Novos Métodos de Programação Linear Inteira Mista para o Problema das Árvores de Classificação Ótimas

Victor José de Sousa Koehler

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2024

Victor José de Sousa Koehler

Novos Métodos de Programação Linear Inteira Mista para o Problema das Árvores de Classificação Ótimas

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Dr. Gilberto Farias de Sousa Filho

Maio de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

K77n Koehler, Victor José de Sousa.
Novos métodos de programação linear inteira mista para o problema das árvores de classificação ótimas / Victor José de Sousa Koehler. - João Pessoa, 2024.
73 f. : il.

Orientação: Gilberto Farias de Sousa Filho.
Coorientação: Thiago Gouveia da Silva.
Dissertação (Mestrado) - UFPB/CI.

1. Árvore classificação - Estrutura de dados. 2. Programação inteira mista. 3. Optimal Classification Trees (OCT). I. Sousa Filho, Gilberto Farias de. II. Silva, Thiago Gouveia da. III. Título.

UFPB/BC CDU 004.6(043)

Elaborado por ANNA REGINA DA SILVA RIBEIRO - CRB-15/24

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Victor José de Sousa Koehler, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 29 de maio de 2024.

Aos vinte e nove dias do mês de maio do ano de dois mil e vinte e quatro, às dez horas, no auditório do Centro de Informática, reuniram-se os membros da Banca Examinadora constituída para julgar o trabalho do sr. Victor José de Sousa Koehler, vinculado a esta Universidade sob a matrícula nº 20211023501, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Gilberto Farias de Sousa Filho (PPGI), Orientador e Presidente da banca; Lucídio dos Anjos Formiga Cabral (PPGI), Examinador Interno e Thiago Gouveia da Silva (IFPB), Examinador Externo à Instituição. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou a finalidade da reunião e passou a palavra ao candidato para que ele fizesse a exposição oral do trabalho de dissertação intitulado: "Novos Métodos de Programação Linear Inteira Mista para o Problema das Árvores de Classificação Ótimas". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Gilberto Farias de Sousa Filho, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 29 de maio de 2024.

Prof. Dr. Gilberto Farias de Sousa Filho

Documento assinado digitalmente
gov.br
GILBERTO FARIAS DE SOUSA FILHO
Data: 29/05/2024 12:09:08-0300
Verifique em https://validar.iti.gov.br

Prof. Gilberto Farias de Sousa Filho
Orientador (PPGI-UFPB)

Documento assinado digitalmente
gov.br
LUCIDIO DOS ANJOS FORMIGA CABRAL
Data: 04/06/2024 13:19:58-0300
Verifique em https://validar.iti.gov.br

Prof. Lucidio dos Anjos Formiga Cabral
Examinador Interno (PPGI-UFPB)

Documento assinado digitalmente
gov.br
Thiago Gouveia da Silva
Data: 04/06/2024 14:17:22-0300
Verifique em https://validar.iti.gov.br

Prof. Thiago Gouveia da Silva
Examinador Externo à Instituição (IFPB)

"Se eu vi mais longe, foi por estar sobre ombros de gigantes"
(Isaac Newton)

DEDICATÓRIA

Dedico este trabalho aos meus queridos pais por seu amor e dedicação.

AGRADECIMENTOS

Agradeço a Deus por seu imenso amor e divina misericórdia.

A todos os meus familiares por todo o suporte, carinho e lições que trouxeram-me aqui e cujo apoio permite-me pensar adiante.

Aos professores do Centro de Informática pelos seus ensinamentos e dedicação no exercício de suas funções, proporcionando-me valiosas experiências. Em particular, agradeço ao meu orientador Dr. Gilberto Farias, e também ao Dr. Thiago Gouveia pela oportunidade de conhecer e participar em suas linhas de pesquisa.

Aos meus colegas de turma e curso que tive o prazer em conhecer e empreender conjuntamente esforços em prol de nosso autoaperfeiçoamento.

RESUMO

Árvores de classificação são um recurso muito útil na área de aprendizado de máquina pela facilidade de interpretação humana de suas predições. Motivado pela melhoria dos algoritmos e da capacidade de hardware para resolução de problemas de programação inteira, trabalhos recentes abordam a criação de árvores de classificação ótima através de modelos lineares, sendo o modelo mais geral intitulado *Optimal Classification Trees* (OCT). Propomos neste trabalho uma reformulação do OCT, inequações válidas baseada no grafo de precedência dos pontos de aprendizado e uma estratégia de *branch-and-cut* sobre restrições que dependem do número n de pontos, que é muito grande em várias instâncias de entrada da literatura. Ademais, apresentamos técnicas e estratégias para mitigar o alto custo computacional atrelado a essa alta dimensionalidade dos pontos através da sua redução, por meio de heurísticas e modelos exatos, visando estabelecer um *trade-off* entre assertividade, representatividade e tempo de execução dos modelos. Os resultados demonstram que a reformulação do modelo reduz em média 57% do tempo computacional do OCT, e que as inequações de precedência diminuem em 58% o número de nós resolvidos pela estratégias de *branch-and-bound*.

Palavras-chave: Classificação, Árvore de decisão, Programação inteira mista.

ABSTRACT

Classification trees are very useful techniques in the field of machine learning due to the ease of human interpretation of their predictions. Motivated by the improvement of algorithms and hardware capacity for solving integer programming problems, recent works address the creation of optimal classification trees through linear models, the most general model being entitled Optimal Classification Trees (OCT). We propose in this work a reformulation of the OCT, valid inequalities based on the precedence graph of the learning input and a branch-and-cut strategy on the restrictions that depend on the number n of points that is very large in several instances of the literature. In addition, we present techniques and strategies to mitigate the high computational cost linked to this high dimensionality of points through their reduction, by means of heuristics and exact models, aiming to establish a trade-off between assertiveness, representativeness and models execution time. The results demonstrate that the model reformulation reduces on average 57% the computational time of the OCT, and that the precedence inequalities decrease in 58% the number of nodes solved by the branch-and-bound strategies.

Keywords: Classification, Decision Tree, Mixed Integer Programming.

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

OCT - Optimal Classification Trees

ML - Machine Learning

PLIM - Programação Linear Inteira Mista

CART - Classification And Regression Trees

Sumário

1 Introdução

A área da Inteligência Artificial tem sido alvo de grande interesse nos últimos anos, tanto no campo acadêmico quanto no midiático. Em especial, os termos Aprendizagem de Máquina (ML, do inglês *Machine Learning*), Redes Neurais e *Deep Learning* têm recebido bastante atenção não apenas em artigos científicos, mas também em publicações voltadas para o público tecnológico. Estamos vivendo uma era de carros autônomos (*self-driven cars*), assistentes virtuais e robôs de bate-papo (*chatbots*) altamente inteligentes. Em geral, o grande sucesso que a Inteligência Artificial tem vivenciado se deve a três fatores: (i) avanços algorítmicos, (ii) evolução do *hardware* computacional, e (iii) grande disponibilidade de dados e *benchmarks* (CHOLLET, 2021).

Por outro lado, vale observar que uma grande classe de problemas estatísticos pode ser expressada de forma natural usando Programação Linear Inteira Mista (PLIM) (ARTHANARI; DODGE, 1993). No entanto, em relação à Inteligência Artificial, utilizar a PLIM é historicamente considerado impraticável, dado que os problemas associados são intratáveis. Neste sentido, é importante ressaltar que os pontos (i) e (ii) também se aplicam à PLIM. Considerando os avanços algorítmicos dos melhores resolvedores do mercado, temos que o CPLEX 11 (2007) é 29 mil vezes mais rápido que o CPLEX 1.2 (1991), enquanto que o Gurobi 6.5 (2015) é 48.7 vezes mais rápido que o Gurobi 1.6 (2009), que é comparável ao CPLEX 11 (BERTSIMAS; DUNN, 2017). Pode-se dizer que o *speed up* total é de 1.4×10^6 entre 1991 e 2015. Por sua vez, em relação aos avanços de *hardware*, temos um aumento de velocidade na casa de 1.6×10^6, comparando o limite de 93.0 PFlop/s de 1993 contra 59.7 GFlop/s de 2016 (STROHMAIER et al., 2015). Combinando estes *speed ups*, temos um aumento de velocidade na casa de 2.2 trilhão. Por exemplo, um problema modelado como PLIM que poderia levar 71 mil anos para ser resolvido 25 anos atrás, agora pode ser resolvido em menos de um segundo (BERTSIMAS; DUNN, 2017).

Tal cenário abre espaço para que a PLIM seja utilizada de forma competitiva para resolver problemas de ML, como pode ser observado no trabalho de Bertsimas, Orfanoudaki e Wiberg (2021), que utiliza técnicas de PLIM para gerar agrupamentos interpretáveis baseados em árvores, e nos trabalhos de Bertsimas e Dunn (2017) e Verwer e Zhang (2019), que utilizam formulações baseadas em PLIM para gerar árvores de classificação ótimas. As árvores de classificação, em especial, possuem duas características que fundamentam a implementação de métodos baseados em PLIM para sua solução: a alta interpretabilidade dos seus resultados e o fato de que seu estado da arte é composto basicamente por heurísticas gulosas.

A interpretabilidade é uma característica extremamente desejável para técnicas de ML. Suponha que um carro autônomo se envolva em um acidente com vítimas, quem

seria o culpado? Suponha que um estudante não seja selecionado como bolsista, ou que seja negado crédito bancário a uma empresa; seria aceitável responder apenas que um algoritmo tomou tal decisão? A sociedade irá aceitar a falta de entendimento? Enfim, a interpretabilidade se configura como um fator cada vez mais importante de toda a área da Inteligência Artificial. Por outro lado, os melhores métodos para o treinamento da árvores de classificação aplicam heurísticas de forma recursiva para tomar cada decisão, fazendo isto de forma isolada e gulosa (JAMES et al., 2021), o que termina por não capturar de forma ótima as características globais do conjunto de dados. Este é o caso do CART (do inglês, Classification And Regression Trees, Breiman et al. (1984)), algoritmo de referência na literatura que influenciou diversas variantes (SOUZA, 2021; BERTSIMAS; DUNN, 2017), ID3 (QUINLAN, J. Ross, 1986), C4.5 (QUINLAN, J Ross, 2014) etc.

Assim sendo, este trabalho propõe uma reformulação do modelo OCT (do inglês, *Optimal Classification Trees*), proposto por Bertsimas e Dunn (2017) e que emprega técnicas de PLIM para obtenção de árvores de classificação ótimas, assim como novas inequações válidas baseadas no grafo de precedências dos pontos de aprendizado. Além disso, para mitigar as dificuldades associadas ao grande número de pontos dos conjuntos de treinamento, propomos uma estratégia de *branch-and-cut* sobre os grupos de restrições sensíveis a esta característica; Ademais, definimos um problema de otimização para a extração e amostragem refinada desse conjunto, na qual ainda apresentamos métodos heurísticos e um exato para sua resolução e, fazendo uso desta abordagem, ainda propomos uma metodologia e um algoritmo para a geração de árvores de classificação com uma topologia mais flexível que as do OCT. Para tal, o restante deste trabalho está organizado como segue: a Seção 2 discute os fundamentos teóricos e o estado da arte dos problemas abordados; a Seção 3 introduz a reformulação do OCT proposta, assim como as suas melhorias; a Seção 4 apresenta a abordagem da amostragem da instância de entrada; a Seção 5 aborda a estratégia de geração de árvores de decisão de topologia flexível; a Seção 6 descreve os experimentos computacionais realizados e discute os resultados obtidos; por fim, a Seção 7 expõe as considerações finais e propostas de trabalhos futuros.

2 Fundamentação teórica

O objetivo desta seção é definir de maneira formal os problemas tratados neste trabalho, bem como apresentar um resumo do estado da arte dos assuntos abordados. Em um primeiro momento será formalizada a definição de um problema de classificação. Na sequência, será apresentada a formalização do problema da Árvore de Decisão e, em seguida, feita uma pequena discussão acerca de sua complexidade computacional.

2.1 Problema de Classificação

Seja dada uma dupla (X, Y), representando o conjunto de dados de treinamento, composto por n pares de observações (x_i, y_i), com $i = 1, 2, \ldots, n$. Considere que cada x_i é um ponto em $\mathbb{R}^p$, onde cada dimensão representa uma das p características (*features*) da amostra, e que cada $y_i \in \{1, 2, \ldots, K\}$ indica qual dos K rótulos está associado ao ponto x_i. Adicionalmente, sem perda de generalidade, assuma que os valores de cada dimensão (característica) estão normalizados no intervalo $0 - 1$, ou seja $x_i \in [0, 1]^p$. O objetivo do problema de classificação é produzir uma ferramenta (também conhecida como classificador) capaz de associar corretamente cada ponto x_i (ou grande parte destes) com o seu respectivo rótulo y_i, de modo que o método seja capaz de prever corretamente o rótulo de pontos que não pertençam ao conjunto X.

São exemplos clássicos de classificadores: Redes Neurais Artificiais (RNA), K-Nearest Neighbours (KNN), Suport Vector Machines (SVM), Regressão Logística, Árvore de Classificação (também conhecida como Árvore de Decisão), Naive Bayes, Floresta Aleatória, entre outros. Neste trabalho, estamos especialmente interessados nas Árvores de Classificação, dado a interpretabilidade que esta provê. Segue a formalização do problema do treinamento de uma Árvore de Decisão.

2.2 Problema da Árvore de Classificação

Dado um problema de classificação (X, Y), a árvore de classificação tem como objetivo um particionamento hierárquico do $\mathbb{R}^p$ em regiões disjuntas, cada uma associada a um rótulo de Y. Tal particionamento pode ser representado como uma árvore, composta por nós de ramificação e folhas. Cada nó de ramificação realiza um particionamento do conjunto X baseado nos parâmetros a e b. Dado um ponto x_i, caso $a^T x_i < b$, x_i deve-se seguir para a ramificação esquerda, caso contrário, deve-se seguir para a ramificação direita. Para o escopo deste trabalho, o vetor a possui apenas um valor igual a 1, enquanto todos os outros são 0, ou seja, cada ramificação seleciona apenas uma dimensão (ou característica), para particionar o conjunto de pontos. Finalmente, cada nó folha possui

um rótulo associado, aquele que aparece com mais frequência no mesmo, e este será usado como previsão para cada ponto que por ventura siga até esta folha.

Laurent e Rivest (1976) demonstraram que o Problema da Árvore de Classificação é NP-Completo por redução do problema EC3 (KARP, 1972), uma variante do Problema da Cobertura Exata (do inglês, *Exact Cover*), no qual cada conjunto possui no máximo cardinalidade 3.

2.3 CART

O funcionamento da heurística é ilustrada no Algoritmo 1: Dado um conjunto de treinamento, o primeiro passo do processo consiste na criação de um nó da árvore, para em seguida, decidir se tal nó torna-se folha (linhas 3-5) ou então é ramificado (linhas 8-16). No primeiro caso, o algoritmo escolhe e atribui a classe que melhor classifica o nó correspondente, retornando-o. No segundo, a função auxiliar $melhorParticao$ é invocada para decidir como dividir o conjunto de entrada em dois subconjuntos, cada qual originará uma ramificação do nó. Essa função busca a partição cujo decréscimo da impureza é máximo, sendo esta definida da seguinte forma:

$$\Delta I_\tau = I(n_p) - P_L \cdot I(n_L) - P_R \cdot I(n_R), \tag{1}$$

onde n_p é o nó raiz da árvore τ, e n_L e n_R são as suas ramificações esquerda e direita, respectivamente; P_L e P_R correspondem as respectivas proporções das alocações entre as duas ramificações, e $I(\cdot)$ é a função que descreve a medida de impureza, sendo o índice de Gini tipicamente utilizado como tal medida em problemas de classificação (CRISTERNA, 2017):

$$I(n_*) = 1 - \sum_{q \in \Omega} P_q^2, \tag{2}$$

onde P_q corresponde a probabilidade da ocorrência de amostras da classe $q \in \Omega$ no nó n_* qualquer. Desse modo, a função $melhorParticao$ escolhe e retorna a partição que gera a melhor divisão, de forma gulosa e local, no nó sendo explorado, de tal modo que essa pode não ser, necessariamente, a decisão ótima em respeito ao objetivo global de minimizar o erro.

2.4 OCT

Esta seção apresenta o modelo de Programação Linear Inteira Mista proposto por Bertsimas e Dunn (2017) para construção de Árvores de Classificação Ótimas (OCT, do

Algoritmo 1: Pseudocódigo do algoritmo CART, adaptado de (CRISTERNA, 2017)

Entrada: $\{X, Y\}$, l_{min}, n_{min}

Construir nó n_p com impureza $I(n_p)$
se $|\{X, Y\}| < n_{min}$ *ou* $I(n_P) = 0$ **então**
  $P_q \leftarrow$ Probabilidade da classe ω_q no nó atual
  $P_{r,q} \leftarrow$ Probabilidade da classe ω_q no nó raiz
  Atribuir a n_p a classe $\omega_q = argmax_q \ P_q/P_{r,q}$
fim
senão
  para $i = 1$ **até** m **faça**
    $\{\tau_i^*, \Delta I_{\tau,i}^*\} \leftarrow melhorParticao(i, \{X, Y\}, l_{min})$
  fim
  $\hat{i} = argmax_i \ \{\Delta I_{\tau,i}^* | 1 \leq i \leq m\}$
  Separar $\{X, Y\}$ em $\{X_1, Y_1\}$ e $\{X_2, Y_2\}$ considerando a $\hat{i}$-ésima característica e partição $\tau_{\hat{i}}^*$
  $n_L \leftarrow CART(\{X_1, Y_1\}, l_{min}, n_{min})$
  $n_R \leftarrow CART(\{X_2, Y_2\}, l_{min}, n_{min})$
  Atribuir n_L e n_R como filhos de n_p
  Atribuir a n_p a característica $\hat{i}$ e limiar $\tau_{\hat{i}}^*$
fim
retorna n_p

Algoritmo 2: Pseudocódigo da função melhorParticao, adaptado de (CRISTERNA, 2017)

Entrada: i, $\{X, Y\}$, l_{min}

$T \leftarrow$ limiares em X de característica $\hat{i}$ cujas partições possuam no mínimo l_{min} amostras
para $\tau \in T$ **faça**
  $\Delta I_\tau \leftarrow$ Decréscimo de impureza obtido pela partição τ
fim
$\tau^* = argmax_\tau \ \{\Delta I_\tau | \tau \in T\}$
retorna $\{\tau^*, \Delta I_{\tau*}\}$

inglês *Optimal Classification Trees*), considerado o estado da arte para este problema. Para encontrar a melhor árvore possível, o modelo deve ser capaz de tomar as seguintes decisões discretas: (a) para cada novo nó, deve-se decidir ramificá-lo ou torná-lo folha; (b) se o nó se tornar folha, deve-se atribuir um rótulo de classificação para ele; (c) se o nó for de ramificação, deve-se escolher um parâmetro e um limiar para ativar a divisão; e (d) quando classificar os pontos de treino de acordo com a árvore em construção, deve-se escolher a qual folha cada ponto será atribuído tal que a estrutura da árvore seja respeitada. As Tabelas 1 e 2 sintetizam, respectivamente, o conjunto de parâmetros e variáveis de decisão utilizados pelo OCT, enquanto que o modelo OCT é apresentado no programa (3) a (16).

Tabela 1: Sumário dos parâmetros do modelo OCT.

Parâmetro	**Definição**
D	profundidade máxima da árvore de decisão
α	penalização aplicada a complexidade da árvore
$\hat{L}$	acurácia da predição pela classe majoritária
Y_{ik}	valor 1 indica que o i-ésimo ponto tem rótulo k, 0 caso contrário
$t = \{1, ..., T\}$	índice dos nós da árvore, sendo $T = 2^{D+1} - 1$
$p(t)$	nó pai de t
$A_L(t)$	conjunto de nós ancestrais de t cuja ramificação esquerda foi usada no caminho do nó raiz até t
$A_R(t)$	similar a $A_L(t)$ é o conjunto de ancestrais da ramificação direita
$\mathcal{T}_B = \{1, ..., \lfloor T/2 \rfloor\}$	nós de ramificação que separam os pontos x na forma $a^T x < b$. Pontos que satisfazem esta condição seguem para a ramificação esquerda na árvore, caso contrário seguem pra ramificação direta
$\mathcal{T}_L = \{\lfloor T/2 \rfloor + 1, ..., T\}$	nós folha que fazem a predição de classe para cada ponto atribuído ao nó folha

Tabela 2: Sumário das variáveis do modelo OCT.

Variável de decisão	
$z_{il} \in \{0, 1\}$	se $z_{il} = 1$ então o i-ésimo ponto está no nó $l \in \mathcal{T}_L$
$a_t \in \{0, 1\}^p$	se $a_{tp} = 1$ então o parâmetro p é usado na restrição de divisão do nó $t \in \mathcal{T}_B$.
$b_t \in \mathbb{R}$	valor limiar de divisão do nó $t \in \mathcal{T}_B$
$d_t \in \{0, 1\}$	se $d_t = 1$ então a restrição de divisão do nó $t \in \mathcal{T}_B$ está ativa
$l_l \in \{0, 1\}$	se $l_t = 1$ então a folha $l \in \mathcal{T}_L$ está ativa
$c_{kt} \in \{0, 1\}$	se $c_{kt} = 1$ então o rótulo k é majoritário no nó t
$N_{kt} \in \mathbb{Z}_0^+$	número total de pontos de rótulo k no nó t
$N_t \in \mathbb{Z}_0^+$	número total de pontos no nó t
$L_t \in \mathbb{Z}_0^+$	número de nós mal classificados no nó t

A função objetivo (3) procura minimizar o total de pontos mal classificados com o termo $\sum_{t \in \mathcal{T}_L} L_t$, e a complexidade da árvore com o termo $\sum_{t \in \mathcal{T}_B} d_t$. O $\hat{L}$ é usado para normalizar os termos em relação ao efeito do α, tornando-o independente do tamanho da

instância. O parâmetro α, definido no algoritmo CART (BREIMAN et al., 1984), controla o compromisso entre a acurácia da solução e a complexidade da árvore e é utilizado durante a fase de regularização do classificador que melhor se ajusta aos dados.

