ABORDAGEM HÍBRIDA PARA O PROBLEMA DE CORTE DE ESTOQUE BIDIMENSIONAL

André Soares Velasco
Instituto Federal Fluminense - IFF/ Universidade Federal Fluminense - UFF
Av. Souza Mota, 350. Parque Fundão - Campos dos Goytacazes - RJ. CEP: 28060-010
asvelasco@iff.edu.br

Eduardo Uchoa
Dep. de Engenharia de Produção - Universidade Federal Fluminense - UFF
Rua Passo da Pátria 156, Bloco D. São Domingos - Niterói - RJ. CEP: 22210-240
uchoa@producao.uff.br

RESUMO

O presente trabalho tem como objeto de estudo o Problema de Corte de Estoque Bidimensional Guilhotinado, no caso sem rotação de itens. Este problema possui grande aplicabilidade em diversos setores produtivos que consideram as ações de corte na transformação de materiais em produtos semiacabados ou finais, tais como: metal mecânico, moveleiro, entre outros. Foi proposto o algoritmo híbrido VU_{2D} para tratar o problema em destaque e os conceitos para a sistematização deste método originaram-se de observações feitas a partir das ampliações do algoritmo Geração de Colunas Híbrido 2D (GCH_{2D}). Em todas as instâncias testadas, as soluções encontradas nunca ficaram mais do que 1 unidade além do limite inferior dado pela Geração de Colunas.

PALAVARAS CHAVE. Padrões de Corte Bidimensionais. Geração de Colunas. GRASP.

Tópicos (Metaheurística, Otimização Combinatória, Programação Matemática)

ABSTRACT

The present work has as object of study the Two-dimensional Guillotine Cutting Stock Problem, in the case without rotation of items. This problem has great applicability in several productive sectors that consider cutting actions in the transformation of materials into semi-finished or finished products, such as: mechanical metal, furniture, among others. It was proposed the hybrid algorithm VU_{2D} to treat the problem in focus and the concepts for the systematization of this method originated from observations made from the extensions of the 2D Hybrid Column Generation algorithm (GCH_{2D}). In all tested instances the solutions found were never more than 1 unit beyond the lower limit given by Column Generation.

KEYWORDS. Two-dimensional Cutting Patterns. Column Generation. GRASP.

Paper topics (Metaheuristics, Combinatorial Optimization, Mathematical Programming)

1. Introdução

O trabalho destaca uma variante clássica dos Problemas de Corte e Empacotamento, denominado Problema de Corte de Estoque Bidimensional Guilhotinado (PCEBG). Este problema é encontrado em setores produtivos de indústrias que consideram cortes ortogonais do tipo guilhotina em materiais retangulares (objetos) para produção de artigos menores e de mesmo formato (itens). Estes objetos apresentam-se estocados em número suficiente para atender uma demanda pré-estabelecida de itens, efetuando cortes paralelos a dois dos seus lados, em sua quantidade mínima. A Figura 1 exibe os principais elementos do PCEBG, com objetos em estoque apresentando dimensões iguais, itens a produzir em quantidades pré-fixadas e padrões de corte a serem processados de forma eficiente para atender essa demanda.

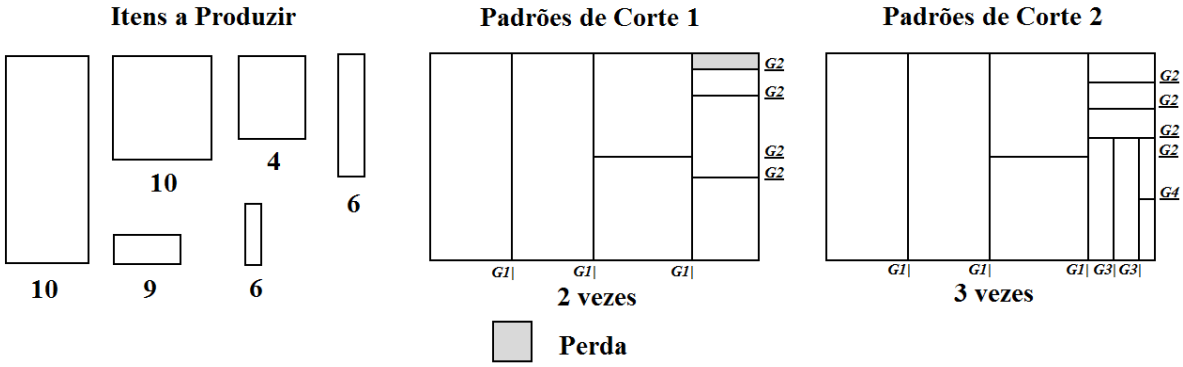

FIGURA 1 – Exemplo do Problema de Corte de Estoque Bidimensional Guilhotinado.

Outra variante em destaque é o Problema de Corte Bidimensional Guilhotinado (PCBG) nos casos com restrição de demanda (restrito) e sem essa restrição (irrestrito), que também são distinguidos de acordo com o valor de utilidade atribuído aos itens e em relação à permissão de rotação destes. O valor de utilidade pode ser a medida de sua área (sem valor) ou estar relacionado à sua importância em presença aos outros itens (com valor). Uma rotação de 90º nos itens pode ser aceita (com rotação) ou impedida (sem rotação). Os cortes ortogonais podem ser guilhotinados e gerar dois novos retângulos ou não guilhotinados. Com as características supracitadas, a quantidade de cada item produzido a partir de cortes guilhotina pode ser limitada (PCBGR) ou ilimitada (PCBGI). Já o número de estágios no padrão de corte está relacionado à quantidade de mudanças permitidas na direção dos cortes guilhotina, ou seja, pode-se admitir apenas k rotações de 90º nas direções destes cortes (k-estagiado) ou não restringir esta quantidade (não estagiado). A Figura 1 destaca padrões de corte com dois e quatro estágios, indicando a sequencia dos k estágios por Gk. Todas essas variantes do PCBG, assim como o PCEBG correspondente, pertencem à classe NP-difícil e a dificuldade na determinação de tais soluções ótimas se deve ao grande número de padrões de corte viáveis estar relacionado ao crescimento de tais instâncias. Na literatura do PCEBG, destacam-se os métodos propostos por [Cintra et al. 2008] e [Furini et al. 2012].

Para tratar o PCEBG, o algoritmo GCH_{2D} proposto por [Velasco e Uchoa 2015] segue a ideia de arredondar a solução fracionária da Geração de Colunas (GC) para uma solução inteira e resolver o Problema Residual, definido pelos itens com demanda ainda não atendida, utilizando sistematicamente o algoritmo RG_{2Da} na produção de padrões com esses itens remanescentes.

