Algoritmo para a Minimização do uso de Recipientes no Problema do Empacotamento Bidimensional com Restrições de Guilhotina

Leonardo Borck†
niverisdade do Vale do Itajaí
Itajaí Santa Catarina Brasil
l.borck@edu.univali.br

Alex Luciano Roesler Rese
Univerisdade do Vale do Itajaí
Itajaí Santa Catarina Brasil
alexrese@univali.br

Thiago Felski Pereira
Univerisdade do Vale do Itajaí
Itajaí Santa Catarina Brasil
thiagofelski@univali.br

ABSTRACT

Cutting and packing problems are prevalent in various industries, logistics, and the construction sector. Therefore, the study of this class of problems is justified due to its potential to reduce material wastage and associated costs, among other benefits. This paper presents a study aimed at minimizing the number of bins required to pack all items in the rectangular two-dimensional variable-sized bin packing problem. This problem involves guillotine constraints without rotations. The heuristics ILS and a modified BCSA were implemented. After conducting preliminary tests, the use of ILS was discarded, and only the proposed modified BCSA was employed to obtain results. To validate the performance and quality of the developed algorithm, various experiments were conducted on benchmark instances from the literature, including the dataset of [1], dataset of [2], and dataset of [3], and compared with several other heuristics. Consequently, it was found that the modified BCSA applied in this work achieved excellent results in most classes, with an average improvement of 14.6% over the compared heuristics in the first dataset and 14.08% in the second.

KEYWORDS

Two-dimensional Bin Packing Problem. Binary Crow Search Algorithm. Guillotine Constraints. Variable-Sized Bins.

1 Introdução

A otimização do corte e empacotamento de materiais apresenta um desafio significativo em diversos setores, influenciando diretamente a eficiência produtiva e os custos logísticos. Na indústria moveleira, a necessidade de transformar painéis de madeira retangulares em componentes menores para produtos, conforme destacado por [4], exemplifica a importância de um planejamento eficiente para minimizar o desperdício de material. De forma similar, na logística, o arranjo eficiente de conteúdo dentro de contêineres é essencial para otimizar custos e maximizar o uso do espaço disponível [5]. Embora o problema de empacotamento pertença à classe NP-difícil, sua resolução eficiente é crucial para aprimorar a eficiência operacional e reduzir custos em diversos setores industriais.

A dificuldade em resolver o problema de empacotamento está em sua complexidade combinatória. O número de configurações possíveis cresce exponencialmente com o número de itens envolvidos, tornando abordagens de força bruta para encontrar solução ótima impraticáveis devido ao tempo computacional necessário [6]. Esse crescimento exponencial no tempo de solução com o aumento do tamanho do problema exige a exploração de métodos alternativos para viabilizar sua execução.

O uso de heurísticas é uma alternativa promissora para encontrar soluções em um tempo computacional viável [7]. Embora as heurísticas não garantam a solução ótima, elas oferecem soluções suficientemente próximas do ótimo em um tempo aceitável. Estudos demonstraram resultados promissores com a aplicação de heurísticas para problemas de corte e empacotamento. Por exemplo, [8] propuseram uma heurística baseada em uma busca local de melhoria modificada, demonstrando que a abordagem pode gerar soluções próximas do ótimo global mesmo para problemas altamente complexos.

Esse progresso ressalta a importância da pesquisa contínua e do desenvolvimento de novas heurísticas para aprimorar a eficiência e a qualidade das soluções para problemas de corte e empacotamento. Este artigo propõe um algoritmo para minimizar o uso de contêineres no problema de empacotamento bidimensional com restrições de corte guilhotinado. O objetivo é desenvolver uma abordagem que minimize o desperdício de material e otimize o planejamento dos cortes e o carregamento de contêineres, oferecendo uma solução prática e eficiente para problemas reais enfrentados na indústria e na logística.

2 Problema

A dificuldade de solucionar esse problema está na variedade combinatória, em que o número de possíveis configurações cresce com o número de itens. À medida que o tamanho do problema aumenta, o tempo necessário para encontrar uma solução ótima aumenta, tornando a abordagem por força bruta inviável [3].

Assim, a utilização de heurísticas, como a ILS (Iterated Local Search - Busca local Iterada), o BCSA (Binary Crow Search Algorithm - Algoritmo de Busca do Corvo Binário) e até mesmo uma busca gulosa ou de algoritmos genéticos se tornam uma alternativa viável. Estes algoritmos encontram soluções próximas à ótima sem o crescimento exponencial do tempo de execução. No entanto, não há garantia de que essas soluções serão ótimas, e é difícil avaliar o quão próximas estão sem conhecer a solução ótima em si. Para isso utiliza-se de bases de dados que informam a solução ótima para as instâncias. Contudo, continua a ser um

desafio para a área de otimização combinatória justificando o interesse em estudos. A versão bidimensional do problema, permite impor restrições de guilhotina, que apenas padrões que sejam possíveis separar todos os itens realizando cortes de ponta a ponta e a possibilidade de rotacionar os itens em 90° para obter um melhor encaixe no recipiente.

