ABORDAGENS DE PROGRAMAÇÃO INTEIRA E PROGRAMAÇÃO POR RESTRIÇÕES PARA PROBLEMAS DE EMPACOTAMENTO BIDIMENSIONAL

Oliviana Xavier do Nascimento e Thiago Alves de Queiroz
Unidade de Matemática e Tecnologia – UFG/Regional Catalão.
Av. Dr. Lamartine Pinto de Avelar, 1120, Setor Universitário, 75704-020, Catalão – GO, Brasil.
olivianaxn@gmail.com
taq@ufg.br

RESUMO

Este artigo propõe e compara abordagens para dois problemas de empacotamento de itens retangulares em um recipiente, sendo eles o problema de empacotamento ortogonal bidimensional (2OPP) e o problema da mochila bidimensional (2KP). A primeira abordagem refere-se a um modelo de programação inteira para resolver ambos os problemas. A segunda refere-se a um modelo de programação por restrições para o 2OPP, enquanto a terceira abordagem, para o 2KP, trata da combinação de um modelo de programação inteira, para o problema da mochila 0-1, com um modelo de programação por restrições apresentado na segunda abordagem. Experimentos computacionais mostram que a resolução das abordagens que envolvem programação por restrições é mais rápida do que lidar diretamente com modelos de programação linear inteira.

PALAVRAS-CHAVE. Problema de Empacotamento Ortogonal Bidimensional, Problema da Mochila Bidimensional, Programação Linear Inteira, Programação por Restrição.

Tópicos: OC – Otimização Combinatória, PM – Programação Matemática.

ABSTRACT

This article proposes and compares approaches for two packing problems of rectangular items in a rectangular container, which are the two-dimensional orthogonal packing problem (2OPP) and the two-dimensional knapsack problem (2KP). The first approach refers to an integer programming model to solve both problems. The second one consists of a constraint programming model for the 2OPP, while the third one, intended to the 2KP, is a combination of an integer programming model, for the 0-1 knapsack problem, with the constraint programming model of the second approach. Computational experiments show that the approaches based on constraint programming are faster than those that solve integer programming models.

KEYWORDS. Two-dimensional Orthogonal Packing Problem, Two-dimensional Knapsack Problem, Integer Linear Programming, Constraint Programming.

Paper topics: OC – Combinatorial Optimization, PM – Mathematical Programming.

1. Introdução

O Problema de Empacotamento Ortogonal Bidimensional, do inglês *Two-dimensional Orthogonal Packing Problem* (2OPP), bem como o Problema da Mochila Bidimensional, do inglês *Two-dimensional Knapsack Problem* (2KP), pertencem à classe de problemas de empacotamento, em que se busca empacotar itens sem sobreposição dentro de um recipiente de acordo com algum objetivo, como maximizar a utilização do recipiente. Ambos problemas consideram um conjunto de itens retangulares que precisam ser empacotados em um único recipiente retangular.

Nesse sentido, o 2KP busca por um subconjunto de itens de máximo valor que pode ser empacotado no recipiente, enquanto o 2OPP verifica se todos os itens do conjunto podem ser empacotados no recipiente. Uma solução viável para ambos os problemas consiste em um conjunto de coordenadas, cada qual associada ao ponto onde cada item deve ser empacotado dentro do recipiente, de forma que as restrições são atendidas. No que diz respeito a complexidade, o 2KP é considerado um problema NP-difícil e o 2OPP, por ser um problema de decisão, é NP-Completo [Garey e Johnson, 1979].

Problemas de corte e empacotamento, tais como o 2KP e o 2OPP, ocorrem na indústria, quando itens retangulares (de aço, madeira ou papel) precisam ser cortados de uma placa retangular grande [Clautiaux et al., 2008]. Ocorrem também no carregamento de contêineres ou veículos de carga, em que caixas precisam ser empacotadas apenas no chão do recipiente, desconsiderando, neste caso, a altura.

