

**FACULDADES INTEGRADAS DE TAQUARA
CURSO SISTEMAS DE INFORMAÇÃO**

**AVALIAÇÃO E APLICAÇÃO DE ALGORITMOS DE RECORTE
PARA GRÁFICAS DE ETIQUETAS PARA CALÇADOS**

CRISTINA RIBEIRO GRAEFF

**Taquara
2010**

CRISTINA RIBEIRO GRAEFF

**AVALIAÇÃO E APLICAÇÃO DE ALGORITMOS DE RECORTE
PARA GRÁFICAS DE ETIQUETAS PARA CALÇADOS**

Trabalho de Conclusão apresentado ao Curso de Sistemas de Informação das Faculdades Integradas de Taquara, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação em novembro de 2010, sob orientação do Prof. Ms. Eurico Jardim Antunes.

**Taquara
2010**

RESUMO

Este trabalho apresenta alguns algoritmos para a solução do problema do corte bidimensional, para peças retangulares. Foram aplicadas três heurísticas, método GRASP, algoritmos genéticos e o método de geração de colunas, analisadas empiricamente, isto é, através de experimentos computacionais realizados para um conjunto de entradas. Foram realizadas comparações entre os resultados obtidos pela execução dos algoritmos, seguindo os critérios de avaliação definidos ao longo deste trabalho. Utilizando como cenário uma indústria gráfica de etiquetas para calçado.

Palavras-chave: Indústria Gráfica, Corte bidimensional, Algoritmos.

ABSTRACT

This paper is based on algorithms to solve the two-dimensional cut problem to rectangular pieces. Three heuristics were applied, GRASP method, genetic algorithms and the column generated method, empirically analyzed, that is, through computational experiments realized to a group of entries. Comparisons were performed between the results obtained from the execution of algorithms, following the evaluation criteria defined throughout this paper. Utilization as scenery one graphic industry of etiquette to shoes.

Key words: *Graphic industry, two-dimensional cut, algorithms.*

Lista de Figuras

Figura 1: Exemplo de tabelas de corte, para o formato original de 66 x 96 cm.....	12
Figura 2: Descrição básica do AG.....	21
Figura 3: Estrutura genérica de um AG.....	21
Figura 4: Algoritmo GRASP.....	24
Figura 5: Algoritmo Next-fit.....	25
Figura 6: Algoritmo Best-fit.....	25
Figura 7: Algoritmo First-fit.....	26
Figura 8: Ilustração do Bottom-Left.....	26
Figura 9: Ilustração chapa e peça.....	31
Figura 10: Números de cortes e possibilidades de padrões.....	32
Figura 11: Material não aproveitado.....	33
Figura 12: Função objetivo.....	35
Figura 13: Definição de faixas.....	35
Figura 14: Diagrama de casos de uso.....	36
Figura 15: Diagrama de classes.....	37
Figura 16: Interação do método GRASP.....	39
Figura 17: Estrutura do método GRASP.....	40
Figura 18: Estrutura do algoritmo genético aplicado ao cenário proposto.....	43
Figura 19: Início da estrutura da árvore.....	44
Figura 20: Estrutura do método geração de colunas.....	45
Figura 21: Layout da tela.....	46
Figura 22: Layout da área de log.....	46
Figura 23: Exemplos de visualização dos padrões.....	47

Lista de Tabelas

Tabela 1: Relação dos percentuais adotados para escolha.....	35
Tabela 2: Exemplos de fitness.....	40
Tabela 3: Tabela de resultados algoritmo genético.....	49
Tabela 4: Tabela de resultados algoritmo GRASP.....	50
Tabela 5: Tabela de resultados algoritmo geração de colunas.....	51

Lista de Siglas e Abreviaturas

ABIEA	Associação Brasileira das Indústrias de Etiquetas Adesivas
AG	Algoritmo Genético
AG's	Algoritmos Genéticos
AM	Algoritmo Memético
DIN	<i>Deutsch Industrien Normen</i>
ERP	Planejamento de Recursos Empresariais
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
PI	Programação Linear Inteira
PL	Programação Linear
RAD	<i>Rapid Application Development</i>
TCP/IP	Protocolo de Controle de Transmissão/Protocolo de Internet

Sumário

1	INTRODUÇÃO.....	9
2	REFERENCIAL TEÓRICO.....	11
2.1	Contexto das indústrias gráficas.....	11
2.1.1	Problemas no corte bidimensional para peças retangulares.....	13
2.2	Aplicações comerciais para corte bidimensional	13
2.3	Métodos e algoritmos.....	14
2.3.1	Técnicas de resolução.....	16
2.3.1.1	Programação linear.....	16
2.3.1.2	Programação linear inteira.....	17
2.3.1.3	Método de geração de colunas.....	18
2.3.1.4	Algoritmos genéticos.....	20
2.3.1.5	Método GRASP.....	23
2.3.2	Principais técnicas de encaixe para o problema do corte.....	24
2.4	Análise de algoritmos.....	27
2.5	Tecnologias utilizadas para o desenvolvimento.....	28
2.5.1	Linguagens de programação.....	28
2.5.2	Escolha da linguagem.....	30
3	METODOLOGIA.....	31
3.1	Métricas adotadas para decisão do melhor algoritmo.....	32
3.2	Escolha dos algoritmos a serem utilizados.....	33
4	DESENVOLVIMENTO.....	35
4.1	Metodologias de desenvolvimento.....	35
4.2	Desenvolvimento do método GRASP.....	37
4.3	Desenvolvimento do algoritmo genético.....	40
4.4	Desenvolvimento do método de geração de colunas.....	43
4.5	Funcionamento, telas e visualização dos padrões.....	45
5	TESTES E RESULTADOS OBTIDOS.....	48
6	CONCLUSÃO.....	52
	REFERÊNCIAS.....	53

1 INTRODUÇÃO

No processo produtivo de uma empresa inúmeras etapas devem ser cumpridas, cada uma com suas metas e objetivos, desenvolvendo assim um produto para o mercado apto a competir com seus concorrentes. O custo torna-se um fator competitivo e faz-se necessário minimizar as perdas. Pensando no processo de produção de etiquetas para empresas calçadistas, observa-se uma deficiência na área do corte de material, onde o mesmo é realizado sem uma avaliação mais detalhada, sobre como esse corte deveria ser realizado, para aproveitar ao máximo a folha.

Esse setor do processo de produção de etiquetas em indústrias gráficas, não conta com um apoio de um sistema que realize esse cálculo, de forma a encontrar as soluções mais aproximadas, para a realização dos cortes. Na tentativa de reduzir as perdas, que são contabilizadas pela área administrativa como prejuízos no processo, serão estudados e aplicados alguns algoritmos, com o objetivo de escolher o que melhor se adapte a situação da empresa.

O problema do corte de peças retangulares consiste em cortar placas retangulares em itens menores, em quantidades e tamanhos específicos, com o objetivo de minimizar as perdas, a quantidade de material a ser cortado, o custo, e consequentemente maximizarem os lucros, a solução para esse problema passa a ser um diferencial, que pode ser aplicado ao corte de materiais diversos, como vidro, chapa de metal, peças de couro, e em nosso caso específico papeis, adesivos e vinis.

A partir da necessidade de automatizar um processo manual, que retorne benefícios para a empresa, como a redução nas perdas de materiais utilizados para a produção de etiquetas, foi realizada uma pesquisa pelos *softwares*¹ e algoritmos já aplicados a essa situação.

No mercado atual existem aplicações que realizam o cálculo para o corte de materiais, como tecidos, madeira e vidro, porém são sistemas comerciais que não disponibilizam detalhes da implementação dos algoritmos. Estes softwares possuem um custo elevado, sendo inviável a aquisição por empresas menores. Como

1 Software = Programa de computador.

exemplos de sistemas comerciais temos o *software Audaces Planos de Corte* da empresa Audaces e *Corte Certo* da empresa Dimensions Softwares.

Através das pesquisas realizadas sobre o problema do corte bidimensional, observou-se uma grande quantidade de trabalhos relacionados ao tema, cada um com métodos diferentes para atingir o mesmo objetivo, encontrar o padrão de corte próximo à solução ótima. As referências pesquisadas sugerem a avaliação de outros tipos de algoritmos, realizando um comparativo entre os métodos. Assim foram escolhidos os algoritmos que através de experimentos práticos serão aplicados e avaliados, para o problema do corte de peças relacionado ao cenário das Gráficas de Etiquetas para Calçados.

Neste trabalho foram aplicados três métodos que serão detalhados no decorrer na seção 4. Para a visualização dos resultados foi necessária a construção de algumas telas para evidenciar o padrão de corte encontrado.

O objetivo deste trabalho é chegar a uma constatação prática de qual algoritmo melhor comportou-se, para o problema apresentado. Buscando contribuir para que a empresa tenha um melhor aproveitamento de seus materiais em um tempo razoável.

Com a avaliação dos algoritmos será possível disponibilizar ao cliente um produto que realmente irá atender suas necessidades, pois todo o processo de desenvolvimento e testes será realizado com as informações reais da empresa, visando o melhor desempenho e resultados obtidos.

Este trabalho está estruturado da seguinte forma: na seção 2 são apresentadas as referências bibliográficas; na seção 3 aborda-se a metodologia aplicada; na seção 4 são apresentados os detalhes do sistema desenvolvido, como descrição de cada algoritmo aplicado, interface das telas de testes e funcionamento do sistema; na seção 5 são exibidos os resultados e, finalmente, tem-se a conclusão sobre o que foi constatado ao final deste trabalho.

2 REFERENCIAL TEÓRICO

Nesta seção são apresentados os fundamentos e conceitos relacionados ao desenvolvimento do produto, um breve contexto da indústria gráfica e do problema do corte bidimensional, citações de softwares existentes, algoritmos e linguagens.

O processo de desenvolvimento do produto é um dos mais importantes dentro de uma empresa, pois pode tornar a empresa mais competitiva no mercado. Assim qualquer melhora nesse processo retorna grandes vantagens à indústria. Durante o processo de produção inúmeros erros são cometidos e devem ser observados, afim de não serem repetidos, a visão de todo o processo do ciclo produtivo torna-se um dos motivos para o gerenciamento de desenvolvimento do produto (ROZENFELD; AMARAL; TOLEDO; CARVALHO, 2006).

As informações manipuladas durante a cadeia produtiva, têm destaque cada vez maior no processo, são sistemas complexos que disponibilizam informações sobre o produto. Os sistemas ERP são exemplos reais de sistemas que auxiliam a empresa, dando suporte as suas diversas áreas, esses sistemas foram baseados no processo de integração entre os setores. Hoje existe a necessidade de soluções para todo o processo de desenvolvimento do produto. Assim faz-se necessário o desenvolvimento de sistemas específicos para as tarefas produtivas (ROZENFELD; AMARAL; TOLEDO; CARVALHO, 2006).

2.1 Contexto das indústrias gráficas

A indústria gráfica já possui 200 anos no mercado brasileiro, porém isso somente foi possível devido à invenção e refinamento de técnicas para a fabricação do papel na China. Todo o avanço tecnológico na indústria se deu após aprimoramento destas técnicas. Com o crescente mercado em 1965 foi criada a Associação Brasileira da Indústria Gráfica (PRONEWS, 2008) e em 1986 a Associação Brasileira das Indústrias de Etiquetas Adesivas (ABIEA, 2009).

O corte do papel nas gráficas é realizado por um operador, utilizando uma guilhotina, o fornecedor do papel já oferece o material necessário em formatos que

facilitam o processo de corte, com tabelas de corte já estipuladas (BAER, 1995).

No ano de 1911, a Associação de Engenheiros Alemães, visando à economia de papel e a racionalização da mão de obra, criaram-se normas de formatos baseado em um sistema métrico, denominado DIN. Os tamanhos padronizados foram calculados, para que as folhas tenham a mesma proporção quando realizada a dobra. No Brasil adotaram-se os padrões de folhas em formatos AA (76 x 112 cm), BB (66 x 96 cm) e AM (87 x 114 cm), porém existem outros formatos, como o papelão que mede 80 x 100 cm (NETO, 1997).