O presente trabalho propõe o algoritmo híbrido VU_{2D}, baseado no GRASP Reativo [Prais e Ribeiro 2000], para resolver o PCEBG sem rotação e não estagiado. Além da metodologia metaheurística adotada, sua arquitetura possui a base o algoritmo GCH_{2D} e destaca a utilização de três estratégias consideradas ampliações deste algoritmo. Basicamente, estas são: a inserção de novas colunas ao PL Mestre, baseadas nas inequações cortes duais propostas por [Valério de Carvalho 2005], que são geradas com a aplicação do algoritmo RG_{2Da}; a utilização do algoritmo RG_{2Dv} no reaproveitamento de padrões de corte com produção de itens em excesso após o arredondamento da solução ótima do PL Mestre; e a geração de múltiplas colunas para o

PL Mestre, via Programação Dinâmica (PD) irrestrita ou combinando esta com a versão reativa RG_{2Dv}. Para avaliar o desempenho do algoritmo proposto VU_{2D}, foram realizados testes em 48 instâncias do PCEBG e comparados com os algoritmos GCH_{2D}, CG e CG^P [Cintra et al. 2008].

2. Formulação do PCEBG

Com os padrões de corte bidimensionais guilhotinados definidos e uma quantidade de objetos em estoque suficiente para atender a uma demanda de itens pré-estabelecida, o PCEBG pode ser modelado como um problema de Programação Linear Inteira. A formulação a seguir, consiste em determinar o número de vezes que cada padrão de corte é utilizado, de forma a atender essa demanda, consumindo-se o menor número possível de objetos.

$$\text{Min} \quad \sum_{j=1}^{n} x_j$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_{ij}\, x_j \geq d_i \;,\; i = 1,...,m \qquad (2.1)$$

$$x_j \geq 0 \;\text{ e }\; \text{Inteiro},\; j = 1,...,n$$

Onde:

- x_j é a variável que representa o número de vezes que o padrão de corte j é utilizado;
- a_{ij} é a constante que indica número de itens i gerados no padrão de corte j;
- d_i é a constante que indica a demanda de itens i;
- m é a constante que indica o número de itens distintos;
- n é a constante que indica o número total de padrões de corte.

O valor de *n* é tipicamente muito grande, impedindo que a Relaxação Linear dessa formulação seja resolvida diretamente por um método como o Simplex. Esse impedimento pode ser contornado com o método de geração de colunas [Dantizg e Wolfe 1960]. A Geração de Colunas pode ser vista como uma extensão do Simplex Revisado, onde o *pricing* das variáveis é realizado através da resolução de um subproblema, e foi primeiramente introduzida por [Gilmore e Gomory 1961] para o PCE Unidimensional. Na Geração de Colunas, o chamado Problema Linear Mestre (PL Mestre) é inicializado com apenas um pequeno subconjunto das variáveis, correspondendo a alguns padrões de corte. A cada iteração, é gerado e inserido um novo padrão, desde que ele possa melhorar o valor da função objetivo, até que se prove que a solução corrente é ótima. Sendo um problema de minimização, deve-se buscar por padrões que levem a variáveis com custo reduzido negativo. Isso resulta no seguinte subproblema:

$$\text{Min} \quad (1 - \sum_{i=1}^{m} \pi_i y_i)$$

$$\text{s.a:} \quad [y_1, ..., y_m]^t \text{ é um padrão viável} \qquad (2.2)$$

$$y_i \geq 0 \;\text{ e }\; \text{Inteiro}$$

Onde:

- y_i é a variável que representa o número de itens i no padrão de corte;
- π_i são as variáveis duais para cada restrição i do PL Mestre.

Neste trabalho, esse subproblema é abordado como o problema da geração de padrões não estagiados para o PCBGI com valor. Assim, para gerar tais padrões irrestritos, utiliza-se o algoritmo *Dynamic Programming* (DP) apresentado em [Cintra et al. 2008], que combina o algoritmo *Discretization using Dynamic Programming* (DDP), proposto por [Herz 1972] para obter os pontos de discretização de um padrão canônico, com as fórmulas de recorrência de [Beasley 1985].

Além disso, diante à dificuldade de resolver esse problema restrito de forma exata e da necessidade de se obter soluções restritas nas respectivas instâncias residuais, que possibilitam melhorar o desempenho dos algoritmos apresentados para o PCEBG, o trabalho considera fundamental o emprego dos algoritmos GRASP Reativos RG_{2Da} e RG_{2Dv}, nos casos sem valor e

com valor do PCBGR. As evoluções do algoritmo GRASP-2D [Velasco et al. 2008] que culminaram inclusive nessas versões reativas do RG_{2D}, podem ser conferidas nos respectivos trabalhos [Velasco e Uchoa 2014] e [Velasco e Uchoa 2016]. Nas próximas seções são apresentadas as ampliações do GCH_{2D} utilizadas na criação do novo algoritmo VU_{2D}.

3. Ampliações do Algoritmo GCH_{2D} para o PCEBG

Inicialmente, o algoritmo GCH_{2D} foi proposto por [Velasco e Uchoa 2015] para resolução do PCEBG sem rotação e não estagiado. Considerando uma instância com m itens diferentes, este é iniciado com o PL Mestre contendo um conjunto de m colunas de padrões homogêneos, com os itens produzidos em sua quantidade máxima em cada objeto, e mais uma coluna gerada a partir da PD irrestrita, com o valor de utilidade dos itens igual às respectivas medidas de área. Caso a solução ótima do PL Mestre não seja inteira, a cada iteração o valor dos itens são atualizados pelo valor das suas respectivas variáveis duais correntes e é gerada uma nova coluna via PD irrestrita. Se algum padrão gerado não levar a um custo reduzido negativo ou o PL Mestre ainda não apresentar solução ótima inteira, a instância residual é obtida a partir do arredondamento para o maior inteiro menor ou igual à solução relaxada corrente. Neste momento, executa-se o algoritmo RG_{2Da} na instância do problema residual até que seus padrões de corte restritos completem a solução inteira do PCEBG original.

Sabendo que a inserção de uma coluna ótima obtida ao resolver o subproblema com o algoritmo DP, atribuindo o valor de utilidade aos itens de acordo com suas variáveis duais no PL Mestre corrente, pode implicar em grande esforço computacional de acordo com o tamanho dos itens e objeto. Além disso, há uma forte tendência de que a solução inteira arredondada da GC apresente padrões de corte com produção em excesso de itens menores e demanda concentrada em outros maiores, criando assim, uma relativa dificuldade na resolução do Problema Residual. Diante destas dificuldades, as subseções seguintes apresentam três diferentes procedimentos indicados como ampliações do algoritmo GCH_{2D}.