Na literatura, a resolução do 2KP, do 2OPP e suas versões aparecem em diversos trabalhos, tais como em: [Hadjiconstantinou e Christofides, 1993], com a proposta de um método exato baseado em uma relaxação Lagrangeana para resolver a versão não guilhotinada do 2KP; [Beldiceanu e Carlsson, 2001] e [Pisinger et al., 2003], que utilizaram técnicas de poda e decomposição, respectivamente, para resolver o 2OPP, sendo que no último trabalho os autores combinaram a técnica de programação por satisfação de restrições com a técnica de geração de colunas e, além disso, consideraram restrições adicionais para obtenção de um padrão de corte guilhotinado; [Caprara e Monaci, 2004], que apresentaram quatro métodos exatos para o 2KP, baseados na relaxação deste na versão unidimensional, sendo que a factibilidade do empacotamento é checada através de um algoritmo exato para o problema de Empacotamento em Faixa; [Fekete e Scheperes, 2004], que elencaram diferentes abordagens para verificar a viabilidade de empacotamentos no contexto do 2OPP; [Clautiaux et al., 2006], que desenvolveram dois algoritmos exatos para o 2OPP, sendo um deles, um algoritmo do tipo *branch-and-bound* e, o outro baseado em uma nova relaxação para o problema; [Fekete et al., 2007], com a proposta de um algoritmo exato, baseado em grafo de intervalos, para buscar por empacotamentos ótimos no contexto do 2KP; [Baldacci e Boschetti, 2007] que resolveram o 2KP na sua versão não guilhotinada por uma estratégia de duas etapas. A primeira etapa consiste na resolução de um problema do tipo mochila 0-1 e a segunda consiste em resolver um 2OPP para checar a factibilidade do empacotamento; [Clautiaux et al., 2008] que propuserem um algoritmo *branch-and-bound* aliado à técnica de programação por restrições para o 2OPP, cujos resultados foram aperfeiçoados por [Mesyagutov et al., 2012], sendo resolvido o caso não guilhotinado.

Neste artigo, a solução ótima do 2KP é obtida resolvendo o problema da mochila 0-1 em conjunto com o 2OPP, em um algoritmo que, no primeiro momento seleciona um subconjunto de itens de valor máximo tal que a soma da área desses itens não é maior do que a área do recipiente. Em seguida, utilizando um modelo de programação por restrições, verifica-se se os itens selecionados podem ser empacotados dentro do recipiente sem sobreposição. Caso não seja possível empacotar o subconjunto, um corte é inserido, e o processo continuar até achar o primeiro subconjunto que seja viável para o empacotamento.

Abordagens híbridas, como a descrita no parágrafo anterior para resolver o 2KP, podem ser encontradas em [Caprara e Monaci, 2004], [Baldacci e Boschetti, 2007] e [Fekete et al., 2007]. Nos dois primeiros trabalhos, primeiro obtém-se um conjunto de itens e depois checa-se a viabilidade de empacotar o conjunto selecionado. O terceiro trabalho resolve o 2KP por meio de

um algoritmo *branch-and-bound* que no primeiro nível da árvore de busca seleciona um subconjunto de itens. No segundo nível da árvore, um problema de empacotamento ortogonal é resolvido, em que se utiliza grafos de intervalos para verificar a viabilidade do empacotamento.

Além da abordagem híbrida para o 2KP, codifica-se um modelo de programação por restrições para o 2OPP e dois modelos de programação inteira, um para o 2KP e outro para 2OPP, respectivamente. Os modelos de programação inteira foram baseados no modelo de [Junqueira et al., 2012], para o qual dois novos conjuntos de restrições foram propostos para ajudar na detecção de sobreposição entre os itens. Todos os modelos são descritos na Seção 2.

Satisfazer a restrição de não sobreposição dos itens em abordagens exatas é considerado um dos grandes gargalos dos problemas de empacotamento. Os modelos de programação inteira apresentados para o 2KP e o 2OPP contam com restrições de não sobreposição relacionadas com a discretização do recipiente em uma malha inteira de pontos. Esta abordagem pode ser ineficiente na busca por uma solução ótima para instâncias com muitos itens diferentes e recipientes de dimensões grandes, uma vez que o modelo passa ter muitas variáveis de decisão.

Em contrapartida, uma abordagem que combina um modelo de programação inteira apenas para selecionar subconjuntos de itens que respeitam a área do recipiente, contribui por diminuir o número de itens na verificação da não sobreposição, a qual pode ser feita por um modelo de programação por restrições. Isso diminui o número de variáveis e, além disso, a técnica de programação por restrições é boa para testar a viabilidade de uma solução [Clautiaux et al., 2008].

Os experimentos computacionais, na Seção 3, mostram o comportamento de cada umas das abordagens durante a resolução de instâncias da literatura. Os resultados confirmam que a restrição de não sobreposição é um dos grandes desafios ainda a ser trabalhado no âmbito de problemas de empacotamento. As conclusões e direções para trabalhos futuros são dados na Seção 4.

