

Heurísticas matemáticas aplicadas ao Problema de Carregamento de Contêineres

Kelly Márcia de Oliveira

Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Ouro Preto
Campus Universitário Morro do Cruzeiro, s/n, Ouro Preto/MG - 35400-000
kelly.marcia@aluno.ufop.edu.br

Túlio Ângelo Machado Toffolo

Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário Morro do Cruzeiro, s/n, Ouro Preto/MG - 35400-000
tulio@toffolo.com.br

RESUMO

Este trabalho tem seu foco no Problema de Carregamento de Contêineres (*Container Loading Problem* – CLP), que tem como objetivo alocar um conjunto de caixas a contêineres minimizando o volume total dos contêineres utilizados. Ao realizar o carregamento de caixas, deve-se observar algumas restrições, a saber: todas as caixas devem ser alocadas; caixas não podem se sobrepor dentro de um contêiner; e caixas devem ser alocadas inteiramente dentro da área do contêiner. Este trabalho propõe uma heurística matemática para o CLP, baseada na estratégia *Relax-and-Fix*. A estratégia utiliza uma solução inicial obtida por meio de um método construtivo e, em seguida, realiza uma busca local utilizando modelos de programação inteira mista. Em testes realizados com instâncias extensivamente utilizadas na literatura, resultados indicam que a abordagem proposta é capaz de gerar soluções de qualidade, tendo inclusive aprimorado o melhor resultado conhecido até então para uma das instâncias.

PALAVRAS CHAVE. Problema de Carregamento de Contêineres, Otimização, Modelos de Programação Inteira Mista, Heurísticas Matemáticas.

ABSTRACT

This paper addresses the Container Loading Problem (CLP), whose objective is to place a set of rectangular boxes in containers with the aim of minimizing the total volume of containers used. When carrying out the loading of boxes, some constraints must be observed, namely: all boxes must be allocated; boxes cannot overlap within a container; and boxes should be allocated entirely within the container area. This work proposes a matheuristic for the CLP, based on Relax-and-Fix. The strategy employs a greedy heuristic to produce an initial solution and then perform a local search using mixed-integer programming models. Results indicate that the proposed approach is capable of generating high-quality solutions in tests carried out with extensively studied instances, having improved the best result known so far for one of the instances.

KEYWORDS. Container Loading Problem, Optimization, Mixed Integer Programming Models, Matheuristics.

1. Introdução

Pelo volume de mercadorias enviadas mundialmente, pode-se inferir que há uma demanda das empresas por estratégias que possibilitem o envio destas mercadorias da forma mais eficiente possível, uma vez que reduzir o número de contêineres utilizados pode resultar em expressiva economia para as empresas. Em virtude do transporte de mercadorias demandar muita energia e liberar uma grande quantidade de substâncias tóxicas no meio ambiente, há também uma preocupação por parte da sociedade devido ao impacto ambiental causado pelo envio destes itens.

Problemas de Corte e Empacotamento envolvem vários problemas de otimização combinatória, nos quais procura-se encontrar a melhor forma de alocar pequenos objetos em objetos maiores de modo que o desperdício na utilização dos objetos maiores seja minimizado. Dentro da tipologia proposta por Dyckhoff [1990], o Problema de Carregamento de Contêineres (*Container Loading Problem – CLP*) é um caso particular dos Problemas de Corte e Empacotamento no qual os pequenos objetos são considerados itens tridimensionais de forma retangular, e os objetos maiores são os contêineres.

Wäscher et al. [2007] propuseram duas classificações para o Problema de Carregamento de Contêineres: minimização de entrada e maximização de saída. No primeiro caso, assume-se que há contêineres suficientes para acomodar todos os itens, assim, deseja-se minimizar a quantidade de contêineres utilizados. Já para os problemas de maximização de saída, os contêineres não são suficientes para acomodar todos os itens, e assim, deve-se maximizar a quantidade de itens alocados. Seguindo a classificação proposta por Wäscher et al. [2007], este trabalho aborda problemas de minimização de entrada.

Embora tenha ampla aplicação na indústria e outros setores, um estudo de Bortfeldt e Wäscher [2013], que analisou diferentes classes do problema de carregamento de contêineres, permite concluir que o CLP ainda é pouco abordado quando comparado com outros problemas de natureza combinatória. Problemas de empacotamento multi-dimensionais estão dentro da classe NP-difícil e têm se mostrado complicados de resolver na prática. Fatores como o grande número de vizinhos a serem explorados dificultam a busca por um melhor resultado em tempo computacional viável. Considerando o interesse das empresas e da sociedade por soluções de qualidade em tempo satisfatório, este trabalho visa contribuir com boas soluções por meio do desenvolvimento de um algoritmo para o CLP.

Este artigo está organizado como segue. O CLP é detalhado na Seção 2, na qual também são apresentados os trabalhos relacionados ao problema. Os métodos aplicados ao CLP são descritos na Seção 3. Na Seção 4 são expostos os resultados dos experimentos computacionais. Por fim, a Seção 5 discorre sobre as conclusões.

