PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN *GILMORE- GOMORY* UNTUK OPTIMASI *CUTTING - STOCK PROBLEM* PADA INDUSTRI PEMOTONGAN KERTAS

SKRIPSI

Oleh
Lailatul Putri Suryaningtyas
NIM 111810101040

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN *GILMORE-GOMORY* UNTUK OPTIMASI *CUTTING - STOCK PROBLEM* PADA INDUSTRI PEMOTONGAN KERTAS

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

oleh

Lailatul Putri Suryaningtyas
NIM 111810101040

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ayahanda Supa'at dan Ibunda Sri Sulisyaningtyas yang senantiasa memberi doa, semangat, dan kasih sayang;
2. Kakak Putri Afrinda Widyaningtyas dan Adik Rifqi Qurnia Putra Pamungkas yang selalu memberi semangat;
3. Seluruh guru dan dosen sejak taman kanak-kanak hingga perguruan tinggi yang telah membimbing dan membagi ilmu dengan tulus;
4. Almamater jurusan Matematika FMIPA Universitas Jember, SMA Negeri 3 Jombang, SMP Negeri 2 Jombang, SD Negeri Jombatan 4 Jombang, TK Al-Manar Surabaya;

MOTTO

Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila telah selesai, tetaplah bekerja keras. Dan hanya kepada Tuhanmulah engkau berharap.
(QS Al-Insyirah 6-8)*)

Musuh yang paling berbahaya di atas dunia ini adalah penakut dan bimbang. Teman yang paling setia, hanyalah keberanian dan keyakinan yang teguh.
(Andrew Jackson)**)

---

*) Departemen Agama Republik Indonesia. 2004. Al-*Quran* dan Terjemahannya. Bandung. CV Penerbit J-ART.

**) https://http://www.ungkapan.com/andrew-jackson

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Lailatul Putri Suryaningtyas

NIM : 111810101040

menyatakan dengan sesungguhnya bahwa kaya ilmiah yang berjudul "Perbandingan Algoritma *Branch and Bound* dan *Gilmore- Gomory* untuk Optimasi *Cutting - Stock Problem* pada Industri Pemotongan Kertas" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan dalam institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Juni 2016

Yang menyatakan,

Lailatul Putri Suryaningtyas

NIM. 111810101040

SKRIPSI

PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN *GILMORE-GOMORY* UNTUK OPTIMASI *CUTTING - STOCK PROBLEM* PADA INDUSTRI PEMOTONGAN KERTAS

Oleh

Lailatul Putri Suryaningtyas

NIM 111810101040

Pembimbing

Dosen Pembimbing Utama : Mohammad Ziaul Arif , S.Si.,M.Sc

Dosen Pembimbing Anggota : Ahmad Kamsyakawuni, S.Si.,M.Kom

PENGESAHAN

Skripsi berjudul "Perbandingan Algoritma *Branch and Bound* dan *Gilmore-Gomory* untuk Optimasi *Cutting - Stock Problem* pada Industri Pemotongan Kertas" telah diuji dan disahkan oleh Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember pada:

hari :

tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji :

Ketua,	Sekretaris,
Mohammad Ziaul Arif, S.Si., M.Sc	Ahmad Kamsyakawuni, S.Si., M.Kom
NIP.19850111 200812 1 002	NIP. 19721129 199802 1 001
Anggota I,	Anggota II,
Drs. Rusli Hidayat, M.Sc.	Dr. M. Fatekurohman, S.Si., M.Si
NIP.19661012 199303 1 001	NIP. 19690606 199803 1 001

Mengesahkan,
Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Jember

Drs. Sujito, Ph.D
NIP. 19610108 198602 1 001

RINGKASAN

Perbandingan Algoritma *Branch and Bound* dan *Gilmore- Gomory* untuk Optimasi *Cutting - Stock Problem* pada Industri Pemotongan Kertas; Lailatul Putri Suryaningtyas; 111810101040; 2016; 45 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Permasalahan *cutting stock* adalah salah satu dari permasalahan optimasi program linear yang pada dasarnya direduksi ke dalam permasalahan program linear dengan nilai integer. Dengan kata lain, permasalahan *cutting stock* merupakan permasalahan optimasi dalam *integer linear programming* (ILP). Permasalahan ini banyak terjadi dalam berbagai aplikasi matematika di bidang perindustrian seperti industri kertas, industri baja, kayu, dan fiber. Masalah *cutting stock* dalam industri kertas adalah masalah yang terjadi ketika sebuah industri kertas memiliki sejumlah gulungan kertas (*stock roll*), dengan lebar yang bervariasi. Pemotongan gulungan kertas dilakukan sesuai dengan permintaan konsumen. Dalam hal ini, diarahkan untuk memotong gulungan kertas yang sisanya seminimal mungkin sesuai dengan permintaan.

Metode yang digunakan untuk menyelesesaikan permasalahan *cutting stock* tersebut, diantaranya adalah metode *Branch and Bound* dan metode *Gilmore-Gomory*. Oleh karena itu penulis tertarik untuk menerapkan metode tersebut untuk menyelesaikan permasalahan *cutting stock* serta mencari nilai minimum sisa pemotongan kertas. Kompleksitas waktu dari kedua metode tersebut juga akan menjadi bahan pertimbangan ketika menentukan algoritma mana yang lebih baik untuk diterapkan pada *Cutting Stock Problem.*

Pada penelitian ini dilakukan simulasi data permintaan untuk beberapa ukuran kertas. Data tersebut kemudian diolah menggunakan algoritma *Branch and Bound* dan *Gilmore-Gomory* dengan bantuan *software* matematika yaitu MATLAB. Dari hasil penelitian dengan bantuan program menunjukkan: (1) Luas

kertas sisa minimum yang dihasilkan menggunakan algoritma *Branch and Bound* adalah sebesar 1069691.1 dengan jumlah kertas master yang dibutuhkan 7808 lembar dan waktu komputasi 0,541491 detik. (2) Luas kertas sisa minimum yang dihasilkan menggunakan algoritma *Branch and Bound* adalah sebesar 595023,036 dengan jumlah kertas master yang dibutuhkan 7650 lembar dan waktu komputasi 0,3556874 detik.

Dari hasil diatas dapat disimpulkan bahwa algoritma *Gilmore-Gomory* lebih optimal untuk mencari sisa pemotongan kertas yang minimum daripada algoritma *Branch and Bound*. Serta waktu komputasi yang dibutuhkan juga lebih efisien. Karena algoritma *Gilmore-Gomory* bersifat global dan memiliki beberapa tahapan untuk melakukan pencarian solusinya berbeda dengan algoritma *Branch and Bound* pencarian solusinya bersifat lokal.

PRAKATA

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik, dan karuniaNya sehingga skripsi yang berjudul "Perbandingan Algoritma *Branch and Bound* dan *Gilmore- Gomory* untuk Optimasi *Cutting - Stock Problem* pada Industri Pemotongan Kertas" dapat terselesaikan. Skripsi ini disusun untuk memenuhi salah satu syarat dalam menyelesaika pendidikan strata 1 (S1) di Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember. Sholawat dan salam semoga tercurahkan keharibaan beliau nabi Muhammad SAW yang telah menjadi pembawa rahmatan lil'alamin.

Penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Drs. Sujito, Ph.D, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember dan Kusbudiono, S.Si. M.Si., selaku Ketua Jurusan Matematika dan Ilmu Pengetahuan Alam Universitas Jember yang telah memberikan fasilitas-fasilitas dalam tahap perkuliahan;
2. Mohammad Ziaul Arif, S.Si., M.Sc. selaku Dosen Pembimbing Utama dan Ahmad Kamsyakawuni, S.Si., M.Kom. selaki Dosen Pembimbing Anggota yang telah memberikan bimbingan dan bantuan untuk pengerjaan skripsi ini;
3. Drs. Rusli Hidayat, M.Sc. selaku Dosen Penguji I dan Dr. M. Fatekurohman, S.Si., M.Si. selaku Dosen Penguji II yang telah memberikan kritik dan saran yang membangun untuk penyempurnaan skripsi ini;
4. Drs. Moh. Hasan, M.Sc.,Ph.D. selaku Dosen Pembimbing Akademik yang telah membimbing dalam pemilihan matakuliah;
5. seluruh dosen dan karyawan Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember yang telah memberikan ilmu serta membantu selama proses perkuliahan berlangsung;

6. sahabat-sahabat terbaikku: Siti Aminatus Solehah, Risan Nur Santi, Rifka Wahyuni, Rani Eka Yulianti, Nahda Fadila Sari, M. Ibrahim Dharma P., Septiandi yang senantiasa membantuku dan menorehkan sebuah pengalaman indah yang tak terlupakan;
7. saudara-saudaraku KRAMAT ’11 yang selalu memberikan dukungan untuk terus semangat
8. semua pihak yang telah membantu terselesaikannya skripsi ini.

Penulis menyadari bahwa dalam menyusun skripsi ini masih terdapat kekurangan baik isi maupun susunannya. Oleh karena itu, penulis berharap semoga skripsi ini dapat memberi manfaat dan sumbangan bagi pembaca.

Jember, Juni 2016

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Permasalahan *cutting stock* adalah salah satu dari permasalahan optimasi program linear yang pada dasarnya direduksi ke dalam permasalahan program linear dengan nilai integer. Dengan kata lain, permasalahan *cutting stock* merupakan permasalahan optimasi dalam *integer linear programming* (ILP). Permasalahan ini banyak terjadi dalam berbagai aplikasi matematika di bidang perindustrian seperti industri kertas, industri baja, kayu, dan fiber. Masalah *cutting stock* dalam industri kertas adalah masalah yang terjadi ketika sebuah industri kertas memiliki sejumlah gulungan kertas (*stock roll*), dengan lebar yang bervariasi. Pemotongan gulungan kertas dilakukan sesuai dengan permintaan konsumen. Dalam hal ini, diarahkan untuk memotong gulungan kertas yang sisanya seminimal mungkin sesuai dengan permintaan (Hillier, 2001).

Pada proses produksi, khususnya pada proses pemotongan kertas, sering menghasilkan sisa potongan material yang tidak dapat digunakan lagi, sehingga memiliki beberapa kerugian pemotongan dengan kata lain merupakan *cutting stock* (Taha, 2007). Hal ini tidak dapat dihindari karena bahan baku yang diterima dari pemasok tidak selalu dapat memenuhi ukuran yang sesuai dengan yang diharapkan oleh setiap proses. *Cutting stock* disebabkan oleh beberapa faktor, salah satu diantaranya adalah peletakan pola pemotongan yang kurang tepat sehingga mengakibatkan ketidakefisienan penggunaan bahan baku.

Permasalahan *cutting stock* berhubungan dengan ukuran panjang dan lebar pada pemotongan kertas dalam industri kertas. Pemilihan satu set *roll* gulungan kertas yang memiliki ukuran yang bervariasi dimana dalam hal ini digunakan untuk rencana pemotongan kertas (Alves, 2007). Hinxman (1980) membedakan antara masalah variasi ukuran kertas dan masalah peminimalisasian *cutting stock*. Pemecahan masalah ini berdampak dalam hasil satu tahap rencana pemotongan yang lebih baik dalam hal penggunaan material. Sisa pemotongan minimum dapat

dihitung dengan memperhatikan semua gulungan kertas yang tersedia dan kendala pada jumlah panjang gulungan kertas yang dapat digunakan. Pola pemotongan tidak terbatas sebelum dipilih set panjang gulungan kertas.