Para a construção da estrutura da árvore de decisão temos que as restrições (4) e (5) garantem que todo ponto $x_i \in X$ está em uma única folha $t \in \mathcal{T}_L$ que esteja ativa ($l_t = 1$). As restrições (6) garantem que se um nó ramificação $t \in \mathcal{T}_B$ estiver ativo então deve ser atribuído um único parâmetro para sua inequação de divisão. As restrições (7) só permitem que um nó ramificação seja ativo se seu pai também estiver ativo. Já os conjuntos de restrições (8) e (9) garantem que todo ponto x_i só pode ser atribuído para uma folha $t \in \mathcal{T}_L$ se atender as inequações de divisão ($a^T x_i < b$) de todos os nós ramificação ativos que são ancestrais a t. As restrições (10) estabelecem o domínio da variável $b_t \in [0, 1]$, que é o limiar da inequação de divisão do nó de ramificação t, sendo válido pois todos os parâmetros dos pontos x_i foram normalizados. Caso $d_t = 0$ então $b_t = 0$, obrigando todos os pontos descerem para a folha da direita de t. Parâmetros ϵ e ϵ_{max} possuem valores suficientemente pequenos e foram adicionadas ao conjunto de restrições (9) para garantir sua linearização. Detalhes da construção destes valores podem ser observados em (BERTSIMAS; DUNN, 2017).

$$\text{Minimize} \quad \frac{1}{\hat{L}} \sum_{l \in \mathcal{T}_L} L_l + \alpha \sum_{t \in \mathcal{T}_B} d_t \tag{3}$$

$$\sum_{l \in \mathcal{T}_L} z_{il} = 1, \qquad i = \{1, .., n\}, \tag{4}$$

$$z_{it} \leq l_t, \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_L, \tag{5}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \qquad \forall t \in \mathcal{T}_B, \tag{6}$$

$$d_t \leq d_{p(t)}, \qquad \forall t \in \mathcal{T}_B \setminus \{1\}, \tag{7}$$

$$a_m^T x_i \geq b_t - (1 - z_{it}), \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_L, \forall m \in A_R(t), \tag{8}$$

$$a_m^T (x_i + \epsilon) \leq b_t + (1 + \epsilon_{max})(1 - z_{it}), \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_L, \forall m \in A_L(t), \tag{9}$$

$$0 \leq b_t \leq d_t, \qquad \forall t \in \mathcal{T}_B, \tag{10}$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \qquad \forall t \in \mathcal{T}_L, \tag{11}$$

$$N_t = \sum_{i=1}^{n} z_{it}, \qquad \forall t \in \mathcal{T}_L, \tag{12}$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \qquad k = \{1, .., K\}, \forall t \in \mathcal{T}_L \tag{13}$$

$$L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \qquad k = \{1, .., K\}, \forall t \in \mathcal{T}_L \tag{14}$$

$$L_t \leq N_t - N_{kt} + n c_{kt}, \qquad k = \{1, .., K\}, \forall t \in \mathcal{T}_L, \tag{15}$$

$$L_t \geq 0, \qquad \forall t \in \mathcal{T}_L. \tag{16}$$

Para realizar a computação dos pontos mal classificados pela árvore construída, temos que as restrições (11) atribuem para cada nó folha ativo uma única classe majoritária. As restrições (12) contabilizam o número de pontos N_t que foram atribuídos em cada folha t, e as restrições (13) contabilizam o número de pontos N_{kt} de cada rótulo k que estão em cada folha t. Por fim, os conjuntos de restrições (14) e (15) garantem que L_t compute a quantidade dos pontos em t com rótulo diferente do rótulo majoritário. O domínio das variáveis está descrito na Tabela 2.

2.5 Erro e Viés

Na tomada de decisão ou no julgamento humano temos a presença do erro. Erro na correção de prova, no diagnóstico médico, na sentença judicial, entre outros. Estes erros

podem ter origem na capacitação do profissional para exercer a função ou em elementos externos como notícias familiares, horário que a análise está acontecendo e até mesmo a decisão tomada no caso anterior. Podemos dividir estes erros em

$$error = bias + noise,$$

onde o viés é classificado como um desvio sistemático e o ruído uma dispersão aleatória. Em Kahneman, Sibony e Sunstein (2021) é apresentada a alegoria do tiro ao alvo, ilustrada na Figura 1, para diferenciar cada tipo de erro. A Figura 1(a) apresenta quatro grupos de cinco tiros em seus alvos, onde cada grupo utiliza uma arma distinta. Observando os resultados podemos afirmar que o grupo 1 acertou praticamente todos seus tiros, os tiros do grupo 2 foram muito dispersos do centro do alvo, sendo este tipo de erro classificado como ruído, enquanto o grupo 3 teve seus tiros deslocados sistematicamente na região inferior esquerda do alvo, motivados pela desregulação da mira de sua arma, este erro classificamos de viés. Já no grupo 4 o resultado é ruidoso e enviesado.

A Figura 1(b) ilustra o resultado dos tiros após a remoção dos alvos. Isto nos permite identificar uma diferença no tratamento dos tipos de erro. Sem o alvo não é possível identificar quem é o grupo enviesado, entretanto, identificar a dispersão aleatória ainda é possível, pois o ruído constitui a variabilidade nos dados que pode ser medida estatisticamente (desvio padrão).

Esta diferença pode ser observada no seguinte exemplo prático. O diretor de marketing de uma corretora de ações solicitou que os corretores da empresa definissem qual seria o crescimento da bolsa brasileira no ano, para usar em uma campanha publicitária. Em média, os 37 corretores indicaram um crescimento de 15% com um desvio padrão de 5%. O ruído das previsões de seus profissionais é facilmente metrificado no desvio padrão, entretanto, o viés só poderá ser conhecido quando se descobrir o "alvo", ou seja, ao final do ano quando soubermos o quanto a bolsa brasileira variou. Neste ano específico ela cresceu apenas 7%, gerando um viés de 8% computado pela diferença do que de fato ocorreu e a média das previsões da equipe.

E os erros nas bases de dados utilizadas na técnica de aprendizado supervisionado, é possível quantificar? Segundo Hastie, Tibshirani e Friedman (2009), no aprendizado supervisionado, consideramos um conjunto de dados onde cada exemplo é uma tupla (x, y), em que x é um vetor de características ou atributos e y é a saída ou rótulo correspondente. Seu objetivo é encontrar a função hipótese $h(x, \theta) \in H$ cujos parâmetros θ mapeiam os vetores x em y minimizando $E(h(x, \theta), y)$, sendo $E(.)$ a função erro que quantifica a divergência entre as classificações $h(x, \theta)$ e as saídas y. Denotamos este conjunto de dados como $D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, sendo D apenas uma amostra de todas as possíveis classificações de um dado contexto, contendo dispersão aleatória (ruído) pro-

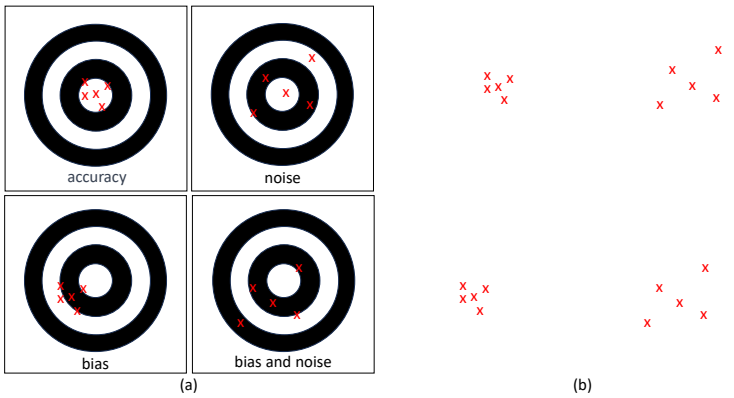


Figura 1: Tipos de erro no tiro ao alvo. (a) Tiros com o alvo; (b) Tiros com a remoção do alvo.

duzida pela fonte dos dados e o viés oriundo da estratificação da amostra e da formação do classificador.

Como dito anteriormente, o ruído pode ser identificado em D, e a técnica de aprendizado supervisionada possui um remédio clássico para este erro, a regularização. Segundo Ng e Jordan (2004), o objetivo da regularização é adicionar uma penalidade em θ durante o processo de treinamento, de forma a restringir a complexidade do modelo para que o mesmo não se sobre-ajuste a detalhes específicos aos dados de treinamento, melhorando sua capacidade de generalização para dados não vistos. Por sua vez, Cross Validation (ARLOT; CELISSE, 2010) é a principal técnica de validação dos hiper parâmetros de regularização para a escolha de θ que menos aprende os ruídos contidos em D. Essa técnica, ao dividir o conjunto de dados em subconjuntos de treinamento e validação de maneira repetida e estratificada, permite uma análise criteriosa da robustez dos hiperparâmetros escolhidos, garantindo que o modelo seja mais eficaz na generalização para novos dados não observados.

Já para eliminar o viés contido em D seria necessário conhecer mais informações além de D. Trabalhos recentes procuram definir frameworks e regras com este objetivo. Kamiran e Calders (2012) apresenta a técnica de restrições de justiça que envolve a seleção cuidadosa das variáveis a serem consideradas, a exclusão de informações sensíveis que possam introduzir viés ou a aplicação de técnicas de pré-processamento para equilibrar a representação de diferentes rótulos. Para mitigar o viés são incorporados termos de regularização que penalizam a discriminação ou ajustando os pesos das amostras para equalizar a influência de diferentes rótulos. Em Hardt, Price e Srebro (2016) é proposta uma formulação matemática que aplica as restrições de justiça equilibrando as taxas de falsos negativos entre os rótulos, ajustando limiares de decisão ou a introdução de pesos específicos a fim de compensar as disparidades existentes.

Niculescu-Mizil e Caruana (2005) e Kull, Silva Filho e Flach (2017) propõem a

calibração de confiança, que ajusta as estimativas de probabilidade previstas pelo modelo para que estejam alinhadas com a probabilidade real de ocorrência do evento, visando garantir que as previsões sejam estatisticamente confiáveis, corrigindo tendências sistemáticas. Em todas as técnicas propostas se faz necessário conhecer o espaço amostral das características e as probabilidades *a priori* dos eventos contidos em D.

Muitas técnicas de aprendizado supervisionado trabalham com modelos de programação matemática para minimizar sua função erro $E(.)$, adicionando um elevado custo computacional durante o processo de aprendizagem sob grandes massas de dados. Support Vector Machine (SVM) é uma das técnicas de melhor desempenho na generalização do aprendizado (VAPNIK, V., 1995), mas precisa resolver um modelo de programação quadrática (QP) para encontrar o θ de seu hiperplano de separação. Já a necessidade de definir os motivos que levaram as funções hipóteses $h(.)$ às suas classificações em áreas como saúde e justiça, impulsionou pesquisas recentes para analisar o *tradeoff* entre acurácia e interpretabilidade (BERTSIMAS; DELARUE et al., 2019). Bertsimas, Orfanoudaki e Wiberg (2021) utiliza técnicas de otimização para encontrar soluções que equilibrem a interpretabilidade e a acurácia na clusterização de dados. Vidal e Schiffer (2020) e Parmentier e Vidal (2021) propõem modelos de programação para construir uma única árvore de decisão, que traga explicações contrafactuais fiéis ao comportamento de uma *tree ensemble* fornecida.

Para estas técnicas tratar massa de dados de grande porte se faz necessário criar subamostras de tamanho reduzido que tragam uma fidelidade à acurácia obtida na amostra original. Diversos estudos já foram propostos utilizando características específicas da técnica SVM (BIRZHANDI; KIM; YOUN, 2022). Em Panda, Chang e Wu (2006) uma subamostra é criada com o conjunto de pontos que tenha a menor média de distâncias aos pontos de rótulos distintos. Li, Cervantes e Yu (2012) propõe executar o SVM sob uma subamostra aleatória e filtrar os dados originais que estejam dentro de esferas centralizadas nos vetores de suporte encontrados inicialmente. Já em Zhu et al. (2014), procura-se identificar os pontos na amostra que estão próximos das fronteiras entre as classes guiados pelos centros de massa dos dados em cada rótulo.

Segundo Kish (1965), a estratificação de uma subamostra é um procedimento que visa preservar a representatividade de diferentes grupos ou estratos presentes nos dados, levando em consideração a variabilidade existente em cada característica, procura-se garantir que os estratos na subamostra sejam semelhantes às proporções na amostra. Assim, ao construir uma subamostra D_{sub} inserimos novos termos nos seus erros

$$error_{\text{sub}} = bias + bias_{\text{sub}} + noise_{\text{sub}},$$

sendo *bias* a componente original contida em D que não pode ser quantificado, pois não

sabemos qual a resposta correta ("alvo"), ou seja, não temos informações além de D; a componente $noise_{\text{sub}}$ que é uma parte do ruído herdado de D, este ruído como o original pode ser eliminado com a técnica de regularização; e a componente $bias_{\text{sub}}$ que é um tipo de viés inerente a toda subamostra, mas com uma diferença para a componente *bias* pois o "alvo"é conhecido e está contido em D.

3 Nova formulação proposta (nOCT)

Esta seção descreve a reformulação proposta para o modelo OCT, doravante denominada nOCT. Adicionalmente, após a apresentação do modelo matemático proposto, seguem duas melhorias propostas para este: um novo conjunto de desigualdades válidas baseadas no conceito de Grafo de Precedência, e uma estratégia de *branch-and-cut* para lidar com os grandes conjuntos de restrições do modelo.

O modelo nOCT adota os parâmetros de entrada D, t, $\mathcal{T}_B$, $\mathcal{T}_L$, Y_{ik}, $\hat{L}$ e as variáveis de decisão z_{it}, a_{jt}, b_t, c_{kt} e L_t com o mesmo significado descrito nas Tabelas 1 e 2. Adicionalmente, define-se $L_L(t)$ como o conjunto de nós folha da subárvore esquerda do nó $t \in \mathcal{T}_B$ e $L_R(t)$ como o conjunto de nós folha da subárvore direita ao $t \in \mathcal{T}_B$. O programa (17) a (26) apresenta uma nova formulação para construção de uma árvore de classificação ótima. Na função objetivo (17), o total de pontos mal classificados é computado pelo termo $\sum_{t \in \mathcal{T}_L} L_t$, enquanto que a complexidade da árvore é computada pelo número de divisões na árvore, dado por $\sum_{j=1}^{p} \sum_{t \in \mathcal{T}_B} a_{jt}$.

$$\text{Minimize} \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{j=1}^{p} \sum_{t \in \mathcal{T}_B} a_{jt} \tag{17}$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \qquad i = \{1, .., n\}, \tag{18}$$

$$\sum_{j=1}^{p} a_{jt} = (\leq)1, \qquad \forall t \in \mathcal{T}_B, \tag{19}$$

$$a_t^T x_i + \frac{\epsilon}{2} \leq b_t + 1 - \sum_{\forall l \in L_L(t)} z_{il}, \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_B, \tag{20}$$

$$a_t^T x_i - \frac{\epsilon}{2} \geq b_t - 1 + \sum_{\forall l \in L_R(t)} z_{il}, \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_B, \tag{21}$$

$$\sum_{k=1}^{K} c_{kt} = 1, \qquad \forall t \in \mathcal{T}_L, \tag{22}$$

$$L_t \geq \sum_{\substack{i=1 \\ Y_{ik} \neq 1}}^{n} z_{it} - n(1 - c_{kt}), \qquad k = \{1, .., K\}, \forall t \in \mathcal{T}_L, \tag{23}$$

$$L_t \leq \sum_{\substack{i=1 \\ Y_{ik} \neq 1}}^{n} z_{it} + n(1 - c_{kt}), \qquad k = \{1, .., K\}, \forall t \in \mathcal{T}_L, \tag{24}$$

$$L_t \geq 0, \qquad \forall t \in \mathcal{T}_L, \tag{25}$$

$$0 \leq b_t \leq 1, \qquad \forall t \in \mathcal{T}_B. \tag{26}$$

Para a construção da estrutura da árvore de classificação, temos que o conjunto de restrições (18) garante que cada ponto seja atribuído para uma única folha $t \in \mathcal{T}_L$. As restrições (19) garantem que no máximo um parâmetro seja escolhido para a restrição de divisão de cada nó $t \in \mathcal{T}_B$. Os conjuntos de restrições (20) e (21) garante que o i-ésimo ponto atenda as restrições de divisão ($a^T x_i < b_t$) de todos nós $t \in \mathcal{T}_B$ que estejam no caminho entre o nó raiz e o nó folha l onde $z_{il} = 1$. Se o i-ésimo ponto estiver em uma folha de $L_L(t)$ então $\sum_{\forall l \in L_L(t)} z_{il} = 1$, logo, $a_t^T x_i + \frac{\epsilon}{2} \leq b_t$, caso contrário, como $b_t \in [0, 1]$, então a restrição (20) fica inativa. Se o i-ésimo ponto estiver em $L_R(t)$, então $\sum_{\forall l \in L_R(t)} z_{il} = 1$, logo, $a_t^T x_i - \frac{\epsilon}{2} \geq b_t$, caso contrário, a restrição (21) fica inativa. Além disto, caso o nó de ramificação t não tenha parâmetro atribuído para sua restrição de divisão ($\sum_{j=1}^{p} a_{jt} = 0$), então o conjunto de restrições (21) obriga todos os pontos que passarem por t a descerem na ramificação esquerda, pois se $\sum_{\forall l \in L_R(t)} z_{il} = 1$, então $-\frac{\epsilon}{2} \geq b_t$, o que é impossível, pois $b_t \in [0, 1]$. É atribuído para ϵ o valor da menor distância não zero entre todos os parâmetros de todos os pontos de X.

Para a computação dos pontos mal classificados pela árvore construída, temos que o conjunto de restrições (22) garantem que cada nó folha terá uma única classe k majoritária. As restrições (23) e (24) contabilizam em L_t a quantidade de pontos que não são da classe majoritária ($c_{kt} = 1$). Finalmente, o domínio das variáveis está descrito na Tabela 2.

3.1 Inequações de precedência

Definição 3.1. *(Grafo precedência) É um grafo acíclico orientado $G = (V, A)$, onde cada vértice $v_i \in V$ está associado ao ponto $x_i \in X$, sendo $w : X \times X \to Z_0^+$ a função que indica quantos parâmetros de x_i possuem valores menores ou iguais aos valores dos parâmetros de x_j. Logo, o arco $(i, j) \in A$ se $w(x_i, x_j) = p$, ou seja, o arco (i, j) indica precedência em todos os valores dos parâmetros do ponto x_i ao ponto x_j. Havendo um caminho entre os vértices i e j de tamanho maior que 1 então o arco (i, j) deve ser removido de A.*

A Figura 2 ilustra o grafo de precedência $\hat{G}$ para um conjunto de pontos $\hat{X}$ de exemplo (Figura 2(a)). Na Figura 2(b) é possível perceber que o vértice 2 tem caminho para todos os outros vértices, pois o ponto x_2 possui valores menores ou iguais, em seus parâmetros, a todos os valores dos outros pontos de $\hat{X}$. O arco $(2, 1)$ foi removido de $A[\hat{G}]$ pois existe o caminho $2 \to 3 \to 1$.

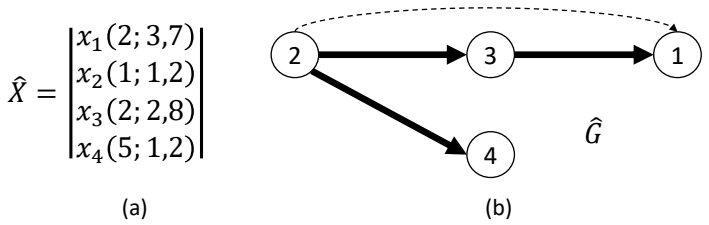


Figura 2: Exemplo do grafo de precedência: (a) Conjunto $\hat{X}$ com 4 pontos, (b) Grafo de precedência $\hat{G} = (V, A)$ dos pontos de $\hat{X}$.

Considerando, sem perda de generalidade, que os índices do conjunto $\mathcal{T}_L$ fazem referência aos nós folha ordenados pela visita à árvore em pré-ordem, e que s e f são os nós folha inicial e final de T_L, respectivamente. Para todo grafo de precedência G, associado a uma instância X, definimos as seguintes inequações de precedência válidas para todo $(i, j) \in A[G]$:

$$z_{if} \leq z_{jf}, \tag{27}$$

$$z_{js} \leq z_{is}, \tag{28}$$

$$z_{jt_j} + z_{it_i} \leq 1, \qquad t_j = \{1, ..., |T_L| - 1\}, t_i = \{t_j + 1, ..., |T_L|\}. \tag{29}$$

Para cada par de pontos $x_i, x_j \in X$ com $(i, j) \in A[G]$ então o conjunto de restrições (27) garantem que se x_i está na última folha f então $z_{jf} = 1$, ou seja, x_j também estará em f, já o conjunto de restrições (28) garantem que se x_j está na primeira folha s então $z_{is} = 1$. Por fim, as restrições (29) garantem que se x_j estiver em qualquer nó $t_j = \{1, ..., |T_L| - 1\}$ então x_i não estará em um nó com índice $t_i > t_j$, e se x_i estiver em t_i então x_j não estará em t_j.

3.2 Algoritmo de *branch-and-cut*

É possível perceber que o principal gargalo computacional para o modelo proposto está no número de pontos (n) usado para a construção da árvore. O número de restrições dos conjuntos (20) e (21) é dependente de n. Por este motivo, optou-se por inserí-las dinamicamente durante a resolução do modelo. Desta forma, este conjunto de restrições é inicialmente relaxado e, para cada solução inteira encontrada durante o processo de *branch-and-bound*, verifica-se se há restrições violadas, inserindo-as no modelo.

O Algoritmo 3 define como as violações são verificadas. Seja ($\bar{z}$, $\bar{a}$, $\bar{b}$) uma solução inteira encontrada, como as restrições (18) garantem que na solução cada ponto x_i está em um único nó folha, fazemos uma varredura para descobrir qual folha f está com valor $\bar{z}_{if} = 1$ (linha 2). Em seguida, procuramos as restrições violadas visitando cada nó ramificação t a partir do pai de f até a raiz da árvore. Em cada nó visitado t, se $f \in L_L(t)$, ou seja, se x_i estiver na subárvore esquerda de t e $\bar{a}_t^T x_i + \frac{\epsilon}{2} > \bar{b}_t$ então adiciona-se a restrição $a_t^T x_i + \frac{\epsilon}{2} \leq b_t + 1 - \sum_{\forall l \in L_L(r)} z_{il}$ no modelo (linhas 5 e 6). Caso $f \in L_R(t)$ e $\bar{a}_t^T x_i - \frac{\epsilon}{2} < \bar{b}_t$ então adicione a restrição $a_t^T x_i - \frac{\epsilon}{2} \geq b_t - 1 + \sum_{\forall l \in L_R(r)} z_{il}$ no modelo (linhas 7 e 8).