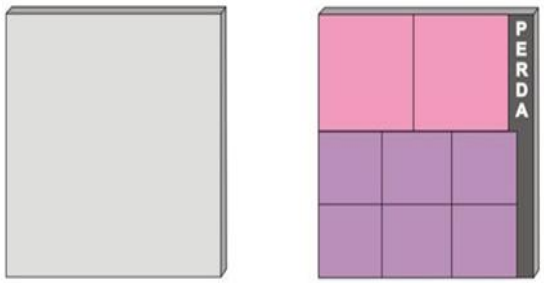

Figura 1: Exemplo para o problema do empacotamento bidimensional [5]

3 Solução Proposta

A implementação de um algoritmo heurístico, permite encontrar uma solução adequada, considerando tempo e qualidade. No trabalho foram aplicadas heurísticas: a ILS e o BCSA e comparado com trabalhos que utilizaram outras heurísticas. Já em relação ao problema foi utilizada sua versão de empacotamento de retângulos bidimensional como base para o estudo com o tamanho de recipientes variáveis, restrições de guilhotina e rotação de 90°.

O objetivo principal deste trabalho foi desenvolver um algoritmo baseado nas meta-heurísticas de ILS e do BCSA, para solucionar o problema do empacotamento bidimensional com recipientes heterogêneos, ou seja, que possuem diversos tamanhos e com restrições de guilhotina. No entanto, este trabalho é concentrado em encontrar padrões que minimizem a quantidade total de recipientes necessária para empacotar todos os itens de modo com que seja reduzido o custo com matéria prima para realização dos cortes dos itens requeridos respeitando as restrições de corte de guilhotina (2VSBP|O|G).

4 Desenvolvimento

Esta seção procura apresentar o funcionamento da ILS e BCSA, as quais foram utilizadas para construção do algoritmo que é aplicado. Foram desenvolvidas as duas abordagens separadamente e deveriam ser aplicadas em sequência de modo com que a entrada do BCSA fossem os resultados obtidos pela ILS. Porém, devido à análise dos experimentos preliminares realizados, uma abordagem se mostrou mais valiosa e, portanto, o uso da ILS foi descartado.

A ILS contempla a definição apresentada por [9]. No Quadro 1 é apresentado o seu pseudocódigo, que será utilizado de base para o algoritmo implementado neste trabalho. Inicialmente devem ser geradas as soluções iniciais, que para tal processo será utilizada a uma construção aleatória. A aleatoriedade da solução inicial oferece uma vantagem significativa na exploração do espaço de busca ajudando a evitar que o algoritmo fique preso em soluções ruins. Permite que o algoritmo explore várias partes do espaço de busca, o que, por sua vez, aumenta as chances de encontrar soluções melhores próximos ou mesmo a ótima global.

Na solução inicial, é aplicada uma busca local, que por definição, cria para cada solução um grupo de soluções semelhantes, chamado de vizinhança. Dada uma solução atual, uma maneira de implementar um algoritmo de busca local é explorar essa vizinhança em busca de uma solução com um valor menor, se estivermos lidando com um problema de minimização. Se uma solução vizinha com um valor menor for encontrada, essa se torna a nova solução atual, e o algoritmo prossegue. No entanto, se não for possível encontrar uma solução vizinha melhor o algoritmo termina. Portanto pode-se dizer que permite somente alterações que melhoram a solução atual, logo o resultado encontrado pelo algoritmo da busca local é um ótimo local.

```
Procedimento padrão da ILS
S0 = GeraSolucaoInicial
S* = BuscaLocal(S0)
Repita
   S' = Perturbacao(S*, historico)
   S*' = BuscaLocal(S')
   S* = CriterioDeAceitacao(S*, S*', historico)
Até que a condição de término seja atendida
Fim
```

Quadro 1: Procedimento padrão da ILS.

O Quadro 2, apresenta o funcionamento do BCSA, é possível observar que primeiramente o enxame de corvos precisa ser inicializado, de maneira a definir uma posição inicial para cada corvo. Junto a este inicializar sua memória, que no primeiro momento tem por definição é a posição inicial. Após a inicialização do enxame de corvos, algumas soluções inviáveis podem ocorrer no espaço de busca. Uma solução é considerada inviável quando viola as restrições de capacidade dos recipientes utilizados. Para isso é aplicado um operador de reparo de soluções inviáveis, onde reorganiza os recipientes que estão sobrecarregados, isto é, consiste primeiro em remover itens dos recipientes sobrecarregados, um por vez, de forma a respeitar suas capacidades de carga. Em seguida, insere os itens removidos um por um nos recipientes anteriormente bem preenchidos que podem acomodá-los.

Na próxima etapa deverá ser atualizada a posição de cada corvo por meio do processo de binarização, a qual é utilizada uma função sigmoide para converter soluções com valor real em binárias. Após isso, a viabilidade das soluções geradas é avaliada através de uma função de aptidão, se for necessário, o operador de reparação proposto é aplicado novamente para restaurar a viabilidade das soluções.