Penelitian mengenai permasalahan *cutting stock* pertama kali berhasil diformulasikan oleh Kantorovich (1960), kemudian Gilmore dan Gomory (1961 dan 1963) adalah peneliti pertama yang mengusulkan *integer linear programming*, dimana dapat dilakukan *traced back* pada *column generation procedure* untuk permasalahan *cutting stock*. Burke et al. (2004) serta de Gelder dan Wagelmans (2009) juga meneliti tentang permasalahan *cutting stock* dengan berdasar pada penelitian *Gilmore dan Gomory*. Matsumoto et al. (2010) juga melakukan penelitian tentang jumlah *setup* dalam mesin dan kombinasi jumlah tumpukan panjang pesanan yang akan dipotong. Dalam penelitian Triyanti (2008), yang bertujuan untuk menerapkan suatu dasar matematis untuk masalah *cutting stock* dengan model minimasi sisa pemotongan dan maksimasi keuntungan dalam bentuk bobot keuntungan dengan metode simpleks dapat meningkatkan keuntungan dari model yang digunakan pada perusahaan yang diteliti. Erma (2014), juga melakukan penelitian tentang permasalahan *cutting stock* yang menggunakan metode Heuristik *Largest In Least Empty* (LILE), metode Gomory (*Cutting Plane Alghoritma*) dan metode *Branch and Bound* sehingga mendapatkan solusi terbaik dari ketiga metode tersebut yaitu berkurangnya sisa pemotongan kertas dan meningkatnya keuntungan dari perusahaan yang diteliti.

Berdasarkan penelitian sebelumnya, maka peneliti akan meneliti lebih lanjut tentang teori *integer linear programming* dengan beberapa algoritma. Dalam penelitian ini ditentukan model pola pemotongan yang optimal setelah itu dilanjutkan penyelesaian masalah *cutting stock* dengan algoritma *Branch and Bound* serta *Gilmore-Gomory.*

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

a. bagaimana perbandingan hasil penyelesaian masalah *cutting stock* untuk menentukan minimisasi sisa pemotongan dengan algoritma *Branch and Bound* dan *Gilmore-Gomory*?
b. bagaimana model matematis pola pemotongan yang optimal dengan algoritma *Branch and Bound* dan *Gilmore-Gomory*?

1.3 Batasan Masalah

Persoalan dalam penulisan ini dibatasi hanya pada pemotongan kertas untuk model pola pemotongan dua dimensi.

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

a. menyelesaikan permasalahan *cutting stock* untuk meminimalkan sisa pemotongan dengan menggunakan perbandingan algoritma *Branch and Bound* dan *Gilmore-Gomory*;
b. menghasilkan alternatif model matematis pola pemotongan kertas dengan algoritma *Branch and Bound* dan *Gilmore-Gomory*

1.5 Manfaat

Adapun manfaat penelitian dari pembahasan masalah ini adalah sebagai berikut:

a. untuk memperdalam dan mengembangkan wawasan disiplin ilmu riset operasi untuk mengkaji permasalahan tentang pengoptimalan permasalahan *cutting stock* pada industri pemotongan kertas menggunakan dengan algoritma *branch and bound* dan *Gilmore-Gomory* yaitu untuk menentukan minimasi sisa pemotongan;
b. sebagai tambahan wawasan dan informasi tentang pengoptimalan masalah *cutting stock* pada industri pemotongan kertas menggunakan algoritma *Branch*

and bound dan *Gilmore-Gomory* sebagai acuan dalam pengembangan penulisan karya tulis ilmiah;

c. memberikan sumbangan atau masukan yang berguna untuk setiap industri pemotongan kertas agar mampu menyelesaikan *cutting-stock problem* yang dialaminya dalam hal mendapatkan keuntungan yang lebih maksimal dengan sisa pemotongan yang minimal.

BAB 2. TINJAUAN PUSTAKA

2.1 Permasalahan *Cutting Stock*

Permasalahan *cutting stock* merupakan suatu permasalahan yang muncul karena banyak dipakai aplikasinya dalam bidang perindustrian. Misalkan di dalam perindustrian pemotongan kertas, bagaimana memanajemen pemotongan kertas supaya dapat meminimumkan sisa pemotongan yang dihasilkan dan dapat membentuk pola pemotongan yang optimal. Dalam hal inilah permasalahan *cutting stock* dapat digunakan untuk menyelesaikan permasalahan di atas sebagai salah satu aplikasi dari permasalahan optimisasi, atau yang lebih spesifik adalah sebuah permasalahan *integer linear programming*. Sebagai salah satu permasalahan *integer linear programming* maka hasil yang diharapkan dalam suatu permasalahan adalah bilangan bulat. Secara spesifik, dimisalkan terdapat suatu ukuran panjang produk L yang dihasilkan *(raw)* dan akan dipotong dalam beberapa pola j dengan ukuran panjang pesanan x *(final)* dari tiap-tiap pesanan jenis i. Dan tentu saja, nilai – nilai tersebut terbatas pada bilangan integer $i, i = 1,2, \dots, n$ (Eiselt, 2007).

Permasalahan *cutting stock* ini bisa diselesaikan dengan formula yang diperkenalkan oleh *Gilmore-Gomory* (1961, 1963). Pola pemotongan yang mungkin akan dienumerasikan sebelumnya. Pola-pola tersebut didefinisikan sebagai suatu vektor $(a_{1j}, \dots, a_{ij}, \dots, a_{mj})$ dimana elemen a_{ij} menunjukkan jumlah berapa kali pesanan dengan ukuran panjang i dihasilkan dalam pola j. Misalkan x_j adalah variabel yang menandakan jumlah *stock roll* kertas yang akan dipotong sesuai dengan pola j. Dalam membentuk program linear sebagai permasalahan utama, maka fungsi objektif yang mungkin adalah meminimumkan sisa potongan dan meminimumkan jumlah total *stock roll* yang dipotong yaitu meminimumkan$\sum_{j=1}^{n} x_j;\ \ j = 1,2,3, \dots, n$. Dua bentuk formula ini adalah sama jika fungsi objektif dalam permasalahan *cutting stock* dimasukkan beberapa sisa *stock roll* kertas yang ditunjukkan sebagai suatu variabel *slack* (Hillier,1995).

Cutting stock problem dalam bentuk matematis adalah masalah meminimalkan $\sum_{j=1}^{n} x_j ; \ j = 1,2,3, \dots, n$ untuk

$$\sum_{j=1}^{n} a_{ij} x_j \geq N_i$$

dimana $N_i, \ i = 1, \dots, m$ adalah jumlah permintaan panjang gulungan l_i,

x_j adalah jumlah j kali pola pemotongan yang digunakan

a_{ij} adalah jumlah panjang gulungan l_i yang diproduksi setiap j kali dari potongan gulungan yang digunakan.

Jika ada perbedaan panjang gulungan utama (*stock roll)* yang tersedia untuk dipotong maka biaya pemotongannya akan berbeda. Dalam hal ini, fungsi tujuan menjadi

$$\sum_{j=1}^{n} c_j x_j \qquad (2.2)$$

dimana c_j adalah biaya roll utama dari pola pemotongan j (Gilmore,1963)

2.2 *Integer Linear Programming* (ILP)

2.2.1 Definisi *Integer Linear Programming*

Integer linear programming merupakan metode atau teknik matematik yang digunakan untuk membantu dalam pengambilan keputusan. Di dalam *integer linear programming*, seluruh fungsinya (fungsi objektif serta fungsi pembatas) haruslah linear.

Terdapat empat asumsi dasar dalam penyelesaian masalah dengan menggunakan model *integer linear programming*, yaitu (Lieberman,1995):

a. Proposionalitas (*proporsionality)* adalah semua koefisien dalam formulasi fungsi tujuan (c_j) dan fungsi kendala (a_{ij}), merupakan koefisien yang bersifat variabel terhadap besarnya variabel keputusan.
b. Divisibilitas (*divisibility*) adalah solusi permasalahan *integer linear programming* dalam hal ini nilai x_j tidak harus dalam bilangan bulat
c. Additivitas (*additively*) adalah total aktivitas sama dengan jumlah additivitas setiap aktivitas individual.
d. Kepastian (*certainty)* adalah koefisien dalam fungsi tujuan (c_j) dan fungsi kendala (a_{ij}) dapat diketahui dengan pasti dan tidak berubah.

Integer Linear Programming (ILP) merupakan bentuk lain dari *Linear programming* (LP) yang muncul karena tidak semua variabel keputusan dapat berupa bilangan pecahan dengan kata lain asumsi *divisibility* melemah bahkan hilang sama sekali.

2.2.2 Model *Integer Linear Programming* (ILP)

Integer linear programming pada intinya berkaitan dengan program linear dimana beberapa atau semua variabel memiliki nilai *integer* (bulat). Model matematis untuk *integer linear programming* serupa dengan model pemrograman linear, perbedaannya hanya pada penambahan satu batasan bahwa variabelnya harus berupa bilangan bulat (Frederick, 1990). Perhitungan *integer linear programming* memiliki beberapa kesulitan, yaitu banyaknya variabel bilangan bulat dan struktur masalah. Selain itu, jumlah dari batasan juga penting, seringkali penambahan banyak batasan akan mengurangi waktu perhitungan karena banyaknya penyelesaian layak dapat dikurangi. Formulasi standar untuk *cutting stock problem* dimulai dengan daftar pesanan m, masing-masing membutuhkan $q_j,\ j = 1, \dots, m$ potongan. Kemudian mendaftar semua kemungkinan kombinasi dari pemotongan yang disebut dengan pola, dengan masing-masing variabel bilangan bulat positif x_i yang mewakili beberapa pola yang akan digunakan. *Integer linear programming* dalam bentuk matematisnya adalah:

$$\min \sum_{i=1}^{n} c_i x_i \tag{2.3}$$

$$s.t\ \sum_{i=1}^{n} a_{ij} x_i \geq q_j, \qquad \forall_j = 1, \dots, m \tag{2.4}$$

$$x_i \geq 0 \text{ , integer}$$

dimana a_{ij} adalah jumlah pesanan yang muncul sebanyak j kali dalam pola i dan c_i adalah biaya dari pola i.

2.3 Algoritma *Branch and Bound*

Untuk masalah pemrograman linear integer ada beberapa pendekatan yang sering dipakai, seperti algoritma percabangan dan pembatasan (*Branch and Bound*) serta algoritma bidang pemotong. Pertimbangan beberapa masalah minimasi *P*:

$$z^* = \min\{f(x); x \in \Omega\} \quad (2.5)$$

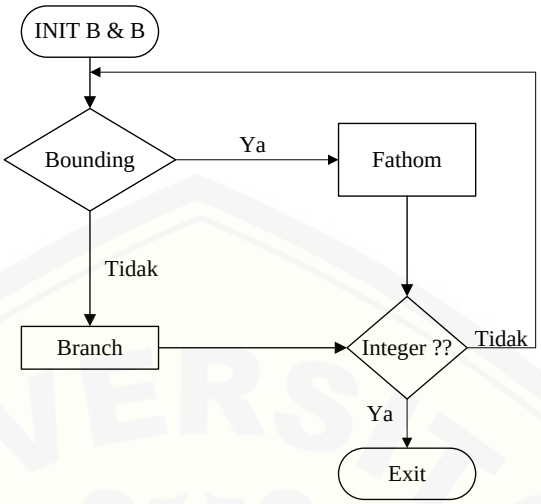
dimana f adalah fungsi objektif dan Ω adalah himpunan dari solusi *feasible*. Sebuah batas bawah pada nilai obyektif optimum z^* adalah beberapa nilai *lb* sehingga $lb \leq z^*$ selalu tepegang. Untuk masalah minimasi, dapat diperoleh, misalnya dengan memecahkan *relaxation* $\underline{P}$ dari *P*

$$\underline{z} = \min\left\{\underline{f}(x); x \in \underline{\Omega}\right\} \quad (2.6)$$

dengan $\Omega \subseteq \underline{\Omega}$ dan $\underline{f}(x) \leq f(x), \forall x \in \Omega$. Selanjutnya akan terjadi perbedaan antara nilai *relaxation* $\underline{z}$ dan akan tertulis batas bawah (*lb*) yang terakhir adalah tujuan sekecil mungkin nilai persamaan 2.5 tidak lebih kecil dari $\underline{z}$, biasanya integer berikutnya di atas *lb*= $\lceil \underline{z} \rceil$. Demikian pula batas atas (*ub*) memenuhi $ub \geq z^*$. Untuk masalah minimasi batas atas mewakili nilai-nilai tujuan beberapa solusi yang layak. Untuk itu pertimbangan partisi dari himpunan Ω ke himpunan bagian $\Omega_1, \dots, \Omega_k$ sehingga $\Omega = \cup_{i=1}^{k} \Omega_i$ dan $\cap_{i=1}^{k} \Omega_i = \emptyset$. Lalu setiap bagian dapat memiliki *subproblem*

$$z_i^* = \min\{f(x); x \in \Omega_i\}, \quad i = 1, \dots, k \quad (2.7)$$

dan batas bawah *lb*, dapat disebut sebagai *local lower bound* yang dinotasikan dengan *llb*. Sehingga dapat dengan mudah melihat bahwa $glb = \min_i lb_i$ adalah *global lower bound*, yaitu berlaku untuk masalah *P*. Setiap *local lower bound* berlaku untuk semua masalah. Dengan demikian *global upper bound* (*gub*), yang dikenal dengan nilai tujuan dari solusi yang layak. Hal tersebut berpisah dari masalah menjadi submasalah oleh partisi himpunan solusi yang layak yang disebut *branching*. Semua masalah yang diperoleh dari percabangan biasanya ditangani sebagai pohon *Branch and Bound* dengan akar seluruh masalah.