Algoritmo 3: separacao_violadas($\bar{z}$, $\bar{a}$, $\bar{b}$)

```
1  para i = 1 to n faça
2  |  f ← encontrar_folha(z̄, i)
3  |  t ← p(f)
4  |  enquanto t ≠ NULL faça
5  |  |  se (f ∈ L_L(t)) e (ā_t^T x_i + ε/2 > b̄_t) então
6  |  |  |  adiciona a_t^T x_i + ε/2 ≤ b_t + 1 − Σ_{∀l∈L_L(r)} z_il
7  |  |  fim
8  |  |  se (f ∈ L_R(t)) e (ā_t^T x_i − ε/2 < b̄_t) então
9  |  |  |  adiciona a_t^T x_i − ε/2 ≥ b_t − 1 + Σ_{∀l∈L_R(r)} z_il
10 |  |  fim
11 |  |  t ← p(t)
12 |  fim
13 fim
```

O Algoritmo 3 tem complexidade $O(n(2^{D-1} + D))$ sendo D a profundidade da

árvore. Quando este algoritmo de separação não encontrar mais nenhuma restrição (20) ou (21) violada, então $(\bar{z}, \bar{a}, \bar{b})$ será considerada uma solução viável e teremos uma árvore de classificação formada.

4 Problema da Subamostra Não-Viciada

Com base nos fundamentos estabelecidos na Seção 2.5, neste capítulo apresentamos métodos para identificar uma subamostra D_{sub} que minimize $bias_{\text{sub}}$ e assim a função h_{sub} aprendida a partir de D_{sub} seja o mais fiel possível da função h de D. Para isto, quantificamos o viés produzido pela subamostragem como

$$bias_{\text{sub}} = \sum_{f \in F} |\bar{x}_f - \mu_f|,$$

onde $\bar{x}_f$ e μ_f são as médias dos valores de cada característica $f \in F$ de D_{sub} e de D, respectivamente. Sendo assim, propomos um problema de otimização que visa encontrar uma subamostra que minimize o valor de $bias_{\text{sub}}$, diminuindo o custo computacional do processo de aprendizagem.

Dada a matriz de características $X \in \mathbb{R}^{n \times d}$ de um conjunto de dados com n pontos e d características em $\mathbb{R}^d$, seja $F = \{1, .., d\}$ o conjunto de índices de características. Além disso, seja μ_f a média do valor da f-ésima característica $X_{:f}$, para todos os $f \in F$. O Problema da Subamostra Não-Viciada (do inglês *Unbiased Subsample Problem*, USP) tem como objetivo encontrar uma subamostra $X_{\text{sub}} \subset X$ contendo $\bar{n}$ pontos que minimize a expressão $\sum_{f \in F} |\bar{x}_f - \mu_f|$. Aqui, $\bar{x}_f$ representa a média do valor da f-ésima característica na subamostra X_{sub}.

Assim sendo, nesta seção propomos resolver o USP primeiro por uma modelagem de programação linear inteira; em seguida, adaptando a meta-heurística populacional BRKGA (GONÇALVES; RESENDE, 2011), do inglês *Biased Random-Key Genetic Algorithm*, através da definição de sua função decodificadora; e, finalmente, por uma adaptação da meta-heurística de trajetória ILS (LOURENÇO; MARTIN; STÜTZLE, 2003), do inglês *Iterated Local Search*, com a definição de três movimentos de vizinhança e um de perturbação.

Adotamos a matriz padronizada $Z \in \mathbb{R}^{n \times d}$ como entrada para as meta-heurísticas propostas. Representamos uma subamostra solução $Z_{\text{sub}} \subset Z$ como um conjunto S de inteiros que representa um subconjunto dos índices dos pontos de Z. A subamostra S tem tamanho $\bar{n} = |S|$ não especificado, e limitado pelo intervalo $LB_{\bar{n}} \leq \bar{n} \leq UB_{\bar{n}}$. Definimos a função avaliação da solução S como

$$f(S) = \sum_{f \in F} |\sum_{i \in S} Z_{if}|. \quad \text{(Proposição 4.1)}$$

4.1 Modelo de Programação Inteira

Para modelar o USP definimos $b_f \in \mathbb{R}$ como a variável que mede o viés da característica $f \in F$ na subamostra X_{sub} e $w_i \in \{0, 1\}$ a variável que indica, caso $w_i = 1$, que o ponto $i \in X$ está contido em X_{sub}.

$$\text{Minimize} \quad USP = \sum_{f \in F} b_f \tag{30}$$

$$\text{s.t.} \quad b_f \geq |\bar{x}_f - \mu_f|, \qquad \forall f \in F, \tag{31}$$

$$\sum_{i}^{n} w_i = \bar{n}, \tag{32}$$

$$w \in \{0, 1\}^n, b \in \mathbb{R}^d. \tag{33}$$

A função objetivo (30) busca minimizar o viés de cada característica f em X_{sub}. O conjunto de restrições (31) define, para toda característica $f \in F$, o valor mínimo de b_f como a diferença em módulo do valor médio $\bar{x}_f = \sum_{i=1}^{n} \frac{X_{if}}{\bar{n}} \cdot w_i$ de X_{sub} e μ_f. A restrição (32) garante que X_{sub} tenha exatamente $\bar{n}$ pontos na solução. E por fim as restrições (33) definem o domínio das variáveis w e b.

Para linearizar o conjunto de restrições (31), removemos sua função módulo e substituímos pelas inequações:

$$b_f \geq (\sum_{i=1}^{n} \frac{X_{if}}{\bar{n}} \cdot w_i) - \mu_f, \qquad \forall f \in F, \tag{34}$$

$$b_f \geq \mu_f - (\sum_{i=1}^{n} \frac{X_{if}}{\bar{n}} \cdot w_i), \qquad \forall f \in F. \tag{35}$$

Propomos uma variação do USP onde, ao invés de fornecer o tamanho específico da subamostra a ser encontrada, fornecemos um intervalo de valores $LB_{\bar{n}} \leq \bar{n} \leq UB_{\bar{n}}$ possíveis para o tamanho $\bar{n}$ da subamostra solução. Podemos modelar esta variação do USP da seguinte forma:

$$\text{Minimize} \quad [USP] = \sum_{f \in F} b_f \tag{36}$$

$$\text{s.t.} \quad b_f \geq \frac{(\sum_{i=1}^{n} X_{if} \cdot w_i)}{\sum_i^n w_i} - \mu_f, \qquad \forall f \in F, \tag{37}$$

$$b_f \geq \mu_f - \frac{(\sum_{i=1}^{n} X_{if} \cdot w_i)}{\sum_i^n w_i}, \qquad \forall f \in F. \tag{38}$$

$$\bar{n}_{LB} \leq \sum_{i}^{n} w_i \leq \bar{n}_{UB}, \tag{39}$$

$$w \in \{0,1\}^n, b \in \mathbb{R}^d. \tag{40}$$

A função objetivo (36) continua minimizando o viés da amostra em cada característica $f \in F$. As restrições (39) garantem o intervalo de valores para o tamanho $\bar{n}$ da subamostra solução. Como agora $\bar{n}$ é variável, o conjunto de restrições (34) e (35) foi reformulado no conjunto de restrições (37) e (38), substituindo o termo $(\sum_{i=1}^{n} \frac{X_{if}}{\bar{n}} \cdot w_i)$ por $\frac{(\sum_{i=1}^{n} X_{if} \cdot w_i)}{\sum_i^n w_i}$ o que torna estas restrições não lineares. Utilizaremos a Proposição 4.1 a seguir para linearizar estas restrições.

Proposição 4.1. *Seja a variável aleatória x com média μ, desvio padrão σ, sua respectiva variável padronizada $z = \frac{x-\mu}{\sigma}$ e uma subamostra qualquer x_{sub}. A soma $\sum z_{sub} \to 0$ se e somente se a média $\bar{x}_{sub} \to \mu$.*

Demonstração. ($\Rightarrow$) Considere $f(x_{\text{sub}})$ como a média dos valores da subamostra x_{sub}, logo, temos

$$f(x_{\text{sub}}) = \frac{\sum x_{\text{sub}}}{n_{\text{sub}}} = \frac{\sum(\sigma \cdot z_{\text{sub}} + \mu)}{n_{\text{sub}}} = \frac{\sigma(\sum z_{\text{sub}})}{n_{\text{sub}}} + \mu,$$

e aplicando o limite para $\sum z_{\text{sub}} \to 0$, temos

$$\lim_{\sum z_{\text{sub}} \to 0} f(x_{\text{sub}}) = \lim_{\sum z_{\text{sub}} \to 0} \frac{\sigma(\sum z_{\text{sub}})}{n_{\text{sub}}} + \mu = \mu.$$

($\Leftarrow$) Considere $f(z_{\text{sub}})$ como a soma dos valores padronizados da subamostra x_{sub}, logo, temos

$$f(z_{\text{sub}}) = \sum z_{\text{sub}} = \sum(\frac{x_{\text{sub}} - \mu}{\sigma}) = \frac{(\sum x_{\text{sub}}) - n_{\text{sub}} \cdot \mu}{\sigma},$$

fazendo $\sum x_{\text{sub}} = n_{\text{sub}} \cdot \bar{x}_{\text{sub}}$, temos $f(z_{\text{sub}}) = \frac{n_{\text{sub}}(\bar{x}_{\text{sub}} - \mu)}{\sigma}$, e aplicando o limite para

$\bar{x}_{\text{sub}} \to \mu$, temos

$$\lim_{\bar{x}_{\text{sub}} \to \mu} f(z_{\text{sub}}) = \lim_{\bar{x}_{\text{sub}} \to \mu} \frac{n_{\text{sub}}(\bar{x}_{\text{sub}} - \mu)}{\sigma} = 0.$$

□

O objetivo do USP é encontrar uma subamostra $X_{\text{sub}} \subset X \in \mathbb{R}^{n \times d}$ onde para todo $f \in F$ temos que $\bar{x}_f \to \mu_f$. Considerando cada vetor de características (colunas da matriz X) uma variável x e aplicando a operação de padronização $z = \frac{x-\mu}{\sigma}$ obtemos a matriz padronizada $Z \in \mathbb{R}^{n \times d}$. Sendo assim, da Proposição 4.1 também podemos afirmar que o objetivo do USP é encontrar uma subamostra $Z_{\text{sub}} \subset Z$ onde para todo $f \in F$ temos que $\sum Z_{\text{sub}f} \to 0$.

Para exemplificar este conceito apresentamos na Figura 3 a matriz exemplo $\hat{X} \in \mathbb{R}^{4 \times 3}$ que possui 4 pontos e 3 características, sendo $\mu = [0,5; 2,5; 170]$ e $\sigma = [0,5; 1,11; 17,68]$, respectivamente, os vetores de médias e desvios padrão das 3 características de X, e sua matriz padronizada $\hat{Z} \in \mathbb{R}^{4 \times 3}$. Sejam $S_1 = \{1, 2\}$ e $S_2 = \{2, 4\}$ conjuntos de índices de pontos de $\hat{X}$ representando duas subamostra com $\bar{n} = 2$, logo, para S_1 temos que $\bar{x} = [0,5; 1,5; 152,5]$ representa suas médias amostrais, e computando seu viés temos $\sum_{f \in \{1,2,3\}} |\bar{x}_f - \mu_f| = 0 + 1 + 17,5 = 18,5$. Já para a subamostra S_2 temos $\bar{x} = [0,5; 2,5; 170]$ e computando seu viés temos $\sum_{f \in \{1,2,3\}} |\bar{x}_f - \mu_f| = 0 + 0 + 0 = 0$, o qual é uma solução ótima. Aplicando a Proposição 4.1 podemos computar o viés de S_1 como $\sum_{f \in \{1,2,3\}} |\sum_{i \in S_1} \bar{Z}_{if}| = 0 + |-1,79| + |-1,98| = 3,77$, já para a subamostra S_2 temos $\sum_{f \in \{1,2,3\}} |\sum_{i \in S_2} \bar{Z}_{if}| = 0 + 0 + 0 = 0$. É importante observar que a adoção da matriz $\hat{Z}$ muda apenas a escala dos valores dos vieses computados em S_1 e S_2 mantendo o comportamento relativo entre as soluções, não interferindo assim no objetivo de encontrar a subamostra com viés minimizado.

$$\hat{X} = \begin{pmatrix} f_1 & f_2 & f_3 \\ [1 & 1 & 155] \\ [0 & 2 & 150] \\ [0 & 3 & 185] \\ [1 & 4 & 190] \end{pmatrix} \qquad \hat{Z} = \begin{pmatrix} f_1 & f_2 & f_3 \\ [+1 & -1{,}34 & -0{,}85] \\ [-1 & -0{,}45 & -1{,}13] \\ [-1 & +0{,}45 & +0{,}85] \\ [+1 & +1{,}34 & +1{,}13] \end{pmatrix}$$

Figura 3: Matriz $\hat{X} \in \mathbb{R}^{4 \times 3}$ exemplo e sua matriz com características padronizadas $\hat{Z} \in \mathbb{R}^{4 \times 3}$.

Por fim, fornecendo ao modelo $[USP]$ a matriz padronizada $Z \in \mathbb{R}^{n \times d}$ temos para todo $f \in F$ que $\mu_f = 0$. E pela Proposição 4.1 podemos reformular as restrições (37) e (38) como

$$b_f \geq \sum_{i \in U} Z_{i,f} \cdot w_i, \qquad \forall f \in F, \tag{41}$$

$$b_f \geq -\sum_{i \in U} Z_{i,f} \cdot w_i, \qquad \forall f \in F, \tag{42}$$

que definem, para cada característica $f \in F$, o limite inferior de b_f como a soma dos valores padronizados $Z_{:f} \cdot w$ da subamostra solução.

4.2 BRKGA para o USP

A meta-heurística BRKGA proposta por Gonçalves e Resende (2011) é um Algoritmo Genético (HOLLAND, 1992) cujos cromossomos são vetores de números aleatórios no intervalo $[0, 1[$ (*random key*). É um algoritmo iterativo (gerações) que manipula uma população de cromossomos com o objetivo de explorar o espaço de soluções do problema específico guiados por uma função de avaliação (*fitness*). A população inicial consiste em p cromossomos cujas chaves aleatórias são geradas independentes entre si. Em cada geração k os p cromossomos são avaliados com a função *fitness*, para assim classificar os indivíduos em dois grupos: um pequeno grupo de p_e indivíduos elite, ou seja, aqueles que apresentaram os melhores valores da função *fitness*, e o grupo remanescente de $p - p_e$ indivíduos não-elite. A população da geração $k + 1$ é construída com: os p_e cromossomos elite; p_m indivíduos mutantes gerados da mesma forma que a população inicial e $p - p_e - p_m$ indivíduos criados a partir da combinação de um cromossomo elite com outro cromossomo não-elite (*crossover*). Ao final de todas as gerações o melhor cromossomo construído é retornado pelo método.

Até aqui descrevemos o algoritmo genético básico do BRKGA, que é o mesmo para qualquer problema, entretanto, para adaptar ao USP um decodificador deve ser definido. Este decodificador é um algoritmo determinístico que tem como entrada a chave aleatória (cromossomo), constrói uma solução viável no espaço de soluções do problema específico, de tal maneira, que seja possível computar a função *fitness* desta solução.

Para definir o decodificador para o USP definimos o cromossomo A de dimensão n, onde a chave $A_j \in [0, 1[$ está associada ao ponto de índice j pertencente a matriz padronizada Z. O Algoritmo 4 define a função decodificadora que inicialmente cria um vetor I dos índices j dos pontos da matriz Z ordenado pelos valores A_j de forma não-crescente. Nas linhas 4-7, para todo valor $\bar{n} \in [LB_{\bar{n}}, UB_{\bar{n}}]$ uma subamostra solução é formada pelo subconjunto de índices $S \subset I$ contendo os $\bar{n}$ primeiros pontos de I, avaliados pela função $f(S)$. Ao final, é retornado como valor da função *fitness* do cromossomo A o menor valor $f(S)$ entre todas as subamostras S construídas.

Algoritmo 4: Função decodificadora que computa a função *fitness* de uma solução do USP mapeada a partir de uma chave aleatória do BRKGA. A: chave aleatória do BRKGA.

```
1  função Decodificador(A)
2  |  I ← argsort(A, decrescente)
3  |  menor_vies ← ∞
4  |  para n̄ ← LB_n̄ até UP_n̄ faça
5  |  |  S ← I_:n̄
6  |  |  se f(S) < menor_vies então
7  |  |  |  menor_vies ← f(S)
8  |  |  fim
9  |  fim
10 |  retorna menor_vies
11 end
```

4.3 ILS para o USP

A meta-heurística ILS possui quatro componentes: uma solução inicial, um procedimento de busca local, uma estratégia de perturbação e um critério de parada. O Algoritmo 5 apresenta a arquitetura de alto nível, definida por Lourenço, Martin e Stützle (2003), que contempla a iteração destes componentes no ILS.

Algoritmo 5: Algoritmo genérico da meta-heurística *Iterated Local Search* definido por Lourenço, Martin e Stützle (2003).

```
1  função ILS()
2  |  s_0 ← GeraSolucaoInicial()
3  |  s* ← BuscaLocal(s_0)
4  |  repita
5  |  |  s' ← Perturbacao(s*, histórico)
6  |  |  s'' ← BuscaLocal(s')
7  |  |  s* ← CriterioAceitacao(s*, s'', histórico)
8  |  até condição de término alcançada
9  |  retorna s*
10 end
```

Seja I o conjunto de índices dos pontos da matriz Z, o ILS proposto para o USP constrói uma solução inicial s_0 composta de $LB_{\bar{n}}$ índices de I escolhidos de forma aleatória (distribuição uniforme). Em seguida é aplicado à s_0 o procedimento de *BuscaLocal* que visa melhorar a solução dada através da exploração de soluções vizinhas melhores, este procedimento será detalhado a frente. De posse da solução s^*, produto da fase de busca local, iniciamos o laço iterativo do algoritmo cuja condição de parada é executar *max_iter* iterações. Em cada iteração aplicamos o procedimento de *Perturbacao* sobre s^* criando uma nova solução s' composta de 70% dos índices de s^* e 30% dos índices de $I \setminus s^*$. Com o intuito de introduzir variabilidade nas soluções iniciais, é possível que a solução s' tenha uma quantidade de índices diferente de s^*, sendo assim, adicionamos na função de

avaliação $f(S)$ uma penalidade para inviabilizar a solução cujo $|S| > UB_{\bar{n}}$ ou $|S| < LB_{\bar{n}}$. Aplicamos sobre s' o mesmo procedimento de *BuscaLocal* gerando a solução s''. Para atualizar s^* para a próxima iteração, o procedimento *CriterioAceitacao* contém um contador *pert_cont* do número de perturbações realizadas, se *pert_cont* $\leq$ *max_pert* então $s^* \leftarrow s''$, sendo *max_pert* o parâmetro que indica quantas perturbações sobre a mesma trajetória de solução podem ser aplicadas, e se *pert_cont* $>$ *max_pert* então $s^* \leftarrow s_{melhor}$ e *pert_cont* $\leftarrow 0$, sendo s_{melhor} a melhor solução encontrada até o momento. Ao final das iterações o procedimento retorna s_{melhor}.

Fase de Busca Local

Seja S uma solução qualquer para o USP e $\bar{S} = I \setminus S$ o conjunto de índices de I que não estão em S. Para a fase de busca local do ILS definimos um conjunto de movimentos de vizinhanças sobre uma solução S, estes movimentos criam uma solução cópia S', aplicando as operações descritas a seguir:

- add(S) : adiciona em S' um índice $i \in \bar{S}$;
- drop(S) : remove de S' um índice $j \in S$;
- swap(S) : remove de S' um índice $k \in S$ e adiciona em S' um índice $l \in \bar{S}$.

Para avaliar a solução S' formada podemos utilizar a função $f(S')$; entretanto, para diminuir o tempo computacional desta avaliação, propomos a adoção do vetor $bias^S \in \mathbb{R}^d$ para cada solução S onde para todo $f \in F$ temos que $bias_f^S = \sum_{i \in S} Z_{if}$. Logo, para computar o valor de $f(S')$ para cada movimento temos

$$f(S') = \begin{cases} \sum_{f \in F} |bias_f^S + Z_{if}| & \text{se add}(S) \\ \sum_{f \in F} |bias_f^S - Z_{jf}| & \text{se drop}(S) \\ \sum_{f \in F} |bias_f^S - Z_{kf} + Z_{lf}| & \text{se swap}(S), \end{cases}$$

tendo este cálculo um tempo computacional $O(d)$. Sendo assim, para computar toda a vizinhança do movimento *add* temos a complexidade $O(|\bar{S}| \cdot d)$, para o movimento *drop* temos $O(|S| \cdot d)$ e *swap* temos $O(|\bar{S}| \cdot |S| \cdot d)$.

O procedimento de *BuscaLocal* proposto define um conjunto $\mathcal{N}$ de movimentos de vizinhança, recebe como entrada uma solução S e a cada iteração explora de forma exaustiva todos os vizinhos de S definidos pelos movimentos de vizinhança de $\mathcal{N}$, finalizando a iteração com a melhor solução S_{bl} encontrada, se $f(S_{bl}) < f(S)$ então $S \leftarrow S_{bl}$ e uma nova iteração é iniciada, caso contrário o algoritmo é finalizado e retorna a solução S.

Definimos o operador $A \oplus B$ como a concatenação de dois movimentos de vizinhança. Por exemplo, no movimento $swap \oplus add$ a solução S é fornecida para o movimento $swap$ que retornará a solução S' que, por sua vez, será fornecida como entrada para o movimento add que retorna a solução S'', sendo S'' a solução da operação $swap \oplus add$. Definimos também três conjuntos de movimentos de vizinhança com distintas complexidades no número de soluções contidas em sua vizinhanças, o $\mathcal{N}_1 = \{add, drop\}$ possui $O(n)$ soluções vizinhas; o $\mathcal{N}_2 = \mathcal{N}_1 \cup \{swap, add \oplus add, drop \oplus drop\}$ possui $O(n^2)$ soluções vizinhas e o $\mathcal{N}_3 = \mathcal{N}_2 \cup \{swap \oplus add, swap \oplus drop\}$ possui $O(n^3)$ soluções vizinhas.

Por fim, propomos para o ILS o parâmetro *tempo_estimado* que define um tempo limite desejado para a finalização do algoritmo, mas não aplicado como uma condição de parada estrita. Para convergir o tempo de finalização do algoritmo ao *tempo_estimado*, iniciamos o procedimento de *BuscaLocal* com o conjunto $\mathcal{N} \leftarrow \mathcal{N}_3$. A cada 5 iterações pegamos o tempo de execução atual e computamos uma estimativa linear para as próximas iterações, se esta estimativa for menor que o valor *tempo_estimado*, continuamos com o mesmo $\mathcal{N}$, caso contrário fazemos $\mathcal{N} \leftarrow \mathcal{N}_2$, após outras 5 iterações se o tempo estimado continuar superior ao valor *tempo_estimado* fazemos $\mathcal{N} \leftarrow \mathcal{N}_1$, a partir deste ponto o conjunto $\mathcal{N}$ ficará fixo até o fim do algoritmo.

