UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA

CURSO DE GRADUAÇÃO EM MATEMÁTICA INDUSTRIAL

DAVI PEREIRA DE OLIVEIRA

ANÁLISE DE HEURÍSTICAS NO MÉTODO DE BRANCH-AND-BOUND BASEADO EM DIAGRAMAS DE DECISÃO APLICADO AO PROBLEMA DO CONJUNTO INDEPENDENTE DE PESO MÁXIMO

FORTALEZA

2025

DAVI PEREIRA DE OLIVEIRA

ANÁLISE DE HEURÍSTICAS NO MÉTODO DE BRANCH-AND-BOUND BASEADO EM DIAGRAMAS DE DECISÃO APLICADO AO PROBLEMA DO CONJUNTO INDEPENDENTE DE PESO MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Matemática Industrial do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Matemática Industrial.

Orientador: Prof. Dr. Tibérius de Oliveira e Bonates

FORTALEZA

2025

DAVI PEREIRA DE OLIVEIRA

ANÁLISE DE HEURÍSTICAS NO MÉTODO DE BRANCH-AND-BOUND BASEADO EM DIAGRAMAS DE DECISÃO APLICADO AO PROBLEMA DO CONJUNTO INDEPENDENTE DE PESO MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Matemática Industrial do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Matemática Industrial.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Tibérius de Oliveira e Bonates (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jesus Ossian da Cunha Silva
Universidade Federal do Ceará (UFC)

Prof. Dr. Luis Gustavo Bastos Pinho
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Agradeço, com profunda gratidão, à minha família, cujo apoio incondicional foi essencial ao longo de minha jornada acadêmica. Sem vocês, nada disso teria sido possível. Obrigado, mãe, Vera. Obrigado, pai, Márcio. Obrigado, meus irmãos. Cada conquista desta caminhada também pertence a vocês.

Ao meu orientador, Prof. Dr. Tibérius de Oliveira e Bonates, expresso minha sincera gratidão pela paciência, dedicação e valiosa orientação. Seus ensinamentos foram fundamentais para o desenvolvimento deste trabalho, e sua inspiração seguirá comigo além desta etapa.

Aos membros da banca examinadora, Prof. Dr. Jesus Ossian da Cunha Silva e Prof. Dr. Luis Gustavo Bastos Pinho, agradeço pela generosidade em compartilhar seu tempo e conhecimento. Suas sugestões e questionamentos foram essenciais para o aprimoramento desta pesquisa, enriquecendo ainda mais este trabalho.

Aos meus colegas de faculdade, que tornaram essa caminhada mais leve e significativa, expresso minha sincera gratidão. Foram anos de desafios, aprendizados e crescimento mútuo, sempre acompanhados por apoio, troca de conhecimentos e momentos inesquecíveis. A convivência com vocês foi fundamental não apenas para minha formação acadêmica, mas também para minha evolução pessoal. Obrigado por cada conversa, cada colaboração e por todas as memórias construídas ao longo dessa trajetória.

Por fim, expresso minha gratidão à Universidade Federal do Ceará (UFC), instituição que me proporcionou uma formação de excelência e um ambiente acadêmico enriquecedor. Agradeço aos professores, funcionários e a todos que contribuíram direta ou indiretamente para minha trajetória universitária. Foi uma honra fazer parte desta instituição e levar comigo todo o conhecimento e experiências adquiridas.

“Um problema bem formulado é um problema meio resolvido.”

(Charles Kettering)

RESUMO

Diagramas de decisão são uma alternativa aos métodos exatos de Programação Linear Inteira (PLI), representando o conjunto de soluções viáveis de um problema de otimização discreta por meio de um grafo acíclico e direcionado, particionado em $n+1$ camadas de nós. Para isso, o problema deve atender a duas condições: possuir uma formulação válida de Programação Dinâmica (PD) e uma função objetivo aditivamente separável. Um Diagrama de Decisão (DD) pode crescer exponencialmente dependendo da sua entrada. Para evitar esse crescimento, duas abordagens são adotadas: o DD restrito e o DD relaxado, que representam, nessa ordem, um subconjunto e um superconjunto das soluções do problema original, ao mesmo tempo que evitam o crescimento exponencial do DD. Com isso, torna-se possível aplicar o método de Branch-and-Bound (BAB) para encontrar soluções ótimas, o que é conhecido como método de BAB baseado em DDs. Este trabalho avalia a influência de heurísticas de exclusão de nós (DD restrito), heurísticas de seleção de nós para mesclagem (DD relaxado) e ordenações de variáveis (usadas para compilar os DDs) na qualidade dos limites retornados. Além disso, investiga-se o impacto da combinação desses fatores no desempenho do BAB baseado em DDs. O problema escolhido para o estudo foi o Problema do Conjunto Independente de Peso Máximo (PCIPM), um caso específico do clássico Problema do Conjunto Independente Máximo (PCIM). O estudo analisou três heurísticas de seleção de nós para mesclagem, três heurísticas de exclusão de nós e três tipos de ordenação de variáveis. Com isso, foi possível tirar conclusões a respeito da influência de cada heurística na qualidade dos limites retornados e a importância de cada fator no desempenho do método de BAB. Além disso, observou-se que heurísticas originalmente desenvolvidas para o PCIM apresentam um desempenho satisfatório para o PCIPM. Uma comparação com um *solver* de PLI também foi realizada, demonstrando a eficácia dos DDs para algumas instâncias.

Palavras-chave: diagrama de decisão; programação dinâmica; otimização discreta; *branch-and-bound* baseado em diagramas de decisão, problema do conjunto independente máximo, problema do conjunto independente de peso máximo.

ABSTRACT

Decision diagrams are an alternative to exact PLI methods, representing the set of feasible solutions of a discrete optimization problem through a directed acyclic graph, partitioned into $n+1$ layers of nodes. For this, the problem must satisfy two conditions: having a valid PD formulation and an additively separable objective function. A DD can grow exponentially depending on its input. To prevent this growth, two approaches are adopted: the restricted DD and the relaxed DD, which represent, in this order, a subset and a superset of the solutions of the original problem while avoiding the exponential growth of the DD. With this, it becomes possible to apply the BAB method to find optimal solutions, which is known as the BAB method based on DDs. This work evaluates the influence of node exclusion heuristics (restricted DD), node selection heuristics for merging (relaxed DD), and variable orderings (used to compile DDs) on the quality of the returned bounds. Furthermore, the impact of combining these factors on the performance of BAB based on DDs is investigated. The chosen problem for this study was the PCIPM, a specific case of the classical PCIM. The study analyzed three node selection heuristics for merging, three node exclusion heuristics, and three types of variable ordering. This allowed us to derive conclusions concerning the influence of each heuristic on the quality of the returned bounds and the importance of each factor on the performance of the BAB method. Additionally, it was observed that heuristics originally developed for PCIM perform satisfactorily for PCIPM. A comparison with a PLI *solver* was also conducted, demonstrating the effectiveness of DDs for some instances.

Keywords: decision diagram; dynamic programming; discrete optimization; decision diagram-based branch-and-bound; maximum independent set problem; maximum weighted independent set problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

BAB	Branch-and-Bound
CF	Corte de Fronteira
DD	Diagrama de Decisão
GB	Grupo de Borda
MNE	Mínimo Número de Estados
MVC	Melhor Valor de Caminho
ON	Ordenação Natural
PCIM	Problema do Conjunto Independente Máximo
PCIPM	Problema do Conjunto Independente de Peso Máximo
PD	Programação Dinâmica
PLI	Programação Linear Inteira
PVC	Pior Valor de Caminho
RA	Remoção Aleatória
RT	Ramificação Tradicional
SGA	Soma dos Graus Atuais
UCE	Última Camada Exata

SUMÁRIO

1 INTRODUÇÃO

"Nos últimos anos, o uso de Diagramas de Decisão (DDs) tem crescido significativamente na área de otimização discreta" (BERGMAN; LOZANO, 2021). Um Diagrama de Decisão (DD) é um grafo dirigido e acíclico capaz de representar o conjunto de soluções viáveis de um problema de otimização discreta, desde que este possua uma formulação válida de Programação Dinâmica (PD) e uma função objetivo aditivamente separável. Um DD possui um nó inicial r e um nó final t, no qual cada caminho p de r a t representa um solução viável para o problema de referência. A concepção inicial dos DDs foi apresentada por Lee (1959), embora seu propósito original não estivesse relacionado à resolução de problemas de otimização discreta. Os DDs também têm sido utilizados de forma híbrida com outros métodos de Programação Linear Inteira (PLI). Trabalhos como o de Gonzalez *et al.* (2020) exploram essa metodologia híbrida.

Os DDs exatos são capazes de codificar todo o conjunto de soluções viáveis de um problema, o que, como consequência, pode levar a um crescimento exponencial de seu tamanho em relação à entrada. Para contornar essa limitação, Andersen *et al.* (2007) propuseram DDs relaxados com largura e tamanho limitados. Um DD relaxado representa todas as soluções viáveis do problema, além de incluir algumas soluções inviáveis. Consequentemente, ele fornece um limite superior para o valor ótimo de um problema de otimização discreta, assumindo que este seja um problema de maximização. A relaxação é realizada por meio de uma operação válida que mescla determinados nós do DD exato. "Resultados experimentais indicam que DDs relaxados são eficazes em problemas estruturados, sendo capaz de superar tecnologias de ponta de programação inteira em diferentes ordens de magnitude" (BERGMAN *et al.*, 2011). Bergman *et al.* (2015) utilizou a ideia de DDs relaxados para a obtenção de limites lagrangianos.

Por outro lado, enquanto os DDs relaxados podem retornar soluções inviáveis, em certos casos o interesse está exclusivamente em encontrar soluções viáveis para problemas específicos de otimização discreta. Nesse contexto, os DDs restritos garantem a obtenção de soluções viáveis para o problema em questão, ou seja, se o problema em questão é de maximização, um limite inferior será retornado pelo DD restrito. A ideia central dos DDs restritos consiste em excluir alguns dos nós do DD exato para controlar o seu crescimento exponencial. Como resultado, algumas soluções parciais são perdidas durante esse processo. DDs restritos foram introduzidos inicialmente por Bergman *et al.* (2014).

A construção dos DDs é realizada com base nas variáveis de decisão do problema em análise. "A ordenação dessas variáveis dentro do DD tem impacto direto na qualidade das relaxações obtidas" (BERGMAN *et al.*, 2012). Além disso, aspectos como a escolha dos nós para mesclagem no DD relaxado e a exclusão de nós no DD restrito também influenciam significativamente a qualidade dos limites alcançados.

Considerando que há limites inferior e superior para um problema de otimização discreta, BERGMAN *et al.* (2016) propuseram a utilização do método de *Branch-and-Bound* (BAB) baseado em DDs para encontrar soluções ótimas.

O Problema do Conjunto Independente de Peso Máximo (PCIPM) é uma generalização do Problema do Conjunto Independente Máximo (PCIM), na qual pesos positivos são atribuídos aos vértices do grafo. Karp (1972) provou indiretamente que ambos os problemas pertencem à classe NP-difícil. Por se tratarem de problemas clássicos de otimização combinatória, eles têm sido amplamente estudados, e diversos métodos eficientes foram desenvolvidos para encontrar boas soluções. Um dos métodos que permite encontrar soluções ótimas para esses problemas é o algoritmo de BAB, introduzido originalmente por Land e Doig (1960), amplamente utilizado na resolução de problemas de PLI, incluindo o PCIM e o PCIPM. Por possuírem uma formulação válida de PD, uma função objetivo aditivamente separável e pertencerem ao domínio da otimização discreta, ambos os problemas podem ser resolvidos por meio de DDs.

1.1 Objetivos

O objetivo principal deste trabalho é realizar uma avaliação computacional dos principais fatores que impactam o desempenho do método BAB baseado em DDs, aplicado ao PCIPM. São objetivos específicos:

- Avaliar ordenações de variáveis relacionadas ao problema;
- Analisar o impacto das heurísticas de exclusão de nós no DD restrito e heurísticas de escolha de nós a serem mesclados no DD relaxado;
- Identificar quais heurísticas propostas são mais adequadas para o problema;
- Estudar o efeito combinado das heurísticas no desempenho do BAB baseado em DDs, aplicado ao PCIPM.

1.2 Organização

No Capítulo 2, são apresentadas as principais definições relacionadas aos DDs, bem como a forma como o PCIPM pode ser resolvido utilizando essas estruturas, além disso, são abordadas as definições de problemas de otimização discreta e detalhados o PCIM e o PCIPM. No Capítulo 3, são descritos a metodologia empregada, as instâncias utilizadas, o ambiente computacional dos experimentos, as ordenações de variáveis consideradas, as heurísticas para seleção de nós a serem mesclados e as heurísticas de eliminação de nós. No Capítulo 4, são apresentados os resultados obtidos, e no Capítulo 5, as conclusões deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Problemas de otimização discreta

Seja $\mathscr{P}$ um problema de otimização discreta, onde $X = \{x_1, \ldots, x_n\}$ representa o conjunto das n variáveis de decisão de $\mathscr{P}$, e $D = D_1 \times \cdots \times D_n$ denota o produto cartesiano dos domínios dessas variáveis, sendo D_j o domínio da variável x_j, para $j = 1, \ldots, n$, com D_j sendo um conjunto finito. Além disso, seja $f : D \to \mathbb{R}$ uma função de valor real definida sobre D, e $C = \{C_1, \ldots, C_m\}$ o conjunto que representa as m restrições de $\mathscr{P}$, onde cada restrição $C_i(x)$ impõe uma relação sobre as variáveis de decisão, ou seja, define o conjunto de soluções de $\mathscr{P}$.