Gambar 2.1 Skema Umum *Branch and Bound*

Pada gambar 2.1 dapat dilihat skema *Branch and Bound* yang merupakan generalisasi dari semua masalah (Junger, 2000). Hal ini dimulai dengan menginisialisasi daftar subproblem L dengan semua masalah P: $L = \{P\}$. Prosedur *bounding* diterapkan pada *subproblem* tertentu (simpul akar pada awalnya). Hal ini dapat digunakan untuk menghitung sebuah *llb* batas bawah lokal dan mencoba untuk meningkatkan batas atas gub. Jika $llb < gub$ maka *node* dapat berisi solusi yang lebih baik (lanjutkan dengan *branch*). Jika *node* fathomed atau tidak dipertimbangkan maka masalah telah diselesaikan tanpa percabangan. Dalam hal ini, hanya ada dua kemungkinan yaitu solusi layak yang telah ditemukan $glb = gub < \infty$ atau problem yang *infeasible* $glb = gub = \infty$. Sebuah subproblem dapat juga jadi tak layak, maka didapatkan $llb = \infty$.

Prosedur branch yang tepecah *node* yang telah diberikan kedalam beberapa submasalah dan menambahkan L. Selama *node* berada dalam L dengan batas bawah lokal $llb < gub$, prosedur *select* merupakan salah satu cara untuk proses selanjutnya (Lodi, 2002). Konsep dasar teknik *Branch and Bound* adalah *divide and conquer*. Dimana *conquered* adalah penyelesaian masalah awal yang terlalu "besar" sehingga penyelesaiannya dibagi dengan submasalah-submasalah sampai submasalah tersebut dapat diababaikan. Sedangkan *dividing* adalah melakukan

pembagian daerah *feasible* dari masalah awal menjadi daerah *feasible* yang lebih kecil. *Conquering (fathoming)* adalah melakukan sebagian penyelesaian masalah dengan membatasi (*bounding*) dengan beberapa nilai opitmal dari setiap submasalah dengan mengabaikan submasalah yang tidak memuat solusi optimal dari masalah awal (Pangestu, 1985). Berdasarkan konsep dasar di atas, teknik *Branch and Bound* memiliki tiga langkah dasar, yaitu:

a. *Branching*

Cara penyelesaian *Branch and Bound* adalah dengan menyelesaikan *integer linear programming* yang tertera pada persamaan (2.3) dengan metode simpleks. Jika solusi optimal yang didapat bernilai bilangan bulat, maka solusi optimal dari *linear programming* dari persamaan (2.3) telah optimal. Akan tetapi jika terdapat solusi yang bernilai pecahan maka daerah *feasible linear programming* dari persamaan (2.3) akan dipecah menjadi dua bagian. Untuk memecah daerah *feasible*, variabel yang bernilai pecahan tersebut dipilih secara acak yang selanjutnya disebut dengan *branching* variabel. *Branching* variabel adalah variabel yang digunakan sebagai cabang dengan memberikan nilai batas (agar *branching* variable menjadi sederhana maka indeks yang dipilih adalah indeks yang terkecil) (Tjutju, 2003).

b. *Bounding*

Setiap submasalah dari *linear programming* diselesaikan dengan metode simpleks sehingga didapat solusi optimal. Jika solusi yang didapatkan bernilai bilangan bulat, maka solusi optimal dari *linear programming* akan menjadi solusi optimal untuk submasalah yang bersesuaian. Jika nilai optimal submasalah kurang dari samadengan nilai optimal *linear programming*, maka akan mengakibatkan nilai optimal dari *linear programming* merupakan batas atas dari nilai optimal submasalah yang bersesuaian (Stephen,1996).

c. *Fathoming*

Suatu submasalah tidak akan diamati jika *linear programming* suatu submasalah tidak memiliki daerah *feasible (infeasible)*, karena sebmasalah akan menjadi *infeasible* juga. Hal tersebut terjadi dikarenakan daerah *feasible* submasalah merupakan subhimpunan dari daerah *feasible linear programming.*

Teknik *Branch and Bound* akan berhenti jika tidak ada lagi submasalah yang diamati. Sehingga solusi optimal dari masalah awal adalah *incumbent* yang baru (Gilmore, 1963).

Notasi yang digunakan dalam penyelesaian algoritma *Branch and Bound* ini dapat dinotasikan sebagai berikut:

a. *LIST* (*k*) merupakan daftar dari perhitungan fitur-fitur secara terurut pada *level* ke *k*.
b. *z* merupakan fitur yang akan dibuang.
c. *k* menunjukkan level dari *tree*
d. *x** menyimpan nilai *bound*.

Selanjutnya algoritma untuk menyelesaikan masalah *integer linear programming* dengan pendekatan *branch and bound*:

Langkah 1 Inisialisasi :

nilai $x^* = -\infty, k = 1, z_0 = 0$

Mendeklarasikan suatu nilai suatu variabel x^* dimana nilai tersebut merupakan suatu nilai untuk menyimpan nilai batas dari suatu persamaan.

Langkah 2 *Generate successors*:

Inisialisasi $LIST(k)$ kosong, yang isinya adalah fitur-fitur $(z_1, \dots, z_{k-1})$ dengan cara: $LIST(k) = \{z_{k-1} + 1, z_{k-1} + 2, \dots, d + k\}$

Mendeklarasikan suatu tingkat keberhasilan suatu algoritma dengan mendaftar suatu perhitungan fitur-fitur terurut pada level dari percabangan. Dimana tingkat keberhasilan dari suatu $LIST(k)$ adalah $(z_1, \dots, z_{k-1})$ yang dapat diperoleh dengan cara $LIST(k) = \{z_{k-1} + 1, z_{k-1} + 2, \dots, d + k\}$ dimana *z* merupakan suatu fitur yang dibuang.

Langkah 3 Memilih *node* baru:

a. Jika $LIST(k)$ kosong, dilanjutkan ke Langkah 5.
b. Jika $LIST(k)$ tidak kosong, maka $z_k = m$, dimana $J_k(z_1, \dots, z_{k-1}, m) = \max_{j \varepsilon LIST\,(k)} J_k(z_k, \dots, z_{k-1}, j)$
c. Kemudian *m* dihapus dari $LIST(k)$.

Pada langkah ini kita membuat suatu keputusan untuk langkah selanjutnya dimana diberikan suatu pilihan jika daftar perhitungan fitur terurut

$(LIST(k))$ kosong maka akan dilanjutkan ke langkah *backtrack* akan tetapi jika daftar perhitungan fitur terurut $(LIST(k))$ tidak kosong maka dideklarasikan suatu variabel suatu fitur yang akan dibuang yaitu $z_k = m$, dimana $J_k(z_1, \dots, z_{k-1}, m) = \max_{j \varepsilon LIST\,(k)} J_k(z_k, \dots, z_{k-1}, j)$. Selanjutnya variabel m akan dihapus.

Langkah 4 *Check Bound* :

a. Jika $J_k(z_1, \dots, z_{k-1}) < x^*$ maka dilanjutkan ke Langkah 5.

b. Jika $k = \bar{d}$, maka dilanjutkan ke Langkah 6.

c. Lainnya $k = k + 1$ dan kembali ke Langkah 2.

Langkah ini merupakan langkah dimana akan memeriksa batas dari suatu nilai batas yang telah disimpan dengan ketentuan jika $J_k(z_1, \dots, z_{k-1}) < x^*$ maka lanjut ke langkah selanjutnya akan tetapi jika $k = \bar{d}$, maka dilanjutkan ke langkah 6. Sedangkan jika muncul nilai selain itu maka akan kembali ke langkah 2.

Langkah 5 *Backtrack* ke *level* sebelumnya:

a. $k = k - 1$

b. Jika $k = 0$, maka algoritma telah selesai.

c. Jika $k \neq 0$, maka kembali ke Langkah 3.

Langkah ini merupakan langkah *backtrack* yaitu penentuan nilai *level* dari sebuah pohon dimana jika nilainya telah sama dengan 0 maka algoritma ini telah selesai. Jika tidak sama dengan 0 maka kembali lagi ke langkah 3.

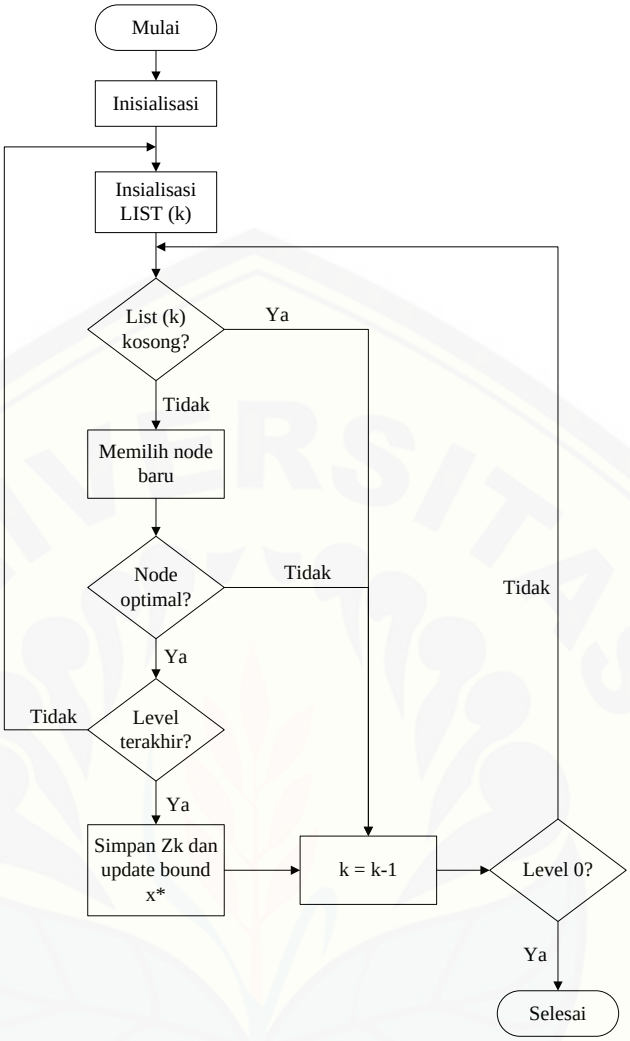
Langkah 6 *Level* terakhir dan *update bound* :

a. $x^* = J_{\bar{d}}(z_1, \dots, z_{\bar{d}})$ disimpan sebagai nilai *bound*

b. $(z_1, z_2, \dots, z_{\bar{d}})$ disimpan sebagai $({z_1}^*, {z_2}^*, \dots, {z_{\bar{d}}}^*)$

c. kemudian kembali ke Langkah 5.

Langkah ini merupakan langkah memperbarui batas dimana suatu nilai batas yang disimpan adalah $x^* = J_{\bar{d}}(z_1, \dots, z_{\bar{d}})$ dan nilai fitur yang dibuang disimpan sebagai $({z_1}^*, {z_2}^*, \dots, {z_{\bar{d}}}^*)$ setelah batas berhasil disimpan maka kembali ke langkah *backtrack* (Fukunaga, 1990).

