = 1$$

$$-y_1 + y_3 \leq 0$$

$$-y_2 + y_4 \leq 0$$

$$y_i = 0 \text{ atau } y_i = 1 \text{ untuk } i = 1,2,3,4.$$

Penyelesaian masalah ini dapat dilihat pada lampiran A.

2.5. PEMROGRAMAN NON LINIER[14]

Operasi-operasi pada bidang industri secara nyata terdiri dari sejumlah besar fungsi-fungsi linier. Ini merupakan kondisi dimana seuatu aktifitas atau variabel mempunyai respon linier terhadap semua kendalanya, dan ditambah lagi, mempunyai suatu faktor harga yang linier, salah satunya tidak berubah terhadap jumlah aktifitasnya. Salah satu contoh dari suatu aktifitas yang benar-benar linier ada dalam variabel transportasi, karena untuk masalah transportasi dalam prakteknya mempunyai harga yang linier, dan hubungan terhadap persamaan kendalanya benar-benar linier, karena jumlah penarikan barang-barang sama dengan jumlah aktifitas dan juga sama dengan jumlah barang yang dikirim ke tujuan.

---

[14] Taha, Hamdi A., Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 786.

Banyak contoh lain yang juga berhubungan dengan kondisi linier dalam masalah-masalah industri. Tetapi ada juga situasi dimana variabel-variabel atau aktifitasnya tidak menghasilkan respon yang linier atau tidak mempunyai harga tiap level aktifitas yang linier. Salah satu contoh yang jelas adalah banyaknya diskon atau potongan harga, dimana pada sejumlah unit pertama dibuat harganya lebih tinggi dari unit-unit yang berikutnya. Fenomena ini ada pada banyak operasi pembelian, dimana harga per unitnya mengalami penurunan bersamaan dengan meningkatnya jumlah barang yang dibeli.

Pemrograman non linier dibagi dalam dua katagori yaitu :

1. Algoritma nonlinier dengan kendala (*constrained nonlinier algorithms*)
2. Algoritma nonlinier tanpa kendala (*unconstrained nonlinier algorithms*)

Yang akan kita bahas lebih lanjut adalah algoritma nonlinier dengan kendala. Suatu permasalahan non linier dengan kendala dapat dinyatakan dalam model matematik seperti berikut ini :

Maksimum (minimum) : $z = f(X)$

Kendala : $g(X) \leq 0$

2.5.1. KONDISI KUHN-TUCKER[15]

Kondisi Kuhn-Tucker merupakan suatu kondisi untuk mengidentifikasikan titik stasioner (*stationary points*) masalah nonlinier dengan kendala yang berupa

---

[15] Taha, Hamdi A., Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 771.

fungsi ketidaksamaan, dan berdasarkan metode Lagrangean. Secara umum, kondisi ini adalah sebagai berikut :

Masalah Optimasi	Kondisi yang dibutuhkan	
	Fungsi Obyektif	Ruang penyelesaian
Maksimasi	Konkaf	Himpunan konvek
Minimisasi	Konvek	Himpunan konvek

Tabel 2.3
Kondisi Kuhn-Tucker

2.5.2. PEMROGRAMAN SEPARABEL[16]

Pemrograman separabel (*Separable programming*) adalah merupakan suatu prosedur khusus yang dimaksudkan untuk menangani kondisi yang tidak linier yang sering terjadi dalam bidang industri. Dalam separable programming, bagian fungsi yang nonlinier dibagi dalam beberapa segmen, dimana tiap-tiap segmen berupa fungsi linier. Fungsi-fungsi linier ini yang diselesaikan untuk mendapatkan penyelesaian optimalnya. Dengan cara ini, beberapa fungsi nonlinier yang nonkonvek dapat ditangani dengan lebih efisien.

Suatu fungsi $f(x_1, x_2, \ldots, x_n)$ dikatakan separabel jika fungsi tersebut dapat diekspresikan dalam bentuk penjumlahan n fungsi variabel tunggal $f_1(x_1), f_2(x_2), \ldots, f_n(x_n)$, sebagai berikut :

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

Sebagai contoh, fungsi linier berikut adalah separabel :

$$h(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \ldots + a_nx_n$$

[16] Taha, Hamdi A., Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 786-789.

dimana a adalah konstanta. Sedangkan fungsi berikut bukan separabel :

$$h(x_1, x_2, \ldots, x_n) = x_1^2 + x_1 \sin(x_2 + x_3) + x_2 e^{x3}$$

Beberapa fungsi nonlinier tidak langsung separabel, tetapi harus dibuat sedemikian hingga menjadi separabel dengan menggunakan substitusi-substitusi. Sebagai contoh, untuk masalah:

Maksimumkan : $z = x_1 x_2$

Misalnya disubstitusikan $y = x_1 x_2$, maka $\ln y = \ln x_1 + \ln x_2$ dan masalah diatas menjadi separabel dengan:

Maksimumkan : $z = y$

Kendala : $\ln y = \ln x_1 + \ln x_2$

Substitusi-substitusi tersebut diasumsikan bahwa x_1 dan x_2 merupakan variabel-variabel yang positif; jika tidak, fungsi logaritma tersebut tidak bisa didefinisikan sebagai fungsi yang separabel.

Jika x_1 dan x_2 diasumsikan bernilai nol, dapat ditangai dengan cara berikut. Misalnya δ_1 dan δ_2 adalah konstanta positif dan didefinisikan sebagai :

$$w_1 = x_1 + \delta_1$$

$$w_2 = x_2 + \delta_2$$

$$x_1 x_2 = w_1 w_2 - \delta_2 w_1 - \delta_1 w_2 + \delta_1 \delta_2$$

Jika $y = w_1 w_2$; maka masalah diatas menjadi :

Maksimumkan : $z = y - \delta_2 w_1 - \delta_1 w_2 + \delta_1 \delta_2$

kendala : $\ln y = \ln w_1 + \ln w_2$

Secara umum, jika kita mempunyai permasalahan kuadratik seperti berikut :

$$\text{Maksimum (minimum)} : z = \sum_{i=1}^{n} f_i(x_i)$$

$$\text{Kendala} : \sum_{i=1}^{n} g_i^j(x_i) \leq b_j , \quad j = 1,2, \ldots , m$$

Maka dapat kita dekati dengan model sebagai berikut :

$$\text{Maksimum (minimum)} : z = \sum_{i=1}^{n} \sum_{k=1}^{Ki} f_i(a_i^k)\, t_i^k$$

$$\text{Kendala} : \sum_{i=1}^{n} \sum_{k=1}^{Ki} g_i^j(a_i^k)\, t_i^k \leq b_j , \; j = 1, 2, \ldots, m$$

$$0 \leq t_i^1 \leq y_i^1$$

$$0 \leq t_i^k \leq y_i^{k-1} + y_i^k \;, \; k = 2, 3, \ldots, K_i - 1$$

$$0 \leq t_i^{Ki} \leq y_i^{Ki-1}$$

$$\sum_{k=1}^{Ki-1} y_i^k = 1$$

$$\sum_{k=1}^{Ki} t_i^k = 1$$

$$y_i^k = 0 \text{ atau } 1$$

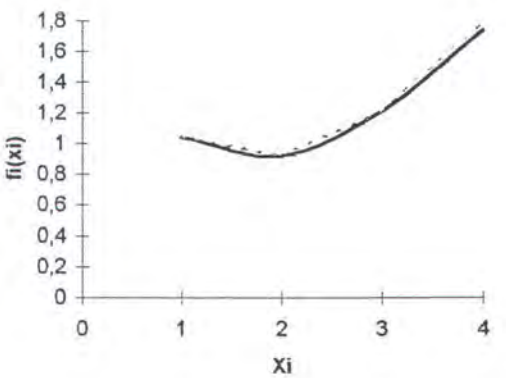
$$k = 1, 2, \ldots, Ki; \; i = 1, 2, \ldots, n$$

Dimana Ki adalah jumlah pembagian segmen dari tiap variabel x_i (*breaking point*).

2.5.3. PEMROGRAMAN SEPARABEL KONVEK[17]

Pemrograman Separabel Konvek (*Separable Convex Programming*) merupakan kasus khusus dari pemrograman separabel, yaitu jika fungsi $g_i^j(x_i)$ konvek sehingga daerah penyelesaiannya adalah suatu himpunan konvek. Sedangkan fungsi tujuannya, $f_i(x_i)$, konvek untuk masalah minimisasi, dan konkaf untuk masalah maksimasi.

Misalnya kita mempunyai masalah minimisasi dengan fungsi $f_i(x_i)$ seperti pada gambar berikut :



Gambar 2.2

Contoh Fungsi Kuadrat $f_i(x_i)$

k adalah titik pembagian (*breaking point*) dari fungsi $f_i(x_i)$, yaitu titik $x_i = 1$, $x_i = 2$ dan $x_i = 3$, sebut saja $a_{0i} = 1$, $a_{1i} = 2$, $a_{2i} = 3$, untuk $k = 1, 2, \ldots, K_i$. Jika x_{ki} menggambarkan pertambahan satu (*increment*) dari variabel x_i pada jarak $(a_{k-1,i}, a_{ki})$ dan ρ_{ki} menunjukkan kemiringan (*slope*) dari segmen garis dengan jarak yang sama, maka :

---

[17] Taha, Hamdi A , Operations Research : An Introduction, Fourth Edition, MacMillan Publishing Company, 1982, halaman 791-793

$$f_i(x_i) \cong \sum_{k=1}^{Ki} (\rho_{ki} x_{ki}) + f_i(a_{0i})$$

$$x_i = \sum_{k=1}^{Ki} x_{ki}$$

$$0 \le x_{ki} \le a_{ki} - a_{k-1,i}\ ,\quad k = 1, 2, \ldots, Ki.$$

Maka secara umum jika kita mempunyai permasalahan non linier :

Maksimum (minimum) : $z = \sum f_i x_i$

Kendala : $\sum g_i^j x_i$

akan dapat kita dekati dengan model :

Minimumkan : $z = \sum_{i=1}^{n} \left(\sum_{k=1}^{Ki} \rho^j_{ki} x_{ki} + f_i(a_{0I}) \right)$
(Maksimum)

Kendala : $\sum_{i=1}^{n} \left(\sum_{k=1}^{Ki} \rho^j_{ki} x_{ki} + g^j_i(a_{0I}) \right) \le b_j$, untuk $j = 1,2, \ldots, m.$

$$0 \le x_{ki} \le a_{ki} - a_{k-1,I}\ ,$$

untuk $k = 1, 2, \ldots, k_i$, $\quad i = 1, 2, \ldots, n$

dimana $\rho_{ki} = \dfrac{f_i(a_{ki}) - f_i(a_{k-1,i})}{a_{ik} - a_{k-1,i}}$ dan $\rho^j_{ki} = \dfrac{g^j_i(a_{ki}) - g^j_i(a_{k-1,i})}{a_{ik} - a_{k-1,i}}$

2.6. ALGORITMA NON LINIER UNTUK MASALAH PEMROGRAMAN BILANGAN BULAT

2.6.1. DEFINISI PERMASALAHAN[18]

Seperti yang telah dijelaskan pada bab pendahuluan subbab permasalahan, suatu masalah *mixed bilangan bulat programming* 1-0 dapat diselesaikan dengan pendekatan *non-linier* . Jika kita mempunyai suatu formulasi permasalahan P sebagai berikut:

Minimum : $z = cx$

Kendala : $Ax \leq b$

$x = 0$ atau $x = 1$

Maka kita asumsikan permasalahan ini dapat didekati dengan fungsi kuadratik nonkonvek berikut:

Minimumkan : $z_Q = \sum (x_i - x_i^{\,2})$

Kendala : $0 \leq x_i \leq 1$

$z \leq e$

dimana e adalah suatu parameter, yang dipilih sedemikian hingga jika semua variabel dari permasalahan P mempunyai jawab layak yang bulat, maka dengan kendala itu akan dapat ditentukan jawaban yang memuaskan. Mencari nilai e adalah dengan menentukan nilai variabel bilangan bulat, yang hasil fungsi tujuannya paling mendekati nilai z sebenarnya dan jawaban tersebut layak.

---

[18] A. Ciriani Tito and C. Leachman, Robert, Optimization in Industry : Mathematical Programming and Modelling Techniques in Practice, John Wiley & Sons, Inc., 1993, hal. 202-203.

2.6.2. ALGORITMA Q[19]

Secara umum, cukup sulit menemukan suatu solusi optimal (*global optimum solution*) dalam masalah pemrograman kuadratik non-konvek. Tetapi untuk masalah ini, persoalannya menjadi lebih mudah. Pertama kita sudah punya kendala bahwa jika $z_Q = 0$, maka semua nilai variabelnya haruslah bernilai 0 atau 1.

Prinsip algoritma ini sangat sederhana. Yaitu terdiri dari prosedur Branch and Bound dengan fungsi tujuan berupa fungsi kuadrat di tiap iterasinya. Bukan berupa fungsi linier seperti biasanya.

- Langkah 1 :

  Menemukan solusi optimum lokal untuk fungsi kuadrat : $z_Q = \sum (x_i - x_i^2)$. Jika $z_Q = 0$, maka jawab layak untuk masalah P* tercapai. Jika tidak, variabel x yang mempunyai nilai tidak bulat diganti nilainya menjadi 0 atau 1, dan dilanjutkan ke langkah berikutnya.

- Langkah 2

  Membuat sebuah pohon biner (*binary tree*). Sebuah node diabaikan jika tidak layak dan pencarian dihentikan jika solusi optimalnya tercapai.

Untuk menjalankan prosedur ini, digunakan algoritma kuadratik dalam perhitungan. Dalam algoritma ini kita menggunakan modifikasi pemrograman separabel bentuk-δ atau pemrograman separabel konvek, dengan asumsi bahwa batas atasnya sudah ditentukan.

---

[19] A. Ciriani Tito and C. Leachman, Robert, Optimization in Industry : Mathematical Programming and Modelling Techniques in Practice, John Wiley & Sons, Inc., 1993, hal. 203-204.