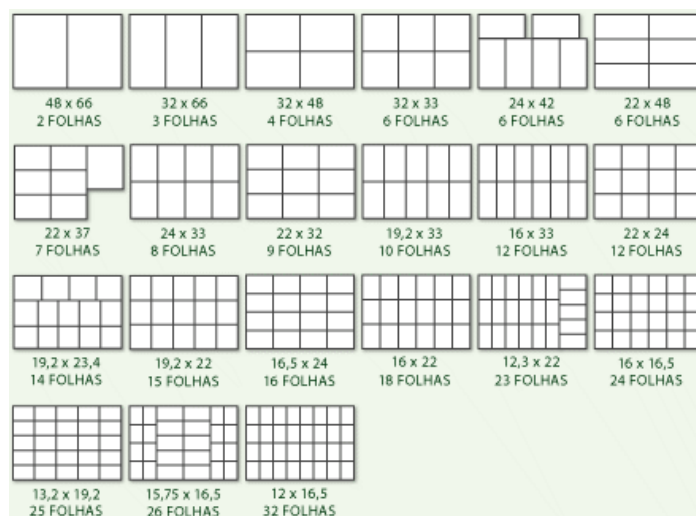


Figura 1: Exemplo de tabelas de corte, para o formato original de 66 x 96 cm.

Fonte: KSR (2009, p. 1)

Alguns sites existentes de fornecedores de materiais (papel, adesivo e vinil), disponibilizam uma funcionalidade na própria página, para cálculo de aproveitamento de papel de formas retangulares, porém não são consideradas nestes cálculos as perdas (sobras), características desse tipo de cálculo, por não se tratar sempre de um formato único. Os materiais são fornecidos em bobinas ou folhas, com tamanhos que facilitam o aproveitamento (KSR, 2009).

Dos problemas relacionados ao corte do papel, assim como diversas outras matérias primas podem-se destacar estudos no corte guilhotinado, peças unidimensionais e peças bidimensionais retangulares, mencionadas na próxima seção.

2.1.1 Problemas no corte bidimensional para peças retangulares

Assim como o problema do empacotamento (ANDRADE, 2006), onde vários itens de formatos diferentes devem ser dispostos em um determinado espaço, o mesmo problema ocorre para o corte de peças retangulares. Estes são problemas bastante estudados e requerem um grau de aproximação dos resultados obtidos, buscando a melhor solução. Nos problemas relacionados ao corte os itens possuem uma demanda maior, onde se pode necessitar de vários itens do mesmo tamanho para a produção, diferente do problema do empacotamento que possui demanda unitária. No primeiro estágio do corte temos dois tópicos iniciais a serem observados, os itens não podem se sobrepor e devem caber inteiramente na área proposta (ANDRADE, 2006).

O problema do corte bidimensional (duas dimensões) consiste em cortar placas de tamanho fixo em itens menores, otimizando assim um processo de cálculo. Como as indústrias recebem esse material em formatos maiores, existe a necessidade de realizar um novo cálculo, para utilização do material em peças menores. Através das entradas do tamanho do material (x e y) e do tamanho da demanda necessária (a e b), pode-se realizar o enquadramento do tamanho final ao tamanho inicial, obtendo a quantidade de itens cortados (ZIMMERMANN, 2002).

2.2 Aplicações comerciais para corte bidimensional

O sistema *Corte Certo* (DIMENSIONS SOFTWARES, 2009) foi criado para otimizar os planos de corte, buscando atender as exigências dos vários tipos de materiais. Com uma integração aos sistemas já existentes de cálculo para esquadrias metálicas, utilizado para chapas de vidro, madeira, acrílico, mármore, gesso acartonado e plástico. Entre suas várias versões disponibilizadas se encontra uma opção para impressão de etiquetas, com uma tela para configuração e ajustes do corte. Como complemento as suas funcionalidades possui impressão de relatórios, a possibilidade de salvar os cálculos efetuados, redução no desperdício de material, aumento da produtividade, planejamento de compra e venda do material

e planejamento de corte.

Audaces Planos de Corte (AUDACES, 2009), tem como funcionalidade a geração automática de corte para peças retangulares. São informadas as dimensões das peças e a matéria prima (chapa, tubo ou bobina) e o programa encontra um plano de corte com perdas mínimas. Possui versões específicas para a indústria de estofados, móveis, metal mecânica, papel e vidros.

Outros sistemas similares para corte de peças: *Otimizar Cortes*² e *RZ CAD Têxtil*³, utilizado para corte em tecidos.

Os sistemas apresentados são comerciais e não disponibilizam seu método de desenvolvimento, não sendo possível identificar qual a heurística aplicada para a geração dos padrões. Contudo já existem vários trabalhos acadêmicos relacionados a essa área, o que possibilitou o levantamento realizado na próxima seção.

2.3 Métodos e algoritmos

Devido à dificuldade em encontrar algoritmos exatos, que resolvam o problema em tempo hábil de execução, faz-se necessário utilizar algoritmos heurísticos e aproximativos, que encontram soluções próximas da solução ótima, com um tempo de execução menor. Geralmente esse tipo de problema se preocupa em minimizar as perdas ou utilizar o máximo do material disponível (FARIA 2006).

Dos estudos realizados nesta área, vários fazem referência à utilização da programação linear inteira, que são estruturados igualmente aos de programação linear, onde caso todas as variáveis do sistema pertençam aos números inteiros, temos assim uma classe da programação linear chamada de programação inteira (PI) (FARIA 2006).

Zimmermann (2002) utilizou o método de programação linear, propondo para a solução duas técnicas a serem utilizadas: Branch-and-Bound, mais conhecida como ramificação e limite, e um método que pode ser utilizado nas técnicas de corte denominado geração de colunas.

2 Otimizar Cortes = www.otimizecortes.com

3 RZ CAD Têxtil = www.rzsistemas.com.br

A geração de colunas muitas vezes é utilizada juntamente com um algoritmo de branch-and-bound, e recebem juntos o nome de algoritmo de branch-and-price (ANDRADE, 2006, p. 22).

O número de interações e resultados possíveis em algoritmos para o problema do corte pode retornar um intenso número de combinações possíveis. Os algoritmos Branch-and-Bound, são algoritmos de enumeração e realizam a exclusão das piores soluções encontradas, juntamente com suas variações. Esse algoritmo delimita a área de soluções possíveis, utilizando dois limitantes que são os valores máximo e mínimo do problema (primal e dual), recalculando esses limites até que iguais, podendo assim afirmar que a solução encontrada é a ideal, ou ainda que a sua diferença seja a mínima possível, limitando o espaço de busca do algoritmo. É construída uma árvore, onde os nós representaram uma parte de um conjunto de soluções, um fator para o desempenho deste algoritmo é cálculo dos limitantes, onde é realizado o corte dos ramos, diminuindo assim o tamanho da busca. O algoritmo branch-and-price, é muito utilizado para problemas de empacotamento, corte unidimensional e bidimensional, realizados através de implementação da busca em árvore e utilizando a geração de colunas em cada nó de busca (ANDRADE, 2006).

O algoritmo First Fit Decreasing, atende e resolve o problema do corte, este algoritmo tem como idéia principal a criação de uma fila de itens, para serem encaixados. Os itens são testados, cada item que não puder ser encaixado em uma das soluções existentes é criado uma nova solução (FARIA, 2006).

Ao longo dos anos a utilização de algoritmos genéticos para solução de problemas de cálculo tem se expandido. Os Algoritmos genéticos são algoritmos de busca baseados na genética e mecanismos de seleção natural, programam formas de busca paralela e aleatória para solucionar problemas de otimização, com uma população inicial de soluções o algoritmo evolui através de operadores genéticos realizando cruzamentos e mutações (FARIA, 2006).

Os algoritmos meméticos (AM) constituem uma nova classe dos algoritmos evolutivos, assim como os algoritmos genéticos também se baseiam no processo de mutação e reprodução. No AM parte da população é formada por agentes, que exploram independentes os espaços de busca. Andrade (2006) aplicou algoritmos

meméticos juntamente com algoritmo Best-Fit para o problema do corte bidimensional.

2.3.1 Técnicas de resolução

Existem várias técnicas para a solução de problemas de otimização, algumas destas serão mencionadas nas próximas seções.

2.3.1.1 Programação linear

A programação linear (PL) é uma técnica matemática bastante utilizada para problemas de otimização, que permite encontrar a solução ótima para certo tipo de problema. Os primeiros conceitos da programação linear foram desenvolvidos durante a II Guerra Mundial e aplicados a programas militares. Com o objetivo de maximizar lucros ou minimizar custos, buscando uma distribuição eficiente dos recursos.

Na PL existem algumas restrições lineares, um poliedro convexo forma o conjunto de pontos viáveis. Existem duas situações na qual uma solução ótima não pode ser encontrada, se as restrições se contradizem, logo a região é vazia e neste caso a PL é inviável, o poliedro pode ser ilimitado na direção da função objetivo, neste caso não existe solução.

Pode-se decompor o processo de organização de um modelo de PL nas seguintes etapas:

- a) definição das atividades;
- b) definição dos recursos;
- c) cálculo dos coeficientes de insumo / produção;
- d) determinação das condições externas;
- e) formalização do modelo;

Existem vários métodos para resolução de programação linear. Entre eles, os três mais conhecidos são: o método simplex, o método de pontos interiores e o método dos elipsóides. O simplex é um algoritmo que busca a solução ótima de um problema de programação linear, em linhas gerais o algoritmo parte de uma solução viável e a partir desta solução inicial vai identificar novas soluções de valor igual, ou melhor, que a primeira encontrada (GOLDBARG; LUNA, 2000).

2.3.1.2 Programação linear inteira

Se todas as variáveis do problema pertencerem ao conjunto dos números inteiros têm-se uma subclasse da programação linear, a programação linear inteira (PI). Os algoritmos descritos abaixo são aplicados na programação linear inteira:

Branch-and-Bound: É uma estratégia de divisão e conquista para problemas de natureza inteira mista. O algoritmo branch-and-bound é também conhecido como algoritmo de enumeração implícita, pois utilizam os princípios de exclusão das soluções ruins, ou seja, realiza o corte dos ramos que não atenderem o critério definido (ZIMMERMANN, 2002).

Esse algoritmo calcula dois limitantes (primais e duais), delimitam os valores que uma solução pode atingir, o objetivo é recalcular esses limitantes até que sejam iguais e assim encontrar a solução ótima. O limitante primal consiste no valor dado a uma solução válida do problema (pior valor que a solução ótima), o limitante dual é o valor otimista que a solução pode atingir. Nos problemas de minimização, que é o caso do corte bidimensional, o limitante primal é o superior e o dual é considerado inferior.

Para encontrar as soluções o algoritmo constrói uma árvore, chamada de árvore de ramificação, onde cada nó apresenta um conjunto de possíveis soluções. Ao encontrar um nó com uma solução completa válida, onde seu valor é igual ao limitante dual, então temos uma solução ótima, caso não seja possível encontrar esta solução, outras soluções serão geradas a partir das atuais, que ficarão em nós filhos (ramificação). Se um dos nós da árvore não tem condições de gerar soluções melhores, sabemos que não adianta avaliar essa parte da árvore, assim esse ramo poderá ser podado, por isso o cálculo do limitante é importante, pois um bom

limitante primal é capaz de identificar nós não promissores da árvore (ANDRADE, 2006).

Branch-and-Cut: Este algoritmo é um algoritmo branch-and-bound no qual planos de cortes são gerados ao longo da árvore de busca. A diferença está no fato de que a busca por soluções rápidas em cada nó é substituída pela procura por limites mais apertados.

Branch-and-Price: Os algoritmos de branch-and-price são muito utilizados para problemas similares ao corte bidimensional, como: empacotamento unidimensional e corte unidimensional.

2.3.1.3 Método de geração de colunas

A geração em colunas é uma técnica utilizada para encontrar variáveis com custos reduzidos, estas são capazes de melhorar a qualidade da solução encontrada. Considerando um problema de minimização, a variável gerada deverá ter um custo reduzido (negativo) e assim diminuir o valor da solução, para problemas de maximização o custo reduzido deve ser positivo. O algoritmo é iniciado com um conjunto de variáveis, que permita a viabilidade do sistema, assim pelo menos uma solução será encontrada (ANDRADE, 2006).

Zimmermann (2002) utilizou o método de geração em colunas para o problema do corte bidimensional guilhotinado. Neste foram consideradas determinadas situações, o corte de placas padrão e a demanda dos retângulos a serem cortados, objetivando a minimização do uso das placas. Abaixo a seguinte formulação do problema

Dada uma lista L de retângulos a serem cortados em placas R e seja P_1, P_2, \dots, P_m os possíveis padrões de corte de retângulos L em uma placa R . Um padrão de corte pode conter vários tipos de retângulos e ser representado como um vetor S_i . Cada elemento deste vetor indica quantos retângulos de um determinado tipo estão cortados naquele padrão P_i de forma que, como agora o objetivo é minimizar a quantidade de placas usadas, pode-se formular o seguinte problema de programação inteira... (ZIMMERMANN, 2002, p. 16).