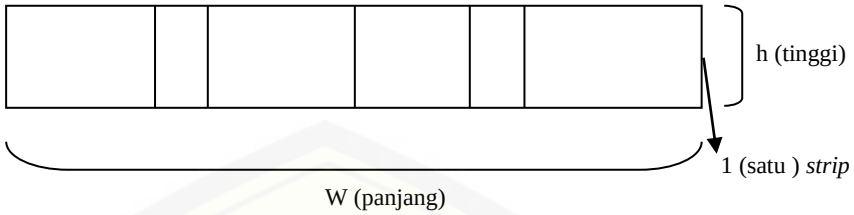


Gambar 2.2 *Flowchart* Algoritma *Branch and Bound*

2.4 Algoritma *Gilmore–Gomory*

Permasalahan *cutting stock* ini dapat diselesaikan dengan formula yang diperkenalkan oleh Gilmore-Gomory (1961, 1963). Pola pemotongan yang mungkin akan dienumerasikan sebelumnya. Pola-pola (*cutting patterns*) tersebut didefinisikan sebagai suatu vektor $(a_{1j}, \dots, a_{ij})$ dimana elemen a_{ij} menunjukkan jumlah beberapa pesanan dengan ukuran panjang *i* dihasilkan dalam pola *j* (*cutting patterns*) (Bazarra, 1977). Pencetus pemodelan matematika untuk masalah *cutting–stock* adalah *Gilmore–Gomory*. Dalam artikelnya yang pertama (P.C Gilmore dan R.E Gomory, 1961). *Gilmore dan Gomory* mencoba membahas

untuk menyelesaikan salah satu jenis *cutting stock* yaitu *one dimension cutting stock problem* (1DCSP).



Gambar 2.3 Gilmore-Gomory 1D Tahap 1

Pola pemotongan pada Gambar 2.3 menjelaskan beberapa unsur dari setiap jenis *stock* yang dipotong. Jika vektor kolom $a^j = (a_{1j}, \dots, a_{mj}) \in \mathbb{Z}_+^m,\ j = 1, \dots, n$, mewakili semua pola pemotongan yang mungkin. Untuk menjadi pola pemotongan yang sah, maka a^j harus memenuhi

$$\sum_{i=1}^{m} l_i\, a_{ij} \leq L \tag{2.8}$$

dimana l_i adalah ukuran dari potongan yang diminta

a_{ij} adalah jumlah beberapa pesanan dengan ukuran panjang *i* dihasilkan dalam pola *j*

L adalah ukuran lebar dari *stock roll*

Selain itu, untuk mempertimbangkan hanya pola yang tepat maka

$$a_{ij} \leq b_i,\ \ i = 1, \dots, m,\ \ j = 1, \dots, n \tag{2.9}$$

dimana b_i adalah permintaan pesananan potongan

Karena syarat diatas dapat mengurangi ruang pencarian dari suatu pola dimana kebutuhan dari *b* semakin menyempit (Nitsche, 1999).

Misalkan $x_j,\ j = 1, \dots, n$, adalah frekuensi (intensitas, perkalian, dan jumlah pola) dari pola yang mungkin. Maka model dari *Gilmore–Gomory* adalah: (Gilmore, 1961)

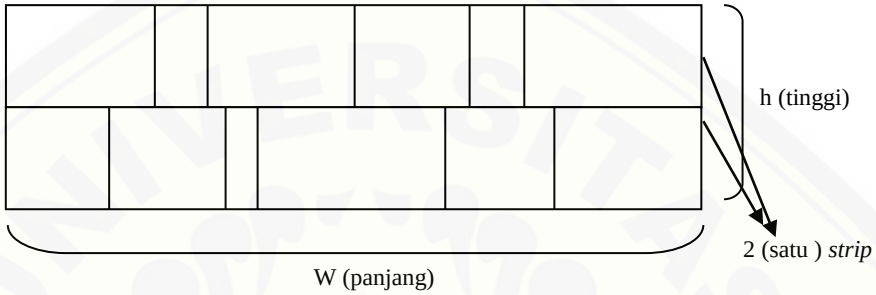
$$z_{G\&G}^{1D-CSP} = min \sum_{j=1}^{n} x_j \tag{2.10}$$

$$s.t. \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \dots, m \tag{2.11}$$

$$x_j \in \mathbb{Z}_+, \quad j = 1, \dots, n \tag{2.12}$$

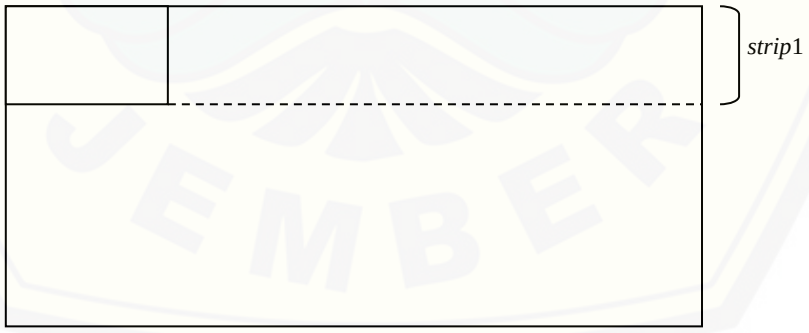
Jumlah dari variabel/ kolom tidak terselesaikan secara eksplisit. Biasanya, pola yang diperlukan dihasikan dari *column generation*. Namun jumlah pola yang

berbeda tidak dapat lebih besar dari panjang *stock* dan biasanya sebanding dengan jumlah potongan. Dari suatu bentuk satu dimensi dipotong menjadi potongan – potongan sesuai ukuran yang diinginkan. Kemudian *Gilmore–Gomory* (Gilmore dan Gomory, 1963) menggembangkan menjadi bentuk dua dimensi. Bentuk satu dimensi seperti gambar 2.3, setelah ditambah satu *strip* akan menjadi bentuk 2 dimensi.



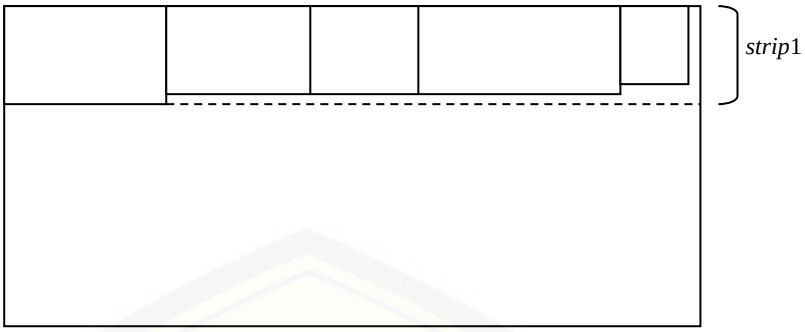
Gambar 2.4 Gilmore-Gomory 1D Tahap 2

Logika dasar untuk menyelesaikan masalah *two dimension cutting stock problem* (2DCSP) adalah mengurutkan sesuai tinggi masing – masing potongan secara *descending*. Kemudian dimasukkan potongan dengan tinggi paling besar ke kiri atas material, tinggi dari potongan pertama yang dimasukkan adalah menjadi tinggi *strip* pertama.



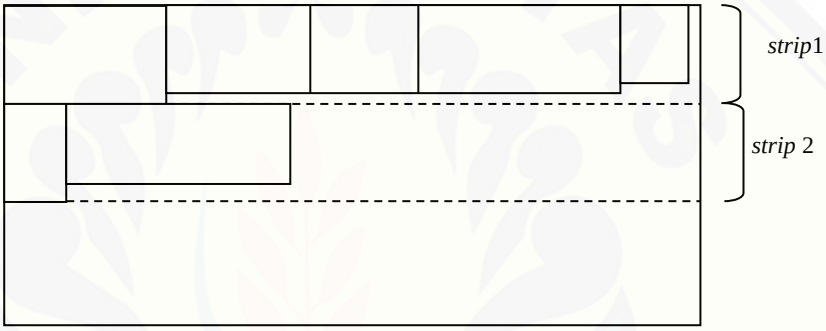
Gambar 2.5 Gilmore-Gomory 2D Tahap 1

Setelah potongan pertama dimasukkan segera diikuti oleh potongan berikutnya sesuai urutan tinggi.



Gambar 2.6 Gilmore-Gomory 2D Tahap 2

Apabila potongan berikut tidak muat dalam *strip*, tinggi potongan berikut menjadi tinggi *strip* baru.



Gambar 2.7 Gilmore-Gomory 2D Tahap 3

Two dimension cutting stock problem terdiri dari penempatan ukuran *pieces* pada suatu *stock roll*. Keuntungan dari setiap *pieces* dapat dimaksimalkan. Pada gambar 2.5 sampai gambar 2.7 merupakan tahapan untuk pemotongan dua dimensi dimana *strip* (pola jalur) yang diproduksi di tahap pertama dipotong-potong di tahap kedua. Setiap potongan dalam satu strip satu dengan *strip* yang lain dibatasi yang berarti bahwa setiap potongan dapat dibedakan oleh jenis dan jumlah suatu unsur dari setiap *pieces*. Formulasi *Gilmore-Gomory* untuk 2D-CSP hampir sama dengan 1D-CSP. Jika vektor kolom $a^j = \left(a_{1j}, \dots, a_{mj}\right) \in \mathbb{Z}_+^m,\ j = 1, \dots, n$, mewakili semua pola garis, maka a^j harus memenuhi $\sum_{i=1}^{m} l_i\, a_{ij} \leq L$ dan $a_{ij} \leq b_i,\ \ i = 1, \dots, m$. Misalkan $x_j,\ j = 1, \dots, n,$ adalah intensitas pola dalam suatu solusi. Maka model matematisnya :

$$z_{G\&G}^{2D-CSP} = min \sum_{j=1}^{n} \left(\sum_{i=1}^{m} -p_i a_{ij}\right) x_j \qquad (2.13)$$

$$s.t. \quad \sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, \dots, m \tag{2.14}$$

$$\sum_{j=1}^{n} w(a^j)x_j \leq W \tag{2.15}$$

$$x_j \in \mathbb{Z}_+, \quad j = 1, \dots, n \tag{2.16}$$

dimana $w(a^j) = max_{i:a_{ij}>0}\{w_i\}$ adalah lebar *strip*

p_i adalah panjang dari setiap potongan

W adalah ukuran panjang dari *stock roll*

Mewakili tujuan sebagai fungsi minimasi dan tetap menjaga analogi dengan 1D-CSP. Model ini dapan dilihat sebagai perwakilan multidimensi *knapsack problem.*

Metode konvensional yang biasa digunakan untuk mencari solusi masalah program linear, tidak dapat langsung diterapkan untuk mencari masalah (2.13). Hal ini karena tidaklah mudah untuk mencari semua kemungkinan *cutting pattern*. Pendekatan alternatif menggunakan *Column Generation* untuk menyelesaikan model program linear relaksasi dari (2.13). Selanjutnya, model (2.13) disebut sebagai *master problem*. Teknik *Column Generation* dikenalkan oleh Gilmore dan Gomory (Gilmore,1963) untuk menyelesaikan masalah *cutting stock* satu dimensi. Teknik ini terbukti efektif untuk menyelesaikan masalah program linear yang matriks koefisiennya tidak mudah untuk ditentukan. Teknik ini didasari oleh metode simpleks yang direvisi (*revised simplex method*), dimana dapat menetukan solusi layak basis dari $(n + m) \times (n + m)$ submatriks dari matriks koefisien yang nonsingular. Submatriks ini disebut matriks basis, dan dinotasikan dengan **B**. Untuk mencari matriks **B** dari masalah (2.13), dengan mencari $(n \times n)$ submatriks yang merepresentasikan *n* buah cutting pattern yang berbeda untuk setiap bahan baku jenis *k*. Submatriks $(n \times n)$ ini dinotasikan dengan **Dk**. *Cutting pattern* ini tidaklah sulit untuk dicari. Sebagai contoh, untuk setiap jenis bahan baku, pilih sebuah matriks diagonal orde *n* dimana setiap entri diagonal ke-*i* berisi jumlah final *i* maksimum yang mungkin ditempatkan pada sebuah bahan baku. $m \times (n + m)$ submatriks sisanya adalah matriks yang setiap entrinya bernilai 1. Setelah menetukan matriks **B**, solusi basis layak diperoleh dari relasi $\boldsymbol{X_B = B^{-1}b}$, dimana **b**=$\left[\frac{d_i}{k_k}\right]$.