Dari fungsi : $z_Q = (x_i - x_i^2)$, kita misalkan tiap variabel x_i merupakan penjumlahan dari dua variabel khusus, sebut saja x_i^* dan x_i^{**} :

$$x_i = x_i^* + x_i^{**}$$

dengan mempertimbangkan kendala

$$0 \leq x_i \leq 1$$

atau sama dengan :

$$0 \leq x_i^* + x_i^{**} \leq 1$$

maka variabel yang baru mempunyai kendala :

$$0 \leq x_i^* \leq \tfrac{1}{2}$$

$$0 \leq x_i^{**} \leq \tfrac{1}{2}$$

Kemudian kita menggunakan formulasi dari algoritma pemrograman separabel konvek. Di mana untuk suatu permasalahan kuadratik :

Minimumkan : $z = \sum f(x)$

Kendala : $\sum g(x) \leq 0$

dengan algoritma separabel dapat didekati dengan :

$$\text{Minimumkan} : z = \sum_{i=1}^{n} \left(\sum_{k=1}^{Ki} \rho^j_{ki} x_{ki} + f_i(a_{0I}) \right)$$

$$\text{Kendala} : \sum_{i=1}^{n} \left(\sum_{k=1}^{Ki} \rho^j_{ki} x_{ki} + g^j_i(a_{0I}) \right) \leq b_j \text{ , untuk } j = 1,2, \ldots, m.$$

$$0 \leq x_{ki} \leq a_{ki} - a_{k-1,I} \text{ ,}$$

$$\text{untuk } k = 1, 2, \ldots, k_i \text{ , } \quad i = 1, 2, \ldots, n$$

dimana $\rho_{ki} = \dfrac{f_i(a_{ki}) - f_i(a_{k-1,i})}{a_{ik} - a_{k-1,i}}$ dan $\rho^j_{ki} = \dfrac{g^j_i(a_{ki}) - g^j_i(a_{k-1,i})}{a_{ik} - a_{k-1,i}}$

Maka secara umum jika kita mempunyai permasalahan seperti di atas :

$$\text{Minimumkan} : \sum c_i(x_i - x_i^2)$$

$$\text{Kendala} : \sum g(x_i) \leq 0$$

$$0 \leq x_i \leq 1$$

kita akan memperoleh model permasalahan baru :

$$\text{Minimumkan} : z_Q^* = \sum \tfrac{1}{2} c_i(x_i^* - x_i^{**})$$

$$\text{Kendala} : \sum g(x_i^* + x_i^{**}) \leq 0$$

$$0 \leq x_i^*, x_i^{**} \leq \tfrac{1}{2}$$

Hal ini dapat dibuktikan dengan cara :

1. Perhitungan

   Range untuk slope yang kita gunakan dalam masalah ini adalah ½. Sehingga satu fungsi f(x) kita bagi dalam dua slope yang masing-masing lebarnya ½. Kita memperoleh nilai K = 2. (k = 0,1,2). Maka kita dapat menghitung, dengan asumsi $f_i(0) = 0$, untuk :

   a. $k = 1$, $a_{1i} = \tfrac{1}{2}$

$$\rho_{1i} = \frac{f_i(x_i = a_{1i}) - f_i(x_i = a_{0i})}{a_{1i} - a_{0i}}$$

$$\rho_{1i} = \frac{f_i(x_i = \tfrac{1}{2}) - f_i(x_i = 0)}{\tfrac{1}{2} - 0}$$

$$\rho_{1i} = \frac{\frac{1}{4} c_i - 0}{\frac{1}{2}} = \frac{1}{2} c_i$$

b. $k = 2, a_{1i} = 1$

$$\rho_{2i} = \frac{f_i(x_i = a_{2i}) - f_i(x_i = a_{1i})}{a_{2i} - a_{1i}}$$

$$\rho_{2i} = \frac{f_i(x_i = 1) - f_i(x_i = \frac{1}{2})}{1 - \frac{1}{2}}$$

$$\rho_{2i} = \frac{0 - \frac{1}{4} c_i}{\frac{1}{2}} = -\frac{1}{2} c_i$$

Dari perhitungan di atas kita memperoleh :

$$z = \sum (\sum \rho^j_{ki} x_{ki} + f_i(a_{0l}))$$

$$z = \sum (\rho_{1i} x_{1i} + \rho_{2i} x_{2i} + 0)$$

$$z = \sum (\frac{1}{2} c_i x_{1i} - \frac{1}{2} c_i x_{2i})$$

Jika $x_{1i} = x_i^*$ dan $x_{2i} = x_i^{**}$, maka terbukti bahwa :

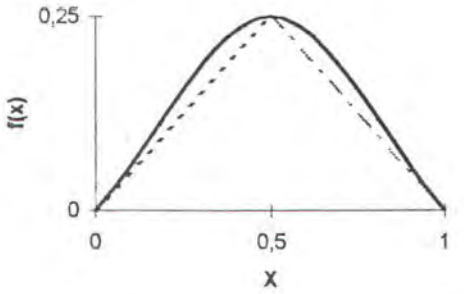
$$z = \sum \frac{1}{2} c_i (x_i^* - x_i^{**})$$

2. Grafik

Jika kita memiliki fungsi $f(x) = x - x^2$, dengan range x adalah ½ , maka kita memiliki data sebagai berikut :

x	0	½	1
f(x)	0	¼	0

Grafik dari fungsi kuadrat di atas, adalah :



Gambar 2.2

Grafik Fungsi $f(x) = x - x^2$

Fungsi tersebut didekati dengan dua fungsi linier, yang masing-masing mempunyai range yang sama. Kedua fungsi tersebut :

a. Garis - - - -

   Garis tersebut mempunyai gradien $f'(x^*) = ¼ / ½ = ½$, maka persamaan fungsinya menjadi $f(x^*) = ½ x^* + c$ (c adalah suatu konstanta)

b. Garis – - – -

   Gradien dari garis ini adalah $f'(x^{**}) = ¼ / -½ = -½$, sehingga persamaan fungsinya menjadi $f(x^{**}) = -½ x^{**} + c$.

Jika kedua fungsi di atas ditambahkan maka diperoleh :

$$f(x^* + x^{**}) = ½ x^* - ½ x^{**} + c$$

atau

$$f(x^* + x^{**}) = ½ (x^* - x^{**}) + c$$

Jadi benar bahwa jika kita mempunyai fungsi kuadrat $z = f(x)$, maka dapat didekati dengan fungsi linier :

$$z = ½ (x^* - x^{**})$$

Model permasalahan z_Q^*, karena berupa fungsi linier dapat diselesaikan dengan metode Simpleks, khususnya metode Simpleks untuk variabel yang dibatasi (*Bounded Variable*), karena dalam model tersebut terdapat kendala : $0 \leq x_i^* \leq ½$ dan $0 \leq x_i^{**} \leq ½$ untuk $i = 1, 2, \ldots, n$.

2.6.3. TEKNIK *HEURISTIC*[20]

Teknik ini merupakan gabungan penggunaan teknik pemeriksaan dan prosedur Branch and Bound, seperti yang terlihat pada gambar 2.3. Secara garis besar prosedur yang digunakan adalah sebagai berikut :

1. Pembentukan model untuk pemrograman separabel seperti yang telah dijelaskan pada subbab sebelumnya.
2. Pertama-tama kita set variabel separabelnya pada batas atas batas bawahnya, sesuai dengan aturan
   - Jika yang masuk dalam proses iterasi adalah variabel separabel x_{j1} maka variabel x_{j2} diset pada batas bawah (= 0).
   - Jika yang masuk dalam proses iterasi adalah variabel x_{j2} maka variabel x_{j1} diset pada batas atasnya (= ½).

   Kemudian permasalahan dirumuskan ulang sesuai dengan nilai variabel yang diset.
3. Prosedur pemrograman separabel dijalankan untuk menemukan solusi optimal lokal dari permasalahan yang baru dirumuskan di atas.

---

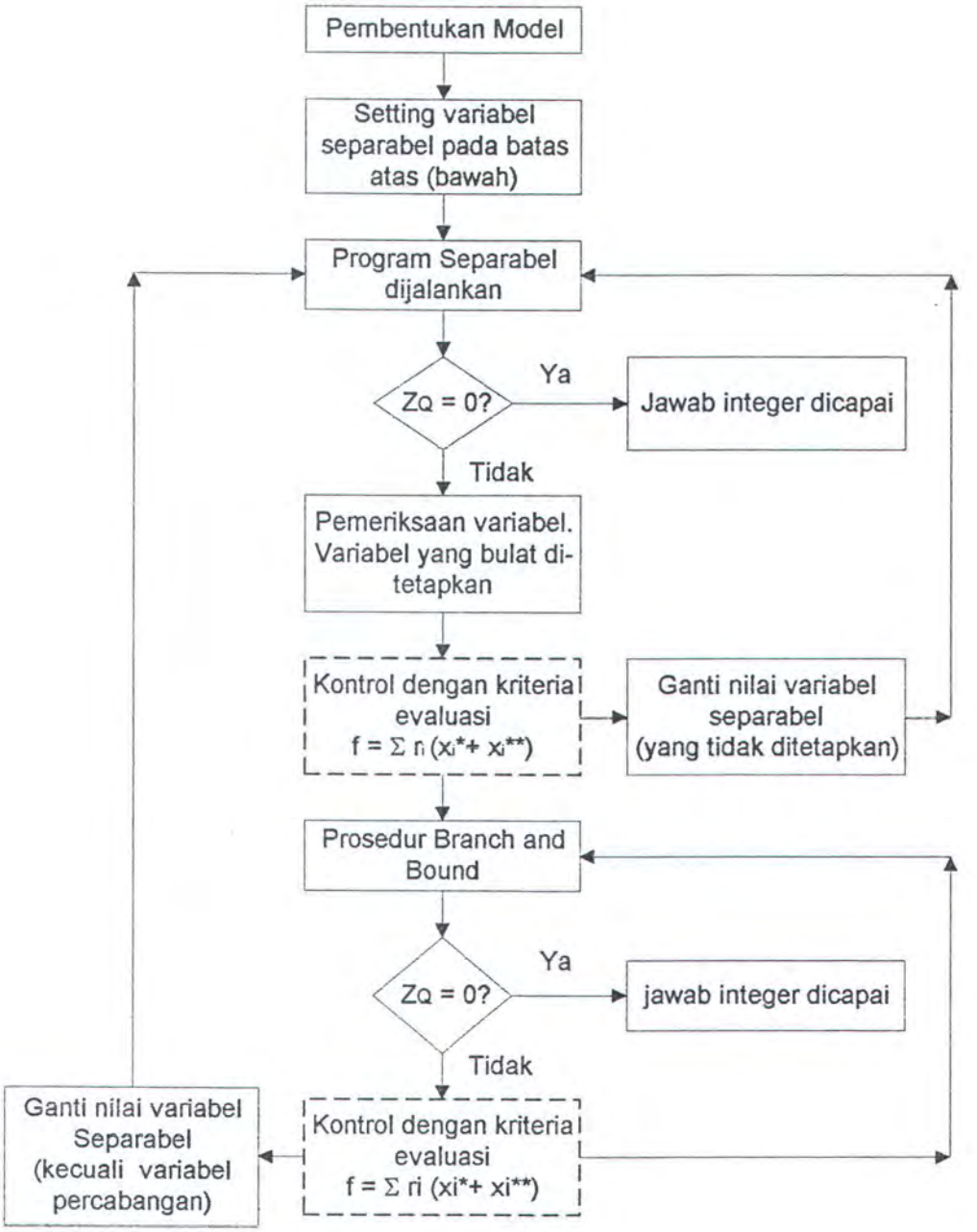
[20] A. Ciriani Tito and C. Leachman, Robert, Optimization in Industry : Mathematical Programming and Modelling Techniques in Practice, John Wiley & Sons, Inc., 1993, hal. 205-207.

4. Dicek apakah $z_Q = 0$. Jika ya, maka optimum dari fungsi kuadrat sudah tercapai atau dengan kata lain jawab layak integer untuk masalah P* telah ditemukan. Jika tidak, pencarian dilanjutkan ke langkah berikutnya.
5. Teknik pemeriksaan. Pemeriksaan dilakukan terhadap pasangan variabel separabel yang dihasilkan. Dicek apakah nilai $x_i^* = x_i^{**} = 0$ atau ½ , yang berarti $x_i^* + x_i^{**} = 0$ atau 1.
6. Selanjutnya dilakukan evaluasi terhadap variabel-variabel separabel. Dengan menggunakan kriteria, yang digambarkan dalam fungsi:

   $$f = \sum r_i (x_i^* + x_i^{**})$$

   Dimana, $r_i = 1$ jika nilai $x_i = 0$; $r_i = -1$ jika $x_i = 1$; dan selain jawaban itu $r_i = 0$. Fungsi ini untuk mencek jumlah variabel 0-1 pada saat itu. Jika f mencapai nilai minimum maka kita ulangi proses dengan teknik pemeriksaan, dengan *setting* variabel yang berbeda. Jika $f = 0$, kita lakukan prosedur Branch and Bound.
7. Variabel yang telah ditetapkan, dibebaskan, untuk digunakan dalam prosedur Branch and Bound.
8. Membuat node pohon biner (*binary tree*), dengan permasalahan yang telah ditentukan
9. Dari node yang dihasilkan dicek apakah $z_Q = 0$. Jika ya, berhenti. Jika tidak lanjutkan ke langkah berikutnya.
10. Kembali dilakukan evaluasi dengan fungsi f, seperti yang telah dijelaskan di atas. Jika nilai f mencapai nilai minimum, prosedur Branch and Bound diulang

dengan membuat node baru untuk submasalah dengan variabel percabangan tertentu.



Gambar 2.3

Flow Diagram Gabungan Teknik Pemeriksaan dan Prosedur Branch&Bound

11. Jika nilai f = 0, kita ulangi proses teknik pemeriksaan. Untuk itu, nilai variabel separabel diganti ke batas sebaliknya (jika sebelumnya diset pada batas atasnya, diganti ke batas bawah, demikian pula sebaliknya), kecuali untuk variabel yang digunakan sebagai variabel percabangan pada prosedur Branch and Bound.
12. Kembali ke pemrograman Separabel.
13. Jika nilai $z_Q = 0$, maka variabel bilangan bulatnya tercapai. Selanjutnya, nilai tiap variabel dimasukkan dalam fungsi tujuan permasalahan awal. Lalu dilihat apakah nilai z yang dihasilkan memenuhi syarat fungsi kendala. Jika memenuhi kita masukkan dalam stack yang menyimpan hasil nilai optimal. Node yang menghasilkan nilai tersebut dihentikan (*fathomed*).
14. Proses diulang sampai semua node telah dihentikan (*fathomed*).

BAB III

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB III

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Perangkat lunak ini dirancang dengan menggunakan pemrograman yang berorientasi pada obyek atau biasa disebut OOP (*Object Oriented Programming*) dengan bahasa Borland C++ 4.5. Ada dua konsep penting dalam pemrograman yang berorientasi pada objek yaitu objek dan kelas.

Dalam OOP data dan fungsi-fungsi yang akan mengoperasikan data digabungkan dalam satu kesatuan yang dapat disebut objek (*object*). Dan objek merupakan anggota dari kelas.

Dalam program ini dibagi dalam beberapa kelas utama yaitu

3.1. CLASS INPUT

Kelas ini digunakan untuk mengambil informasi formulasi permasalahan yang diinputkan dan menyimpannya dalam bentuk matriks.

3.2. CLASS SETMATRIKS

Kelas ini digunakan untuk menset matriks ke dalam bentuk yang diinginkan. Misalnya ke model permasalahan kuadratik, ke model pemrograman separabel, dan sebagainya. Struktur data dari kelas ini adalah sebagai berikut:

```
const double M = 1000;
struct pos
{
        int x;
        int y;
};

class SetMatriks
{
        .
        .
        .

        public:
                SetMatriks(FMatrix* a, int* sg, char* pr);
                ~SetMatriks();

                void SetSlack();
                void pivoting(int rpv, int cpv);
                void SetMatrixStand(FMatrix* a);
                void SetZ();
                void SetZQ1(FMatrix* a);
                void SetZQ2(FMatrix* a);
                void SendMatZQ1(FMatrix* a);
                void SendMatZQ2(FMatrix* a);
                void SendMatStandar(FMatrix* a);
                int  ColZQ();
                int  ColStand() {return col2;}
};
```

Penjelasan dari tiap fungsi anggota kelas tersebut adalah:

1. *Constructor* SetMatriks

   Dengan parameter FMatrix* *a*, merupakan matriks model permasalahan, int **sign*, merupakan array dari bilangan integer yang mewakili tanda ketidaksamaan dari fungsi kendala permasalahan, dan char* *problem* yang berisi bentuk permasalahan, maksimasi (Max) atau Minimisasi (Min). Contoh bentuk matriks untuk permasalahan :

Minimumkan : $z = 3x_1 + 5x_2$

dengan kendala : $x_1 \leq 4$

$2x_2 = 12$

$3x_1 + 2x_2 \geq 18$

adalah matriks ukuran 4X3:

3	5	0
1	0	4
0	2	12
3	2	18

2. Fungsi SetSlack();

Fungsi ini digunakan untuk membentuk matriks tambahan bagi bentuk matriks standar dalam proses algoritma Simpleks. Yaitu matriks untuk variabel tambahan (*slack variable*) dan variabel tiruan (*artificial variable*). Matriks tersebut disimpan dalam variabel dummyMat.