2. Problemas de Empacotamento: 2KP e 2OPP

No Problema da Mochila Bidimensional, o 2KP, considera-se um conjunto $S_1 = \{1, 2, \ldots, m\}$ de itens retangulares, em que cada item i de S_1 possui comprimento l_i, largura w_i e valor v_i, e um recipiente (mochila) de comprimento L e largura W. Busca-se, com o 2KP, determinar um subconjunto de S_1 cuja soma dos valores dos itens seja máxima e que os itens selecionados não se sobrepõem uns aos outros e caibam totalmente dentro do recipiente. Claramente, a soma de suas áreas deve ser menor ou igual à área do recipiente.

Desse modo, um modelo de programação inteira para o 2KP pode ser obtido a partir da formulação (1)-(3), que é uma relaxação e se limita a resolver a versão unidimensional do problema. Percebe-se que este modelo não impede que haja sobreposição dos itens selecionados. A variável de decisão utilizada na formulação é z_i, que assume valor 1 caso o item i seja colocado no recipiente e valor 0, caso ocorra o contrário, para todo $i=1, 2, ..., m$.

$$\max \sum_{i=1}^{m} v_i z_i \tag{1}$$

Sujeito a:

$$\sum_{i=1}^{m} (l_i w_i) z_i \leq LW \tag{2}$$

$$z_i \in \{0,1\} \qquad i=1, 2, ..., m \tag{3}$$

No Problema de Empacotamento Ortogonal Bidimensional, o 2OPP, considera-se como instância um conjunto $S_2 = \{1, \ldots, m\}$ de itens retangulares, em que cada item i de S_2 possui comprimento l_i, largura w_i, e um recipiente de comprimento L e largura W. Ao contrário do 2KP,

em que o objetivo é selecionar um subconjunto de itens de valor máximo, o 2OPP busca por verificar se todos os itens de S_2 podem ser empacotados dentro recipiente sem sobreposição e respeitando as suas dimensões. Essa solução é dita factível e é um limitante para a solução ótima do 2KP. ou é até mesmo a solução ótima caso S_2 seja um subconjunto de S_1 de valor máximo.

2.1 Formulação Inteira para o 2KP

[Junqueira et al., 2012] apresentaram um modelo de programação linear inteira para o problema de carregamento de um único contêiner, que pode ser visto como um Problema da Mochila Tridimensional (3KP) quando cada item tem apenas uma cópia. Este modelo é baseado na formulação de [Beasley, 1985] para o problema de corte não guilhotinado bidimensional.

A partir do modelo apresentado por [Junqueira et al., 2012], propõe-se na formulação (6)-(11), um modelo de programação inteira para o 2KP. Para isso, considere $X=\{0, 1, 2, ..., L\}$ e $Y=\{0, 1, 2, ..., W\}$, tal que:

$$X_i = \{p \in X \mid 0 \leq p \leq L - l_i\} \qquad i = 1, \ldots, m \tag{4}$$

$$Y_i = \{q \in Y \mid 0 \leq q \leq W - w_i\} \qquad i = 1, \ldots, m \tag{5}$$

Considere também a seguinte variável de decisão binária x_{ipq}, que é igual a 1 se o item i é empacotado com seu canto inferior esquerdo na posição (p, q), tal que $i = 1, \ldots, m$, $p \in X_i$ e $q \in Y_i$; e igual a 0, caso contrário.

$$\max \sum_{i=1}^{m} \sum_{p \in X_i} \sum_{q \in Y_i} v_i x_{ipq} \tag{6}$$

Sujeito a:

$$\sum_{i=1}^{m} \sum_{\{p \in X_i \mid s - l_i + 1 \leq p \leq s\}} \sum_{\{q \in Y_i \mid t - w_i + 1 \leq q \leq t\}} x_{ipq} \leq 1 \qquad s \in X, t \in Y \tag{7}$$

$$\sum_{i=1}^{m} \sum_{\{p \in X_i \mid s - l_i + 1 \leq p \leq s\}} \sum_{q \in Y_i} w_i x_{ipq} \leq W \qquad s \in X \tag{8}$$

$$\sum_{i=1}^{m} \sum_{p \in X_i} \sum_{\{q \in Y_i \mid t - w_i + 1 \leq q \leq t\}} l_i x_{ipq} \leq L \qquad t \in Y \tag{9}$$