3.1 Geração de Colunas VCG

A proposta feita por [Valério de Carvalho 2005] para o Problema de Corte de Estoque Unidimensional reduz consideravelmente o tempo computacional da geração de colunas, com a diminuição do número de colunas geradas e de iterações degeneradas através da restrição do espaço dual do PL Mestre com uma família de cortes duais válidos. Eles equivalem a um novo tipo de coluna que possui coeficiente -1 em certo item i e coeficiente 1 em algum item j menor do que i, no sentido que j sempre cabe no espaço ocupado por i. De modo mais geral, pode haver colunas com coeficiente 1 em um conjunto de itens J tal que todos eles caibam no espaço ocupado por i, ou até coeficiente maiores que 1, indicando mais de uma cópia de um item podem ocupar o espaço de i.

Adaptada a ideia para o PCEBG, a coluna denominada VCG é inicializada como um vetor coluna nulo de tamanho m, onde cada item i pode ser considerado um objeto e é indicado com coeficiente -1 na posição i deste vetor. A substituição deste item i por outros menores é encarada como um PCBGR, com o algoritmo RG_{2Da} se encarregando de construir um padrão de corte, possivelmente não homogêneo, com perda menor ou igual ao parâmetro $perda_{VCG}$. Este parâmetro corresponde ao percentual de perda aceitável na substituição que caracteriza uma coluna VCG. O procedimento Cols_VCG insere estas colunas no PL Mestre antes da sua execução e após sua convergência, o Problema Residual é obtido da seguinte forma:

- As variáveis correspondentes às colunas normais de padrões são arredondadas para baixo e produzem uma solução inteira parcial;
- As variáveis correspondentes às colunas VCG também são arredondadas para baixo e indicam quais trocas devem ser realizadas na solução inteira parcial;
- O Problema Residual é dado pela demanda não atendida nessa nova solução.

É importante ressaltar que a utilização de colunas VCG no PL Mestre não apenas reduziu o tempo de execução da GC, o que já era esperado, mas também tiveram o efeito de gerar Problemas Residuais mais fáceis de serem tratados via RG_{2Da}. Acredita-se que isso aconteça porque as colunas VCG fazem com que o PD gere mais padrões contendo itens grandes, se tornando mais atrativos porque podem ser convertidos em itens menores nas substituições. Sendo

assim, o Problema Residual apresenta-se com menos itens grandes e com mais itens pequenos, e estes itens são mais facilmente encaixados nos objetos com a utilização do algoritmo RG_{2Da}.

3.2 Reaproveitamento de Padrões com Produção Excedente

A presente subseção é destinada ao procedimento denominado REAP, que considera o aprimoramento dos padrões da solução inteira arredondada responsável pela produção de determinados itens além da demanda. Neste contexto, no considerado processo de otimização que utiliza padrões de corte gerados iterativamente, admitindo-se à possibilidade de produção de itens em quantidade excedente, o fato de considerar o reaproveitamento destes padrões incorpora diferencial competitivo com a promovida redução de tamanho do Problema Residual.

O processo de reaproveitamento inicia-se depois de se obter a solução ótima relaxada do PL Mestre, com as devidas modificações atribuídas a existência de variáveis básicas VCG, identificando os padrões de corte x_j cujas variáveis básicas são maiores ou iguais a 1 e apresentam coeficiente positivo na linha dos possíveis itens com produção acima da demanda.

Sejam os conjuntos de itens produzidos acima da demanda e de padrões de corte com possibilidade de reaproveitamento, chamados respectivamente de I_{Subs} e P_{Reap}. Para cada padrão de corte na solução inteira aproximada, se apresenta algum item de I_{Subs}, então este constitui-se um padrão de P_{Reap}. Nos padrões de P_{Reap}, os itens com produção excedente e pertencente à I_{Subs} são excluídos para que seus espaços, juntamente com as perdas em tais padrões, possam ser reaproveitados por novos itens, com demanda em falta e que caibam nessa área, pertencentes ao conjunto I_{Reap}. Para cada padrão de P_{Reap}, aos itens que não possuem produção remanescente são atribuídos novos valores de utilidade que garantam sua produção no padrão reaproveitado e estes itens vão compor o conjunto I_{Imp}. Como mencionado, se possível, o padrão deve ser reaproveitado com a execução do algoritmo RG_{2Dv}, a partir de uma instância contendo os itens de I_{Imp} e I_{Reap}. Na consolidação dessa melhoria, consequência da maior valorização do padrão de corte, o número de repetições do novo padrão reaproveitado está condicionado ao menor valor inteiro que considere:

- O número de reproduções do padrão original na solução inteira aproximada;
- A razão entre a produção excedente de cada item de I_{Subs} e suas respectivas quantidades na coluna corrente;
- A demanda residual dos itens de I_{Reap} incluídos nesse reaproveitamento.

Em seguida, sempre que houver o reaproveitamento de um padrão de corte, as informações relativas à solução inteira aproximada e aos dados da instância do Problema Residual, assim como os elementos dos conjuntos I_{Subs}, I_{Reap} e I_{Imp}, devem ser atualizadas.

Na execução do algoritmo GCH_{2D}, a resolução do subproblema com um método exato para o PCBGI, conduz o PL Mestre para uma solução ótima que apresenta a grave tendência de garantir a produção dos itens menores em depreciação aos maiores com o arredondamento. Dessa forma, uma demanda residual constituída apenas de itens maiores, com dimensões que não favorecem os devidos reaproveitamentos das perdas internas e dos itens menores com excedentes, atribui considerável dificuldade à utilização da ampliação REAP.

Analisando o processo realizado com procedimento REAP e o ganho promovido pela inserção de colunas VCG, que ao considerar a troca de itens maiores por menores promove em ação contrária um impedimento da produção em excesso destes menores e um incremento na produção dos itens maiores envolvidos nessa substituição, a ação conjunta destas ampliações torna-se interessante uma vez que as prováveis trocas obtidas com as colunas VCG produzem ganhos que não são possíveis apenas com o REAP e, também, a possibilidade de se ter itens maiores em I_{Subs} aumentaria as chances de obter um reaproveitamento de valor significativo.

3.3 Geração de Múltiplas Colunas

No algoritmo GCH_{2D} original, durante o processo de convergência da GC, uma coluna fornecida ao PL Mestre provém da resolução do subproblema via PD, com o padrão de corte canônico maximal, obtido com cortes do tipo guilhotina nos respectivos pontos de discretização. Quando as dimensões do objeto são relativamente grandes quando comparadas as mesmas nos itens ou a instância possui muitos itens com dimensões singulares, as respectivas combinações cônicas acarretam em muitos pontos de discretização, promovendo reconhecida dificuldade

computacional. Por outro lado, enquanto não se obtém a solução ótima do PL Mestre, é possível encontrar outras colunas com custo reduzido negativo, além da coluna maximal, indicadas por y_u.

