V Simpósio de Iniciação Científica 2022
Ciência, Tecnologia e Inovação no Brasil

Formulação do Problema do Empacotamento Bidimensional Com Conflitos

Ana Clara N. dos Santos[1] (IC), Prof. Dr. Pedro H. D. B. Hokama (PQ)[1] , Prof. Dr. Mário C. S. Felice (PQ)[2]
[1]*Universidade Federal de Itajubá,* [2]*Universidade Federal de São Carlos*

Palavras-chave: Otimização. Problema do Empacotamento. Programação Linear Inteira.

Introdução

Os problemas de otimização são definidos como problemas computacionais com o objetivo de encontrar a melhor solução de todas as possíveis e estão presentes em diversas situações do cotidiano. Em empresas dos mais variados ramos existem processos logísticos que visam a organização dos produtos de forma compacta, para diminuir custos e melhorar a forma de utilizar o espaço em seu transporte e armazenamento. Esse procedimento é característico dos Problemas de Corte e Empacotamento, exemplos podem ser vistos em situações nas quais pedaços de aço, madeira ou papel precisam ser cortados de retângulos maiores, ou mesmo no carregamento de veículos e contêineres.

O Problema do Bin Packing, ou Empacotamento, consiste em minimizar o número de retângulos semelhantes chamados de recipientes ou bins usados para empacotar um conjunto de retângulos menores, sujeito a algumas restrições acerca dos itens: sua orientação é fixa, não sendo permitidas rotações e eles não devem se sobrepor ou exceder a borda do recipiente. Nesta pesquisa trabalhos com a variação do problema que apresenta conflitos entre os itens, então a incompatibilidade entre eles deve ser respeitada, itens conflitantes não devem ser empacotados no mesmo recipiente.

O problema de otimização supracitado é *NP-difícil*, ou seja, ele não pode ser solucionado em tempo polinomial a menos que *P=NP*, logo possui elevado grau de complexidade. Abordamos o Bin Packing Bidimensional com Conflitos (2BPPC) utilizando Programação Linear para buscar o melhor empacotamento possível. Para tanto, são empregues adaptações de algoritmos, formulações e limitantes da literatura. Ademais, são propostas abordagens e técnicas que buscam otimizar a solução do problema.

O objetivo deste trabalho é implementar e analisar o impacto de abordagens exatas para o 2BPPC. Por se tratar de um problema *NP-difícil*, a maior parte dos trabalhos da literatura optam por utilizar heurísticas ou métodos aproximados de resolução para problemas com a mesma classificação. O propósito é explorar técnicas que forneçam as soluções exatas, em tempo admissível para instâncias de tamanho considerável.

Metodologia

Inicialmente, foi desenvolvido um estudo sobre a Programação Linear Inteira (PLI); modelo matemático que descreve problemas de otimização combinatória, nos quais tanto as restrições quanto o critério de otimização são funções lineares (Dasgupta et al., 2008). Após isso, a formulação proposta por Gendreau et al.(2004) para o Bin Packing com Conflitos foi implementada e é apresentada a seguir:

Dado um conjunto finito V de n itens cada um com altura h_i e largura w_i, próprias e um conjunto Q de bins idênticos, com altura H e largura W e um grafo G=(V, ε) de conflitos entre os itens. Deve-se considerar as variáveis binárias y_k que indica se o bin k está sendo utilizado (y_k = 1) ou não (y_k = 0), e $x_{i,k}$ que indica se o item i está presente no bin k, assumindo valor 1 se está e 0 caso contrário. Considere $\boldsymbol{\mathcal{S}}$ como o conjunto de todas as soluções de empacotamento possíveis e S como o conjunto de todos os empacotamentos, inclusive os infactíveis.

Função Objetivo: Minimizar $\sum_{k=1}^{n} y_k$ (1)

Sujeito a:

$$\sum_{i=1}^{n} w_i h_i x_{i,k} \leq WHy_k \qquad k \in \{1,\dots n\} \qquad (2)$$

$$\sum_{k=1}^{n} x_{i,k} = 1 \qquad i \in \{1,\dots n\} \qquad (3)$$

$$x_{i,k} + x_{j,k} \leq 1 \qquad (i,j) \in \varepsilon,\ k \in \{1,\dots n\} \qquad (4)$$

$$\sum_{i \in S} x_{i,k} \leq |S| - 1 \qquad S \notin \boldsymbol{\mathcal{S}}, k \in \{1,\dots n\} \qquad (5)$$