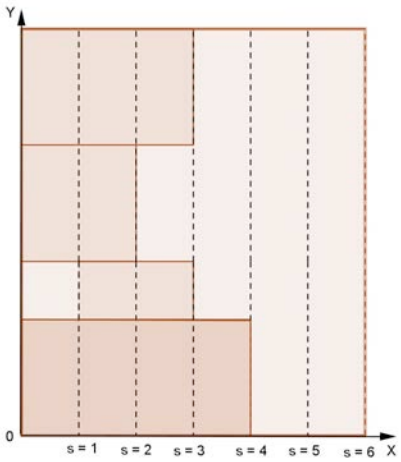
$$\sum_{p \in X_i} \sum_{q \in Y_i} x_{ipq} \leq 1 \qquad i = 1, \ldots, m \tag{10}$$

$$x_{ipq} \in \{0,1\} \qquad \begin{array}{l} i = 1, \ldots, m \\ p \in X_i, q \in Y_i \end{array} \tag{11}$$

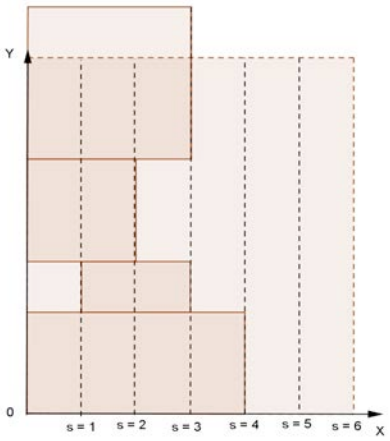
Na formulação (6)-(11), a função objetivo (6) visa maximizar o valor total de itens empacotados no recipiente. As restrições (7) impedem que ocorra sobreposição dos itens no recipiente, de forma que cada ponto de coordenadas *(s, t)* seja coberto por no máximo um item *i* empacotado em algum ponto *(p, q)*. Visando melhorar a formulação no que diz respeito a questão da não sobreposição, propõe-se as restrições em (8), as quais garantem que para toda coordenada $s \in X$, a soma da largura dos itens que interceptam *s* não ultrapasse a largura *W* do recipiente. De maneira análoga, propõe-se as restrições em (9), as quais garantem que para toda coordenada $t \in Y$, a soma dos comprimentos dos itens que interceptam *t* não ultrapasse o comprimento *L* do recipiente. As restrições em (10), por sua vez, garantem que no máximo 1 item de cada tipo *i* será empacotado no recipiente e as restrições (11) definem o domínio das variáveis de decisão.

A Figura 1 traz exemplos de empacotamento para as restrições em (8) e (9). As Figuras 1(a) e 1(b), respectivamente, mostram um empacotamento que satisfaz o conjunto de restrições em (8) e outro que não o satisfaz. Da mesma forma, as Figuras 1(c) e 1(d) trazem, respectivamente, um exemplo para as restrições em (9).

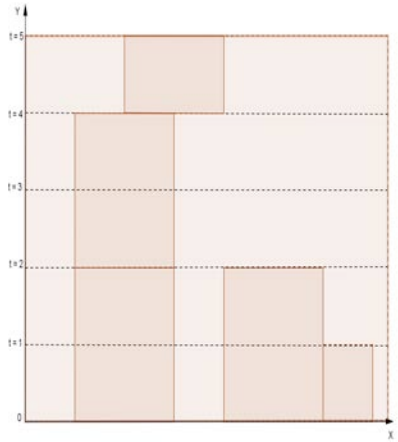
(a) Satisfaz as restrições em (8).



(b) Não satisfaz as restrições em (8).



(c) Satisfaz as restrições em (9).



(b) Não satisfaz as restrições em (9).

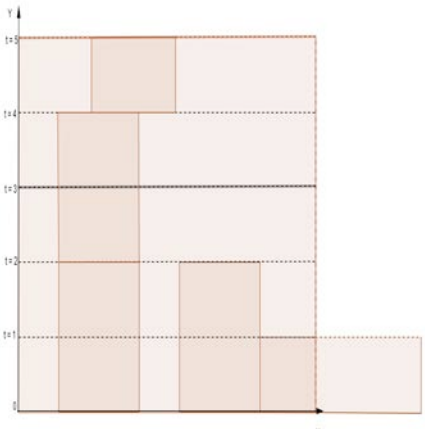


Figura 1 - Exemplo de empacotamentos para as restrições em (8) e (9).

2.2. Formulação Inteira para o 2OPP

Como mencionado anteriormente, uma solução para o 2OPP é considerada factível ou infactível, uma vez que não há objetivo a ser atendido. Ela é factível se todos os itens da instância puderem ser empacotados dentro do recipiente, conforme as restrições impostas. Logo, um

modelo para um 2OPP deve determinar se existe um conjunto de coordenadas associadas aos itens da instância considerada.