Uma vez avaliada a qualidade das soluções, a memória de cada corvo deve ser atualizada. Se a nova posição de um corvo for melhor que a posição memorizada mais recentemente, o corvo

atualiza sua memória com esta nova posição. As etapas anteriores são repetidas até que um tempo máximo de iteração seja atingido.

```
Procedimento padrão do BCSA
E* = InicializaEnxame(E0, S0)
Se AsSolucoesSaoViaveis(E*) Retornar Falso
Senão
  E* = ReparaSolucoesInviaveis(E*)
Repita
  E' = AtualizaPosicao(E*)
  Se AsSolucoesSaoViaveis(E') Retornar Falso
  Senão
      E' = ReparaSolucoesInviaveis(E')
  E* = CriterioDeAceitacao(E', E*)
Até que a condição de término seja atendida
Fim
```

Quadro 2: Procedimento padrão do BCSA.

Na Figura 2, é apresentada a representação visual do fluxo de execução do algoritmo por meio de um diagrama de sequência, onde mostram a realização de três execuções, sendo a primeira, a execução da ILS gerando as soluções iniciais para o BCSA, a segunda somente a ILS e por fim, o BCSA.

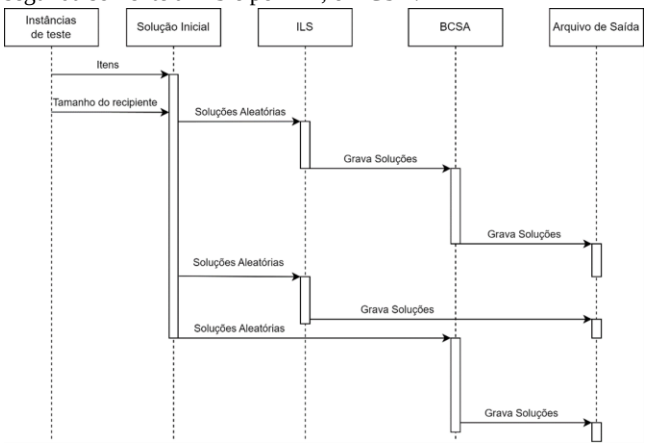

Figura 2: Diagrama de sequência do algoritmo.

4.1 Experimentos

Para avaliar a desempenho do algoritmo proposto para este trabalho, foram realizadas simulações em diferentes classes da literatura. Essas instâncias possuem conjuntos de dados com recipientes com tamanhos idênticos ou heterogêneos. Para isso, foram selecionadas instâncias presentes no repositório criado por [10], que unifica diversos conjuntos de instâncias da literatura além de que todos os conjuntos de dados incluídos no repositório estão formatados na mesma estrutura.

Os experimentos foram divididos em dois conjuntos de instâncias presentes na literatura, com dez classes cada, O conjunto 1, será composto apenas por instâncias com recipientes quadrados de tamanhos idênticos, as classes I - VI propostas por [1] e classes VII - X propostas por [2], já o Conjunto 2, possui dez classes que foram propostas por [3] e apresenta recipientes retangulares com tamanhos heterogêneos.

Cada uma dessas classes possui dez instâncias com 20, 40, 60, 80 e 100 itens retangulares e o número de recipientes não é definido e poderá variar dependendo da execução do algoritmo. Portanto, as classes que são utilizadas em cada conjunto contemplam, ao todo, um número de 500 instâncias.
As primeiras seis classes utilizadas no Conjunto 1, propostas por [1] e são caracterizadas da seguinte forma:

- Classe I: hj e wj aleatórias no intervalo [1, 10], W = H = 10;
- Classe II: hj e wj aleatórias no intervalo [1, 10], W = H = 30;
- Classe III: hj e wj aleatórias no intervalo [1, 35], W = H = 40;
- Classe IV: hj e wj aleatórias no intervalo [1, 35], W = H = 100;
- Classe V: hj e wj aleatórias no intervalo [1, 100], W = H = 100;
- Classe VI: hj e wj aleatórias no intervalo [1, 100], W = H = 300;

Onde h e w representam respectivamente a altura e largura dos itens em unidades. H e W a altura e largura dos recipientes em unidades. Nas últimas quatro classes, propostas por [2], os itens são classificados em quatro tipos.

- Tipo 1: wj aleatória no intervalo, hj aleatória no intervalo;
- Tipo 2: wj aleatória no intervalo, hj aleatória no intervalo;
- Tipo 3: wj aleatória no intervalo, hj aleatório no intervalo;
- Tipo 4: wj aleatório no intervalo, hj aleatório no intervalo;

Os tamanhos dos recipientes têm altura e largura definidas como 100 unidades para todas as classes, enquanto os itens são os seguintes:

- Classe VII: Tipo 1 com probabilidade de 70%, já os Tipos 2, 3 e 4 com probabilidade de 10% cada;
- Classe VIII: Tipo 2 com probabilidade de 70%, já os Tipos 1, 3 e 4 com probabilidade de 10% cada;
- Classe IX: Tipo 3 com probabilidade de 70%, já os Tipo 1, 2 e 4 com probabilidade de 10% cada;
- Classe X: Tipo 4 com probabilidade de 70%, já os Tipos 1, 2 e 3 com probabilidade de 10% cada;

Já as instâncias que compõem o Conjunto 2, propostas por [3], foram criadas com base nas instâncias com os conjuntos de recipientes de tamanhos idênticos de [1] e [2]. Para isso foram criados 5 tipos de recipientes em cada instância. Os tamanhos dos recipientes foram escolhidos de forma uniforme e aleatória a partir de [W/2, W] × [H/2, H], onde W e H são a largura e a altura dos compartimentos do problema original.