2. Problema de Carregamento de Contêineres

Dentro da tipologia apresentada por Wäscher et al. [2007], este trabalho aborda problemas de minimização de entrada, mais precisamente Problemas de Carregamento de Contêineres de Tamanhos Diversos (*Multi Stock Size Cutting Stock Problems – MSSCSP*), que neste trabalho é referido simplesmente como Problema de Carregamento de Contêineres (CLP). Neste problema, assume-se que há contêineres suficientes para acomodar todos os itens. Assim, deseja-se minimizar a quantidade de contêineres utilizados e alocar os itens em contêineres de forma que: (i) todos os itens sejam alocados a contêineres; (ii) itens não se sobreponham dentro dos contêineres; e (iii) itens sejam alocados inteiramente dentro da área dos contêineres.

Existem várias abordagens presentes na literatura para encontrar uma solução do CLP. Algumas destas abordagens, como as propostas por Ivancic [1988] e Bischoff e Ratcliff [1995], são

denominadas *sequenciais*. Neste tipo de abordagem, os contêineres são preenchidos sequencialmente. A cada iteração, o CLP é decomposto no Problema de Carregamento de Contêiner Único (*Single Container Loading Problem* – SCLP). Ou seja, dado um único contêiner, são geradas várias permutações de caixas de modo a preencher da melhor forma possível o espaço no interior do contêiner, maximizando o volume de itens enviados. Quando não for possível inserir itens em um contêiner, outro contêiner é utilizado até que todos os itens sejam alocados.

Bischoff e Ratcliff [1995] adaptaram o método proposto por Bischoff et al. [1995], originalmente projetado para o Problema de Carregamento de Paletes do Distribuidor (*Distributor's Pallet Loading Problem* – DPLP). O algoritmo realiza a alocação de itens de baixo para cima no contêiner. Dadas todas as áreas onde é possível alocar um item, sempre a de menor altura é escolhida primeiro. Para escolher a posição em que os itens serão alocados, o algoritmo gera todas as possíveis combinações de blocos de até duas caixas. O bloco que melhor ocupar a área selecionada é alocado. O processo de seleção da área, formação blocos e alocação de itens é repetido até que todos os itens sejam alocados.

Eley [2003], Che et al. [2011] e Zhu et al. [2012] empregaram formulações inteiras para o problema combinadas com heurísticas para maximizar o volume alocado dentro dos contêineres. Em todos os trabalhos são gerados padrões de empacotamento para o SCLP. Cada trabalho faz uso de diferentes critérios para construir esses padrões, que são combinados para gerar uma solução para o CLP.

Takahara [2006] propôs um algoritmo de duas fases para o problema. Na primeira fase, as soluções são geradas usando uma estratégia gulosa baseada no método *First-Fit*. Os itens são separados por tipos que são armazenados em uma lista de tipos de itens I . Também é gerada uma lista O de seis possíveis orientações para cada item. Em seguida, os itens são carregados em contêineres percorrendo-se a lista I e escolhendo-se a primeira orientação da lista O . Dessa forma, todos os itens de mesmo tipo são alocados em sequência. Se um item não couber na área disponível, escolhe-se a próxima orientação da lista O . Se ainda assim o item não couber no contêiner, o próximo contêiner disponível é escolhido. Na segunda fase, a solução encontrada pela estratégia gulosa é refinada por meta-heurísticas. As soluções são submetidas ao métodos de Busca Local e Têmpera Simulada e os resultados obtidos por cada método são comparados. A vizinhança para os dois métodos presentes na segunda etapa consiste em fazer troca na posição de dois elementos na lista I e na lista O . Ao final, retorna-se a melhor solução encontrada. Na fase de refinamento, a Têmpera Simulada obteve os melhores resultados. No final da execução, o algoritmo foi capaz de melhorar em uma unidade duas das melhores soluções conhecidas até então.

Mais tarde, Takahara [2008] propôs um algoritmo de busca local *multi-start* para o CLP baseado em sua proposição anterior [Takahara, 2006]. O algoritmo é constituído de duas etapas. A primeira etapa é semelhante à primeira descrita por Takahara [2006], usando o método *First-Fit* para gerar uma solução inicial. Porém, ao invés de ser gerada somente uma solução inicial após a ordenação da lista I de itens e O de orientações, são geradas várias soluções iniciais a partir de diferentes ordens de entrada dos itens e orientações, sendo que o número máximo de soluções iniciais geradas é definido por parâmetro. A segunda etapa é a de busca local. A cada solução inicial gerada, são realizadas trocas entre dois elementos na lista I e na lista O , para em seguida avaliar a solução gerada por estas mudanças até que um número máximo de iterações seja atingido. Embora o método proposto não tenha melhorado nenhuma solução conhecida, o autor aprimorou o desempenho do tempo computacional em 50% comparado seu trabalho anterior.