Em vista disso, as restrições em (10) podem ser rescritas trocando o sinal de desigualdade (menor ou igual) pelo sinal de igualdade, em que todo item *i* passa, a partir de então, a ser obrigatoriamente empacotado dentro do recipiente. Além disso, desconsiderando a função objetivo em (6), o modelo de programação linear inteira para o 2KP em (6)-(11) é adaptado no seguinte modelo para o 2OPP, a saber:

$$\sum_{i=1}^{m} \sum_{\{p \in X_i \mid s - l_i + 1 \le p \le s\}} \sum_{q \in Y_i \mid t - w_i + 1 \le q \le t} x_{ipq} \le 1 \qquad s \in X, t \in Y \qquad (12)$$

$$\sum_{i=1}^{m} \sum_{\{p \in X_i \mid s - l_i + 1 \le p \le s\}} \sum_{q \in Y_i} w_i x_{ipq} \le W \qquad s \in X \qquad (13)$$

$$\sum_{i=1}^{m} \sum_{p \in X_i} \sum_{\{q \in Y_i \mid t - w_i + 1 \le q \le t\}} l_i x_{ipq} \le L \qquad t \in Y \qquad (14)$$

$$\sum_{p \in X_i} \sum_{q \in Y_i} x_{ipq} = 1 \qquad i = 1, \ldots, m \qquad (15)$$

$$x_{ipq} \in \{0,1\} \qquad i = 1, \ldots, m \quad p \in X_i, q \in Y_i \qquad (16)$$

2.3. Formulação de Programação por Restrições para o 2OPP

A técnica de programação por restrições busca pela resolução de modelos que envolvem variáveis de decisão e restrições. As variáveis de decisão indicam as decisões a serem tomadas no problema, cada uma possuindo um domínio de possíveis valores. As restrições impõem limites na combinação entre as variáveis. O objetivo é obter uma atribuição para as variáveis, respeitando o domínio dado, de forma a satisfazer todas as restrições. A ideia é a mesma da programação inteira, com diferença que a programação por restrições não busca pela solução ótima, apenas uma que seja viável.

Sendo assim, o 2OPP também pode ser resolvido por meio da programação por restrições. Foi assim que [Clautiaux et al. 2008] propuseram um algoritmo *branch-and-bound* para o 2OPP combinado com um modelo de escalonamento para reduzir o domínio das variáveis.

Um modelo básico para o 2OPP considera dois tipos de variáveis de decisão, isto é, X_i e Y_i que representam um ponto $p = (X_i, Y_i)$ onde o item *i* pode ser empacotado, dentro do recipiente, com relação ao seu canto inferior esquerdo. Estas variáveis possuem o domínio igualmente especificado em (4) e (5), respectivamente. Uma vez sendo definidas para todos os itens, tem-se que a não sobreposição entre quaisquer pares de itens *i* e *j* é evitada por meio da restrição:

$$[X_i + l_i \le X_j] \text{ ou } [X_j + l_j \le X_i] \text{ ou } [Y_i + w_i \le Y_j] \text{ ou } [Y_j + w_j \le Y_i] \qquad (17)$$

Assim, o modelo de programação por restrições considerado tem apenas as variáveis X_i e Y_i, com o seu respectivo domínio, e as restrições em (17) para todos os pares de itens.

2.4. Programação Inteira e Programação por Restrições para o 2KP

A formulação (1)-(3) claramente não impede que itens se sobreponham no caso do 2KP, muito menos que o empacotamento tenha todos os itens respeitando as dimensões do

recipiente, ficando tal modelo carecido de mecanismos que impeçam isto de ocorrer ou que, ao menos, sejam capazes de detectar quando isto ocorre. Note que a formulação (6)-(11) para o 2KP possui restrições para impedir a sobreposição dos itens, mas que pode resultar em um número muito grande de combinações a serem testadas. Estas considerações também são válidas para a formulação (12)-(16) para o 2OPP.

Por outro lado, a formulação (1)-(3) é uma formulação efetivamente rápida para ser resolvida, pois considera apenas uma variável de decisão por item da instância de entrada. Ao mesmo tempo, o modelo de programação por restrições para o 2OPP verifica se há pelo menos uma configuração de empacotamento tal que os itens não se sobreponham no recipiente. Assim, ele desempenha algo análogo as restrições de não sobreposição dos modelos inteiros, porém de maneira mais eficiente por apenas observar factibilidade. A literatura também já tem afirmado que o uso da programação por restrições é indicado para problemas de decisão, mas não de otimização.