Cria-se vetor b que representa a demanda de um retângulo. A formulação apresenta duas restrições: a enormidade de padrões possíveis e a dificuldade para

se resolver um problema de programação matemática. A partir do método de geração de colunas Zimmermann formulou os seguintes passos para a solução do problema do corte bidimensional.

- a) é escolhida a placa na qual serão cortados os retângulos;
- b) define-se uma base, podendo ser uma matriz diagonal, quadrada, com ordem igual a n (n é o número total de retângulos considerados). Esta matriz é chamada de B . Cada elemento da diagonal corresponde a quantidade máxima de cada retângulo em uma placa. A demanda de cada retângulo dividida pela sua quantidade determina de forma ineficiente, a quantidade necessária de placas para suprir a demanda de cada retângulo. Forma-se a solução inicial num vetor de n posições, chamado de x ;
- c) resolve-se o sistema de equações $yB = [1,1,\dots,1]$ (com n 1's), obtendo os valores do vetor y ;
- d) encontra-se um padrão de corte na placa. Acha-se um vetor temporário S_i com a_1 retângulos do tipo 1, a_2 retângulo do tipo 2, ..., a_n retângulos do tipo n . A somatória dos valores dos pesos dos retângulos vezes seus respectivos valores de a deve ser menor ou igual ao peso da placa. Considerando estas condições o padrão pode ser aceito, e a_1, a_2, \dots, a_n formam um vetor que será chamado de a . O vetor transposto de a será a nova coluna a ser incluída em B . Se não foi possível determinar um padrão novo então não existe solução ou a solução ótima foi encontrada;
- e) resolve-se o sistema $Bd = a$, encontrando-se d ;
- f) compara-se a divisão dos valores de x pelos valores de d . O menor valor encontrado correspondente é associado a uma variável t , e a coluna que gerou o valor de t é a coluna que será substituída pelo valores do vetor a ;
- g) a solução x é atualizada pegando-se o valor de x anterior menos os valores de t vezes d . Retorna-se ao passo c;

O método de geração de colunas é utilizado devido a quantidades de padrões de corte que possam ser formados no vetor S . São gerados os padrões e a cada iteração, é testado se um novo padrão de corte pode influenciar na otimização. Cria-se então um novo vetor, onde cada posição apresenta uma lista de padrões já gerados, na qual cada nó apresenta somente a identificação do retângulo (um

número de 0 à n) e sua quantidade. Assim têm-se n padrões de corte com o padrão i tendo x_i placas.

Observa-se que cada nova geração somente prossegue se os padrões encontrados considerem as restrições impostas no passo d do método de geração de colunas (ZIMMERMANN, 2002).

2.3.1.4 Algoritmos genéticos

Os problemas relacionados à otimização são baseados na codificação do problema, na função objetivo que se deseja maximizar ou minimizar e o espaço de soluções associados. A codificação é a descrição matemática do problema, com parâmetros e restrições, que indica o valor da função objetivo, com a definição se um conjunto de parâmetros é bom ou não para resolver o problema.

Os algoritmos genéticos foram desenvolvidos por John Holland em 1960 e são inspirados no conceito da evolução natural, buscando uma solução potencial para um problema específico em uma estrutura similar a de um cromossomo onde é aplicado operadores de seleção e recombinação (crossover). Assim como as características hereditárias são transmitidas através as gerações genes, a informação da solução também será recombinada e repassada através das interações do algoritmo genético. Os AG's mais simples geralmente trabalham com um conjunto de possíveis soluções, chamadas de cromossomos, estes possuem três tipos de representação: binária, inteira ou real, dependendo do problema a ser resolvido.

A implementação do algoritmo começa com uma população aleatória de indivíduos (cromossomos), cada um com uma aptidão, ou fitness. Estas estruturas através de operações genéticas de mutações e cruzamentos geram uma nova população de indivíduos, utilizando o princípio de reprodução e sobrevivência dos mais aptos. Cada um dos indivíduos gerados representa uma possível solução para o problema. Assim o AG procura a melhor solução, buscando a otimização da função objeto.

A descrição básica de um algoritmo genético pode ser observada pela Figura 2.

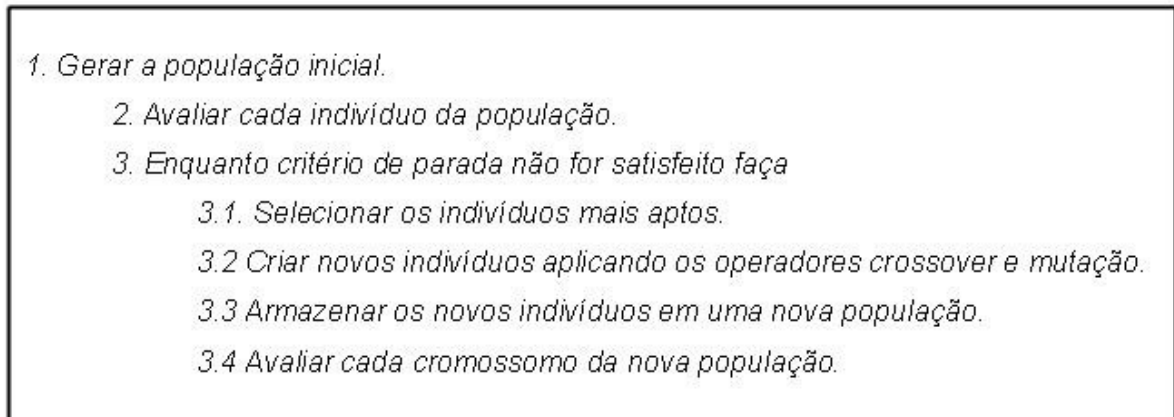


Figura 2: Descrição básica do AG

O algoritmo genético possui a seguinte estrutura de passos para execução:

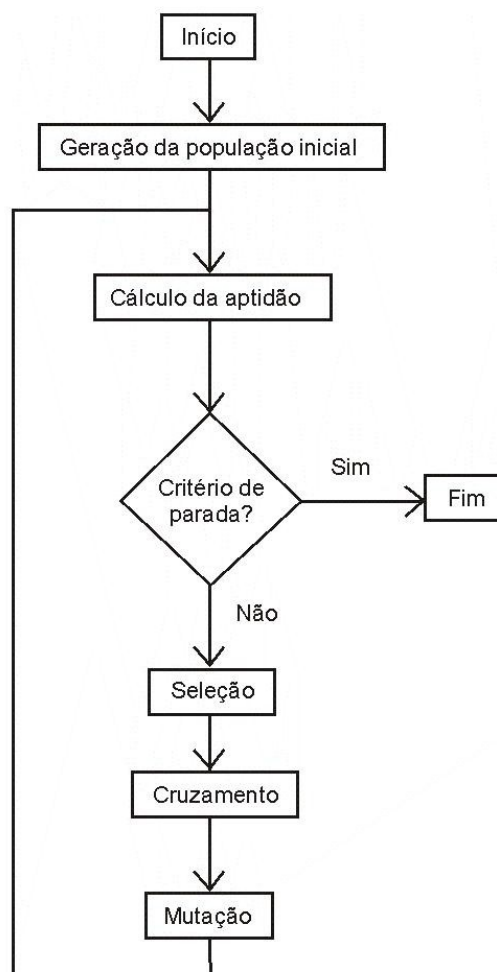


Figura 3: Estrutura genérica de um AG

Franceschette (2008) aplicou algoritmo genético para o problema de corte bidimensional. No seu estudo propôs um algoritmo genético para resolver o problema de uma empresa, que tinha a necessidade de cortar chapas de aço para gerar novas peças a serem utilizadas na manufatura de seus produtos. Com a seguinte definição do problema:

Dado um número finito m de itens retangulares (denominados peças) de largura l_i , comprimento c_i e demanda d_i a serem obtidos de uma chapa de dimensões $L \times C$, encontrar padrões de corte da chapa que atendam a demanda de itens solicitados com menor desperdício possível (FRANCESCHETTE, 2008, p. 44).

Para demonstrar a utilização do AG, foi utilizada construção de um padrão de corte para duas peças e a partir deste conjunto de dados, desenvolve-se um padrão de corte em faixas. A população inicial de cromossomos forma-se pelo conjunto das faixas combinadas. As construções destas faixas formaram os genes dos cromossomos, feita por um algoritmo de encaixe.

As duas saídas foram usadas no cálculo do fitness do cromossomo, onde o mesmo é avaliado se é bom ou não. Quando gerados os cromossomos da população inicial, ou gerado um novo cromossomo, através de uma operação genética o fitness deve ser calculado para cada um deles, considerando que cada faixa é um cromossomo este deve ter o menor desperdício possível. O valor do sub-fitness é definido pelo desperdício de material de cada gene, no fitness considera-se o percentual da demanda atendida e o desperdício da chapa. Com a construção da população inicial são aplicados os operadores genéticos.

Crossover, aplicado o operador é realizado o sorteio, para a escolha dos pais que participarão do cruzamento, os cromossomos são agrupados em duplas, determinado por sorteio um ponto de cruzamento entre os pais, que realizarão a troca de informação genética. Do cruzamento de dois pais são gerados dois filhos com novos fitness, caso um deles seja considerado ruim, o filho é descartado. A população gerada de filhos possui o mesmo número de pais, inicialmente utilizados, caso alguns dos filhos sejam descartados outros serão gerados.

Mutação aplica-se o operador de mutação na população de filhos gerada, essa operação é importante, pois realiza a troca de material genético entre os filhos. Ela ocorre dentro de um cromossomo em um de seus genes, assim o gene

escolhido é trocado por algum aleatoriamente, escolhido dentro da população. Novamente caso o gene escolhido não for bom escolhe-se outro gene para a mutação. O fitness sempre é recalculado após o processo.

Nova população, após a aplicação dos operadores a nova população toma o lugar da anterior. São listados os pais e filhos na ordem dos seus fitness, no estudo realizado, quanto menor o fitness melhor é seu padrão de corte. Através desta sequência de algoritmos se espera chegar perto da solução do problema, pois a cada nova geração são calculados valores de fitness melhores (FRANCESCHETTE, 2008).

2.3.1.5 Método GRASP

O método GRASP, surgiu na década de 80, originalmente desenvolvido, por Feo e Resende. Uma combinação entre a heurística construtiva e o método de busca local.

É um algoritmo iterativo onde cada iteração é composta por duas fases: uma fase de construção onde uma solução viável para o problema é criada e uma fase de busca local, posterior à construção, que tem como objetivo tentar melhorar a solução inicial obtida na fase de construção. As iterações GRASP são independentes, ou seja, na iteração atual não se leva em conta nenhuma informação das iterações anteriores. O critério de parada normalmente usado é um número máximo de iterações. A melhor solução obtida ao final da execução do GRASP é a solução final, na Figura 4 temos a instrução genérica da heurística GRASP (TRINDADE, 2005).

```

Procedimento GRASP ( )
1. Entrada de Dados ( );
2. Para (critério de parada GRASP não satisfeito) faça
3. Construa Solução Gulosa Aleatória (solução);
4. Busca Local (solução, Viz (solução));
5. Atualiza Solução (solução, melhor solução encontrada);
6. Fim-para;
7. Retorna (melhor solução encontrada);
Fim GRASP

```

Figura 4: Algoritmo GRASP

Fonte: TRINDADE (2005, p. 17)

2.3.2 Principais técnicas de encaixe para o problema do corte

Existem várias formas eficientes para realizar o encaixe das peças com duas dimensões. Dentre os algoritmos desenvolvidos para tratar essa situação os mais utilizados são: Next-Fit, First-Fit, Best-Fit e Bottom-Left. Os chamados algoritmos de níveis encaixam as peças em faixas, da esquerda para a direita, sendo que o início de um novo nível coincide com o topo da peça mais alta do nível anterior. São métodos que possuem como principal vantagem a rapidez.