4.4 BRKGA e ILS Híbrido

Enquanto o ILS proposto encontra de forma relativamente rápida ótimos locais devido ao seu mecanismo de exploração de vizinhanças, a sua busca do espaço completo de soluções é negativamente afetada pela simplicidade e dependência da aleatoriedade e solução anterior dos seus mecanismos de pertubação e reinicio. Por outro lado, o BRKGA sofre o oposto: Enquanto seus mecanismos de construção, cruzamento e mutação são robustos e exploram bem o espaço de soluções, a ausência de exploração de vizinhança exige que esses três últimos mecanismos assumam o papel de encontrar ótimos locais, o que torna a convergência significativamente mais lenta.

Assim sendo, duas abordagens são tipicamente utilizadas nesse caso: A primeira consiste em aplicar uma busca local, construída separadamente do BRKGA e específica ao problema tal como o decodificador, em cada indivíduo gerado pelo BRKGA. Na segunda, aplica-se a busca local apenas em alguns indivíduos, de forma a poupar recursos computacionais. Considerando o problema em questão, o Algoritmo 6 apresenta a solução Híbrida do BRKGA e ILS proposta neste trabalho utilizando a segunda abordagem discutida, que intercala as execuções das duas heurísticas com o intuito de combinar os seus benefícios.

Algoritmo 6: Algoritmo Híbrido combinando BRKGA e ILS.

```
1  função Hybrid(n_iter_iniciais, n_repeticoes, n_iter_ils, n_iter_brkga)
2      s* ← n_iter_iniciais gerações de BRKGA()
3      para n_repeticoes faça
4          Defina s* como a solução inicial do ILS
5          s* ← n_iter_ils reinícios de ILS()
6          Injete s* na população do BRKGA
7          s* ← n_iter_brkga gerações de BRKGA()
8      fim
9      retorna s*
10 end
```

5 Usando o USP para resolver o OCT

O modelo OCT proposto em Bertsimas e Dunn (2017) constrói uma árvore de classificação de altura D cujo erro de classificação sobre os dados de treinamento (E_{in}) é minimizado. O número de restrições do modelo depende da quantidade n de amostras da base de treinamento e do valor de D. Para instâncias com n de porte médio, construir uma árvore com $D \geq 3$ é caro em termos computacionais e impraticável com $D \geq 4$.

Para contornar estas limitações propomos um algoritmo iterativo, doravante denominado OCT de topologia dinâmica (OCT_{din}), que procura identificar, por meio de um processo de validação, a topologia da árvore de classificação que melhor se ajuste aos dados. Em cada iteração usamos a heurística USP para encontrar a subamostra de tamanho mínimo necessária para a complexidade da topologia corrente sem perda na generalização do aprendizado.

Para isto, definimos a topologia $\mathcal{T} = \{t_1, .., t_k\}$ de uma árvore de classificação binária como um conjunto de nós ramificação e nós folha (classificação) e sua disposição hierárquica, cada nó de ramificação t possui dois filhos que pode ser outro nó ramificação ou nó folha. Na topologia não definimos, para os nós de ramificação, qual característica nem o valor limiar a ser testado ao decidir em qual subárvore as amostras descerão, nem definimos qual a classe majoritária de cada nó folha.

A Figura 4 ilustra alguns exemplos de topologia de árvores de decisão binária, como, por exemplo, a topologia $\mathcal{T}_1$ composta apenas por um nó de ramificação (raiz) e seus dois nós folha, e a topologia $\mathcal{T}_2$ que além do nó raiz tem seu filho a esquerda como nó de ramificação e o filho a direita como nó folha. As topologias podem ser desbalanceadas como as topologias $\mathcal{T}_2$, $\mathcal{T}_4$, $\mathcal{T}_5$ e $\mathcal{T}_6$, e completas como a topologia $\mathcal{T}_3$ ($D = 2$) e $\mathcal{T}_7$ ($D = 3$).

Como todo procedimento de aprendizado supervisionado, recebemos como entrada a base de dados rotulada (X, y) que, por sua vez, é divida em uma base $(\hat{X}, \hat{y})$ com 80% da base original e a base de teste (X_{test}, y_{test}) contendo o restante dos dados. Esta divisão se

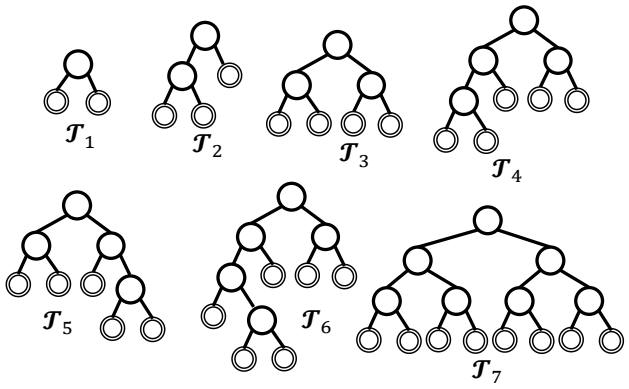


Figura 4: Exemplos de topologias de árvores binárias de decisão.

faz necessária para computar o quanto a árvore construída a partir de $(\hat{X}, \hat{y})$ generalizou seu aprendizado na base (X_{test}, y_{test}). Mas para o processo de escolha de qual topologia da árvore de classificação se ajustará melhor aos dados, precisamos dividir a base $(\hat{X}, \hat{y})$ na base de treinamento (X_{train}, y_{train}) contendo 80% e (X_{val}, y_{val}) contendo 20%.

Inicialmente, começamos o procedimento com a topologia mais simples possível contendo apenas o nó raiz como ramificação e seus nós folha (topologia $\mathcal{T}_1$ da Figura 4). Cada iteração, ilustrada na Figura 5, é dividida em 4 fases:

- Na primeira, guiado pelo conceito da dimensão VC, identificamos um intervalo $[LB_{\bar{n}}, UB_{\bar{n}}]$ para quantidade de amostras necessárias para que uma árvore de classificação com a complexidade da topologia $\mathcal{T}$ corrente possa generalizar seu aprendizado;

- Na segunda, a heurística USP encontrará uma subamostra $(\bar{X}, \bar{y})$ contendo $\bar{n} \in [LB_{\bar{n}}, UB_{\bar{n}}]$ amostras de (X_{train}, y_{train}) minimizando seu viés de amostragem;

- Em seguida, o modelo OCT_{top} reformulado para construir árvores de classificação ótimas com topologia fixa infere a árvore de classificação $\bar{h}$ da base $(\bar{X}, \bar{y})$;

- E por fim, na quarta fase, duas condições de parada do algoritmo são analisadas: Na primeira, com auxílio da base (X_{val}, y_{val}) testamos a existência de *overfitting* em $\bar{h}$; se existir, finalizamos o procedimento e retornamos h_{best} que, a árvore de classificação com menor erro de classificação sobre a base (X_{val}, y_{val}) entre todas as iterações; se não houver *overfitting*, avançamos para a segunda condição de parada que consiste em analisar a necessidade de aumentar a complexidade da topologia $\mathcal{T}$, guiado por uma função taxa de erro para cada nó folha de $\mathcal{T}$. Não havendo mais ramificações a serem feitas, o algoritmo finaliza retornando h_{best}; havendo novas ramificações a topologia $\mathcal{T}$ é atualizada e uma nova iteração é reiniciada.

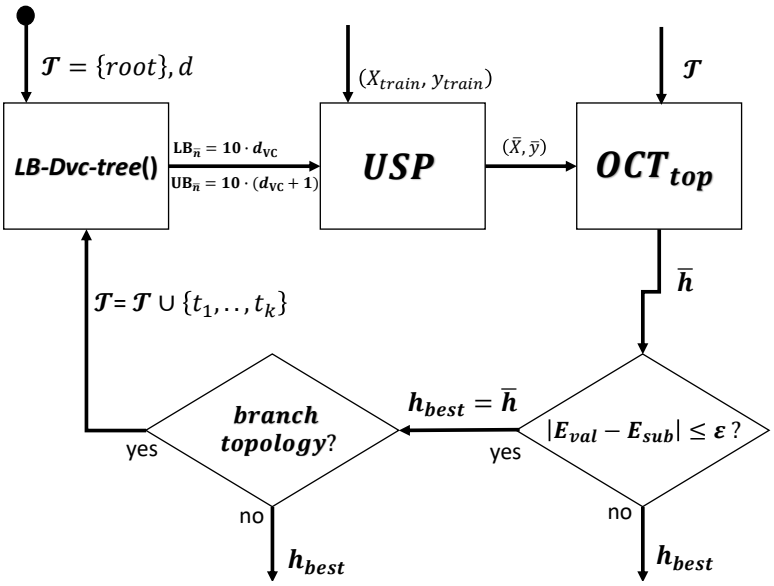


Figura 5: Esquema para resolver o OCT com o auxílio do USP.

Maiores detalhes das fases deste procedimento proposto estão descritos nas seções seguintes.

5.1 Definição do tamanho da subamostra para o USP

A dimensão VC (d_{VC}) (VAPNIK, V. N., 2013), é a principal medida teórica de complexidade de um modelo de aprendizado, e uma boa regra prática diz que ter pelo menos $10 \times d_{\text{VC}}$ amostras para treinar o modelo é suficiente para encontrar uma função que esteja ϵ-próxima da melhor função de classificação para este modelo.

Propomos utilizar os conceitos da topologia de uma árvore de classificação binária e dimensão VC para identificar a quantidade mínima de dados que a subamostra $(\bar{X}, \bar{y})$ precisa ter para garantir a generalização do aprendizado da árvore de classificação inferida. Para isto, adotamos o resultado apresentado no trabalho de Yildiz (2015). Neste, são propostos limites inferiores da dimensão VC de árvores de classificação univariada. Este limite foi comparado a um algoritmo de busca que calcula exaustivamente a dimensão VC de árvores de decisão, mostrando que seus limites propostos são precisos para árvores com topologia simples.

O Algoritmo 7, proposto por Yildiz (2015), computa a dimensão VC para árvores de classificação inferidas a partir de dados contínuos, recebendo como entrada a topologia $\mathcal{T}$ e o número de características dos dados d. Para cada nó ramificação da topologia que contenha duas folhas como nós filho adiciona-se $\lfloor \log_2(d) \rfloor + 1$ à d_{VC}, se tiver apenas um nó folha como filho adiciona-se 1 à d_{VC}, não havendo nó folha como filho nada é adicionado.

Dados $\mathcal{T}$ e d, computamos o intervalo do tamanho da subamostra como $LB_{\bar{n}} = 10\times$ LB-Dvc-tree$(\mathcal{T}, d)$ e $UB_{\bar{n}} = 10 \times (\text{ LB-Dvc-tree}(\mathcal{T}, d) + 1)$, e fornecemos este intervalo

Algoritmo 7: O pseudocódigo do algoritmo recursivo para encontrar um limite inferior da dimensão VC de uma árvore de classificação univariada para dados contínuos. $\mathcal{T}$: topologia da hipótese da árvore de classificação, d: número de características dos dados.

função $d_{\texttt{VC}}$ `LB-Dvc-tree`$(\mathcal{T}, d)$
  se $\mathcal{T}$ *is a leaf node* **então**
    retorna 1
  fim
  se *left and right subtrees of* $\mathcal{T}$ *are leaves* **então**
    retorna $\lfloor \log_2(d) \rfloor + 1$
  fim
  $\mathcal{T}_L \leftarrow$ left subtree of $\mathcal{T}$
  $\mathcal{T}_R \leftarrow$ right subtree of $\mathcal{T}$
  retorna LB-Dvc-tree$(\mathcal{T}_L, d)$ + LB-Dvc-tree$(\mathcal{T}_R, d)$
end

juntamente com a base de treinamento (X_{train}, y_{train}) à heurística USP que extrairá a subamostra $(\bar{X}, \bar{y})$ de tamanho $\bar{n} \in [LB_{\bar{n}}, UB_{\bar{n}}]$ com viés minimizado em relação à (X_{train}, y_{train}).

5.2 Modelo OCT de topologia fixa

Finalmente, o modelo OCT_{top} trabalha com a construção de uma árvore de classificação, inferida da base de treinamento, contendo uma topologia $\mathcal{T}$ fixa fornecida como entrada, ou seja, todos os nós ramificação e nós folhas de $\mathcal{T}$ estarão presentes na solução. A topologia é representada pelos parâmetros $\mathcal{T}_B$ que é a lista de nós ramificação, $\mathcal{T}_L$ que é a lista de nós folha, $L_L(t)$ como o conjunto de nós folha da subárvore esquerda do nó $t \in \mathcal{T}_B$ e $L_R(t)$ como o conjunto de nós folha da subárvore direita ao $t \in \mathcal{T}_B$. Como representação da base de treinamento, temos um vetor de n pontos, x, classificados com os rótulos do conjunto K, mapeados pelo parâmetro binário Y_{ik} que indica se o ponto x_i é classificado com o rótulo $k \in K$ caso $Y_{ik} = 1$. Cada ponto x_i possui p características normalizadas, ou seja, para todo $j \in \{1, .., p\}$ temos $x_{ij} \in [0, 1]$.

Definimos as seguintes variáveis binárias de decisão $z_{il} = 1$ indicando que o ponto x_i está no nó folha $l \in \mathcal{T}_L$; $a_{jt} = 1$ indicando que a característica $j \in \{1, .., p\}$ é utilizada pelo nó de ramificação $t \in \mathcal{T}_B$ e $c_{kl} = 1$ indicando que o rótulo $k \in K$ é utilizado na classificação dos pontos atribuídos ao nó folha l. E por fim, a variável contínua $b_t \in [0, 1]$ representa o valor limiar das restrições de divisão $a^T x_i < b_t$ de cada nó de ramificação t.

$$OCT_{top} = \text{Maximize} \quad \sum_{i=1}^{n} \sum_{l \in \mathcal{T}_L} z_{il} \tag{43}$$

$$s.t. \quad \sum_{l \in \mathcal{T}_L} z_{il} \leq 1, \qquad i = \{1, .., n\}, \tag{44}$$

$$\sum_{k=1}^{K} c_{kl} = 1, \qquad \forall l \in \mathcal{T}_L, \tag{45}$$

$$\sum_{\substack{i=1 \\ Y_{ik}=1}}^{n} z_{il} \leq n_k \cdot c_{kl}, \qquad k = \{1, .., K\}, \forall l \in \mathcal{T}_L, \tag{46}$$

$$\sum_{j=1}^{p} a_{jt} = 1, \qquad \forall t \in \mathcal{T}_B, \tag{47}$$

$$a_t^T x_i + \frac{\epsilon}{2} \leq b_t + 1 - \sum_{\forall l \in L_L(t)} z_{il}, \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_B, \tag{48}$$

$$a_t^T x_i - \frac{\epsilon}{2} \geq b_t - 1 + \sum_{\forall l \in L_R(t)} z_{il}, \qquad i = \{1, .., n\}, \forall t \in \mathcal{T}_B, \tag{49}$$

$$0 \leq b_t \leq 1, \qquad \forall t \in \mathcal{T}_B. \tag{50}$$

A função objetivo (43) procura maximizar o total de pontos corretamente classificados nas folhas de $\mathcal{T}_L$. Para a construção da estrutura da árvore de classificação, o conjunto de restrições (44) garante que cada ponto do vetor x será atribuído a no máximo um nó folha $l \in \mathcal{T}_L$. Para cada nó folha $l \in \mathcal{T}_L$ temos que l tem uma única classe $k \in K$ para classificar todos seus pontos atribuídos (restrições (45)) e, sendo n_k o número de pontos em x rotulados com k, então apenas os pontos x_i rotulados com k ($Y_{ik} = 1$) podem ser atribuídos à l caso $c_{kl} = 1$ (restrições (46)). Já as restrições (47) garantem que no máximo uma característica j dos dados seja escolhido para a restrição de divisão de cada nó $t \in \mathcal{T}_B$. Os conjuntos de restrições (48) e (49) garantem que o i-ésimo ponto atenda as restrições de divisão ($a^T x_i < b_t$) de todos os nós $t \in \mathcal{T}_B$ que estejam no caminho entre o nó raiz e o nó folha l onde $z_{il} = 1$. Se o i-ésimo ponto estiver em uma folha de $L_L(t)$ então $\sum_{\forall l \in L_L(t)} z_{il} = 1$, logo, $a_t^T x_i + \frac{\epsilon}{2} \leq b_t$, caso contrário, como $b_t \in [0, 1]$, então a restrição (48) fica inativa. Se o i-ésimo ponto estiver em $L_R(t)$, então $\sum_{\forall l \in L_R(t)} z_{il} = 1$, logo, $a_t^T x_i - \frac{\epsilon}{2} \geq b_t$, caso contrário, a restrição (49) fica inativa. É atribuído para ϵ o valor da menor distância não zero entre todos os parâmetros de todos os pontos do vetor x. Finalmente, o domínio da variável b_t está descrito nas restrições (50).

5.3 Generalização do aprendizado

O Algoritmo *minimal cost-complexity prunning* (BREIMAN et al., 1984) é um dos procedimentos mais eficientes de regularização para uma árvore decisão T. Parametrizado em α ele procura penalizar a complexidade de T representada pelo seu número de nós folha $|\tilde{T}|$ através da seguinte função erro aumentada

$$E_{aug}(T) = E_{in}(T) + \alpha|\tilde{T}|,$$

sendo $E_{in}(T)$ o erro de classificação de T nos dados de treinamento. O aumento de α penaliza a quantidade de folhas de T, diminuindo sua complexidade e, consequentemente, o grau de liberdade de aprendizado de T, logo, aumentando o $E_{in}(T)$.

A Figura 6 apresenta a relação entre o parâmetro α da regularização e os erros de aprendizado obtidos nos dados de treinamento (E_{in}) e validação (E_{val}). Quando $\alpha = 0$ temos que T é construída sem penalidade em sua complexidade, obtendo $E_{in} = 0\%$, entretanto, o E_{val} obtido de dados não utilizados na construção de T ficou em torno de 11% que é muito acima de E_{in} (*overfitting*), não generalizando o aprendizado. Com o aumento do valor de α é possível perceber uma convergência de E_{in} e E_{val}, eliminando o *overfitting* de T. Também é possível perceber que após um certo limiar no crescimento do valor de α, apesar dos valores de E_{in} e E_{val} estarem muito próximos, o valor do erro aumenta devido a excessiva diminuição do grau de liberdade de aprendizado de T, fenômeno conhecido como *underfitting*.

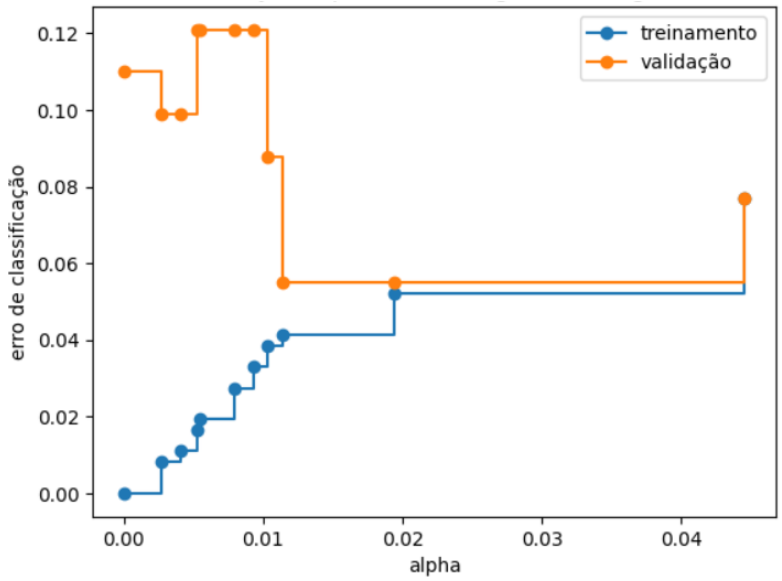


Figura 6: Comparação do erro de classificação x complexidade da árvore de decisão durante processo de regularização.

Para analisar a generalização do aprendizado da árvore de decisão $\bar{h}$ construída pelo modelo OCT_{top} propomos identificar o fenômeno do *overfitting* analisando a diferença entre $E_{sub} = \frac{1}{\bar{n}} \sum_{x \in \bar{X}} 1[\bar{h}(x) \neq \bar{y}]$ que é o erro de classificação de $\bar{h}$ sobre a subamostra $(\bar{X}, \bar{y})$ e $E_{val} = \frac{1}{n_{val}} \sum_{x \in X_{val}} 1[\bar{h}(x) \neq \bar{y}]$ que é o erro de classificação de $\bar{h}$ sobre (X_{val}, y_{val}). Testando $|E_{sub} - E_{val}| \leq \epsilon$ podemos identificar ausência de *overfitting*, entretanto, como iniciamos o procedimento construindo $\bar{h}(x)$ com a topologia de complexidade mínima, aumentando sua complexidade nas próximas iterações, devemos eliminar também o *underfitting*, para isto, a condição de parada será quando $|E_{sub} - E_{val}| > \epsilon$, retornando o h_{best} com menor valor de E_{val} entre todas as iterações.

5.4 Ramificação da topologia da árvore de decisão

Caso a árvore de decisão $\bar{h}$ construída na iteração corrente não tenha gerado *overfitting*, propomos o aumento da complexidade da árvore de decisão a ser inferida em uma próxima iteração do procedimento. Para isto, propomos o crescimento da topologia $\mathcal{T}$ corrente, abrindo seus nós folha em nós de ramificação, cada um com dois nós folha. Para isto definimos para cada nó folha $t \in \mathcal{T}$ a função taxa de erro

$$R(t) = r(t) * p(t),$$

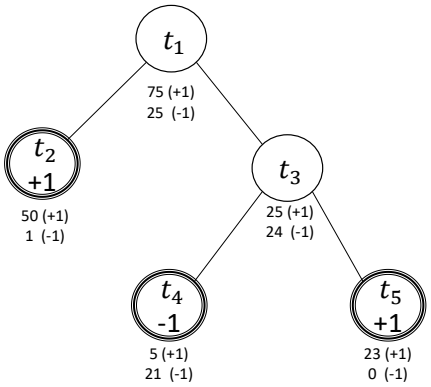
onde:

- n_t : número de amostras atribuídas ao nó t.
- $r(t)$: proporção das amostras de classe minoritária no nó t.
- $p(t)$: proporção das amostras contidas no nó t $(\frac{n_t}{n})$.