Buscando unir as potencialidades da formulação (1)-(3), que pode ser vista como uma formulação relaxada para o 2KP, e do modelo de programação por restrição do 2OPP, propõe-se a seguinte abordagem para o 2KP.

Seja U o subconjunto de itens de S_1 selecionados pela formulação (1)-(3), o qual é um subconjunto de valor máximo. O modelo de programação por restrições para o 2OPP é, então, chamado para verificar se todos os itens de U podem ser empacotados no recipiente. Se sim, tem-se, então, uma solução ótima. Caso os itens não possam ser empacotados, um corte é inserido na formulação para que, assim, outro subconjunto U_1 (diferente de U) seja obtido. Novamente, o modelo de programação por restrições é chamado para verificar a viabilidade de U_1. Estes passos são repetidos até que o modelo de programação por restrições encontre um subconjunto factível. O corte mencionado refere-se à inserção da desigualdade:

$$\sum_{i \in U} Z_i \leq |U| - 1 \tag{18}$$

A desigualdade (18) evita que seja obtido um subconjunto de itens igual ao U, uma vez que ele não atende a viabilidade do empacotamento. Isso faz com que formulação agora selecione um subconjunto diferente de máximo valor.

Desse modo, o número de variáveis envolvidas na verificação da viabilidade do empacotamento diminui, o que deve tornar mais rápida a procura por uma solução. Por exemplo, seja caso particular em que há 10 itens de dimensões diferentes para serem empacotados em um recipiente de comprimento e largura iguais a 500. Assim, o número de variáveis no modelo de programação inteira para o 2KP é de 2.500.000, enquanto a formulação (1)-(3) tem apenas 20.

3. Experimentos Computacionais

Todos os algoritmos foram codificados na linguagem C++ e os experimentos ocorreram em um computador com processador Intel Core i5-3570 de 3.40 GHz, 8 GB de memória RAM e sistema operacional Linux. Os modelos de programação inteira e de programação por restrições foram codificados utilizando as classes e as bibliotecas do IBM ILOG CPLEX Optimization Studio 12.6.1 [IBM, 2014].

As instâncias usadas para testar e comparar as abordagens para o 2KP foram obtidas da OR-Library [Beasley, 1990], tal que foi considerada apenas uma cópia para cada item. No caso do 2OPP, foram utilizadas as mesmas instâncias de [Clautiaux et al. 2008]. A resolução de cada instância foi limitada em 3600 segundos, sendo a melhor solução (ou a ótima, caso não atinja o tempo limite) encontrada dentro deste tempo limite reportada.

3.1. Resultados

Na Tabela 1 encontram-se os resultados para o 2OPP sobre as instâncias usadas por [Clautiaux et al. 2008]. A tabela traz se a solução encontrada é factível ou não e o tempo gasto para encontrá-la, para o modelo de programação inteira (12)-(16), de programação por restrições e o algoritmo *branch-and-bound* desenvolvido por [Clautiaux et al. 2008], respectivamente. As instâncias em que o tempo limite foi excedido foram marcadas com um asterisco (*).

Observando os resultados na Tabela 1, a programação por restrições foi superior ao modelo de programação inteira para 20 das 31 instâncias consideradas, apesar dele não ter conseguido resolver uma instância dentro do tempo limite imposto. Ao mesmo tempo, nenhuma das abordagens consideradas neste trabalho foram superiores à de [Clautiaux et al. 2008].

Tabela 1- Resultados para o conjunto de 31 instâncias do 2OPP.