O algoritmo next-fit possui uma técnica de encaixe mais simples. Nele, a primeira peça é inserida no canto inferior esquerdo da placa. As próximas peças são inseridas através de um processo iterativo, ao lado da peça anteriormente inserida, até que a largura restante se torne insuficiente para alocar mais uma peça. Quando isso ocorre, forma-se uma faixa, com altura correspondente à altura da peça com a maior dimensão neste sentido. Após a formação da primeira faixa, a próxima peça será inserida no canto esquerdo, acima da primeira peça da faixa anterior. Este processo se repete, formando-se outras faixas, até que todas as peças da solução sejam inseridas (TEMPONI, 2007). Segue na Figura 5 a ilustração de um exemplo do procedimento.


```

procedimento Next-fit()
1  W <- largura do objeto;
2  Hi <- altura do item i, i=1, ..., n;
3  Wi <- largura do item i, i=1, ..., n;
4  WDFj <- W {largura disponível da faixa j, j=1, ..., n};
5  j <- 1;
6  para i=1 até i=n faça
7    se (Wi <= WDFj) então
8      WDFj <- WDFj - Wi {inserir o item i na faixa j}
9    senão
10     j <- j + 1
11    WDFj <- WDFj - Wi {inserir o item i na faixa j}
12  fim-se;
13 fim-para;
fim Next-fit();

```

Figura 5: Algoritmo Next-fit

Fonte: TEMPONI (2007, p. 32)

O procedimento first-fit insere a peça na primeira faixa, se ela não couber, então insere na próxima faixa, assim sucessivamente, até que o método encontre alguma faixa em que exista espaço suficiente para alocá-la. Caso a peça não se encaixe em nenhuma faixa existente, então se cria uma nova faixa (TEMPONI, 2007), veja Figura 6.

```

procedimento First-fit()
1  W <- largura do objeto;
2  Hi <- altura do item i, i=1, ..., n;
3  Wi <- largura do item i, i=1, ..., n;
4  WDFj <- W {largura disponível da faixa j, j=1, ..., n};
5  j <- 1;
6  para i=1 até i=n faça
7    enquanto (Wi <= WDFj) faça
8      j <- j + 1
9    fim-enquanto
10   WDFj <- WDFj - Wi {inserir o item i na faixa j}
11  fim-para;
fim First-fit();

```

Figura 6: Algoritmo First-fit

Fonte: TEMPONI (2007, p. 33)

O algoritmo best-fit, insere a peça na faixa em que a área do item (A_i) melhor se ajusta à área disponível da faixa (A_{dfj}). Para isso o método calcula a relação entre essas duas áreas (Rel), para todas as faixas. Em seguida, a peça é inserida na faixa que apresenta menor relação. Caso a peça não caiba em nenhuma faixa existente, então se cria uma nova faixa (TEMPONI, 2007), conforme Figura 7.

```

procedimento Best-fit()
1  W <- largura do objeto;
2  Hi <- altura do item i, i=1, ..., n;
3  Wi <- largura do item i, i=1, ..., n;
4  HFi <- 0 {altura da faixa j, j=1, ..., n};
5  WDFj <- W {largura disponível da faixa j, j=1, ..., n};
6  j <- 1;
7  para i=1 até i=n faça
8    MELHOR_rel <- alfa
9    Ai <- Hi * Wi {área do item i}
10   para j=1 até j=n faça
11     ADFj <- HFi * WDFj; {área disponível na faixa j}
12     REL <- ADFj / Ai; {relação das áreas}
13     se (Wi <= WDFj) e (Hi <= HFi) e (REL < MELHOR_rel) então
14       MELHOR_rel <- REL
15       k <- j {armazena a faixa que contém a melhor relação}
16     fim-se
17   fim-para
18   WDFk <- WDF - Wi {insere o item i na faixa k}
19 fim-para
fim Best-fit();

```

Figura 7: Algoritmo Best-fit

Fonte: TEMPONI (2007, p. 34)

O bottom-left funciona da seguinte maneira: as peças são, inicialmente, posicionadas no canto superior direito da placa, sendo inseridas na ordem em que aparecem na lista de peças de cada placa. Em seguida, cada peça é deslocada verticalmente para baixo até encostar-se em outra peça ou atingir o limite da placa. Depois disso a peça é deslocada horizontalmente para esquerda até que atinja outra peça ou o limite da placa. Esse movimento para baixo e para a esquerda se repete até que não seja mais possível movimentar a peça (TEMPONI, 2007).

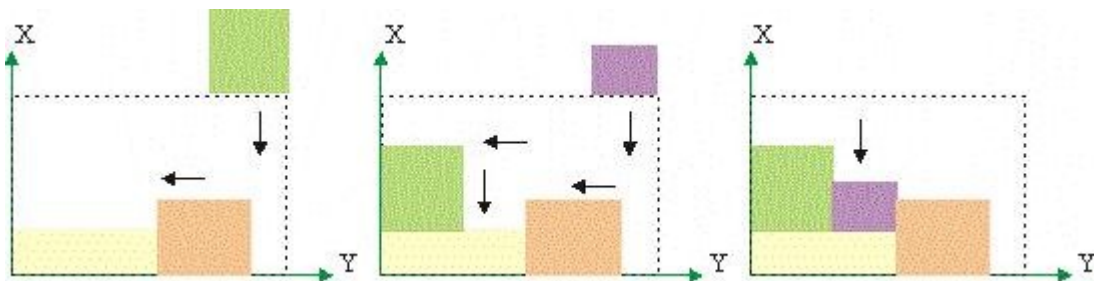


Figura 8: Ilustração do Bottom-Left

Os algoritmos pesquisados possuem métodos distintos para a solução do mesmo problema. Quando mencionada a comparação de algoritmos é necessário ter conhecimento de alguns conceitos de análise de algoritmos, estes são descritos

na próxima seção.

2.4 Análise de algoritmos

Um algoritmo é uma sequência de operações definidas e muito eficazes, que quando executadas terminam em um determinado tempo, retornando uma solução ou informando que a solução não pode ser encontrada. Quando se fala em análise de algoritmos alguns critérios devem ser considerados como:

- a) o tempo de processamento em relação ao volume de dados;
- b) espaço de memória utilizado para os dados;
- c) extensão do código fonte;
- d) resultados obtidos corretamente;
- e) prever entradas inválidas;

Um dos itens importante para a obtenção de resultados é o desempenho, existe a necessidade de escolher entre os vários algoritmos o mais eficiente. Pode-se ainda melhorar os algoritmos conhecidos ou desenvolver novos, para problemas que já possuem solução (MARTINS, 2009).

Uma das formas de avaliação dos algoritmos é aplicá-los em um dado computador e simular as diversas entradas e saídas que o mesmo poderá retornar, esse processo é denominado avaliação empírica. Segundo Martins (2009) o melhor é avaliar o pior caso (o maior tempo gasto), o que já pode inviabilizar o algoritmo e o caso médio, que geralmente é como o programa vai se comportar na maior parte dos casos. A avaliação teórica consiste em encontrar uma fórmula matemática que expresse o processo de execução do algoritmo (MARTINS, 2009).

A análise da complexidade de um algoritmo é realizada de uma forma muito particular, pois cada algoritmo possui uma medida de complexidade, com parâmetros particulares. O esforço computacional requerido por um algoritmo pode ser medido pelo seu tempo e quantidade de memória utilizada. O seu desempenho pode ser medido pelo custo de suas sequências de execução (TOSCANI; VELOSO, 2008).

Quando falamos em encontrar o melhor algoritmo, para dado problema, isso envolve a comparação de sua complexidade. A complexidade de um algoritmo é

medida segundo um modelo matemático que supõe que este vai trabalhar sobre uma entrada de dados com tamanho N . O processo de execução de um algoritmo pode ser dividido em etapas denominadas passos (número fixo de operações básicas, tempo constantes, operação de maior frequência chamada dominante). O número de passos de um algoritmo é considerado como o número de execuções da operação dominante em função das entradas, desprezando-se constantes aditivas ou multiplicativas. Definimos a expressão matemática de avaliação do tempo de execução de um algoritmo como sendo uma função que fornece o número de passos efetuados pelo algoritmo a partir de certa entrada (TOSCANI; VELOSO, 2008).

É importante observar que o tempo de execução vai depender de vários fatores, tais como a linguagem utilizada, a máquina e em casos particulares de uma determinada entrada. A seguir são apresentadas algumas linguagens de programação e definida a escolha.

2.5 Tecnologias utilizadas para o desenvolvimento

Quando é realizada a escolha da linguagem para o desenvolvimento de uma aplicação, são feitas avaliações de diversas tecnologias disponíveis, buscando utilizar a que vai retornar maiores benefícios ao projeto. Nesta fase devem-se observar quais os itens serão considerados, para a escolha da linguagem.

Será avaliada a escolha através os seguintes itens: escalabilidade, velocidade de desenvolvimento, manutenibilidade, facilidade de aprendizado e custo de propriedade.

2.5.1 Linguagens de programação

Um dos fatores relevantes para o desempenho do sistema é a linguagem a ser utilizada. A seguir serão apresentadas as características de algumas linguagens já utilizadas no desenvolvimento de sistemas (TOSCANI; VELOSO, 2008).

A linguagem C++ foi criada para suportar programas orientados a objetos, desenvolvida para ser tão eficiente quanto ao C, suporta múltiplos paradigmas de programação como programação estruturada e orientada a objetos.

Dentre as vantagens desta linguagem podemos citar a possibilidade de programação de alto e baixo nível, flexibilidade, portabilidade e consistência, adequa-se a projetos grandes vastas disponibilidade de suporte, não está sob o domínio de uma empresa, assim como o java e é padronizado pela ISO. Como desvantagens temos, pela compatibilidade com o C acabou herdando os problemas de sintaxe, os compiladores atuais nem sempre produzem código mais otimizado (velocidade e tamanho), exige um grande período para aprendizagem e a biblioteca padrão não cobre áreas como threads, conexões TCP/IP e manipulação de sistemas de arquivo (SCHILDT, 1990).

Java é uma linguagem de programação orientada a objetos (OO), desenvolvida pela Sun Microsystems muito semelhante a sintaxe da linguagem C++. É uma linguagem simples, não possui sobrecarga de operadores e a memória alocada dinamicamente é gerada pela própria linguagem. O java possui uma máquina virtual que realiza uma verificação em tempo de execução quanto aos acessos de memória, evitando travas nos programas, como aberturas de arquivos e outros eventos que podem gerar esta situação.

Os programas feitos em java são divididos em duas partes, as classes e os métodos, que realizam tarefas e retornam informações. Essa linguagem possui algumas vantagens, possui um vasto conjunto de bibliotecas, facilidade na criação de programas multitarefa e distribuídos, suporta caracteres Unicode, possui recursos de rede, utilização de protocolos TCP/IP (HTTP e FTP), segurança e desempenho.

O Delphi possui um ambiente de desenvolvimento fácil de usar e conta com uma grande biblioteca de componentes visuais (VCL, Visual component library), utiliza o Object Pascal como base, atingindo uma maior capacidade e velocidade de produção. O Delphi é uma ferramenta para desenvolvimento rápido de aplicações (RAD).

O Object Pascal é uma extensão da linguagem pascal e contém uma série de conceitos da programação orientada a objetos, assim é possível a programação em Delphi utilizando esses conceitos, que permitem uma melhor abstração da realidade e o reaproveitamento de código fonte, uma das características das ferramentas RAD (CANTÙ, 2003).

2.5.2 Escolha da linguagem

Considerando os itens mencionados no início desta seção a linguagem escolhida foi o Delphi, pelos motivos expostos a seguir:

- a) essa linguagem foi escolhida por facilitar a programação de componentes visuais e pelo reaproveitamento de código, que será útil na implementação dos algoritmos estudados, por se tratar de uma ferramenta RAD o tempo gasto no desenvolvimento será menor;
- b) possui um tempo de aprendizado menor em relação às demais apresentadas (CANTÙ, 2003);
- c) a empresa estudada para possível implantação, já utiliza sistemas nesta linguagem, facilitando futuras integrações que possam ocorrer e sem custo de licença;

3 METODOLOGIA

O objetivo de cada algoritmo aplicado é encontrar o melhor padrão de corte, que contém o maior número de peças, reduzindo o número de folhas que precisaram ser cortadas, para suprir a demanda.

Foi realizada a escolha de três algoritmos com metodologias diferentes, adaptados ao ambiente proposto e realizados os testes necessários para a eleição da melhor performance. Esse desempenho será medido através dos critérios levantados com a empresa descritos na seção 3.1 e a escolha dos algoritmos teve como base as definições que serão apresentadas na seção 3.2.

No processo de levantamento dos dados, foram definidos alguns retornos de informações que serão necessárias.

- a) quantidade de peças que contém o padrão;
- b) quantidade de folhas originais para suprir a demanda;
- c) percentual de sobra;
- d) tempo de processamento;

A demanda é a quantidade de folhas no formato final que será necessária, para produzir o pedido. Esse processo também poderá ser utilizado para cálculo do *original*⁴, não restrito somente a parte do corte. A demanda será calculada somente após o algoritmo ter encontrado a quantidade de peças que serão dispostas no formato original, obtidos pelas entradas A e B, correspondente ao tamanho original da folha e as entradas X e Y, correspondente ao tamanho que deverá ser cortado.