Admitindo que a resolução desta fase apresente grande esforço computacional ao calcular para cada subproblema os considerados pontos de discretização e sua coluna associada ao padrão maximal, comparados ao tempo relacionado à resolução do Problema Residual com o RG_{2Da}, reconsidera-se a atualização do PL Mestre, com a inserção não mais de uma coluna e sim de todas que apresentem custo reduzido negativo. Por consequência, reduz-se de forma significativa o tempo de processamento desta etapa em instâncias consideradas mais difíceis.

Geralmente, a solução ótima do PL Mestre em questão não é única. Nesse processo, caso a variável associada à coluna de um padrão de corte não maximal, com perda substituível por um item p_i com valor de utilidade π_i positivo e, ainda assim, atende a demanda d_i deste item ao entrar na base da solução ótima corrente, a sequência de bases visitadas pelo Método Simplex, não garante a troca desta coluna por outra que, além de possuir os mesmos itens, as suas respectivas quantidades garantem o *status* maximal. Uma vez que a variável supracitada não sai da base e sua permanência junta às demais vem atender esta demanda d_i sem produção em excesso, a folga da restrição de p_i é nula. Assim, pelo Teorema das Folgas Complementares, o valor atualizado da variável dual π_i pode ser zero ou não, e isto dificulta a convergência para solução ótima e enfraquece a operacionalidade das estratégias baseadas nas colunas VCG e no procedimento REAP.

O novo procedimento MultCols_PDG considera a cada chamada do subproblema, a geração de novas colunas a partir de uma cooperação entre a PD irrestrita e o algoritmo RG_{2Dv}. Nesta colaboração, os padrões de corte y_u produzidos pela PD que possuem perda total superior a área do menor item p_a, com valor de utilidade π_a positivo no subproblema corrente, têm seus respectivos itens incluídos no conjunto I_{Imp} e os demais itens com valor de utilidade positivo e área menor ou igual a perda total, em quantidades que o somatório das respectivas áreas não ultrapasse a perda em questão, vão constituir o conjunto I_{Mel}. Assim, como no procedimento REAP, aos itens pertencentes à I_{Imp} são atribuídos novos valores de utilidade que garantam as respectivas quantidades no padrão gerado pelo RG_{2Dv}. Necessariamente, as colunas associadas a este novo padrão de corte y_u é adicionada ao PL Mestre se conter todos os itens do padrão original e pelo menos um item de I_{Mel}. Caso a perda total dos padrões y_u não supere o valor da área do item p_a, estes são incorporados diretamente ao PL Mestre.

4. Algoritmo VU_{2D} para o PCEBG

Nesta seção é apresentado o algoritmo VU_{2D} para o caso não estagiado e sem rotação dos PCEBG. Este algoritmo é fruto de um delineamento que não apenas considera a inclusão das variáveis VCG, o reaproveitamento de padrões de corte com produção excedente e a geração de múltiplas colunas com os algoritmos PD e RG_{2Dv}, inseridas ao PL Mestre enquanto apresentam custos reduzidos negativos, além da versão reativa do RG_{2Da} na resolução do Problema Residual, mas também propõe uma abordagem mais geral para resolução do PCEBG, baseada na metodologia GRASP Reativo.

A concepção dessa proposta constitui-se no fato de que o processo utilizado pelo algoritmo GCH_{2D}, para constituir a solução do Problema Residual, pode não ser tão eficiente ao sempre considerar, após uma iteração para cada valor de $\alpha \in A$, o padrão de maior valor na sequência desta composição, e conduzir prematuramente a ótimos locais. Uma questão importante a ser considerada é que, desta forma o processo não vem se beneficiando de informações obtidas em estágios anteriores. Isto é, os padrões não apresentam vínculos e são produzidos de maneira completamente independentes, deixando muitas vezes de explorar sistematicamente o espaço de soluções na vizinhança corrente.

Como resultado destas deduções e considerando que estratégias adaptativas para o parâmetro α contribuem para o sucesso da técnica GRASP, principalmente, quando aplicada a problemas cuja solução ótima apresenta certa tendência gulosa, são apresentados os mecanismos utilizados nas fases características da Geração de Colunas e do Problema Residual do algoritmo VU_{2D} a seguir.

4.1 Geração de Colunas no Algoritmo VU_{2D}

A estrutura do PL Mestre nos algoritmos VU_{2D} diferencia-se da formulação (2.1) relaxada e, portanto, da apresentada nos algoritmos GCH_{2D}. Isto se deve as colunas VCG que são incialmente geradas e introduzidas juntamente com as colunas homogêneas, distintas por apresentarem a quantidade máxima de cópias produzidas de um respectivo item no objeto, e uma coluna maximal obtida via PD quando os itens possuem valor de utilidade igual a área. Após a primeira rodada deste PL Mestre, calculam-se as variáveis duais associadas às restrições do modelo, fazendo com que os valores destas configurem como os respectivos novos valores dos itens. Na sequência, a atualização desse modelo se dá com a resolução do subproblema sendo realizada pelo procedimento MultCols_PDG, onde encarrega-se de garantir não somente uma nova coluna ao PL Mestre, mas todas as colunas que têm custo reduzido negativo e perda total inferior a área do menor item com valor positivo na iteração corrente. O presente processo encerra-se com a impossibilidade do algoritmo supracitado produzir novas colunas que melhorem a convergência do PL Mestre para o valor ótimo da instância em questão.

Finalmente, a partir da solução ótima do PL Mestre terminal, é determinada uma solução inteira aproximada, arredondando-se para baixo os valores das variáveis básicas na solução relaxada. Ainda nesta solução inteira, é feito uma tentativa de reaproveitamento dos padrões de corte, com itens sendo produzidos em quantidade acima do solicitado, a partir do procedimento REAP, para então definir a instância do Problema Residual.

4.2 Problema Residual nos Algoritmos VU_{2D}

Na fase destinada à resolução do Problema Residual, é proposto um algoritmo baseado no método GRASP Reativo, atuando no processo de seleção dos padrões de corte que integram a melhor solução do Problema Residual. Nessa etapa do algoritmo VU_{2D}, não se institui como critério de parada satisfazer a demanda completamente, assim como nos algoritmos GCH_{2D}, mas um número máximo de iterações na determinação e melhor utilização parâmetro α_{pce} reativo.

Basicamente, inicia-se com uma fina calibragem do parâmetro α_{pce}. Nesta primeira fase de diversificação, com $iter_{Cpce}$ iterações para cada $\alpha_{pce} \in A_{pce} = \{0.92, 0.94, 0.96, 0.98, 1\}$, analisando as respectivas frequências $f\alpha_{pce}$ e os valores $valorS_{pce}$ das melhores soluções S_{pce} encontradas, especifica-se o parâmetro reativo α_{Rpce} e a melhor solução desta fase para o PCEBG. A fase reativa e de finalização do VU_{2D} consiste em executar o um número distinto de $iter_{Rpce}$ iterações, agora com $\alpha_{pce} \in A'_{pce} = \{\alpha_{Rpce}, \alpha_{Rpce} \pm 0.01\}$, buscando ótimos locais superiores ou até um ótimo global no espaço de soluções considerado.