Os resultados obtidos por cada abordagem podem ser comparados na Tabela 1. As siglas IV, BI, EL, TA6, TA8, CH, ZH referem-se respectivamente aos trabalhos de Ivancic [1988], Bischoff

e Ratcliff [1995], Eley [2003], Takahara [2006], Takahara [2008], Che et al. [2011], e Zhu et al. [2012]. Os melhores resultados conhecidos (*Best-Known Solutions* – BKS) estão destacados em negrito. Comparando os resultados obtidos, é possível observar que o trabalho de Zhu et al. [2012] apresenta os melhores resultados encontrados na literatura para as instâncias propostas por Ivancic [1988]. O trabalho foi capaz de igualar o melhor resultado para 46 instâncias e superou os outros métodos em uma instância, diminuindo em uma unidade o número de contêineres utilizados para a instância 27.

Tabela 1: Comparativo entre os principais resultados encontrados para as instâncias de Ivancic [1988].

Inst.	IV	BI	EL	TA6	TA8	CH	ZH	BKS	Inst.	IV	BI	EL	TA6	TA8	CH	ZH	BKS
1	26	27	25	25	25	25	25	25	25	6	5	5	5	5	5	5	5
2	11	11	10	10	10	10	10	10	26	3	3	3	3	3	3	3	3
3	20	21	20	20	20	19	19	19	27	5	5	5	5	5	5	4	4
4	27	29	26	26	26	26	26	26	28	10	11	10	10	10	10	10	10
5	65	61	51	51	51	51	51	51	29	18	17	17	17	17	17	17	17
6	10	10	10	10	10	10	10	10	30	24	24	22	22	22	22	22	22
7	16	16	16	16	16	16	16	16	31	13	13	13	13	13	12	12	12
8	5	4	4	4	4	4	4	4	32	5	4	4	4	4	4	4	4
9	19	19	19	19	19	19	19	19	33	5	5	5	5	5	4	4	4
10	55	55	55	55	55	55	55	55	34	9	9	8	8	8	8	8	8
11	18	19	17	16	16	16	16	16	35	3	3	2	2	2	2	2	2
12	55	55	53	53	53	53	53	53	36	18	19	14	14	14	14	14	14
13	27	25	25	25	25	25	25	25	37	26	27	23	23	23	23	23	23
14	28	27	27	27	27	27	27	27	38	50	56	45	45	45	45	45	45
15	11	11	11	11	11	11	11	11	39	16	16	15	15	15	15	15	15
16	34	28	26	26	26	26	26	26	40	9	10	8	9	9	8	8	8
17	8	8	7	7	7	7	7	7	41	16	16	15	15	15	15	15	15
18	3	3	2	2	2	2	2	2	42	4	5	4	4	4	4	4	4
19	3	3	3	3	3	3	3	3	43	3	3	3	3	3	3	3	3
20	5	5	5	5	5	5	5	5	44	4	4	4	3	3	3	3	3
21	24	24	20	20	20	20	20	20	45	3	3	3	3	3	3	3	3
22	10	11	8	9	9	8	8	8	46	2	2	2	2	2	2	2	2
23	21	22	20	20	20	19	19	19	47	4	3	3	3	3	3	3	3
24	6	6	6	5	5	5	5	5									

3. Método proposto

Devido à natureza combinatória do CLP, avaliar todas soluções possíveis não é uma tarefa que possa ser realizada em tempo viável considerando problemas reais ou mesmo as instâncias disponíveis na literatura. Dentre os métodos utilizados para a resolução de problemas de otimização combinatória, optou-se pela utilização de heurísticas combinadas com modelos de programação inteira mista. Estes modelos, aliados a métodos heurísticos, podem ser resolvidos por *solvers* que estão em constante evolução. Segundo Achterberg e Wunderling [2013], apenas entre 1998 e 2012 o solver CPLEX, por exemplo, tornou-se cerca de 90 vezes mais rápido. Assim, o método proposto será diretamente beneficiado por melhorias futuras nos *solvers* de programação inteira mista. Esta característica torna metodologias como a proposta neste trabalho particularmente desejáveis [Fischetti et al., 2009].

A heurística matemática implementada neste trabalho é denominada *Relax-and-Fix*, e pode ser vista como uma adaptação do método proposto por Wolsey [1998]. A heurística consiste em uma busca local que utiliza um modelo de programação inteira mista. A ideia principal é fixar um grupo de variáveis selecionadas a partir de uma solução inicial e, assim, gerar subproblemas de mais fácil resolução do que o problema original. A cada iteração, um subproblema diferente é resolvido com o objetivo de melhorar a solução corrente. Este procedimento é executado repetidas

vezes, potencialmente resultando em soluções que, embora não sejam comprovadamente ótimas, apresentem melhorias sobre a solução inicial. O método proposto é apresentado pelo Algoritmo 1.