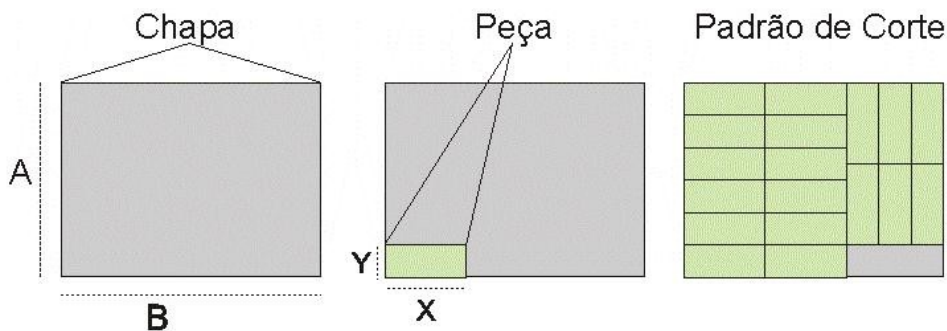


Figura 9: Ilustração chapa e peça

⁴ Original = Uma representação bidimensional, composta de imagens e/ou textos, reproduzidas em múltiplas cópias por meio de um processo de pré-impressão e impressão.

Como o processo de corte é realizado através de guilhotina, que percorre a chapa de um lado ao outro, podemos caracterizar esse tipo de corte, como corte guilhotinado, com peças e chapas do tipo retangular definido como ortogonal, o número de cortes não será levado em consideração neste trabalho, na Figura 10 temos representados dois padrões de corte, com a mesma quantidade de peças, porém com disposições diferentes, o pontilhado demonstra o número de cortes necessários para produzir as peças. No caso das aplicações propostas os dois tipos de padrões podem ser considerados como satisfatórios, pois ambos possuem o mesmo número de peças.

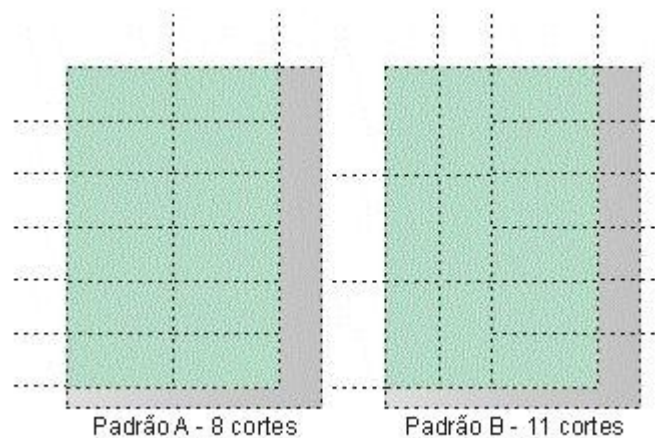


Figura 10: Números de cortes e possibilidades de padrões

Definiu-se que todas as chapas e peças deverão ter suas dimensões expressas em milímetros e que serão representadas por números inteiros. Somente a peça pode girar em um ângulo de 90° , as peças não podem se sobrepor e o padrão de corte encontrado não pode ser maior que a chapa disponível.

3.1 Métricas adotadas para decisão do melhor algoritmo.

Considerando o que foi apresentado na seção 2.5, podemos identificar o algoritmo que terá ou não um bom desempenho, já pela sua estrutura. Porém neste trabalho os algoritmos serão avaliados empiricamente.

Através de levantamentos foram definidos os pontos que terão maior relevância na escolha. Considerando as necessidades do cenário apresentado,

temos as seguintes métricas descritas abaixo:

- a) tempo de processamento: é o tempo que o sistema levará para encontrar a solução aproximada;
- b) percentual de sobra: é o percentual de material que não poderá ser aproveitado (desperdício). Ilustrado na figura 11;

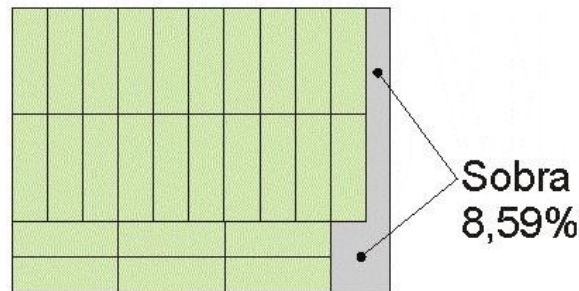


Figura 11: Material não aproveitado

O peso de cada uma das variáveis relevantes para obtenção do melhor algoritmo é dado pela tabela abaixo:

Tabela 1: Relação dos percentuais adotados para escolha

Métrica	Percentual
Tempo de processamento	50%
Percentual de sobra	50%
	100%

3.2 Escolha dos algoritmos a serem utilizados

Para realizar a escolha dos algoritmos que serão desenvolvidos neste trabalho foram utilizados os seguintes critérios:

- a) desempenhos satisfatórios já apresentados em outros trabalhos;
- b) estudos relacionados aos problemas do corte bidimensional;
- c) facilidade de adaptação, para a linguagem escolhida;
- d) diversificação de metodologias entre os algoritmos;

Os algoritmos escolhidos para aplicação segundo os critérios foram: método GRASP, algoritmos genéticos e método de geração em colunas.

De todos os trabalhos relacionados ao problema do corte bidimensional pesquisados, os algoritmos mais aplicados foram os mencionados, todos possuem uma metodologia diferente e com um bom grau de desempenho e precisão.

4 DESENVOLVIMENTO

Todos os algoritmos aplicados foram descritos e mencionados na seção 2. Utilizando os conceitos de PL e PI foi possível a criação da função objetivo, que será utilizada para todos os métodos aplicados.

Definição da Função Objetivo:

$\{\text{Min}\}Z = X_1 + X_2 + X_3 + \dots + X_n$	(1)
Sujeito a: $P_1X_1 + P_2X_2 + \dots + P_nX_n \leq A$	(2)
$X_j \geq 0, j = 1 \dots n$	(3)

Figura 12: Função objetivo

- (1) Minimizar o desperdício de material.
- (2) Total de itens dispostos não pode ultrapassar a área disponível.
- (3) Número de folhas cortadas do item j , para o padrão encontrado.

Nesta seção é introduzido o conceito de faixa, que corresponde a um conjunto de peças ou itens, que irão completar o padrão de corte, veja Figura 13.

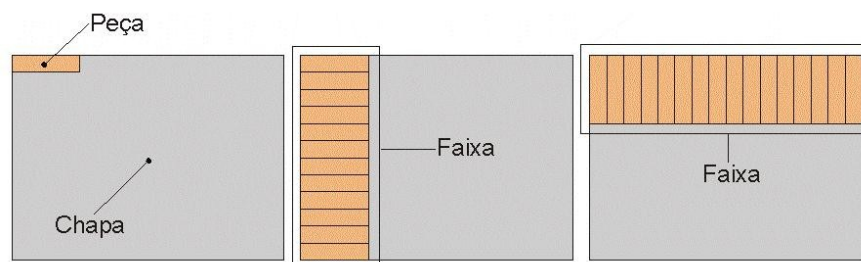


Figura 13: Definição de faixas

4.1 Metodologias de desenvolvimento

Mesmo não sendo o foco deste trabalho, visando-se a boa construção do *software* para experimentos, aplicou-se a metodologia para desenvolvimento incremental, onde incrementos iniciais atuam como um protótipo para ajudar a descobrir requisitos para os incrementos posteriores, utilizando conceitos de

modelagem UML apresentada a seguir através de artefatos.

A linguagem escolhida possibilita a utilização de orientação a objetos, esse método é adotado comercialmente e diversas plataformas de desenvolvimento disponibilizam suporte através de ferramentas CASE, que aceleram o desenvolvimento. Dentre os diversos recursos desta metodologia pode-se citar o reaproveitamento de código, que será utilizado neste desenvolvimento.

Para o desenvolvimento dos diagramas foi utilizada a ferramenta Jude.

Nas Figuras 14 e 15 são apresentados os diagramas de casos de uso e de classes.

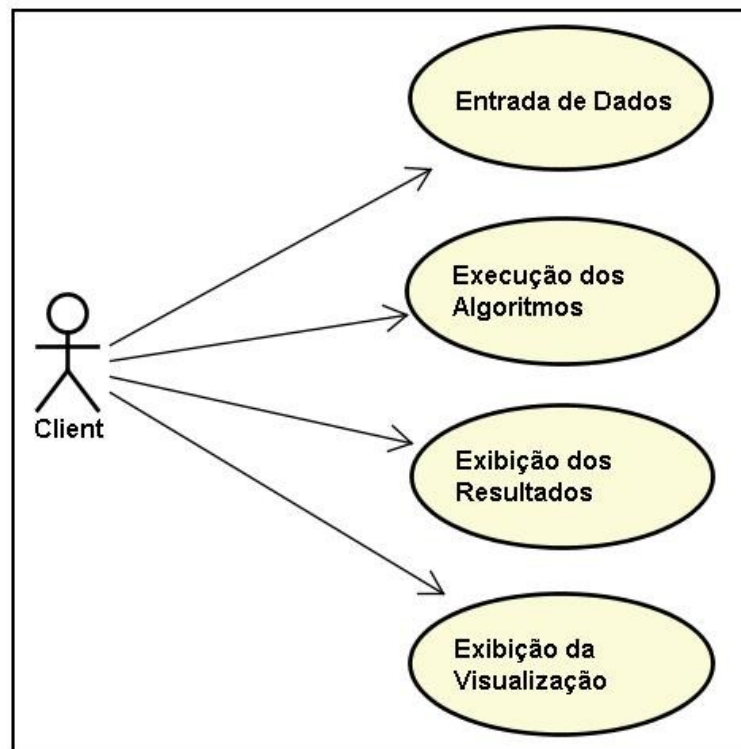


Figura 14: Diagrama de casos de uso

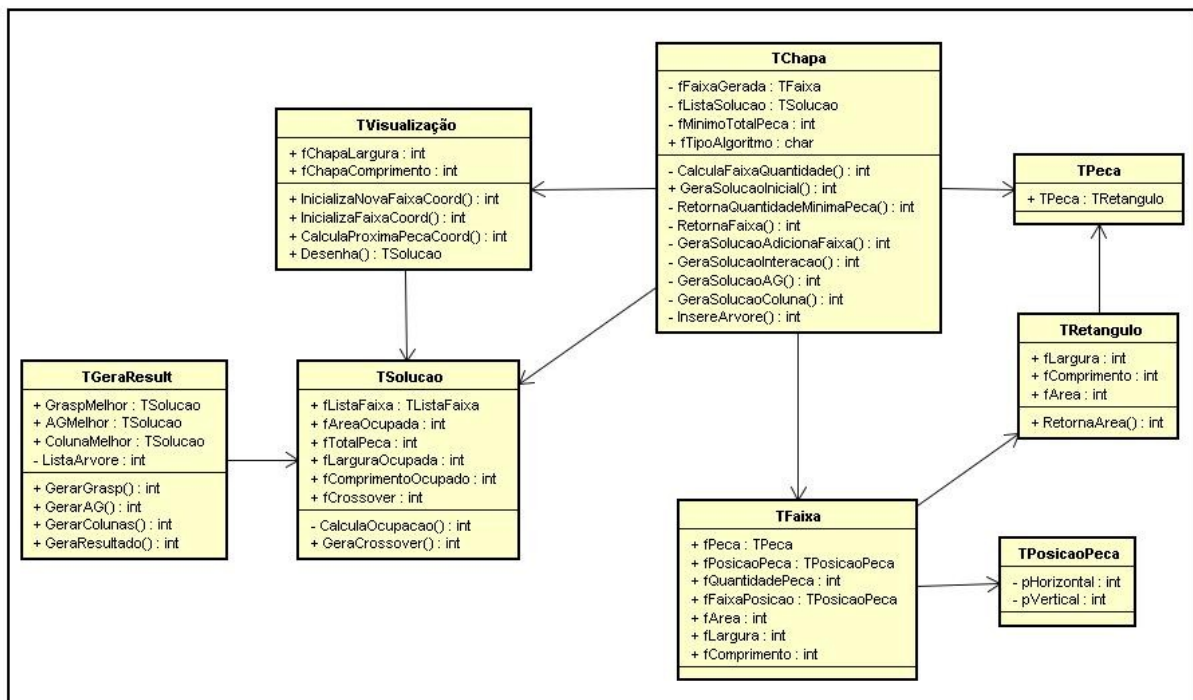


Figura 15: Diagrama de classes

4.2 Desenvolvimento do método GRASP

Foi aplicada a metodologia GRASP, para o problema do corte bidimensional, com peças retangulares, objetivando encontrar o padrão de corte ideal para melhor aproveitamento da chapa. Utilizou-se como base os trabalhos de Trindade (2005) e Velasco (2010).