Untuk menetukan apakah solusi yang diperoleh optimal atau tidak, untuk setiap jenis bahan baku *k*, kita *generate* sebuah kolom baru $\boldsymbol{y^k}$ dengan elemen $y_i^k, i = 1,2, \dots, n$. Misal π_k adalah *shadow price* yang berasosiasi dengan kapasitas dari konstrain untuk bahan baku jenis *k* dimana π_k adalah elemen ke *k* dari matriks $\boldsymbol{cP^{-1}}$ dengan **c** adalah matrik baris berdimensi *m* yang entrinya adalah ongkos perunit setiap bahan baku dan **P** adalah matriks diagonal berorde *m* yang semua entrinya bernilai 1. *Reduced cost* yang berasosiasi dengan basis **B** untuk setiap bahan baku jenis *k* adalah

$c_k - \pi_k - \sum_{i=1}^{n} \lambda_i y_i^k$

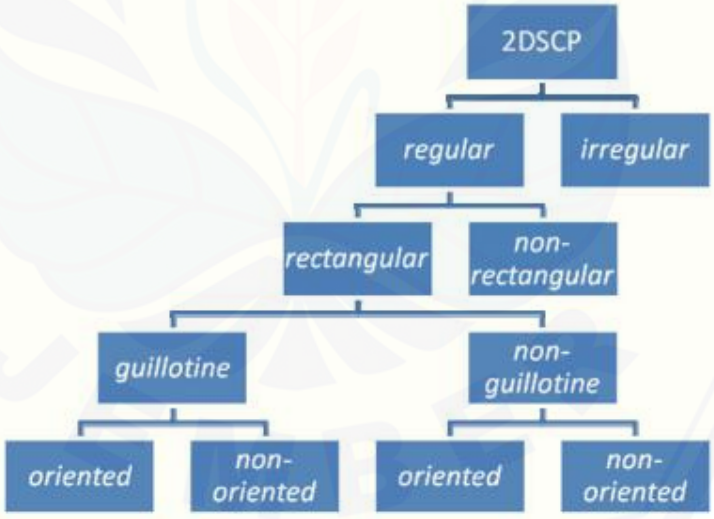
Shadow price λ dengan elemen $\lambda_i, i = 1,2, \dots, n$ adalah $\lambda = \mathbf{1_m} \boldsymbol{D_k}^{-1}$ adalah *shadow price* yang berasosiasi dengan masing-masing permintaan *final i*, dimana $\mathbf{1_m}$ adalah matriks baris berdimensi *m* yang semua entrinya bernilai 1. Kondisi optimal tercapai jika minimum *reduced cost* bernilai tak negatif. Jika minimum *reduced cost* bernilai negatif, kolom baru $\mathbf{y^k}$ akan masuk sebagai basis. Kolom baru akan terus digenerate selama kondisi optimal *master problem* (2.13) belum tercapai. Keterbatasan algoritma Gilmore dan Gomory asli adalah bahwa hal itu tidak menangani integralistik, sehingga solusinya mungkin berisi pecahan, misalnya pola tertentu harus diproduksi 3,67 kali. Pembulatan ke bilangan bulat terdekat sering tidak bekerja, dalam arti bahwa hal itu dapat menyebabkan solusi sub-optimal dan / atau *infeasibility* bawah- atau over-produksi dari beberapa perintah (dan mungkin dengan adanya kendala permintaan dua sisi). Keterbatasan ini diatasi dalam algoritma modern, yang dapat memecahkan untuk optimalitas (dalam arti mencari solusi dengan sisa minimal) (Goulimis,1990).

2.5 Klasifikasi Bentuk Pemotongan

Salah satu jenis dari *cutting stock problem* adalah *two dimensional cutting stock problem*. Jenis ini umumnya banyak dipakai untuk suatu kumpulan bahan yang akan dipotong daro bentuk *rectangular* besar yang ukurannya sudah ditetapkan, atau pun berasal dari gulungan bahan, dengan tujuan untuk meminimalkan total biaya termasuk didalamnya biaya bahan baku.

Two dimensional cutting stock problem dapat diklasifikasikan menjadi pemotongan bentuk *regular* dan *irregular*. Bentuk *irregular* merupakan bentuk yang tidak beraturan, bahan dipotong dengan bentuk apapun seperti yang terjadi dalam industry baju, sepatu kulit, furniture (bentuk khusus), mobil, dan bahkan industry pesawat terbang. Pemotongan bentuk regular berbentuk rectangular atau bentuk geometri lain banyak dipakai dalam industri pemotongan kertas, lembaran metal, dan industri kayu.

Bentuk pemotongan *regular rectangular* dapat menggunakan *teknik guillotine*, yaitu memotong dari ujung ke ujung dari bentuk *rectangle* tersebut, atau pun dengan teknik *non-guillotine*. Pemotongan *rectangular* kemudian dapat diklasifikasikan menjadi pemotongan *oriented*, di mana panjang dari bentuk yang akan dipotong, disejajarkan paralel sesuai dengan panjang bahan utama (Cheng, 1994). Dengan kata lain dalam *oriented* tidak dilakukan perputaran (*rotation*) potongan sama sekali. Klasifikasi ini dapat diilustrasikan dalam Gambar 2.8



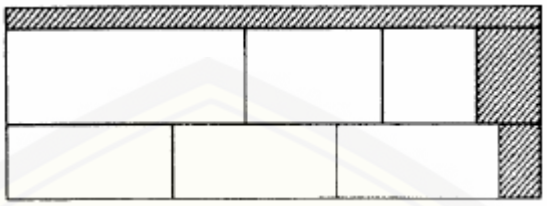
Gambar 2.8 Klasifikasi 2DCSP

2.6 Tingkat (*Stage*) Pemotongan

Dalam perkembangan algoritma untuk 2DCSP, muncul istilah "*stage*". *Stage* dalam 2DCSP adalah berapa kali pemotongan horisontal dan vertikal harus

dilakukan. Tiga jenis pola yang lazim digunakan sampai saat ini adalah 2-*stage*, 3-*stage*, dan n-*stage*.

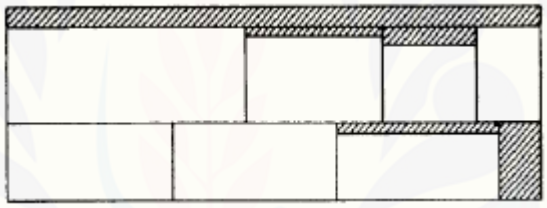
a. Pola 2-*stage*



Gambar 2.9 Pola 2-*stage*

Pola ini hanya terjadi jika potongan yang diinginkan mempunyai ukuran tinggi yang sama. Disebut 2-*stage* (dua tingkat) karena dalam pola ini terjadi 2 kali tahap potong, pertama memotong horisontal, kedua memotong vertikal.

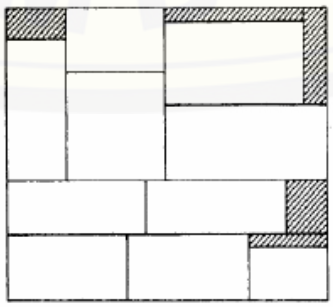
b. Pola 3-*stage*



Gambar 2.10 Pola 3-*stage*

Pola ini adalah pola yang sering dipakai karena memudahkan peletakan dan memudahkan perhitungan. Potongan-potongan yang diinginkan dalam pola ini mempunyai ukuran yang bervariatif. Disebut 3-*stage* (tiga tingkat) karena dalam pola ini terjadi 3 kali tahap potong, pertama memotong horisontal, kedua vertikal, yang terakhir adalam memotong waste (sisa potong) yang terbentuk. *f*

c. Pola n-*stage*



Gambar 2.11 Pola n-*stage*

Pola ini adalah hasil dari algoritma yang kompleks, karena benar-benar memampatkan potongan sehingga diperoleh sisa potong yang minimal. Potonganpotongan yang diinginkan dalam pola ini mempunyai ukuran yang bervariatif juga. Disebut n-*stage* (n tingkat) karena kerumitan pemotongan dari pola yang terbentuk.

Dalam kenyataannya, sering ada batasan atau permintaan khusus dalam tiap masalah yang terjadi. Dalam skripsi ini akan dibahas adalah pemotongan secara *guillotine orthogonal*, yaitu potongan-potongan berbentuk *rectangular* dan hanya dapat dipotong horisontal atau vertikal dari satu sisi ke sisi lain. Jika dalam industri pemotongan digunakan mesin, maka perlu diperhatikan jumlah *stage* dalam lembaran material. Dalam setiap *stage*, pemotongan horisontal dan vertikal dapat dilakukan, tetapi tidak secara bersamaan. Potongan yang sudah melalui satu *stage* tidak dapat dikembalikan ke *stage* sebelumnya. Kondisi ini membatasi pemotongan horisontal dan vertikal, dan tinggi maksimum dari tingkatan pemotongan dalam tiap lembar bahan baku. Contoh pada pola three-*stage*, di mana *stage* pertama hanya dapat memotong horisontal, *stage* kedua vertikal, dan *stage* terkahir horisontal lagi (Zimet,1988).

2.7 Jenis-Jenis Kertas

Sering kali kita melihat berbagai jenis kertas, seperti buku tulis, koran, kertas ujian, majalah, kertas bungkus nasi, sampai kertas bungkus Mie instan, namun sering kita tidak mengetahui jenis kertas yang digunakan. Berikut ini akan dijabarkan berbagai jenis kertas beserta ukurannya masing-masing yang sering digunakan dalam percetakan (Sumarto,2006).

a. Jenis Kertas

- **Art Paper**

  Jenis kertas ini mempunyai tekstur permukaan yang licin dan halus. Kertas ini biasanya digunakan untuk mencetak brosur, majalah atau catalog.

- **Art Carton**

  Kertas jenis ini karakteristiknya sama dengan art paper, hanya lebih tebal. Biasa dipakai untuk mencetak kartu nama, cover buku, brosur, paperbag , map dan lain sebagainya.

- **HVS**

  Jenis bahan kertas ini memiliki permukaan kasar. Biasa digunakan untuk fotocopy atau printer. Tidak hanya itu jenis kertas ini juga dapat digunakan untuk mencetak buku.

- **BC**

  Jenis kertas ini memiliki tekstur yang halus namun tidak *coated*. Kertas jenis ini tersedia dalam beragam warna, dimana biasanya digunakan untuk mencetak kartu nama, sertifikat dan lain-lain.

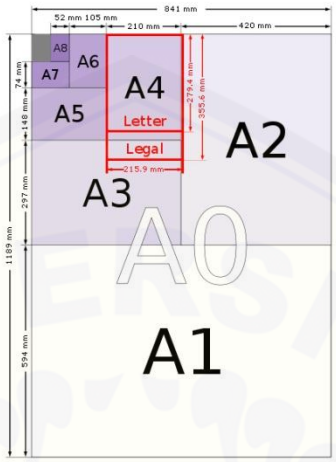
b. Ukuran kertas

Dalam industriapat pemotongan kertas terapat beberapa ukuran kertas diantaranya adalah sebagai berikut

1. Seri A

- A0 = 84,1 x 118,9 cm, biasa disebut ukuran Plano, ukuran di toko kertas sebelum dipotong.
- A1 = 59,4 x 84,1cm
- A2 = 42,0 x 59,4cm
- A3 = 29,7 x 42,0cm
- A3+ = 31,8 x 48,0cm
- A4 = 21,0 x 29,7cm
- A5 = 14,8 x 21,0cm
- A6 = 10,5 x 14,8cm
- A7 = 7,4 x 10,5cm
- A8 = 5,2 x 7,4cm
- A9 = 3,7 x 5,2cm
- A10 = 2,6 x 3,7cm

Ukuran kertas pada Seri A digunakan untuk cetakan umum dan perkantoran serta penerbitan. Dasar ukuran adalah A0 yang luasnya setara dengan satu meter persegi biasa juga disebut ukuran Plano.