Prosesnya, pertama adalah menghitung berapa banyaknya variabel tambahan dan variabel tiruan, dengan melihat tanda ketidaksaman dari fungsi-fungsi kendala permasalahan.

* Jika tandanya kurang atau sama dengan (≤), maka yang dibutuhkan adalah menambahkan satu variabel tambahan.
* Jika tandanya lebih besar atau sama dengan (≥) maka fungsi kendalanya dikurangi satu variabel tambahan dan ditambah satu variabel tiruan
* Jika tandanya sama dengan (=), maka ditambahkan satu variabel tiruan.

Dimana variabel tiruannya mempunyai nilai konstanta yang besar sekali (menggunakan metode Big-M), dalam program ini menggunakan nilai

M=1000. Bentuk permasalahan maksimasi atau minimisasi juga mempengaruhi nilai konstanta M. Jika masalahnya maksimasi, maka konstanta M bernilai negatif (–) Sedangkan untuk masalah minimisasi, konstantanya bernilai positif. Bentuk matriks tambahan untuk permasalahan diatas adalah matriks ukuran 4X4 seperti berikut :

0	M	0	M
1	0	0	0
0	1	0	0
0	0	-1	1

3. Fungsi SetMatrixStand

Suatu bentuk permasalahan permograman linier harus dirubah ke bentuk standar lebih dahulu jika akan diproses dengan metode Simplek. Bentuk standar ini dibentuk dengan menambahkan variabel tambahan (*slack variable*) dan atau variabel tiruan (*artificial variable*). Matriks bentuk standar untuk permasalahan di atas adalah seperti berikut ini :

3M-3	4M-5	0	0	-M	0	30M
1	0	1	0	0	0	4
0	2	0	1	0	0	12
3	2	0	0	-1	1	18

Dalam fungsi ini, matriks asal digabungkan dengan matriks tambahan yang dibuat dalam fungsi SetSlack. Kemudian mengkonversikan ke dalam bentuk standar, dimana konstanta variabel yang menjadi basis dari fungsi tujuan harus bernilai 0.

4. Fungsi Pivoting

   Fungsi ini digunakan untuk mengkonversikan bentuk gabungan matriks asal dengan matriks tambahan ke bentuk standar, jika ditemukan adanya konstanta M pada fungsi tujuan. Yang tujuannya adalah untuk meng-nol-kan konstanta variabel tambahan untuk fungsi ketidaksaman (≤) dan variabel tiruan untuk fungsi ketidaksamaan (=) atau (≥) pada fungsi tujuannya. Fungsi ini dipanggil di dalam fungsi SetMatrixStandar. Hasilnya disimpan dalam variabel matriks Mat.

5. Fungsi SetZQ1

   Fungsi ini digunakan untuk merubah bentuk permasalahan z ke bentuk permasalahan non linier z_Q. Parameter yang digunakan adalah FMatrix* a, yang merupakan matriks untuk permasalahan z. Dari contoh permasalahan z diatas, maka bentuk matriks permasalahan z_Q adalah :

3	-3	5	-5	0
1	1	0	0	4
0	0	2	2	12
3	3	2	2	18

   Dan hasilnya disimpan dalam variabel MatZQ1. Matriks ini untuk memudahkan konversi ke bentuk pemrograman separabel.

6. Fungsi SetZQ2

   Fungsi ini digunakan untuk merubah matriks dari bentuk permasalahan kuadratik, z_Q, ke bentuk permasalahan pemrograman separabel, z_Q*. Parameter yang digunakan adalah variabel matriks, yang mewakili matriks permasalahan kuadratik atau non linier yang akan dikonversi. Caranya adalah dengan

mengalikan ½ tiap konstanta fungsi tujuan dari permasalahan non linier. Hasilnya disimpan dalam variabel matriks MatZQ2.

7. Fungsi SendMatZQ1

   Fungsi ini digunakan untuk mengirimkan variabel matriks MatZQ1, yang mewakili matriks permasalahan non linier, untuk digunakan pada fungsi yang memanggil. Fungsi ini menggunakan parameter variabel matriks, yang mewakili matriks yang menerima.

8. Fungsi SendMatZQ2

   Sama dengan fungsi SendMatZQ2, fungsi ini digunakan untuk mengirimkan variabel matriks MatZQ2, yang mewakili matriks permasalahan pemrograman separabel.

9. Fungsi SendMatStandar

   Fungsi ini digunakan untuk mengirimkan variabel matriks standar, dengan parameter variabel matriks yang menerima.

3.3. CLASS SIMPLEXBASECLASS

Kelas ini merupakan sekumpulan fungsi yang dapat digunakan oleh kelas Simplex ataupun kelas SimplexBounded. Fungsi-fungsi tersebut adalah:

```
class SimplexBaseClass
{
        FMatrix* A;
        int      row,col,col2;
```

```
public:
        SimplexBaseKelas(FMatrix* a,int cl2);
        ~SimplexBaseKelas() {delete A;}

        void   GaussJordan(FMatrix* a,int rpv, int cpv);
        void   SendMatrix(FMatrix* a);
        double min(double a, double b, double c);
        double max(double a, double b, char* prob);
};
```

1. Fungsi GaussJordan

   Fungsi ini merupakan operasi eliminasi Gauss-Jordan, dengan parameter variabel matriks yang akan dieliminasi, yaitu matriks *a*, variabel baris yang akan meninggalkan basis (*leaving variable*), yaitu variabel *rpv*, dan variabel kolom yang akan masuk basis (*entering variable*), yaitu variabel *cpv*. Algoritmanya secara lengkap dapat ditulis sebagai berikut :

   - Menentukan elemen pivot yaitu nilai dari perpotongan *entering column* dengan baris dari variabel yang akan meninggalkan basis, baris pivot.
   - Setiap nilai pada baris pivot dengan nilai elemen pivot.
   - Lakukan operasi baris elementer terhadap *entering column*. Pertama, menentukan nilai pengali yaitu nilai pada baris yang akan diproses. Kemudian semua kolom pada baris tersebut dikurangi hasil perkalian antara nilai pada baris pivot dengan nilai pengali.

2. Fungsi Min

   Fungsi ini digunakan untuk mencari nilai paling minimum dari beberapa nilai variabel yang diinputkan.

3. Fungsi Max

Dengan menggunakan parameter double a, double b dan char* prob, fungsi ini digunakan untuk mencari nilai positif terbesar atau nilai negatif terbesar dari beberapa nilai berdasarkan model permasalahannya. Parameter char* prob mewakili model permasalahan, yaitu Max atau Min. Jika masalahnya adalah Max, maka dicari nilai negatif terbesar, sebaliknya jika masalahnya Min, maka yang dicari adalah nilai positif terbesar.

3.4. CLASS SIMPLEX

Kelas Simplex merupakan sekumpulkan fungsi-fungsi yang digunakan untuk menyelesaikan suatu permasalahan pemrograman linier dengan metode Simpleks. Struktur dari kelas ini adalah :

```
class Simplex
{
        .
        .
        .

        public:
        Simplex(FMatrix* MatZ, int cl2, char* prob);
        ~Simplex();
        void ProsesSimplex();
        void init(void);
        void pivot(void);
};
```

Adapun fungsi-fungsi yang menjadi bagian dari kelas ini adalah :

1. Fungsi ProsesSimplex

   Fungsi ini merupakan fungsi utama dari kelas Simplex. Fungsi ini menjalankan fungsi pivot sampai solusi optimalnya tercapai, yaitu dengan melihat nilai dari konstanta fungsi tujuannya, pada baris ke nol dari matriknya. Untuk masalah maksimasi, jika konstanta pada baris ke nol semuanya bernilai nol atau positif maka penyelesaian optimalnya tercapai dan proses dihentikan. Sedangkan untuk masalah minimisasi, jika konstanta pada baris ke nol semuanya bernilai nol atau negatif maka penyelesaian optimalnya tercapai, dan iterasi dihentikan.

2. Fungsi init

   Merupakan fungsi inisialisasi awal untuk seluruh proses dalam kelas.

3. Fungsi pivot

   Didalam fungsi ini proses metode simpleks sebenarnya dijalankan. Secara lengkap, algoritma dari fungsi ini dapat ditulis sebagai berikut :

   - Pada baris ke-0, dicari nilai paling negatif untuk permasalahan maksimasi atau paling positif untuk permasalahan minimisasi. Kolom tempat nilai tersebut dinamakan *entering column*.
   - Menentukan *entering row*, yaitu mencari nilai perbandingan positif terkecil antara nilai setiap baris pada *entering column* dengan kolom terakhir, yang dimulai pada baris ke-1.
   - Melakukan operasi Gauss-Jordan dengan *entering column* dan *entering row* sebagai parameternya.
   - Selanjutnya diset nilai variabelnya dengan melihat nilai variabel basis.

3.5. CLASS SIMPLEXBOUNDED

Kelas ini merupakan sekumpulan fungsi dan obyek yang digunakan untuk menyelesaikan permasalahan *simplex bounded variable* atau metode simplek yang menggunakan variabel pembatas. Dimana metode simplek *bounded* ini nantinya akan digunakan untuk menyelesaikan pemrograman non linier *separable.* Secara garis besar adalah sebagai berikut:

```
class SimplexBounded
{
        .
        .
        .
        .

    public:
        SimplexBounded(FMatrix* a,int cl2, char* prob);
        ~SimplexBounded();

        void SendMatrix(FMatrix* a);
        void SendX(double* x);
        void init();
        void pivot(void);
        void SetX();
        void ProsesSimplexBounded();
};
```

Secara umum, fungsi-fungsi yang terdapat kelas ini mempunyai persamaan dengan fungsi-fungsi yang terdapat pada kelas Simplex sebelumnya. Yang membedakannya hanya terdapat pada fungsi pivotnya saja. Penjelasan untuk fungsi pivot adalah sebagai berikut :

1. Pada baris ke-0, dicari nilai paling negatif untuk permasalahan maksimasi atau paling positif untuk permasalahan minimisasi. Kolom tempat nilai tersebut dinamakan *entering column.*

2. Menentukan *entering row*, dengan cara :

   a. Jika nilai baris pada kolom *entering* lebih besar dari 0, maka nilai perbandingan, disebut θ_1, adalah nilai perbandingan antara nilai pada kolom terakhir dengan nilai pada *entering column*.

   b. Jika nilai baris pada kolom *entering* kurang atau sama dengan 0, maka nilai perbandingan, disebut θ_2, adalah nilai perbandingan antara hasil pengurangan nilai batas atas dengan nilai pada kolom terakhir, dengan nilai pada *entering column.*

   Selanjutnya dicari nilai positif minimum, sebut saja θ, diantara nilai θ_1, θ_2 dan batas atasnya untuk menentukan *entering row*.

3. Operasi yang dilakukan selanjutnya tergantung nilai θ.

   a. Jika $\theta = \theta_1$, dilakukan operasi Gauss-Jordan dengan parameter *entering row* dan *entering column*.

   b. Jika $\theta = \theta_2$, dilakukan operasi Gauss-Jordan dengan parameter *entering row* dan *entering column*, kemudian variabel yang meninggalkan basis, misal x_j, ditentukan pada batas atasnya, $x_j' = x_j - 0.5$, dan selanjutnya x_j' yang ikut dalam proses.

   c. Jika $\theta = 0.5$, variabel yang akan masuk basis, yaitu variabel pada *entering column*, misal x_j, ditentukan pada batas atasnya, $x_j' = x_j - 0.5$, dan selanjutnya x_j' yang ikut dalam proses. Variabel x_j', tetap menjadi variabel nonbasis.

4. Selanjutnya diset nilai variabelnya dengan melihat nilai variabel basis.

3.6. CLASS HEURISTIC

Kelas ini merupakan kelas utama, yang memproses gabungan antara teknik pemeriksaan dengan prosedur Branch and Bound. Kelas ini terdiri dari beberapa fungsi yaitu :

1. Fungsi SettingVariabel

   Fungsi ini digunakan untuk menetapkan *setting* variabel awal. Dimana variabel separabel x_{j2} diset pada batas bawah (0). Kemudian permasalahan yang disimpan dalam matriks, diformulasikan ulang dengan $x_{j2} = 0$. Hasilnya disimpan dalam matriks baru untuk permasalahan baru.

2. Fungsi FixingVariabel

   Fungsi ini digunakan untuk menetapkan pasangan variabel separabel yang keduanya sama-sama bernilai 0 atau 1. Variabel yang ditetapkan disimpan dalam suatu array.

3. Fungsi Separable

   Fungsi ini sebenarnya hanya memanggil prosedur Simpleks dengan variabel pembatas (*Bounded Variable*), untuk menyelesaikan formulasi masalah z_Q^*.

4. Fungsi Changing

   Fungsi ini untuk merubah *setting* variabel separabelnya. Jika semula x_{j2} yang diset pada batas bawahnya ($x_{j2} = 0$), maka sekarang x_{j1} diset pada batas atas ($x_{j1} = 0.5$). Selanjutnya permasalahan baru dibentuk dengan *setting* variabel yang baru.

5. Fungsi Evaluation

Fungsi ini mengembalikan nilai f yang telah dihitung berdasarkan variabel yang dihasilkan, dengan aturan seperti yang telah dijelaskan sebelumnya.

6. Fungsi SetZQ

Fungsi ini mengembalikan nilai z_Q yang telah dihitung dari permasalahan yang sedang diproses.

7. Fungsi BranchBound

Fungsi ini digunakan untuk membuat node dari pohon biner, dan tiap nodenya mempunyai struktur sebagai berikut :

```
struct node
{
    FMatrix* Matriks;
    StructVar* VarX;
    StructVar* VarBranch;
    int    jum1;
    int    jum2;
    node*  left;
    node*  right;
};
```

Matriks digunakan untuk menyimpan permasalahan untuk node tersebut. Sedang *VarX* untuk menyimpan variabel yang ikut dalam proses atau yang merupakan variabel-variabel dari formulasi permasalahannya, dan *VarBranch* merupakan variabel untuk menyimpan variabel percabangan pada saat itu. Lalu jum1 adalah jumlah atau banyaknya variabel proses dan jum2 merupakan jumlah variabel percabangannya.

Prosedur ini hanya membuat satu node setiap kali dipanggil. Setiap kali itu juga nilai z_Q yang dihasilkan diperiksa, untuk kemudian ditentukan apakah

diteruskan menggunakan prosedur ini atau ganti dengan teknik pemeriksaan atau sudah mencapai nilai integer. Jika suatu node mempunyai nilai $z_Q = 0$, variabel-variabelnya diperiksa kelayakannya. Jika variabelnya tidak layak terhadap fungsi kendalanya, proses dilanjutkan dengan membuat node baru yang merupakan anak cabang dari node lainnya yang belum mencapai nilai $z_Q=0$.