Conforme sugere o algoritmo, para o contexto deste trabalho temos a seguinte sequência de passos:

- a) seleciona-se a peça P e através do método de construção, realiza a geração das faixas, tanto na vertical quanto na horizontal, criando uma lista de faixas geradas, onde cada faixa tem um parâmetro que determina a quantidade de itens que contém cada faixa criada, a área desta faixa, a posição da peça e da faixa (vertical ou horizontal);
- b) através de um método construtor cria-se a chapa, com informações de altura, comprimento e área. Pela lista de faixas geradas são encontradas as primeiras soluções viáveis do problema, geradas através de um procedimento. Cria-se assim um vetor com as soluções iniciais, contendo as

faixas geradas que foram possíveis de serem encaixadas no padrão. Por um outro procedimento, são verificadas as sobras, na tentativa de realizar mais um encaixe de alguma outra faixa diferente das faixas geradas nos padrões iniciais, caso seja possível essa solução é adicionada a lista de soluções;

c) a segunda fase do algoritmo é a melhoria das soluções encontradas na fase de construção, onde o procedimento busca melhorar a solução realizando novas interações de faixas, através das soluções iniciais encontradas;

d) quanto maior for a quantidade de peças encaixadas no padrão encontrado, mais próximo estará da solução ótima;

Na Figura 16 observa-se um exemplo desta interação.

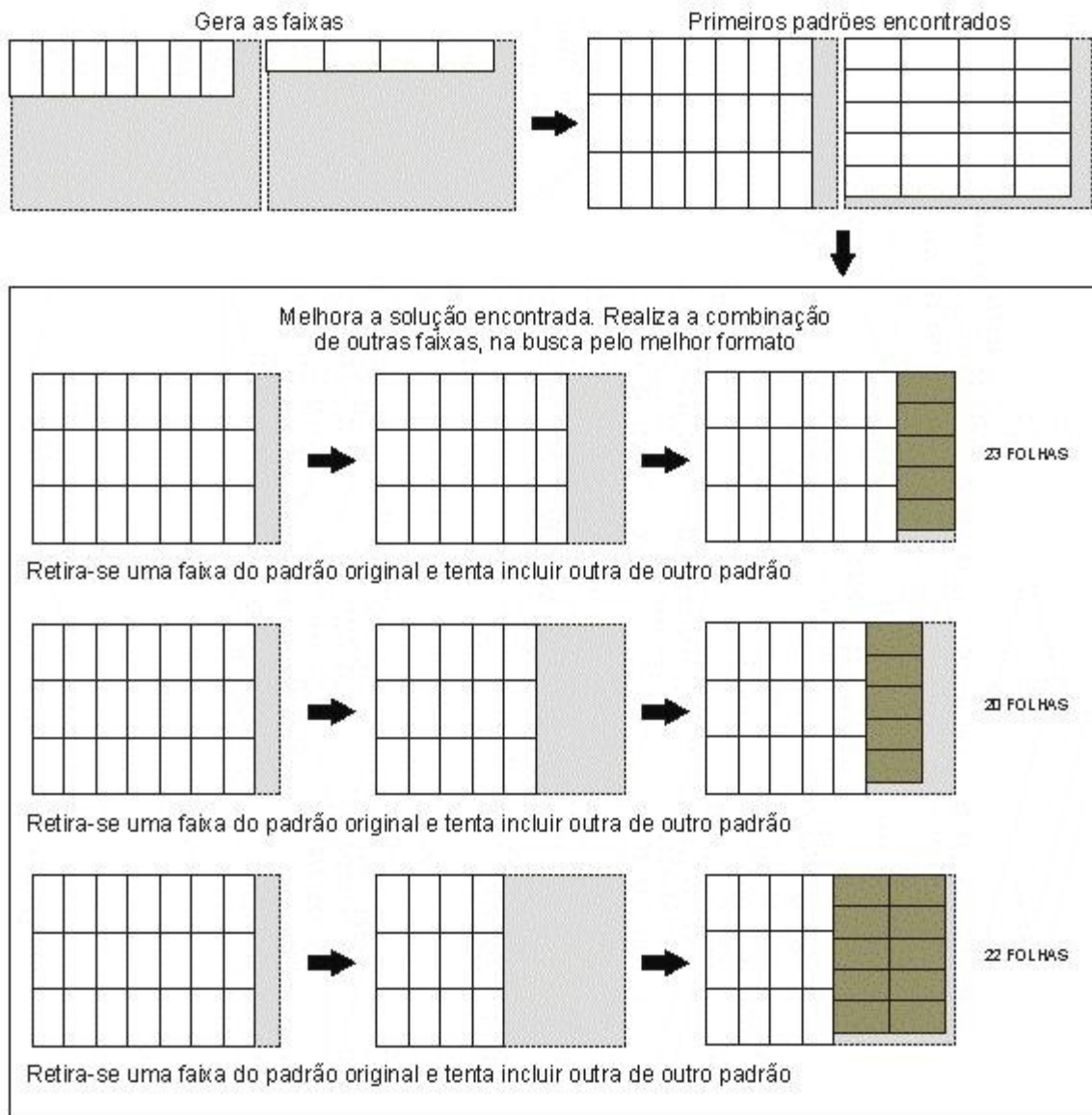


Figura 16: Interação do método GRASP

Foram utilizados dois padrões, que através da interação entre eles gerou um novo, com um número maior de folhas e um percentual de sobra menor. No exemplo ilustrado (Figura 16), esse resultado foi possível através da exclusão da última faixa, para a inclusão de outra, retornando o padrão com um número maior de peças dispostas na chapa. Na Figura 17 é apresentado o fluxograma do método GRASP.

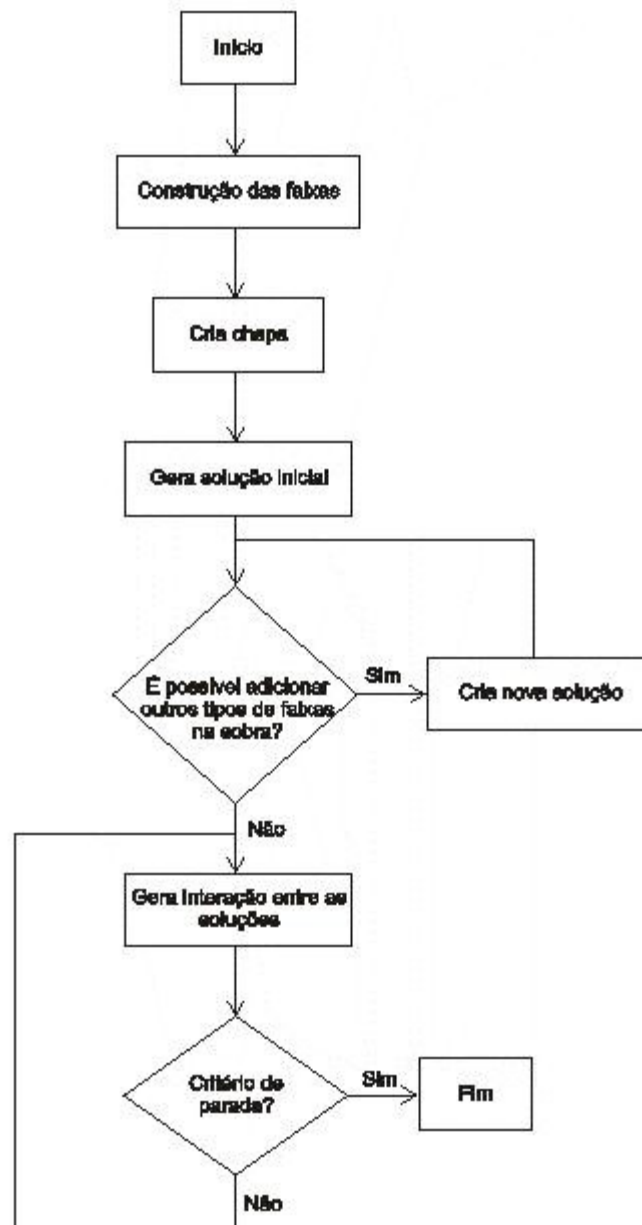


Figura 17: Estrutura do método GRASP

4.3 Desenvolvimento do algoritmo genético

Na utilização de algoritmos genéticos as peças, faixas e o padrão de corte são interpretados como genes, cromossomos e indivíduos, pois o algoritmo tem como característica o conceito da evolução natural. Utilizou-se como base para o desenvolvimento o trabalho de Franceschette (2002).

Seguindo essas características foram definidas as variáveis do problema,

onde as faixas geradas passam a ser denominadas como os genes, os cromossomos são um conjunto de soluções possíveis, compostas por estes genes.

Inicialmente são gerados os genes, que passam a ter, além das informações de posições da faixa e da peça o percentual de sobra, que será necessário para o cálculo da função fitness. Com a informação dos genes é possível gerar uma população inicial de indivíduos, cada um representando uma possível solução.

O cálculo da aptidão é realizado toda a vez que uma nova solução é gerada, após passarem pelos operadores de crossover e mutação. Cada cromossomo vai corresponder a um padrão de corte composto pelas faixas geradas, esse padrão deve possuir o menor desperdício. O fitness passa a ser o percentual de sobra, calculado em milímetros quadrados, quanto menor o fitness do indivíduo gerado, melhor ele será. Segue abaixo exemplos de entradas e seus fitness.

Tabela 2: Exemplos de fitness

Chapa		Peça		Área Total	Área ocupada	Área Sobra	Fitness
240	320	25	41	76800	75850	950	1,23%
240	320	30	39	76800	74880	1920	2,50%
960	660	320	330	633600	633600	0	0
960	660	123	220	633600	622380	11220	1,77%
3540	7890	1234	150	27930600	27765000	165600	0,59%
3540	7890	200	300	27930600	27840000	90600	0,32%
15000	2800	850	1400	42000000	40460000	1540000	3,66%
15000	2800	1540	792	42000000	41469120	530880	1,26%
54200	8880	950	2540	481296000	477774000	3522000	0,73%
54200	8880	2450	3200	481296000	462560000	18736000	3,89%

São escolhidos dois pais, que geram dois novos filhos. O cruzamento (Crossover) entre os pais consiste na troca de informação genética, os cromossomos de cada par de indivíduos são particionados, através do chamado ponto de corte. Para esta situação foi implementada uma função randômica, que escolhe de maneira aleatória o ponto de corte a partir do número de faixas do pai. O número gerado é a quantidade de faixas retiradas, para inclusão da informação genética (faixas) do outro pai. Após o cruzamento é realizado o cálculo do fitness de cada filho gerado, caso o filho gerado não tenha um fitness melhor ou igual a de seus pais ele é descartado e um novo filho é gerado. Se essa condição for satisfeita temos uma nova população, pode ocorrer de um dos pais gerados já ter 100% de aproveitamento, geralmente ocorre para peças quadradas, neste caso não são gerados filhos, pois a solução encontrada já possui um percentual de 0% de sobra.

O operador de mutação se aplica a cada filho de forma individual, neste caso tenta-se incluir um novo gene de maneira aleatória, normalmente a probabilidade de ocorrer é pequena, mas garante que nenhum espaço de busca seja ignorado.

Foi criado um procedimento principal que realiza os cruzamentos, mutações e calcula o fitness de cada indivíduo gerado, ao final os novos filhos que compõem a nova população são armazenados em um vetor onde o melhor dos filhos será visualizado. É apresentado na Figura 18 o fluxograma do algoritmo genético aplicado.



Figura 18: Estrutura do algoritmo genético aplicado ao cenário proposto

4.4 Desenvolvimento do método de geração de colunas

No desenvolvimento do método de geração de colunas foram utilizadas as

referências do trabalho de Zimmermann (2002) e Andrade (2006).

Conforme já exposto na seção 2 o algoritmo de geração de colunas é aplicado juntamente com o algoritmo branch-and-bound e juntos correspondem ao algoritmo branch-and-price.

Para o desenvolvimento deste método foi necessária a utilização de uma estrutura de árvore, com utilização de ponteiros. Segue a sequência de passos do desenvolvimento do algoritmo de geração de colunas.