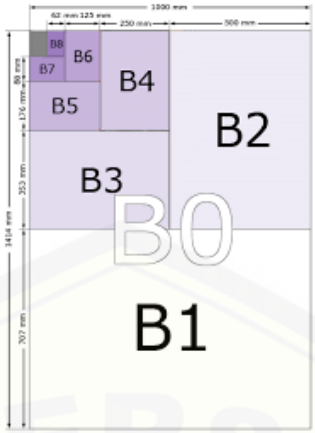


Gambar 2.12 Ukuran kertas Seri A

2. Seri B

- B0 = 100,0 x 141,4cm
- B1 = 70,7 x 100,0cm
- B2 = 50,0 x 70,7cm
- B3 = 35,3 x 500cm
- B4 = 25,0 x 35,3cm
- B5 = 17,6 x 25,0cm
- B6 = 12,5 x 17,6cm
- B7 = 8,8 x 12,5cm
- B8 = 6,2 x 8,8cm
- B9 = 4,4 x 6,2cm
- B10 = 3,1 x 4,4cm

Ukuran kertas putih Seri B digunakan untuk poster dan lukisan dinding



Gambar 2.13 Ukuran Kertas Seri B

3. SERI F

- **F4 /Folio** = 21,5 x 33,0 cm.

Ukuran kertas Seri F digunakan untuk fotocopy dan perkantoran (Sumarto, 2006).

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

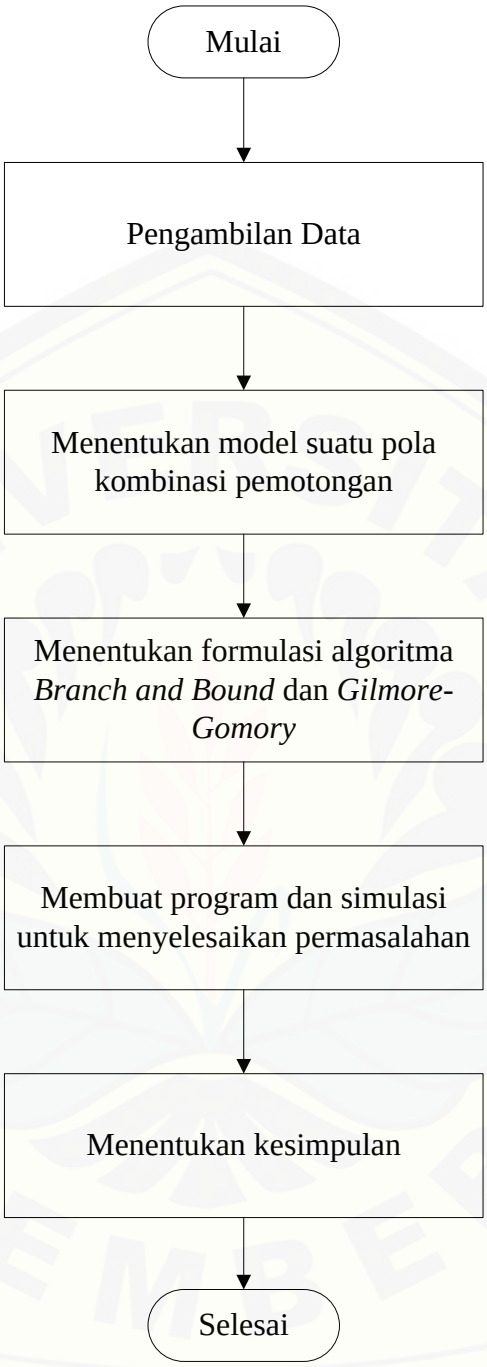
Data yang akan digunakan dalam penelitian ini adalah data sekunder. Data sekunder berasal dari industri pemotongan kertas untuk mendapatkan data sebagai berikut :

a. data ukuran kertas *rectangle* (bahan baku).
b. data ukuran kertas *pieces* serta banyaknya permintaan masing-masing *pieces*.
c. data sisa kertas dari pola pemotongan saat ini.
d. sistem pemenuhan permintaan saat ini.

3.2 Langkah Penelitian

Langkah-langkah penelitian dari perbandingan optimasi *cutting - stock problem* pada industri pemotongan kertas menggunakan algoritma *branch and bound* dan *gilmore – gomory* sebagai berikut:

a. mengumpulkan data ukuran *stock roll* , ukuran lebar kertas sesuai dengan jumlah permintaan.
b. menentukan model suatu pola pemotongan *stock roll* dan sisa pemotongan kertas.
c. menyelesaikan permasalahan minimalisasi sisa pemotongan dan maksimalisasi keuntungan dengan penerapan program linier menggunakan algoritma *Branch and Bound* dan *Gilmore – Gomory*.
d. membandingkan solusi penyelesaian masalah antara algoritma *Branch and Bound* dan *Gilmore - Gomory*.



Gambar 3.1 Skema Penelitian

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan data, hasil, dan pembahasan, maka dapat diperoleh kesimpulan dari penyelesaian masalah cutting-stock pada industri kertas dengan algoritma *Branch and Bound*dan *Gilmore-Gomory* sebagai berikut.

a. Penerapan algoritma *Branch and Bound* menghasilkan jumlah sisa pemotongan 1069691,1 cm^2 dengan jumlah master yang dibutuhkan 7808 lembar serta waktu perhitungan 0,541491 detik, sedangkan algoritma *Gilmore-Gomory* menghasilkan jumlah sisa pemotongan 595023,036 cm^2 dengan jumlah master yang dibutuhkan 7650 lembar serta waktu perhitungan 0,3556874 detik. Berdasarkan hasil perhitungan kedua algoritma tersebut, algoritma *Gilmore-Gomory* lebih optimal untuk mencari sisa pemotongan kertas yang minimum daripada algoritma *Branch and Bound*. Serta waktu komputasi yang dibutuhkan juga lebih efisien. Karena algoritma *Gilmore-Gomory* bersifat global dan memiliki beberapa tahapan untuk melakukan pencarian solusinya berbeda dengan algoritma *Branch and Bound* pencarian solusinya bersifat lokal.

b. Persamaan program linear yang diselesaikan dengan algoritma *Branch and Bound* dan algoritma *Gilmore-Gomory* adalah

   Meminimumkan $Z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20}$

   Terhadap kendala

$$
\begin{aligned}
2x_{17} + 2x_{18} &\geq 2500 \\
x_4 + x_{10} + 4x_{12} + x_{15} + x_{19} + x_{20} &\geq 2000 \\
8x_{13} + x_{20} &\geq 1500 \\
16x_1 + x_3 + x_6 + 2x_8 &\geq 3000 \\
x_3 + x_4 + x_6 + x_8 + x_9 + x_{10} + 32x_{14} + x_{15} + 3x_{16} + x_{19} &\geq 3500 \\
x_3 + x_4 + x_7 + x_{11} + x_{15} + x_{19} &\geq 3000
\end{aligned}
$$

$$2x_8 + 2x_{10} + 2x_{16} + x_{20} \geq 4500$$
$$5x_6 + x_{11} + x_{20} \geq 4000$$
$$2x_3 + 11x_5 + 3x_7 + x_8 + x_{11} + 2x_{16} \geq 4500$$
$$x_8 + 22x_9 \geq 4000$$
$$14x_2 + x_{20} \geq 5000$$
$$x_j \geq 0, x \in \mathbb{Z}_+, j = 1,2, \dots ,20$$

5.2 Saran

Selain algoritma *Branch and Bound* dan algoritma *Gilmore-Gomory* penelitian ini dapat dikembangkan dengan menggunakan algoritma lain yang dapat digunakan untuk menyelesaikan masalah *cutting stock*. Peneliti selanjutnya juga dapat menambah dimensi yang akan diteliti.

DAFTAR PUSTAKA

Alves, C. 2007. *Solving the Assortment* dan *Trim Loss Problem with Branch*-dan-*Price*-dan-*Cut.* Portugal: Universidade do Minho

Bazarra, Mokhtar, S dan John J. Jarvis.1977. *Linear Programming* dan *Network Flows*. Canada: Simultaneously.

Burke, E. K., C. Cordier, H. Marchand, R. Laundy, L. A. Wolsey. 2004. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, *52*(4), 655–671.

Nitsche, G. Scheithauer, J. Terno. 1999. Tighter relaxations for the cutting stock problem, *European Journal of Operational Research*, 112 (1), 654-663.

De Gelder, E. R., dan Wagelmans, a. P. M. 2009. The two-dimensional cutting stock problem within the roller blind production process. *Statistica Neerl*dan*ica*, *63*(4), 474–489.

Eiselt, H.A . 2007. *Linear Programming and its Applications*. Berlin: Springer.

Erma, S. 2014. *Integer Linear Programming (ILP) Sebagai Metode Matematis dalam Penyelesaian Permasalahan Cutting Stock di PT. Pusaka Prima Mandiri*. Medan : Universitas Negeri Medan.

Fukunaga, K.1990. *Introduction to Statistical Pattern Recognition second ed*. USA: Academic Press, Inc.

Gilmore, P.C dan Gomory, R.E.1961. A Linear Progamming Approach To The Cutting Stock Problem. *Operational Research*,9(1):849-589.

Gilmore, P.C dan Gomory, R.E.1963. A Linear Progamming Approach To The Cutting Stock Problem-Part II. *Operational Research,* 11(1):863-888.

Goulimis, C. 1990. Solusi optimal untuk masalah saham pemotongan. *European Journal of Operational Research* 44(1): 197-208.

Hillier, Frederick dan Gerald Lieberman. 1995. *Pengantar Riset Operasi, 5th Edition*, Jakarta:Erlangga.

Hinxman, A. I. 1980. The trim-loss dan assortment problems: A survey. *European Journal of Operational Research*, 5, 8–18.

Junger, M. dan S. Thienel. 2000. *The ABACUS system for branch and cut and price algorithm in integer linear programming*, Software: Practice and Experience, 30(11), 1325-1352.

Kantorovitch, L.V. 1962. *Mathematical method of organizing* dan *planning production. reprinted in Management Sci, 6,* 366-422.

Luenberger, David G. 1973.*Linier* dan *Nonlinear Programming,2nd edition*. Canada : Addison-Wesley Publishing Company, Inc.

Lodi, A. dan M. Monaci. 2002. Integer linear programming model for 2-staged two dimensional knapsack problem, *Mathematical Programming, Ser. B*, 94(1), 173-187.

Matsumoto, K., Umetani, S., dan Nagamochi, H. 2010. On the one-dimensional stock cutting problem in the paper tube industry. *Journal of Scheduling*, 14(3), 281–290.

Pal, S. 2005. Improving Branch and Price Algorithms for Solving One Dimensional Cutting Stock Problem. *MTech Seminar Report*. Mumbai:Indian Institute of Technology.

Stephen, N. dan Ariela Sofer. 1996. *Linier* dan *Nonlinear Programming*. Singapore : McGraw-Hill Inc.

Subagyo, Pangestu, Marwan Asri, dan T. Hani Hdanoko. 1985. *Dasar-dasar Operations Research, edisi ke-2*. Yogyakarta: BPF.

Sumarto, Rumasari Hadi dan Lukas Dwiantara. 2006. *Sekretaris Profesional*. Yogyakarta : Kanisius

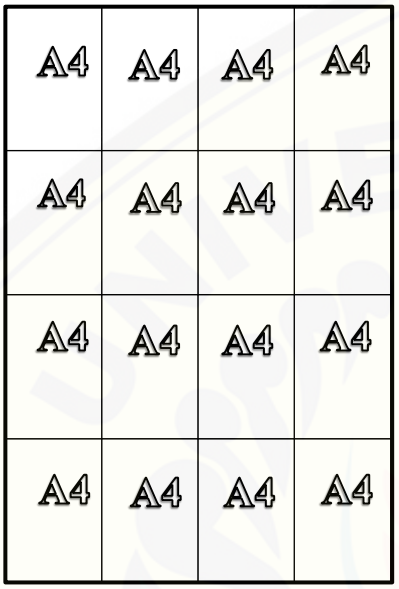
Taha, H A. 2007. *Riset Operasi : Suatu Pengantar*, Edisi ke-5. Jakarta: Binarupa Aksara.

Tjutju, T dan Ahmad Dimyati. 2003. *Operations Research: Model-model Pengambilan Keputusan*. Bandung: Sinar Baru Algensindo.