Na fase de construção característica da GRASP para o PCEBG, os padrões de corte S_{pc} são produzidos com o algoritmo RG_{2Da}, quando executados uma única iteração para cada $\alpha_{pc} \in A_{pc} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, com respectivo valor total de utilidade $valorS_{pc}$ dado pelo somatório das medidas de área dos itens relacionados. Seja C_{pc} um conjunto finito formado pelos melhores padrões de corte S_{pc} para cada α_{pc} e $\beta_{pc} = \max\{valorS_{pc}; S_{pc} \in C_{pc}\}$ o valor do melhor padrão em C_{pc}, a $LRC_{pc} = \{S_{pc} \in C_{pc} / \alpha_{pce}.\beta \leq valorS_{pc} \leq \beta\}$ é constituída dos padrões de corte mais significantes para compor a solução S_{pce} do PCEBG. É importante ressaltar que este parâmetro α_{pce}, com tendência fortemente gulosa, regula a quantidade e, principalmente, a qualidade dos padrões presentes na lista LRC_{pc}. Sendo assim, para cada iteração com α_{pce}, uma solução corrente S_{pce} é construída considerando sempre a seleção aleatória de um padrão de corte S_{pc} em LRC_{pc}.

Para fase de busca local da GRASP, é proposta uma estratégia de movimentos que consiste em atribuir aos itens um diferente valor de utilidade, adicionando sua medida de área ao número do padrão de corte em que sua demanda é atendida na solução vigente, multiplicado por um bônus. Este bônus varia de acordo com o tamanho da instância do problema, ou seja, com a quantidade de itens e suas respectivas dimensões, e o valor adotado é 10% da média das medidas de área dos itens distintos do Problema Residual. Posteriormente, com as devidas atualizações no valor dos itens e o objetivo de aprimorar a solução, a instância residual corrente é resolvida pelo algoritmo RG_{2Dv} até atingir um número $iter_{bl}$ de iterações especificadas.

Em seguida, é representado no pseudocódigo do algoritmo VU_{2D} e alguns dos parâmetros e das variáveis utilizadas são descritas a seguir.

- S_R^* : conjunto de padrões de corte da melhor solução do PCEBG;
- Z_{SR}^* : número total de objetos na melhor solução do PCEBG;
- S_{rel}^* : solução relaxada ótima do PL Mestre;
- S_{apr} : solução inteira aproximada do PL Mestre;
- Z_{Sapr} : Número de objetos na solução inteira aproximada;
- $itens_{PR}$: número total de itens distintos da instância residual.

Algoritmo VU_{2D} ($perda_{VCG}$, $iter_{Cpce}$, $iter_{Rpce}$, $iter_{bl}$)

1. $S_R^* = \emptyset$, $Z_{SR}^* = \infty$, $f\alpha_{Rpce} = 0$;
2. Gere as colunas VCG para o PL Mestre, executando *Cols_VCG*;
3. Determine as colunas $[y_1,..., y_m]^t$ de padrões homogêneos para o PL Mestre;
4. Gere nova coluna para o PL Mestre executando *DP,* com $v(p_i) = c_i.l_i$;
5. Resolva o PL Mestre;
6. Calcule as variáveis duais π_i;
7. Gere múltiplas colunas para o PL Mestre executando *MultCols_PDG*, com $v(p_i) = \pi_i$;
8. **Se** $(1 - \sum_{i=1}^{m} \pi_i y_u < 0)$ **então** Adicione as novas colunas y_u ao PL Mestre e volte ao passo 5;
9. Determine a solução inteira aproximada arredondando para S_{rel}^* baixo, S_{apr} e Z_{Sapr};
10. Execute *REAP* para os possíveis reaproveitamentos e atualize S_{apr} e Z_{Sapr};
11. Determine dados da instância residual e $itens_{PR}$;
12. Atualize $S_R^* = S_{apr}$ e $Z_{SR}^* = Z_{Sapr}$;
13. **Se** $(itens_{PR} > 0)$ **então**
14. **Para** $(\alpha_{pce} \in A_{pce})$ **faça**
15. **Para** $(iter \leq iter_{Cpce})$ **faça**
16. **Para** $(\alpha_{pc} \in A_{pc})$ **faça**
17. Execute RG_{2Da}, com *maxiter* = 1 e $\alpha = \alpha_{pc}$;
18. Escolha aleatoriamente um padrão de corte em LRC_{pc};
19. Atualize dados da instância residual e $itens_{PR}$;
20. **Para** $(iter \leq iter_{bl})$ **faça**
21. Execute *busca local* na S_{pce} corrente;
22. **Se** $(Z_{SR}^* > Z_{Spce})$ **então**
23. Atualize $S_R^* = S_{pce}$, $Z_{SR}^* = Z_{Spce}$, $\alpha_{Rpce} = \alpha_{pce}$, $f\alpha_{Rpce} = 1$;
24. **Se** $(Z_{SR}^* = Z_{Spce})$ **então**
25. **Se** $(f\alpha_{Rpce} < f\alpha_{pce})$ **então**
26. Atualize $S_R^* = S_{pce}$, $Z_{SR}^* = Z_{Spce}$, $\alpha_{Rpce} = \alpha_{pce}$, $f\alpha_{Rpce} = f\alpha_{pce}$;
27. **Se** $(f\alpha_{Rpce} = f\alpha_{pce})$ **então**
28. Escolha, aleatoriamente, α_{Rpce} e atualize S_R^*, Z_{SR}^* e $f\alpha_{Rpce}$;
29. **Para** $(\alpha_{pce} \in A'_{pce})$ **faça**
30. **Para** $(iter \leq iter_{Rpce})$ **faça**
31. **Para** $(\alpha_{pc} \in A_{pc})$ **faça**
32. Execute RG_{2Da}, com *maxiter* = 1 e $\alpha = \alpha_{pc}$;
33. Escolha aleatoriamente um padrão de corte em LRC_{pc};
34. Atualize dados da instância residual;
35. **Para** $(iter \leq iter_{bl})$ **faça**
36. Execute *busca local* na S_{pce} corrente;
37. **Se** $(Z_{SR}^* > Z_{Spce})$ **então**
38. Atualize $S_R^* = S_{pce}$, $Z_{SR}^* = Z_{Spce}$;
39. Retorna (S_R^*, Z_{SR}^*);

Fim VU_{2D}

5. Resultados Computacionais

Os algoritmos abordados para o PCEBG foram implementados em linguagem C/C++, as relaxações do PL Mestre foram resolvidas pelo CPLEX Optimization Studio 12.5.1 e os testes foram realizados em um computador com processador Intel(R) Core(TM)i5-3320M 2.60 GHz, com 4GB de memória RAM e sistema operacional Windows 7 de 64 bits.