BAB IV

ANALISA HASIL

BAB IV

ANALISA HASIL

Dengan menggunakan perangkat lunak yang telah dibuat, kita dapat melihat hasil penyelesaian suatu permasalahan pemrograman bilangan bulat nol-dan-satu dengan beberapa fungsi kendala. Misalnya kita mempunyai permasalahan sebagai berikut :

Minimumkan : $z = 3x_1 + 5x_2 + 6x_3 + 9x_4 + 10x_5 + 10x_6$

dengan kendala : $-2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 - 2x_6 \geq 2$

$-5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 + x_6 \geq -2$

$5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 - x_6 \geq 3$

$x_j = 0$ atau 1

Sebelum kita melakukan langkah-langkah pada prosedur utama, kita lihat dulu nilai optimum jika variabelnya tidak dibatasi dengan kendala 0 dan 1. Permasalahan z diatas dapat diselesaikan dengan metode Simpleks. Iterasinya adalah sebagai berikut :

- Iterasi 0 :

	x_1	x_2	x_3	x_4	x_5	x_6	R_1	s_1	s_2	R_2	s_3	RHS
z	2997	4995	994	1991	2990	-3010	0	-1000	0	0	-1000	5000
R_1	-2	6	-3	4	1	-2	1	-1	0	0	0	2
s_2	5	3	-1	-3	2	-1	0	0	1	0	0	2
R_2	5	-1	4	-2	2	-1	0	0	0	1	-1	3

☞ Iterasi 1 :

	x_1	x_2	x_3	x_4	x_5	x_6	R_1	s_1	s_2	R_2	s_3	RHS
z	4662	0	3491.5	-1339	2157.5	-1345	-832.5	-167.5	0	0	-1000	3335
x_2	-0.33	1	-0.5	0.67	0.167	-0.33	0.167	-0.167	0	0	0	0.33
s_2	6	0	0.5	-0.5	1.5	0	-0.5	0.5	1	0	0	1
R_2	4.67	0	3.5	-1.33	2.167	-1.33	0.167	-0.167	0	1	-1	3.33

☞ Iterasi 2 :

	x_1	x_2	x_3	x_4	x_5	x_6	R_1	s_1	s_2	R_2	s_3	RHS
z	0	0	3103	2546	992	-1345	-444	-556	-777	0	-1000	2558
x_2	0	1	-0.472	0.389	0.25	-0.333	0.1389	-0.139	0.056	0	0	0.389
x_1	1	0	0.083	-0.083	0.25	0	-0.083	0.083	0.167	0	0	0.167
R_2	0	0	3.11	2.56	1	-1.333	0.56	-0.56	-0.778	1	-1	2.56

☞ Iterasi 3 :

	x_1	x_2	x_3	x_4	x_5	x_6	R_1	s_1	s_2	R_2	s_3	RHS
z	0	0	0	-2.89	-5.39	-15.14	-998.1	-1.89	-1.25	-997.4	-2.61	9.11
x_2	0	1	0	0.78	0.4	-0.54	0.22	-0.22	-0.063	0.15	-0.15	0.78
x_1	1	0	0	-0.9	0.2	0.04	-0.1	0.1	0.1875	-0.03	0.03	0.098
x_3	0	0	1	0.82	0.32	-0.43	0.18	-0.18	-0.25	0.32	-0.32	0.82

Dari hasil iterasi di atas, kita memperoleh nilai variabel : x_1=0.1; x_2=0.78; x_3=0.82; x_4=0; x_5=0; x_6=0. Sehingga nilai z = 9.11.

Lalu kita ubah permasalahan di atas ke dalam bentuk permasalahan non linier, yaitu dengan fungsi tujuan berupa fungsi kuadrat, sebut saja sub masalah 1.1, yaitu :

$$\text{Min} : z_Q = 3(x_1-x_1^2)+5(x_2-x_2^2)+6(x_3-x_3^2)+9(x_4-x_4^2)+10(x_5-x_5^2)+10(x_6-x_6^2)$$

dengan kendala : $-2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 - 2x_6 \geq 2$

$$-5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 + x_6 \geq -2$$

$$5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 - x_6 \geq 3$$

$$0 \le x_j \le 1$$

Dengan pemrograman separabel, permasalahan di atas dapat didekati dengan permasalahan baru, kita beri nama subproblem 1.2, yaitu :

Min: $z_Q^*=3/2(x_{11}-x_{12})+5/2(x_{21}-x_{22})+3(x_{31}-x_{32})+9/2(x_{41}-x_{42})+5(x_{51}-x_{52})+5(x_{61}-x_{62})$

kendala : $-2x_{11}-2x_{12}+6x_{21}+6x_{22}-3x_{31}-3x_{32}+4x_{41}+4x_{42}+ x_{51}+ x_{52}-2x_{61}-2x_{62} \ge 2$

$-5x_{11}-5x_{12}-3x_{21}-3x_{22}+ x_{31}+ x_{32}+3x_{41}+3x_{42}-2x_{51}-2x_{52}+ x_{61}+ x_{62} \ge -2$

$5x_{11}+5x_{12}- x_{21}- x_{22}+4x_{31}+4x_{32}-2x_{41}-2x_{42}+2x_{51}+2x_{52}- x_{61}- x_{62} \ge 3$

$0 \le x_{j1}, x_{j2} \le ½$

Selanjutnya kita jalankan prosedur *Heuristic* , yang merupakan gabungan antara teknik pemeriksaan dengan prosedur Branch and Bound. Langkah-langkahnya adalah sebagai berikut :

- Langkah 1 : Setting variabel separabel.

Dalam pemrograman Separabel, setiap variabel $x_j = x_{j1} + x_{j2}$. Pada masalah ini tiap variabel x_j, hanya x_{ji} yang bisa masuk dalam basis. Sehingga jika kita memilih i = 1 maka r = 2. Karena r>i, maka x_{jr} diset pada batas bawahnya yaitu 0. Kita memperoleh bahwa $x_{12} = 0$; $x_{22} = 0$; $x_{32} = 0$; $x_{42} = 0$; $x_{52} = 0$; Dengan demikian kita mempunyai submasalah pemrograman separabel baru, sub masalah 2.2, yaitu :

Min: $z_Q^*=3/2x_{11} + 5/2x_{21} + 3x_{31} + 9/2x_{41} + 5x_{51} + 5x_{61}$

kendala : $-2x_{11} + 6x_{21} - 3x_{31} + 4x_{41} + x_{51} - 2x_{61} \ge 2$

$-5x_{11} - 3x_{21} + x_{31} + 3x_{41} - 2x_{51} + x_{61} \ge -2$

$5x_{11} - x_{21} + 4x_{31} - 2x_{41} + 2x_{51} - x_{61} \ge 3$

$0 \le x_{j1} \le ½$

- Langkah 2 : Menyelesaikan formulasi di atas dengan pemrograman separabel

Permasalahan separabel di atas dapat diselesaikan dengan metode Simpleks untuk *Bounded Variable.* Iterasinya adalah sebagai berikut :

▸ Iterasi 0 :

	x_{11}	x_{21}	x_{31}	x_{41}	x_{51}	x_{61}	R_1	s_1	s_2	R_2	s_3	RHS
z	2998.5	4997.5	997	1995.5	2995	-3005	0	-1000	0	0	-1000	5000
R_1	-2	6	-3	4	1	-2	1	-1	0	0	0	2
s_2	5	3	-1	-3	2	-1	0	0	1	0	0	2
R_2	5	-1	4	-2	2	-1	0	0	0	1	-1	3

▸ Iterasi 1 :

x_{21} = 4997.5 adalah *entering variable*. θ_1=min{2/6, 2/3} = 2/6. θ_2=(0.5-3)/1= -2.5. Karena θ_2 bernilai negatif, maka $\theta_2 = \infty$ dan u_j = 0.5. Maka θ = min{0.33, ∞, 0.5} = 0.33. Karena $\theta = \theta_1$, x_{21} masuk basis dan R_1 meninggalkan basis.

	x_{11}	x_{21}	x_{31}	x_{41}	x_{51}	x_{61}	R_1	s_1	s_2	R_2	s_3	RHS
z	4664.3	0	3495.8	-1336.2	2162.1	-1339.2	-832.9	-167.1	0	0	-1000	3334.2
x_{21}	-0.33	1	-0.5	0.67	0.167	-0.33	0.167	-0.167	0	0	0	0.33
s_2	4	0	-2.5	-1	2.5	-2	0.5	-0.5	1	0	0	3
R_2	4.67	0	3.5	-1.33	2.167	-1.33	0.167	-0.167	0	1	-1	3.33

▸ Iterasi 2 :

x_{11} = 4664.3 jadi *entering variable*. θ_1 = min{3/4, 3.33/4.67} = 0.71. θ_2=(0.5-0.33)/0.33 = 0.5. u_j = 0.5. Maka θ = min{0.71, 0.5, 0.5} = 0.5. Kita pilih $\theta = u_j$, maka x_{11} tetap menjadi variabel non-basis, dengan x_{11} = 0.5 - x_{11}'. Tabel iterasi yang baru adalah sebagai berikut :

	x_{11}'	x_{21}	x_{31}	x_{41}	x_{51}	x_{61}	R_1	s_1	s_2	R_2	s_3	RHS
z	-4664.3	0	3495.8	-1336.2	2162.1	-1339.2	-832.9	-167.1	0	0	-1000	1002
x_{21}	0.33	1	-0.5	0.67	0.167	-0.33	0.167	-0.167	0	0	0	0.5
s_2	-4	0	-2.5	-1	2.5	-2	0.5	-0.5	1	0	0	1
R_2	-4.67	0	3.5	-1.33	2.167	-1.33	0.167	-0.167	0	1	-1	1

▸ Iterasi 3:

x_{31} akan masuk basis. $\theta_1 = 1/3.5 = 0.286$; $\theta_2 = \min\{(0.5-0.5)/0.5, (0.5-1)/2.5\} = 0$; $u_j = 0.5$. Maka $\theta = \min\{0.286, 0, 0.5\} = 0$. Karena $\theta = \theta_2$ maka x_{31} masuk basis dan x_{21} meninggalkan basis dan disubstitusikan $x_{21} = 0.5 - x_{21}'$. Tabelnya menjadi :

	x_{11}'	x_{21}'	x_{31}	x_{41}	x_{51}	x_{61}	R_1	s_1	s_2	R_2	s_3	RHS
z	-2333.8	-6991.5	0	3324.8	3327.3	-3669.7	323.3	-1332.3	0	0	-1000	1002
x_{31}	-0.67	2	1	-1.33	-0.33	0.67	-0.33	0.33	0	0	0	0
s_2	-5.67	5	0	-4.33	1.67	-0.33	-0.33	0.33	1	0	0	1
R_2	-2.33	-7	0	3.33	3.33	-3.67	1.33	-1.33	0	1	-1	1

▸ Iterasi 4:

x_{51} akan masuk basis. $\theta_1 = \min\{1/1.67, 1/3.33\} = 0.33$; $\theta_2 = (0.5-0)/0.33 = 1.667$; $u_j = 0.5$; maka $\theta = \min\{0.33, 1.667; 0.5\} = 0.33$. Karena $\theta = \theta_1$ maka x_{51} masuk basis menggantikan R_2. Tabel iterasinya adalah sebagai berikut :

	x_{11}'	x_{21}'	x_{31}	x_{41}	x_{51}	x_{61}	R_1	s_1	s_2	R_2	s_3	RHS
z	-4.69	-4.09	0	-2.5	0	-9.59	-998.6	-1.39	0	-998.2	-1.79	3.79
x_{31}	-0.9	1.3	1	-1	0	0.3	-0.2	0.2	0	0.1	-0.1	0.1
s_2	-4.5	8.5	0	-6	0	1.5	-1	1	1	-0.5	0.5	0.5
x_{51}	-0.7	-2.1	0	1	1	-1.1	0.4	-0.4	0	0.3	-0.3	0.3

Pada iterasi ini, sudah optimum, karena semua konstanta pada baris ke nol bernilai negatif atau nol. Nilai tiap variabel separabel yang diperoleh adalah :

* $x_{11}' = 0$; $x_{11} = 0.5 - x_{11}' = 0.5$.
* $x_{21}' = 0$; $x_{21} = 0.5 - x_{21}' = 0.5$;
* $x_{31} = 0.1$
* $x_{41} = 0$
* $x_{51} = 0.3$

* $x_{61} = 0$;

Dari nilai variabel separabel di atas, kita dapat menentukan variabel untuk permasalahan non linier, dimana $x_j = x_{j1} + x_{j2}$. Variabel-variabel tersebut adalah :

* $x_1 = x_{11} + x_{12} = 0.5 + 0 = 0.5$
* $x_2 = x_{21} + x_{22} = 0.5 + 0 = 0.5$
* $x_3 = x_{31} + x_{32} = 0.1 + 0 = 0.1$
* $x_4 = x_{41} + x_{42} = 0 + 0 = 0$
* $x_5 = x_{51} + x_{52} = 0.3 + 0 = 0.3$
* $x_6 = x_{61} + x_{62} = 0 + 0 = 0$

- Langkah 3 : Mencek nilai z_Q, yaitu nilai fungsi tujuan dari permasalahan non linier.

Dari variabel-variabel yang telah ditentukan di atas, kita dapat menghitung nilai fungsi obyektifnya, yaitu:

$z_Q = 3(0.5–0.5^2)+5(0.5–0.5^2)+6(0.1–0.1^2)+9(0 – 0^2)+10(0.3 – 0.3^2)+10(0–0^2)$

$z_Q = 4.64$

Karena nilai z_Q tidak sama dengan nol, maka solusi optimal globalnya belum tercapai. Maka iterasi dilanjutkan ke langkah berikutnya.

- Langkah 4 : Pemeriksaan atau penetapan variabel-variabel.

Dari pasangan-pasangan variabel separabel di atas, dicari pasangan yang keduanya bernilai nol atau keduanya bernilai 0.5. Saat ini kita memiliki pasangan-pasangan variabel sebagai berikut: {0.5, 0; 0.5, 0; 0.1,0; 0,0; 0.3,0; 0,0}. Pasangan-

pasangan variabel tersebut tidak memenuhi syarat teknik pemeriksaan, sehingga tidak ada variabel yang ditetapkan.

- Langkah 5 : Melakukan evaluasi terhadap pasangan-pasangan variabel separabel.

  Dengan menggunakan kriteria evaluasi sebagai berikut :

$$f = \sum r_i(x_{i1} + x_{i2}),$$

dimana $r_i = 1$ jika $x_i = 0$; $r_i = -1$ jika $x_i = 1$; dan $r_i = 0$ untuk x_i lainnya. Untuk permasalahan kita, nilai f dapat dihitung sebagai berikut :

$f = (0 * 0.5) + (0 * 0.5) + (0 * 0.1) + (1 * 0) + (0 * 0.3) + (1 * 0)$

$f = 0.$

- Langkah 6 : Membebaskan semua variabel yang telah ditetapkan.

Karena nilai $f = 0$, maka kita menggunakan prosedur Branch and Bound untuk proses selanjutnya, yaitu dengan menggunakan pohon biner yang tiap nodenya mempunyai submasalah yang berbeda dengan fungsi tujuan berupa fungsi non linier. Sebelumnya semua variabel yaitu variabel $\{x_1, x_3, x_4, x_5, x_6\}$ yang nilainya telah ditetapkan, dibebaskan untuk diset ulang.

- Langkah 7 : Membuat node prosedur Branch and Bound.