Seja x uma solução viável de $\mathscr{P}$, então $x \in D$ satisfaz todas as restrições $C_i(x)$. O conjunto $Sol(\mathscr{P})$ denota o conjunto de todas as soluções viáveis de $\mathscr{P}$. Uma solução x^* é ótima para $\mathscr{P}$ se $f(x^*) \geq f(x)$ para todo $x \in Sol(\mathscr{P})$, sendo $z^* = f(x^*)$ o valor ótimo de $\mathscr{P}$. Portanto, $\mathscr{P}$ é um problema de otimização discreta na forma:

$$\begin{aligned} &\text{Maximizar} && f(x) \\ &\text{sujeito à} && C_i(x), \quad i = 1, \ldots, m \quad (\mathscr{P}) \\ & && x \in D \end{aligned}$$

Exemplo 1. *Problema da mochila 0 - 1: Dada uma mochila com capacidade máxima C e um conjunto de n itens distintos. Cada item i, para $i = 1, \ldots, n$, possui um peso p_i e um valor v_i associado. Considere $x_i \in \{0,1\}$ a decisão de alocar ou não o item i na mochila. O problema está em alocar os itens cujo somatório de v_i seja máximo e cujo o peso total não ultrapasse a capacidade C. Portanto, o problema da mochila 0 - 1 é formulado da seguinte maneira:*

$$\begin{aligned} &\textit{Maximizar} && \sum_{i=1}^{n} v_i x_i \\ &\textit{sujeito à} && \sum_{i=1}^{n} p_i x_i \leq C \\ & && x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{aligned}$$

2.2 Conjunto independente de peso máximo

Dado um grafo $G = (V, E)$, onde V representa o conjunto de vértices e E denota o conjunto de arestas, um conjunto independente é definido como um subconjunto $I \subseteq V$, no qual, para todo par de vértices $\{i, j\} \subseteq I$, $\{i, j\} \notin E$. O objetivo do PCIM é encontrar o maior conjunto independente. Por outro lado, o PCIPM surge quando, para cada vértice $j \in V$, existe

um peso associado $w_j \geq 0$. O objetivo é encontrar um conjunto independente I tal que a soma dos pesos w_j seja máximo.

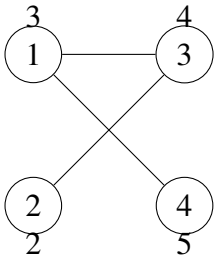
Karp (1972) demonstrou que o problema da clique máxima é NP-difícil. Uma clique K de um grafo G é um subconjunto $K \subseteq V$ tal que, para todo par de vértices $\{i, j\} \subseteq K$, existe uma aresta $e \in E$ conectando i e j. Note que o problema da clique máxima e o PCIM são problemas complementares. Se K é uma clique máxima em G, então K será um conjunto independente máximo no grafo complementar $\overline{G}$, no qual, para todo par de vértices $\{i, j\} \subseteq K$, as arestas $e \in E$ não existem. Como consequência, conclui-se o PCIM também é um problema NP-difícil. Sendo o PCIPM uma generalização direta, ele herda a mesma complexidade computacional. O problema de otimização discreta que representa o PCIPM é denotado na Equação 2.1. A mesma formulação é válida para o PCIM, só alterando cada peso w_j do vértice j para 1:

Expressar no texto a formulação matemática do proble

$$\begin{aligned} \text{Maximizar} \quad & \sum_{j=1}^{n} w_j x_j \\ \text{sujeito à} \quad & x_i + x_j \leq 1, \quad \forall \{i, j\} \in E \\ & x_j \in \{0, 1\}, \quad \forall j \in V \end{aligned} \tag{2.1}$$

Exemplo 2. *Considere a instância do PCIPM ilustrada na figura abaixo:*

Figura 1 – Exemplo PCIPM



Fonte: Próprio autor.

Melhorar a descrição do exemplo:V = {1,2,3,4}W = {3, 2, 4

No Exemplo 2, o conjunto $\{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{2,4\}, \{3,4\}\}$ denota as soluções viáveis da instância ilustrada. O conjunto independente $\{3,4\}$ é o conjunto independente de peso máximo da instância, com $z^* = 9$.

2.3 Diagramas de decisão

Diagrama de Decisão (DD) é uma estrutura de multigrafo direcionado, acíclico e ponderado utilizado para representar o conjunto $Sol(\mathscr{P})$ de um problema de otimização discreta

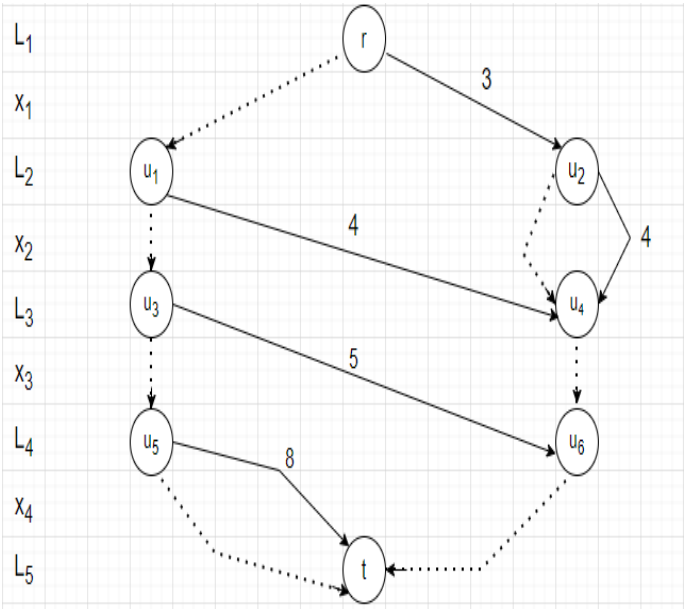
$\mathscr{P}$. Formalmente, BERGMAN *et al.* (2016) descrevem um DD como $B = (U, A, d)$, onde U é o conjunto de nós, A o conjunto de arcos, e d representa os rótulos associados a esses arcos, no qual, o conjunto de nós está particionado em camadas $L_1, \ldots, L_{n+1}$, em que n é $|X|$, a camada L_1 e L_{n+1} possuem apenas um nó, r e t, respectivamente. Para cada nó $u \in L_j$, existe um ou mais arcos $a \in A$ direcionados que o conecta a um nó $u' \in L_{j+1}$. O rótulo do arco (u, u'), por sua vez, representa a atribuição do valor $d(a)$ à variável x_j. Desse modo, qualquer caminho $p = (a^1, \ldots, a^n)$ de r a t, caracteriza uma atribuição as variáveis $x \in X$, ou seja, $x_j = d(a^j)$, para $j = 1, \ldots, n$. Essa atribuição é denotada por x^p. Todos os caminhos de r a t em B formam o conjunto $Sol(\mathscr{P})$.

Exemplo 3. *Considere a seguinte instância do problema da mochila 0 - 1 (Exemplo 1).*

$$\begin{aligned} \textit{Maximizar} \quad & 3x_1 + 4x_2 + 5x_3 + 8x_4 \\ \textit{sujeito à} \quad & 5x_1 + 5x_2 + 6x_3 + 6x_4 \leq 10 \\ & x_j \in \{0,1\}, \quad j = 1, \ldots, 4 \end{aligned}$$

O DD referente a essa instância pode ser visto na Figura 2, neste caso, como $D_j = \{0,1\}$, para $j = 1, \ldots, n$, B é definido como um DD binário, isto é, para cada nó $u \in L_j$, o seu grau de saída é no máximo 2.

Figura 2 – DD binário correspondente ao Exemplo 3



Fonte: Próprio autor.

Nota: Os arcos sólidos representam atribuições com valor 1 a variável x_j, enquanto os arcos pontilhados indicam atribuições com valor 0. Se um arco tem valor diferente de zero (arcos sólidos), seu valor é exibido.

Na Figura 2, observa-se o conjunto $Sol(\mathscr{P}) = \{(0, 0, 0, 0), (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1), (1, 1, 0, 0)\}$ e que cada caminho de r a t na figura 1 codifica uma solução viável da instância. Dessa forma, o caminho mais longo de r a t em B corresponde à solução ótima x^*, como um DD é um grafo acíclico e com pesos positivos nas arestas a solução pode ser obtida em tempo polinomial por meio de algoritmos de caminho mínimo, como Dijkstra (1959), facilmente adaptado para identificar o caminho mais longo, no caso em que $\mathscr{P}$ é um problema de maximização. No exemplo dado, o caminho mais longo é dado por $p = ((r,u_1),(u_1,u_3),(u_3,u_5),(u_5,t))$, com atribuição $x^p = (0,0,0,1)$ e $z^* = 8$.

Um DD possui algumas características importantes para o seu estudo:

- $|L_j|$ é a largura da camada L_j, ou seja, o número de nós presente na camada;
- $max_j\{|L_j|\}$ representa a camada com o maior número de nós, definindo assim a largura do DD;
- $|D(x_j)|$ define o grau máximo de saída que um nó $u \in L_j$ pode assumir.

2.3.1 Programação dinâmica

Programação Dinâmica (PD) é um método recursivo utilizado para resolver problemas de otimização discreta, nos quais os subproblemas de um problema $\mathscr{P}$ se sobrepõem, ou seja, o mesmo subproblema aparece mais de uma vez no espaço de busca. Esse método explora essa característica de sobreposição, armazenando soluções de subproblemas previamente resolvidos para evitar cálculos redundantes, diferente de algoritmos recursivos tradicionais, em que um mesmo subproblema é resolvido cada vez que aparece no espaço de busca.

Para que o problema $\mathscr{P}$ possa ser resolvido por meio de DDs, é necessária uma formulação baseada em PD. Para fins de formalismo, considere que cada subproblema é representado por um estado s. Seja S o espaço dos estados, particionado em $n+1$ estágios, de modo que $S_j = \{\forall s^j\} \cup \{\hat{0}\}$, para $j = 2,\ldots,n$, onde $\hat{0}$ representa um estado inviável. Dessa forma, S_1 contém apenas um estado inicial, enquanto S_{n+1} compreende os k estados terminais. S é particionado da seguinte maneira:

$$S_1 = \{\hat{r}\};$$
$$S_j = \{\forall s^j\} \cup \{\hat{0}\}, \quad j = 2,\ldots,n;$$
$$S_{n+1} = \{\hat{t_1},\ldots,\hat{t_k},\hat{0}\}.$$

A função de transição $t_j : S_j \times D_j \rightarrow S_{j+1}$, para $j = 1,\ldots,n$, é a operação que modifica o estado $s^j \in S_j$ para o estado $s^{j+1} \in S_{j+1}$, essa modificação é controlada por uma

atribuição de um valor $d \in D_j$ à variável x_j. Além disso, a aplicação de t_j sobre um estado inviável resulta em outro estado inviável, ou seja, $t_j(\hat{0}, d) = \hat{0}$ para qualquer $d \in D_j$ e qualquer j. Por outro lado, $h_j : D_j \to \mathbb{R}$, para $j = 1, \ldots, n$, representa a função de custo associada à transição de estados.

Portanto, uma formulação de PD para $\mathscr{P}$ nas variáveis $(s,x) = (s^1, \ldots, s^{n+1}, x_1, \ldots, x_n)$ é definida da seguinte forma, :

$$\begin{aligned} \text{Max} \quad & \hat{f}(s,x) = \sum_{j=1}^{n} h_j(s^j, x_j) \\ & s^{j+1} = t_j(s^j, x_j), \quad \forall x_j \in D_j, \quad j = 1, \ldots, n \\ & s^j \in S_j, \quad j = 1, \ldots, n+1. \end{aligned}$$

Numerar a função objetivo e equaçõ

Exemplo 4. *BERGMAN et al. (2016) apresentam uma formulação de PD para o problema da mochila 0-1 (Exemplo 1):*

$$\begin{aligned} \textit{Max} \quad & \sum_{j=1}^{n} v_j x_j \\ & s^{j+1} = s^j + p_j x_j, \quad j = 1, \ldots, n \\ & s^1 = 0, s^{n+1} \leq C \\ & x_j \in \{0,1\}, \quad j = 1, \ldots, n. \end{aligned}$$

Numerar a função objetivo e equaçõ

Nessa formulação, o estado s^j no estágio j representa o somatório dos pesos dos itens $1, \ldots, j-1$ que foram considerados ou não na alocação. O espaço de estados inicial S_1 contém apenas um estado, $s^1 = 0$, indicando que, inicialmente, nenhum item ainda foi avaliado. A função de transição t_j é formulada do seguinte modo:

$$t_j(s^j, x_j) = \begin{cases} s^j, & se \quad x_j = 0 \\ s^j + p_j, & se \quad x_j = 1 \quad e \quad s^j + p_j x_j \leq C \\ \hat{0}, & se \quad x_j = 1 \quad e \quad s^j + p_j x_j > C \end{cases}$$

Observe que a função de transição aplicada a um estado s^j, no qual a variável de referência x_j é igual a 0, jamais resultará em um estado $\hat{0}$. Por outro lado, ao aplicar a função de transição a um estado s^j, onde a variável de referência x_j é igual a 1, verifica-se que, se a contribuição p_j, isto é, o peso do item j, exceder a capacidade C da mochila, o estado resultante será $\hat{0}$, caso contrário, $s^{j+1} = s^j + p_j$. A função de custo h_j é definida como $h_j(s^j, x_j) = v_j x_j$ neste caso.

No entanto, segundo Hooker (2013) DDs diferem de um método de PD em aspectos significativos. No caso de um DD, o controle de transição é definido exclusivamente pelas variáveis de decisão. Por outro lado, em um método de PD, a transição está associada apenas às variáveis de estado, enquanto variáveis de estado explícitas não estão presentes em um DD. Os custos em um grafo de transição de PD são mais complexos de calcular, pois dependem diretamente do estado. Por outro lado, em DDs, como a dependência ocorre apenas em relação às variáveis de decisão, os custos podem ser computados de forma mais direta, desde que $f(x)$ seja uma função aditivamente separável. Uma função é considerada aditivamente separável quando:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i).$$

2.3.2 Diagramas de decisão exato

Assumindo que existe uma formulação de PD válida (definida em BERGMAN *et al.* (2016)), para $\mathscr{P}$, um diagrama B é considerado exato para $\mathscr{P}$ se, para cada nó $u \in L_j$, existir uma correspondência com um estado $s \in S_j$, ou seja, todos os estados viáveis da formulação PD de $\mathscr{P}$ devem ser representados em B. Sendo assim, todos os caminhos caminhos de r a t codificam uma solução viável em $\mathscr{P}$. Os estados inviáveis $\hat{0}$ não precisam ser representados em B, pois $t_j(\hat{0},d) = \hat{0}$ para qualquer $d \in D$, o que permite que sejam omitidos sem prejuízo.

O Algoritmo 1 apresenta o procedimento para a construção de um DD exato para um problema $\mathscr{P}$. A primeira camada, L_1, contém apenas um nó, correspondente ao estado inicial definido na formulação de PD. A partir desse ponto, o algoritmo percorre as n variáveis de decisão, construindo iterativamente as $n+1$ camadas de B. A construção da camada L_{j+1} baseia-se na camada L_j. Sempre que uma transição válida ocorre em um nó $u \in L_j$, um novo nó $u' \in L_{j+1}$ é criado, além disso, um arco entre u e u' é adicionado $(b_d(u) \leftarrow u')$ e o custo desse arco $(v(u,u') \leftarrow h_j(u,d))$. Os k nós da camada L_{n+1} são combinados em um único nó, t.