A seguir, são apresentados os testes computacionais com os algoritmos GCH_{2D} e VU_{2D}, que se distinguem em dois grupos de instâncias pelo grau de dificuldade na sua resolução. No primeiro grupo, tem-se os problemas inicialmente propostos por [Beasley 1985] para o PCBGI,

adaptadas e disponibilizadas por [Cintra et al. 2008]. Assim como em [Velasco e Uchoa 2015], os resultados nestas instâncias são comparadas com os obtidos pelos algoritmos CG e CG^P [Cintra et al. 2008], com estes algoritmos sendo implementados em linguagem C e executados em um computador com processador Intel Pentium IV 1.8 GHz, com 512 Mb de memória RAM, sistema operacional Linux e resolvedor CLP (COIN-OR LP Solver). Vale ressaltar que no benchmark de endereço www.cpubenchmark.net, a informação é que os testes executados em tal processador são 3.8 vezes mais lentos, quando a comparação é feita com o processador utilizado nas rodadas dos algoritmos GCH_{2D} e VU_{2D}.

Nas execuções do algoritmo VU_{2D}, adota-se a configuração dos parâmetros $perda_{VCG}$ = 10% e $iter_{bl}$ = 1 nas ações de melhoria. Também são fixados $iter_{Cpce}$ = 40 e $iter_{Rpce}$ = 100 como máximo de iterações promovidas nas respectivas fases de calibragem e reativa, totalizando 500 iterações a cada rodada como critério de parada. Nos algoritmos RG_{2Da} e RG_{2Dv}, opta-se pelos parâmetros $\psi = 0$, $\varphi = \delta = 1$, *pitens* = 10%, com os valores do parâmetro α de 0.1 a 1 e incrementados de 0.1, executando apenas 1 iteração para cada valor de α na fase de diversificação, em que o melhor padrão gerado é escolhido para compor a solução.

TABELA 1 – Resultados do Algoritmo GCH_{2D}.

Instância	m	Dimensões Objeto	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução RG_{2D}	Tg(s)	Solução GCH_{2D}	T(s)	Solução GCH_{2Dm}	Tm(s)	Solução CG/CGp	Tc(s)
gcut1d	10	(250, 250)	21	293.25	292	2	<0.01	294	<0.01	294.0	0.01	294	0.03
gcut2d	20	(250, 250)	35	344.25	343	2	<0.01	345	<0.01	345.0	0.01	345	0.28
gcut3d	30	(250, 250)	75	331.50	320	12	0.01	*332	0.03	332.8	0.04	333	0.77
gcut4d	50	(250, 250)	93	835.83	823	13	0.02	*836	0.08	836.9	0.07	837	5.09
gcut5d	10	(500, 500)	22	196.83	194	3	<0.01	*197	<0.01	197.6	0.01	198	0.03
gcut6d	20	(500, 500)	43	342.67	338	5	<0.01	*343	0.01	343.5	0.01	344	0.21
gcut7d	30	(500, 500)	53	591.00	585	6	<0.01	*591	0.01	591.9	0.01	592	0.33
gcut8d	50	(500, 500)	133	690.00	675	16	0.02	*691	0.08	691.5	0.10	692	3.60
gcut9d	10	(1000, 1000)	24	130.67	126	5	<0.01	*131	<0.01	131.2	0.01	132	0.06
gcut10d	20	(1000, 1000)	33	293.00	290	3	<0.01	293	0.01	293.0	0.01	293	0.09
gcut11d	30	(1000, 1000)	74	329.38	318	12	0.01	*330	0.04	330.7	0.04	331	1.07
gcut12d	50	(1000, 1000)	114	671.50	662	10	0.01	672	0.06	672.2	0.07	672	3.35

TABELA 2 – Resultados do Algoritmo VU_{2D}.

Instância	m	Dimensões Objeto	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução RG_{2D}	Tg(s)	Solução VU_{2D}	T(s)	Solução VU_{2Dm}	Tm(s)	Solução CG/CGp	Tc(s)
gcut1d	10	(250, 250)	26	293.25	292	2	0.14	294	0.21	294.0	0.22	294	0.03
gcut2d	20	(250, 250)	106	344.25	343	2	0.12	345	0.41	345.0	0.42	345	0.28
gcut3d	30	(250, 250)	227	331.50	322	10	1.85	*332	2.84	*332.0	2.86	333	0.77
gcut4d	50	(250, 250)	389	835.83	826	10	0.85	*836	3.70	*836.0	3.76	837	5.09
gcut5d	10	(500, 500)	30	196.83	193	4	0.50	*197	0.57	*197.0	0.60	198	0.03
gcut6d	20	(500, 500)	85	342.67	342	1	0.09	*343	0.35	*343.0	0.38	344	0.21
gcut7d	30	(500, 500)	159	591.00	590	1	0.02	*591	0.81	*591.0	0.88	592	0.33
gcut8d	50	(500, 500)	456	690.00	680	11	1.32	*691	5.07	*691.0	5.13	692	3.60
gcut9d	10	(1000, 1000)	63	130.67	127	4	0.65	*131	0.80	*131.0	0.88	132	0.06
gcut10d	20	(1000, 1000)	70	293.00	292	1	0.11	293	0.32	293.0	0.34	293	0.09
gcut11d	30	(1000, 1000)	224	329.38	320	10	1.48	*330	2.58	*330.0	2.67	331	1.07
gcut12d	50	(1000, 1000)	337	671.50	668	4	0.57	672	2.72	672.0	2.81	672	3.35

As Tabelas 1 e 2 apresentam os resultados obtidos em 10 execuções com os algoritmos GCH_{2D} e VU_{2D}. As seis primeiras colunas correspondem ao nome da instância, número de itens distintos, dimensões do objeto, número de colunas geradas pela PD, limite inferior obtido pela GC, valor da solução aproximada obtida por arredondamento da solução fracionária. As quatro colunas seguintes se referem à melhor das 10 rodadas, a solução do problema residual, o tempo gasto com no residual (Tg), o valor da solução final do PCEBG obtida e o tempo total da rodada, incluindo a GC (T). As duas colunas seguintes são o valor médio da solução final obtida em cada

uma das rodadas. As colunas restantes, com valores retirados de [Cintra et al. 2008], mostram as melhores soluções e o tempo gasto pelos algoritmos daqueles autores. Como os algoritmos CG e CGp obtiveram resultados muito similares, as tabelas os mostram apena uma coluna. Destacam-se nessas tabelas os valores de soluções marcados com *, pois indicam as soluções melhoradas obtidas pelos algoritmos.

TABELA 3 – Resultados do Algoritmo GCH_{2D}.