- a) primeiramente são geradas as soluções iniciais do problema, que irão compor a raiz da árvore;
- b) a partir destas soluções calculam-se os limitantes primal e dual, que correspondem aos valores superiores e inferiores que a solução pode atingir. Esses valores são importantes, pois irão definir se a solução é viável ou não;
- c) a árvore é criada e cada nó representa uma parte de um conjunto de soluções. As soluções criadas nas folhas são geradas através da combinação das faixas. Como o algoritmo possui o conceito de dividir para conquistar, a árvore é gerada dividindo o problema em dois, conforme ilustrado na Figura 19, são geradas duas sub-árvores;
- d) a poda do nó ocorre quando essa solução possui um valor menor que o limitante primal calculado. O critério para que a solução seja considerada são os valores primal e dual, onde estes devem ser maior ou igual aos limitantes calculados no passo b;
- e) realiza a busca na árvore por uma solução maior ou igual ao limitante dual;



Figura 19: Início da estrutura da árvore

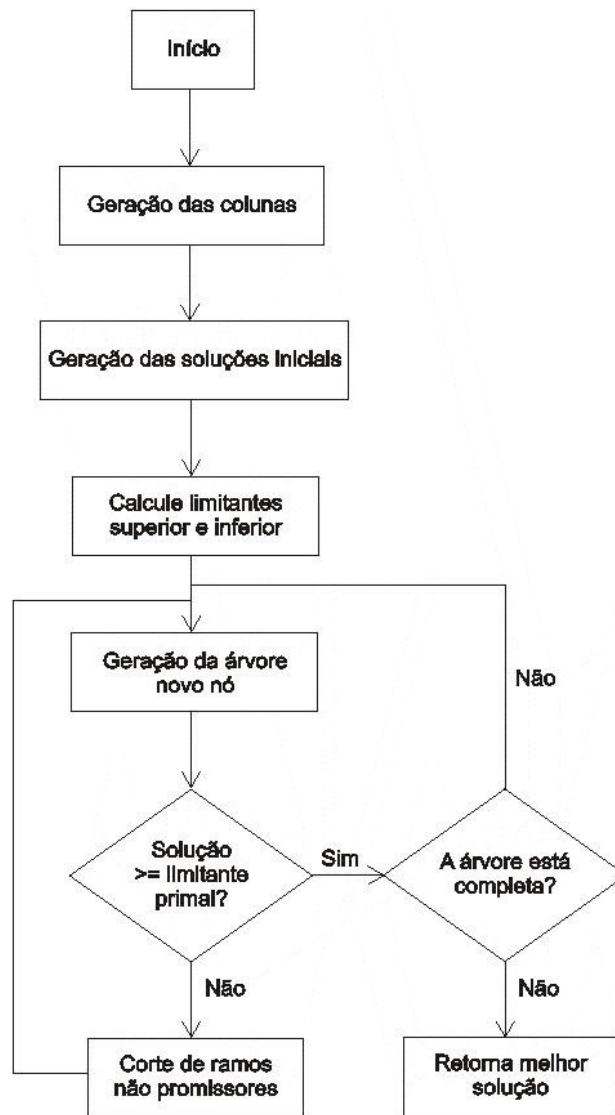


Figura 20: Estrutura do método geração de colunas

4.5 Funcionamento, telas e visualização dos padrões

Para realizar os testes com os algoritmos foi desenvolvida uma tela principal, onde são informadas as entradas de dados, para a chapa, peça e demanda. Depois de informadas, seleciona-se os algoritmos que serão testados, a chamada destes algoritmos retornará o melhor resultado encontrado, como mostra a Figura 21.

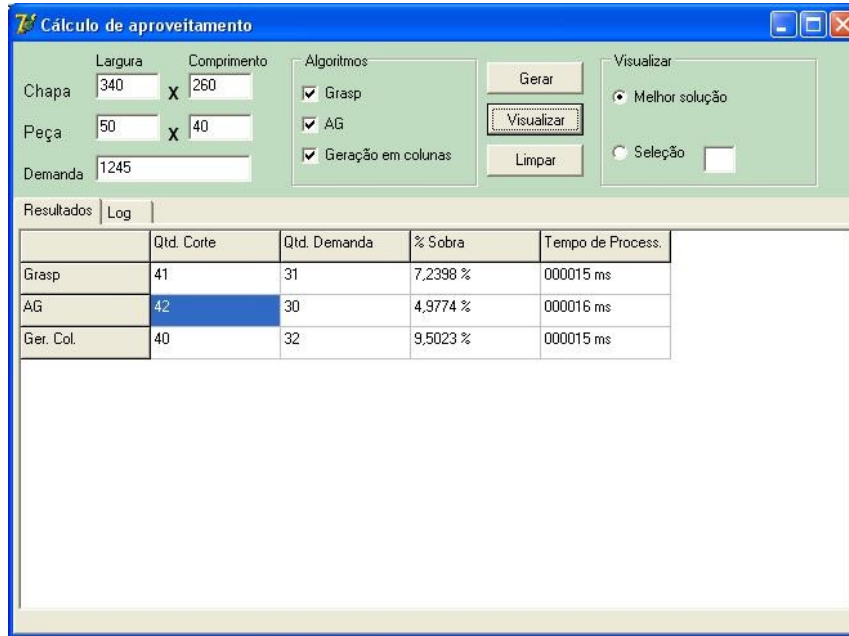


Figura 21: Layout da tela

Com o objetivo de realizar uma melhor conferência dos padrões gerados, foi incluída uma aba de “Log”, onde podem ser visualizados em modo texto os diferentes padrões encontrados. A Figura 22 demonstra a forma de representação dos dados.

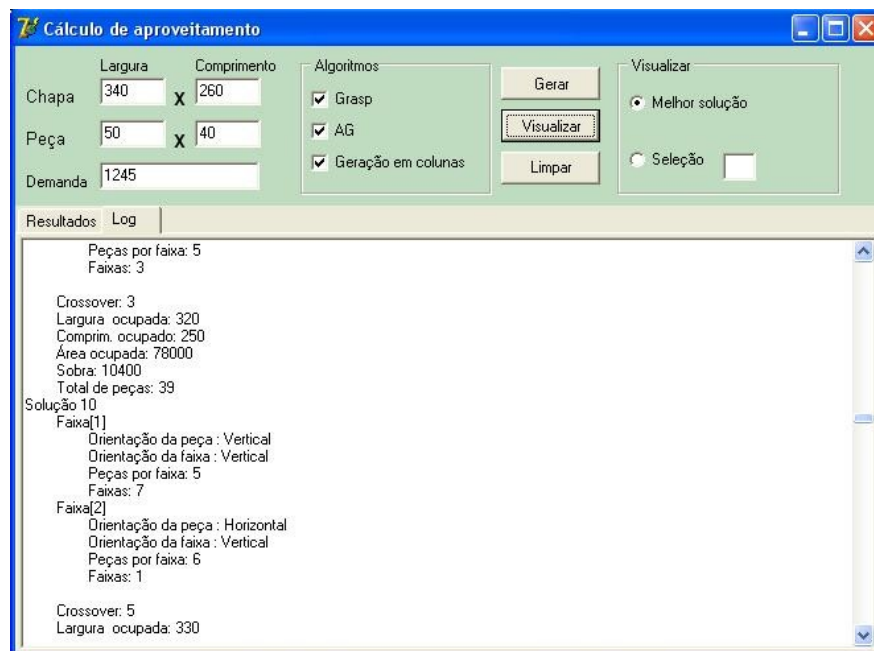


Figura 22: Layout da área de log

A tela principal chama um novo forms com a visualização do melhor padrão de corte encontrado, para cada um dos algoritmos selecionados.

Para gerar a visualização foi necessária a criação de três novos procedimentos, um que realiza o desenho da chapa e dentro deste foram utilizados outros dois que realizam a inicialização das coordenadas das faixas e das peças, para que não fiquem sobrepostas. Na Figura 23 podemos observar exemplos de padrões encontrados, cada uma das faixas gerada recebeu uma cor, para que possa ser identificada visualmente, sabendo assim a coordenada de cada uma.

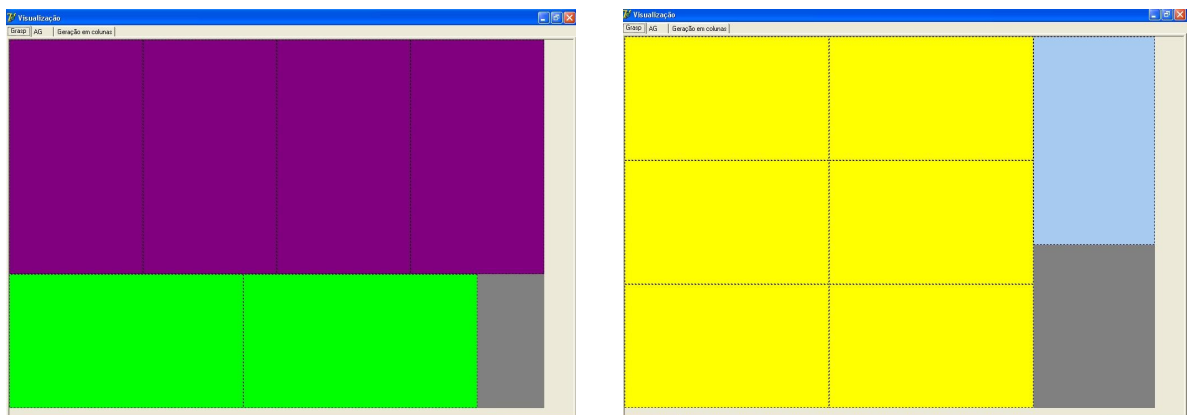


Figura 23: Exemplos de visualização dos padrões

Os padrões gerados somente permitem visualização, não sendo possível alteração manual das peças. Nesta tela os resultados são dispostos em 3 abas, uma para cada algoritmo.

5 TESTES E RESULTADOS OBTIDOS

Após a finalização dos algoritmos e da visualização iniciou-se os testes com as chapas e peças. Foram coletadas as dimensões de chapas padrões utilizadas pela empresa, juntamente com outros formatos aleatórios, o mesmo critério foi adotado para as peças e demanda.


Uma amostragem dos resultados gerados pelos experimentos podem ser observados nas Tabelas 3, 4 e 5.

Através dos resultados gerados é possível identificar algumas características de cada algoritmo aplicado.

- a) todos se mostraram estáveis para peças grandes e médias, não havendo oscilação significativa no tempo de processamento. Contudo o algoritmo GRASP teve uma avaliação melhor neste item;
- b) para peças menores o único que se manteve com o mesmo padrão de tempo de execução foi o algoritmo genético, os demais levaram um pouco mais de tempo para encontrar a solução, como se pode observar nas linhas 24 e 35, selecionadas nas tabelas. No algoritmo GRASP isso ocorreu devido ao fato do maior número de interações que precisa realizar para encontrar a solução onde, quanto menor for à peça, maior será esse valor. O que não ocorre com o algoritmo genético que encontra a solução com um número de interações menores, consequência da função randômica para realização do ponto de corte. Porém, isso pode refletir em diferentes resultados a cada execução. O algoritmo de geração em colunas foi o que apresentou o maior tempo de processamento para esse tipo de peça, como melhoria para esse algoritmo pode-se optar por limitante primal mais alto, proporcionando um número maior de podas na árvore, reduzindo assim a busca;
- c) algumas das soluções encontradas não foram compatíveis entre os algoritmos, sendo visualizada essa diferença nas linhas 20 e 39, onde foram gerados padrões com totais de peças diferentes;
- d) considerando as métricas definidas neste trabalho os algoritmos aplicados tiveram a seguinte ordem de desempenho: Algoritmos genéticos, método GRASP e método de geração de colunas;

Tabela 3: Tabela de resultados algoritmo genético

L	Chapa		Peça		Demanda	Peças	Nº Folhas	Sobra	Tempo
1	320	240	25	25	1200	108	12	12,1	15
2	320	240	12	5	5000	1280	4	0	15
3	320	240	30	15	2541	168	16	1,56	16
4	320	240	255	41	560	5	112	31,93	15
5	320	240	39	30	6871	64	108	2,5	16
6	960	660	320	240	15874	8	1985	3,03	16
7	960	660	211	23	1258	127	10	2,72	16
8	960	660	351	112	6584	12	549	25,54	31
9	960	660	752	80	8752	8	1094	24,04	16
10	2500	5000	220	150	2500	374	7	1,26	15
11	2500	5000	320	240	380	155	5	4,76	31
12	2500	5000	754	124	7824	128	62	4,26	16
13	2500	5000	880	550	5540	23	241	10,94	16
14	2000	3000	200	100	587	500	2	0	16
15	2000	3000	240	320	2584	75	35	4	16
16	2000	3000	540	620	421	15	17	16,3	16
17	2000	3000	710	85	5879	95	62	4,44	16
18	15000	2800	1200	2400	374	12	32	17,71	16
19	15000	2800	850	1400	789	35	23	0,83	15
20	15000	2800	1540	792	1241	28	45	18,68	32
21	15000	2800	4562	701	2541	9	283	31,47	16
22	3540	7890	1234	150	6984	146	48	3,24	16
23	3540	7890	300	200	784	455	2	2,25	15
24	3540	7890	150	57	13384	3236	5	0,94	16
25	3540	7890	789	354	7541	100	76	0	16
26	54200	8880	3250	2450	5942	54	111	10,66	16
27	54200	8880	2540	950	124	198	1	0,73	15
28	54200	8880	24500	3200	341	4	86	34,84	16
29	54200	8880	3200	3200	3574	32	112	31,91	15
30	3200	3200	400	40	2568	640	5	0	15
31	3200	3200	657	381	1256	36	35	11,99	16
32	3200	3200	650	650	4521	16	283	33,98	16
33	3200	3200	799	741	684	16	43	7,49	16
34	340	260	150	20	467	28	17	4,97	31
35	340	260	5	4	6247	4420	1	0	16
36	320	240	18	16	574	283	3	0,11	31
37	320	340	17	11	124	580	1	0,31	15
38	340	500	131	71	4522	17	266	6,99	16
39	660	500	73	8	2200	562	4	0,54	15
40	660	500	14	11	1254	2142	1	0,04	16

 Oscilação das quantidades de peças geradas no padrão encontrado, em comparação com as demais tabelas.