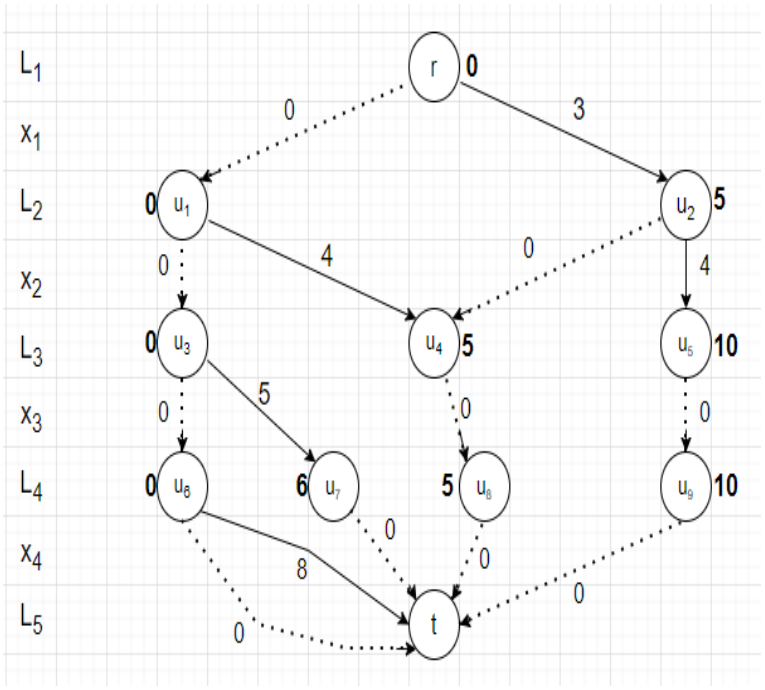
Com isso, cada nó $u \in B$ é equivalente a um estado $s \in S$ na formulação de PD. Observe que o Algoritmo 1 é responsável por construir o conjunto $Sol(\mathscr{P})$. Contudo, dependendo da complexidade do problema $\mathscr{P}$, o número de nós/estados em B pode crescer exponencialmente em relação ao número de variáveis de decisão, tornando a construção computacionalmente custosa.

Algoritmo 1: *compila_dd_exato*

```
1:  Inicialize L_1 ← {r}
2:  Para j = 1 até n faça
3:      L_{j+1} ← ∅
4:      Para u ∈ L_j faça
5:          Para d ∈ D_j faça
6:              Se t_j(u,d) ≠ 0̂ então
7:                  u' ← t_j(u,d)
8:                  Se ∄u ∈ L_{j+1} | u = u' então
9:                      L_{j+1} ← L_{j+1} ∪ {u'}
10:                     b_d(u) ← u'
11:                     v(u,u') = h_j(u,d)
12:                 senão
13:                     b_d(u) ← u'
14:                     v(u,u') = h_j(u,d)
15:                 fim se
16:             fim se
17:         fim para
18:     fim para
19: fim para
20: t ← Junção de todos os nós da camada L_{n+1}
21: L_{n+1} ← {t}
22: Retorne B
```

Exemplo 5. *Considere a instância do problema da mochila 0-1 apresentada no Exemplo 3. O DD exato associado a essa instância é ilustrado na Figura 3.*

Figura 3 – DD exato correspondente ao Exemplo 3



Fonte: Próprio autor.

2.3.3 Diagramas de decisão restritos

Um DD exato pode crescer exponencialmente em relação ao tamanho de sua entrada. Uma abordagem alternativa para encontrar soluções viáveis em $\mathscr{P}$ é a construção de DDs restritos. Para isso, o tamanho das camadas em B são limitadas por um valor W. B é considerado um DD restrito quando:

$$\begin{aligned} &Sol(B) \subseteq Sol(\mathscr{P}) \\ &f(x^p) \geq v(p), \text{ para todo caminho } p \text{ de } r \text{ a } t \text{ em } B, \text{ com } x^p \in Sol(\mathscr{P}). \end{aligned} \tag{2.2}$$

Um DD restrito pode ser comparado a heurísticas utilizadas em métodos de PLI, pois tem como objetivo fornecer soluções viáveis para o problema. Se $\mathscr{P}$ for um problema de maximização, as soluções viáveis obtidas definirão um limite inferior para o valor ótimo. Por outro lado, se $\mathscr{P}$ for um problema de minimização, as soluções viáveis encontradas definirão um limite superior. A construção de um DD restrito é relativamente simples, envolvendo apenas uma modificação no Algoritmo 1 e o acréscimo do parâmetro W. Após a linha 18 do Algoritmo 1, ainda dentro do primeiro laço "para", deve-se inserir o seguinte trecho (o restante do algoritmo é o mesmo do Algoritmo 1):

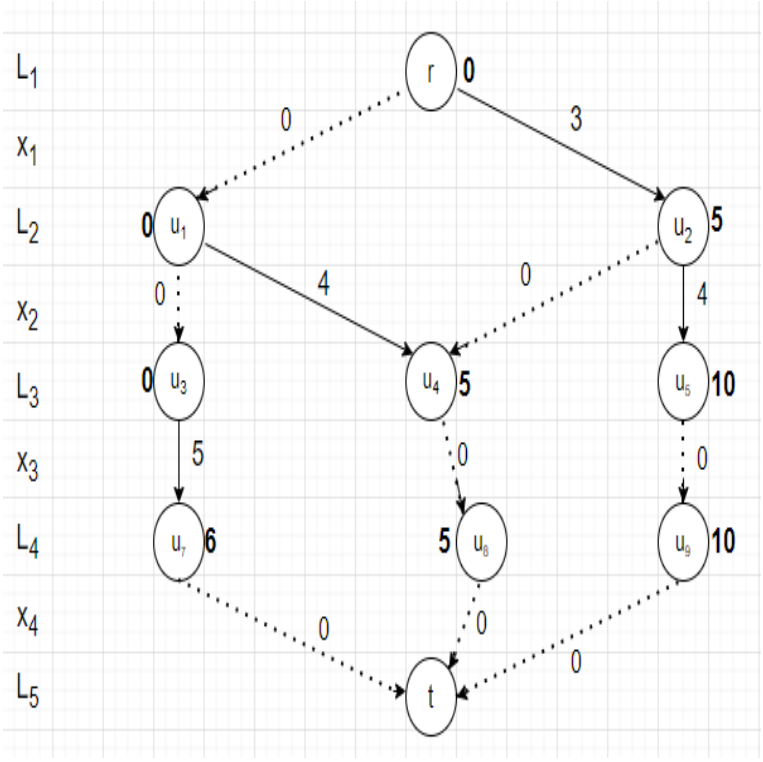
Algoritmo 2: *compila_dd_restrito*

1: **Se** $|L_{j+1}| > W$ **então**
2: **Enquanto** $|L_{j+1}| > W$ **faça**
3: M $\leftarrow$ *seleciona_nos*(L_{j+1})
4: $L_{j+1} = L_{j+1} \backslash M$
5: **fim enquanto**
6: **fim se**

De acordo com Bergman *et al.* (2014), a escolha dos nós a serem removidos de uma camada pode impactar significativamente a qualidade dos limites obtidos. Isso ocorre porque um nó pertencente à solução ótima x^* pode ser inadvertidamente removido ou nós que codificam soluções próximas do ótimo. Dessa forma, os principais fatores que influenciam a qualidade das soluções obtidas, além da complexidade do problema $\mathscr{P}$, são o parâmetro W, que determina o grau de remoção de nós em uma camada, e a heurística para seleção dos nós a serem removidos, representada no Algoritmo 2 pela função *seleciona_nos*. Este último, em particular, desempenha um papel crucial na definição da qualidade dos limites obtidos.

Exemplo 6. *Considere a instância do problema da mochila 0-1 apresentada no Exemplo 3. No Exemplo 5, foi construído um DD exato para essa instância. Neste exemplo, será construído um DD restrito para a mesma instância, adotando-se um tamanho máximo de camada $W = 3$, os nós a serem removidos foram selecionados com base na ordem de criação.*

Figura 4 – DD restrito correspondente ao Exemplo 5



Fonte: Próprio autor.

No Exemplo 6, observa-se que o nó u_6 foi removido devido ao critério adotado para a exclusão de nós. Como consequência, a solução ótima x^*, com a atribuição $x^p = (0,0,0,1)$ e valor ótimo $z^* = 8$, deixou de estar codificada em B. Assim, o caminho $p = ((r,u_2),(u_2,u_5),(u_5,u_9),(u_9,t))$, que possui $v(p) = 7$ e corresponde à atribuição $x^p = (1,1,0,0)$ será a solução viável retornada caso seja aplicado um algoritmo de caminho mais longo.

2.3.4 Diagramas de decisão relaxados

Na Seção 2.3.3, foi apresentado que os DDs restritos fornecem soluções viáveis para $\mathscr{P}$. Quando $\mathscr{P}$ é um problema de maximização, os DDs restritos fornecem um limite inferior. Por outro lado, se $\mathscr{P}$ for um problema de minimização, eles fornecem um limite superior. Portanto, neste caso, $Sol(B) \subseteq Sol(\mathscr{P})$. DDs relaxados vão partir de uma premissa oposta.

B é considerado um DD relaxado quando representa um superconjunto das soluções viáveis de $\mathscr{P}$. Isso significa que B inclui todas as soluções viáveis de $\mathscr{P}$, além de soluções inviáveis. Como consequência, se $\mathscr{P}$ for um problema de maximização, B fornecerá limites superiores. Por outro lado, se $\mathscr{P}$ for um problema de minimização, B retornará limites inferiores. Formalmente, B será considerado um DD relaxado quando:

$$\begin{aligned} &Sol(B) \supseteq Sol(\mathscr{P}) \\ &f(x^p) \leq v(p), \text{ para todo caminho } p \text{ de } r \text{ a } t \text{ em } B, \text{ com } x^p \in Sol(\mathscr{P}). \end{aligned} \tag{2.3}$$

DDs relaxados podem ser comparados a uma relaxação linear em métodos de PLI. O processo de construção de um DD relaxado é semelhante ao de um DD restrito, utilizando como base o Algoritmo 1, que é responsável pela criação de um DD exato. As camadas de B serão limitadas por um valor W: se a camada $|L_j|$ exceder W, então $|L_j| - W + 1$ nós serão selecionados para serem mesclados em um único nó, os estados dos nós mesclados será descrito por meio de um operador $\oplus(M)$ válido. De acordo com BERGMAN *et al.* (2016), o operador $\oplus(M)$ é válido quando as duas condições da Equação 2.3 são satisfeitas. O comprimento v de cada arco que entra em cada nó u que foi escolhido para a mesclagem é modificado para $\Gamma_M(v,u)$. No Algoritmo 1 após a linha 18, ainda dentro do primeiro laço "para", deve-se inserir o seguinte trecho (o restante do algoritmo é o mesmo do Algoritmo 1) para compilar um DD relaxado:

Algoritmo 3: *compila_dd_relaxado*

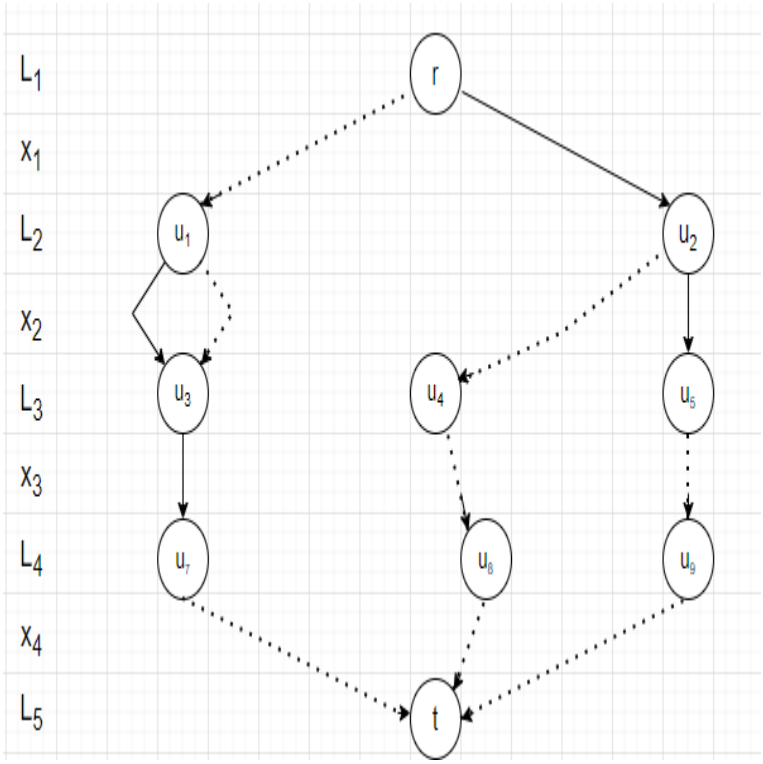
1: **Se** $|L_{j+1}| > W$ **então**
2: M $\leftarrow$ *seleciona_nos_mesclagem*(L_{j+1})
3: $u' \leftarrow \oplus M$
4: **Para** $u \in L_j$ **faça**
5: **Para** $i \in D_j \mid b_i(u) \in M$ **faça**
6: $v(a_i(u)) \leftarrow \Gamma_M(v(a_i(u)), b_i(u))$
7: $b_i(u) \leftarrow u'$
8: **fim para**
9: **fim para**
10: $L_{j+1} \leftarrow L_{j+1} \backslash \{M\}$
11: $L_{j+1} \leftarrow L_{j+1} \cup \{u'\}$
12: **fim se**

A heurística responsável pela seleção dos nós para a mesclagem é fundamental para a qualidade dos limites encontrados. É importante observar que uma mesclagem específica

adiciona a B um estado inviável. No entanto, esse estado não será descartado, uma vez que o objetivo é obter uma relaxação para $\mathscr{P}$, resultando em um limite superior ou inferior, dependendo se $\mathscr{P}$ é um problema de maximização ou minimização, respectivamente.