Instância	m	Dimensões Objeto	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução RG_{2D}	Tg(s)	Solução GCH_{2D}	T(s)	Solução GCH_{2Dm}
ASX	20	(1400, 700)	41	53.600	48	6	0.000	*54	0.062	54.5
ASY	20	(1700, 850)	93	29.107	21	9	0.031	*30	9.843	30.6
ASZ	20	(2000, 1000)	95	20.497	10	11	0.031	*21	43.399	21.6
ALX	20	(1400, 700)	80	459.470	450	11	0.016	461	1.950	461.1
ALY	20	(1700, 850)	87	299.180	291	9	0.045	*300	25.585	300.0
ALZ	20	(2000, 1000)	86	214.320	204	11	0.031	*215	103.240	215.5
AVX	20	(1400, 700)	59	329.575	324	6	0.016	*330	0.062	330.3
AVY	20	(1700, 850)	72	200.764	193	9	0.016	202	0.889	202.0
AVZ	20	(2000, 1000)	79	141.572	134	8	0.032	*142	5.070	142.7
BSX	30	(1400, 700)	62	73.688	64	10	0.015	*74	1.216	74.9
BSY	30	(1700, 850)	146	45.546	32	15	0.047	47	85.363	47.0
BSZ	30	(2000, 1000)	155	32.543	18	16	0.062	34	339.752	34.0
BLX	30	(1400, 700)	45	1,244.000	1,238	6	0.015	*1,244	0.171	1,244.0
BLY	30	(1700, 850)	73	606.750	593	15	0.047	608	4.586	608.8
BLZ	30	(2000, 1000)	156	415.533	402	15	0.046	417	123.723	417.0
BVX	30	(1400, 700)	84	553.875	545	10	0.015	555	0.483	555.3
BVY	30	(1700, 850)	117	355.534	346	10	0.015	*356	6.130	357
BVZ	30	(2000, 1000)	153	242.808	229	15	0.047	244	104.551	244.0
CSX	40	(1400, 700)	169	87.573	76	13	0.031	89	13.496	90.0
CSY	40	(1700, 850)	216	57.082	37	22	0.094	59	200.803	59.7
CSZ	40	(2000, 1000)	226	40.474	27	15	0.046	42	510.13	42.0
CLX	40	(1400, 700)	150	1,106.320	1088	20	0.042	1108	6.046	1108.5
CLY	40	(1700, 850)	205	718.141	703	16	0.046	*719	59.404	719.9
CLZ	40	(2000, 1000)	202	511.030	489	24	0.141	513	207.839	513.0
CVX	40	(1400, 700)	168	383.760	367	18	0.031	385	9.734	385.0
CVY	40	(1700, 850)	194	255.634	241	16	0.031	257	86.018	257.4
CVZ	40	(2000, 1000)	225	182.528	164	20	0.093	184	397.176	184.0
DSX	50	(1400, 700)	248	103.670	81	24	0.094	105	62.294	106.1
DSY	50	(1700, 850)	280	69.485	50	21	0.078	71	395.694	71.0
DSZ	50	(2000, 1000)	292	49.796	32	19	0.125	51	1,299.195	51.0
DLX	50	(1400, 700)	234	1,268.330	1244	27	0.085	1271	44.516	1271.4
DLY	50	(1700, 850)	266	836.283	812	26	0.092	838	250.582	838.8
DLZ	50	(2000, 1000)	278	599.726	576	25	0.094	601	739.351	601.6
DVX	50	(1400, 700)	221	881.483	868	15	0.038	883	28.525	883.8
DVY	50	(1700, 850)	286	574.561	553	23	0.890	576	257.522	576.2
DVZ	50	(2000, 1000)	278	411.405	391	22	0.070	413	899.315	413.0

Das instâncias utilizadas no PCEBG, verifica-se nas tabelas 1 e 2 que os algoritmos GCH_{2D} e VU_{2D} encontraram 11 soluções ótimas, ou seja, com o número de padrões de corte produzidos igual ao limite inferior arredondado para cima. Os resultados do VU_{2D} quando comparados com os demais algoritmos, não somente evidenciam sua eficiência com 8 soluções médias melhores, mas também apresentam tempos computacionais relativamente superiores.

Para observar o comportamento das ampliações dos algoritmos GCH_{2D} que resultaram no algoritmo VU_{2D}, foram executados novos testes em outras 36 instâncias, também, da literatura e que apresentam maior grau de dificuldade. Apresentadas por [Imahori et al. 2005], essas novas instâncias são codificadas como uma tripla ordenada, de acordo com os seguintes parâmetros: quantidade de itens distintos, escala de demanda destes itens e tamanho do objeto.

Utilizando a mesma configuração dos testes anteriores, os resultados produzidos em 10 execuções do algoritmo GCH_{2D} estão representados na Tabela 3. A indicação do * em 11 soluções assinala que o Limite Inferior da instância foi alcançado. Observando a frequência destas soluções na coluna referente às soluções médias, é importante observar que apenas nas instâncias ALY e BLX os ótimos foram gerados em todas as rodadas. Já na Tabela 4, verifica-se que 21 soluções ótimas são obtidas com a execução de VU_{2D}. As soluções médias relacionadas a estes ótimos, também, demostram a superioridade desta versão quando comparada as mesmas instâncias na tabela anterior.

Na tabela 5, tem-se um resumo das atuações dos respectivos algoritmos, retiradas a partir dos dados apresentados nas tabelas 3 e 4, em 10 execuções nas 36 instâncias apresentadas por [Imahori et al. 2005]. Os métodos são indicados nas linhas desta tabela e suas colunas apresentam os números de soluções cujo resultado é igual ao Limite Inferior arredondado para cima, com valor igual ao Limite Inferior mais 1, de soluções médias igual ao melhor conhecido e a média dos desvios da melhor solução. Além dos quantitativos anteriormente discutidos, a coluna com a média dos desvios da melhor solução obtida ratifica as ponderações sobre as ampliações promovidas no GCH_{2D} e evidenciam o eficiente desempenho da proposta VU_{2D}.

TABELA 4 – Resultados do Algoritmo VU_{2D}.