Membuat, yang kita namakan node-0, mengawali langkah pertama prosedur Branch and Bound ini. Sub masalah untuk node-0 adalah submasalah 1.1. Submasalah tersebut kita selesaikan dengan pemrograman separabel, untuk memperoleh nilai tiap variabel separabelnya. Iterasi-iterasinya adalah sebagai berikut :

▸ Iterasi 0 :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	2998.5	3001.5	4997.5	5002.5	997	1003	1995.5	2004.5	2995	3005
R_1	-2	-2	6	6	-3	-3	4	4	1	1
s_2	5	5	3	3	-1	-1	-3	-3	2	2
R_2	5	5	-1	-1	4	4	-2	-2	2	2

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-3005	-2995	0	-1000	0	0	-1000	5000
-2	-2	1	-1	0	0	0	2
-1	-1	0	0	1	0	0	2
-1	-1	0	0	0	1	-1	3

▸ Iterasi 1 :

x_{41} = 1.5 dipilih menjadi variabel yang akan masuk basis. Maka dapat ditentukan $\theta_1 = \min\{6/7, 0/2\} = 0$. $\theta_2 = \infty$ dan $u_j = 0.5$. Maka $\theta = \min\{0,\infty,0.5\} = 0$. Karena $\theta = \theta_1$, dan s_2 meninggalkan basis, maka tabel baru menjadi:

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	4666	4669	-5	0	3498.3	3504.3	-1339	-1330	2161.3	2171.3
x_{21}	-0.333	-0.333	1	1	-0.5	-0.5	0.667	0.667	0.1667	0.1667
s_2	6	6	0	0	0.5	0.5	-5	-5	1.5	1.5
R_2	4.667	4.667	0	0	3.5	3.5	-1.333	-1.333	2.1667	2.1667

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-1337.5	-1327.5	-833.75	-166.25	0	0	-1000	3332.5
-0.333	-0.333	0.1667	-0.1667	0	0	0	0.333
0	0	-0.5	0.5	1	0	0	1
-1.333	-1.333	0.667	-0.667	0	1	-1	3.333

▸ Iterasi 2 :

x_{41} dipilih menjadi variabel yang akan masuk basis. Nilai kontanta pada kolom tersebut adalah $\alpha = \{9,-2\}$. Jika $\theta_1 = 6/9$, $\theta_2 = (0.5 - 0)/-(-2) = 0.25$, maka $\theta = \min\{0.67,0.25,0.5\} = 0.25$. Karena $\theta = \theta_2$, maka x_{41} masuk dalam basis, dan

x_{32} keluar. Kemudian x_{32} disubstitusikan pada batas atasnya ($0.5 - x_{32}'$), sehingga x_{32}' masuk dalam tabel operasi menggantikan x_{32}. Tabelnya menjadi :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-5	0	3109.2	3115.2	2551.3	2560.3	994	1004
x_{21}	0	0	1	1	-0.472	-0.472	0.389	0.389	0.25	0.25
x_{12}	1	1	0	0	0.083	0.083	-0.833	-0.833	0.25	0.25
R_2	0	0	0	0	3.11	3.11	2.55	2.55	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-1337.5	-1327.5	-444.67	-555.33	-778.17	0	-1000	2554.3
-0.333	-0.333	0.1389	-0.1389	0.056	0	0	0.389
0	0	-0.083	0.083	0.167	0	0	0.167
-1.333	-1.333	0.55	-0.55	-0.78	1	-1	2.556

- Iterasi 3 :

x_{52} menjadi calon variabel basis. Dengan $\alpha = \{1.5, -0.5\}$, maka $\theta_1 = 3.75/1.5 = 2.5$ dan $\theta_2 = (0.5 - 0.25)/-(-0.5) = 0.5$. Maka $\theta = \min\{2.5, 0.5, 0.5\} = 0.5$. Karena $\theta=\theta_2$ atau θ = batas atasnya, dipilih salah satu yang mempunyai operasi paling mudah, dalam hal ini dipilih θ = batas atasnya. Sehingga x_{52} tetap menjadi variabel non basis, tetapi yang masuk dalam operasi selanjutnya adalah batas atasnya, x_{52}', dengan substitusi $x_{52} = 0.5 - x_{52}'$. Tabel baru menjadi :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	6591.8	-6546	-6	0	5116.8	5125.8	2643.2	2653.2
x_{32}	0	0	-2.12	2.12	1	1	-0.824	-0.824	-0.53	-0.53
x_{12}	1	1	0.176	-0.176	0	0	-0.765	-0.765	0.294	0.294
R_2	0	0	6.59	-6.59	0	0	5.12	5.12	2.65	2.65

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-3536.4	-3526.4	471.56	-1471.5	-411.68	0	-1000	1821.4
0.7	0.7	-0.294	0.294	-0.118	0	0	0.235
-0.059	-0.059	-0.059	0.059	0.177	0	0	0.147
-3.53	-3.53	1.47	-1.47	-0.41	1	-1	1.824

▸ Iterasi 4 :

Dari tabel pada iterasi 3, dipilih x_{12} menjadi calon variabel basis, dengan $\alpha=\{1.75,-0.25\}$. Jika $\theta_1 = 3/1.75 = 1.7$ dan $\theta_2 = (0.5-0.5)/-(-0.25) = 0$, maka $\theta=\min\{1.7,0,0.5\} = 0$. Jadi $\theta = \theta_2$. Sama seperti iterasi 2, x_{12} masuk basis, dan $x_{41}=0.5 - x_{41}'$. Sehingga tabel yang baru adalah :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	0	-5	3106.8	-3113	2553.3	2562.3	995.25	1005.3
x_{21}	0	0	1	-1	-0.47	0.47	0.389	0.289	0.25	0.25
x_{12}	1	1	0	0	0.083	-0.083	-0.83	-0.83	0.25	0.25
R_2	0	0	0	0	3.11	-3.11	2.55	2.55	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-1339.2	-1329.2	-443.97	-556.03	-777.9	0	-100	997.38
-0.33	-0.33	0.139	-0.139	0.055	0	0	0.125
0	0	-0.083	0.083	0.167	0	0	0.125
-1.333	-1.333	0.55	-0.55	-0.78	1	-1	0.998

▸ Iterasi 5 :

x_{42} menjadi variabel yang akan masuk basis. $\alpha = \{7,-4\}$. $\theta_1 = 3/7$, $\theta_2=(0.5-0)/-(-4) = 0.125$, maka $\theta = \min\{0.43, 0.125, 0.5\} = 0.125$. Jadi $\theta = \theta_2$, sehingga melalui proses yang sama seperti iterasi sebelumnya, dengan mensubstitusikan $x_{12}=0.5-x_{12}'$, maka tabel yang baru adalah :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	0	-5	0	-6	1.259	10.259	-3.37	6.634
x_{21}	0	0	1	-1	0	0	0.78	0.78	0.4	0.4
x_{12}	1	1	0	0	0	0	-0.9	-0.9	0.22	0.22
x_{31}	0	0	0	0	1	-1	0.82	0.82	0.32	0.32

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-7.68	2.32	-998.76	-1.241	-1.187	-998.6	-1.38	-1.241
-0.54	-0.54	0.22	-0.22	-0.0625	0.152	-0.152	0.277
0.036	0.036	-0.098	0.098	0.1875	-0.03	0.03	0.098
-0.43	-0.43	0.179	-0.179	-0.25	0.32	-0.32	0.32

DAFTAR PUSTAKA

A. Ciriani Tito and C. Leachman, Robert : "Optimization in Industri: Mathematical Programming and Modelling Techniques in Practice". John Wiley & Sons Inc., 1993

Taha, Hamdi A. : "Operations Research : An Introduction" , 3 rd Edition, Macmillan Publishing Co. Inc., New York, 1982.

Hillier, Frederick S. and Lieberman, Gerald J. : "Introduction to Operations Research", Third Edition, Holden-Day Inc., Oakland, California, 1980.

Bazaraa, Mokhtar S. and Jarvis John J. : "Linier Programming and Network Flows", John Wiley & Sons, Inc., 1977.

Dimyati, Tjutju Tarliah dan Dimyati, Ahmad : "Operations Research : Model-Model Pengambilan Keputusan", Sinar Baru, Bandung, 1987.

Pala Windu, I Ketut : "Perancangan dan Pembuatan Perangkat Lunak Mixed Integer Linier Programming dengan Metode Branch and Bound", Jurusan Teknik Komputer, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya, 1995.

Shammas, Namir Clement : "What Every Borland® C++ 4 Programmer Should Know" , Sams Publishing, Indianapolis, Indiana, 1994.

Pappas, Chris H. and Murray, William H. : "Borland C++ Handbook", Third Edition,

Walnum, Clayton : "Object-Oriented Programming With Borland C++ 4", Que Corporation, 1994.

Pohl, Ira : "C++ for C Programmers", The Benjamin/Cummings Publishing Company, Inc., 1989.

Susanto, Kartono : "Pemrograman Berorientasi pada Objek dengan Borland C++" , Andi Offset, Yogyakarta, 1995.

LAMPIRAN

CONTOH KASUS

CONTOH KASUS

Suatu perusahaan mengambil keputusan untuk melakukan ekspansi usaha dengan membangun sebuah pabrik baru di salah satu tempat Los Angeles atau San Fransisco. Perusahaan itu juga mempertimbangkan untuk membangun sebuah gudang di kota yang dipilih untuk dibangun pabrik baru tersebut. *Net Present Value* (total keuntungan terhadap nilai uang) dari tiap-tiap alternatif, ditunjukkan pada tabel 2.1. dibawah. Sedangkan total dana yang tersedia sebesar 25 juta dollar. Dari tabel nanti kita dapat menganalisa data dan memutuskan pabrik serta gudang akan didirikan di kota apa.

Nomor Pilihan	Pertanyaan Ya atau Tidak	Variabel Keputusan	Net Present Value	Jumlah biaya
1	Pabrik di LA	y_1	7 juta dollar	20 juta dollar
2	Pabrik di SF	y_2	5 juta dollar	15 juta dollar
3	Gudang di LA	y_3	4 juta dollar	12 juta dollar
4	Gudang di SF	y_4	3 juta dollar	10 juta dollar

Dari tabel di atas dapat kita lihat bahwa antara pilihan nomor 1 dan 2 terdapat hubungan *mutually exclusive alternatives* , yang berarti jika pilihan nomor 1 jawabnya "ya" maka yang nomor 2 jawabnya "tidak". Ini dapat dirumuskan dengan :

$$y_1 + y_2 = 1$$

Sedangkan pilihan nomor 3 dan 4 jawabannya bergantung pada jawaban nomor 1 dan 2. Jika jawab nomor 1 adalah "ya" maka jawab nomor 3 "ya" dan jika

jawab nomor 2 "ya" maka jawab nomor 4 adalah "ya". Karena keputusan tempat untuk membangun gudang diambil setelah keputusan tempat untuk membangun pabrik ditetapkan. Disini terjadi *decisions contingent* , dan formulasi fungsi kendalanya dapat ditulis sebagai berikut :

$$y_3 - y_1 \leq 0$$

$$y_4 - y_2 \leq 0$$

Model matematikanya selengkapnya adalah :

Maksimumkan : $Z = 7y_1 + 5y_2 + 4y_3 + 3y_4$

dengan kendala :

$$\begin{aligned} 20y_1 + 15y_2 + 12y_3 + 10y_4 &\leq 25 \\ y_1 + y_2 &= 1 \\ -y_1 + y_3 &\leq 0 \\ -y_2 + y_4 &\leq 0 \end{aligned}$$

$y_i = 0$ atau 1 untuk i = 1,2,3,4.

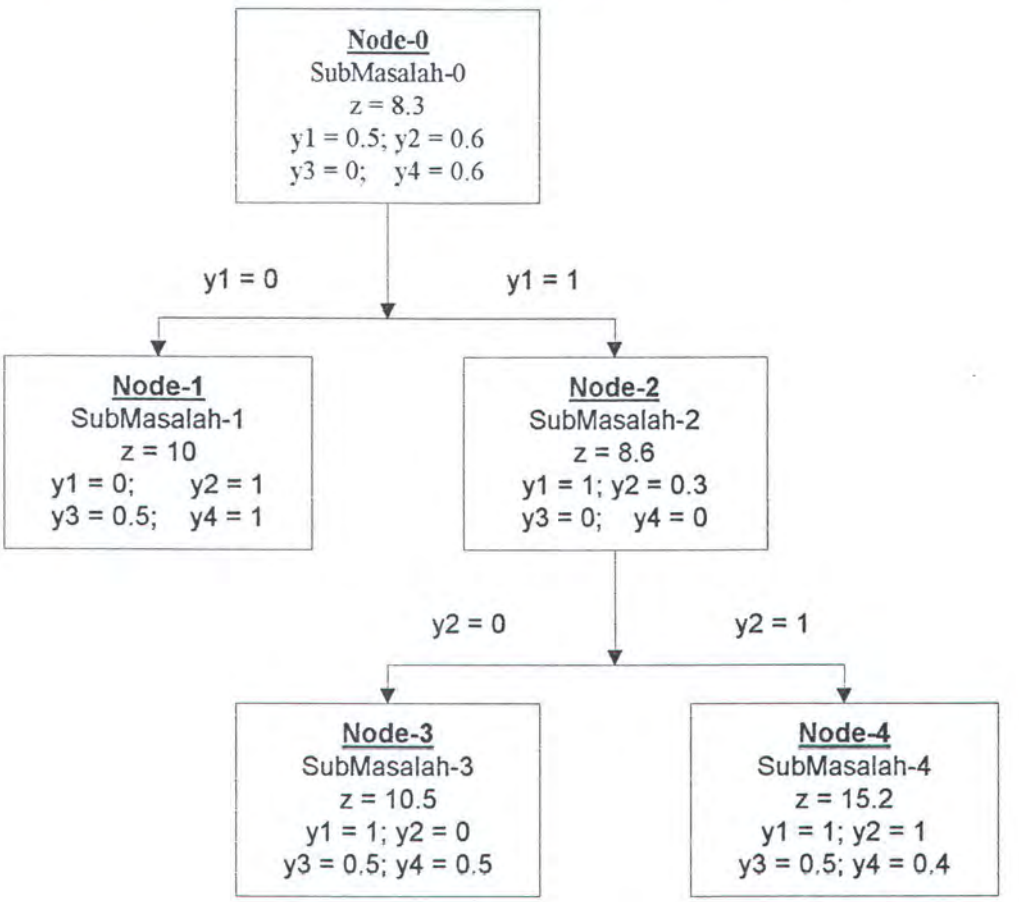
Dari model matematika di atas, diperoleh bentuk pendekatan non-liniernya adalah sebagai berikut :

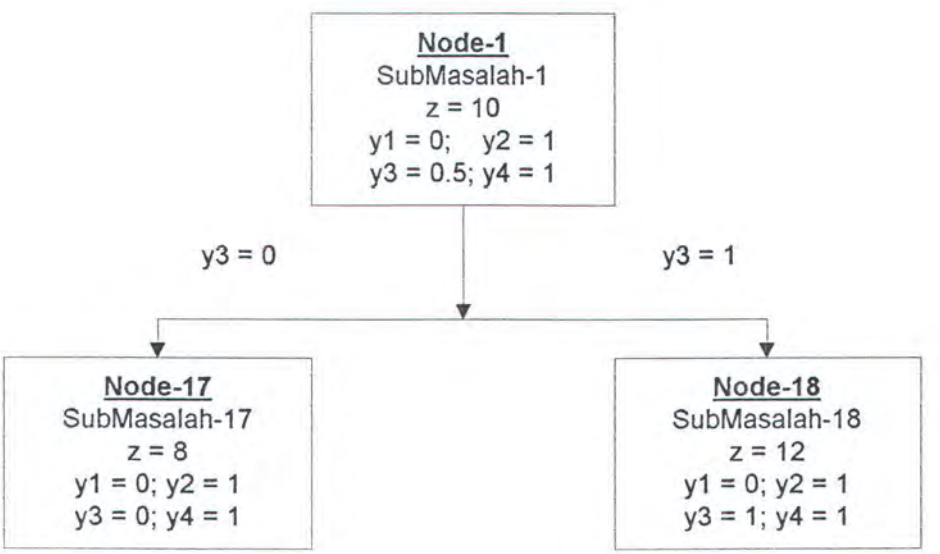
Maksimumkan : $Z_Q = 7y_1 - 7y_1^2 + 5y_2 - 5y_2^2 + 4y_3 - 4y_3^2 + 3y_4 - 3y_4^2$