Exemplo 7. *Dada uma formulação de PD para o problema $\mathscr{P}$, é possível construir um DD relaxado utilizando o Algoritmo 3. A Figura 5 ilustra o DD relaxado gerado, assumindo que o tamanho máximo de camada W seja igual a 3. Suponha que o nó u_3 resulta de uma operação de mesclagem válida $\oplus M$.*

Figura 5 – DD relaxado correspondente a um determinado problema $\mathscr{P}$



Fonte: Próprio autor.

2.3.5 Ordenação de variáveis

Como apresentado na Seção 2.3.1, as transições em um DD são baseadas exclusivamente nas variáveis de decisão de $\mathscr{P}$, mais precisamente em seus respectivos domínios. Dessa forma, não é obrigatório seguir a ordem natural das variáveis durante a construção de B nos algoritmos 1, 2 e 3. A escolha da ordenação das variáveis influencia drasticamente os tamanhos das camadas em um DD. Escolher a ordenação que minimiza o tamanho de DD é considerado um problema NP-difícil. De acordo com Cappart *et al.* (2019), a qualidade dos limites obtidos pelos DD relaxado e restrito depende significativamente da ordenação de variáveis escolhida.

2.4 Branch-and-Bound baseado em diagramas de decisão

O método clássico de BAB, amplamente utilizado em PLI, enumera implicitamente todas as soluções possíveis de $\mathscr{P}$, estruturando-as em subproblemas organizados em uma árvore. A enumeração das soluções é realizada de maneira estratégica. Para um dado subproblema, determinam-se limites superior e inferior. Supondo que $\mathscr{P}$ seja um problema de maximização, o limite superior é obtido por meio de uma relaxação linear, enquanto o limite inferior é calculado utilizando uma heurística ou outro método capaz de encontrar uma solução viável. Se o limite superior de um subproblema for inferior à melhor solução viável encontrada na árvore até o momento, esse subproblema é descartado, pois os subproblemas gerados por ele não podem conter a solução ótima. A ramificação dos subproblemas é determinada pela relaxação linear. De acordo com Huang *et al.* (2021), diversas decisões impactam diretamente a eficiência do método BAB, como a escolha das variáveis fracionárias para ramificação, a definição de planos de cortes e as regras de poda aplicadas durante o processo.

Como foi visto anteriormente, cada nó em um DD representa um subproblema de $\mathscr{P}$. O método BAB baseado em DD segue os mesmos princípios do BAB tradicional, os limites superior e inferior são obtidos diretamente por meio do DD relaxado e do DD restrito, respectivamente. Diferentemente do método tradicional de BAB, que realiza a ramificação com base nas variáveis de decisão fracionárias, o método de BAB baseado em DDs executa a ramificação em subproblemas ou nós não mesclados do DD relaxado.

2.4.1 Corte exato

Seja $\overline{B}$ o DD relaxado retornado pelo Algoritmo 3. Um nó $\overline{u} \in \overline{B}$ é considerado exato se, para todo caminho $r - \overline{u}$ em $\overline{B}$, o nó $\overline{u}$ corresponder exclusivamente a um único estado s^j, ou seja, $\overline{u}$ não pode ter sido gerado a partir de um determinado nó que tenha sofrido o processo de mesclagem ainda no Algoritmo 3. Um conjunto S de nós pertencentes a $\overline{B}$ será chamado de conjunto de corte exato se todos os nós em S forem nós exatos em $\overline{B}$ e todo caminho p de r a t passa por algum nó pertencente a S.

Considere B um DD exato,

~~B é um DD exato,~~ seja $v^*(B_{uu'})$ o caminho mais longo entre os nós $u - u'$ em $B_{uu'}$. Para um nó $u \in B$, $P|_u$ é as restrições em $\mathscr{P}$ cujas soluções viáveis correspondem as caminhos de r a t em B que incluem u.

Lema 2.4.1. *De acordo BERGMAN et al. (2016), se B for um DD exato para $\mathscr{P}$, então, para*

qualquer nó u pertencente a B, a seguinte afirmação é válida:

$$v^*(B_{ru}) + v^*(B_{ut}) = z^*(P|_u)$$

Melhorar a escrita da demonstração!

Demonstração. $z^*(P|_u)$ é o comprimento do maior caminho *r–t* em B que contém u, e qualquer caminho desse tipo possui comprimento $v^*(B_{ru}) + v^*(B_{ut})$. □

Teorema 2.4.1. *De acordo com BERGMAN et al. (2016), se B for um DD relaxado construído utilizando um modelo válido de PD para o problema $\mathscr{P}$, e S for um conjunto de corte exato de B, então:*

$$z^*(\mathscr{P}) = \max_{u \in S}\{z^*(\mathscr{P}|_u)\}$$

Demonstração. Seja B o DD exato para $\mathscr{P}$ criado utilizando o mesmo modelo de PD. Como cada nó $\overline{u} \in S$ é exato, ele possui um nó correspondente u em B (isto é, um nó associado ao mesmo estado), e S é um conjunto de corte de B. Assim,

$$z^*(\mathscr{P}) = \max_{u \in S}\{v^*(B_{ru}) + v^*(B_{ut})\} = \max_{u \in S}\{z^*(\mathscr{P}|_u)\}.$$

□

A seleção de nós para formar o conjunto de corte exato S desempenha um papel crucial na eficiência do algoritmo. Conforme discutido por BERGMAN *et al.* (2016), três abordagens principais são sugeridas para determinar os nós que compõem S.

- Ramificação Tradicional (RT): O método mais simples para a seleção de nós exatos ocorre quando $S = L_2$. Como todos os nós da camada L_2 são descendentes do nó inicial r, cada nó u em L_2 representa exatamente um estado s^j, o que implica que u será um nó exato;
- Última Camada Exata (UCE): Este método seleciona os nós da última camada de $\overline{B}$ onde não houve mesclagem de nós, e os utiliza para compor o conjunto S.
- Corte de Fronteira (CF): Esse método seleciona os nós para compor S seguindo a seguinte abordagem:

$$\text{CF}(\overline{B}) = \{u \in \overline{B} \mid u \text{ é exato e } b_0(u) \text{ ou } b_1(u) \text{ é relaxado}\}.$$

Exemplo 8. *Considere o DD relaxado ilustrado na Figura 5. Utilizando o método de seleção de nós RT, temos que $S = \{u_1, u_2\}$. Caso o método aplicado fosse o UCE, o conjunto S ainda seria formado por u_1 e u_2, uma vez que houve uma mesclagem de nós na camada L_3, identificando L_2 como a última camada exata. Por outro lado, ao adotar o método CF, o conjunto S seria constituído por u_1, u_8 e u_9.*

2.4.2 Pseudocódigo do método branch-and-bound baseado em diagramas de decisão

O Algoritmo 4 mostra o processo para se obter a solução ótima para um determinado problema $\mathscr{P}$ a partir do método de BAB baseado em DDs.

Algoritmo 4: *BAB_DD*

Inicialize $Q \leftarrow \{r\}$
$z_{opt} \leftarrow -\infty$
$v^*(r) \leftarrow 0$
Enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow seleciona\_no(Q)$
  $Q \leftarrow Q \backslash \{u\}$
  Crie um DD relaxado $\overline{B}_{ut}$ seguindo o Algoritmo 3, $v_r \leftarrow v^*(u)$
  Se $v^*(\overline{B}_{ut}) > z_{opt}$ **então**
    Seja S o corte exato de $\overline{B}_{ut}$
    Para $u' \in S$ **faça**
      $v^*(u') \leftarrow v^*(u) + v^*(\overline{B}_{uu'})$
      $Q \leftarrow Q \cup \{u'\}$
    fim para
    Crie um DD restrito B'_{ut} seguindo o Algoritmo 2, $v_r = v^*(u)$
    Se $v^*(B'_{ut}) > z_{opt}$ **então**
      $z_{opt} \leftarrow v^*(B'_{ut})$
    fim se
  fim se
fim enquanto
Retorne z_{opt}

2.5 Resolução do PCIPM via diagramas de decisão

Considere o PCIPM descrito na Seção 2.2. Para resolver o PCIPM utilizando o método de BAB baseado em DD, é fundamental definir previamente elementos como uma formulação de PD e uma regra de mesclagem de nós que seja válida. Nas próximas seções, será dado um enfoque especial ao PCIPM, porém, todas as definições apresentadas para o PCIPM também são aplicáveis ao PCIM.

2.5.1 Formulação de programação dinâmica para o PCIPM

Para que um problema $\mathscr{P}$ possa ser resolvido utilizando DDs, como discutido na Seção 2.3.1, é necessário a definição de uma formulação de PD. Autores como Bergman *et al.* (2014) e BERGMAN *et al.* (2016) apresentam uma formulação válida para abordar o PCIPM.

Um aspecto fundamental na formulação de PD é a definição de estado. Para o PCIPM, o estado s^j representa o conjunto de vértices que ainda não foram incluídos em um conjunto independente. Como consequência, o estado inicial $\hat{r}$, pertencente ao espaço de estados S_1, será igual ao conjunto de vértices V, pois, nesse estágio, nenhum vértice foi ainda atribuído a um conjunto independente.

A próxima definição da formulação de PD é a função de transição t_j. Considere $N(j) = \{j' | (j,j') \in E\}$ a vizinhança do vértice j. A função de transição aplicada a um estado s^j no estágio j é controlada pela variável de decisão $x_j \in \{0,1\}$. Se $x_j = 0$, o estado s^{j+1}, pertencente ao estágio $j+1$, é definido como $s^j \setminus \{j\}$, pois a decisão de não incluir o vértice j resulta apenas na sua exclusão do estado s^j. Neste caso, a função de custo de transição é $h_j = 0$. Se $x_j = 1$, o estado s^{j+1} será dado por $s^j \setminus (N(j) \cup \{j\})$, pois a decisão de incluir o vértice j resulta em sua exclusão do estado s^j. Além disso, de acordo com a definição do PCIPM, um conjunto independente é formado por vértices não adjacentes. Dessa forma, a vizinhança $N(j)$ do vértice j também deve ser excluída para garantir que s^{j+1} não contenha nenhum nó vizinho de um nó que já fez parte da solução parcial associada a s^j, para este caso, a função de custo de transição é $h_j = w_j$. Note que se $x_j = 1$ e $j \notin s^j$ o estado $\hat{0}$ será gerado. Portanto, a função de transição para o PCIPM é denotada da seguinte maneira:

$$t_j(s^j, x_j) = \begin{cases} s^j \setminus \{j\}, & \text{se} \quad x_j = 0 \\ s^j \setminus (N(j) \cup \{j\}), & \text{se} \quad x_j = 1 \quad \text{e} \quad j \in s^j \\ \hat{0}, & \text{se} \quad x_j = 1 \quad \text{e} \quad j \notin s^j. \end{cases}$$

A função de transição t_j, aplicada a um estado s^j pertencente ao estágio j, é controlada pela variável de decisão $x_j \in \{0,1\}$. Em ambos os casos ($x_j = 0$ ou $x_j = 1$), o vértice j será excluído do novo estado gerado, s^{j+1}, pertencente ao estágio $j+1$. Como consequência, todos os estados no estágio $n+1$ serão iguais ao conjunto vazio $\emptyset$. Dessa forma, o espaço de estados

do PCIPM é definido como:

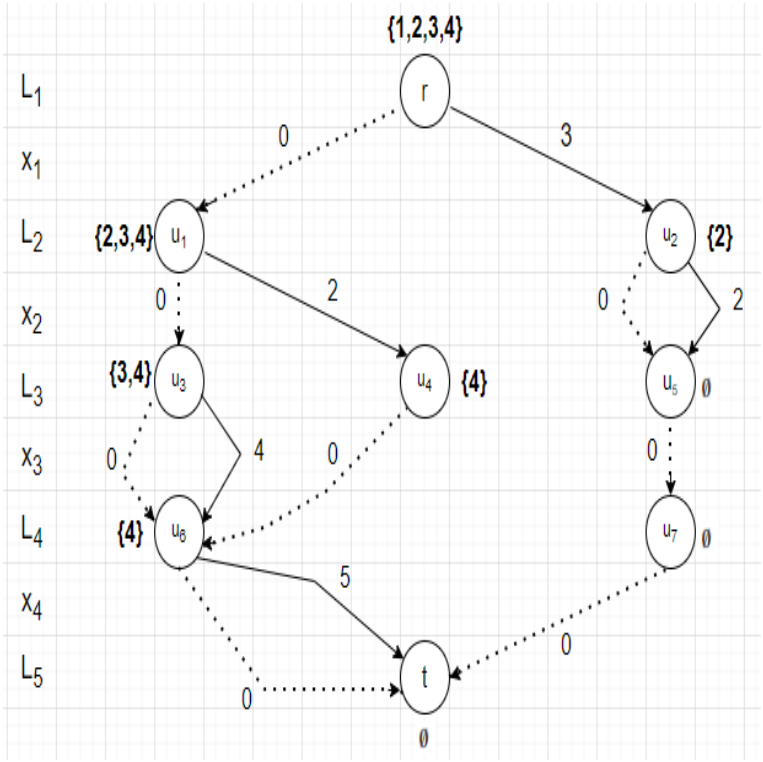
$$S_1 = \{V\};$$
$$S_j = \{2^{v_j}\}, \quad j = 2, \ldots, n;$$
$$S_{n+1} = \{\emptyset\}.$$

2.5.2 *Diagrama de decisão exato para o PCIPM*

Uma vez definida uma formulação válida de PD para o PCIPM, é possível, a partir do Algoritmo 1, construir um DD exato.

Exemplo 9. *Considere a instância do PCIPM descrita no Exemplo 2. A Figura 6 ilustra o DD exato B, caso o Algoritmo 1 fosse aplicado a essa instância.*

Figura 6 – DD exato correspondente a instância do PCIPM do Exemplo 2



Fonte: Próprio autor.