Todos os experimentos foram realizados em um computador com o processador AMD Ryzen 5 5600G que opera na frequência

de processamento de 3.9GHz à 4.45 GHz, possui 6 núcleos e 12 processadores lógicos. Também utiliza memória ram de 32 GB operando na frequência de 3200MHz, executando o sistema operacional Windows 11 de 64 bits.

4.2 Parâmetros

Cada instância passou por 30 execuções completas do algoritmo, explorando os parâmetros de forma exaustiva, a fim de avaliar seu desempenho. O uso de múltiplas repetições visa introduzir variabilidade aos fatores não controláveis dos experimentos, que resultam de seleções feitas de forma aleatória, seguindo uma distribuição uniforme:
Parâmetros da ILS:

- Taxa de perturbação (p): {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9};
- Número máximo de iterações sem melhora: {500}.
- Parâmetros do BCSA:
- Tamanho do Bando (b): {20, 35, 50};
- Tamanho do voo (v): {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
- Número máximo de iterações: {500}.

O número de replicações para cada configuração de parâmetros escolhido, utilizado para a formação da média estatística, com base na tabela t da distribuição t de Student apresentada por [11], este valor pode ser justificado, devido a diminuição da aproximação a partir de amostras com o tamanho de 30 ou mais elementos. Para esgotar as configurações de parâmetros que foram definidas são necessárias 270 execuções, que executa 500 instâncias cada, se considerarmos que cada instância necessite de 1 segundos para ser finalizada cada instância, com 30 replicações de cada configuração serão necessárias pelo menos 1125 horas, equivalente a 47 dias para cada base de testes.

2 Redução dos Parâmetros

Para realizar todos os experimentos em tempo hábil foi necessário efetuar uma redução nos parâmetros, previamente sugeridos. Visto que a quantidade de experimentos geraria aproximadamente uma diversidade de 9.9 milhões execuções para os dois conjuntos de instâncias propostos, se cada uma destas levasse 1 segundo este processo demoraria 115 dias ininterruptos de execuções.

Para realizar a redução dos parâmetros foi realizada uma execução preliminar somente com a Classe I do conjunto de instâncias proposto por [1], totalizando um total de 50 instâncias de teste, em 30 replicações. Na ILS apenas o número de iterações máximas sem melhora foi reduzido, pois as soluções estavam convergindo antes desse valor. No BCSA, a configuração de parâmetros foi selecionada, com base nos melhores resultados balanceando o número de recipientes utilizados e a porcentagem de utilização média.

Parâmetros da ILS:

- Taxa de perturbação (p): {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9};
- Número máximo de iterações sem melhora: {250}.

Parâmetros do BCSA:

- Tamanho do Bando (b): {35};
- Tamanho do voo (v): {5};
- Número máximo de iterações: {80}.

5 Resultados

O objetivo desta seção é apresentar o processo de análise dos resultados obtidos com a aplicação do algoritmo proposto nesse trabalho de conclusão nos dois conjuntos de instâncias da literatura descritos na Seção 4.1. No decorrer do capítulo são exibidos os resultados deste trabalho, divididos por conjunto de instâncias. São, respectivamente, o conjunto de instâncias com recipientes quadrados de tamanhos idênticos e o conjunto de instâncias com recipientes retangulares de tamanhos heterogêneos. Por fim, é realizada uma análise geral dos resultados, onde são comparados os resultados com a literatura.

5.1 Conjunto de Tamanhos Idênticos

Nessa subseção são apresentados os resultados da execução do conjunto de classes proposto por [1] e por [2]. A Figura 3 e a Figura 4 apresentam a média dos recipientes utilizados segmentados pelos números de itens presentes em cada instância e por classe, respectivamente. Nota-se que instâncias com mais itens para empacotar necessitam de mais recipientes. Além disso, é notável que nas classes 2, 4, 6 possuem menos recipientes, isso se dá, pois, a área dos recipientes, nessas classes é consideravelmente maior que a área de um único item, fazendo com que muitos itens possam ser empacotados no mesmo recipiente.

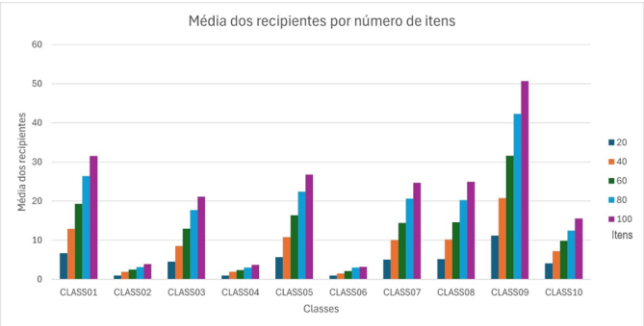

Figura 3: Média dos recipientes por número de itens.