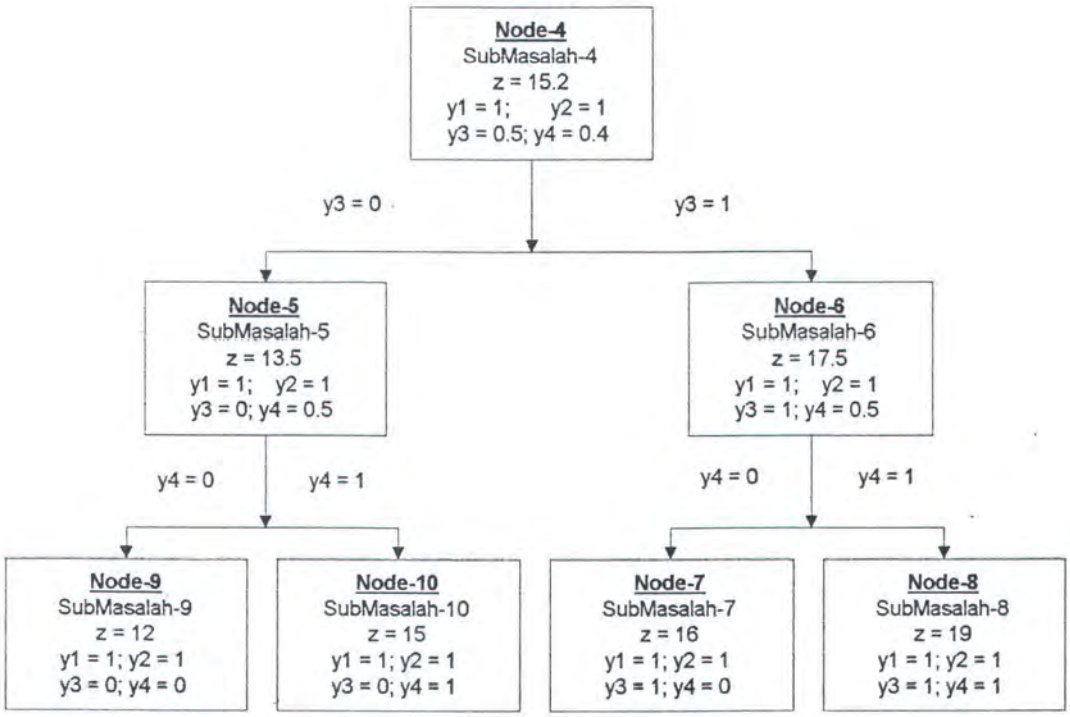
dengan kendala :

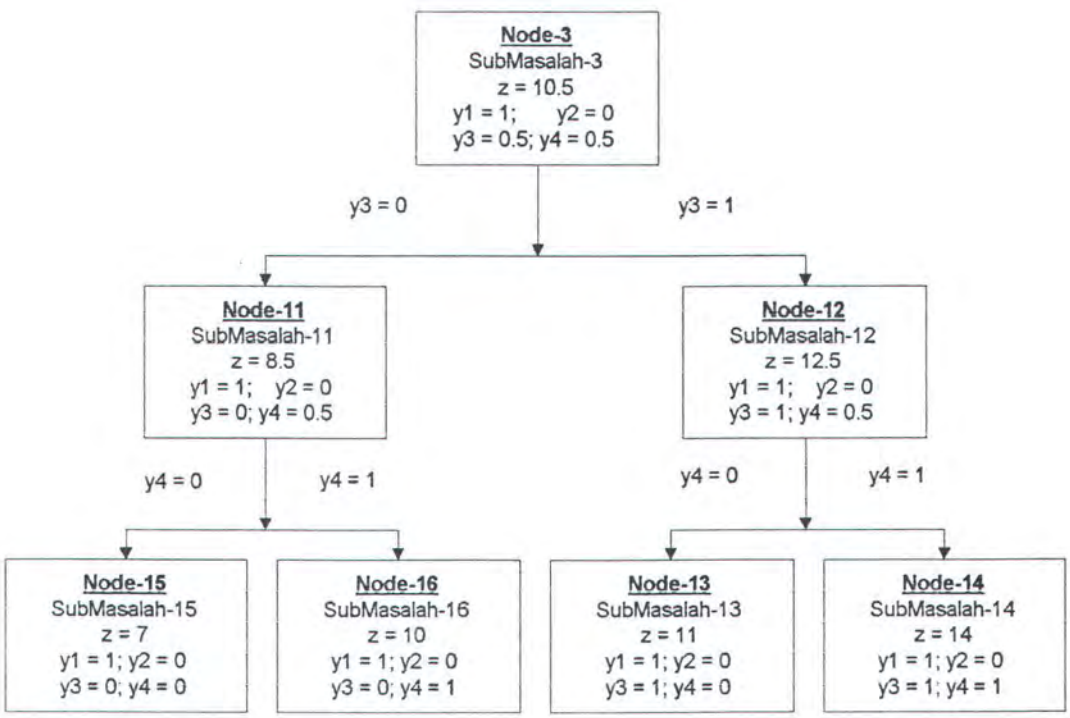
$$\begin{aligned} 20y_1 + 15y_2 + 12y_3 + 10y_4 &\leq 25 \\ y_1 + y_2 &= 1 \\ -y_1 + y_3 &\leq 0 \\ -y_2 + y_4 &\leq 0 \\ 0 \leq y_i &\leq 1 \end{aligned}$$

Model diatas diselesaikan dengan pemrograman separabel. Secara lengkap penyelesaiannya dapat digambarkan dalam diagram pohon biner berikut :









Dari semua node diatas yang memenuhi syarat kendala adalah node-15 dan node-17. Nilai optimal untuk permasalahan ini adalah nilai paling maksimum

diantara nilai kedua node tersebut. Maka nilai optimal untuk permasalahan ini adalah 8, dengan nilai tiap variabel $y_1 = 0$; $y_2 = 1$; $y_3 = 0$; $y_4 = 1$.

Dengan melihat hasil tersebut kita dapat menyimpulkan, bahwa jika pengusaha tersebut ingin melakukan ekspansi usaha di kota lain dengan dana 25 juta dollar, untuk hasil yang paling menguntungkan, pengusaha tersebut dapat mendirikan pabrik baru dan gudangnya sekaligus di San Fransisco.

▸ Iterasi 6 :

x_{21} menjadi calon variabel basis. $\alpha = \{-0.8, 0.3\}$, maka $\theta_1 = 0.125/0.25 = 0.5$ dan $\theta_2 = (0.5 - 2.125)/-(-1.75) = -0.93$, karena $\theta_2 < 0$, maka diabaikan atau $\theta_2 = \infty$. Sehingga $\theta = \min\{0.5, \infty, 0.5\} = 0.5$. Dipilih θ = batas atasnya, karena lebih mudah operasinya. Tabel yang baru menjadi :

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-13.2	8.2	0	-6	-9	0	-8.667	1.333
x_{42}	0	0	1.29	-1.29	0	0	1	1	0.52	0.52
x_{12}	1	1	1.16	-1.16	0	0	0	0	0.69	0.69
x_{31}	0	0	-1.06	1.06	1	-1	0	0	-0.1	-0.1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-0.6	9.397	-1001.7	1.7	-0.36	-1000.6	0.62	-4.897
-0.69	-0.69	0.29	-0.29	-0.08	0.195	-0.195	0.356
-0.59	-0.59	0.16	-0.16	0.11	0.15	-0.15	0.4195
0.14	0.14	-0.06	0.06	-0.18	0.16	-0.16	0.029

▸ Iterasi 7

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	13	-16	5.4	-10.4	0	-6	-9	0	2.4	12.4
x_{42}	-1.2	1.2	-0.08	0.08	0	0	1	1	-0.3	-0.3
x_{62}	-1.7	1.7	-1.98	1.98	0	0	0	0	-1.2	-1.2
x_{31}	0.24	-0.24	-0.78	0.78	1	-1	0	0	0.06	0.06

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-10	0	-999.13	-0.873	1.48	-998.23	-1.77	-6.1863
0	0	0.1	-0.1	-0.22	0.02	-0.02	0.45
1	1	-0.3	0.3	-0.2	-0.25	0.25	0.14
0	0	-0.02	0.02	-0.16	0.2	-0.2	0.0098

▸ Iterasi 8

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	0	-3	48.83	-53.83	-55.4	49.4	-9	0	-0.875	9.125
x_{42}	0	0	-4	4	5	-5	1	1	-1.11	-1.11
x_{62}	0	0	-7.67	7.67	7.25	-7.25	0	0	-0.75	-0.75
x_{11}	1	-1	-3.33	3.33	4.25	-4.25	0	0	0.25	0.25

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-10	0	-998.04	-1.96	10.2	-1009.1	9.1	-6.73
0	0	1.11	-1.11	-1	1	-1	0.5
1	1	-0.42	0.42	-1.33	1.167	-1.167	0.208
0	0	-0.083	0.083	-0.67	0.83	-0.83	0.042

- Iterasi 9

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	0	-3	9.33	-14.3	-6	0	0.875	-9.875	-0.875	9.125
x_{32}'	0	0	0.8	-0.8	-1	1	-0.2	0.2	0	0
x_{62}	0	0	-1.867	1.867	0	0	-1.45	1.45	-0.75	-0.75
x_{11}	1	-1	0.0667	-0.067	0	0	-0.85	0.85	0.25	0.25

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-10	0	-998	-1.96	0.29	-999.2	-0.79	-6.73
0	0	0	0	0.2	-0.2	0.2	0
1	1	-0.42	0.42	0.12	-0.283	0.283	0.2083
0	0	-0.083	0.083	0.183	-0.0167	0.0167	0.042

- Iterasi 10

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	0	-3	0	-5	5.67	-11.67	3.21	-12.21	-0.875	9.125
x_{21}	0	0	1	-1	-1.25	1.25	-0.25	0.25	0	0
x_{62}	0	0	0	0	-2.33	2.33	-1.92	1.92	-0.75	-0.75
x_{11}	1	-1	0	0	0.083	-0.083	-0.083	0.083	0.25	0.25

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-10	0	-998	-1.96	-2.04	-996.86	-3.125	-6.73
0	0	0	0	0.25	-0.25	0.25	0
1	1	-0.42	0.42	0.583	-0.75	0.75	0.21
0	0	-0.083	0.083	0.167	0	0	0.042

- Iterasi 11

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-36.5	33.5	0	-5	2.625	-8.625	33.625	-42.63	-10	0
x_{21}	-1.11	1.11	1	-1	-1.25	1.25	-0.25	0.25	0	0
x_{62}	3	-3	0	0	-2.083	2.083	-4.42	4.42	0	0
x_{52}	4	-4	0	0	0.333	-0.333	-3.33	3.33	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-2.39	-7.61	-1000	0.075	0.123	-1002.6	2.585	-9.52
-0.06	0.06	0.06	-0.04	0.189	-0.2	0.2	0.0094
-0.23	0.23	0.15	-0.15	-0.25	0.17	-0.17	0.038
-0.75	0.75	0.17	-0.17	-0.15	0.57	-0.57	0.2925

- Iterasi 12

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-13.66	10.66	0	-5	-13.24	7.235	0	-9	-10	0
x_{21}	-0.17	0.17	1	-1	-1.132	1.132	0	0	0	0
x_{41}	-0.68	0.68	0	0	0.5	-0.5	1	-1	0	0
x_{52}	1.74	-1.74	0	0	2	-2	0	0	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-2.39	-7.61	-1000	0.075	0.123	-1002.6	2.585	-9.52
-0.06	0.06	0.04	-0.04	0.189	-0.2	0.2	0.0094
-0.23	0.23	0.15	-0.15	-0.25	0.17	-0.17	0.038
-0.75	0.75	0.17	-0.17	-0.15	0.57	-0.57	0.2925

- Iterasi 13

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	0	-5	-20.64	14.64	-15.7	6.7	-10	0
x_{21}	0	0	1	-1	-1.25	1.25	-0.25	0.25	0	0
x_{12}'	-1	1	0	0	0.694	-0.694	1.472	-1.472	0	0
x_{52}	0	0	0	0	3.11	-3.11	2.56	-2.56	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
1.167	-11.7	-1002.4	2.44	3.97	-1005.3	5.25	-10.11
0	0	0	0	0.25	-0.25	0.25	0
-0.33	0.33	0.22	-0.22	-0.36	0.25	-0.25	0.055
-1.33	1.33	0.56	-0.56	-0.778	1	-1	0.389

- Iterasi 14

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-11.71	6.71	-6	0	-12.77	3.767	-10	0
x_{32}'	0	0	0.8	-0.8	-1	1	-0.2	0.2	0	0
x_{12}'	-1	1	0.56	-0.56	0	0	1.33	-1.33	0	0
x_{52}	0	0	2.5	-2.5	0	0	1.93	-1.93	1	1

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
1.667	-11.17	-1002.4	2.44	1.044	-1002.3	2.32	-10.11
0	0	0	0	0.2	-0.2	0.2	0
-0.33	0.33	0.22	-0.22	-0.22	0.11	-0.11	0.0556
-1.33	1.33	0.56	-0.56	-0.156	0.378	-0.378	0.389

- Iterasi 15

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-5	0	-6	0	-7.55	-1.45	-7.3	-2.7
x_{32}'	0	0	0	0	-1	1	-0.82	0.82	-0.32	0.32
x_{12}'	-1	1	0	0	0	0	0.9	-0.9	-0.22	0.22
x_{22}'	0	0	-1	1	0	0	-0.78	0.78	-0.4	0.4

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-2.4	-7.57	-1000.9	0.95	0.625	-1001.3	1.304	-10.41
0.4	-0.4	-0.18	0.18	0.25	-0.32	0.32	0.0357
-0.04	0.04	0.098	-0.098	-0.1875	0.027	-0.027	0.0804
0.54	-0.54	-0.22	0.22	0.0625	-0.15	0.15	0.045

- Iterasi 16

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-5	0	-1.94	-4.056	-4.22	-4.77	-5.99	-4
s_3	0	0	0	0	-3.11	3.11	-2.56	2.56	-1	1
x_{12}'	-1	1	0	0	-0.083	0.083	0.83	-0.83	-0.25	0.25
x_{22}'	0	0	-1	1	0.47	-0.47	-0.389	0.389	-0.25	0.25

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-4.167	-5.83	-1000.2	0.22	-0.389	-1000	0	-10.56
1.33	-1.33	-0.56	0.56	0.78	-1	1	0.111
0	0	0.083	-0.083	-0.167	0	0	0.083
0.33	-0.33	-0.139	0.139	-0.056	0	0	0.0278