2.5.3 *Operação de mesclagem válida para o PCIPM*

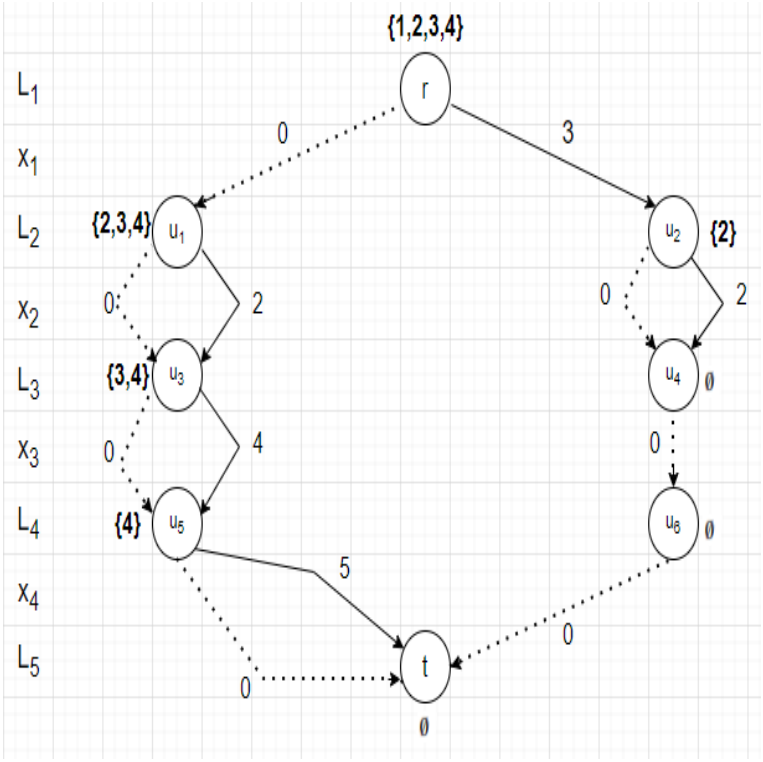
A Seção 2.5.2 exemplificou um DD exato para uma instância do PCIPM. Para a construção de um DD exato para um determinado problema $\mathscr{P}$ só é necessário uma formulação

de PD. Por outro lado, para construir um DD relaxado, além de uma formulação de PD válida para $\mathscr{P}$, é necessário uma operação de mesclagem $\oplus M$ válida de nós.

BERGMAN *et al.* (2016) apresentam um operador de mesclagem válido para o PCIPM, no qual os estados dos nós selecionados para a mesclagem são simplesmente unidos. Especificamente, se $M = \{u_i \mid i \in I\}$, o estado resultante dos nós mesclados será dado por $\oplus(M) = \bigcup_{i \in I} u_i$. O custo de transição permanece inalterado, $\Gamma_M(v,u) = v$. Como a inviabilidade ocorre apenas quando um vértice $j \notin s^j$ e $x_j = 1$. Isso assegura que nenhuma solução viável seja perdida, validando a operação de mesclagem.

Exemplo 10. *O Exemplo 9 apresentou um DD exato para a instância do PCIPM descrita no Exemplo 2. Suponha que o Algoritmo 3 seja aplicado a essa instância, com o tamanho máximo de camada $W = 2$ e a seleção de nós para mesclagem seguindo a ordem de criação na camada L_j de referência. A Figura 7 ilustra o DD relaxado gerado pelo Algoritmo 3.*

Figura 7 – DD relaxado correspondente a instância do PCIPM do Exemplo 2.



Fonte: Próprio autor.

Observe que a operação de mesclagem resulta em um limite superior com valor 11, codificado pela atribuição $v^p = (0,1,1,1)$. Contudo, essa configuração representa uma solução inviável para a instância considerada, pois existe uma aresta entre os vértices 2 e 3. Contudo, observa-se na Figura 6 que, caso os nós u_4 e u_5 fossem mesclados em vez de u_3 e u_4,

o caminho correspondente à solução ótima $p = (r, u_1, u_3, u_6, t)$*, com atribuição* $v^p = (0,0,1,1)$ *seria preservado.*

3 METODOLOGIA

Neste capítulo, detalha-se a escolha da metodologia adotada, incluindo as heurísticas de exclusão de nós no DD restrito, heurísticas de seleção de nós para a mesclagem no DD relaxado e as estratégias de ordenação de variáveis empregadas. No capítulo anterior, verificou-se a influência dessas heurísticas nos DDs, bem como o impacto dessas estratégias na qualidade dos limites. Além disso, este capítulo apresenta as instâncias do PCIPM utilizadas e o ambiente computacional empregado nas análises. O corte exato selecionado para os experimentos foi a UCE, que considera a última camada de nós no DD relaxado onde não houve mesclagem de nós.

3.1 Heurísticas de exclusão de nós utilizadas

A escolha dos nós a serem excluídos no DD restrito pode impactar diretamente a qualidade dos limites retornados, uma vez que soluções parciais ótimas ou promissoras podem deixar de ser representadas devido à exclusão de determinados nós. Neste trabalho, foram empregadas três heurísticas para a exclusão de nós:

- **Pior Valor de Caminho (PVC)**: esta heurística exclui, em uma determinada camada L_j do DD restrito, os $|L_j| - W$ nós que possuem os piores valores de caminho mais longo da origem r até si mesmos;
- **Melhor Valor de Caminho (MVC)**: esta heurística exclui, em uma determinada camada L_j do DD restrito, os $|L_j| - W$ nós que apresentam os melhores valores de caminho mais longo da origem r até si mesmos;
- **Remoção Aleatória (RA)**: esta heurística exclui, em uma determinada camada L_j do DD restrito, $|L_j| - W$ nós de maneira aleatória.

As duas primeiras heurísticas fazem uso do artifício da ordenação. Geralmente, o pior caso de um algoritmo de ordenação apresenta complexidade de $O(n^2)$, onde n é o tamanho da entrada. No contexto dessas heurísticas, em que o tamanho da entrada corresponde ao limite de camada W, a complexidade no pior caso é dada por $O((2W)^2)$ por camada, pois $|L_{j-1}| \leq W$ e $|D_j| = 2$. Por se tratar de uma escolha aleatória simples, a heurística RA apresenta uma complexidade de tempo constante, denotada por $\Theta(1)$.

3.2 Heurísticas de seleção de nós utilizadas

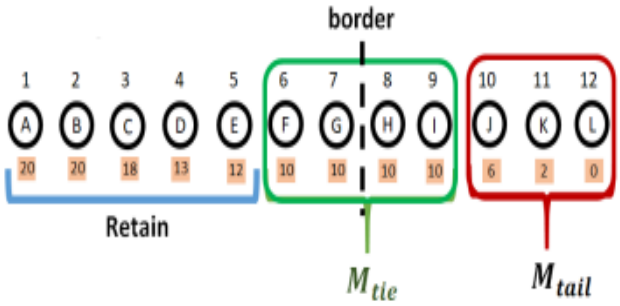
A seleção de nós para a operação de mesclagem no DD relaxado pode exercer influência direta na qualidade do limite obtido. Nesta pesquisa, foram empregadas três heurísticas para a seleção de nós. As duas primeiras foram propostas por BERGMAN *et al.* (2016), enquanto a última, foi introduzida por Nafar e Römer (2024).

- **MinLP**: Ordena-se L_j em ordem crescente pelo valor do caminho mais longo da origem r até a si mesmos e mesclam-se os $|L_j| - W + 1$ primeiros nós da camada. A lógica por trás dessa heurística é que a inviabilidade é introduzida no DD durante a mesclagem de nós, e aqueles com valores de caminhos mais longos menos promissores têm menores chances de contribuir para soluções ótimas.
- **MinTam**: Ordena-se L_j em ordem decrescente pelo tamanho dos estados dos nós e mesclam-se os $|L_j| - W + 1$ primeiros nós da camada. Como conjuntos maiores de vértices geralmente possuem mais elementos em comum, essa heurística prioriza a mesclagem de nós que representam regiões similares do espaço de soluções.
- **Grupo de Borda (GB)**: Ordena-se L_j em ordem crescente pelo valor do caminho mais longo. Após a ordenação, verifica-se se os nós nas posições W e $W-1$ da camada L_j possuem o mesmo valor de caminho mais longo. Caso positivo, identifica-se se há outros nós na borda com o mesmo valor de caminho mais longo. Todos esses nós serão então mesclados em um único nó, denominado M_{tie}[1]. Os últimos nós da camada L_j que possuem valores de caminho mais longo inferiores aos nós em M_{tie} também serão mesclados em um único nó, denominado M_{tail}. Caso os nós nas posições W e $W-1$ possuam valores de caminho mais longo distintos, a heurística MinLP será utilizada em substituição à GB. A ideia central dessa heurística é controlar o erro decorrente da superestimativa do comprimento da solução parcial, causado pelo redirecionamento dos arcos de entrada durante a operação de mesclagem. Ao mesclar nós com o mesmo valor de caminho mais longo, torna-se possível mitigar esse erro em um dos lados do DD.

Assim como nas heurísticas de exclusão de nós no DD restrito, as heurísticas de seleção de nós para mesclagem também utilizam o conceito de ordenação. Portanto, em geral, no pior caso, sua complexidade será novamente $O((2W)^2)$ por camada, onde W representa o tamanho máximo que cada camada pode assumir.

[1] "tie"vem do inglês para "empate".

Figura 8 – Exemplo da heurística de seleção de nós GB.



Fonte: Figura retirada de (NAFAR; RÖMER, 2024)
Nota: Os nós em "Retain" permanecerão inalterados na camada, enquanto os grupos de nós M_{tie} e M_{tail} serão submetidos ao processo de mesclagem.

3.3 Estratégias de ordenação de variáveis utilizadas

Outro fator de grande relevância para a qualidade dos limites obtidos, tanto no DD restrito quanto no DD relaxado, é a ordenação das variáveis utilizada na construção dos DDs. Para a ordenação de variáveis, foram consideradas três abordagens distintas nos experimentos, a segunda ordenação foi proposta por BERGMAN *et al.* (2016) e a última foi introduzida por Nafar e Römer (2024):

- **Ordenação Natural (ON)**: a construção dos DDs segue a ordenação normal das variáveis, dessa forma, essa seleção apresenta uma complexidade constante, $\Theta(1)$.;
- **Mínimo Número de Estados (MNE)**: O próximo vértice ou variável a ser selecionado é aquele que aparece no menor número de estados na camada atual. Com isso, no pior caso, essa seleção tem complexidade $O(2W|V|)$ por camada. A lógica por trás dessa ordenação é que, se um vértice não pertence a nenhum estado na camada anterior, torna-se impossível adicionar um arco um, isso resulta em menos decisões viáveis a serem tomadas.
- **Soma dos Graus Atuais (SGA)**: cada estado na camada atual pode ser interpretado como um subgrafo induzido. O próximo vértice ou variável a ser escolhido é aquele que apresenta a menor soma dos graus nos subgrafos induzidos. Portanto, essa seleção no pior caso tem como complexidade $O(2W|V|^2)$ por camada. A ideia por trás dessa heurística é que vértices com menores graus tendem a participar de um maior número de conjuntos independentes. Assim, decidir antecipadamente sobre esses vértices pode levar a uma exploração do espaço de busca mais próxima da solução ótima.

3.4 Ambiente computacional e instâncias

Para a realização dos experimentos, utilizou-se a plataforma *Google Colab*, que disponibiliza um servidor equipado com um processador Intel(R) Xeon(R) CPU @ 2,20GHz, memória RAM de 12,7GB e arquitetura de 64 bits. Os códigos utilizados nos experimentos foram implementados na linguagem de programação *Python*. A plataforma utiliza a versão *Python 3.10.12*, e a geração de gráficos foi realizada com o auxílio da biblioteca *Matplotlib 3.8.0*.

As instâncias do PCIPM foram geradas aleatoriamente, considerando diferentes níveis de densidade. Como ambos os problemas envolvem grafos não direcionados, a densidade é calculada utilizando a Equação 3.1:

$$Densidade = \frac{2 \cdot |E|}{|V| \cdot |V-1|}. \tag{3.1}$$

Foram geradas 10 instâncias para o PCIPM. As instâncias foram inicialmente resolvidas utilizando o *solver* de código aberto *OR-Tools*. Este *solver* não possui restrições quanto ao tamanho dos problemas, ao contrário de *solvers* comerciais como o *CPLEX* e o *Gurobi* que limitam o tamanho dos problemas nas suas versões gratuitas. No entanto, para instâncias maiores, o *OR-Tools* apresenta tempos de solução mais elevados em comparação com essas opções comerciais.

A Tabela 1 apresenta as informações das instâncias geradas para o PCIPM. Ela especifica o número de vértices, o número de arestas, a densidade, a solução ótima encontrada pelo *solver OR-Tools*, e o tempo, em segundos, necessário para encontrar cada solução.

Tabela 1 – Especificações das instâncias do PCIPM.

ID	n	m	Densidade	Solução	Tempo (s)
Instância 1	85	357	0,1	2882	0,44
Instância 2	117	1357	0,2	2883	35,49
Instância 3	106	1669	0,3	2179	16,3
Instância 4	107	2268	0,4	1568	15,84
Instância 5	102	2575	0,5	1348	2,70
Instância 6	88	2296	0,6	846	1,65
Instância 7	89	2741	0,7	704	3,74
Instância 8	115	5243	0,8	762	11,33
Instância 9	80	2844	0,9	385	1,11
Instância 10	89	3915	1,0	176	0,28

Fonte: Elaborada pelo autor.

Nota: A coluna n indica a quantidade de vértices, enquanto a coluna m representa o número de arestas.

4 ANÁLISE E RESULTADOS

Uma série de análises foram conduzidas com as heurísticas descritas no capítulo anterior. Neste capítulo, são apresentados os resultados das combinações entre heurísticas de exclusão de nós e ordenações de variáveis no DD restrito. Além disso, são exibidos os resultados das combinações entre heurísticas de seleção de nós para a mesclagem e ordenações de variáveis no DD relaxado, bem como a análise das combinações que integram heurísticas de exclusão de nós, seleção de nós e ordenação de variáveis no BAB baseado em DDs. Tudo isso aplicado ao PCIPM. A nomenclatura adotada para as heurísticas e ordenações de variáveis segue a mesma convenção apresentada no Capítulo 3. Como exemplo, a sigla SGA representa a ordenação de variáveis com base na Soma dos Graus Atuais.