Algoritmo 1: FIXAÇÃO FORTE

Entrada: Tempo máximo t_{Max}
Saída: Uma solução do CLP
início
1 $S \leftarrow \text{Melhor}(\text{BestFit}, \text{FirstFitDecreasing})$
2 **enquanto** $\text{tempo} < t_{Max}$ **faça**
3 $G, \bar{G} \leftarrow \text{FormarGrupos}(S)$
4 $\mathcal{P}' \leftarrow \text{Subproblema}(S, G, \bar{G})$
5 $S \leftarrow \text{Solução do subproblema } \mathcal{P}'$
6 **fim**
7 **retorna** S
8 **fim**

Etapa 1: O algoritmo é iniciado pela escolha da melhor solução inicial S dentre as geradas pelos métodos construtivos *Best-Fit* e *First Fit Decreasing* (Linha 1). A seguir, os dois métodos construtivos são detalhados:

- *First Fit Decreasing*: é um método construtivo proposto por Crainic et al. [2008], baseado no conceito de pontos extremos (PE's). O método parte do princípio de que quando um item i com dimensões p_i , q_i e r_i é adicionado a um determinado contêiner, ele gera uma série de novos pontos em potencial, os pontos extremos, nos quais os itens restantes podem ser acomodados. Os itens são adicionados no contêiner da seguinte maneira:
 1. Para a inserção dos itens nos contêineres, os itens são ordenados por volume de forma decrescente, e se houver empate, os itens são ordenados pela altura de modo decrescente como critério de desempate.
 2. Se o contêiner estiver vazio, o item será colocado na posição $(0, 0, 0)$, o que gera PE's nas posições $(p_i, 0, 0)$, $(0, q_i, 0)$ e $(0, 0, r_i)$;
 3. Caso contrário, o item é colocado na posição (x_i, y_i, z_i) e os novos PE's são obtidos projetando os pontos $(x_i + p_i, y_i, z_i)$, $(x_i, y_i + q_i, z_i)$, e $(x_i, y_i, z_i + r_i)$.
 4. Se houver mais de um item no qual um ponto possa ser projetado, o algoritmo escolherá o mais próximo.
 5. Um novo contêiner é utilizado quando os itens restantes não couberem na lista de pontos extremos existentes.
- *Best-Fit*: é uma adaptação para o CLP do método proposto originalmente por Burke et al. [2004] para um problema de corte de duas dimensões. A ideia básica do algoritmo é que quando um item i com dimensões p_i , q_i e r_i é adicionado a um determinado contêiner, ele gera segmentos de linhas denominados *skylines*, nos quais os itens restantes podem ser acomodados. A implementação deste método segue as recomendações feitas por Imahori e Yagiura [2010]. Os itens são adicionados no contêiner da seguinte forma:
 1. O algoritmo replica todas possíveis rotações de um item em uma lista de itens I . Em seguida, os itens são ordenados pela largura, e se houver empate, pela profundidade;

2. Se o contêiner estiver vazio, o primeiro item $i \in I$ será colocado na posição $(0, 0, 0)$ do contêiner, o que gera dois *skylines* $\overline{0p_i}$ e $\overline{p_i L_j}$ como demonstrado pela Figura 1(a), sendo que L_j representa a largura do contêiner j ;
3. Toda vez que um item é alocado a um contêiner suas demais rotações são descartadas da lista I de itens;
4. Em seguida todos os itens $k \in I$ com largura e profundidade iguais a p_i e q_i são empilhados até que a altura do contêiner não seja ultrapassada. Suas demais rotações são removidas como indicado no passo 3;
5. Se já houver itens inseridos no contêiner, o algoritmo escolhe o item $i \in I$ que melhor ocupa o *skyline* mais inferior para ser inserido na base do contêiner, Figura 1(b). Em seguida, os itens são empilhados como indicado no passo 4;
6. Se nenhum item couber nos *skylines* disponíveis, é feita a junção de dois *skylines* vizinhos, resultando na inutilização de uma área do contêiner, como indica a Figura 1(c).
7. Se após todas as junções possíveis de *skylines* vizinhos, ainda não for possível alocar nenhum item nos *skylines* existentes, um novo contêiner é utilizado.

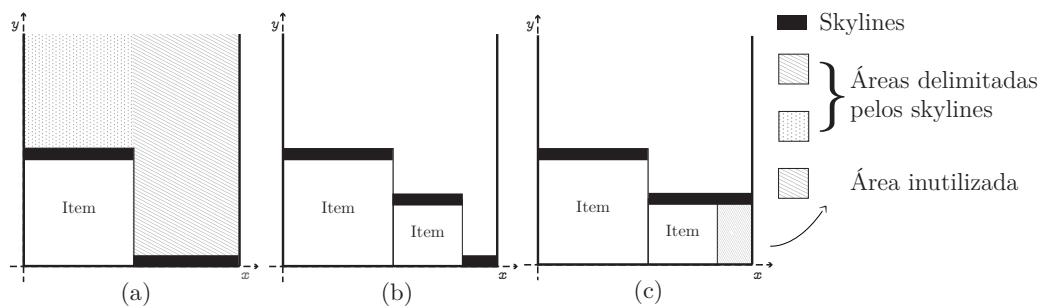


Figura 1: Representação dos *skylines* Fonte: Adaptado de Imahori e Yagiura [2010].