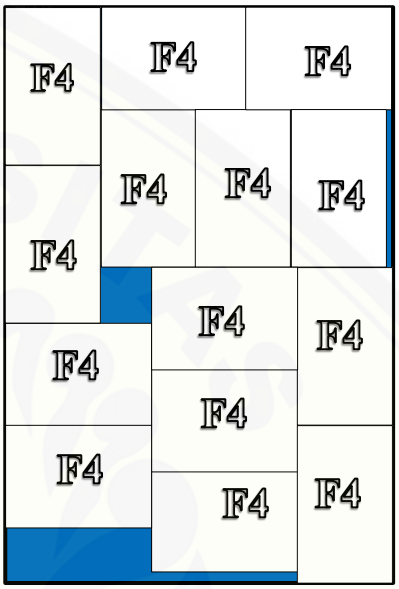
Triyanti, Razaullah, S.Rehman, I. Hussain. 2008. *Usulan Perbaikan Metode Pemilihan Alternatif Pemotongan Roll Dengan Integer Linier Programming (Studi Kasus : PT Pelita Cengkareng Paper & CO, Tanggerang).* Medan: Universitas Sumatera Utara.

LAMPIRAN

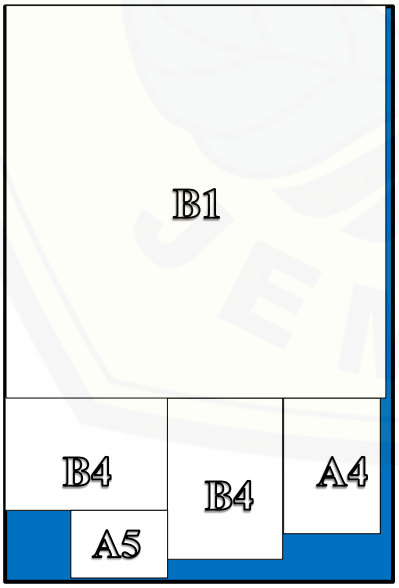
A. Gambar Pola Pemotongan Kertas



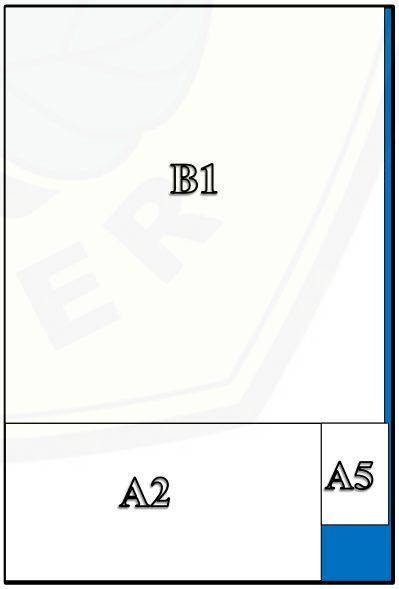
Pola Pemotongan 1



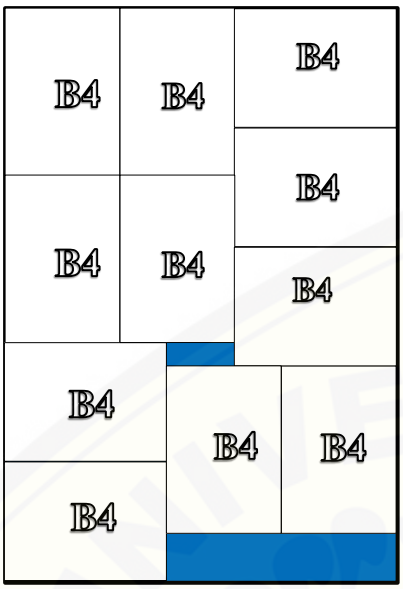
Pola Pemotongan 2



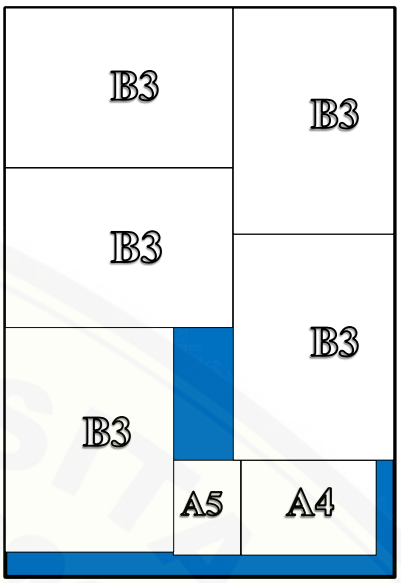
Pola Pemotongan 3



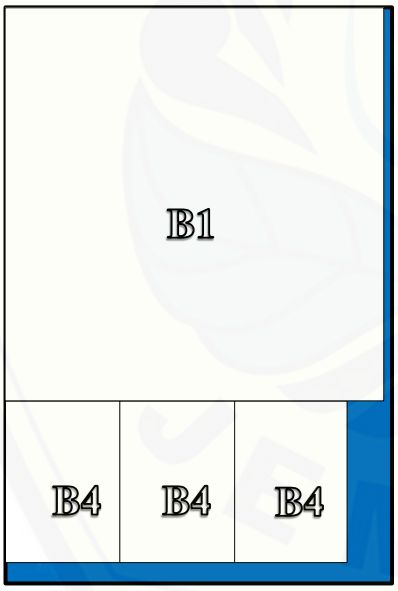
Pola Pemotongan 4



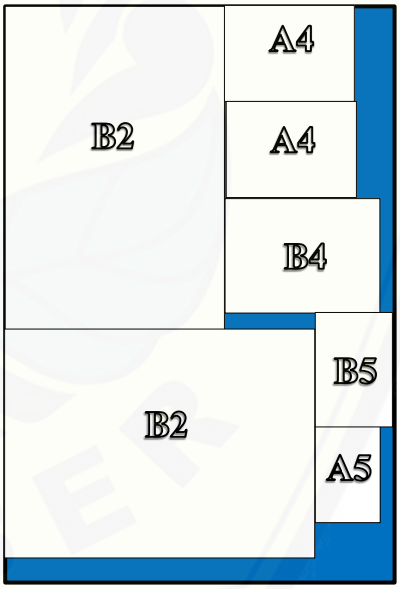
Pola Pemotongan 5



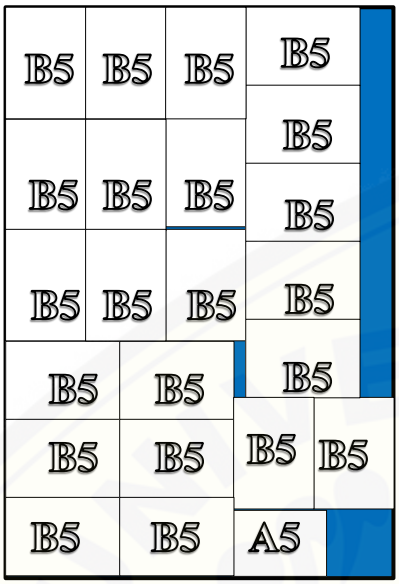
Pola Pemotongan 6



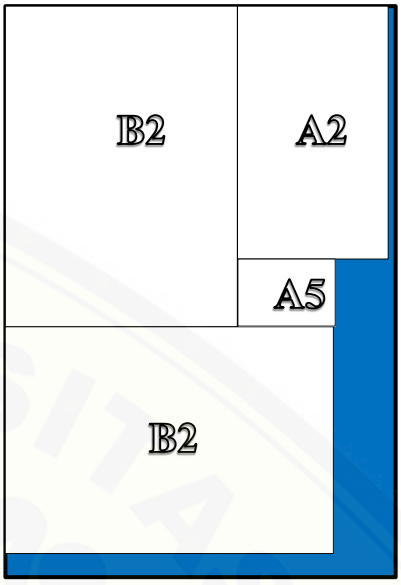
Pola Pemotongan 7



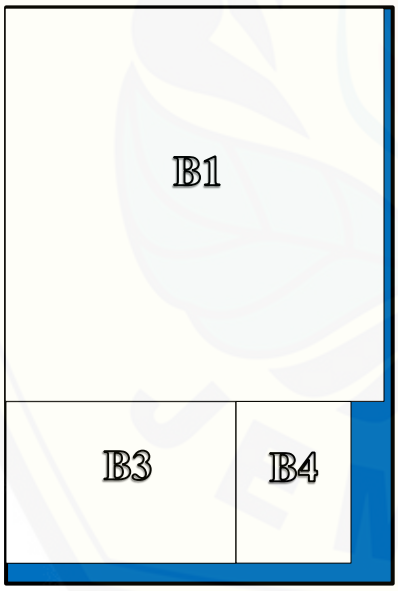
Pola Pemotongan 8



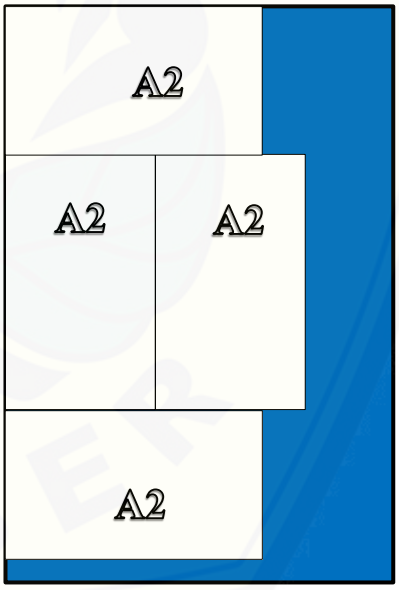
Pola Pemotongan 9



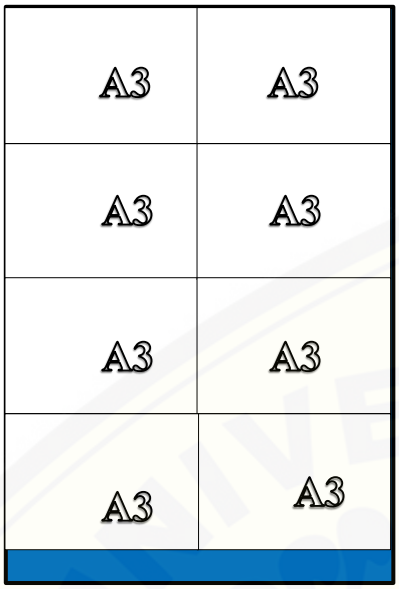
Pola Pemotongan 10



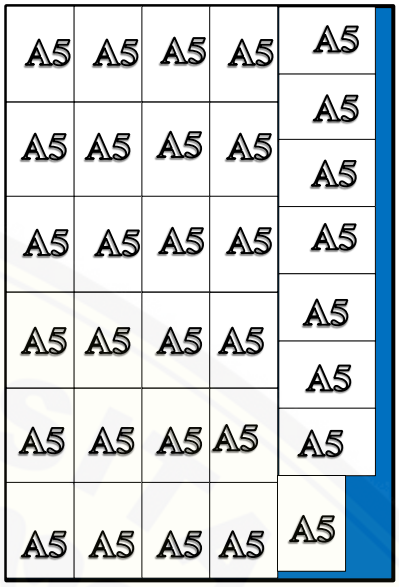
Pola Pemotongan 11



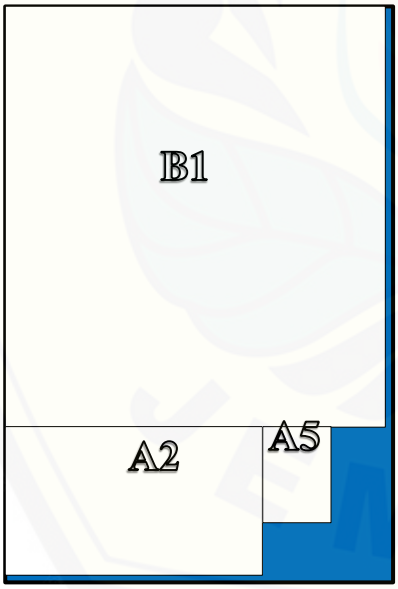
Pola Pemotongan 12



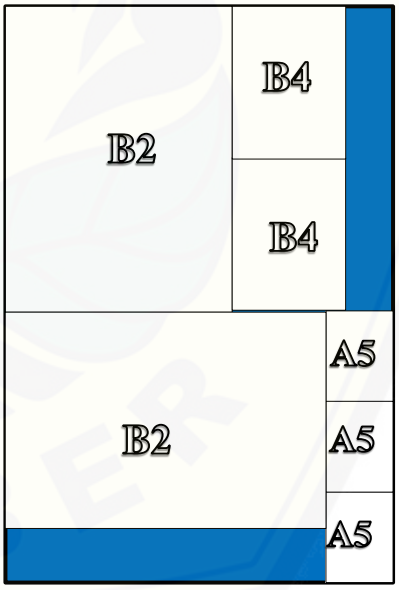
Pola Pemotongan 13



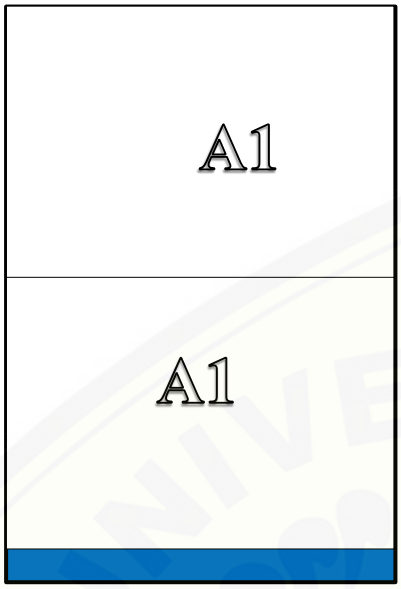
Pola Pemotongan 14



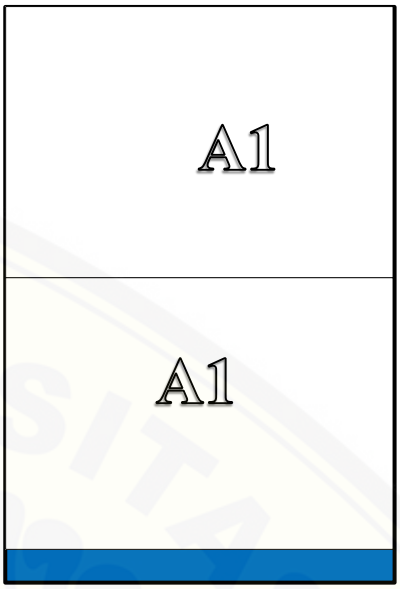
Pola Pemotongan 15



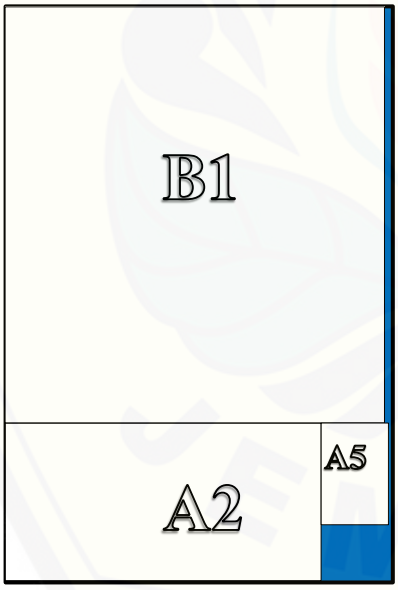
Pola Pemotongan 16



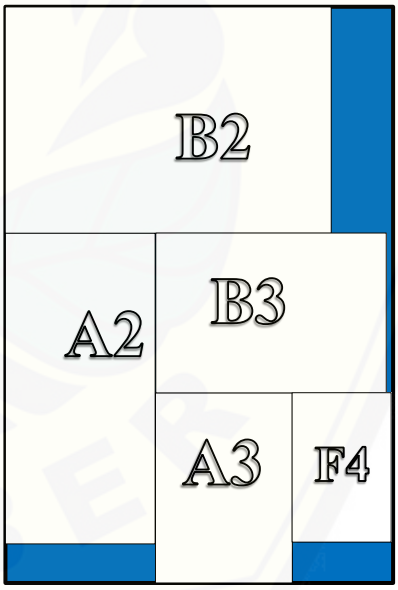
Pola Pemotongan 17



Pola Pemotongan 18



Pola Pemotongan 19



Pola Pemotongan 20

B. Pola Pemotongan Kertas

Pola (*j*)	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	F4	Sisa
1	0	0	0	16	0	0	0	0	0	0	0	20,29
2	0	0	0	0	0	0	0	0	0	0	14	66,49
3	0	0	0	1	1	1	0	0	2	0	0	229,99
4	0	1	0	0	1	1	0	0	0	0	0	123,99
5	0	0	0	0	0	0	0	0	11	0	0	291,99
6	0	0	0	1	1	0	0	5	0	0	0	239.99
7	0	0	0	0	0	1	0	0	3	0	0	281,99
8	0	0	0	2	1	0	2	0	1	1	0	48,79
9	0	0	0	0	1	0	0	0	0	22	0	8,69
10	0	1	0	0	1	0	2	0	0	0	0	123,89
11	0	0	0	0	0	1	0	1	1	0	0	281,99
12	0	4	0	0	0	0	0	0	0	0	0	20,29
13	0	0	8	0	0	0	0	0	0	0	0	20,29
14	0	0	0	0	32	0	0	0	0	0	0	53,89
15	0	1	0	0	1	1	0	0	0	0	0	123,89
16	0	0	0	0	3	0	2	0	2	0	0	232,09
17	2	0	0	0	0	0	0	0	0	0	0	8,41
18	2	0	0	0	0	0	0	0	0	0	0	4,41
19	0	1	0	0	1	1	0	0	0	0	0	123,89
20	0	1	1	0	0	0	1	1	0	0	1	247,79

C. Script GUI Program Optimasi Pemotongan Kertas Menggunakan Algoritma *Branch and Bound* dan *Gilmore-Gomory*

```
clc; clear all;
close all;
set(0,'Units','points')
Screen = get(0,'screensize');
pos=[0 0 500 480];
ulang=0;

win2=figure(...
'units','points',...
'position',[Screen(3:4)/2-pos(3:4)/2 pos(3:4)],...
'color',[.19 .9 .9],...
'resize','off',...
'menubar','none',...
'toolbar','none',...
'numbertitle','off',...
'name','Branch and Bound dan Gilmore Gilmory');

%====================================
label1=uicontrol('parent',win2,...
    'units','points',...
    'position',[10 400 480 75],...
    'style','text',...
    'string',{'PENYELESAIAN PERMASALAHAN PEMOTONGAN KERTAS';
    'ALGORITMA BRANCH AND BOUND DAN ';
    'ALGORITMA GILMORE GOMORY';'';
    'Oleh: Lailatul Putri Suryaningtyas'},...
    'fontname','times new roman','BackgroundColor',[.9 .9
.5],...
    'fontsize',12,'fontweight','bold');

%=======================================================
hp0 = uipanel('parent',win2,...
             'Title','Input data','FontSize',12,...
             'units','points',...
             'fontweight','bold',...
             'BackgroundColor',[.8 .8 .0],...
             'Position',[10 310-0 480 85]);
tabel02 = uitable('Parent',win2,...
    'units','point',...
    'hitTest','on',...
    'backgroundcolor',[1 1 .5; .5 1 1],...

'ColumnEditable',true,'columnwidth',{40,40,40,40,40,40,40,40,4
0,40,40},...
    'fontname','times new roman','data',zeros(1,11),...
    'foregroundcolor',[0 0 0],'rowname',{'Demand'},...

'fontsize',10,'columnname',{'A1','A2','A3','A4','A5','B1','B2'
,'B3','B4','B5','F4'},...
    'Position',[15 340+0 400 35]);
label1=uicontrol('parent',win2,...
    'units','points',...
```

```
    'position',[15 315-0 100 15],...
    'style','text','horizontalalignment','left',...
    'string','Jumlah Pola: ',...
    'fontname','times new roman','BackgroundColor',[.8 .8
.0],...
    'fontsize',12,'fontweight','bold');