4.1 Resultados das heurísticas de exclusão de nós no DD restrito

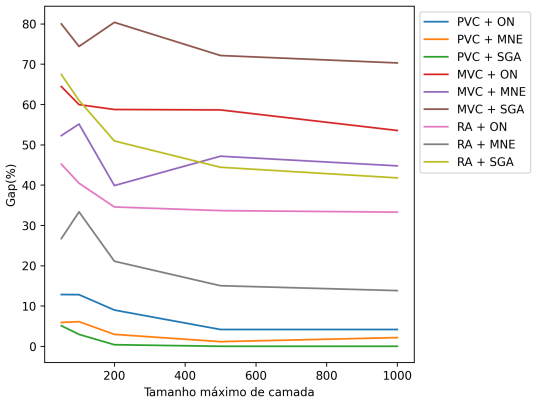
A escolha da heurística de exclusão de nós no DD restrito desempenha um papel crucial na qualidade dos limites inferiores obtidos. Além da análise individual dessas heurísticas, outras duas decisões relevantes são consideradas: o limite W para o tamanho das camadas e a ordenação das variáveis. Nos experimentos, todas as combinações possíveis entre as ordenações de variáveis e as heurísticas de exclusão de nós serão avaliadas. Considerando que existem 3 heurísticas de exclusão de nós e 3 tipos de ordenação de variáveis, é possível formar um total de 9 combinações distintas. Para o parâmetro W, serão testados os valores $W \in \{50, 100, 200, 500, 1000\}$.

A primeira análise avalia a distância dos limites inferiores em relação à solução ótima. Para isso, utilizou-se a Equação 4.1, que calcula o *gap* e representa, em porcentagem, a diferença entre os limites inferiores e a solução ótima da instância. Em seguida, será feito uma análise sobre tempo de execução dessa heurísticas.

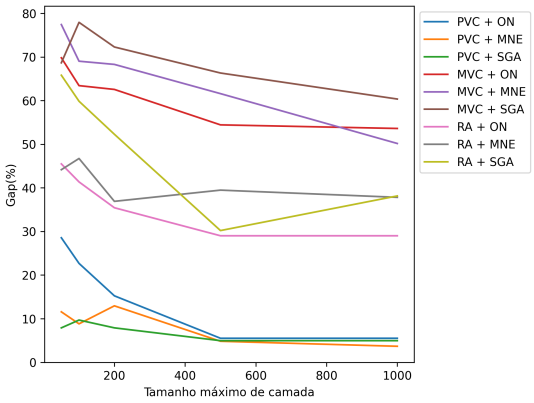
$$gap = \left| \frac{limite - Sol(x^*)}{Sol(x^*)} \right| \cdot 100 \tag{4.1}$$

A Figura 9 ilustra o *gap* obtido com essas combinações em relação à variação do parâmetro W. Das 10 instâncias testadas, a Figura 9 apresenta os resultados de 5 instâncias selecionadas, representando diferentes valores de densidade. Essas instâncias foram escolhidas para evidenciar o impacto das combinações propostas em cenários com características distintas. O objetivo é analisar como a variação da densidade e a variação do parâmetro W influenciam o desempenho das heurísticas e a eficácia das estratégias adotadas.

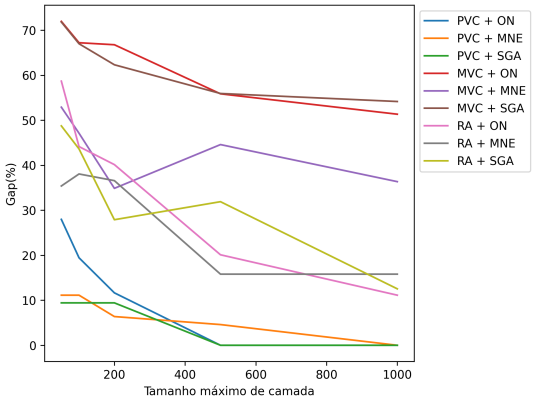
Figura 9 – Resultados do *gap* das combinações de heurísticas de exclusão de nós e ordenações de variáveis para o PCIPM.



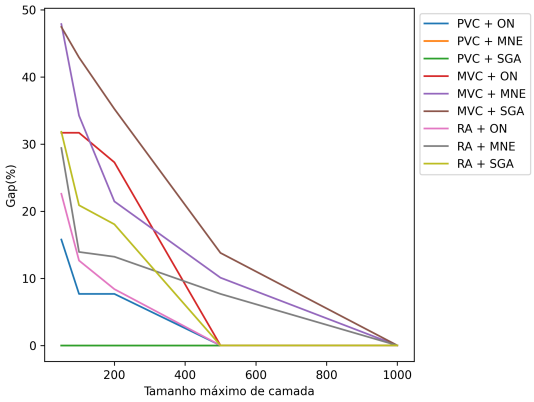
(a) Instância 1 - Densidade = 0.1



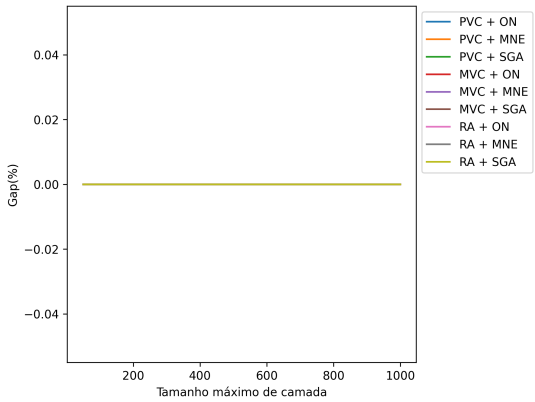
(b) Instância 3 - Densidade = 0.3



(c) Instância 5 - Densidade = 0.5



(d) Instância 7 - Densidade = 0.7



(e) Instância 10 - Densidade = 1

Fonte: Próprio autor.
Nota: Nas legendas, o sinal de "+" significa utilização conjunta. Por exemplo, PVC + ON significa o uso da heurística de exclusão de nós Pior Valor de Caminho em conjunto com a ordem natural das variáveis.

Como pode ser observado, a heurística de exclusão de nós PVC combinada com todas as ordenações de variáveis propostas apresentou os menores valores de *gap*. Destaca-se,

em especial, a combinação da heurística com a ordenação de variáveis SGA, que apresentou os melhores resultados em todas as instâncias analisadas para valores de W menores que 200. Assim, pode-se inferir que manter os nós com os maiores valores de caminho mais longo até o momento é uma estratégia eficiente. Por outro lado, a heurística que exclui os nós com os menores valores de caminho longo até o momento (MVC) apresentou os piores resultados em relação ao *gap*, sendo até mesmo inferior a uma estratégia não estruturada, como a RA, que remove nós de maneira aleatória na camada. Isso evidencia ainda mais que a estratégia PVC se sobressai das demais estratégias escolhidas para a exclusão de nós.

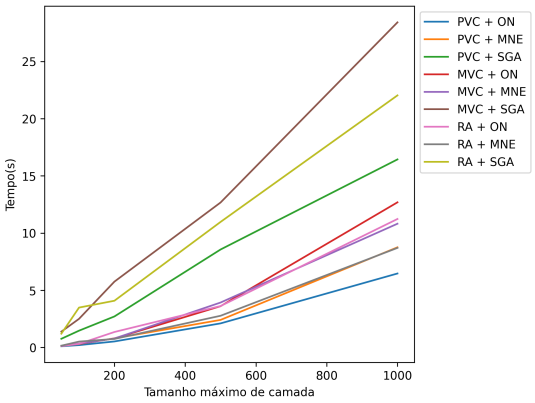
É evidente a influência das diferentes estratégias de ordenação de variáveis adotadas. Observa-se que a ordenação de variáveis natural (ON) apresentou desempenho inferior em comparação com ordenações de variáveis consideradas mais sofisticadas. No entanto, destaca-se que a combinação da ordenação SGA com a heurística de exclusão de nós MVC resultou nos piores desempenhos em todas as instâncias avaliadas. Esse resultado pode ser atribuído à lógica da ordenação SGA, que visa reduzir o espaço de busca ao selecionar, preferencialmente, o vértice de menor grau nos subgrafos induzidos representados pelos estados da camada. Por outro lado, a heurística MVC tende a eliminar os nós mais promissores da camada que excedem o limite W, comprometendo o desempenho geral da solução.

O caso que se desvia da curva é a instância 10 (Figura 9e), na qual todas as estratégias encontraram a solução ótima para todos os valores de W. Isso ocorre devido à menor complexidade dessa instância, que possui um espaço de busca reduzido. Como a instância apresenta alta densidade, intuitivamente, ela contém menos conjuntos independentes, reduzindo significativamente o espaço de busca. Por outro lado, instâncias com menores valores de densidade apresentam um espaço de busca mais amplo, tornando-se boas candidatas para avaliar a qualidade das heurísticas. Esse comportamento é evidente nos resultados obtidos para as instâncias 1 e 3 (Figura 9a e Figura 9b), que demonstram uma maior discrepância no desempenho entre as diferentes heurísticas testadas.

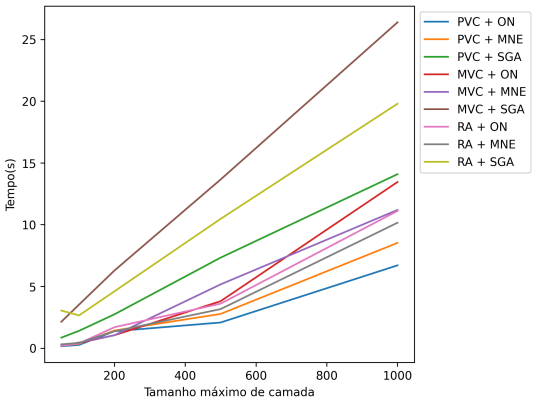
Outra métrica essencial para avaliar a qualidade das heurísticas de exclusão de nós é o tempo de execução. Como essas heurísticas são aplicadas em combinação com estratégias de ordenação de variáveis, a análise não deve ser limitada apenas às heurísticas de exclusão de nós. É igualmente importante considerar o impacto das diferentes ordenações de variáveis no desempenho geral, permitindo uma avaliação mais abrangente do comportamento das combinações propostas. A Figura 10 apresenta o tempo de execução das combinações entre heurísticas de

exclusão de nós e estratégias de ordenação de variáveis em relação à variação do parâmetro W. As instâncias selecionadas para essa análise são as mesmas exibidas na Figura 9, permitindo uma comparação direta entre os resultados.

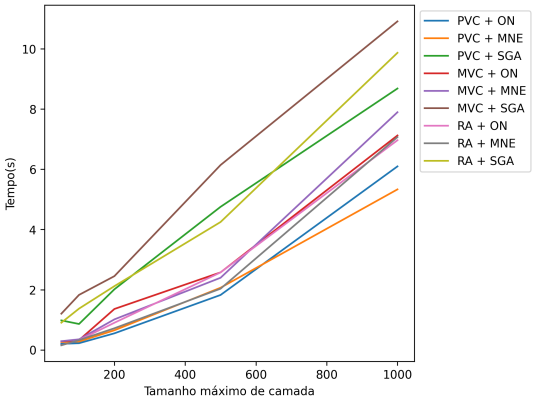
Figura 10 – Resultados do tempo das combinações de heurísticas de exclusão de nós e ordenações de variáveis para o PCIPM.



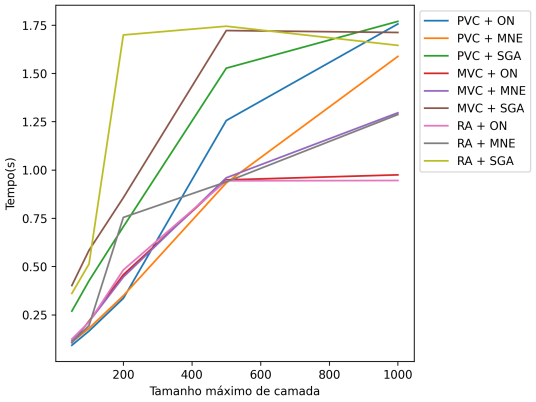
(a) Instância 1 - Densidade = 0.1



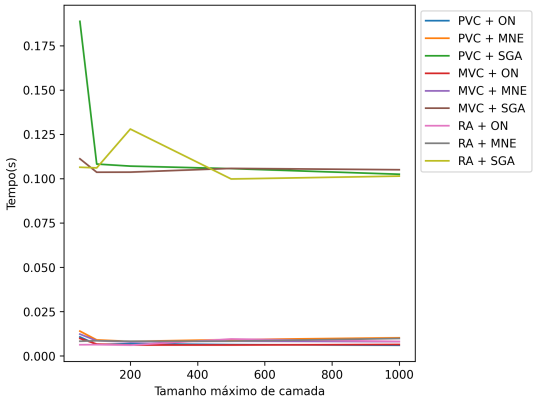
(b) Instância 3 - Densidade = 0.3



(c) Instância 5 - Densidade = 0.5



(d) Instância 7 - Densidade = 0.7



(e) Instância 10 - Densidade = 1

Fonte: Próprio autor.
Nota: Nas legendas, o sinal de "+" significa utilização conjunta.

Os resultados apresentados na Figura 10 reforçam a ineficiência da combinação entre a heurística de exclusão de nós MVC e a ordenação de variáveis SGA. Além de apresentar elevados valores de *gap* (Figura 9), essa estratégia também se destaca pelos piores tempos de execução entre as combinações analisadas.

Em geral, a ordenação de variáveis SGA obteve os piores resultados relacionado ao tempo. Isso já era de se esperar, pois comparada com as demais ordenações propostas, ela possui a maior a complexidade, no pior caso. Como a ordenação de variáveis ON apresenta complexidade constante, a combinação dessa ordenação com as heurísticas de exclusão de nós obteve os melhores resultados em termos de tempo de execução.

4.2 Resultados das heurísticas de seleção de nós no DD relaxado

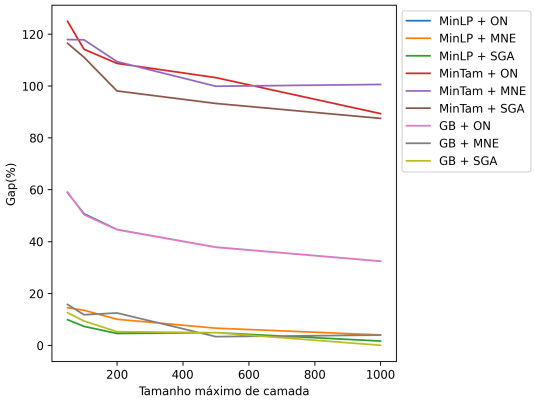
Na seção anterior, foi analisado o impacto das heurísticas de exclusão de nós no DD restrito, o qual fornece limites inferiores para o PCIPM. Nesta seção, serão apresentados os resultados das análises realizadas para avaliar o desempenho das heurísticas de seleção de nós para a mesclagem no DD relaxado aplicado ao PCIPM. As análises realizadas nesta seção são, essencialmente, as mesmas descritas na seção anterior. Todas as combinações entre heurísticas de seleção de nós e estratégias de ordenação de variáveis foram testadas. Considerando que há 3 heurísticas de seleção de nós e 3 tipos de ordenação de variáveis, obtém-se um total de 9 combinações distintas. Essas combinações serão avaliadas em função da variação do parâmetro W, que, por sua vez, assume os valores $W \in \{50, 100, 200, 500, 1000\}$.