Etapla 2: Após a geração de uma solução inicial, a primeira tarefa consiste na escolha de quais itens serão submetidos à busca local (Linha 3 do Algoritmo 1). É possível adotar várias estratégias para esta tarefa, sendo que as seguintes abordagens foram testadas na execução deste trabalho:

- **Escolha de grupos de itens:** Com base em uma solução S , seleciona-se um ou mais conjunto de itens vizinhos. Dentro desta estratégia, há diversas questões que podem ser abordadas e que deixam a tarefa mais complexa como: qual a quantidade ideal de conjuntos a serem escolhidos; de quais contêineres retirar as conjuntos; e qual o melhor critério para escolher os conjuntos (aleatório, maior volume, etc).
- **Escolha de contêineres:** Nesta estratégia, são escolhidos um ou mais contêineres de uma solução S . Todos os itens alocados aos contêineres escolhidos são selecionados para busca local. Assim como na formação de grupos, esta abordagem requer um grande número de decisões em relação a escolha dos contêineres. Deve-se definir qual a quantidade ideal de contêineres escolhidos, qual critério deve ser adotado (aleatório, maior volume, menor volume, etc).

Etapla 3: Depois da seleção de itens para busca local, é gerado um novo subproblema $\mathcal{P}'$ (Linha 4 do Algoritmo 1). Para a resolução de um subproblema, deve-se definir qual modelo será usado na

etapa de busca local. O modelo proposto por Chen et al. [1995] suporta a utilização de contêineres de diversos tamanhos e a rotação de itens. Este modelo foi escolhido para implementação por ter obtido os melhores resultados nos experimentos apresentados por Silva et al. [2019]. De acordo com os autores, para uma instância com número de itens igual n , a formulação terá $3n^2 + 10n$ variáveis e $(13n^2 + 31n)/2$ restrições. O modelo utiliza a notação apresentada na Tabela 2 e é apresentado pelas Equações (1)–(25).

Tabela 2: Descrição das variáveis do modelo

Parâmetro	Descrição
I	Conjunto de itens.
J	Conjunto de contêineres disponíveis.
M	Número grande arbitrário.
p_i, q_i, r_i	Largura, profundidade e altura do item i .
L_j, W_j, H_j	Largura, profundidade e altura do contêiner j .
s_{ij}	Variável binária que indica se o item i é alocado no contêiner j .
n_j	Variável binária que indica se o contêiner j é usado.
x_i, y_i, z_i	Variáveis contínuas que indicam as coordenadas iniciais do eixo X , Y , e Z do item i .
l_{xi}, l_{yi}, l_{zi}	Variáveis binárias que indicam se a largura do item i é paralela ao eixo X , Y ou Z do contêiner.
w_{xi}, w_{yi}, w_{zi}	Variáveis binárias que indicam se a profundidade do item i é paralela ao eixo X , Y , ou Z do contêiner.
h_{xi}, h_{yi}, h_{zi}	Variáveis binárias que indicam se a altura do item i é paralela ao eixo X , Y ou Z do contêiner.
t_{ik}^r	Variável binária que indica se a posição relativa r é escolhida para o par de itens i e k .

A variável binária t_{ik}^r indica a posição $r \in R = \{1 \dots 6\}$ do item i em relação ao item k . A variável t_{ik}^1 é igual a 1 se o item i está à esquerda do item k e 0 caso contrário. De forma análoga, $t_{ik}^2, t_{ik}^3, t_{ik}^4, t_{ik}^5$, e t_{ik}^6 indicam se o item i está à direita, atrás, na frente, abaixo ou acima do item k , respectivamente. Note que estas variáveis são instanciadas apenas quando $i < k$, evitando restrições e variáveis desnecessárias, uma vez que se i está à direita de k , k está automaticamente à esquerda de i .

O problema tem como objetivo:

$$\min \sum_{j \in J} L_j W_j H_j n_j \quad (1)$$

Sujeito a:

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leq x_k + (1 - t_{ik}^1)M \quad \forall i, k \in I : i < k \quad (2)$$

$$x_k + p_k l_{xk} + q_k w_{xk} + r_k h_{xk} \leq x_i + (1 - t_{ik}^2)M \quad \forall i, k \in I : i < k \quad (3)$$

$$y_i + q_i w_{yi} + p_i l_{yi} + r_i h_{yi} \leq y_k + (1 - t_{ik}^3)M \quad \forall i, k \in I : i < k \quad (4)$$

$$y_k + q_k w_{yk} + p_k l_{yk} + r_k h_{yk} \leq y_i + (1 - t_{ik}^4)M \quad \forall i, k \in I : i < k \quad (5)$$

$$z_i + r_i h_{zi} + q_i w_{zi} + p_i l_{zi} \leq z_k + (1 - t_{ik}^5)M \quad \forall i, k \in I : i < k \quad (6)$$

$$z_k + r_k h_{zk} + q_k w_{zk} + p_k l_{zk} \leq z_i + (1 - t_{ik}^6)M \quad \forall i, k \in I : i < k \quad (7)$$

$$\sum_{r \in R} t_{ik}^r \geq s_{ij} + s_{kj} - 1 \quad \forall i, k \in I : i < k, \forall j \in J \quad (8)$$

$$\sum_{r \in R} t_{ik}^r \leq 1 \quad \forall i, k \in I : i < k \quad (9)$$

$$\sum_{j \in J} s_{ij} = 1 \quad \forall i \in I \quad (10)$$

$$s_{ij} \leq n_j \quad \forall i \in I, \forall j \in J \quad (11)$$

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leq L_j + (1 - s_{ij})M \quad \forall i \in I, \forall j \in J \quad (12)$$