edit11=uicontrol('parent',win2,...
    'units','points',...
    'position',[80 315-0 80 18],...
    'style','edit','horizontalalignment','left',...
    'string','0',...
    'fontname','times new roman',...
    'fontsize',12,'fontweight','bold');

label1=uicontrol('parent',win2,...
    'units','points',...
    'position',[195 315-0 100 15],...
    'style','text','horizontalalignment','left',...
    'string','Jumlah Master: ',...
    'fontname','times new roman','BackgroundColor',[.8 .8
.0],...
    'fontsize',12,'fontweight','bold');
edit12=uicontrol('parent',win2,...
    'units','points',...
    'position',[280 315-0 80 18],...
    'style','edit','horizontalalignment','left',...
    'string','0',...
    'fontname','times new roman',...
    'fontsize',12,'fontweight','bold');

proses2=uicontrol('parent',win2,...
    'units','points',...
    'position',[395 315-0 60 15],...
    'style','Pushbutton',...
    'callback','proses',...
    'string','OK',...
    'fontname','times new roman',...
    'fontsize',12);
%============================================================

hp0 = uipanel('parent',win2,...
            'FontSize',11,...
            'units','points',...
            'fontweight','bold',...
            'BackgroundColor',[.8 .8 .0],...
            'Position',[10 130+0 480 175]);
tabel01 = uitable('Parent',win2,...
    'units','point',...
    'hitTest','on',...
    'backgroundcolor',[1 1 .5; .5 1 1],...
    'ColumnEditable',true,...
    'fontname','times new roman',...
    'foregroundcolor',[0 0 0],...
    'fontsize',10,...
    'Position',[15 160+0 470 140]);
```

```
set(tabel01,'columnname',{'A1','A2','A3','A4','A5','B1','B2','
B3','B4','B5','F4','Sisa'});

proses3=uicontrol('parent',win2,...
    'units','points',...
    'position',[350 135+0 60 15],...
    'style','Pushbutton',...
    'callback','hasil1',...
    'string','Proses ',...
    'fontname','times new roman',...
    'fontsize',12);
proses4=uicontrol('parent',win2,...
    'units','points',...
    'position',[415 135+0 60 15],...
    'style','Pushbutton',...
    'callback','RISET',...
    'string','Reset ',...
    'fontname','times new roman',...
    'fontsize',12);

% label31=uicontrol('parent',win2,...
%     'units','points',...
%     'position',[245 210-0 100 15],...
%     'style','text','horizontalalignment','left',...
%     'string','Metode: ',...
%     'fontname','times new roman','BackgroundColor',[.8 .8
.0],...
%     'fontsize',12,'fontweight','bold');
step1 =
uicontrol('Style','checkbox','units','points','String','Branch
And Bound','fontsize',11,...
    'backgroundcolor',[.8 .8 .0],...
    'pos',[15 135 120 20],'parent',win2);
step2 =
uicontrol('Style','checkbox','units','points','String','Gilmor
e Gomoory','fontsize',11,...
    'backgroundcolor',[.8 .8 .0],...
    'pos',[150 135 120 20],'parent',win2);

%============================================================

hp0 = uipanel('parent',win2,...
            'FontSize',11,...
            'title','Output',...
            'units','points',...
            'fontweight','bold',...
            'BackgroundColor',[.8 .8 .0],...
            'Position',[10 5 480 120]);
label_output1=uicontrol('parent',win2,...
    'units','points',...
    'position',[15 10 230 100],...
    'style','listbox','horizontalalignment','left',...
    'string',{'Metode Branch And Bound';'Jumlah Master:
';'Pola: ';'Jumlah: ';'Hasil Produksi: ';'Luas Sisa:
';'Keterangan: '},...
    'fontname','times new roman',...
```

```
    'fontsize',12,'fontweight','bold');

label_output2=uicontrol('parent',win2,...
    'units','points',...
    'position',[250 10 230 100],...
    'style','listbox','horizontalalignment','left',...
    'string',{'Metode Gilmore Gomoory';'Jumlah Master:
';'Pola: ';'Jumlah: ';'Hasil Produksi: ';'Luas Sisa:
';'Keterangan: '},...
    'fontname','times new roman',...
    'fontsize',12,'fontweight','bold');

set(tabel02,'data',[0 0 0 0 0 0 0 0 0 0 0]);

menu11=uimenu('parent',win2,...
    'Label','Open Pola','callback','open_file');
menu12=uimenu('parent',win2,...
    'Label','Save
Pola','callback','simpan_pola','enable','off');
menu13=uimenu('parent',win2,...
    'Label','Hitung
Manual','callback','Hitung_Manual','enable','off');
```