A primeira análise realizada refere-se ao *gap*, descrito na Equação 4.1, que, conforme discutido anteriormente, indica, em porcentagem, o quão distante o limite encontrado está do valor da solução ótima x^*. O *gap* das combinações será avaliado em função da variação do parâmetro W. Essa análise busca verificar a eficiência das abordagens ao aproximarem-se da solução ótima conforme diferentes configurações do parâmetro. Já a segunda análise aborda o tempo de execução dessas combinações, também considerando a variação do parâmetro W, permitindo uma comparação do desempenho de cada abordagem em termos de tempo computacional.

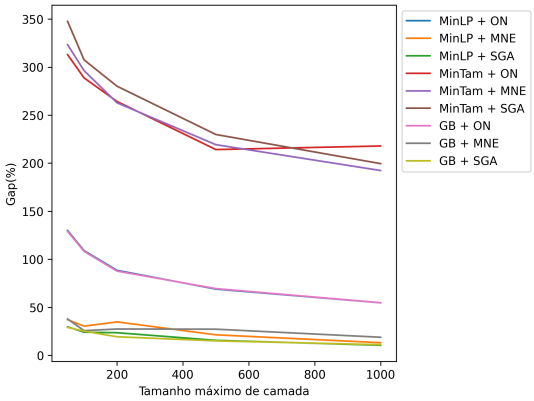
Das 10 instâncias consideradas neste trabalho, a Figura 11 apresenta os resultados da primeira análise para 5 instâncias selecionadas, de forma semelhante ao realizado na seção anterior. Essas instâncias foram escolhidas com base em suas características representativas, refletindo a diversidade do conjunto de dados. Elas representam diferentes valores de densidade,

um parâmetro essencial no PCIPM, pois influencia diretamente o grau de dificuldade de cada instância.

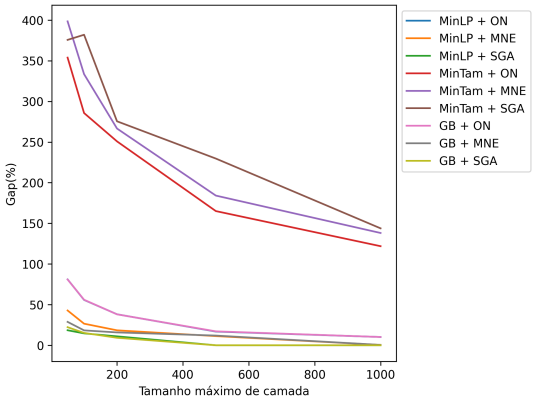
Figura 11 – Resultados do *gap* das combinações de heurísticas de seleção de nós e ordenações de variáveis para o PCIPM.



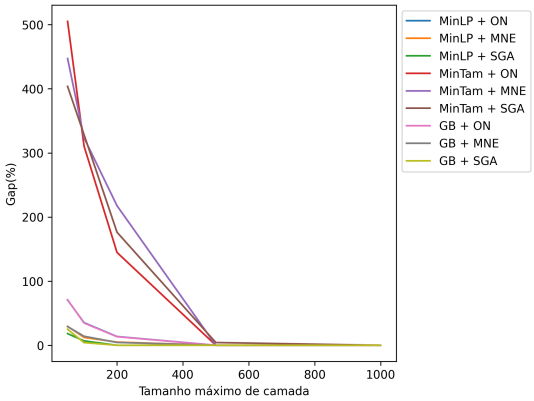
(a) Instância 1 - Densidade = 0.1



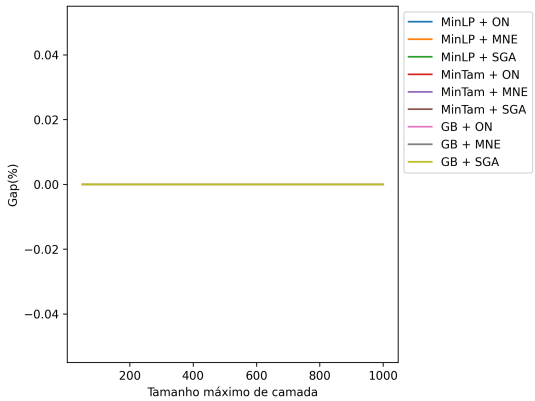
(b) Instância 3 - Densidade = 0.3



(c) Instância 5 - Densidade = 0.5



(d) Instância 7 - Densidade = 0.7



(e) Instância 10 - Densidade = 1

Fonte: Próprio autor.
Nota: Nas legendas, o sinal de "+" significa utilização conjunta.

Com base nesses resultados, pode-se inferir que as heurísticas de seleção de nós mais eficazes para a mesclagem são a MinLP e a GB, especialmente quando combinadas com as ordenações de variáveis MNE e SGA. Era esperado que a heurística de seleção de nós MinLP apresentasse um bom desempenho, pois ela segue a mesma lógica da heurística de exclusão de nós PVC, que obteve os melhores resultados relacionado ao *gap* no DD restrito. Ambas as heurísticas compartilham a estratégia de priorizar nós com os caminhos mais longos na camada que excedeu o limitante W. A ordenação de variáveis menos sofisticada, ON, apresentou desempenho insatisfatório em relação ao *gap*, mesmo quando combinada com as heurísticas de seleção de nós MinLP e GB. Estas últimas obtiveram os melhores resultados para essa métrica quando associadas às demais ordenações de variáveis analisadas neste trabalho.

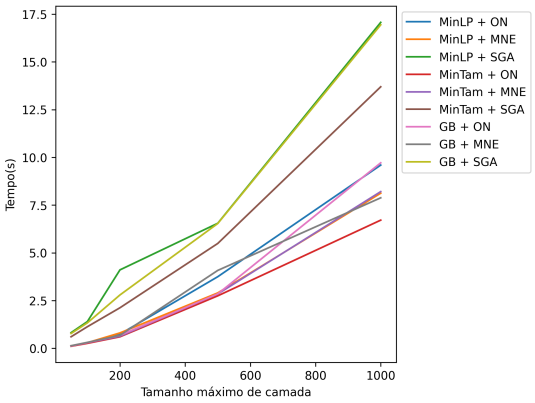
No geral, a combinação da heurística GB com a ordenação SGA foi a que apresentou os melhores limites superiores. Nafar e Römer (2024) demonstraram que essa estratégia é eficaz para o PCIM. Com base nos resultados obtidos, observa-se que essa combinação também é eficiente para o PCIPM, fornecendo bons limites superiores mesmo para pequenos valores de W, onde ocorre uma maior mesclagem de nós.

Por outro lado, fica evidente que a heurística MinTam não é uma boa estratégia para a seleção de nós na mesclagem, já que as três combinações em que ela está presente apresentaram consistentemente os três piores resultados em todas as instâncias analisadas. O resultado é ainda mais insatisfatório para pequenos valores de W, onde a demanda por desempenho dessas heurísticas é maior. O caso atípico refere-se novamente à instância 10, em que todas as combinações para todos os valores de W apresentaram *gap* igual a 0%. Como já explicado anteriormente, isso ocorre devido à alta densidade da instância, caracterizada por um grande número de arestas em seu grafo. Essa característica reduz significativamente o número de conjuntos independentes, resultando em um espaço de busca menor em comparação com as demais instâncias.

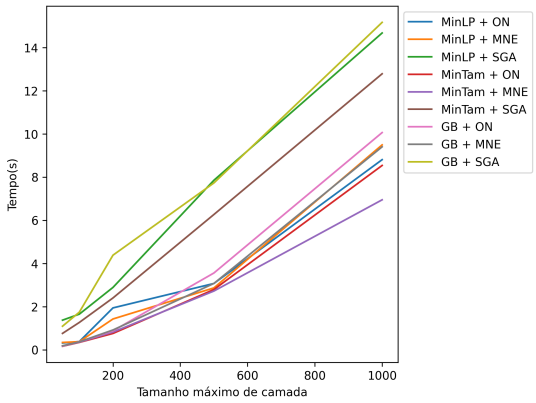
A próxima análise dessas combinações será em relação ao tempo de execução. Embora ordenações de variáveis mais sofisticadas, como a SGA, apresentem bons limites tanto para o DD restrito quanto para o DD relaxado, elas possuem um tempo de execução maior em comparação com ordenações menos elaboradas, como a ON. Nas análises realizadas na seção anterior, verificou-se que as combinações envolvendo a ordenação SGA apresentaram um tempo de execução significativamente maior em comparação com as combinações em que essa ordenação não esteve presente. Nesta seção, será realizado a mesma análise feita na seção

anterior, testando o tempo de execução das combinações propostas em função da variação do parâmetro W. As instâncias testadas na Figura 12 são as mesmas utilizadas nas análises da Figura 11, permitindo uma comparação direta entre os resultados.

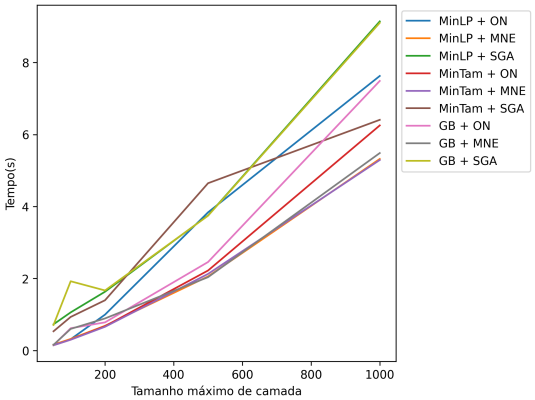
Figura 12 – Resultados do tempo das combinações de heurísticas de seleção de nós para a mesclagem e ordenações de variáveis para o PCIPM.



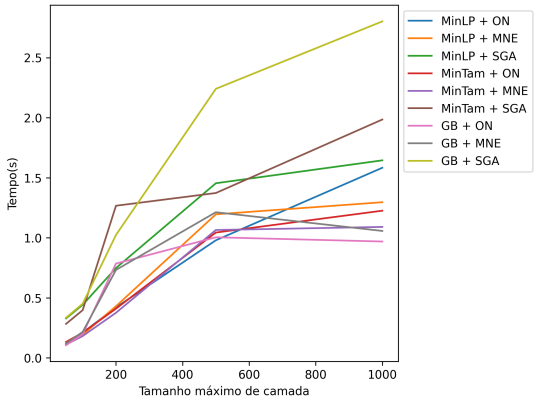
(a) Instância 1 - Densidade = 0.1



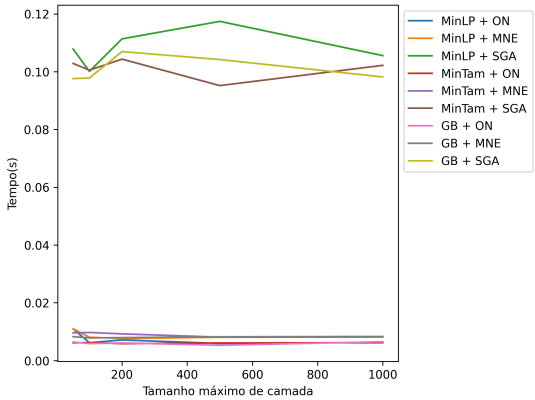
(b) Instância 3 - Densidade = 0.3



(c) Instância 5 - Densidade = 0.5



(d) Instância 7 - Densidade = 0.7



(e) Instância 10 - Densidade = 1

Fonte: Próprio autor.
Nota: Nas legendas, o sinal de "+" significa utilização conjunta.

De acordo com os resultados apresentados, a ordenação de variáveis SGA, mesmo quando combinada com heurísticas de seleção de nós, continua apresentando os piores tempos de execução, comportamento que já havia sido observado em sua combinação com heurísticas de exclusão de nós.

Outro resultado relevante, que não foi abordado na seção anterior, é que, à medida que o valor do parâmetro W aumenta, o tempo de execução das combinações também tende a crescer. Isso ocorre porque um maior número de nós é criado por camada, resultando em um tempo de execução mais elevado conforme o parâmetro aumenta. O único caso que se desvia do comportamento esperado é o resultado da instância 10. Devido à sua baixa complexidade, todas as combinações apresentam tempos de execução constantes e muito próximos de 0 segundos.

Em relação aos melhores resultados de tempo de execução, destacam-se as combinações envolvendo as ordenações MNE e ON. Esse comportamento era esperado, pois ambas possuem complexidade inferior à da ordenação SGA. A análise pode se concentrar apenas nas ordenações variáveis, já que as heurísticas de seleção de nós apresentam complexidade praticamente igual. A mesma ideia é válida para as heurísticas de exclusão de nós.

4.3 Resultados das heurísticas no branch-and-bound baseado em DDs

Na Seção 4.1, foram apresentados os resultados relacionados à combinação entre as heurísticas de exclusão de nós e ordenações de variáveis escolhidas. Já na Seção 4.2, foram exibidos os resultados referentes à combinação entre as heurísticas de seleção de nós para a mesclagem e ordenações de variáveis propostas, ambas analisadas em função da variação do parâmetro W, que define o limite de nós por camada no DD. Nesta seção, serão apresentados os resultados das combinações no método de BAB baseado em DDs, aplicado ao PCIPM.

Como o método de BAB baseado em DDs precisa de limites inferiores retornados pelo DD restrito e limites superiores retornados pelo DD relaxado, então ele tem como parâmetros, uma heurística de exclusão de nós, uma heurística de seleção de nós para a mesclagem e uma ordenação de variáveis, além do limite de camada W. Considerando que há 3 heurísticas de exclusão de nós, 3 heurísticas de seleção de nós para a mesclagem e 3 ordenações de variáveis, o total de combinações de parâmetros possíveis seria 27 (fixando um valor de W), dificultando a análise. No entanto, como já foi demonstrada a qualidade dessas estratégias nas seções anteriores, optou-se por selecionar as duas melhores heurísticas de exclusão de nós, as duas melhores heurísticas de seleção de nós para a mesclagem e as duas melhores ordenações de variáveis, com

base nos resultados obtidos em relação ao *gap*, isso resultaria em um total de 8 combinações (fixando um valor de W). Com base nos critérios estabelecidos, as heurísticas de exclusão de nós selecionadas foram PVC e RA. As heurísticas de seleção de nós para a mesclagem escolhidas foram MinLP e GB, enquanto as ordenações de variáveis selecionadas foram MNE e SGA.