	Formulação Inteira		**Programação por Restrições**		**Solução de [Clautiaux et al., 2008]**	
Instância	Solução	Tempo	Solução	Tempo	Solução	Tempo
E00N10	Infactível	0,52	Infactível	0,51	Infactível	< 0,00
E00N15	Infactível	20,76	Infactível	4,81	Infactível	0,06
E00N23	Infactível	170,90	-	3600*	Infactível	25,00
E02F17	Factível	5,83	Factível	81,75	Factível	0,02
E02F20	Factível	7,50	Factível	3,86	Factível	0,01
E02F22	Factível	3,14	Factível	0,28	Factível	0,01
E02N20	Infactível	42,61	Infactível	1,55	Infactível	0,01
E03N10	Infactível	0,11	Infactível	0,08	Infactível	< 0,00
E03N15	Infactível	74,64	Infactível	7,22	Infactível	1,78
E03N16	Infactível	132,3	Infactível	300,13	Infactível	3,70
E03N17	Infactível	38,68	Infactível	8,05	Infactível	0,11
E04F15	Factível	2,91	Factível	2,21	Factível	0,14
E04F17	Factível	3,65	Factível	0,21	Factível	0,03
E04F19	Factível	1,27	Factível	3,01	Factível	0,02
E04F20	Factível	2,24	Factível	0,66	Factível	0,37
E04N15	Infactível	21,43	Infactível	33,35	Infactível	2,66
E04N17	Infactível	29,09	Infactível	3,00	Infactível	< 0,00
E05F15	Factível	3,40	Factível	1,78	Factível	0,01
E05F18	Factível	5,60	Factível	69,32	Factível	0,02
E05F20	Factível	2,46	Factível	0,23	Factível	0,98
E05N15	Infactível	1,45	Infactível	163,83	Infactível	1,57
E07F15	Factível	2,25	Factível	0,35	Factível	0,17
E07N10	Infactível	0,70	Infactível	0,10	Infactível	< 0,00
E07N15	Infactível	0,26	Infactível	0,41	Infactível	0,01
E08F15	Factível	3,16	Factível	0,36	Factível	0,01
E10N10	Infactível	0,19	Infactível	0,07	Infactível	< 0,00
E10N15	Infactível	0,30	Infactível	117,4	Infactível	1,34
E13N10	Infactível	1,06	Infactível	1,14	Infactível	0,06
E13N15	Infactível	3,34	Infactível	10,56	Infactível	< 0,00
E15N15	Infactível	2,34	Infactível	1,26	Infactível	0,01
E20F15	Factível	0,85	Factível	0,01	Factível	0,01

A Tabela 2 possui os resultados sobre as instâncias para o 2KP. Para cada instância há o valor encontrado para função objetivo e o tempo gasto em segundos, ambos obtidos pelo modelo de programação inteira (12)-(16) e a abordagem que combina programação inteira aliada à programação por restrições (na Seção 2.4). Para a última abordagem, a tabela traz também o número de cortes, que representa quantas vezes a desigualdade (18) foi inserida durante a otimização e, além disso, traz a solução inicial dada pelo modelo da mochila sem nenhum corte.

Ainda na Tabela 2, os campos marcados com traço (-) significam que a abordagem não conseguiu encontrar uma solução para a referida instância antes do tempo limite de 3600 segundos ser atingido. Os campos marcados com um asterisco (*) significam que o valor da função objetivo vem de uma solução em que não há a garantia de que os itens selecionados podem ser empacotados de forma viável, pois o tempo limite de 3600 segundos foi atingido antes da programação por restrições retornar com uma resposta.

Os resultados na Tabela 2 mostram que para as instâncias mais simples, com poucos itens e dimensões pequenas do recipiente, como as do tipo *ngcut*, os dois métodos apresentaram bons resultados, embora a formulação inteira tenha consumido maior tempo computacional. Porém, para as demais instâncias, a formulação inteira não conseguiu fornecer uma solução em 3600 segundos, mostrando que a restrição de não sobreposição é o principal gargalo do empacotamento.

Por sua vez, a formulação inteira combinada com a programação por restrições para o 2KP conseguiu resolver algumas instâncias que o modelo de programação inteira não conseguiu, em particular a cgcut2, gcut1 e gctu5. Apesar destas últimas instâncias terem poucos itens, o recipiente tem dimensões grandes na gcut1 e gctu5, com 250x250 e 500x500, respectivamente. Além disso, nota-se que o modelo de programação por restrições ainda depende da quantidade de pontos considerados na malha para o recipiente (veja o domínio das variáveis). Por isso, ele acaba tendo dificuldades para verificar a viabilidade do empacotamento em instâncias maiores.

4. Conclusões e Trabalhos Futuros

Neste trabalho foram comparados modelos de programação inteira e programação por restrições para resolver o Problema da Mochila Bidimensional, o 2KP, e o Problema de Empacotamento Ortogonal Bidimensional, o 2OPP. Adaptou-se um modelo de programação inteira para o 2DK, que por sua vez foi adaptado para o 2OPP, enquanto se considerou um modelo básico de programação por restrições para o 2OPP, o qual foi combinado com uma modelo de programação inteira para o 2KP.

Os experimentos computacionais mostraram que a combinação de um modelo de programação inteira com um modelo de programação por restrições conduz a resultados melhores que os obtidos ao aplicar somente um modelo de programação inteira para o 2KP.

Da mesma forma, o modelo de programação por restrições para o 2OPP, mesmo sendo básico, teve um comportamento melhor comparado ao modelo de programação inteira. Pode-se observar que a programação por restrições é uma técnica eficiente para detectar a viabilidade do empacotamento, do que mostrar que a instância é infactível.