$$y_i + q_i w_{yi} + p_i l_{yi} + r_i h_{yi} \leq W_j + (1 - s_{ij})M \quad \forall i \in I, \forall j \in J \quad (13)$$

$$z_i + r_i h_{zi} + q_i w_{zi} + p_i l_{zi} \leq H_j + (1 - s_{ij})M \quad \forall i \in I, \forall j \in J \quad (14)$$

$$l_{xi} + l_{yi} + l_{zi} = 1 \quad \forall i \in I \quad (15)$$

$$w_{xi} + w_{yi} + w_{zi} = 1 \quad \forall i \in I \quad (16)$$

$$z_{xi} + z_{yi} + z_{zi} = 1 \quad \forall i \in I \quad (17)$$

$$l_{xi} + w_{xi} + h_{xi} = 1 \quad \forall i \in I \quad (18)$$

$$l_{yi} + w_{yi} + h_{yi} = 1 \quad \forall i \in I \quad (19)$$

$$l_{zi} + w_{zi} + h_{zi} = 1 \quad \forall i \in I \quad (20)$$

$$l_{xi}, l_{yi}, l_{zi}, w_{xi}, w_{yi}, w_{zi}, h_{xi}, h_{yi}, h_{zi} \in \{0, 1\} \quad \forall i \in I \quad (21)$$

$$t_{ik}^r \in \{0, 1\} \quad \forall i, k \in I : i < k, \forall r \in R \quad (22)$$

$$s_{ij} \in \{0, 1\} \quad \forall i \in J, \forall j \in J \quad (23)$$

$$n_j \in \{0, 1\} \quad \forall j \in J \quad (24)$$

$$x_i, y_i, z_i \geq 0 \quad \forall i \in I \quad (25)$$

A função objetivo (1) consiste em minimizar a soma do volume dos contêineres utilizados. As Restrições (2)–(7) garantem que os itens não se sobrepõem se estiverem no mesmo contêiner, o que é verificado pelas Restrições (8). As Restrições (9) foram introduzidas ao modelo de Chen et al. [1995] com o objetivo de reduzir simetria, pois apenas uma variável $t_{ik}^r \forall r \in R$ que assuma valor 1 já é suficiente para atender as restrições de sobreposição. As Restrições (10) asseguram que cada item vai ser alocado a exatamente um contêiner e as Restrições (11) proíbem que itens sejam alocados a contêineres não utilizados. As Restrições (12)–(14) asseguram que todos os itens alocados em um contêiner cabem inteiramente no interior do mesmo, e as Restrições (15)–(20) garantem que cada dimensão do item esteja paralela a exatamente um eixo do contêiner (ou seja, apenas uma rotação é selecionada para cada item). Por fim, as Restrições (21)–(25) indicam quais variáveis são binárias e impedem que as demais assumam valores negativos.

As instâncias utilizadas no desenvolvimento deste trabalho consideram apenas um tamanho de contêiner. Assim, é possível substituir as Restrições (12)–(14) pelas Restrições (26)–(28), reduzindo o número de restrições em (12)–(14) de $3 \times |I| \times |J|$ para apenas $3 \times |I|$.

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leq L \quad \forall i \in I \quad (26)$$

$$y_i + q_i w_{yi} + p_i l_{yi} + r_i h_{yi} \leq W \quad \forall i \in I \quad (27)$$

$$z_i + r_i h_{zi} + q_i w_{zi} + p_i l_{zi} \leq H \quad \forall i \in I \quad (28)$$

Após a definição do modelo, a seguinte abordagem para a fixação de variáveis é proposta: com base em uma solução corrente S , e considerando E o conjunto dos itens selecionados para a busca local, as variáveis são adicionadas em dois grupos distintos G e $\bar{G}$. O grupo G é formado por todas as variáveis relacionadas aos itens $i \in E$, ou seja, $x_i, y_i, z_i, t_{ik}^r, l_{xi}, l_{yi}, l_{zi}, w_{xi}, w_{yi}, w_{zi}, h_{xi}, h_{yi}, h_{zi}$ e s_{ij} ; e as variáveis binárias n_j relacionadas aos contêineres $j \in J$. Considerando que V é o conjunto de todas as variáveis do modelo, o grupo $\bar{G}$ contém as demais variáveis, ou seja:

$\bar{G} = V/G$. Os valores das variáveis $v \in \bar{G}$ obtidos na solução S são fixados, enquanto o modelo de programação inteira mista é responsável por determinar os valores das variáveis $v \in G$. Note que a resolução deste modelo equivale a encontrar a melhor solução vizinha em uma grande vizinhança que inclui todas as soluções que podem ser obtidas se mantivermos as decisões representadas pelas variáveis no conjunto $\bar{G}$.