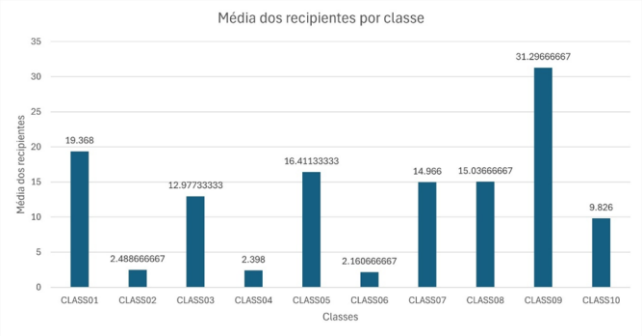

Figura 4: Gráficos da média dos recipientes por classe.

A Figura 5 e a Figura 6 apresentam, respectivamente, a porcentagem média de utilização dos recipientes por item e por classe. É possível verificar que nas classes 2, 4 e 6 possui uma média consideravelmente menor que as outras classes, principalmente nas instâncias que possuem poucos itens. Isso se dá, pelo mesmo motivo explicado anteriormente, em alguns casos, por exemplo, com 20 itens, foi necessário apenas um recipiente para empacotar todos os itens, atingindo apenas 42.4% de ocupação no pior caso.

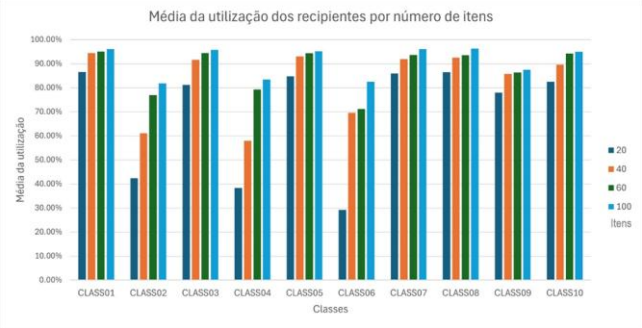

Figura 5: Utilização média dos recipientes por número de itens.

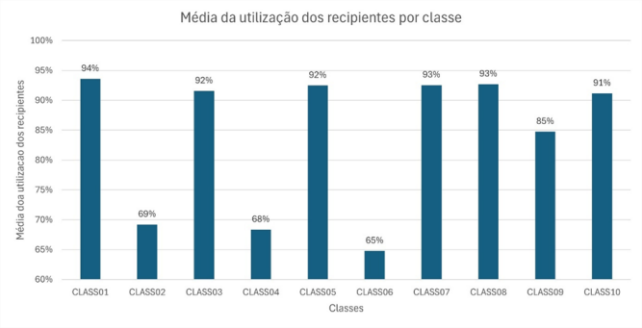

Figura 6: Utilização média dos recipientes por classe.

5.2 Conjunto de Tamanhos Heterogêneos

Nesta subseção são apresentados os resultados da aplicação do algoritmo nas instâncias que compõem o conjunto de [3]. Com 5 tipos de recipientes predefinidos por instância, os quais, possuem tamanhos gerados aleatoriamente que variam da metade até tamanho total do recipiente original, de modo com que sejam necessários mais recipientes para empacotar todos os itens, devido ao tamanho reduzido. O conjunto ainda mantém certos comportamentos das instâncias originais. E no geral, a média de utilização dos recipientes para este conjunto é maior. Isso pode ocorrer, por conta de os recipientes serem retangulares, assim como os itens, em alguns casos.

As Figura 7 e Figura 8 apresentam a média dos recipientes utilizados segmentado por número de itens e por classe. Onde, é possível verificar, que o uso de recipientes aumentou em todas as classes, principalmente onde o tamanho dos itens é muito próximo ao da definição dos recipientes no conjunto original. Além disso, o comportamento se manteve independentemente da quantidade de itens.

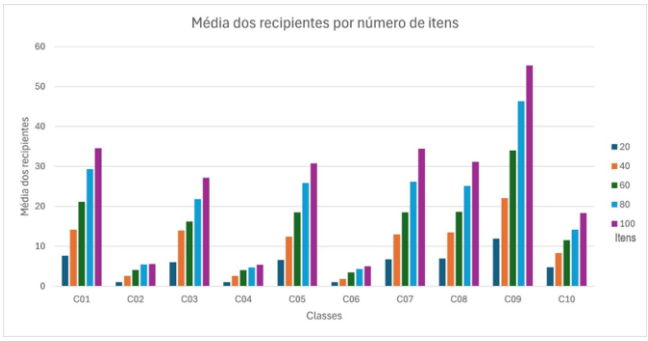

Figura 7: Média dos recipientes por número de itens.

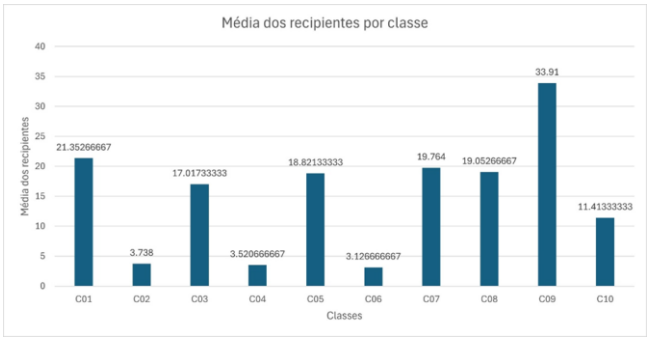

Figura 8: Média dos recipientes por classe.