A Figura 7 apresenta uma árvore de classificação exemplo e uma tabela contendo os valores $R(t)$ para cada um dos seus nós folha. É possível perceber que a folha t_4 possui 26 pontos atribuídos, sendo 21 classificados com rótulo majoritário -1 e 5 pontos classificados com rótulo minoritário $+1$, logo, $R(t_4) = \frac{5}{100} = 5\%$, já a folha t_5 não possui nenhum ponto mal classificado, logo, $R(t_5) = 0\%$. É possível perceber que a soma $R(t_2)+R(t_4)+R(t_5) = 6\%$ representa o erro de classificação geral da árvore exemplo.

Sejam t_l e t_r os nós esquerdo e direito do nó t. Propomos as seguintes estratégias de ramificação dos nós folhas de uma topologia $\mathcal{T}$:

- Gulosa: considerando $t = \arg\max_{l \in \mathcal{T}} R(l)$, se $R(t) \geq 5\%$ então $\mathcal{T} = \mathcal{T} \cup \{t_l, t_r\}$;
- Exploratório: $\mathcal{T} = \mathcal{T} \cup \{t_l, t_r \ \mid t \in \mathcal{T} \text{ e } R(t) \geq 5\%\}$.



	n_t	$r(t)$	$p(t)$	$R(t)$
t_2	51	$\frac{1}{51}$	$\frac{51}{100}$	$\mathbf{\frac{1}{100}}$
t_4	26	$\frac{5}{26}$	$\frac{26}{100}$	$\mathbf{\frac{5}{100}}$
t_5	23	0	$\frac{23}{100}$	$\mathbf{0}$

Figura 7: Árvore de classificação binária $\{+1, -1\}$ exemplo e sua tabela com valores de taxa de erro para cada nó folha t.

Na estratégia gulosa, a folha $l \in \mathcal{T}$ que tiver o maior valor $R(l)$ e $R(l) \geq 5\%$ se tranforma em um nó ramificação e seus nós filhos adicionados como folhas de $\mathcal{T}$. Já na estratégia exploratória, qualquer nós folha l cujo $R(l) \geq 5\%$ terá seus filhos adicionados como folha em $\mathcal{T}$. Na árvore exemplo da Figura 7 apenas a folha t_4 seria ramificada, criando uma nova topologia a ser analisada na próxima iteração do algoritmo.

Se ao final do processo de ramificação não houver adição de nenhum novo nó na topologia $\mathcal{T}$, então atingimos uma condição de parada retornando a árvore de decisão h_{best} como saída do procedimento. Caso ocorra atualização em $\mathcal{T}$ uma nova iteração do método é inicializada usando a nova topologia construída nesta fase.

6 Resultados computacionais

6.1 nOCT

A fim de avaliarmos a efetividade das formulações e estratégias propostas, nesta seção são apresentados e discutidos os resultados de experimentos computacionais realizados, comparando seus tempos de execução e a capacidade de generalização sobre os dados. Para tanto, os métodos descritos neste trabalho foram implementados na linguagem de programação C++, utilizando o compilador GNU GCC versão 10.5.0 (opção de compilação -O3) e com auxílio do resolvedor matemático CPLEX na versão 20.1, tendo sido executados em um computador equipado com processador AMD Ryzen 5700U com 36 GB de RAM, executando o sistema operacional Ubuntu 20.04 e limitados a 1 *thread* de execução.

As Tabelas 4, 5 e 6 descrevem os resultados dos experimentos realizados em função do tempo de execução em segundos fixando-se a profundidade $D = 1, 2$ e 3, respectivamente, bem como mantendo-se $\alpha = 0$ para todas as execuções. Para o escopo deste trabalho, foram selecionados, para o conjunto de entrada, 31 problemas de classificação também presentes em Bertsimas e Dunn (2017) e disponíveis no repositório de Machine Learning UCI (DUA; GRAFF, 2017). Cada linha das tabelas representa a execução de uma instância em particular, identificada pela coluna *Inst.*, e cujo nome, tamanho, quantidade de características e de classes estão indicados, respectivamente, nas colunas *Nome*, n, p e K da Tabela 3.

As demais colunas caracterizam alguma variante em particular de uma das formulações anteriormente apresentadas. As colunas OCT^{bc} e OCT denotam o modelo proposto em (BERTSIMAS; DUNN, 2017), com e sem uma estratégia explícita de *branch-and-cut*, respectivamente. Por sua vez, as estratégias propostas neste artigo estão presentes nas colunas $nOCT$, $nOCT^{bc}$, $nOCT^{p}$, $nOCT^{bcp}$, $nOCT^{P(d\leq 1)}$, $nOCT^{P(d\leq 8)}$, $nOCT^{e}$, $nOCT^{ebc}$, $nOCT^{ep}$, $nOCT^{ebcp}$, $nOCT^{eP(d\leq 1)}$ e $nOCT^{eP(d\leq 8)}$, nas quais o sobrescrito bc denota a utilização da estratégia de *branch-and-cut* descrita na Seção 3.2; p indica a inserção das inequações de precedência da Seção 3.1; $P(d \leq \mathcal{D})$ descreve a adição dessas mesmas inequações, porém inicialmente relaxadas e inseridas dinamicamente, desde que a profundidade do nó da árvore de *branch-and-cut* seja menor que $\mathcal{D}$; e, por fim, e denota a utilização do operador de igualdade ($=$) na expressão 19, enquanto sua ausência expressa a utilização do operador menor ou igual ($\leq$). Ademais, valores marcados com **TL** indicam que a execução ultrapassou o tempo limite, enquanto **ML** denota as instâncias que excederam a memória disponível do computador e, consequentemente, tiveram suas execuções abruptamente interrompidas.

As Tabelas 8, 9 e 10 apresentam informações adicionais destas mesmas execuções para as profundidades $D = 1, 2$ e 3, respectivamente. Cada coluna da tabela denota

Tabela 3: Descrição das instâncias da base UCI utilizadas.

Inst.	Nome	n	p	K
a	Soybean-small	35	35	4
b	Acute-inflammations-2	90	6	2
c	SPECT-heart	60	22	2
d	Acute-inflammations-1	90	6	2
e	Congressional-voting-records	174	16	2
f	Hepatitis	60	19	2
h	Fertility	75	9	2
i	Echocardiogram	45	6	2
j	Hayes-roth	132	4	3
k	Iris	112	4	3
l	Seeds	157	7	3
m	Thyroid-disease-new-thyroid	161	5	3
n	Wine	133	13	3
o	Haberman-survival	229	3	2
p	Heart-disease-Cleveland	222	13	5
q	Balance-scale	468	4	3
r	Planning-relax	136	12	2
s	Breast-cancer-diagnostic	512	9	2
t	Parkinsons	146	22	2
u	SPECTF-heart	60	44	2
v	Mammographic-masses	622	5	2
w	Blood-transfusion	561	4	2
x	Banknote-authentication	1029	4	2
y	Climate-model-crashes	405	18	2
z	Breast-cancer-prognostic	426	30	2
A	Breast-cancer	145	32	2
B	Ionosphere	263	34	2
C	Thoraric-surgery	352	16	2
D	Qsar-biodegradation	791	41	2
E	Connectionist-bench-sonar	156	60	2
F	Dermatology	268	34	6

uma amostra do grupo de instâncias, enquanto as linhas representam, para cada modelo denotado na primeira coluna, as seguintes informações, simbolizada na segunda coluna: *Cortes*, a quantidade de restrições adicionadas ao resolvedor durante toda a execução do problema, incluindo aquelas acrescentadas dinamicamente pelo algoritmo de *branch-and-cut*; *Nós*, a quantidade final de nós da árvore de *branch-and-bound* explorados pelo resolvedor e; *Preced*, a quantidade de cortes adicionados em função das inequações derivadas do grafo de precedência, quando usado pela formulação. A relaxação linear na raiz da árvore de *branch-and-bound* foi omitida por ter valor aproximadamente zero na grande maioria dos casos.

É possível constatar nas Tabelas 4 e 5 que a estratégia proposta, $nOCT$, completa sua execução consistentemente em tempo inferior ao modelo da literatura, alcançando o ótimo necessitando apenas de 4% e 79% do tempo necessário do OCT, em média, nas respectivas tabelas. Por outro lado, ainda que o modelo $nOCT^e$ tenha apresentado um desempenho médio inferior ao $nOCT$ em $D = 1$, embora superior ao OCT, essa estratégia obteve os melhores tempos computacionais médios dos conjuntos $D = 2$ e $D = 3$, atingindo 70% e 84% do tempo médio do OCT respectivamente. No total, o mesmo conquistou 26 dos melhores tempos de execução de 93 execuções, das quais 14 foram em $D = 2$. Estes ganhos podem ser explicados através das Tabelas 8 e 9, que demonstram que o $nOCT$ possui, em média, 55% de restrições a menos que o OCT, resultando em apenas 63% da quantidade de nós abertos na resolução da árvore de *branch-and-bound*.

Dito isso, a Tabela 6 evidencia a dificuldade da obtenção de resultados em tempo hábil por todos os métodos ao aumentar-se a profundidade da árvore para $D = 3$. Nenhum dos métodos e variantes apresentados neste artigo resolveram 21 dos 31 conjuntos testados nessa parametrização. Dos 10 restantes, as estratégias $nOCT$ e $nOCT^e$ concluíram todas, enquanto o método da literatura terminou apenas 6. No total, considerando apenas as instâncias em que tanto o OCT quanto o $nOCT^e$ terminaram a execução em todos os grupos, o tempo de execução foi, em média, 57% menor na estratégia proposta, representando, no conjunto todo, uma diminuição de 88% do tempo de execução de todas as execuções somadas.

Contraintuitivamente, as variantes das formulações que dispunham de um algoritmo de *branch-and-cut* para adição dinâmica das restrições tiveram resultados pouco satisfatórios, sendo o OCT^{bc} 15% mais lento em $D = 2$ que seu correspondente sem *branch-and-cut*, OCT, enquanto $nOCT^{ebc}$ demandou quase 38% de tempo computacional a mais em comparação com $nOCT^e$, provocando ocasionalmente o término prematuro dos programas por falta de memória. Tal degradação pode ser explicada através da grande quantidade de cortes totais adicionados ao modelo durante a execução desses algoritmos, que no caso do $nOCT^{bc}$ atingiu em média 115% em comparação ao original sem cortes dinâmicos, enquanto que no OCT^{bc} esse valor chega a ser diversas ordens de magnitude

Tabela 4: Tempo de execução das formulações para D=1

Nome	OCT	OCT^{bc}	nOCT	nOCT^{bc}	nOCT^{p}	nOCT^{bcp}	nOCT$^{P(d\leq1)}$	nOCT$^{P(d\leq3)}$	nOCT^{e}	nOCT^{ebc}	nOCT^{ep}	nOCT^{ebcp}	nOCT$^{eP(d\leq1)}$	nOCT$^{eP(d\leq3)}$
a	0,07	0,08	0,03	0,04	0,05	0,03	0,04	0,03	0,04	0,07	0,07	**0,02**	0,04	0,04
b	0,10	0,14	0,06	0,07	0,11	0,13	0,06	0,09	0,04	**0,03**	0,10	0,07	0,04	0,04
c	0,05	0,14	**0,02**	0,10	0,06	0,11	**0,02**	**0,02**	0,03	0,07	0,05	0,07	**0,02**	0,03
d	0,07	0,15	0,06	0,05	0,05	0,21	0,04	**0,03**	0,04	0,08	0,04	0,08	**0,03**	0,04
e	0,38	0,73	0,24	0,23	0,30	**0,12**	0,30	0,26	0,20	0,17	0,24	0,27	0,18	0,25
f	0,16	0,26	0,12	0,11	0,12	0,10	0,11	0,16	0,10	0,09	**0,05**	0,08	0,08	0,08
h	0,15	0,15	0,09	0,11	0,08	0,15	0,07	0,09	**0,06**	0,07	0,08	0,09	**0,06**	0,08
i	0,15	0,12	0,05	0,06	0,11	**0,04**	0,05	0,06	**0,04**	0,06	**0,04**	0,05	**0,04**	**0,04**
j	0,43	0,59	0,16	0,30	**0,11**	0,35	0,17	0,20	0,24	0,36	0,24	0,30	0,25	0,26
k	0,34	0,42	0,20	**0,01**	0,40	0,41	0,21	0,43	0,13	**0,01**	0,14	0,28	0,36	0,06
l	2,71	1,16	0,63	0,69	0,88	1,13	1,20	1,48	0,30	**0,17**	1,28	1,07	0,66	0,51
m	0,56	0,88	0,66	0,57	0,71	0,84	**0,42**	0,50	0,44	0,48	0,54	0,48	0,52	0,52
n	0,66	1,59	0,42	0,61	0,36	0,33	0,40	0,61	**0,18**	0,62	0,37	0,59	**0,18**	0,78
o	1,08	2,16	0,56	0,64	0,55	0,63	0,39	0,53	0,33	0,68	0,40	0,67	**0,27**	0,29
p	5,89	7,12	4,70	4,88	3,65	4,38	3,39	**3,03**	6,45	4,39	5,19	6,12	5,83	18,69
q	1,59	6,62	1,31	2,79	4,48	7,74	4,33	4,48	**0,99**	2,82	4,24	3,35	1,11	3,98
r	110,61	3,55	5,81	1,46	2,37	1,12	8,72	**0,62**	0,97	2,14	1,53	1,55	0,68	1,13
s	4,65	4,88	0,90	1,89	3,89	13,22	1,02	1,30	**0,54**	2,64	2,89	6,62	1,70	1,76
t	1,10	2,45	0,81	0,73	0,64	0,77	0,69	0,69	0,45	1,65	**0,44**	1,92	0,52	0,48
u	3,20	3,20	3,36	0,60	4,07	0,58	3,58	3,48	0,68	1,10	**0,48**	1,12	0,51	0,53
v	5,14	10,61	3,26	3,23	1,07	7,43	1,60	1,58	1,21	3,47	1,34	1,90	**1,03**	5,94
w	3,63	9,91	4,05	4,73	1,75	1,65	0,85	**0,70**	1,67	3,34	0,78	1,03	0,74	0,94
x	18,51	110,32	8,32	18,88	30,13	73,35	32,31	38,00	**4,15**	17,74	9,73	61,78	6,28	7,77
y	15,79	23,62	4,93	8,60	4,13	7,43	5,28	5,96	50,41	3,61	48,90	**3,47**	39,01	46,47
z	19,21	15,25	**3,89**	7,85	6,66	14,99	5,02	4,70	31,09	4,15	4,82	23,14	25,85	12,98
A	140,47	7,74	11,00	3,17	9,95	2,84	1,97	**1,19**	3,25	4,34	4,46	4,08	1,92	4,17
B	12,26	15,63	2,60	5,52	2,39	6,50	**2,21**	2,70	4,59	5,01	7,38	3,50	3,19	3,63
C	3,84	5,62	1,11	1,96	3,58	5,83	**1,09**	1,97	2,30	4,39	3,01	3,04	2,46	2,36
D	TL	3359,46	73,48	181,87	56,14	258,47	272,74	76,17	285,56	209,55	100,86	147,44	**26,56**	51,88
E	TL	96,87	167,72	12,06	167,92	**10,02**	194,44	192,09	52,36	25,17	53,44	23,21	47,67	43,06
F	8,41	19,77	2,73	8,29	1379,93	19,89	2,21	109,70	**1,35**	6,54	395,35	81,03	1,97	6,81
Média	243,91	119,72	9,78	8,78	54,41	14,22	17,58	14,61	14,52	9,84	20,92	12,21	**5,48**	6,95

Tabela 5: Tempo de execução das formulações para D=2

Nome	OCT	OCT^{bc}	nOCT	nOCT^{bc}	nOCT^{p}	nOCT^{bcp}	nOCT$^{P(d\leq1)}$	nOCT$^{P(d\leq3)}$	nOCT^{e}	nOCT^{ebc}	nOCT^{ep}	nOCT^{ebcp}	nOCT$^{eP(d\leq1)}$	nOCT$^{eP(d\leq3)}$
a	0,03	7,66	**0,02**	2,65	**0,02**	1,49	**0,02**	**0,02**	**0,02**	1,34	**0,02**	0,76	**0,02**	**0,02**
b	0,03	1,34	**0,01**	0,11	0,02	0,74	**0,01**	**0,01**	**0,01**	0,61	0,03	0,29	**0,01**	**0,01**
c	22,55	96,32	1,62	0,96	2,72	4,03	1,47	1,36	0,89	1,28	1,32	2,39	**0,75**	0,92
d	0,03	13,56	**0,01**	1,87	0,02	0,58	0,02	0,02	0,02	1,97	0,02	1,27	**0,01**	**0,01**
e	43,02	420,86	5,50	5,84	9,61	7,85	5,67	7,65	**5,18**	5,45	7,25	19,17	7,85	6,96
f	185,01	324,12	**23,55**	75,09	41,60	37,08	23,80	51,62	49,22	79,86	36,01	46,74	49,92	62,73
h	43,61	112,68	17,85	33,59	24,50	34,11	24,29	21,70	**15,83**	19,33	19,01	41,56	15,94	31,00
i	29,64	96,72	7,27	9,32	6,31	17,03	7,31	7,19	6,17	11,71	5,82	9,40	6,63	**3,86**
j	36,23	366,45	11,37	31,05	22,07	41,44	24,63	36,47	**10,96**	37,83	30,33	65,78	25,34	43,13
k	14,15	122,94	11,15	8,25	4,10	13,47	12,58	28,33	**2,79**	4,84	3,67	4,19	5,55	18,37
l	386,69	TL	1001,65	284,07	49,86	681,75	496,21	218,02	**30,04**	111,91	38,87	236,15	72,48	85,73
m	461,90	599,19	84,64	126,23	29,02	381,32	52,62	63,45	**22,96**	132,40	24,87	381,13	36,76	68,89
n	101,25	1041,86	50,12	216,56	**21,58**	313,91	168,51	68,58	85,32	232,98	33,58	86,40	88,60	45,63
o	321,48	2871,06	122,43	403,08	161,26	274,72	108,85	149,42	**64,12**	403,00	123,85	191,74	83,18	124,89
p	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
q	TL	TL	164,07	1341,43	440,92	2344,16	701,78	880,43	**77,80**	1768,67	432,51	1202,95	241,84	673,12
r	TL	TL	TL	TL	TL	TL	TL	TL	**2081,99**	TL	2534,69	TL	2366,19	2568,67
s	2748,26	TL	423,54	1007,37	613,04	2515,69	2085,72	1012,64	**230,59**	2054,85	904,08	1936,60	976,32	1602,37
t	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
u	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
v	TL	TL	221,07	TL	177,71	TL	726,57	878,94	358,42	TL	**173,37**	424,80	669,49	770,97
w	2989,20	TL	923,74	TL	**227,19**	TL	592,09	639,01	732,87	TL	523,36	669,83	699,58	602,90
x	TL	TL	TL	TL	TL	ML	TL	TL	**1631,06**	TL	TL	ML	TL	TL
y	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
z	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
A	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
B	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
C	TL	TL	1916,81	TL	2543,89	TL	2568,32	3391,42	**1365,11**	TL	2933,08	TL	1944,30	2954,81
D	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
E	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
F	TL	TL	TL	TL	TL	TL	TL	TL	TL	ML	TL	TL	TL	TL
Média	1980,10	2286,28	1554,40	1856,37	1534,69	1957,08	1638,72	1634,07	**1379,72**	1898,97	1529,86	1681,33	1512,61	1589,19

Tabela 6: Tempo de execução das formulações para D=3

Nome	OCT	OCT^{bc}	nOCT	nOCT^{bc}	nOCT^{p}	nOCT^{bcp}	nOCT$^{P(d\leq1)}$	nOCT$^{P(d\leq3)}$	nOCT^{e}	nOCT^{ebc}	nOCT^{ep}	nOCT^{ebcp}	nOCT$^{eP(d\leq1)}$	nOCT$^{eP(d\leq3)}$
a	**0,0**	10,7	**0,0**	0,9	**0,0**	80,0	**0,0**	**0,0**	**0,0**	29,1	**0,0**	32,5	**0,0**	**0,0**
b	0,1	19,5	**0,0**	86,7	0,1	3,1	**0,0**	**0,0**	**0,0**	657,3	0,1	5,5	**0,0**	**0,0**
c	TL	TL	5,5	54,8	16,1	240,0	**4,9**	5,4	5,7	66,5	23,1	122,7	8,0	8,8
d	0,1	6,9	**0,0**	9,1	0,1	15,3	**0,0**	**0,0**	**0,0**	4,8	0,1	8,7	**0,0**	**0,0**
e	TL	TL	1105,1	2401,8	2403,8	TL	1891,0	TL	**789,1**	1775,1	TL	TL	869,3	TL
f	2748,7	TL	**51,8**	949,4	362,2	534,6	119,1	85,0	96,5	TL	690,5	1798,6	204,2	288,8
h	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
i	TL	TL	416,8	1079,3	477,9	2882,9	241,7	285,5	367,5	1489,9	550,6	1129,3	234,8	**223,9**
j	TL	TL	TL	TL	TL	ML	TL	TL	TL	TL	TL	ML	TL	TL
k	54,4	TL	88,7	TL	**8,7**	1288,4	104,7	38,4	25,1	1593,8	15,6	140,0	82,1	39,6
l	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
m	TL	TL	TL	TL	TL	ML	TL	TL	TL	TL	TL	TL	TL	TL
n	3246,1	TL	157,1	TL	1768,0	TL	332,8	3247,4	**105,0**	TL	199,3	TL	374,9	107,7
o	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
p	TL	TL	TL	TL	TL	ML	TL	TL	TL	TL	TL	TL	TL	TL
q	TL	TL	TL	TL	TL	ML	TL	TL	TL	TL	TL	TL	TL	TL
r	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
s	TL	TL	TL	ML	TL	ML	TL	TL	TL	TL	TL	ML	TL	TL
t	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
u	TL	TL	3257,4	TL	**3243,4**	TL	3592,6	3278,1	3568,3	TL	TL	TL	TL	TL
v	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
w	TL	TL	TL	ML	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
x	TL	TL	TL	ML	TL	ML	TL	TL	TL	TL	TL	ML	TL	TL
y	TL	ML	TL	ML	TL	ML	TL	TL	TL	TL	TL	TL	TL	TL
z	TL	TL	TL	TL	TL	ML	TL	TL	TL	ML	TL	ML	TL	TL
A	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
B	TL	TL	TL	TL	TL	TL	TL	TL	TL	ML	TL	ML	TL	TL
C	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	ML	TL	TL
D	TL	TL	TL	ML	TL	ML	TL	TL	TL	ML	TL	TL	TL	TL
E	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
F	TL	TL	TL	TL	TL	TL	TL	TL	TL	ML	TL	ML	TL	TL
Média	3098,4	3252,8	2602,7	2934,9	2705,8	2949,8	2641,5	2778,7	**2598,6**	2968,3	2718,7	2891,5	2612,0	2692,5

Tabela 7: Tempo de execução médio das formulações para as diferentes profundidades experimentadas

D	OCT	OCT^{bc}	nOCT	nOCT^{bc}	nOCT^{p}	nOCT^{bcp}	nOCT$^{P(d\leq1)}$	nOCT$^{P(d\leq3)}$	nOCT^{e}	nOCT^{ebc}	nOCT^{ep}	nOCT^{ebcp}	nOCT$^{eP(d\leq1)}$	nOCT$^{eP(d\leq3)}$
1	243,91	119,72	9,78	8,78	54,41	14,22	17,58	14,61	14,52	9,84	20,92	12,21	**5,48**	6,95
2	1980,10	2286,28	1554,40	1856,37	1534,69	1957,08	1638,72	1634,07	**1379,72**	1898,97	1529,86	1681,33	1512,61	1589,19
3	3098,4	3252,8	2602,7	2934,9	2705,8	2949,8	2641,5	2778,7	**2598,6**	2968,3	2718,7	2891,5	2612,0	2692,5

superior ao número de restrições do OCT.