O primeiro experimento consistiu em fixar um tempo de 10 minutos (600 segundos) para cada combinação de parâmetros e verificar se ela conseguiu resolver a instância dentro desse limite. O parâmetro W também foi fixado em 50. A Tabela 2 apresenta os resultados desse experimento, onde o valor 0 indica que a combinação resolveu a instância no tempo estabelecido, enquanto o valor 1 indica que não foi possível. A primeira coluna apresenta as combinações de parâmetros utilizadas no método de BAB, incluindo a heurística de exclusão de nós, a heurística de seleção de nós para mesclagem e a ordenação das variáveis, respectivamente.

Tabela 2 – Status das combinações de parâmetros fixando W em 50.

Combinação de parâmetros	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$(PVC, MinLP, MNE)$	0	1	0	0	0	0	0	0	0	0
$(PVC, MinLP, SGA)$	0	1	0	0	0	0	0	0	0	0
(PVC, GB, MNE)	0	1	0	0	0	0	0	0	0	0
(PVC, GB, SGA)	0	1	0	0	0	0	0	0	0	0
$(RA, MinLP, MNE)$	1	1	0	0	0	0	0	0	0	0
$(RA, MinLP, SGA)$	1	1	0	0	0	0	0	0	0	0
(RA, GB, MNE)	1	1	0	0	0	0	0	0	0	0
(RA, GB, SGA)	1	1	0	0	0	0	0	0	0	0

Fonte: Elaborada pelo autor.
Nota: As colunas numéricas representam as instâncias do PCIPM.

A partir da Tabela 2, é possível observar que as instâncias mais difíceis de serem resolvidas foram as instâncias 1 e 2, sendo que, no caso da instância 2, nenhuma combinação conseguiu resolvê-la dentro do tempo estabelecido. Em relação à instância 1, as combinações que utilizaram a heurística de exclusão de nós RA não foram capazes de resolvê-la no tempo proposto, evidenciando que essa escolha de heurística impactou negativamente os limites inferiores obtidos pelo DD restrito e, como consequência, afetou todo o desempenho do método de BAB baseado em DDs. Excluindo as instâncias 1 e 2, todas as combinações conseguiram resolver as demais instâncias dentro do tempo estipulado.

Na próxima análise, além de fixar um tempo limite de 10 minutos para cada combinação de parâmetros, o parâmetro W foi ajustado para 50, como no caso anterior. O objetivo dessa análise é avaliar o tempo de execução de cada combinação, dentro limite estabelecido, e compará-lo com o desempenho do *solver OR-Tools*. Os resultados obtidos estão apresentados

na Tabela 3, o tempo de execução está em segundos. A última coluna da Tabela 3 apresenta a média dos tempos de execução para cada combinação. As combinações com status igual a 1 para algumas instâncias na Tabela 2 correspondem às que atingiram o limite de tempo de 600 segundos na Tabela 3, uma vez que o parâmetro W permaneceu inalterado e fixado em 50.

Tabela 3 – Tempo das combinações de parâmetros fixando W em 50.

Combinação de parâmetros	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Média**
$(PVC, MinLP, MNE)$	200,73	600,08	52,10	15,59	4,12	1,67	0,68	1,03	0,36	0,01	87,64
$(PVC, MinLP, SGA)$	380,04	600,13	96,08	25,48	10,23	3,30	1,15	3,52	0,65	0,10	112,07
(PVC, GB, MNE)	213,76	600,33	56,31	15,13	4,13	1,19	0,69	0,98	0,36	0,01	89,29
(PVC, GB, SGA)	344,14	600,07	97,37	25,86	9,84	2,44	1,08	2,22	0,68	0,11	108,38
$(RA, MinLP, MNE)$	600,04	600,05	122,43	24,77	7,15	1,49	0,71	1,00	0,38	0,01	135,8
$(RA, MinLP, SGA)$	600,14	600,50	241,52	55,17	14,05	2,88	1,54	2,45	0,40	0,10	151,88
(RA, GB, MNE)	600,03	600,09	121,71	23,15	5,83	2,28	0,70	1,01	0,21	0,01	135,5
(RA, GB, SGA)	600,41	600,38	240,76	54,73	15,11	3,50	1,81	2,70	0,38	0,11	151,99
$OR-Tools$	0,44	35,49	16,3	15,84	2,70	1,65	3,74	11,33	1,11	0,28	8,89

Fonte: Elaborada pelo autor.

Nota: As colunas numéricas representam as instâncias do PCIPM e a última linha da tabela não representa uma combinação de parâmetros, mas sim os resultados obtidos pelo *solver* OR-Tools.

Com base nos resultados obtidos, pode-se inferir alguns aspectos. O primeiro diz respeito às instâncias 1 e 2, nas quais algumas combinações de parâmetros propostas não foram capazes de resolver as instâncias dentro do tempo estipulado ou tiveram um tempo de execução elevado, enquanto o *solver OR-Tools* conseguiu alcançar esse resultado. Vale destacar que o PCIPM é um problema clássico de otimização, motivo pelo qual a maioria dos *solvers* já implementa estratégias eficientes para reconhecê-lo resolvê-lo em um tempo reduzido. No entanto, a partir da instância 4 (com exceção da instância 5, em que o solver obteve o melhor desempenho), algumas combinações apresentaram tempos de execução inferiores ao do *solver*. Além disso, a partir da instância 7, todas as combinações superaram o desempenho do *solver*. Em média, o *solver OR-Tools* apresentou um desempenho superior em termos de tempo, influenciado, naturalmente, por sua eficiência em resolver as instâncias mais complexas.

Comparando apenas os tempos das combinações no método de BAB baseado em DDs, é possível observar novamente a influência das ordenações de variáveis. Nota-se que as combinações com a ordenação MNE apresentam, em média, um tempo de execução inferior ao das combinações com a ordenação SGA.

Outra análise importante a ser realizada diz respeito à quantidade de nós explorados por cada combinação. Este experimento é essencial, pois reflete a qualidade dos limites obtidos tanto no DD restrito quanto no DD relaxado, além de evidenciar a influência da ordenação de variáveis adotada. Assim como nos experimentos anteriores, será estabelecido um tempo

máximo de 10 minutos para cada combinação. Além disso, o parâmetro W permanecerá fixado em 50. A Tabela 4 traz os resultados obtidos.

Tabela 4 – Número de nós explorados por cada combinação de parâmetros fixando W em 50.

Combinação de parâmetros	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Média**
$(PVC, MinLP, MNE)$	4312	17591	1465	624	333	145	97	98	50	1	2471,6
$(PVC, MinLP, SGA)$	1863	2575	1045	422	320	190	50	101	50	1	661,7
(PVC, GB, MNE)	4650	17547	1613	670	333	145	97	98	50	1	2520,4
(PVC, GB, SGA)	1662	2533	1094	422	320	190	50	101	50	1	642,3
$(RA, MinLP, MNE)$	25007	18515	5860	1419	649	284	97	98	50	1	5198
$(RA, MinLP, SGA)$	1119	1045	5354	1777	645	281	100	101	50	1	1047,3
(RA, GB, MNE)	24777	18365	5832	1332	649	284	97	98	50	1	5148,5
(RA, GB, SGA)	1111	1042	5296	1735	740	281	100	101	50	1	1045,7

Fonte: Elaborada pelo autor.
Nota: As colunas numéricas representam as instâncias do PCIPM.

A Tabela 4 apresenta resultados interessantes, sendo o principal relacionado ao número médio de nós explorados pelas combinações. Nota-se que as combinações com a ordenação de variáveis SGA exploraram, em média, um menor número de nós do que as demais, o que pode ser atribuído à sua finalidade de reduzir o espaço de busca para mais próximo da solução ótima. Por outro lado, como mostrado na Tabela 3, as combinações que utilizam essa ordenação apresentam tempos de execução mais elevados.

- Qual a configuração do solver? Configuração padrão?

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste estudo, foi analisada a influência de diversos fatores na qualidade dos limites obtidos por DDs, incluindo heurísticas para exclusão de nós no DD restrito, estratégias de seleção de nós para mesclagem no DD relaxado, ordenações de variáveis e o impacto do limite de tamanho de camada W. Além disso, foi analisado o impacto de todos esses fatores no desempenho do método de BAB baseado em DDs, aplicado ao PCIPM.

Os resultados indicam que a combinação entre heurísticas de exclusão de nós (DD restrito) ou heurísticas de seleção de nós (DD relaxado) com diferentes ordenações de variáveis é essencial para obter limites de boa qualidade para o PCIPM. Um resultado relevante são os obtidos pela heurística GB, originalmente proposta para o PCIM. No entanto, como demonstrado, essa heurística também apresenta um desempenho excelente para o PCIPM.

No caso do BAB baseado em DDs, verificou-se que a integração de todos esses fatores é crucial para o desempenho do algoritmo. Por exemplo, observou-se que as combinações de parâmetros que incluíam a heurística de exclusão de nós RA não apresentaram um desempenho satisfatório, devido a natureza não estruturada da heurística, que remove nós de maneira aleatória quando o limite de tamanho W é extrapolado na camada. Por outro lado, os resultados obtidos indicam que as combinações com a ordenação de variáveis MNE resolveram as instâncias, em média, em um tempo menor do que aquelas que utilizaram a ordenação SGA. No entanto, as combinações que utilizaram a ordenação de variáveis SGA exploraram, em média, um número menor de nós em comparação às combinações que adotaram a ordenação MNE. Esse resultado deve-se à qualidade dos limites fornecidos pelos DDs que utilizam a ordenação SGA, conforme observado nos resultados das Seções 4.1 e 4.2. Outro resultado relevante foi o desempenho das combinações em algumas instâncias, nas quais superaram o *solver OR-Tools*, resolvendo-as em um tempo significativamente menor.

Os próximos passos deste estudo incluem a avaliação de heurísticas adicionais da literatura, bem como o desenvolvimento de novas abordagens. As principais tarefas a serem realizadas são as seguintes: 1) Testar outras seleções de corte exato S, visando avaliar a sua importância no desempenho do método de BAB baseado em DDs. 2) Analisar a influência do parâmetro W no desempenho do método de BAB baseado em DDs. 3) Comparar os resultados com um *solver* mais robusto e instâncias clássicas da literatura. 4) Comparar os limites retornados pelo DD relaxado com relaxações lineares, conforme feito por BERGMAN *et al.* (2016).

REFERÊNCIAS

ANDERSEN, H. R.; HADZIC, T.; HOOKER, J. N.; TIEDEMANN, P. A constraint store based on multivalued decision diagrams. In: SPRINGER. **Principles and Practice of Constraint Programming–CP 2007: 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007. Proceedings 13**. [S.l.], 2007. p. 118–132.

BERGMAN, D.; CIRE, A. A.; HOEVE, W.-J. V.; HOOKER, J. N. Variable ordering for the application of bdds to the maximum independent set problem. In: SPRINGER. **Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems: 9th International Conference, CPAIOR 2012, Nantes, France, May 28–June1, 2012. Proceedings 9**. [S.l.], 2012. p. 34–49.

BERGMAN, D.; CIRE, A. A.; HOEVE, W. J. V.; HOOKER, J. **Decision diagrams for optimization.** [S.l.]: Springer., 2016.

BERGMAN, D.; CIRE, A. A.; HOEVE, W.-J. van. Lagrangian bounds from decision diagrams. **Constraints**, Springer, v. 20, p. 346–361, 2015.

BERGMAN, D.; CIRE, A. A.; HOEVE, W.-J. van; YUNES, T. BDD-based heuristics for binary optimization. **Journal of Heuristics**, Springer, v. 20, p. 211–234, 2014.

BERGMAN, D.; HOEVE, W.-J. V.; HOOKER, J. N. Manipulating MDD relaxations for combinatorial optimization. In: SPRINGER. **Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011. Proceedings 8**. [S.l.], 2011. p. 20–35.

BERGMAN, D.; LOZANO, L. Decision diagram decomposition for quadratically constrained binary optimization. **INFORMS Journal on Computing**, INFORMS, v. 33, n. 1, p. 401–418, 2021.

CAPPART, Q.; GOUTIERRE, E.; BERGMAN, D.; ROUSSEAU, L.-M. Improving optimization bounds using machine learning: Decision diagrams meet deep reinforcement learning. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. v. 33, n. 01, p. 1443–1451.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, 1959.

GONZALEZ, J. E.; CIRE, A. A.; LODI, A.; ROUSSEAU, L.-M. Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem. **Constraints**, Springer, v. 25, n. 1, p. 23–46, 2020.

HOOKER, J. N. Decision diagrams and dynamic programming. In: SPRINGER. **Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 10th International Conference, CPAIOR 2013, Yorktown Heights, NY, USA, May 18-22, 2013. Proceedings 10**. [S.l.], 2013. p. 94–110.

HUANG, L.; CHEN, X.; HUO, W.; WANG, J.; ZHANG, F.; BAI, B.; SHI, L. Branch and bound in mixed integer linear programming problems: A survey of techniques and trends. **arXiv preprint arXiv:2111.06257**, 2021.

KARP, R. Reducibility among combinatorial problems. In: . [S.l.: s.n.], 1972. v. 40, p. 85–103. ISBN 978-3-540-68274-5.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. **Econometrica**, v. 28, p. 497, 1960. Disponível em: <https://api.semanticscholar.org/CorpusID:35442133>.

LEE, C.-Y. Representation of switching circuits by binary-decision programs. **The Bell System Technical Journal**, Nokia Bell Labs, v. 38, n. 4, p. 985–999, 1959.

NAFAR, M.; RÖMER, M. Strengthening relaxed decision diagrams for maximum independent set problem: Novel variable ordering and merge heuristics. In: SCHLOSS DAGSTUHL–LEIBNIZ-ZENTRUM FÜR INFORMATIK. **30th International Conference on Principles and Practice of Constraint Programming (CP 2024)**. [S.l.], 2024.

Vocês conhecem a interpretação de programação dinâmica para o o problema da mochila como um problema de caminho mais longo?Funciona sobre um grafo aciclico.