Quanto a porcentagem de utilização dos recipientes, é mostrado na Figura 9 e Figura 10, que houve um aumento muito estável, atingindo ocupação de acima de 90% para todas as quantidades de itens empacotadas em todas as classes, exceto nas classes 2, 4 e 6. De qualquer modo, a média geral de todas as classes se manteve acima de 92% e no melhor caso aproximadamente 99% de ocupação.

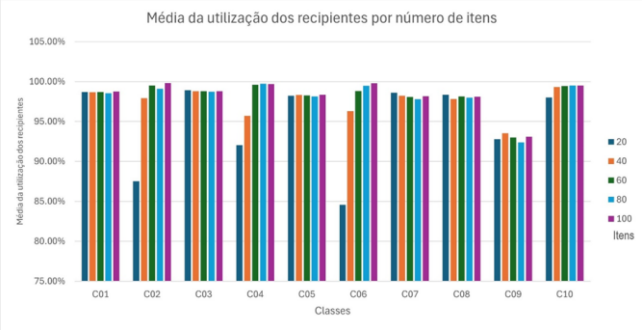

Figura 9: Utilização média dos recipientes por itens.

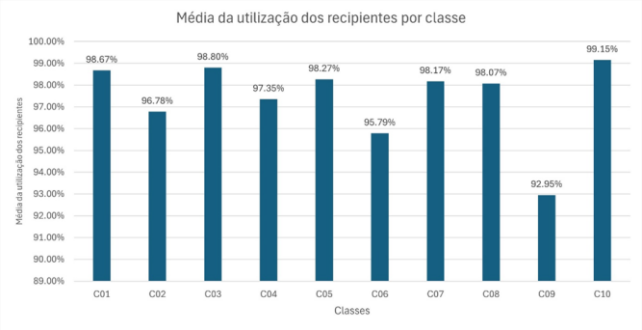

Figura 10: Utilização média dos recipientes por classe.

5.3 Conjunto de Tamanhos Heterogêneos

As comparações realizadas nesta subseção têm o objetivo de analisar os resultados obtidos através do BCSA modificado, com os resultados encontrados na literatura que utilizaram os mesmos conjuntos de dados. Na Tabela 1, é analisada apenas a porcentagem de utilização média dos recipientes em cada classe e por fim uma média geral, para essa análise foi utilizado a mesma equação para obter a porcentagem de utilização apresentada por [12], onde y representa a porcentagem da área dos recipientes que tem itens contidos nelas. Uma utilização de 95%, por exemplo, corresponde a 5% de área do recipiente de sobra.

A análise foi realizada comparando os resultados obtidos neste trabalho heurísticas existentes na literatura, são: GDRR [12], FFDH (First Fit Decreasing Height – Primeira Melhora com Redução de Altura) [13], BFDH (Best-Fit Decreasing Height Algorithm – Algoritmo de Melhor Melhora com Redução de Altura) [14], FCOG (Floor-Ceiling Algorithm - Algoritmo Piso-Teto) [2], BFDH* (Improved Best-Fit Decreasing Height Algorithm – Algoritmo Aprimorado de Melhor Melhora com Redução de Altura) [15], BFS (Best-Fit with Stacking Algorithm – Algoritmo de Melhor Melhora com Pilha) [16] e HHA [17].

Nota-se que o BCSA obteve ótimos resultados, superando consistentemente todas as outras heurísticas consideradas. Com uma média de utilização geral de 97.40%, é a heurística mais eficaz na maioria das classes analisadas, vale ressaltar que o GDRR é o algoritmo que obteve resultados mais próximos do BCSA modificado, em especial, nas classes 2, 4 e 6.

Classe	Este Trabalho	GDRR	FFDH	BFDH	FCOG	BFDH*	BFS	HHA
1	**98.7**	94.1	86.3	86.6	87.3	87.4	88.6	91.5
2	**96.8**	96.5	83.7	83.7	85.2	85	85.1	96.3
3	**98.8**	91.5	81.1	81.7	81.9	81.9	82.2	86.6
4	**97.4**	93.5	80	80	82.7	82.1	82	91.7
5	**98.3**	89.3	80.4	81.1	81.3	81.2	81.4	84.6
6	**95.8**	92.7	79.1	79.3	80.7	80.5	79.5	90.2
7	**98.2**	90.1	79.9	80.2	80.8	80.6	80.9	86.9
8	**98.1**	89.4	80.7	81.1	81.3	81.2	81.6	85.9
9	**93.0**	75.6	72.8	72.6	72.6	72.8	73	74.3
10	**99.1**	93	83.3	83.8	85.4	84.6	85.5	90.3
Média de utilização [%]	**97.40**	90.66	80.73	81.01	81.92	81.73	81.98	87.83

Tabela 1: Resultados das instâncias.

É valido salientar, que os resultados obtidos neste trabalho também mostram ser eficiente para o 2VSBP, evidenciado por uma análise de gap. Em termos gerais, o método proposto apresentou uma melhora média de 14.08%, sendo 6,92% em relação ao GDRR, 17,12% em relação ao FFDH, 16,82% em relação ao BFDH, 15,94% em relação ao FCOG, 16,09% em relação ao BFDH*, 15,84% em relação ao BFS e 9,82% em relação ao HHA. As heurísticas que mais se aproximam são o GDRR e o HHA, com 6,92% e 9,82%, respectivamente. Especialmente na classe 2 do conjunto, onde obtiveram 0.31% e 0.52%, respectivamente. Isso se dá, devido ao BCSA modificado, se sair melhor em ambientes onde é necessário utilizar um número grande de recipientes. Em contraste, os maiores gaps foram em relação ao FFDH e BFDH.