Por fim, além de ter como objetivo a redução do volume de contêineres utilizados, o subproblema ainda conta com uma função objetivo auxiliar visando excluir o contêiner com menor volume de itens alocados. Este contêiner é denominado c_0 . Assim a função objetivo apresentada em 1 é substituída pela função objetivo $\min \sum_{j \in J} L_j W_j H_j n_j + \sum_{i \in c_0} p_i q_i r_i s_{ic_0}$. O princípio por trás da estratégia é retirar os itens do c_0 e tentar alocá-los em outros contêineres. Desta forma, talvez seja possível obter uma nova solução na qual o c_0 seja eliminado após algumas iterações, e assim obter uma solução com o número de contêineres inferior ao da solução inicial encontrada mais facilmente. Também busca-se promover alterações na solução S , de forma a diversificar o conjunto de soluções avaliadas, potencialmente evitando alguns ótimos locais.

Etapa 4: A solução corrente S é então atualizada com a solução obtida pela resolução do modelo para o subproblema $\mathcal{P}'$ (Linha 5 do Algoritmo 1). Ao final, a melhor solução S encontrada é retornada (Linha 7 do Algoritmo 1).

4. Experimentos Computacionais

Esta seção descreve os experimentos realizados para avaliar o métodos proposto, e compara os resultados obtidos com os disponíveis na literatura. Foram escolhidas as 47 instâncias providas por Ivancic [1988]. Estas instâncias foram amplamente utilizadas na literatura revisada para avaliar o desempenho de algoritmos para o CLP, e caracterizam um caso especial do problema, no qual há apenas um tipo de contêiner disponível para seleção. Dessa maneira, a função objetivo apresentada em (1) pode ser simplificada para $\min \sum_{j \in J} n_j$, sendo que n_j é a variável binária que indica se o contêiner j é utilizado pela solução ($n_j = 1$) ou não ($n_j = 0$).

Todos os experimentos foram executados em um computador Dell PowerEdge R710 com processador Intel Xeon CPU E5620 @ 2.40GHz e 78GB de memória RAM executando o sistema operacional *CentOS Linux 7 64 bits*. Os algoritmos foram implementados na linguagem de programação Java 8, e o *solver* Gurobi 9.0¹ foi utilizado para a resolução dos modelos. Durante todo o processamento, apenas a aplicação desenvolvida foi executada para que não houvesse interferência nos resultados obtidos.

Para a execução dos testes, foi determinado um tempo limite de execução de 30 minutos. A análise dos resultados consiste inicialmente em observar a diferença entre as soluções obtidas e as soluções encontradas na literatura. Os resultados obtidos encontram-se na Tabela 3. A sigla SI representa o melhor valor da solução inicial encontrada pelos métodos construtivos, a sigla FIX representa a solução obtida pelo método proposto, BKS representa o melhor resultado encontrado na literatura (*Best Known Solution*), e DIF representa a diferença entre o método proposto e a literatura.

Analisando os resultados é possível observar que o *Relax-and-Fix* foi capaz de diminuir a quantidade total de contêineres em 70 unidades em relação ao total obtido pelas soluções iniciais. Enquanto o *Relax-and-Fix* encontrou um número total de 708 contêineres utilizados para todas as instâncias, o número total de contêineres utilizados pela na literatura é de 691. Ao realizar o cálculo do GAP entre o total de contêineres obtido pela Fixação Forte e o total dos melhores resultados conhecidos, que é definido por $(\sum_1^{47} \text{FIX} - \sum_1^{47} \text{BKS}) / \sum_1^{47} \text{FIX} \times 100$, obtém-se o

¹<http://www.gurobi.com/>

Tabela 3: Resultado para as instâncias de Ivancic [1988].

Inst.	SI	FIX	BKS	DIF	Inst.	SI	FIX	BKS	DIF	Inst.	SI	FIX	BKS	DIF
1	25	25	25	0	17	9	8	7	1	33	5	5	4	1
2	10	9	10	-1	18	3	2	2	0	34	9	9	8	1
3	20	20	19	1	19	4	3	3	0	35	3	3	2	1
4	27	26	26	0	20	6	5	5	0	36	14	14	14	0
5	55	52	51	1	21	23	22	20	2	37	23	23	23	0
6	10	10	10	0	22	11	9	8	1	38	45	45	45	0
7	16	16	16	0	23	22	20	19	1	39	18	15	15	0
8	4	4	4	0	24	7	6	5	1	40	12	9	8	1
9	24	19	19	0	25	5	5	5	0	41	21	15	15	0
10	55	55	55	0	26	4	3	3	0	42	5	4	4	0
11	18	16	16	0	27	6	6	4	2	43	4	3	3	0
12	55	53	53	0	28	11	10	10	0	44	4	3	3	0
13	27	25	25	0	29	22	17	17	0	45	3	3	3	0
14	33	27	27	0	30	26	24	22	2	46	2	2	2	0
15	15	11	11	0	31	15	13	12	1	47	4	4	3	1
16	33	26	26	0	32	5	4	4	0					

valor de 2.40%. Ao comparar os resultados individualmente, o método proposto igualou o melhor resultado conhecido para 31 instâncias, enquanto que para doze instâncias a diferença foi de uma unidade a mais comparada com os resultados da literatura. A maior diferença encontrada foi de dois contêineres a mais em relação aos resultados já conhecidos para as instâncias 21, 27 e 30. Para a instância 2, o *Relax-and-Fix* obteve uma solução na qual são utilizados nove contêineres, enquanto na literatura são utilizados dez contêineres, indicando que o método proposto pode ser combinado com outros métodos para gerar soluções melhores que as conhecidas atualmente.