 Oscilação do tempo de processamento para peças menores, em comparação com as demais tabelas.

Tabela 4: Tabela de resultados algoritmo GRASP

L	Chapa		Peça		Demanda	Peças	Nº Folhas	Sobra	Tempo
1	320	240	25	25	1200	108	12	12,1	16
2	320	240	12	5	5000	1280	4	0	16
3	320	240	30	15	2541	168	16	1,56	16
4	320	240	255	41	560	5	112	31,93	16
5	320	240	39	30	6871	64	108	2,5	15
6	960	660	320	240	15874	8	1985	3,03	15
7	960	660	211	23	1258	127	10	2,72	15
8	960	660	351	112	6584	12	549	25,54	16
9	960	660	752	80	8752	8	1094	24,04	0
10	2500	5000	220	150	2500	374	7	1,26	16
11	2500	5000	320	240	380	155	5	4,76	0
12	2500	5000	754	124	7824	128	62	4,26	15
13	2500	5000	880	550	5540	23	241	10,94	0
14	2000	3000	200	100	587	500	2	0	16
15	2000	3000	240	320	2584	75	35	4	15
16	2000	3000	540	620	421	15	17	16,3	15
17	2000	3000	710	85	5879	95	62	4,44	16
18	15000	2800	1200	2400	374	12	32	17,71	0
19	15000	2800	850	1400	789	35	23	0,83	0
20	15000	2800	1540	792	1241	28	45	18,68	31
21	15000	2800	4562	701	2541	9	283	31,47	15
22	3540	7890	1234	150	6984	146	48	3,24	15
23	3540	7890	300	200	784	455	2	2,25	16
24	3540	7890	150	57	13384	3236	5	0,94	31
25	3540	7890	789	354	7541	100	76	0	0
26	54200	8880	3250	2450	5942	54	111	10,66	16
27	54200	8880	2540	950	124	198	1	0,73	16
28	54200	8880	24500	3200	341	4	86	34,84	15
29	54200	8880	3200	3200	3574	32	112	31,91	16
30	3200	3200	400	40	2568	640	5	0	16
31	3200	3200	657	381	1256	36	35	11,99	16
32	3200	3200	650	650	4521	16	283	33,98	15
33	3200	3200	799	741	684	16	43	7,49	16
34	340	260	150	20	467	28	17	4,97	16
35	340	260	5	4	6247	4420	1	0	31
36	320	240	18	16	574	283	3	0,11	62
37	320	340	17	11	124	580	1	0,31	16
38	340	500	131	71	4522	17	266	6,99	15
39	660	500	73	8	2200	559	4	1,07	16
40	660	500	14	11	1254	2134	1	0,41	31

 Oscilação das quantidades de peças geradas no padrão encontrado, em comparação com as demais tabelas.




 Oscilação do tempo de processamento para peças menores, em comparação com as demais tabelas.

Tabela 5: Tabela de resultados algoritmo geração de colunas

L	Chapa		Peça		Demanda	Peças	Nº Folhas	Sobra	Tempo
1	320	240	25	25	1200	108	12	12,1	16
2	320	240	12	5	5000	1280	4	0	47
3	320	240	30	15	2541	168	16	1,56	15
4	320	240	255	41	560	5	112	31,93	16
5	320	240	39	30	6871	64	108	2,5	16
6	960	660	320	240	15874	8	1985	3,03	16
7	960	660	211	23	1258	123	11	5,78	16
8	960	660	351	112	6584	10	659	37,95	16
9	960	660	752	80	8752	8	1094	24,04	0
10	2500	5000	220	150	2500	363	7	4,16	32
11	2500	5000	320	240	380	155	5	4,76	16
12	2500	5000	754	124	7824	128	62	4,26	16
13	2500	5000	880	550	5540	20	277	22,56	15
14	2000	3000	200	100	587	500	2	0	31
15	2000	3000	240	320	2584	75	35	4	16
16	2000	3000	540	620	421	15	17	16,3	16
17	2000	3000	710	85	5879	92	64	7,46	15
18	15000	2800	1200	2400	374	12	32	17,71	16
19	15000	2800	850	1400	789	35	23	0,83	16
20	15000	2800	1540	792	1241	27	46	21,59	47
21	15000	2800	4562	701	2541	9	283	31,47	16
22	3540	7890	1234	150	6984	146	48	3,24	16
23	3540	7890	300	200	784	442	2	5,05	31
24	3540	7890	150	57	13384	3224	5	1,3	62
25	3540	7890	789	354	7541	100	76	0	15
26	54200	8880	3250	2450	5942	48	124	20,58	15
27	54200	8880	2540	950	124	198	1	0,73	16
28	54200	8880	24500	3200	341	4	86	34,84	16
29	54200	8880	3200	3200	3574	32	112	31,91	16
30	3200	3200	400	40	2568	640	5	0	16
31	3200	3200	657	381	1256	32	40	21,77	0
32	3200	3200	650	650	4521	16	283	33,98	16
33	3200	3200	799	741	684	16	43	7,49	0
34	340	260	150	20	467	16	18	11,76	16
35	340	260	5	4	6247	4420	1	0	109
36	320	240	18	16	574	273	3	3,64	110
37	320	340	17	11	124	580	1	0,31	31
38	340	500	131	71	4522	17	266	6,99	16
39	660	500	73	8	2200	558	4	1,25	16
40	660	500	14	11	1254	2115	1	1,3	78

 Oscilação das quantidades de peças geradas no padrão encontrado, em comparação com as demais tabelas.

 Oscilação do tempo de processamento para peças menores, em comparação com as demais tabelas.

6 CONCLUSÃO

No desenvolvimento deste trabalho constatou-se que na comparação do algoritmo genético com o GRASP observou-se um desempenho melhor do GRASP, quando utilizadas peças médias e grandes. Contudo esse desempenho se reduz quando empregado o uso de peças menores.

O método de geração de colunas utiliza uma estrutura de árvore, onde dependendo do tamanho da árvore gerada o tempo gasto para encontrar a melhor solução pode ser maior que os demais algoritmos aplicados.

Pelos resultados e testes obtidos pode-se eleger como o melhor algoritmo para a situação exposta, o algoritmo genético, pois apresentou um grau de estabilidade maior que os demais.

Como trabalhos futuros, pode-se sugerir uma melhor avaliação nas formas parametrizáveis dos algoritmos GRASP e geração de colunas, assim como a aplicação de outros tipos de heurísticas não mencionados, ou o desenvolvimento completo de um *software* para a empresa utilizando os resultados obtidos neste trabalho.

Com a utilização desta solução no cenário estudado será possível realizar o cálculo do corte das peças com uma maior agilidade, atingindo o objetivo de otimizar o processo na empresa.

REFERÊNCIAS

ABIEA, **Associação Brasileira das Indústrias de Etiquetas Adesivas**. Disponível em: <<http://www.abiea.org.br/>>. Acesso em: 11 out. 2009.

ANDRADE, Carlos Eduardo de. Um algoritmo exato para o Problema de Empacotamento Bidimensional em Faixas. **Dissertação** (Mestrado Ciência da Computação) – Instituto de Computação, Universidade Federal de Campinas, Campinas, 2006.

ANNES, Ricardo. **Fundamentos de Programação Orientada a Objeto**. Disponível em:<<http://pucrs.campus2.br/~annes/delphi5.html>>. Acesso em: 16 ago. 2009.

AUDACES, **Planos de Corte**. Disponível em:<http://www.audaces.com/novo/pt/produtos/planos_corte_estofados.php>. Acesso em: 16 ago. 2009.

BAER, Lorenzo. **Produção Gráfica**. São Paulo, Editora Senac São Paulo, 6ª edição, 1995.

CANTÙ, Marco. **Dominando o Delphi 7, “A Bíblia”**. São Paulo: Pearson Education do Brasil, 2003.

CORTE CERTO, **Softwares para otimização de planos de corte**. Disponível em:<<http://www.cortecerto.com>>. Acesso em: 17 ago 2009.

FARIA, Anderson Oliveira. Otimização do Problema de Corte e Empacotamento Unidimensional Utilizando Algoritmos Genéticos. **Monografia** (Bacharelado em Ciência da Computação) - Universidade Federal de Lavras, Lavras, 2006.

GOLDBARG, Marco Cesar, LUNA, Henrique Pacca. **Otimização Combinatória e Programação Linear**. Rio de Janeiro: Campus Editora, 2000.

MARTINS, Camilla Brandel. **Apostila da Disciplina Análise de Algoritmos**. Disponível em: <<http://www.apostilando.com/download.php?cod=212&categoria=L%F3gica%20de%20Programa%E7%E3o>>. Acesso em: 23 nov. 2009.

NETO, Mário Carramillo. **Produção Gráfica II**. São Paulo: Global Editora, Coleção Contato Imediato, 1997.

PRESSMAN, Roger S.. **Engenharia de Software**. São Paulo: Makron Books do Brasil Editora Limitada, 2005.

ROZENFELD, Henrique, AMARAL, Daniel Capaldo, TOLEDO, José Carlos de, CARVALHO, Jonas de. Entenda hoje como sua indústria vai ser amanhã, **Revista Produtos & Serviços**. Edição n° 312. Dezembro de 2006.

PRONEWS, **Revista Pronews, Uma viagem na história da indústria gráfica**. Edição n° 106. Dezembro de 2008.

SCHILDT, Herbert. **C Completo e Total**. São Paulo: Makron Books, 1990.

SILVA, Cláudio Batista. **Banco de Dados Com Delphi**. Disponível em: <<http://www.apostilando.com/download.php?cod=3001&categoria=Delphi>> Acesso em: 23 nov. 2009.

SUZANO, **Papel e Celulose**. Disponível em: <<http://www.suzano.com.br/portal/main.jsplumChannelId=402880911995F9F401199A419A78677D>>. Acesso em: 17 ago. 2009.

TEMPONI, Elias Carlos Correa. Uma Proposta de Resolução do Problema de Corte Bidimensional via Abordagem Metaheurística. **Dissertação** (Mestrado em Modelagem Matemática e Computacional) - Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2007.

TOSCANI, Laira Vieira, VELOSO, Paulo A. S.. **Complexidade e Algoritmos**. Rio Grande do Sul: Bookmann Editora, 2008.

TRINDADE, Viviane de Aragão. Desenvolvimento e Análise Experimental da Metaheurística GRASP para um Problema de Planejamento de Sondas de Manutenção. **Dissertação** (Mestrado em Ciência da Computação) - Universidade Federal Fluminense, Niterói, 2005.

RECET, **Associação dos Centros Tecnológicos de Portugal**. Disponível em: <<http://www.recet.pt/pi/serigrafia.php?pag=4>>. Acesso em: 11 out. 2009.

KSR, **Distribuidora de papeis e produtos gráficos**. Disponível em: <<http://www.ksronline.com.br/servicos.php>>. Acesso em: 17 ago. 2009.

VELASCO, André Soares. Um Algoritmo Heurístico Baseado na GRASP para o Problema de Corte Bidimensional Guilhotinado Restrito. Artigo. Disponível em: <<http://revista.feb.unesp.br/index.php/gepros/article/download/218/168>>. Acesso em: 12 fev. 2010.

ZIMMERMANN, Marcos Paulo. Sistema de Apoio ao Corte Bidimensional Guilhotinado Aplicado ao Corte de Chapas de Papelão Utilizando Programação Linear. **Monografia** (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2002.