- Iterasi 17

	x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	x_{41}	x_{42}	x_{51}	x_{52}
z	-3	0	-3.4	-1.6	-2.7	-3.3	-3.6	-5.4	-5.6	-4.4
s_3	0	0	4	-4	-5	5	-1	1	0	0
x_{12}'	-1	1	-0.6	0.6	0.2	-0.2	0.6	-0.6	-0.4	0.4
s_1	0	0	-7.2	7.2	3.4	-3.4	-2.8	2.8	-1.8	1.8

x_{61}	x_{62}	R_1	s_1	s_2	R_2	s_3	rhs
-4.7	-5.3	-1000	0	-0.3	-1000	0	-10.6
0	0	0	0	1	-1	1	0
0.2	-0.2	0	0	-0.2	0	0	0.1
2.4	-2.4	-1	1	-0.4	0	0	0.2

Pada iterasi ini, sudah optimum, karena semua konstanta (c_j - z) pada baris ke nol bernilai negatif atau nol. Nilai tiap variabel separabel yang diperoleh adalah :

* $x_{11} = 0$
* $x_{12}' = 0.1$; $x_{12} = 0.5 - x_{12}' = 0.5 - 0.1 = 0.4$;
* $x_{21} = 0$;
* $x_{22}' = 0$; $x_{22} = 0.5 - x_{22}' = 0.5$;
* $x_{31} = 0$
* $x_{32}' = 0$; $x_{32} = 0.5 - x_{32}' = 0.5$
* $x_{41} = 0$;
* $x_{42}' = 0$; $x_{42} = 0.5 - x_{42}' = 0.5$
* $x_{51} = 0$
* $x_{52}' = 0$; $x_{52} = 0.5 - x_{52}' = 0.5$
* $x_{61} = 0$
* $x_{62}' = 0$; $x_{62} = 0.5 - x_{62}' = 0.5$

Dari nilai variabel separabel di atas, kita dapat menentukan variabel untuk permasalahan non linier, dimana $x_j = x_{j1} + x_{j2}$. Variabel-variabel tersebut adalah :

* $x_1 = x_{11} + x_{12} = 0 + 0.4 = 0.4$
* $x_2 = x_{21} + x_{22} = 0 + 0.5 = 0.5$

* $x_3 = x_{31} + x_{32} = 0 + 0.5 = 0.5$

* $x_4 = x_{41} + x_{42} = 0 + 0.5 = 0.5$

* $x_5 = x_{51} + x_{52} = 0 + 0.5 = 0.5$

* $x_6 = x_{61} + x_{62} = 0 + 0.5 = 0.5$

Dari variabel-variabel yang telah ditentukan di atas, kita dapat menghitung nilai fungsi obyektifnya, yaitu:

$z_Q=3(0.4-0.4^2)+5(0.5-0.5^2)+6(0.5-0.5^2)+9(0.5-0.5^2)+10(0.5-0.5^2)+10(0.5-0.5^2)$

$z_Q = 0.72 + 1.25 + 1.5 + 2.25 + 2.5 + 2.5$

$z_Q = 10.72$

Selanjutnya kita dapat membuat node baru dengan $z_Q = 1.75$ dan $x_j=\{0.5,0.5,0.5, 0.5, 0.5\}$ untuk j = 1,2,3,4,5.

Node-0
Submasalah 1.2
$z_Q = 10.72$
$x_1 = 0.4$; $x_2 = 0.5$;
$x_3 = 0.5$; $x_4 = 0.5$;
$x_5 = 0.5$; $x_6 = 0.5$;

Karena nilai z_Q tidak sama dengan nol, maka iterasi dilanjutkan ke langkah berikutnya.

- Langkah 8 : Mengevaluasi hasil dengan kriteria evaluasi f.

  Nilai f dapat dihitung sebagai berikut :

$f = (0 * 0.4) + (0 * 0.5) + (0 * 0.5) + (0 * 0.5) + (0 * 0.5) + (0 * 0.5)$

$f = 0.$

Karena f tidak mencapai nilai minimum atau negatif, maka kita ulangi dengan teknik pemeriksaan dengan setting yang berbeda.

- Langkah 9 : Merubah nilai variabel separabel pada batas atasnya.

Pada langkah 1, kita set variabel separabelnya pada batas bawah. Sekarang kita set pada batas atasnya, yaitu $x_{11}=0.5$; $x_{21}=0.5$; $x_{31}=0.5$; $x_{41}=0.5$; $x_{51}=0.5$; $x_{61}=0.5$. Maka submasalah 1.2 dirubah menjadi submasalah baru, submasalah 3.2, yaitu:

Min: $z_Q^* = -3/2x_{12} - 5/2x_{22} - 3x_{32} - 9/2x_{42} - 5x_{52} - 5x_{62} + 10.75$

kendala : $2x_{12} - 6x_{22} + 3x_{32} - 4x_{42} - x_{52} + 2x_{62} \leq 0$

$-5x_{12} - 3x_{22} + x_{32} + 3x_{42} - 2x_{52} + x_{62} \geq 0.5$

$-5x_{12} + x_{22} - 4x_{32} + 2x_{42} - 2x_{52} + x_{62} \leq 0.5$

$0 \leq x_{j1} \leq ½$

- Langkah 10 : Selesaikan permasalahan pemrograman separabel di atas.

Dengan metode Simplex Bounded seperti pada langkah-langkah sebelumnya, kita selesaikan tiap iterasi dari pemrograman separabel tersebut.

- Iterasi 0 :

	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	s_1	R_1	s_2	s_3	rhs
z	-4998.5	-2997.5	1003	3004.5	-1995	1005	0	0	-1000	0	5010.8
s_1	2	-6	3	-4	-1	2	1	0	0	0	0
R_1	-5	-3	1	3	-2	1	0	1	-1	0	0.5
s_3	-5	1	-4	2	-2	1	0	0	0	1	0.5

- Iterasi 1 :

	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	s_1	R_1	s_2	s_3	rhs
z	-3496.3	-7504.3	3256.4	0	-2746.1	2507.3	-751.1	0	-1000	0	124.4
x_{42}	-0.5	1.5	-0.75	1	0.25	-0.5	0.25	0	0	0	0.125
R_1	-3.5	-7.5	3.25	0	-2.75	2.5	-0.75	1	-1	0	0.125
s_3	-4	-2	-2.5	0	-2.5	2	-0.5	0	0	1	0.25

- Iterasi 2 :

	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	s_1	R_1	s_2	s_3	rhs
z	10.62	10.46	0	0	9.27	2.35	0.35	-1001.9	1.96	0	-0.81
x_{42}	-1.31	-0.23	0	1	-0.39	0.077	0.077	0.23	-0.23	0	0.15
x_{32}	-1.08	-2.3	1	0	-0.85	0.77	-0.23	0.31	-0.31	0	0.04
s_3	-6.69	-7.77	0	0	-4.62	3.92	-1.077	0.77	-0.77	1	0.35

- Iterasi 3 :

	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	s_1'	R_1	s_2	s_3'	rhs
z	0	-1.86	0	0	1.95	8.57	-1.36	-1000.7	0.74	-1.59	-1.05
x_{42}	0	1.287	0	1	0.52	-0.69	0.29	0.08	-0.08	0.195	0.184
x_{32}	0	-1.06	1	0	-0.1	0.14	-0.06	0.184	-0.184	0.161	0.063
x_{12}	1	1.16	0	0	0.69	-0.59	0.16	-0.12	0.12	0.15	0.023

- Iterasi 4 :

	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	s_1'	R_1	s_2	s_3'	rhs
z	0	63.8	-62.1	0	8.4	0	2021	-1012.2	12.17	-11.58	-4.98
x_{42}	0	-4	5	1	4.44	0	3.89	1	-1	1	0.5
x_{62}	0	-7.67	7.25	0	-0.75	1	-0.42	1.33	-1.33	1.17	0.46
x_{12}	1	-3.33	4.25	0	0.25	0	-0.08	0.67	-0.67	0.83	0.29

- Iterasi 5 :

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}	s_1'	R_1	s_2	s_3'	rhs
z	0	0	17.67	-15.9	8.4	0	2.21	-996.2	-3.79	4.38	-4.98
x_{22}	0	1	-1.25	0.25	-1.11	0	-9.71	-0.25	0.25	-0.25	0
x_{62}	0	0	-2.33	1.92	-0.75	1	-0.42	-0.58	0.58	-0.75	0.46
x_{12}	1	0	0.08	0.83	0.25	0	-0.08	-0.17	0.17	0	0.29

- Iterasi 6 :

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}'	s_1'	R_1	s_2	s_3'	rhs
z	0	0	0	-1.45	2.7	-7.6	-0.95	-1000.6	0.63	-1.3	-5.295
x_{22}	0	1	0	-0.78	0.4	0.54	0.22	0.063	-0.06	0.15	0.022
x_{32}	0	0	1	-0.82	0.32	0.43	0.18	0.25	-0.25	0.32	0.018
x_{12}	1	0	0	0.9	0.22	-0.04	-0.1	-0.19	0.19	-0.03	0.29

- Iterasi 7:

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}'	s_1'	R_1	s_2	s_3'	rhs
z	0	0	-8.4	5.44	0	-11.17	-2.44	-1002.7	2.72	-4	-5.44
x_{22}	0	1	-1.25	0.25	0	1.11	-8.33	-0.25	0.25	-0.25	3.47
x_{52}	0	0	3.11	-2.56	1	1.33	0.56	0.78	-0.78	1	0.056
x_{12}	1	0	-0.69	1.47	0	-0.33	-0.22	-0.36	0.36	-0.25	0.278

- Iterasi 8:

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}'	s_1'	R_1	s_2	s_3'	rhs
z	0	-21.8	18.8	0	0	-11.17	-2.44	-997.3	-2.7	1.44	-5.44
x_{42}'	0	4	-5	1	0	4.44	-3.3	-1	1	-1	1.39
x_{52}	0	10.2	-9.7	0	1	1.33	0.56	-1.78	1.78	-1.56	0.056
x_{12}	1	-5.89	6.67	0	0	-0.33	-0.22	1.11	-1.11	1.22	0.278

- Iterasi 9:

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}'	s_1'	R_1	s_2	s_3'	rhs
z	-2.83	-5.14	0	0	0	-10.23	-1.82	-1000.4	0.42	-2.01	-6.23
x_{42}'	0.75	-0.42	0	1	0	-0.25	-0.17	-0.17	0.17	-0.08	0.21
x_{52}	1.45	1.683	0	0	1	0.85	0.23	-0.17	0.17	0.22	0.46
x_{32}	0.15	-0.88	1	0	0	-0.05	-0.03	0.17	-0.17	0.18	0.042

- Iterasi 10:

	x_{12}	x_{22}	x_{32}	x_{42}'	x_{52}	x_{62}'	s_1'	R_1	s_2'	s_3'	rhs
z	-2.83	-5.14	0	0	0	-10.23	-1.82	-1000.4	-0.42	-2.01	-6.44
x_{42}'	0.75	-0.42	0	1	0	-0.25	-0.17	-0.17	-0.17	-0.08	0.125
x_{52}	1.45	1.683	0	0	1	0.85	0.23	-0.17	-0.17	0.22	0.375
x_{32}	0.15	-0.88	1	0	0	-0.05	-0.03	0.17	0.17	0.18	0.125

Jadi diperoleh : $x_{12} = 0$; $x_{22} = 0$; $x_{32} = 0.125$; $x_{42} = 0.375$; $x_{52} = 0.375$; $x_{62} = 0.5$;

Dengan setting variabel pada langkah sebelumnya, yaitu $x_{11} = 0.5$; $x_{21} = 0.5$; $x_{31}=0.5$; $x_{41} = 0.5$; $x_{51} = 0.5$; $x_{61} = 0.5$, kita dapat menentukan bahwa $x_1 = 0.5$; $x_2=0.5$; $x_3 = 0.625$; $x_4 = 0.875$; $x_5 = 0.875$; $x_6 = 1$.

- Langkah 11 : Periksa nilai z_Q

$z_Q=3(0.5-0.5^2)+5(0.5-0.5^2)+6(0.625-0.625^2)+9(0.875-0.875^2)+$

$10(0.875-0.875^2)+10(1-1^2)$

$z_Q = 0.75 + 1.25 + 1.40625 + 0.984375 + 1.09375 + 0$

$z_Q = 5.48375$

- Langkah 12 : Teknik Pemeriksaan atau penetapan variabel

Dari pasangan-pasangan variabel separabel diperiksa, dan kita dapatkan pasangan $\{x_{61} = 0.5; x_{62} = 0.5\}$. Maka kita tetapkan $x_6 = 1$. Permasalahan non linier yang baru, yaitu sub masalah 3.1 :

Min : $z_Q = 3(x_1-x_1^2)+5(x_2-x_2^2)+6(x_3-x_3^2)+9(x_4-x_4^2)+10(x_5-x_5^2)$

dengan kendala : $-2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 \geq 4$

$-5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 \geq -3$

$5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 \geq 4$

$0 \leq x_j \leq 1$

Dengan pemrograman separabel, permasalahan di atas dapat didekati dengan permasalahan baru, kita beri nama subproblem 3.2, yaitu :

Min: $z_Q^*=3/2(x_{11}-x_{12})+5/2(x_{21}-x_{22})+3(x_{31}-x_{32})+9/2(x_{41}-x_{42})+5(x_{51}-x_{52})$

kendala : $-2x_{11}-2x_{12}+6x_{21}+6x_{22}-3x_{31}-3x_{32}+4x_{41}+4x_{42}+ x_{51}+ x_{52} \geq 4$

$-5x_{11}-5x_{12}-3x_{21}-3x_{22}+ x_{31}+ x_{32}+3x_{41}+3x_{42}-2x_{51}-2x_{52} \geq -3$

$5x_{11}+5x_{12}- x_{21}- x_{22}+4x_{31}+4x_{32}-2x_{41}-2x_{42}+2x_{51}+2x_{52} \geq 4$

$0 \leq x_{j1}, x_{j2} \leq ½$

- Langkah 13 : Mengevaluasi hasil dengan kriteria evaluasi f.

$f = 0 + 0 + 0 + 0 + 0 - 1$

$f = -1$

Karena f mencapai nilai minimum, maka mengulangi teknik pemeriksaan atau metode penetapan variabel.

- Langkah 14 : Merubah nilai variabel separabel pada batas bawah

Dari pasangan-pasangan variabel yang tersisa atau yang belum ditetapkan di set ulang pada batas bawah, yang berarti bahwa : $x_{12}=0$; $x_{22}=0$; $x_{32}=0$; $x_{42}=0$; $x_{52}=0$. Maka sub masalah 3.2 menjadi sub masalah baru , sub masalah 3.2.1 :

Min: $z_Q^*=3/2x_{11}+5/2x_{21}+3x_{31}+9/2x_{41}+5x_{51}$

kendala : $-2x_{11}+6x_{21}-3x_{31}+4x_{41}+ x_{51} \geq 4$

$-5x_{11}-3x_{21}+ x_{31} +3x_{41}-2x_{51} \geq -3$

$5x_{11}- x_{21} +4x_{31}-2x_{41}+2x_{51} \geq 4$

$0 \leq x_{j1}, x_{j2} \leq ½$

- Langkah 15 : Menjalankan pemrograman separabel untuk masalah tersebut.