Por sua vez, as estratégias que empregam o grafo de precedência $nOCT^{p}$ e $nOCT^{ep}$ obtiveram resultados mistos, com médias ligeiramente inferiores a suas contrapartes sem essas inequações, obtendo os melhores tempos entre os algoritmos em 12 instâncias. Mais precisamente, a formulação $nOCT^{p}$ foi, em média, 3.1% mais lenta que o $nOCT$ enquanto que $nOCT^{ep}$ demandou 22% de tempo a mais que $nOCT^{e}$, algo que pode ser atribuído ao grande número de inequações adicionados ao modelo em decorrência do grafo de precedência, que representou 85% de todos os cortes adicionados. Entretanto, é importante observar a redução de 58% do número de nós abertos pela estratégia $nOCT^{p}$, o que sugere que a construção de uma estratégia dinâmica de adição das inequações de precedência possa melhorar a sua eficiência, como é o caso das formulações $nOCT^{eP(d\leq 1)}$ e $nOCT^{eP(d\leq 8)}$, que obtiveram os melhores tempos computacionais médios do grupo $D = 1$, equivalente a 56% do tempo do $nOCT$ e 2,2% do OCT, alcançando conjuntamente um total de 8 melhores tempos nessa segmentação.

Por fim, as últimas estratégias apresentadas, $nOCT^{bcp}$ e $nOCT^{ebcp}$, que utilizam tanto o grafo de precedência quanto o algoritmo de *branch-and-cut*, obtiveram conjuntamente apenas cinco vezes os melhores tempos entre os algoritmos, quase todas instâncias pequenas, e das quais duas ocorreram empatadas com alguma outra formulação.

6.2 Comparativo das estratégias do USP

A Tabela 11 relaciona o custo da função objetivo e o tempo de execução, em segundos, das estratégias para obtenção de soluções do USP. Cada linha representa uma instância do conjunto de entrada, cada qual extraída do repositório público de *datasets* UCI Machine Learning (DUA; GRAFF, 2017). As colunas n e p indicam, respectivamente, a quantidade de observações e de características de cada instância do conjunto de entrada. As colunas contidas em *Upperbound* descrevem o custo da função objetivo da melhor solução inteira encontrada por cada algoritmo (quanto menor, melhor). Por sua vez, as colunas em *Tempo de execução* denotam o tempo cronológico, em segundos, decorrido desde o início até o término da execução dos algoritmos.

As colunas *MIP* correspondem a modelagem de Programação Linear Inteira abordado na Seção 4, enquanto *BRGKA*, *ILS* e *Híbrido* compreendem as estratégias apresentadas nas Seções 4.2 - 4.4. Considerando o caráter não determinístico dos últimos três, os números apresentados representam a média de dez execuções independentes. Além disso, observando-se que o propósito desses métodos é, principalmente, a prestação de suporte a outros métodos, provendo-os com uma entrada de menor tamanho porém representativa, e levando em conta o custo computacional, principalmente dos modelos de programação linear inteira, foi fixado um tempo limite de 30 minutos para cada execução. Os casos em

Tabela 8: Informações estatísticas adicionais para D=1

Inst.	OCT		OCT^{bc}		nOCT		$nOCT^{bc}$		$nOCT^{p}$		$nOCT^{bcp}$		$nOCT^{P(d\leq1)}$		$nOCT^{P(d\leq3)}$		$nOCT^{e}$		$nOCT^{ebc}$		$nOCT^{ep}$		$nOCT^{ebcp}$		$nOCT^{eP(d\leq1)}$		$nOCT^{eP(d\leq3)}$	
	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós
a	209	276	1156	97	**126**	**19**	157	118	176	24	127	**19**	174	20	379	21	**126**	63	157	178	176	47	127	**19**	173	53	424	54
b	472	54	3857	181	283	38	**281**	325	535	12	531	28	2408	39	867	**10**	283	28	298	72	535	24	527	55	460	25	1262	25
c	322	114	8307	450	**193**	**6**	232	114	441	7	459	68	319	**6**	325	**6**	**193**	7	233	167	441	20	455	84	312	7	418	7
d	472	92	5091	371	**283**	63	325	220	535	**9**	529	59	681	**9**	1329	**9**	**283**	53	312	154	535	10	552	41	693	23	694	13
e	892	9	1.0e4	156	**535**	**7**	628	80	1051	10	975	11	1768	**7**	1768	**7**	**535**	17	587	133	1051	**7**	1083	138	733	23	2537	17
f	322	1029	1.1e4	703	**193**	828	225	693	209	877	247	534	207	654	235	709	**193**	358	234	480	209	**110**	256	424	199	355	200	371
g	397	551	6350	354	**238**	236	305	500	428	100	449	333	486	**95**	4218	113	**238**	152	312	379	428	199	461	231	466	121	2535	117
h	247	1597	4007	611	**148**	157	167	612	254	609	266	186	347	208	837	204	**148**	163	170	534	254	192	273	215	282	**74**	752	77
i	688	352	3.0e4	1451	**413**	349	508	1409	881	**72**	870	216	1742	120	7654	105	**413**	203	505	1252	881	89	876	195	1268	111	8103	114
j	588	1161	2.2e4	801	353	712	**166**	**11**	1271	112	1301	180	2971	102	9263	115	353	56	**166**	**11**	1271	43	1298	126	1422	51	2982	**11**
k	813	5917	3.6e4	1741	**488**	1410	641	1853	2298	271	2379	479	5628	336	2.3e4	439	**488**	264	618	485	2298	511	2359	559	3563	**86**	1.3e4	152
l	833	398	4.1e4	889	**500**	502	661	2238	1694	**189**	1794	600	1846	425	7734	287	**500**	695	627	1502	1694	201	1784	405	1346	224	1.1e4	194
m	693	1282	1.2e5	3026	**416**	754	505	1396	478	528	534	816	420	681	640	879	**416**	**193**	513	1871	478	512	560	1577	**416**	**193**	638	1674
n	1167	334	1.0e5	2851	**700**	649	831	3125	2184	40	2242	148	6113	33	1.4e4	32	**700**	87	848	2163	2184	**30**	2264	217	6025	41	1.2e4	41
o	1150	6236	2.9e5	1.1e4	**691**	5267	903	1.0e4	1263	2468	1425	7498	1614	2714	1.7e4	**2407**	**691**	6276	904	1.1e4	1263	4120	1455	1.0e4	**691**	6276	6040	1.8e4
p	2368	450	2.2e5	3942	**1421**	255	1609	4383	4359	62	4442	234	7851	114	6.3e4	88	**1421**	231	1591	3444	4359	**58**	4390	136	1438	247	3.7e4	60
q	702	4.3e5	1.7e5	9313	**421**	2.5e4	555	8854	433	7549	568	7436	**421**	3.4e4	678	**1657**	**421**	2613	552	1.6e4	433	4704	552	1.2e4	437	1992	665	3120
r	2582	2051	2.3e5	1988	**1549**	49	1954	1456	8123	**26**	8164	1370	8970	40	3.6e4	38	**1549**	151	1984	2189	8123	45	8127	614	6694	127	3.6e4	77
s	752	3206	1.4e5	5966	**451**	**1030**	522	4387	**451**	**1030**	522	4387	**451**	**1030**	**451**	**1030**	**451**	1175	543	7861	**451**	1175	543	7861	**451**	1175	**451**	1175
t	322	1.3e4	1.3e5	1.0e4	**193**	2.1e4	223	4466	**193**	2.1e4	223	4466	**193**	2.2e4	**193**	2.2e4	**193**	**3157**	223	9935	**193**	**3157**	223	9935	**193**	**3157**	**193**	**3157**
u	3132	2851	9.3e5	5282	**1879**	953	2432	2432	4493	**13**	4697	336	2.6e4	28	4.0e4	31	**1879**	154	2499	5998	4493	**13**	4537	131	4489	33	3.7e4	113
v	2827	1449	9.5e5	8506	**1696**	2436	2134	6226	4676	39	4685	95	1.2e4	20	1.4e4	17	**1696**	1248	2256	5453	4676	**14**	5101	78	4656	36	1.3e4	30
w	5167	4741	8.9e6	3.5e4	**3100**	1358	3790	2.3e4	2.5e4	410	2.6e4	9051	4.5e4	2535	1.2e5	2291	**3100**	615	3759	1.4e4	2.5e4	50	2.5e4	4214	2.5e4	35	7.3e4	**31**
x	2047	1.2e4	1.3e6	1.5e4	**1228**	**2437**	1602	1.2e4	**1228**	**2437**	1602	1.2e4	**1228**	**2437**	**1228**	**2437**	**1228**	5.5e4	1442	8459	**1228**	5.5e4	1442	8459	**1228**	5.1e4	**1228**	5.1e4
y	2152	1.7e4	7.5e5	1.1e4	**1291**	1586	1546	1.5e4	6911	929	7011	8104	7321	2070	8535	1516	**1291**	3.8e4	1610	9774	6911	**330**	7162	9706	1463	2.1e4	9647	5942
z	747	4.8e5	4.2e5	2.0e4	**448**	4.1e4	590	2.0e4	564	3.4e4	666	1.8e4	588	3496	1823	**2370**	**448**	1.2e4	585	2.4e4	564	1.2e4	698	2.5e4	590	5786	1078	1.4e4
A	1337	1.6e4	8.3e5	1.7e4	**802**	841	970	1.4e4	1296	1003	1502	1.4e4	1073	**696**	3460	1310	**802**	4621	1060	1.2e4	1296	5850	1736	6564	1320	2503	4718	1853
B	1782	578	2.6e5	3107	**1069**	416	1421	3186	4865	145	5015	1676	**1069**	417	8.9e4	188	**1069**	1378	1258	6523	4865	**116**	4943	1074	1.4e4	546	6.3e4	203
C	3977	2.1e6	2.8e8	1.0e6	**2386**	3.0e4	3012	1.7e5	2446	2.5e4	3172	1.8e5	2492	1.1e5	3473	2.4e4	**2386**	8.9e4	3115	2.0e5	2446	3.6e4	3170	1.2e5	2523	**6055**	3817	1.7e4
D	802	7.6e6	4.0e6	1.3e5	**481**	4.7e5	637	**5.0e4**	**481**	4.7e5	637	**5.0e4**	**481**	5.3e5	**481**	5.3e5	**481**	1.3e5	637	9.8e4	**481**	1.3e5	637	9.8e4	**481**	1.1e5	**481**	1.1e5
E	1386	2047	7.4e5	1.1e4	**833**	1361	1086	7156	1043	1.4e6	1258	1.3e4	1035	377	5452	7.1e4	**833**	**353**	1085	5896	1043	2.7e5	1249	7.3e4	1204	694	4004	2181

Tabela 9: Informações estatísticas adicionais para D=2

Inst.	OCT		OCT^{bc}		nOCT		nOCT^{bc}		nOCT^{p}		nOCT^{bcp}		nOCT$^{P(d\leq1)}$		nOCT$^{P(d\leq3)}$		nOCT^{e}		nOCT^{ebc}		nOCT^{ep}		nOCT^{ebcp}		nOCT$^{eP(d\leq1)}$		nOCT$^{eP(d\leq3)}$	
	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós
a	527	**0**	3.6e5	2.0e4	288	**0**	**285**	1.3e4	488	**0**	503	6328	288	**0**	288	**0**	288	**0**	308	6681	488	**0**	493	4252	288	**0**	288	**0**
b	1218	**0**	6.7e4	2429	657	**0**	**515**	249	1665	**0**	1534	569	657	**0**	657	**0**	657	**0**	694	2802	1665	**0**	1473	122	657	**0**	657	**0**
c	828	5.6e4	4.8e6	2.1e5	**447**	1797	509	2786	1439	1177	1428	2667	1270	1057	8532	548	**447**	868	516	2943	1439	**490**	1436	1646	614	530	6350	498
d	1218	**0**	9.3e5	2.3e4	**657**	**0**	702	9166	1665	**0**	1578	215	**657**	**0**	**657**	**0**	**657**	**0**	756	9081	1665	**0**	1592	798	**657**	**0**	**657**	**0**
e	2310	1.5e4	1.5e7	3.9e5	**1245**	345	1272	5141	3309	549	3151	1091	4769	325	2.2e4	210	**1245**	501	1311	3982	3309	**173**	3203	1638	2759	834	2.3e4	295
f	828	4.5e5	1.5e7	6.6e5	**447**	8.7e4	450	3.8e5	511	1.3e5	545	1.8e5	511	**6.5e4**	1128	1.7e5	**447**	1.7e5	552	4.3e5	511	1.4e5	534	2.6e5	496	1.7e5	1086	2.1e5
g	1023	7.7e4	4.3e6	2.5e5	**552**	6.2e4	591	1.6e5	1312	4.7e4	1368	9.1e4	1581	5.4e4	2.3e4	3.6e4	**552**	4.6e4	578	9.2e4	1312	**3.1e4**	1362	1.1e5	736	4.8e4	1.3e4	6.6e4
h	633	9.1e4	5.0e6	3.6e5	**342**	3.9e4	366	8.2e4	766	1.6e4	789	8.5e4	519	3.6e4	4087	2.2e4	**342**	3.4e4	374	9.8e4	766	2.2e4	796	5.2e4	420	3.1e4	4267	**1.4e4**
i	1776	2.6e4	1.5e7	4.9e5	**959**	1.5e4	1078	7.0e4	2831	6770	2838	1.8e4	6120	7209	8.0e4	7046	**959**	1.5e4	1048	9.9e4	2831	**6115**	2684	2.7e4	2078	1.5e4	8.0e4	8761
j	1516	1.3e4	3.6e6	1.6e5	819	1.7e4	809	2.7e4	4491	775	4404	3885	6461	1556	6.8e4	2345	819	5323	**774**	1.7e4	4491	**714**	4325	2885	7547	3357	4.1e4	2067
k	2101	3.5e5	8.9e7	4.2e6	**1134**	9.8e5	1302	5.0e5	8374	7909	8327	1.4e5	1.1e4	1.3e5	1.1e5	1.6e4	**1134**	4.8e4	1262	2.5e5	8374	**6765**	8267	5.6e4	9766	4.0e4	1.2e5	1.1e4
l	2153	2.3e5	1.3e7	4.0e5	**1162**	9.0e4	1253	2.1e5	5938	**4239**	6061	8.7e4	6533	2.3e4	5.3e4	1.6e4	**1162**	2.8e4	1385	2.6e5	5938	4499	6029	8.4e4	8001	1.5e4	5.6e4	1.5e4
m	1789	7.1e4	2.0e7	7.0e5	**966**	3.1e4	1159	3.1e5	1214	**8782**	1309	3.6e5	1227	8.5e4	3113	3.5e4	**966**	5.6e4	1070	5.5e5	1214	2.0e4	1249	2.1e5	**966**	6.0e4	2127	2.4e4
n	3025	3.8e5	1.1e8	2.7e6	**1630**	1.7e5	1940	1.0e6	7566	2.9e4	7551	7.6e4	1.9e4	3.5e4	2.4e5	3.5e4	**1630**	8.4e4	1888	1.1e6	7566	2.4e4	7544	7.0e4	1.7e4	4.9e4	1.7e5	**1.5e4**
o	2970	1.1e6	1.4e8	2.7e6	**1605**	1.6e6	1967	3.2e6	3893	**4.1e5**	4112	7.7e5	5346	6.7e5	6.9e4	4.7e5	**1605**	2.6e6	1947	3.9e6	3893	4.1e5	4243	8.3e5	1612	2.4e6	5.0e4	1.8e6
p	6144	3.8e6	1.3e8	1.6e6	**3311**	4.1e4	5259	1.3e6	1.5e4	5625	1.5e4	5.4e4	3.7e4	1.8e4	7.6e5	**5142**	**3311**	3.7e4	4027	1.7e6	1.5e4	6614	1.5e4	2.4e4	1.7e4	2.3e4	8.5e5	1.1e4
q	1816	3.3e6	1.1e8	**3.0e6**	**979**	8.1e6	1200	1.6e7	1027	7.6e6	1276	1.6e7	**979**	5.9e6	1998	6.6e6	**979**	5.7e6	1225	1.6e7	1027	6.4e6	1275	1.6e7	**979**	6.4e6	1493	7.4e6
r	6704	1.6e6	1.8e8	1.0e6	**3611**	2.1e5	3912	1.1e6	3.0e4	**2.8e4**	2.9e4	1.9e5	7.2e4	1.9e5	4.7e5	8.3e4	**3611**	1.4e5	3845	2.2e6	3.0e4	3.6e4	2.9e4	1.8e5	3.8e4	9.7e4	7.4e5	7.0e4
s	1946	**2.4e6**	1.9e8	2.8e6	**1049**	2.7e6	1286	1.1e7	**1049**	2.7e6	1286	1.1e7	**1049**	3.1e6	**1049**	3.2e6	**1049**	7.6e6	1252	1.2e7	**1049**	8.1e6	1256	1.3e7	**1049**	7.7e6	**1049**	7.7e6
t	828	1.0e7	2.3e8	**9.7e6**	**447**	1.4e7	503	1.8e7	**447**	1.4e7	503	1.8e7	**447**	1.4e7	**447**	1.4e7	**447**	1.4e7	500	1.9e7	**447**	1.3e7	500	1.9e7	**447**	1.3e7	**447**	1.3e7
u	8134	1.0e6	2.1e8	9.6e5	**4381**	8.3e4	5421	4.3e6	1.5e4	8368	1.5e4	3.6e5	5.4e4	2.7e4	3.6e5	1.7e4	**4381**	1.5e5	5494	3.5e6	1.5e4	**7868**	1.5e4	3.4e4	4.4e4	1.4e4	3.6e5	1.9e4
v	7341	1.5e6	1.7e8	1.4e6	**3954**	7.4e5	4747	4.6e6	1.6e4	**1.7e4**	1.6e4	3.6e5	5.1e4	4.6e4	3.9e5	3.8e4	**3954**	6.0e5	4792	4.5e6	1.6e4	3.7e4	1.6e4	1.1e5	5.6e4	4.4e4	4.1e5	4.6e4
w	1.3e4	4.6e5	2.3e8	6.2e5	**7230**	6.0e5	8300	2.5e6	9.6e4	**1468**	*	*	1.0e5	7.1e4	1.9e6	4281	**7230**	3.3e5	8524	2.7e6	9.6e4	6833	*	*	2.9e5	1.2e4	2.1e6	7048
x	5313	7.6e5	1.1e8	1.3e6	**2862**	9.6e5	3124	4.9e6	**2862**	9.7e5	3122	4.8e6	**2862**	1.0e6	**2862**	1.0e6	**2862**	7.6e5	3235	6.8e6	**2862**	7.7e5	3229	6.7e6	**2862**	6.5e5	**2862**	**6.4e5**
y	5586	6.1e5	1.1e8	7.7e5	**3009**	7.9e5	3488	5.2e6	2.5e4	2.2e5	2.6e4	3.2e5	1.7e4	2.5e5	2.3e5	**9.9e4**	**3009**	1.1e6	3676	4.7e6	2.5e4	1.9e5	2.6e4	3.8e5	3.4e4	7.9e5	1.2e5	4.6e5
z	1933	3.0e6	2.0e8	3.9e6	**1042**	5.2e6	1271	1.7e7	1506	**2.8e6**	1721	1.4e7	1511	3.3e6	4782	4.5e6	**1042**	6.3e6	1332	1.6e7	1506	6.2e6	1746	1.4e7	1448	6.3e6	6478	6.0e6
A	3467	9.6e5	1.0e8	1.0e6	**1868**	9.0e5	2337	5.8e6	3844	8.8e5	4238	1.9e6	4007	6.6e5	1.8e4	**5.1e5**	**1868**	1.2e6	2258	4.2e6	3844	8.3e5	4213	5.4e6	4850	2.0e6	2.4e4	1.3e6
B	4624	8.2e5	2.1e8	1.6e6	**2491**	1.3e6	2889	6.9e6	1.8e4	5.0e5	1.8e4	4.9e5	4.5e4	5.6e5	4.8e5	5.5e5	**2491**	1.6e6	3036	7.5e6	1.8e4	**3.5e5**	1.8e4	6.7e5	1.3e4	6.8e5	5.1e5	3.7e5
C	1.0e4	**1.4e5**	8.0e7	3.7e5	**5564**	2.7e5	6861	2.1e6	5804	1.9e5	7012	1.1e6	5675	3.0e5	6398	3.2e5	**5564**	9.9e5	6643	9.5e5	5804	2.3e5	7024	1.3e6	**5564**	6.2e5	6532	7.4e5
D	2076	**1.6e6**	1.4e8	1.8e6	**1119**	1.8e6	1377	9.9e6	**1119**	1.8e6	1377	1.0e7	**1119**	2.6e6	**1119**	2.5e6	**1119**	3.1e6	1360	1.1e7	**1119**	3.1e6	1360	1.1e7	**1119**	3.2e6	**1119**	3.2e6
E	3580	1.4e6	6.8e7	5.0e5	**1935**	2.1e6	2337	1.6e6	2775	**1.2e5**	2999	6.5e5	3183	2.8e6	8402	6.3e5	**1935**	6.2e5	*	*	2775	9.4e5	3067	1.9e6	2423	6.3e5	6001	1.4e6