Instância	m	Dimensões Objeto	Colunas Geradas	Limite Inferior	Solução Aprox.	Solução RG_{2D}	Tg(s)	Solução VU_{2D}	T(s)	Solução $VU_{2D}m$
ASX	20	(1400, 700)	214	53.600	49	5	1.081	*54	5.065	54.0
ASY	20	(1700, 850)	1336	29.107	21	9	10.169	*30	77.083	30.0
ASZ	20	(2000, 1000)	1638	20.497	10	11	6.131	*21	67.688	21.0
ALX	20	(1400, 700)	541	459.470	450	10	5.632	*460	19.094	460.0
ALY	20	(1700, 850)	801	299.180	291	9	6.818	*300	66.687	300.0
ALZ	20	(2000, 1000)	1243	214.320	204	11	6.599	*215	130.420	215.0
AVX	20	(1400, 700)	214	329.575	323	7	1.934	*330	3.556	330.0
AVY	20	(1700, 850)	584	200.764	193	9	5.023	202	14.227	202.0
AVZ	20	(2000, 1000)	1075	141.572	134	8	4.868	*142	18.222	142.0
BSX	30	(1400, 700)	553	73.688	67	7	2.527	*74	10.747	74.0
BSY	30	(1700, 850)	2633	45.546	32	14	8.876	*46	111.602	46.0
BSZ	30	(2000, 1000)	3462	32.543	19	14	10.499	*33	299.973	33.9
BLX	30	(1400, 700)	225	1,244.000	1241	3	0.343	*1244	3.540	1244.0
BLY	30	(1700, 850)	1277	606.750	597	10	4.056	*607	18.462	607.3
BLZ	30	(2000, 1000)	2173	415.533	403	13	8.580	*416	128.497	416.0
BVX	30	(1400, 700)	629	553.875	548	6	1.904	*554	10.328	554.8
BVY	30	(1700, 850)	1277	355.534	345	11	5.054	*356	26.394	356.0
BVZ	30	(2000, 1000)	2405	242.808	228	16	11.825	244	98.685	244.0
CSX	40	(1400, 700)	1774	87.573	73	16	7.129	89	30.061	89.0
CSY	40	(1700, 850)	3583	57.082	37	22	10.905	59	126.360	59.0
CSZ	40	(2000, 1000)	4705	40.474	26	15	11.325	*41	197.916	41.9
CLX	40	(1400, 700)	1242	1,106.320	1088	15	8.456	*1107	26.552	1107.7
CLY	40	(1700, 850)	3338	718.141	702	17	10.000	*719	106.189	719.0
CLZ	40	(2000, 1000)	3793	511.030	489	23	16.364	*512	179.992	512.9
CVX	40	(1400, 700)	1809	383.760	368	17	8.658	385	26.161	385.0
CVY	40	(1700, 850)	2898	255.634	242	15	11.310	257	74.287	257.0
CVZ	40	(2000, 1000)	4698	182.528	164	20	15.304	184	187.777	184.0
DSX	50	(1400, 700)	3540	103.670	81	24	12.106	105	70.178	105.1
DSY	50	(1700, 850)	4871	69.485	50	21	17.285	71	156.905	71.0
DSZ	50	(2000, 1000)	8206	49.796	32	19	18.268	51	331.844	51.0
DLX	50	(1400, 700)	3124	1,268.330	1244	26	14.961	1270	57.486	1270.0
DLY	50	(1700, 850)	4811	836.283	812	26	16.053	838	134.972	838.0
DLZ	50	(2000, 1000)	6568	599.726	576	25	17.581	601	251.425	601.0
DVX	50	(1400, 700)	2645	881.483	869	13	6.957	*882	37.885	882.9
DVY	50	(1700, 850)	6233	574.561	553	23	13.572	576	99.949	576.0
DVZ	50	(2000, 1000)	5634	411.405	391	22	16.115	413	148.793	413.0

TABELA 5 – Síntese das Melhores Soluções dos Algoritmos GCH_{2D} e VU_{2D}.

Método	Ótimos Anotados	Limite Inferior + 1	Solução Média igual ao Ótimo	Desvio Médio
GCH2D	11	25	2	1.103
VU2D	21	15	15	0.569

6. Conclusões e Perspectivas Futuras

A combinação das ampliações com os algoritmos GCH_{2D} resultou na indicação do algoritmo híbrido VU_{2D} para o PCEBG no caso sem rotação. Para observar o comportamento do algoritmo VU_{2D}, foram executados testes em 12 instâncias apresentadas por [Cintra et al. 2008] e em 36 instâncias mais difíceis propostas por [Imahori et al. 2005]. Os resultados computacionais obtidos até o momento, foram bastante positivos e apontaram uma superioridade da versão VU_{2D}. Nas 36 instâncias de [Imahori et al. 2005], conseguiu 21 ótimos provados e as outras soluções encontradas nunca ficaram mais do que 1 unidade além do limite inferior dado pela Geração de Colunas. Esses resultados indicam que o limite da Geração de Colunas para o PCEBG, mesmo quando os subproblemas são definidos como PCBGI, onde não se considera a demanda, é bastante forte na prática. Como trabalhos futuros, pretende-se identificar uma melhor calibragem dos parâmetros envolvidos no VU_{2D} e implementar o algoritmo VU_{2Dr} para o caso rotacionado do PCEBG.

Referências Bibliográficas

Beasley, J. E. (1985). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, p. 297-306.

Cintra, G. F., Miyazawa, F. K., Wakabayashi, Y., e Xavier, E. C. (2008). Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191(1), 61-85.

Dantzig, G.B. e Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, v.8, p.101-111.

Furini, F., Malaguti, E., Durán, R. M., Persiani, A., e Toth, P. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, *218*(1), 251-260.

Gilmore, P.C. e Gomory, R.E. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, v.9, p. 849-859.

Gilmore, P.C. e Gomory, R.E. (1965). Multistage cutting problems of two and more dimensions, *Operations Research*, v.13, p. 94-119.

Herz, J.C. (1972). A recursive computational procedure for two-dimensional stock-cutting. *IBM Journal of Research and Development*, pp. 462–469.

Imahori, S., Yagiura, M., Umetani, S., Adachi, S., e Ibaraki, T. (2005). Local search algorithms for the two-dimensional cutting stock problem with a given number of different patterns. *Metaheuristics: Progress as Real Problem Solvers*, p. 181-202.

Prais, M., e Ribeiro, C.C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, v. 12, n. 3, p. 164-176.

Valério de Carvalho, J.M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, v. 17, n. 2, p. 175-182.

Velasco, A.S., Paula Junior, G.G. e Vieira Neto, E. (2008), Um Algoritmo Heurístico Baseado na GRASP para o Problema de Corte Bidimensional Guilhotinado e Restrito. *Revista Gepros - Gestão da Produção, Operações e Sistemas*, ed.1, p.129-141.

Velasco, A. S. e Uchoa, E. (2014). *Geração de Padrões de Corte Bidimensionais Guilhotinados via Grasp*. In: XLVI SBPO - Simpósio Brasileiro de Pesquisa Operacional, Salvador.

Velasco, A. S. e Uchoa, E. (2015). *Geração de Colunas para o Problema de Corte de Estoque Bidimensional Guilhotinado*. In: XLVII SBPO - Simpósio Brasileiro de Pesquisa Operacional, Ipojuca.

Velasco, A. S. e Uchoa, E. (2016). *Algoritmos baseados em Grasp, Programação Dinâmica e Inteira para o Problema de Corte Bidimensional Guilhotinado Restrito*. In: XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional, Vitória.