$y_k \in \{0,1\}$ $k \in \{1,\dots n\}$ (6)

$x_{i,k} \in \{0,1\}$ $i \in \{1,\dots n\}, k \in \{1,\dots n\}$ (7)

O objetivo definido por (1) consiste em minimizar o número de recipientes utilizados. A restrição (2) garante que a capacidade dos bins seja respeitada, enquanto (3) certifica-se de que cada item será alocado em apenas um recipiente e (4) assegura que dois itens que possuem conflito (aresta entre eles) não podem ser alocados no mesmo bin. A restrição (5) garante que os itens sejam empacotáveis, isto é, que os conjuntos de itens que excedem os limites do recipiente quando empacotados juntos não serão alocados no mesmo bin. As restrições (6) e (7) apontam o domínio das variáveis binárias.
Após isso foram desenvolvidas uma série de técnicas, abordagens e heurísticas para otimizar o tempo de execução do programa. As principais serão descritas a seguir:

Fixando Item (F): predetermina que o primeiro item deverá ser alocado ao recipiente 1.

$$x_{1,1} = 1$$

Bins Consecutivos (B): Condiciona o uso do bin n, ao bin anterior n − 1 já ter sido utilizado.

$$y_{k-1} \geq y_k \qquad k \in \{2,\dots n\}$$

Restringindo Bins dos Itens (I): Restringe os bin aos quais os itens podem ser alocados. Serão considerados apenas metade dos itens, para evitar que restrições fracas sejam adicionadas.

$$\sum_{k=1}^{i} x_{i,k} = 1 \qquad i \in \{1,\dots, n/2\}$$

Restringindo Itens Largos ou Altos (R2): Decompõe os itens em dois conjuntos; altos e largos, em função da metade da altura e da largura do Bin, tal que $A_{altos} = \{ i \in V : h_i > H/2\}$ e $A_{largos} = \{ i \in V : w_i > W/2\}$. Em seguida, percorre todos os bins, de forma que a altura dos itens classificados como largos é acumulada e deve corresponder a no máximo a altura do bin (8), e a largura dos itens altos também é somada e restrita a no máximo a largura do recipiente (9).

$$\sum_{k=1}^{n} x_{i,k} h_i \leq H \qquad i \in A_{largos} \qquad (8)$$

$$\sum_{k=1}^{n} x_{i,k} w_i \leq W \qquad i \in A_{altos} \qquad (9)$$

Restringindo Itens (R3): Os itens são classificado em altos e largos, em função de um terço da altura e da largura do Bin, tal que $A_{altos} = \{ i \in V : h_i > H/3\}$ e $A_{largos} = \{ i \in V : w_i > W/3\}$. Considerando o recipiente discretizado em 3 faixas de altura e 3 de largura, são contabilizadas quantas faixas por inteiro cada elemento ocupa através dos pesos (10) e (11), após isso são novamente acumuladas as alturas dos elementos (12) e largura (13). Ambas são limitadas a serem menores ou iguais a $2H$ e $2W$, respectivamente.

$$\text{pesoW}(i) = \left\lfloor \frac{w_i}{W/3 + \epsilon} \right\rfloor \qquad i \in A_{largos} \qquad (10)$$

$$\text{pesoH}(i) = \left\lfloor \frac{h_i}{H/3 + \epsilon} \right\rfloor \qquad i \in A_{altos} \qquad (11)$$

$$\sum_{k=1}^{n} \text{pesoW}(i) x_{i,k} h_i \leq 2H \qquad i \in A_{largos} \qquad (12)$$

$$\sum_{k=1}^{n} \text{pesoH}(i) x_{i,k} w_i \leq 2W \qquad i \in A_{altos} \qquad (13)$$