Tabela 10: Informações estatísticas adicionais para D=3

Inst.	OCT		OCT^{bc}		nOCT		$nOCT^{bc}$		$nOCT^{p}$		$nOCT^{bcp}$		$nOCT^{P(d\leq1)}$		$nOCT^{P(d\leq3)}$		$nOCT^{e}$		$nOCT^{ebc}$		$nOCT^{ep}$		$nOCT^{ebcp}$		$nOCT^{eP(d\leq1)}$		$nOCT^{eP(d\leq3)}$	
	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós	Cortes	Nós
a	1303	**0**	5.3e5	2.4e4	612	**0**	**552**	5884	1712	**0**	1734	2.2e5	612	**0**	612	**0**	612	**0**	627	1.6e5	1712	**0**	1724	1.2e5	612	**0**	612	**0**
b	3070	**0**	1.1e6	3.9e4	1405	**0**	1407	4.5e5	6949	**0**	6243	905	1405	**0**	1405	**0**	1405	**0**	**1238**	5.1e6	6949	**0**	6438	2590	1405	**0**	1405	**0**
c	2080	2.6e6	1.3e8	2.8e6	**955**	2000	1100	4.7e4	6411	2141	6250	5.4e4	1433	**1664**	9607	2194	**955**	2160	969	4.4e4	6411	2530	6330	2.8e4	1285	1910	1.5e4	2190
d	3070	**0**	4.5e5	8763	1405	**0**	1323	5.1e4	6949	**0**	6625	7686	1405	**0**	1405	**0**	1405	**0**	**1156**	2.8e4	6949	**0**	6411	4516	1405	**0**	1405	**0**
e	5842	1.9e5	5.1e7	4.4e5	**2665**	3.3e4	2759	3.0e5	1.4e4	1.6e4	1.4e4	2.5e4	5141	4.1e4	7.8e4	3.2e4	**2665**	2.7e4	3051	2.5e5	1.4e4	**1.4e4**	1.4e4	7.3e4	5525	2.2e4	7.0e4	5.1e4
f	2080	2.4e6	1.0e8	3.9e6	**955**	**1.1e5**	1037	4.1e6	1307	7.1e5	1294	2.0e6	1065	2.6e5	1372	2.0e5	**955**	2.5e5	971	1.0e7	1307	1.5e6	1337	7.3e6	**955**	6.4e5	1240	6.9e5
g	2575	3.2e6	1.1e8	3.5e6	1180	7.6e6	**1160**	1.1e7	5360	**1.5e6**	5371	3.5e6	2579	5.2e6	2.8e4	2.7e6	1180	6.0e6	1181	1.0e7	5360	1.9e6	5344	2.9e6	2477	4.9e6	3.0e4	2.7e6
h	1585	6.1e6	1.1e8	7.9e6	**730**	1.4e6	763	6.4e6	3062	6.1e5	3037	5.7e6	1103	6.7e5	8790	5.1e5	**730**	1.1e6	763	8.3e6	3062	7.3e5	2949	2.5e6	1137	7.4e5	8465	**4.4e5**
i	4480	1.3e6	9.1e7	1.8e6	**2051**	3.4e6	2122	7.8e6	1.2e4	**2.8e5**	*	*	1.4e4	1.1e6	1.4e5	8.4e5	**2051**	3.3e6	2170	5.1e6	1.2e4	4.8e5	1.2e4	3.4e5	7840	1.7e6	1.4e5	3.6e5
j	3820	4.7e4	7.6e7	3.4e6	1751	6.4e4	1756	8.8e6	2.2e4	**67**	2.1e4	1.3e5	1.4e4	4.4e4	7.5e4	4257	1751	4.6e4	**1682**	4.5e6	2.2e4	766	2.1e4	3.7e4	1.9e4	6.2e4	7.4e4	9589
k	5305	1.9e6	8.7e7	2.8e6	2426	2.9e6	2578	9.3e6	4.2e4	2.1e5	4.2e4	3.4e5	2.6e4	1.1e6	8.2e4	5.1e5	2426	3.8e6	**2406**	8.5e6	4.2e4	**1.5e5**	4.1e4	2.8e5	1.7e4	6.2e5	1.8e5	2.9e5
l	5437	1.5e6	5.6e7	2.4e6	2486	4.0e6	**2465**	7.9e6	2.9e4	**2.3e5**	*	*	1.2e4	1.1e6	5.2e4	3.9e5	2486	2.5e6	2525	7.2e6	2.9e4	4.0e5	2.8e4	8.3e5	1.3e4	1.2e6	1.6e5	4.0e5
m	4513	4.7e6	8.4e7	2.6e6	2066	2.1e5	2103	7.6e6	3430	9.6e5	3262	5.2e6	2152	2.0e5	7138	3.3e6	2066	**6.2e4**	**1935**	7.2e6	3430	2.7e5	3259	7.6e6	2066	3.9e5	4803	6.5e4
n	7657	1.4e6	1.7e8	2.1e6	3490	1.9e6	**3241**	7.5e6	3.6e4	**2.7e4**	3.5e4	1.4e5	1.9e4	4.6e5	3.3e5	1.1e5	3490	2.5e6	3270	6.5e6	3.6e4	3.9e4	3.5e4	9.8e4	2.5e4	8.5e5	3.4e5	1.0e5
o	7498	6.4e5	1.1e8	1.2e6	3433	8.8e5	**3142**	3.2e6	1.6e4	7.5e5	*	*	8005	2.7e6	6.1e4	**5.9e5**	3433	7.6e5	3288	3.9e6	1.6e4	9.0e5	1.6e4	1.3e6	1.3e4	1.5e6	9.6e4	7.5e5
p	1.6e4	4.5e5	1.2e8	1.1e6	7091	9.3e5	**6294**	3.2e6	7.2e4	**4424**	*	*	1.1e5	4.1e4	5.6e5	6.7e4	7091	5.2e5	8695	4.4e6	7.2e4	6468	6.9e4	2.1e4	8.7e4	1.6e5	7.6e5	1.4e4
q	4588	**1.7e6**	9.3e7	2.0e6	2095	2.9e6	2118	8.8e6	2359	1.8e6	2418	7.8e6	2113	3.2e6	2784	3.2e6	2095	3.9e6	**1966**	9.7e6	2359	3.6e6	2346	9.4e6	2095	3.9e6	2517	3.3e6
r	1.7e4	2.8e5	1.8e8	6.7e5	7735	5.3e5	*	*	1.5e5	**8232**	*	*	2.5e5	9.4e4	1.2e6	4.9e4	7735	6.5e5	**6075**	2.9e6	1.5e5	9895	*	*	7.6e4	1.3e5	1.2e6	5.3e4
s	4918	**1.1e6**	1.4e8	2.6e6	2245	3.4e6	2346	8.7e6	2245	3.3e6	2346	8.7e6	2245	2.3e6	2245	2.3e6	2245	2.5e6	**2121**	8.2e6	2245	2.3e6	**2121**	8.3e6	2245	2.2e6	2245	2.2e6
t	2080	**4.0e6**	1.6e8	5.8e6	955	8.5e6	**922**	1.4e7	955	8.3e6	**922**	1.4e7	955	6.9e6	955	6.7e6	955	8.7e6	949	1.6e7	955	8.5e6	949	1.5e7	955	1.0e7	955	1.0e7
u	2.1e4	2.0e5	1.0e8	4.4e5	9385	1.0e6	**8186**	3.3e6	6.7e4	2.5e4	6.3e4	5.1e4	5.0e4	2.1e5	4.8e5	5.2e4	9385	5.0e5	8382	2.9e6	6.7e4	4.9e4	6.4e4	5.4e4	1.1e5	8.6e4	5.8e5	**868**
v	1.9e4	7.6e5	1.1e8	6.7e5	8470	1.4e6	*	*	7.4e4	5455	7.1e4	2.2e4	1.2e5	8.2e4	3.4e5	2.5e4	8470	1.1e6	**6838**	3.5e6	7.4e4	**4474**	7.1e4	5.4e4	6.1e4	1.4e5	8.1e5	3.6e4
w	3.4e4	2.2e5	8.6e7	2.0e5	1.5e4	1.6e5	*	*	5.0e5	324	*	*	3.8e5	364	1.4e6	**10**	1.5e4	2.2e5	**1.2e4**	1.8e6	5.0e5	**10**	*	*	4.7e5	1.0e5	1.2e6	137
x	1.3e4	6.3e5	*	*	6130	3.2e5	*	*	6130	3.3e5	*	*	6130	3.4e5	6130	3.4e5	6130	2.5e5	**4400**	3.6e6	6130	**2.4e5**	**4400**	3.6e6	6130	3.6e5	6130	3.7e5
y	1.4e4	6.7e5	8.4e7	5.4e5	6445	6.8e5	**4759**	4.2e6	1.3e5	**1.4e4**	*	*	5.1e4	1.4e5	2.2e5	1.1e5	6445	4.3e5	*	*	1.3e5	3.0e4	*	*	7.7e4	2.3e5	2.3e5	2.2e5
z	4885	1.2e6	1.0e8	2.1e6	2230	2.2e6	**2081**	5.9e6	4782	**8.3e5**	4807	6.1e6	2506	1.7e6	8669	1.4e6	2230	3.5e6	2209	9.6e6	4782	2.2e6	4573	6.1e6	2948	3.2e6	4827	1.8e6
A	8779	3.9e5	5.4e7	5.4e5	4000	7.5e5	**3145**	4.7e6	1.5e4	**2.5e5**	1.4e4	1.3e6	1.1e4	5.0e5	7290	7.8e5	4000	4.1e5	*	*	1.5e4	3.2e5	*	*	6352	3.5e5	2.1e4	3.7e5
B	1.2e4	6.0e5	1.3e8	6.4e5	5335	8.3e5	5042	3.6e6	8.9e4	**1.1e4**	8.8e4	1.4e5	8.4e4	1.7e5	8.6e5	3.0e4	5335	5.2e5	**4423**	4.5e6	8.9e4	1.3e4	*	*	4.9e4	5.4e5	9.1e5	4.4e4
C	2.6e4	**4.0e4**	5.6e7	7.0e4	1.2e4	1.4e5	*	*	1.3e4	2.1e5	*	*	1.2e4	1.1e5	1.4e4	9.2e4	1.2e4	2.0e5	*	*	1.3e4	1.5e5	**1.1e4**	1.3e6	1.2e4	1.1e5	1.4e4	5.8e5
D	5248	**7.6e5**	9.9e7	8.7e5	2395	1.2e6	2238	4.4e6	2395	1.2e6	2238	4.5e6	2395	1.4e6	2395	1.4e6	2395	2.3e6	**2198**	3.9e6	2395	2.3e6	**2198**	3.9e6	2395	1.3e6	2395	1.3e6
E	9040	1.2e6	8.8e7	3.7e5	4139	8.1e5	**3949**	4.0e6	8759	5.7e5	8264	1.0e6	4977	1.0e6	2.3e4	4.7e5	4139	2.4e5	*	*	8759	6.3e5	*	*	6950	7.6e5	3.1e4	**1.6e5**

Tabela 11: Upperbound e Tempo de Execução das estratégias propostas para o USP

Instância	n	p	Upperbound				Tempo de Execução			
			BRKGA	ILS	Híbrido	MIP	BRKGA	ILS	Híbrido	MIP
Echocardiogram	45	6	517.1	311.6	210.5	**118.9**	58.8	44.7	**43.5**	*1800.0*
Fertility	75	9	2842.1	2018.4	1998.9	**1964.1**	**67.0**	164.3	124.9	*1800.0*
Hayes-roth	132	4	0.0	0.0	0.0	0.0	0.7	2.2	0.6	**0.0**
Iris (Classe 0)	37	4	360.8	360.8	360.8	360.8	56.4	**13.5**	24.1	1487.3
Iris (Classe 1)	37	4	217.8	217.8	217.8	217.8	57.3	**14.6**	25.2	*1800.0*
Iris (Classe 2)	38	4	72.2	72.2	72.2	72.2	56.7	**16.3**	25.9	*1800.0*
SPECT	60	22	11762.8	**11293.3**	**11293.3**	**11293.3**	**63.2**	64.7	148.0	1725.2
SPECTF	60	44	26963.2	25213.5	25176.3	**25160.1**	**64.8**	123.6	117.4	115.5
Soybean-small	35	35	8323.7	**8275.0**	**8275.0**	**8275.0**	57.4	42.4	40.9	**21.4**

que esse limite é excedido são indicados pelo valor de *1800* segundos em itálico na coluna *Tempo de Execução*.

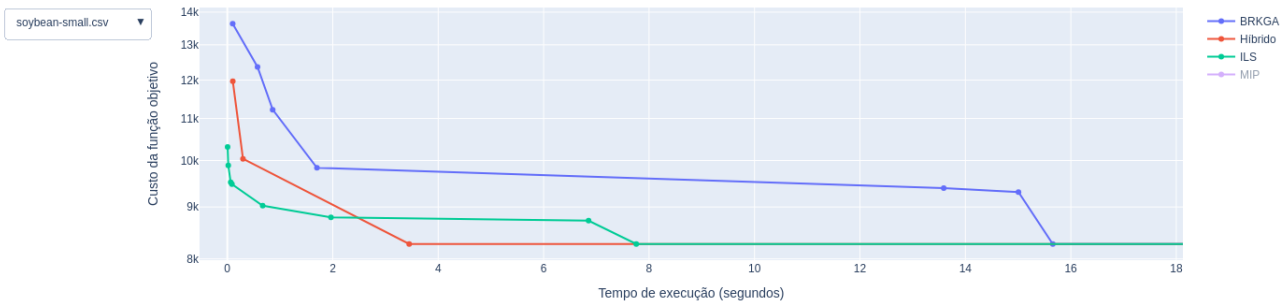


Figura 8: Evolução da melhor solução encontrada em função do tempo de execução por cada algoritmo em uma instância exemplo

Os resultados indicam consistentemente que os melhores *upperbounds* são alcançados pelo MIP, embora seja crucial ressaltar que o mesmo opera sob um tempo limite de 30 minutos, enquanto as heurísticas demandam apenas alguns segundos para chegar a resultados comparáveis. O MIP demonstrou a otimalidade do *upperbound* em 5 de 9 instâncias avaliadas, sendo que em apenas 1 delas as heurísticas não alcançaram o ótimo na média. Além disso, das quatro instâncias restantes, em 2 ocasiões as heurísticas atingiram o mesmo *upperbound* que o MIP.

Ademais, em termos de *upperbound*, observou-se que o BRKGA apresenta consistentemente um desempenho inferior quando comparado ao ILS, independentemente de terminar mais rapidamente, mais lentamente ou com um tempo de execução semelhante. Similarmente, o ILS demonstrou ser consistentemente inferior ao Híbrido. A Figura 8 evidencia esse comportamento em um exemplo representativo: o BRKGA inicia com um custo elevado, porém converge após um período prolongado. Por outro lado, o ILS inicia com um custo inferior que rapidamente diminui ainda mais, mas logo após permanece estagnado por um intervalo considerável de tempo. O Híbrido, por sua vez, inicia com um

custo alto devido à execução das suas iterações iniciais internas do BRKGA. No entanto, sua abordagem combinada com o ILS resulta em uma convergência mais rápida e estável para soluções de menor custo.

Além disso, os tempos de execução revelam que não há uma correlação direta entre as dimensões das instâncias n e p e o tempo de execução. Em última análise, recomenda-se o uso do Híbrido devido ao menor custo computacional e à robustez demonstrada. Entretanto, para cenários em que o custo financeiro referente a aquisição de licenças de *solvers* comerciais robustos e o tempo não sejam preocupações, mas a qualidade da solução seja prioritária, a utilização do MIP pode ser considerada uma opção mais adequada.

6.3 Avaliação da efetividade do USP

A Tabela 12 apresenta o conjunto de instâncias selecionados do repositório UCI Machine Learning (DUA; GRAFF, 2017) para o escopo dos próximos experimentos. Cada linha contém o identificador da instância (coluna *Inst.*), seu nome público (*Nome*), a quantidade de observações (n), de características (p) e de rótulos distintos do conjunto de saída (K). Ademais, cada *dataset* foi divido de forma aleatória em treino e teste, correspondendo respectivamente a 75% e 25% do todo.

Tabela 12: Descrição das instâncias da base UCI utilizadas.

Inst.	Nome	n	p	K
a	Acute-inflammations-2	120	6	2
b	SPECT-heart	80	22	2
c	Iris	150	4	3
d	Hayes-roth	132	4	3
e	Congressional-voting-records	435	16	2
f	Thyroid-disease-new-thyroid	215	5	3
g	Wine	178	13	3
h	Balance-scale	625	4	3
i	Seeds	210	7	3
j	Haberman-survival	306	3	2
k	Ionosphere	351	34	2
l	Heart-disease-Cleveland	303	13	5
m	Banknote-authentication	1372	4	2
n	Dermatology	366	34	6
o	Climate-model-crashes	540	18	2

As Tabelas 13, 14 e 15 comparam a utilização das instâncias originais e das reduzidas pelo USP em métodos de aprendizado de máquina. Mais precisamente, são apresentadas duas técnicas: O OCT (do inglês, Optimal Classification Trees, (BERTSIMAS; DUNN, 2017)), uma formulação MIP de custo computacional bastante elevado para geração de árvores de decisão ótimas; e CART (do inglês, Classification and Regression Trees, (BREIMAN et al., 1984)), um conhecido método heurístico que escala bem para grandes instâncias e, portanto, dispensa a utilização de estratégias como USP. Assim

sendo, as colunas *OCT* e *CART* denotam o desempenho dessas respectivas técnicas recebendo como entrada o conjunto de instâncias originais; enquanto que OCT_u corresponde a execução do OCT que tem como entrada os *subsets* gerados pelo USP Híbrido, e cuja instância de entrada é proporcional a dimensão VC da árvore a ser construída, calculado a partir do Algoritmo 7. Por fim, as colunas OCT_r representam uma execução do OCT recebendo um *subset* completamente aleatório, de tamanho similar aquele gerado pelo USP, com intuito de avaliar a efetividade deste em comparação com uma simples amostragem aleatória. Ademais, as colunas n indicam a quantidade de observações que cada estratégia de fato recebeu como entrada, ao passo que cada linha representa uma instância identificada pela coluna *Inst.*. Com exceção dos algoritmos determinísticos (OCT), todos os valores representam a média de 10 execuções.

A análise realizada fixando-se o parâmetro de profundidade $D = 1$, exposto na Tabela 13, revelou uma redução substancial das dimensões de n, diminuindo de 260 para 44, correspondendo a uma média de 83% de diminuição. Esta redução teve um impacto significativo no tempo médio de execução, caindo de 10 segundos para 0.14. Mesmo procedendo-se com a exclusão do *outlier* da instância "o", a redução ainda mantém-se 95%. Embora esta otimização tenha sido expressiva, a maioria dos tempos de execução do OCT permaneceram dentro de limites toleráveis, dispensando, portanto, a necessidade de aplicação de subamostragem, ainda que seus desempenhos tenham sido satisfatórios.

Em relação à acurácia, a redução do número de instâncias permitiu que tanto o OCT_r quanto o OCT_u alcançassem os menores erros durante o treinamento, mas isso não se refletiu completamente nos resultados dos testes. Houve variações nessa métrica entre o treinamento e o teste: o OCT perdeu 0.072 pontos, o OCT_u perdeu 0.046 pontos, enquanto o OCT e CART obtiveram pequenos ganhos de 0.009 e 0.008, respectivamente. A média da acurácia mais alta foi observada no OCT com 0.739, seguido por CART com 0.723 e OCT_u com 0.721, mostrando-se próximo do desempenho do CART. Por fim, o OCT_r teve a menor generalização, com uma acurácia de 0.707, apesar de ter alcançado a maior acurácia durante o treinamento.

A análise para a profundidade $D = 2$, contida na Tabela 14, apresentou resultados semelhantes, exceto pela exclusão da instância b devido ao seu tamanho ser menor que o requerido para subamostragem de acordo com a dimensão VC. Apesar disso, houve uma redução média de 70% nas dimensões de n (de 274 para 81). Enquanto o OCT excedeu o tempo limite em 5 instâncias, o OCT_r excedeu em 2 e o OCT_u em 1. Ainda assim, o tempo de execução foi mais consistente entre OCT_u e OCT_r, com médias de 1386 e 1469 segundos, respectivamente, enquanto o OCT levou, em média, 2714 segundos, o que corresponde a quase 96% a mais de tempo que o OCT_u. Entretanto, embora as duas propostas de subamostragem tenham obtido altas acurácias no treinamento, houve uma queda nos resultados dos testes, com o OCT alcançando a maior acurácia nesse quesito,

0.835, seguido pelo OCT_u com 0.830, CART com 0.827 e, por fim, OCT_r com 0.826. Dessa forma, é possível observar que o OCT_u obteve sucesso ao posicionar-se entre o OCT e o CART, superando ainda o desempenho do OCT_r.

Por sua vez, na profundidade $D = 3$, expressa pela Tabela 15, cinco instâncias foram suprimidas a semelhança da tabela anterior, levando o OCT a exceder o tempo limite em todas as instâncias restantes, incluindo um estouro de memória na instância k. Devido aos maiores requisitos da dimensão VC, a redução de dimensionalidade foi de aproximadamente 56% (de 341 para 151). Enquanto o OCT alcançou o tempo máximo de execução na média (7200 segundos), o OCT_r e o OCT_u completaram duas instâncias, refletindo em médias menores. O overfitting tornou-se mais evidente nesta complexidade, com o OCT_u e o OCT_r obtendo as maiores acurácias médias no treinamento (0.874 e 0.871, respectivamente), enquanto no teste o OCT_u alcançou a maior média (0.808), seguido pelo OCT com 0.804, OCT_r com 0.800 e, enfim, CART com 0.790. Todos os algoritmos começaram a apresentar efeitos sutis de overfitting devido à complexidade aumentada e ao custo computacional elevado, como evidenciado pela consistente queda na acurácia entre treino e teste.

6.4 Avaliação da efetividade do OCT de topologia dinâmica

Fazendo uso da mesma estrutura que as tabelas anteriormente apresentadas, a Tabela 17 destaca os métodos apresentados na Seção 5, nomeadamente OCT_{din} e OCT^g_{din}, que correspondem, respectivamente, as estratégias Exploratórias e Gulosas descritas na Seção 5.4. Concomitantemente, os métodos OCT e OCT_u são extraídos da Tabela 14, na qual $D = 2$, e reapresentados em suas respectivas colunas para tornar a visualização mais conveniente: A sua escolha se deu por OCT_u apresentar tempo de execução médio similar as técnicas estudadas nesta seção. Considerando que os métodos de topologia dinâmica não possuem o parâmetro de profundidade D, mas suas árvores crescem e apresentam topologia irregular, as colunas *Folhas* indicam a quantidade de nós folhas da melhor árvore encontrada por esses métodos; Para OCT e OCT_u, essa coluna é suprimida, pois seu valor é sempre 4, dado que $D = 2$.

Apesar do desempenho limitado, os resultados obtidos ao comparar os quatro métodos discutidos revelaram conclusões significativas em termos de tempos de execução e acurácia nos testes. Notavelmente, os algoritmos OCT_u, OCT_{din} e OCT^g_{din} demonstraram tempos de execução semelhantes, enquanto o OCT exibiu um tempo quase duas vezes maior. Em apenas uma ocasião cada, os algoritmos OCT_{din} e OCT^g_{din} superaram o desempenho em velocidade. Por outro lado, em todas as 12 instâncias restantes, o OCT_u se mostrou o mais rápido, sem o OCT conseguir ser mais veloz em nenhuma das instâncias.