5.4 Migração para C#

A implementação original do BCSA em C++ estava levando cerca de 4 horas para rodar 10 repetições apenas para a classe 1 conjunto de instâncias de Berkey e Wang (1987) e Lodi, Martello e Vigo (1999). Considerando que o problema possui 10 classes no total, o tempo total de execução seria impraticável para a análise completa, visto que são dois conjuntos de dados, o que comprometeu a viabilidade do estudo dentro do prazo disponível. A análise completa de cada conjunto requer a execução do algoritmo em todas as 10 classes, com 30 repetições para gerar variação dos fatores não controláveis dos experimentos, dado por escolhas com critério aleatório, sob distribuição uniforme, e garantir a precisão dos resultados.

Isso se dá devido a representação da solução ser uma matriz binária, para os casos menores onde necessitam de poucos recipientes, o algoritmo consegue obter resultados em tempo satisfatório, mas de acordo com que aumenta o número de linhas (itens) e colunas (recipientes) da matriz, cada operação leva mais tempo para ser executada, tendo em vista, que a cada iteração é necessário percorrer a matriz diversas vezes. Foi realizada uma tentativa de paralelismo do algoritmo em C++, mas devido à falta de domínio da linguagem, não foi possível.

Inicialmente o algoritmo foi migrado para a linguagem C# devido a experiencia com a linguagem e o uso da biblioteca LINQ que demonstrou trabalhar mais rapidamente com listas muito grandes do que o C++ com vetores muito grandes, só está migração já gerou resultados positivos em instâncias maiores,

com 100 itens, por exemplo. Além disso convertido os laços de repetição principais do algoritmo para Parallel.ForEach que de maneira muito facilitada utiliza o máximo de threads do processador, isso fez com que o uso de processador durante a execução passasse de aproximadamente 20% no algoritmo em C++ para 100% durante todo o processo em C++. A Figura 11 apresenta o consumo de processamento.

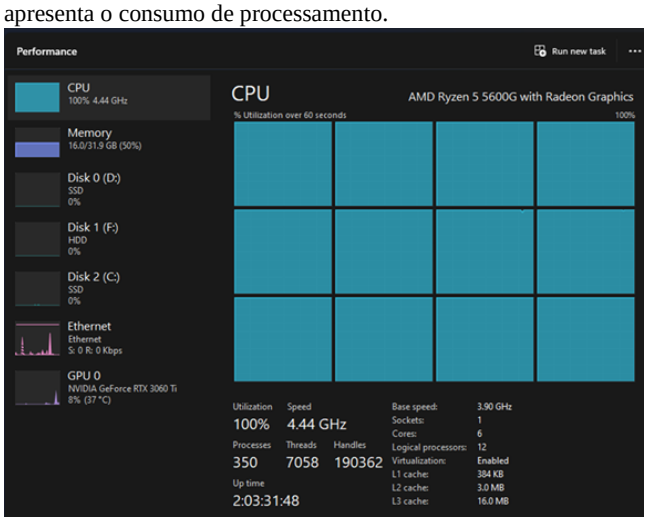

Figura 11: Uso de processamento durante a execução do BCSA em C#.

5.4.1 Resultados

Com a implementação do paralelismo, o tempo total de execução para rodar as 10 classes com 30 repetições foi reduzido para apenas 41 minutos e 11 segundos para o conjunto com recipientes de tamanho fixo e 50 minutos e 27 segundos para os com tamanho aleatório, representando uma melhoria substancial em relação à implementação original em C++.

Figura 12: Tempo execução com paralelismo. (a) conjunto com tamanho fixo; (b) conjunto com tamanho aleatório.

6 Considerações Finais

A otimização dos processos de corte e empacotamento bidimensional representa um desafio relevante em múltiplos setores, impactando diretamente a eficiência da produção e os custos logísticos. Nesse contexto o presente trabalho teve como objetivo o desenvolvimento de um algoritmo com base na ILS e o BCSA para a minimização do espaço ocupado por objetos em recipientes, respeitando as restrições de guilhotina.

A partir do estudo realizado, foi iniciada etapa de implementação, que é a criação de um algoritmo de busca heurística baseado na ILS e no BCSA. Após a codificação do algoritmo foi realizada a pré-análise dos parâmetros em relação à demanda de tempo. Neste processo foram filtrados os melhores parâmetros para as heurísticas, visto que as outras apresentavam menor qualidade e/ou demandavam muito tempo para encontrar a solução final.

Foi realizada uma série de execuções parciais preliminares dos conjuntos de dados, que demonstraram que a aplicação da ILS não gerou melhora em relação ao algoritmo de BCSA sozinho. Portanto, optou-se por utilizar apenas o BCSA modificado que foi apresentado neste trabalho. Durante a implementação, surgiram desafios relacionados ao tempo de execução do algoritmo, o que demandou a migração do algoritmo para C# e a implementação de paralelismo, possibilitando assim a execução do algoritmo dentro do prazo necessário.