5. Conclusões

O presente trabalho teve como objetivo o desenvolvimento de uma heurística matemática aplicável ao CLP e a avaliação da qualidade do método proposto. Na Seção 2 foi realizada a revisão de diversos trabalhos na literatura referentes ao CLP, o que proporcionou uma melhor compreensão acerca do problema. Realizou-se ainda o estudo de diversos métodos e conceitos empregados na resolução do CLP a fim de escolher uma abordagem a ser implementada. A Seção 3 apresentou os principais conceitos do método proposto e detalhes referentes à implementação do *Relax-and-Fix*.

Após a análise dos resultados, é possível concluir que o *Relax-and-Fix* é um método bastante promissor, reduzindo a quantidade total de contêineres utilizados em 70 unidades em relação às soluções iniciais, se igualando a melhor solução conhecida em 31 ocasiões e sendo capaz de melhorar uma melhor solução conhecida, indicando que pode ser combinado com outros métodos para gerar soluções de qualidade.

6. Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) por meio da concessão de bolsa de mestrado. Os autores agradecem também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) pelo acesso ao portal de periódicos, e ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Ouro Preto pelos recursos cedidos para a realização deste trabalho.

Referências

Achterberg, T. e Wunderling, R. (2013). *Mixed Integer Programming: Analyzing 12 Years of Progress*, p. 449–481. ISBN 978-3-642-38188-1.

- Bischoff, E. E., Janetz, F., e Ratcliff, M. (1995). Loading pallets with non-identical items. *European journal of operational research*, 84(3):681–692. URL [https://doi.org/10.1016/0377-2217\(95\)00031-K](https://doi.org/10.1016/0377-2217(95)00031-K).
- Bischoff, E. E. e Ratcliff, M. (1995). Issues in the development of approaches to container loading. *Omega*, 23(4):377–390. URL [https://doi.org/10.1016/0305-0483\(95\)00015-G](https://doi.org/10.1016/0305-0483(95)00015-G).
- Bortfeldt, A. e Wäscher, G. (2013). Constraints in container loading - a state-of-the-art review. *European Journal of Operational Research*, 229(1):1 – 20. ISSN 0377-2217. URL <https://doi.org/10.1016/j.ejor.2012.12.006>.
- Burke, E. K., Kendall, G., e Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Oper. Res.*, 52(4):655 – 671. ISSN 0030-364X. URL <https://doi.org/10.1287/opre.1040.0109>.
- Che, C. H., Huang, W., Lim, A., e Zhu, W. (2011). The multiple container loading cost minimization problem. *European Journal of Operational Research*, 214(3):501 – 511. ISSN 0377-2217. URL <https://doi.org/10.1016/j.ejor.2011.04.017>.
- Chen, C., Lee, S., e Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68 – 76. ISSN 0377-2217. URL [https://doi.org/10.1016/0377-2217\(94\)00002-T](https://doi.org/10.1016/0377-2217(94)00002-T).
- Crainic, T. G., Perboli, G., e Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *Inform. Journal on computing*, 20(3):368–384.
- Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145 – 159. ISSN 0377-2217. URL [https://doi.org/10.1016/0377-2217\(90\)90350-K](https://doi.org/10.1016/0377-2217(90)90350-K).
- Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1):45–60. ISSN 1436-6304. URL <https://doi.org/10.1007/s002910200113>.
- Fischetti, M., Lodi, A., e Salvagnin, D. (2009). Just mip it! In *Matheuristics*, p. 39–70. Springer.
- Imahori, S. e Yagiura, M. (2010). The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. *Computers and Operations Research*, 37(2):325 – 333. ISSN 0305-0548. URL <https://doi.org/10.1016/j.cor.2009.05.008>.
- Ivancic, N. J. (1988). *An Integer Programming Based Heuristic Approach to the Three Dimensional Packing Problem*. Case Western Reserve University.
- Silva, E. F., Toffolo, T. A. M., e Wauters, T. (2019). Exact methods for three-dimensional cutting and packing: a comparative study concerning single container problems. *Computers and Operations Research*. ISSN 0305-0548. URL <https://doi.org/10.1016/j.cor.2019.04.020>.
- Takahara, S. (2006). A simple meta-heuristic approach for the multiple container loading problem. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, p. 2328–2333. URL <https://www.doi.org/10.1109/ICSMC.2006.385210>.

- Takahara, S. (2008). A multi-start local search approach to the multiple container loading problem. In Bednorz, W., editor, *Greedy Algorithms*, chapter 4. IntechOpen, Rijeka. URL <https://doi.org/10.5772/6358>.
- Wäscher, G., Haußner, H., e Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130. ISSN 0377-2217. URL <https://doi.org/10.1016/j.ejor.2005.12.047>.
- Wolsey, L. (1998). *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley. ISBN 9780471283669.
- Zhu, W., Huang, W., e Lim, A. (2012). A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research*, 223(1):27–39. URL <https://doi.org/10.1016/j.ejor.2012.05.039>.