Tabela 13: Comparação de Tempo de Execução e Acurácia do OCT de acordo com a estratégia de amostragem para D=1

Inst.	n			Tempo de Execução			Acurácia (Treino)				Acurácia (Teste)			
	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	CART	OCT	OCT_r	OCT_u	CART
a	90	35	36	0.084	**0.027**	0.031	0.911	**0.937**	0.917	0.911	**0.933**	0.893	**0.933**	**0.933**
b	60	55	52	0.061	0.049	**0.032**	0.717	**0.738**	**0.738**	0.717	**0.750**	0.630	0.710	**0.750**
c	112	35	38	0.187	**0.019**	0.036	0.670	**0.709**	0.688	0.670	**0.658**	**0.658**	0.647	**0.658**
d	132	35	44	0.364	**0.047**	0.053	0.485	0.589	**0.595**	0.477	**0.714**	0.507	0.457	0.536
e	174	55	50	0.325	**0.043**	0.044	0.966	0.960	**0.980**	0.966	0.983	0.983	0.983	0.983
f	161	35	38	1.417	**0.023**	0.034	0.832	0.874	**0.921**	0.832	0.759	0.770	**0.778**	0.759
g	133	45	49	0.418	**0.068**	0.085	0.684	**0.733**	0.683	0.684	**0.733**	0.720	0.724	**0.733**
h	468	35	30	1.997	0.048	**0.027**	0.647	0.714	**0.733**	0.647	0.599	**0.624**	0.599	0.599
i	157	35	41	0.588	**0.017**	0.239	0.669	**0.731**	0.634	0.656	**0.660**	0.577	0.623	**0.660**
j	229	25	24	0.631	0.026	**0.022**	0.760	0.816	**0.917**	0.747	**0.740**	0.704	0.727	0.727
k	263	65	62	9.518	0.420	**0.380**	0.833	**0.880**	0.845	0.833	0.773	0.802	**0.845**	0.773
l	222	45	44	7.184	**0.290**	0.291	0.568	**0.658**	0.545	0.541	0.560	0.573	**0.587**	0.493
m	1029	35	34	11.356	**0.026**	0.040	0.861	**0.914**	0.882	0.861	**0.831**	0.825	**0.831**	**0.831**
n	268	65	69	4.983	**0.697**	0.701	0.515	0.514	**0.516**	0.504	0.467	0.471	0.482	**0.489**
o	405	55	60	114.368	**0.133**	0.167	0.914	0.916	**0.917**	0.914	**0.919**	0.864	0.889	**0.919**
Média	260	43	44	10.232	**0.129**	0.145	0.735	**0.779**	0.767	0.731	**0.739**	0.707	0.721	0.723

Tabela 14: Comparação de Tempo de Execução e Acurácia do OCT de acordo com a estratégia de amostragem para D=2

Inst.			n	Tempo de Execução			Acurácia (Treino)				Acurácia (Teste)			
	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	CART	OCT	OCT_r	OCT_u	CART
a	90	65	64	0.018	**0.014**	**0.014**	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
c	112	65	61	34.724	6.012	**4.149**	0.964	0.963	**0.974**	0.964	**0.947**	0.937	0.942	**0.947**
d	132	65	66	46.484	**14.163**	16.064	0.598	**0.655**	0.648	0.553	0.536	0.600	**0.607**	0.536
e	174	105	107	52.815	18.370	**14.480**	**0.966**	**0.966**	0.964	**0.966**	**0.983**	0.969	0.976	**0.983**
f	161	65	69	210.883	**5.804**	5.831	0.969	0.982	**0.989**	0.944	**0.944**	0.915	0.933	0.907
g	133	85	81	290.540	40.903	**27.749**	0.977	0.981	**0.983**	0.917	**0.933**	0.907	0.907	0.889
h	468	65	70	309.675	**12.588**	17.967	0.731	**0.822**	0.814	0.707	0.675	0.689	0.701	**0.713**
i	157	65	67	440.739	**10.296**	19.087	0.936	**0.969**	0.955	0.917	**0.943**	0.879	0.887	0.925
j	229	45	42	610.919	3.762	**1.970**	0.777	0.862	**0.929**	0.777	**0.753**	0.668	0.727	**0.753**
k	263	125	120	7200.000	7200.000	7200.000	0.913	**0.928**	0.923	0.913	**0.909**	0.886	0.877	**0.909**
l	222	85	86	7200.000	**3242.126**	3915.804	0.613	**0.678**	0.660	0.599	0.560	**0.581**	0.536	0.533
m	1029	65	64	7200.000	**9.154**	14.642	0.908	**0.978**	0.969	0.907	**0.892**	0.875	0.854	**0.892**
n	268	125	128	7200.000	7200.000	**3791.772**	0.765	0.765	**0.770**	0.660	0.744	0.769	**0.787**	0.678
o	405	105	108	7200.000	**2810.831**	4381.288	0.936	**0.964**	0.935	0.914	0.867	0.889	0.890	**0.919**
Média	274	80	81	2714.057	1469.573	**1386.487**	0.861	**0.894**	**0.894**	0.838	**0.835**	0.826	0.830	0.827

Tabela 15: Comparação de Tempo de Execução e Acurácia do OCT de acordo com a estratégia de amostragem para D=3

Inst.	n			Tempo de Execução			Acurácia (Treino)				Acurácia (Teste)			
	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	CART	OCT	OCT_r	OCT_u	CART
d	132	125	88	7200.000	7200.000	7200.000	0.689	0.677	**0.730**	0.614	0.679	0.686	**0.700**	0.500
f	161	125	117	7200.000	7107.512	**4969.382**	0.981	0.989	**0.995**	0.944	**0.981**	0.922	0.926	0.907
h	468	125	127	7200.000	7200.000	7200.000	0.756	**0.850**	0.839	0.754	0.701	0.726	**0.744**	0.707
i	157	125	122	7200.000	7200.000	7200.000	0.936	0.966	**0.982**	0.930	0.925	0.909	0.906	**0.943**
j	229	85	88	7200.000	7200.000	7200.000	0.803	0.849	**0.861**	0.777	0.623	0.603	0.673	**0.753**
k	263	245	240	ML	**7200.000**	**7200.000**	ML	**0.916**	0.896	0.913	ML	0.884	0.868	**0.898**
l	222	165	162	7200.000	7200.000	7200.000	0.649	**0.658**	0.633	0.644	**0.600**	0.584	0.567	0.573
m	1029	125	132	7200.000	6948.832	**6820.302**	0.876	**0.995**	0.994	0.907	0.875	0.917	**0.927**	0.892
n	268	245	240	7200.000	7200.000	7200.000	**0.869**	0.856	0.864	0.787	**0.922**	0.871	0.858	0.833
o	405	205	203	7200.000	7200.000	7200.000	0.938	**0.954**	0.949	0.943	**0.926**	0.902	0.916	0.896
Média	341	157	151	7200.000	7165.634	**6938.968**	0.833	0.871	**0.874**	0.821	0.804	0.800	**0.808**	0.790

Tabela 16: Comparação de Tempo médio de Execução e Acurácia do OCT de acordo com a estratégia de amostragem para diferentes profundidades

D	n			Tempo de Execução			Acurácia (Treino)				Acurácia (Teste)			
	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	OCT	OCT_r	OCT_u	CART	OCT	OCT_r	OCT_u	CART
1	260	43	44	10.232	**0.129**	0.145	0.735	**0.779**	0.767	0.731	**0.739**	0.707	0.721	0.723
2	274	80	81	2714.057	1469.573	**1386.487**	0.861	**0.894**	**0.894**	0.838	**0.835**	0.826	0.830	0.827
3	341	157	151	7200.000	7165.634	**6938.968**	0.833	0.871	**0.874**	0.821	0.804	0.800	**0.808**	0.790

Ao excluir o OCT_u da análise, o algoritmo OCT obteve três dos melhores tempos, todos em instâncias de menor escala. Entretanto, essa contribuição não influenciou a média final, mantendo o OCT aproximadamente duas vezes mais lento do que os novos algoritmos. Nas demais instâncias, os modelos de topologia dinâmica, OCT_{din} e OCT^g_{din}, demonstraram consistentemente maior rapidez.

No que se refere à acurácia nos testes, os novos métodos apresentaram resultados pouco satisfatórios. O algoritmo OCT alcançou a maior média, registrando 0.835, seguido pelo OCT_u com 0.830. O OCT_{din} obteve um resultado ligeiramente inferior, com 0.827, e o OCT^g_{din} registrou a menor acurácia, atingindo 0.804. Importante ressaltar que tanto o OCT_{din} quanto o OCT^g_{din} apresentaram as piores acurácias também no treinamento, indicando que este comportamento não pode ser atribuído ao overfitting.

A hipótese de que o baixo desempenho dos novos métodos pode estar relacionado ao número de folhas foi considerada. Ao filtrar as 5 instâncias em que o OCT_{din} resultou em 4 ou mais folhas, a acurácia foi de 0.774, enquanto o OCT atingiu 0.763 e o OCT_u 0.753. De modo similar, ao aplicar o mesmo filtro ao OCT^g_{din} (em 3 instâncias), a acurácia foi de 0.848, comparada a 0.784 do OCT e 0.798 do OCT_u. Considerando o baixo número de instâncias envolvidas nesta análise, investigações adicionais mitigando esse problema se fazem necessárias para determinar se tal observação se mantém.

Tabela 17: Comparação de Tempo de Execução e Acurácia do OCT regular e de topologia dinâmica

Inst.	Folhas		Tempo de Execução				Acurácia (Treino)				Acurácia (Teste)			
	OCT_{din}	OCT^{g}_{din}	OCT	OCT_u	OCT_{din}	OCT^{g}_{din}	OCT	OCT_u	OCT_{din}	OCT^{g}_{din}	OCT	OCT_u	OCT_{din}	OCT^{g}_{din}
a	3	2	0.018	**0.014**	97.660	107.800	**1.000**	**1.000**	**1.000**	0.911	**1.000**	**1.000**	**1.000**	0.933
c	3	3	34.724	**4.149**	139.105	188.167	0.964	**0.974**	0.938	0.938	**0.947**	0.942	0.921	0.921
d	1	1	46.484	**16.064**	76.058	61.080	0.598	**0.648**	0.386	0.386	0.536	**0.607**	0.464	0.464
e	2	2	52.815	**14.480**	46.134	98.285	**0.966**	0.964	**0.966**	**0.966**	**0.983**	0.976	**0.983**	**0.983**
f	3	1	210.883	**5.831**	103.602	92.330	0.969	**0.989**	0.925	0.714	**0.944**	0.933	0.907	0.648
g	3	4	290.540	**27.749**	100.945	201.088	0.977	**0.983**	0.857	0.977	**0.933**	0.907	0.867	0.911
h	4	4	309.675	**17.967**	123.032	137.894	0.731	**0.814**	0.707	0.707	0.675	**0.701**	**0.701**	**0.701**
i	4	3	440.739	**19.087**	181.103	139.276	0.936	**0.955**	0.879	0.904	**0.943**	0.887	0.868	0.849
j	2	2	610.919	**1.970**	230.556	215.864	0.777	**0.929**	0.747	0.747	**0.753**	0.727	0.727	0.727
k	3	2	7200.000	7200.000	**262.988**	277.773	0.913	**0.923**	0.909	0.821	0.909	0.877	**0.920**	0.875
l	6	1	7200.000	**3915.804**	7200.000	7200.000	0.613	**0.660**	0.599	0.532	**0.560**	0.536	**0.560**	**0.560**
m	4	2	7200.000	**14.642**	152.581	291.151	0.908	**0.969**	0.888	0.861	**0.892**	0.854	0.886	0.831
n	6	6	7200.000	**3791.772**	7200.000	7200.000	0.765	0.770	0.851	**0.951**	0.744	0.787	0.856	**0.933**
o	1	1	7200.000	4381.288	670.222	**122.351**	**0.936**	0.935	0.914	0.914	0.867	0.890	**0.919**	**0.919**
Média	3	2	2714.057	1386.487	1184.570	**1166.647**	0.861	**0.894**	0.826	0.809	**0.835**	0.830	0.827	0.804

7 Conclusões

Neste trabalho, abordamos o problema da geração de uma Árvore de Classificação Ótima utilizando técnicas de Programação Linear Inteira Mista. Propomos uma reformulação, intitulada nOCT, para o algoritmo estado-da-arte *Optimal Classification Trees*, OCT, bem como introduzimos inequações válidas para o problema por meio da geração de um grafo de precedências, além de apresentarmos uma estratégia de *branch-and-cut* para a inserção de restrições sob demanda no modelo.

Comparamos e discutimos as estratégias propostas com os métodos disponíveis na literatura, e constatamos que as estratégias nOCT e variantes são capazes de encontrar uma árvore ótima em menor tempo computacional, ao passo que a inclusão das inequações de precedência e da estratégia de *branch-and-cut* podem, em alguns casos, proporcionar ainda um pequeno ganho adicional no tempo de execução. Estas novas inequações diminuíram em mais da metade o número de nós resolvidos pelo *branch-and-bound*, mas por causa da sua quantidade excessiva, convém a realização de investigações adicionais de estratégias mais eficientes para suas inserções.

Propomos ainda algumas estratégias para a redução da dimensionalidade, mantendo a qualidade e representatividade, das instâncias, obtendo sucesso ao aplicá-las diretamente em conjunto com o OCT, porém com resultados mistos quando empregados em estratégias de crescimento progressivo das árvores.

Como possíveis oportunidades de investigações adicionais desse tema, listamos a possibilidade de explorar reformulações ou novas desigualdades fortes para a estratégia de grafo de precedência de modo a incrementar a relaxação linear enquanto diminui-se a grande quantidade de cortes atualmente adicionados, investigar estratégias de *branch-and-cut* mais eficientes, realizar um estudo sobre a inserção dinâmica de exemplares do conjunto de entrada dentro do modelo de modo a reduzir o *overhead* de reinicializações dos *solvers*, bem como implementar formulações ou estratégias mais adequadas a árvores com valores maiores de D.

Referências

ARLOT, Sylvain; CELISSE, Alain. A survey of cross-validation procedures for model selection. **Statistics Surveys**, Amer. Statist. Assoc., the Bernoulli Soc., the Inst. Math. Statist., e the Statist. Soc. Canada, v. 4, none, p. 40–79, 2010. DOI: 10.1214/09-SS054. Disponível em: ¡https://doi.org/10.1214/09-SS054¿.

ARTHANARI, T. S.; DODGE, Yadolah. **Mathematical programming in statistics**. New York: Wiley, 1993. (Wiley classics library). ISBN 9780471592129.

BERTSIMAS, Dimitris; DELARUE, Arthur et al. **The Price of Interpretability**. [S.l.: s.n.], 2019. arXiv: 1907.03419 [cs.LG].

BERTSIMAS, Dimitris; DUNN, Jack. Optimal classification trees. **Machine Learning**, v. 106, n. 7, p. 1039–1082, 2017. DOI: 10.1007/s10994-017-5633-9. Disponível em: ¡https://doi.org/10.1007/s10994-017-5633-9¿.

BERTSIMAS, Dimitris; ORFANOUDAKI, Agni; WIBERG, Holly. Interpretable clustering: an optimization approach. **Machine Learning**, Springer, v. 110, n. 1, p. 89–138, 2021.

BIRZHANDI, Pardis; KIM, Kyung Tae; YOUN, Hee Yong. Reduction of training data for support vector machine: a survey. **Soft Computing**, v. 26, n. 8, p. 3729–3742, 2022. ISSN 1433-7479. DOI: 10.1007/s00500-022-06787-5. Disponível em: ¡https://doi.org/10.1007/s00500-022-06787-5¿.

BREIMAN, Leo et al. **Classification and regression trees**. [S.l.]: Routledge, 1984.

CHOLLET, Francois. **Deep learning with Python**. [S.l.]: Simon e Schuster, 2021.

CRISTERNA, Arturo Rodrıguez. **Clasificación Multiclase con Aprendizaje Especializado en Parejas de Clases Orientado al BI-RADS para Ultrasonografía**. 2017. Tese (Doutorado) – Instituto Politécnico Nacional.

DUA, Dheeru; GRAFF, Casey. **UCI Machine Learning Repository**. [S.l.: s.n.], 2017. Disponível em: ¡http://archive.ics.uci.edu/ml¿.

GONÇALVES, José; RESENDE, Mauricio. Biased random-key genetic algorithms for combinatorial optimization. **J. Heuristics**, v. 17, p. 487–525, out. 2011. DOI: 10.1007/s10732-010-9143-1.

HARDT, Moritz; PRICE, Eric; SREBRO, Nathan. Equality of opportunity in supervised learning. **Advances in Neural Information Processing Systems**, v. 29, p. 3315–3323, 2016.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. Overview of Supervised Learning. In: THE Elements of Statistical Learning: Data Mining, Inference, and Prediction. [S.l.]: Springer New York, 2009. cap. 1, p. 9–41.

HOLLAND, John H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. [S.l.]: The MIT Press, abr. 1992.

JAMES, Gareth et al. **An Introduction to Statistical Learning: with Applications in R**. New York, NY: Springer US, 2021. (Springer Texts in Statistics). ISBN 9781071614174. DOI: 10.1007/978-1-0716-1418-1. Disponível em: ¡https://link.springer.com/10.1007/978-1-0716-1418-1¿. Acesso em: 14 jan. 2023.

KAHNEMAN, Daniel; SIBONY, Olivier; SUNSTEIN, Cass R. **Noise - A Flaw In Human Judgement**. 1. ed. United Kingdom: William Collins Publishing, 2021.

KAMIRAN, Faisal; CALDERS, Toon. Data preprocessing techniques for classification without discrimination. In: EUROPEAN Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. [S.l.]: Springer, 2012. P. 97–112.

KARP, Richard M. Reducibility among combinatorial problems. In: COMPLEXITY of computer computations. [S.l.]: Springer, 1972. P. 85–103.

KISH, Leslie. Survey Sampling. **Wiley Series in Probability and Mathematical Statistics**, Wiley, 1965.

KULL, Meelis; SILVA FILHO, Telmo; FLACH, Peter. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. In: INTERNATIONAL Conference on Machine Learning. [S.l.]: PMLR, 2017. P. 1587–1595.

LAURENT, Hyafil; RIVEST, Ronald L. Constructing optimal binary decision trees is NP-complete. **Information processing letters**, v. 5, n. 1, p. 15–17, 1976.

LI, Xiaoou; CERVANTES, Jair; YU, Wen. Fast Classification for Large Data Sets via Random Selection Clustering and Support Vector Machines. **Intell. Data Anal.**, IOS Press, NLD, v. 16, n. 6, p. 897–914, nov. 2012. ISSN 1088-467X.

LOURENÇO, Helena R.; MARTIN, Olivier C.; STÜTZLE, Thomas. Iterated Local Search. In: **Handbook of Metaheuristics**. Edição: Fred Glover e Gary A. Kochenberger. Boston, MA: Springer US, 2003. P. 320–353. ISBN 978-0-306-48056-0. DOI: 10.1007/0-306-48056-5_11. Disponível em: ¡https://doi.org/10.1007/0-306-48056-5_11¿.

NG, Andrew Y.; JORDAN, Michael I. Feature selection, L1 vs. L2 regularization, and rotational invariance. In: PROCEEDINGS of the 21st International Conference on Machine Learning (ICML). Banff, Alberta, Canada: [s.n.], 2004. P. 78.

NICULESCU-MIZIL, Alexandru; CARUANA, Rich. Predicting Good Probabilities with Supervised Learning. In: PROCEEDINGS of the 22nd International Conference on Machine Learning. Bonn, Germany: Association for Computing Machinery, 2005. (ICML '05), p. 625–632. ISBN 1595931805. DOI: 10.1145/1102351.1102430. Disponível em: ¡https://doi.org/10.1145/1102351.1102430¿.

PANDA, Navneet; CHANG, Edward Y.; WU, Gang. Concept Boundary Detection for Speeding up SVMs. In: PROCEEDINGS of the 23rd International Conference on Machine Learning. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006. (ICML '06), p. 681–688. ISBN 1595933832. DOI: 10.1145/1143844.1143930. Disponível em: ¡https://doi.org/10.1145/1143844.1143930¿.

PARMENTIER, Axel; VIDAL, Thibaut. Optimal Counterfactual Explanations in Tree Ensembles. In_____. **Proceedings of the 38th International Conference on Machine Learning**. [S.l.]: PMLR, jul. 2021. v. 139. (Proceedings of Machine Learning Research), p. 8422–8431.

QUINLAN, J Ross. **C4. 5: programs for machine learning**. [S.l.]: Elsevier, 2014.

_____. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

SOUZA, Cleber Batista De. **Árvores de decisão: a evolução do CART ao BART**. Dez. 2021. Mestrado em Estatística – Universidade de São Paulo, São Paulo. DOI: 10.11606/D.45.2021.tde-05042022-095004. Disponível em: ¡https://www.teses.usp.br/teses/disponiveis/45/45133/tde-05042022-095004/¿. Acesso em: 31 dez. 2023.

STROHMAIER, Erich et al. **Top500 Supercomputer Sites**. [S.l.: s.n.], 2015. Disponível em: ¡https://top500.org/statistics/perfdevel/¿. Acesso em: 26 jun. 2022.

VAPNIK, Vladimir. Support-Vector Networks. **Machine Learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

VAPNIK, Vladimir N. **The Nature of Statistical Learning Theory**. 1. ed. New York: Springer New York, 2013.

VERWER, Sicco; ZHANG, Yingqian. Learning Optimal Classification Trees Using a Binary Linear Program Formulation. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 33, n. 01, p. 1625–1632, jul. 2019. DOI: 10.1609/aaai.v33i01.33011624. Disponível em: ¡https://ojs.aaai.org/index.php/AAAI/article/view/3978¿.

VIDAL, Thibaut; SCHIFFER, Maximilian. Born-Again Tree Ensembles. In: PROCEEDINGS of the 37th International Conference on Machine Learning. [S.l.]: JMLR.org, 2020. (ICML'20).

YILDIZ, Olcay Taner. VC-dimension of univariate decision trees. **IEEE Trans Neural Netw Learn Syst**, v. 2, n. 26, p. 378–387, 2015.

ZHU, Fa et al. Neighbors' distribution property and sample reduction for support vector machines. **Applied Soft Computing**, v. 16, p. 201–209, 2014. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2013.12.009. Disponível em: ¡https://www.sciencedirect.com/science/article/pii/S1568494613004298¿.