Para avaliar o desempenho foram utilizados dois conjuntos de instâncias diferentes e comparados com diversas outras heurísticas presentes na literatura. A análise dos resultados mostrou que a implementação realizada neste trabalho consegue atingir soluções de qualidade razoável em tempo polinomial. Além disso, os resultados indicaram que o BCSA modificado obteve um desempenho superior nos conjuntos de dados aplicados, em contraste com os trabalhos comparados. Em média, foi observada uma melhoria de 14,6% no primeiro conjunto de dados e 14,08% no segundo.

AGRADECIMENTOS

Agradeço imensamente aos meus pais, à minha namorada e ao meu professor orientador e todos que ajudaram para a conclusão deste artigo. Seu apoio e orientação foram essenciais para a conclusão desta etapa.

REFERÊNCIAS

[1] BERKEY, J. O.; WANG, P. Y. Two-dimensional finite bin-packing algorithms. Journal of the Operational Research Society, 38(5), 423–429, 1987. https://doi.org/10.2307/2582731.

[2] LODI, MARTELLO, Silvano; VIGO, Daniele. (1999) Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. INFORMS Journal on Computing 11(4):345-357.

[3] PISINGER, David; Sigurd Mikkel. The two-dimensional bin packing problem with variable bin sizes and costs, Discrete Opiomization 2. P. 154-167, 2005.

[4] CAVALI, Roberto. Problemas de corte e empacotamento na indústria de móveis: um estudo de caso. 2004. Disponível em: http://hdl.handle.net/11449/94286. Acesso em: 06 maio 2023.

[5] SEABRA JR, Edward; POZZO, Daniel Marcos Dal; VERGOPOLAN, Paulo Roberto; ROSA, Mônica Cristina Lunkes da. Modelo de algoritmo heurístico proposto para o problema de carregamento de container. Revista Técnico-Científica do CREA-PR, 2019.
[6] HÄMÄLÄINEN, W. Class NP, NP-complete, and NP-hard problems. 2006. Disponível em: https://www.cs.joensuu.fi/pages/whamalai/daa/npsession.pdf. Acesso em: 06 maio 2023.
[7] MARTELLO, Silvano and TOTH, Paolo. Knapsack problems: algorithms and computer implementations. Chichester: John Wiley & Sons, 1990. 296 p. DOI: 10.12691/jcsa-2-2-2.
[8] El-ASHMAWI, Walaa H; ELMINAAM, Diaa Salama Abd. A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem. Applied Soft Computing, v. 82, 2019.
[9] LOURENÇO, Helena R.; MARTIN, Olivier C.; STÜTZLE, Thomas. Iterated Local Search. In: GLOVER, Fred; KOCHENBERGER, Gary A. Handbook of Metaheuristics. Palo Alto: Kluwer Academic Publishers, 2003. Cap. 11. p. 321 353.
[10] OLIVEIRA, O.; GAMBOA, D.; SILVA, E. Resources for two-dimensional (and three-dimensional) cutting and packing solution methods research. In Proceedings of the 16th international conference on applied computing: vol. 53 (pp. 131–138), 2019.
[11] SANTOS FILHO, Herondino dos. Distribuição T - Student. Macapá: Unifap, 2014. 26 slides, color. Disponível em: https://www2.unifap.br/herondino/files/2014/04/8-DISTRIBUI%C3%87%C3%83O-T-STUDENT.pdf. Acesso em: 16 out. 2023.
[12] GARDEYN, Jeroen; WAUTERS, Tony. A goal-driven ruin and recreate heuristic for the 2D variable-sized bin packing problem with guillotine constraints. European Journal Of Operational Research. Numa, Belgium, p. 432-444. 20 nov. 2021. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0377221721009826. Acesso em: 12 out. 2023.
[13] COFFMAN, Edward G., Jr.; GAREY, Michael R.; JOHNSON, David S.; TARJAN, Robert E. Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing, v. 9, n. 4, p. 808–826, 1980.
[14] COFFMAN, Edward G., Jr.; SHOR, Peter. Average-case analysis of cutting and packing in two dimensions. European Journal of Operational Research, v. 44, n. 2, p. 134–144, 1990. Disponível em: https://doi.org/10.1137/0209062. Acesso em: 30 maio 2024.
[15] BORTFELDT, Andreas. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research, v. 172, n. 3, p. 814–837, 2006. Disponível em: https://doi.org/10.1016/j.ejor.2004.11.016. Acesso em: 30 maio 2024.
[16] ORTMANN, Francois G.; NTENE, Nelson; VAN VUUREN, Jan H. New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. European Journal of Operational Research, v. 203, n. 2, p. 306–315, 2010. Disponível em: https://doi.org/10.1016/j.ejor.2009.07.024. Acesso em: 30 maio 2024.
[17] HONG, S.; ZHANG, D.; LAU, H. C.; ZENG, X.; SI, Y. W. A hybrid heuristic algorithm for the 2D variable-sized bin packing problem. European Journal of Operational Research, 238(1), 95–103, 2014. https://doi.org/10.1016/j.ejor.2014.03.049.