Restringindo itens R4 e Rk seguem a mesma lógica, entretanto R4 considera a discretização das dimensões do recipiente em 4 faixas e Rk testa todas as possíveis divisões com $k \in \{2,\dots, n\}$ e possui duas variações: a que utiliza o k que resulta na maior soma e a que utiliza os 4 k's que resultam nas 4 maiores.
Adicionando Lower Bound (L1): Utilizamos o limitante inferior para o número de bins analisado por Carlier et al. (2007), no qual o conjunto dos itens V é decomposto em conjuntos de grandes, médios, altos, largos e pequenos em função dos inteiros p e q, tal que $1 \leq p \leq H/2$ e $1 \leq q \leq W/2$. Após a classificação dos itens, é feito o cálculo de quantos quadrados do bin discretizado ele ocupa (m). Para cada combinação de p e q, o m de todos os itens, exceto os grandes, são somados e usados para obter o número de bins necessários para os itens menores. Por fim, esse valor é somado ao número de itens grandes e médios. Ao final, o maior Lower Bound é adicionado ao modelo.
Adicionando Upper Bound (L2): para definir um limite superior para nosso modelo, utilizamos uma abordagem na qual todos os itens primeiramente eram alocados em prateleiras, as quais possuem a altura do primeiro item ali colocado e herdam o conflito de cada um dos que ali estejam. Sempre que possível o item é colocado em uma prateleira já existente, quando ocorrem conflitos ou as dimensões disponíveis na

prateleira são excedidas, uma nova é alocada. Após todos os itens serem destinados a alguma prateleira, estas são empacotadas nos bins, seguindo as mesmas restrições de tamanho e compatibilidade. Ao final, tem-se o número máximo de bins necessários para resolver o problema do empacotamento bidimensional com conflitos para aquela instância.

Ordenando Decrescente (O): O número de conflitos de cada um dos itens é contabilizado e baseado nessas quantidades eles são ordenados de forma decrescente, modificando o vetor recebido pela instância. Desse modo, os itens com mais conflitos serão alocados primeiro, o que evita desperdício de espaço em bins, que poderia ocorrer caso ele fosse alocado depois de diversos itens e não pudesse ser empacotado nos recipientes até então utilizados, devido a presença de um ou mais itens incompatíveis com ele.

Além dessas melhorias descritas, implementamos **Adicionando Conflitos Por Incompatibilidade De Dimensões (D)** que percorre os itens, comparando cada um deles com todos os itens que o sucedem, se a soma das larguras e alturas for maior que a dimensão do bin, não podem ser empacotados juntos, portanto adiciona um conflito ao modelo, que indica que aqueles itens são incompatíveis e **Adicionando Conflitos por Cliques (C)** que utilizada a biblioteca Cliquer (Niskanen, 2002) adaptada para C++. Uma clique em um grafo não-direcionado corresponde a um subconjunto de seus vértices, tal que cada dois vértices são adjacentes. No problema abordado, uma clique representa itens que não podem ser empacotados no mesmo bin, já que possuem conflitos com todos os outros que também compõe a clique. As cliques maximais são encontradas através de funções dessa biblioteca e por fim, é adicionada a restrição de que itens da mesma clique não podem ser empacotados no mesmo recipiente.

A formulação e todas as melhorias descritas foram implementados em C++, utilizando o solver CEPLEX Optimizer, entretanto o número de conjuntos não empacotáveis é exponencial, o que faz com que não seja possível inserir todas as restrições, portanto é necessário que sejam relaxadas. Detectar se a restrição é ou não violada configura um problema *NP-difícil*, devido a isso foi utilizado o modelo de programação por restrição feito por Charbel, outro aluno do grupo de pesquisa, que utilizou o CP Optimizer.

Resultados e discussão

Nesta seção as 12 melhorias propostas serão divididas em 6 grupos correlacionados para facilitar a execução e análise dos testes:

- B: F + B + I;
- R: R2 + R3 + Rk;
- O;
- L: L1 + L2;
- C;
- D.

Instâncias	Sem melhorias	B	B + R	BR + O	BRO + L	BROL + C	BROLC + D
2kdc04	599.907	104.347	279.889	155.025	8.66946	5.17564	0.023324
2kdc15	599.754	1.65436	0.23698	0.198044	0.435205	0.100885	0.092971
2kdc22	599.756	599.76	11.0794	8.76045	1.40648	2.21111	4.27224
2kdc25	32.4865	17.8452	8.55595	3.66576	1.85757	61.2479	0.894743
2kdc37	599.764	599.778	0.094082	0.159267	0.306882	0.165651	0.035975
2kdc42	599.755	396.491	0.664451	0.299657	0.564721	0.209384	0.073309
2kdc51	599.754	599.745	0.125806	0.15208	0.122862	0.084419	0.033927
2kdc64	0.69135	0.051874	0.052231	0.05268	0.031958	0.019548	0.019908

Tabela 1 – Comparação entre tempo total (s)