Para trabalhos futuros, pretende-se investigar estratégias que diminuam o número de variáveis nos modelos apresentados, tornando mais rápido o processo de busca por uma solução. Uma opção é considerar os pontos de discretização de [Herz, 1972].

Agradecimentos

Os autores agradecem o apoio financeiro recebido do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) e da Fundação de Amparo à Pesquisa do Estadod de Goiás (FAPEG).

Tabela 2 - Resultados para o conjunto de 38 instâncias do 2KP.

Instância	Formulação Inteira		Formulação Relaxada + Programação Por Restrição			
	Solução	Tempo	Cortes	Solução	Tempo	Solução inicial sem cortes
cgcut1	163	0,04	0	163	0,13	163
cgcut2	-	3600	0	2303	0,14	2303
cgcut3	-	3600	9	1720*	3600	1840
gcut1	-	3600	139	48368	5,79	62488
gcut2	-	3600	290	62193*	3600	62496
gcut3	-	3600	27	62496*	3600	62499
gcut4	-	3600	2	62498*	3600	62496
gcut5	-	3600	195	195582	407,13	249854
gcut6	-	3600	79	379765*	3600	249992
gcut7	-	3600	14	249978*	3600	249995
gcut8	-	3600	0	249994*	3600	249994
gcut9	-	3600	29	968757*	3600	997256
gcut10	-	3600	59	997038*	3600	999917
gcut11	-	3600	9	1000000*	3600	999925
gcut12	-	3600	2	999957*	3600	999996
gcut13	-	3600	0	8999120*	3600	8999120
m1	-	3600	27	15009*	3600	15589
m2	-	3600	0	74091*	3600	74091
m3	-	3600	0	150021*	3600	150021
m4	-	3600	28	269423*	3600	278294
m5	-	3600	12	594231*	3600	603452
mw1	-	3600	11	3469*	3600	3656
mw2	-	3600	0	18075*	3600	18075
mw3	-	3600	0	34032*	3600	34032
mw4	-	3600	4	53460*	3600	54693
mw5	-	3600	5	156868*	3600	161641
ngcut1	141	0,02	2	141	1,35	168
ngcut2	198	0,04	2	198	0,14	206
ngcut3	201	0,76	66	201	0,85	241
ngcut4	207	0,01	0	207	0,12	207
ngcut5	262	0,06	6	262	0,25	296
ngcut6	289	0,64	13	289	0,52	309
ngcut7	374	0,05	0	374	0,13	374
ngcut8	673	0,16	0	673	0,13	673
ngcut9	839	0,96	16	839	0,62	867
ngcut10	998	0,54	1	998	0,13	1063
ngcut11	1195	9,07	7	1195	0,45	1439
ngcut12	1615	302,35	63	1615	3,44	1799

Referências

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Baldacci, R.; Boschetti, M. A. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, v. 183, n.3, p. 1136 – 1149, 2007.

Beldiceanu, N. e Carlsson, M. (2001). Sweep as a generic pruning technique applied to the non-overlapping rectangles constraints. *Lecture Notes in Computer Science*, 2239:377-391.

Caprara, A. e Monaci, M. (2004). On the two-dimensional knapsack problem. *Operations Research Operations Research Letters*, 32:5–14.

Clautiaux, F., Carlier, J., e Moukrim, A. (2007). A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, 183:1196–1211.

Clautiaux, F., Jouglet, A., Carlier, J., e Moukrim, A. (2008). A new exact method for the two-dimensional orthogonal packing problem. *Computer & Operations Research*, 35:944–959.

Fekete, S. P. e Schepers, J. (2004). A combinatorial characterization higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29(2):353–368.

Fekete, S. P., Schepers, J., e Veen, J. C. V. D. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Computer & Operations Research*, 55(3):569–587.

Hadjiconstantinou, E. e Christofides, N. (1993). An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83:39–56.

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

IBM. *ILOG® CPLEX® Optimization Studio V12.6.1 documentation*. IBM Corporation, 2014.

Junqueira, L., Morabito, R., e Yamashita, D. S. (2012). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39:74-85.

Mesyagutov, M., Scheithauer, G., e G. Belov. (2012). Lp bounds in various constraint programming approaches for orthogonal packing. *Computers & Operations Research*, 39(10):2425–2438.

Psinger, D. e Sigurd, M. (2003). *On using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem*. Department of Computer Science, University of Copenhagen.