☙ Iterasi 0:

	X11	X21	X31	X41	X51	R1	S1	S2	R2	S3	RHS
Z	2998.5	4997.5	997	1995.5	2995	0	-1000	0	0	-1000	8000
R1	-2	6	-3	4	1	1	-1	0	0	0	4
S2	5	3	-1	-3	2	0	0	1	0	0	3
R2	5	-1	4	-1	2	0	0	0	1	-1	3

☙ Iterasi 1:

	X11	X21	X31	X41	X51	R1	S1	S2	R2	S3	RHS
Z	2998.5	-4997.5	997	1995.5	2995	0	-1000	0	0	-1000	5501.25
R1	-2	-6	-3	4	1	1	-1	0	0	0	1
S2	5	-3	-1	-3	2	0	0	1	0	0	1.5
R2	5	1	4	-1	2	0	0	0	1	-1	4.5

☙ Iterasi 2:

	X11	X21	X31	X41	X51	R1	S1	S2	R2	S3	RHS
Z	0	-3198.4	1596.7	3794.5	1795.6	0	-1000	-599.7	0	-1000	4601.7
R1	0	-7.2	-3.4	2.8	1.8	1	-1	0.4	0	0	1.6
S2	1	-0.6	-0.2	-0.6	0.4	0	0	0.2	0	0	0.3
R2	0	4	5	1	0	0	0	-1	1	-1	3

- Iterasi 3:

	X_{11}'	X_{21}	X_{31}	X_{41}	X_{51}	R_1	S_1	S_2	R_2	S_3	RHS
Z	-6324.33	-699.3	331.8	0	-1000	0	-1000	665.2	0	-1000	3336.8
R_1	-4.67	-10	-4.33	0	-1	1	-1	1.33	1	-1	0.67
S_2	1.67	1	0.33	1	0	0	0	-0.33	0	0	0.33
R_2	-1.67	3	4.67	0	0	0	0	-0.67	0	0	2.67

- Iterasi 4:

	X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	R_1	S_1	S_2	R_2	S_3	RHS
Z	-819.36	4803.36	5443.59	0	0	-1179.64	179.64	-907.68	0	-1000	2550.41
R_1	-1.27	-2.73	-1.18	0	1	0.27	-0.27	0.36	1	-1	0.18
S_2	0.82	-0.82	-0.45	1	0	0.18	-0.18	-0.09	0	0	0.45
R_2	-0.82	4.82	5.54	0	0	-0.18	0.18	-0.91	0	0	2.55

- Iterasi 5:

	X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	R_1	S_1	S_2	R_2	S_3	RHS
Z	8979.1	-4995.1	0	-11975.9	0	997.8	-1997.8	-1996.4	0	-1000	2006.05
R_1	-3.4	-0.6	0	2.6	1	-0.2	0.2	0.6	1	-1	0.3
S_2	-1.8	1.8	1	2.2	0	-0.4	0.4	0.2	0	0	0.1
R_2	9	-5	0	-1.2	0	2	-2	-2	0	0	2

- Iterasi 6:

	X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	R_1	S_1	S_2	R_2	S_3	RHS
Z	0	-6579.65	0	-5109.53	-2640.9	469.62	-1469.62	-411.85	0	-1000	1447.87
R_1	1	0.176	0	-0.765	0.294	0.059	-0.059	-0.176	1	-1	0.059
S_2	0	2.12	1	0.824	0.53	-0.29	0.29	-0.12	0	0	0.21
R_2	0	-6.5	0	-5.12	-2.65	1.47	-1.47	-0.41	0	0	1.47

- Iterasi 7:

	X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	R_1	S_1	S_2	R_2	S_3	RHS
Z	0	-6579.65	0	-5109.53	-2640.91	-469.62	-1469.62	-411.85	0	-1000	1243.06
R_1	1	0.176	0	-0.765	0.294	-0.059	-0.059	-0.176	1	-1	0.0294
S_2	0	2.12	1	0.824	0.53	0.29	0.29	-0.12	0	0	0.35
R_2	0	-6.5	0	-5.12	-2.65	-1.47	-1.47	-0.41	0	0	0.74

Diperoleh : $x_{11} = 0.5 - 0.03 = 0.47$; $x_{21} = 0.5$; $x_{31} = 0.35$; $x_{41} = 0.5$; $x_{51} = 0.5$;

- Langkah 16 : Periksa nilai z_Q

$z_Q = 0.7473 + 1.25 + 1.365 + 2.25 + 2.5$

$z_Q = 8.1123$

z_Q tidak sama dengan nol.

- Langkah 17 : Evaluasi dengan fungsi f.

  $f = (0 * 0.47) + (0 * 0.5) + (0 * 0.35) + (0 * 0.25) + (0 * 0.25)$

  $f = 0$

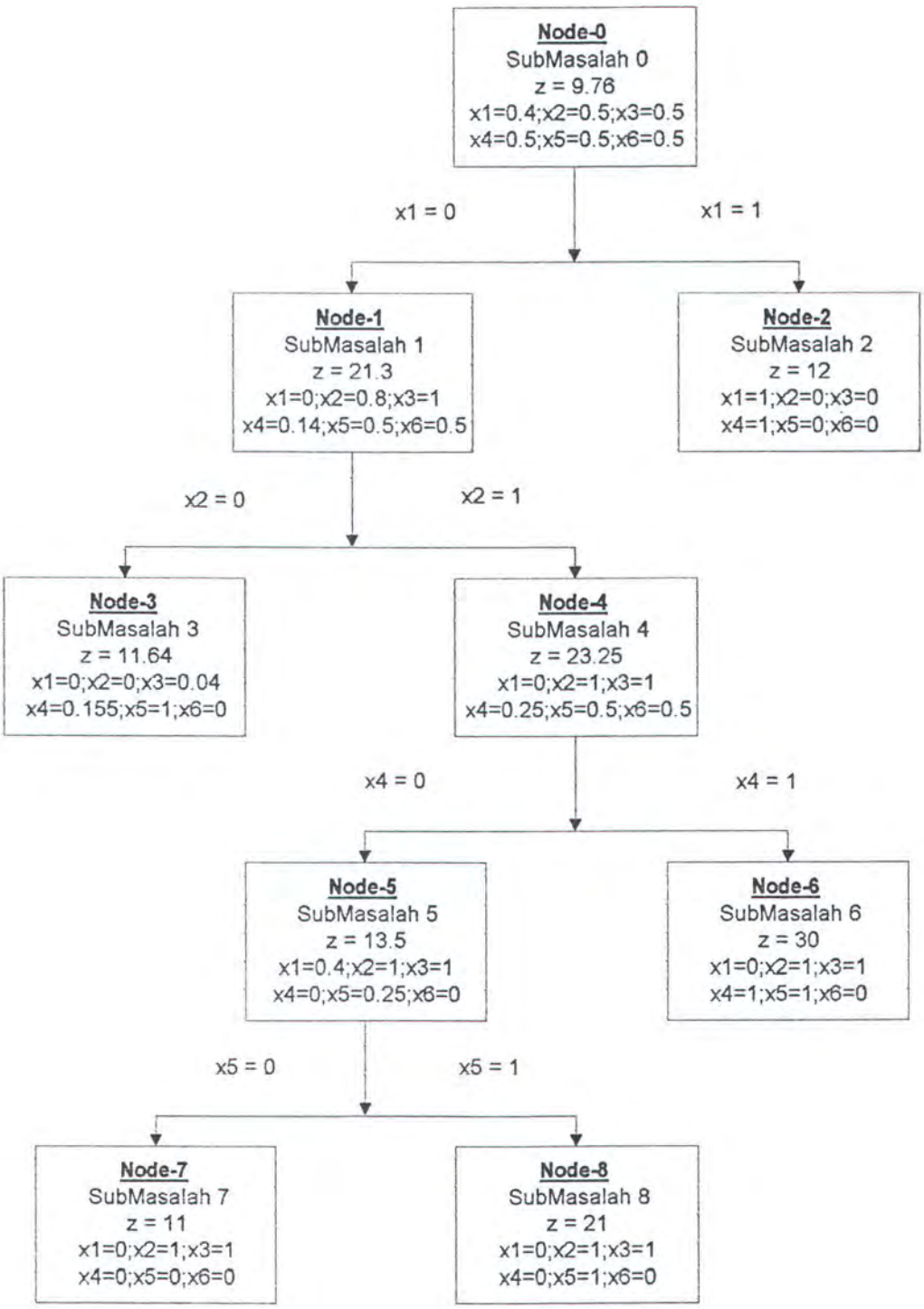
  Karena f = 0, maka kita lakukan kembali prosedur Branch and Bound

- Langkah 18 : Membuat node-1 yang merupakan anak cabang node-0 dan seterusnya mengulang langkah-langkah sebelumnya kita dapatkan beberapa node untuk pohon binernya, seperti pada gambar 4.1.

- Pada node-2 nilai $z_Q = 0$, yang berarti bahwa semua variabelnya sudah mencapai nilai bulat 0 atau 1. Selanjutnya dihitung nilai z dari nilai variabel yang didapat dan diperiksa apakah semua variabelnya memenuhi syarat fungsi kendalanya.

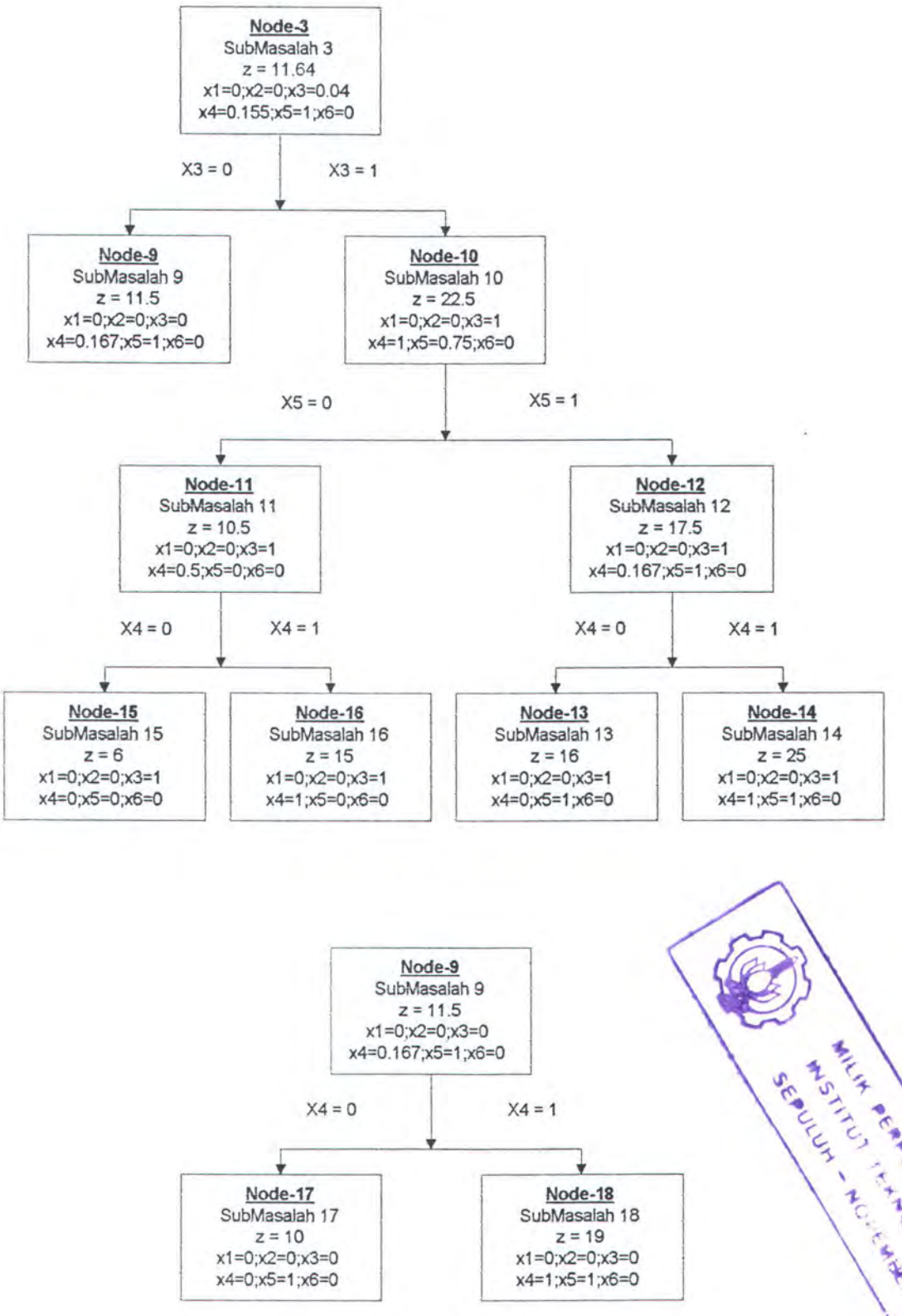
- Akhirnya pada node-7 kita mendapatkan jawaban yang memenuhi semua persyaratan maka untuk permasalahan di atas, kita dapatkan $z = 11$; $x_1 = 0$; $x_2 = 1$; $x_3 = 1$; $x_4 = 0$; $x_6 = 0$; $x_7 = 0$. Dan jawaban tersebut merupakan jawab layak optimum untuk permasalahan z .

- Secara umum waktu yang diperlukan untuk menjalankan prosedur ini cukup singkat. Untuk menyelesaikan satu permasalahan dengan pemrograman separabel, waktunya tidak lebih dari satu detik. Semakin banyak variabel yang dipakai, semakin banyak iterasi yang dibutuhkan dalam pemrograman separabel.



Gambar 4.1.

Diagram Pohon Biner 1



Gambar 4.2.

Diagram Pohon Biner 2

BAB V

PENUTUP

BAB V

PENUTUP

Dari analisa hasil perhitungan dengan menggunakan perangkat lunak ini, dapat diambil beberapa kesimpulan, adalah sebagai berikut :

- Suatu permasalahan pemrograman linier bilangan bulat biner dapat didekati dengan suatu permasalahan non linier, yang tidak mengakibatkan suatu kesulitan penyelesaian seperti umumnya penyelesaian masalah pemrograman non linier.
- Semakin banyak variabel yang digunakan semakin banyak iterasi yang digunakan dalam pemrograman separabel, karena dalam pemrograman separabel, variabel yang digunakan menjadi dua kali variabel permasalahan asal.
- Semakin banyak fungsi kendala juga menyebabkan semakin besar iterasi yang diperlukan untuk meyelesaikan permasalahan yang diberikan.
- Pada umumnya permasalahan minimisasi mempunyai fungsi kendala ketidaksamaan lebih besar atau sama dengan (≥). Sehingga hal ini juga menyulitkan dan memperpanjang waktu yang digunakan, karena adanya tambahan variabel palsu (*artificial variable*) yang mempunyai nilai konstanta sangat besar (metode Big-M).

Perangkat lunak ini dimaksudkan untuk membantu memecahkan masalah pemrograman bilangan bulat yang umumnya banyak mengalami kesulitan. Tetapi

tidak menutup kemungkinan, bahwa perangkat lunak ini juga banyak kekurangannya.

Karena kesulitan mencari contoh yang mempunyai variabel yang cukup besar serta kendala yang banyak, penulis hanya memberikan beberapa contoh yang paling banyak mempunyai enam variabel dan tiga fungsi kendala.

Akan lebih baik lagi jika perangkat lunak ini dicoba untuk menyelesaikan contoh permasalahan penyambilan keputusan yang sebenarnya, karena akan lebih terlihat keuntungan dan kerugiannya jika dibandingkan dengan metode-metode yang lain. Karena terbatasnya waktu, penulis tidak sempat mencari contoh permasalahan yang nyata.

Karena itu untuk pengembangan lebih lanjut dari perangkat lunak ini, penulis mengharapkan tidak hanya pemrograman bilangan biner saja yang bisa diselesaikan dengan pendekatan non linier ini, tetapi lebih meluas ke permasalahan pemrograman bilangan bulat pada umumnya.

DAFTAR PUSTAKA