De um total de 64 instâncias, 8 foram escolhidas aleatoriamente para serem executadas a cada grupo de melhorias desenvolvidas. Primeiramente foram registrados os resultados da execução do PLI sem nenhuma melhoria, na qual a média de tempo total foi 453.9835s. A partir de então, foram incrementados os grupos de melhorias, iniciando-se pelo grupo B, que resultou numa diminuição do tempo médio para 289.9590s, devido a redução de empacotamentos possíveis ao se fixar o primeiro item, determinar o uso dos bins de forma consecutiva e de restringir os bins aos quais a metade dos itens poderia ser alocada. Em seguida, o grupo R foi adicionado ao programa e obteve uma redução extremamente significativa, com a média de 37.5872s por instância. Entretanto provocou um aumento de 175.5s na instância 2kdc04, o que pode ocorrer dependendo do perfil de cada uma das entradas, mas isso foi compensado pela diminuição relevante do tempo de execução de outras 6 instâncias, inclusive as instâncias 2kdc22, 2kdc37 e 2kdc51 que estavam acima de 599s e com a adição do grupo B obtiveram tempos inferiores a 12s. Após isso, com a melhoria O inserida a média caiu para 21.0391s, e a primeira instância que havia aumentado seu tempo no incremento anterior, voltou a apresentar melhora. Logo depois, o grupo L, que adiciona os limitantes superior e inferior foi ativado, gerando uma média de 1.6744s e a instância 2kdc04 apresentou uma queda superior a 145s, resultado da delimitação do possível número de bins, diminuindo o número de empacotamentos a serem testados. A penúltima melhoria acrescentada foi Adicionando Conflitos por Clique (C), que provocou um aumento na média para 8.6518s, em virtude da instância 2kdc25 que levou cerca de 59.39s a mais para ser executada, devido ao número de restrições redundantes que inseriu no modelo para essa entrada. Por último, a melhoria que adiciona conflitos por incompatibilidade de dimensões(D) foi inserida no programa, a média voltou

a diminuir, indo para 0.6808s, consequência dos novos conflitos, que evitaram que empacotamentos que não eram possíveis devido ao tamanho dos itens fossem testados.
Quando analisadas as 64 instâncias os impactos finais de todas as técnicas, heurísticas, pré-processamentos e cortes desenvolvidos durante a pesquisa, a média de tempo diminuído é de 414.2s e 96.87% das instâncias obtiveram redução no seu tempo de execução, apenas 2 instâncias pioraram em relação ao seu desempenho inicial.

Conclusões

Neste trabalho foram implementadas a formulação em PLI para o Bin Packing Bidimensional com Conflitos e 12 melhorias. A partir da análise dos resultados obtidos, é possível afirmar que os métodos utilizados se mostraram eficientes em relação ao desempenho do programa. Logo os objetivos propostos foram atingidos.
Com os estudos realizados foi possível compreender melhor os conceitos relacionados à Programação Linear e aos problemas NP-difíceis, possibilitou também um maior aprendizado sobre a utilização de ferramentas para resolver problemas de otimização combinatória e técnicas de projeto de algoritmos.
Dessa forma, considero que a pesquisa teve grande impacto na minha formação como cientista da computação. Permitindo um primeiro contato com a pesquisa científica e colaborando para o estudo dessa área tão vasta do conhecimento.

Agradecimento

Agradecimentos ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pela bolsa de Iniciação Científica através do programa PIBIC 134055/2021-7, aos professores Pedro Henrique Del Bianco Hokama e Mário César San Felice e à Universidade Federal de Itajubá.

Referências

Carlier, J., Clautiaux, F., e Moukrim, A. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. Computers & Operations Research, v. 34, n. 8, p. 2223–2250, 2007.

Chen, Y., Wang, F., Ma, Y., e Yao, Y. A distributed framework for solving and benchmarking security constrained unit commitment with warm start. IEEE Transactions on Power Systems, v.35, n. 1, p. 711–720, 2019.

Dasgupta, S., Papadimitriou, C. H., e Vazirani, U. V. Algorithms. Algorithms: McGraw-Hill Higher Education New York, 2008.

IBM. What are user cuts and lazy constraints?, 2021. Disponível em: https://www.ibm.com/docs/en/icos/20.1.0?topic=pools-what-are-user-cuts-lazy-constraints. Acesso em: 09/09/2022.

Niskanen, S. Cliquer - routines for clique searching, 2002. Disponível em: https://users. aalto.fi/˜pat/cliquer.html. Acesso em: 09/09/2022.

Pearce, R. H. Towards a general formulation of lazy constraints. . Tese de Doutorado. The University of Queensland, Brisbane, 2019.