KALI NUNES FERREIRA

ANÁLISE DO ESPAÇO DE INSTÂNCIAS:
UM ESTUDO SOBRE A FERRAMENTA MATILDA

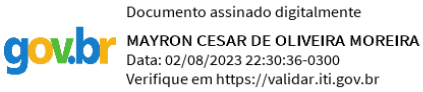
LAVRAS – MG

2023

KALI NUNES FERREIRA

ANÁLISE DO ESPAÇO DE INSTÂNCIAS:
UM ESTUDO SOBRE A FERRAMENTA MATILDA

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação



Prof. Dr. Mayron César de Oliveira Moreira

Orientador

LAVRAS – MG

2023

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Ferreira, Kali Nunes

Análise do Espaço de Instâncias : Um estudo sobre a ferramenta MATILDA / Kali Nunes Ferreira. 1^a ed. – Lavras : UFLA, 2023.

43 p. : il.

Monografia(bacharelado)–Universidade Federal de Lavras, 2023.
Orientador: Prof. Dr. Mayron César de Oliveira Moreira.
Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

KALI NUNES FERREIRA

ANÁLISE DO ESPAÇO DE INSTÂNCIAS: UM ESTUDO SOBRE A FERRAMENTA MATILDA

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação

APROVADA em 25 de Julho de 2023.

Prof. Ph.D. Mayron César de Oliveira Moreira	UFLA
Prof. Dr. Janderson Rodrigo de Oliveira	UFLA
Prof. Ph.D. Paulo Henrique Sales Guimarães	UFLA

Prof. Dr. Mayron César de Oliveira Moreira
Orientador

LAVRAS – MG
2023

Dedicado a minha irmã, que me mostrou que o futuro pode valer a pena.

AGRADECIMENTOS

Agradeço aos meus pais por não desistirem de mim quando eu já havia desistido.
Agradeço ao meu professor e orientador Mayron pela paciência e por ter me ensinado tanto.
Agradeço aos meus amigos da Tribo do Chapéu Elegante Douglas, Guilherme, João Paulo, Júlio, Marlene, Matheus e Ricardo, aos amigos do Discord Gabo e Midas, e aos que conheci na faculdade, Ayden, Álvaro, Brendo, Gabriel, Igor, Marcos, Padu e Zophie.

*The problem is not per se teen pregnancy, it's **unwed** pregnancy.*
(Matt Walsh)
I violate my kids' consent all the time, in the sense that I force them to do things they don't want to do.
(Matt Walsh)
Matt Walsh é um propagandista do genocídio transgênero, famoso por acusar a comunidade LGBTQIA+ de pedofilia.

RESUMO

A avaliação objetiva de heurísticas é um problema há décadas na computação, pela dificuldade de escolher um conjunto de instâncias de teste que explore todas as fraquezas e virtudes de um algoritmo livre de viés, e também a dificuldade de determinar o que faz uma instância ser fácil ou difícil. Afim de avançar o entendimento sobre o comportamento desses programas, foi desenvolvido o método de Análise de Espaço de Instância, que seleciona as características de maior impacto na performance de algoritmos para resolver as instâncias testadas, e permite uma visualização clara de um plano em que projeta estas instâncias. Esse método também oferece um sistema que prevê o melhor algoritmo para resolver uma dada instância baseado na posição projetada a partir de suas características. Este trabalho explica o funcionamento desse método, e o aplica utilizando *software* MATILDA, desenvolvido para facilitar o acesso ao método. É demonstrada a aplicação da MATILDA, tanto para analisar as instâncias existentes quanto para gerar novas instâncias, ao problema da mochila, um problema clássico de otimização na literatura de computação.

Palavras-chave: Otimização. Espaço de instância. Heurística. Problema da mochila. Pesquisa operacional.

ABSTRACT

The objective avaliation of heuristics has been a problem for decades in the computer science field, due to the difficulty both of choosing a test instance set that explores all weakness and strengths of an algorithm without bias, and of what makes an instance be easy or hard to solve. The Instance Space Analysis method was developed with the goal of advancing the understanding of these programs' behaviours, a method that selects the features of the tested instance that had the most impact over given algorithms' performances and allows clear visualization of a 2d plane where all these instances are projected. This method also offers a program that antecipates the best algorithm to solve a given instance based on it's projected position from it's features. This work explains this method's inner working, and applies it using the MATILDA software, developed with the goal to facilitate access to the method. MATILDA's application is demonstrated both for analysing existing instances of and for guiding the generation of new instances of the knapsack problem, a classic optimization problem in computer science literature.

Keywords: Optimization. Instance space. Heuristics. Knapsack problem. Operation research.

SUMÁRIO

1 INTRODUÇÃO

A seleção de testes ao estudar o desempenho de um algoritmo é um passo importante para demonstrar seus pontos fracos e fortes. Esses dados empíricos têm sua relevância amplificada quando o problema estudado possui diversas características que impactam na capacidade de resolução de problemas por algoritmos. Em seu artigo, Hooker (1995) argumenta que o campo que mais carece disso é o de estudo de heurísticas, que são métodos de resolver um problema que buscam retornar uma resposta factível mas não necessariamente a ótima. Ele responsabiliza a dificuldade de utilizar métodos puramente analíticos combinado com uma mentalidade competitiva da comunidade acadêmica por fomentar um estado improdutivo e anti-intelectual de busca pelo "melhor algoritmo", que apresenta algoritmos como sucessos ou falhas, mas sem exploração científica do que leva ao sucesso ou a falha. Como provam Wolpert e Macready (1997), o próprio conceito de um "melhor algoritmo" universal é falho, e deve haver cautela ao selecionar os testes para que a nuance entre algoritmos comparados seja apropriadamente representada.

Nesse contexto, Smith-Miles (2009) faz um estudo da literatura referente ao problema de Rice (1976) de escolher o melhor algoritmo pra resolver a instância de um problema. Uma instância é uma configuração específica de um problema, um conjunto de valores para os parâmetros de um problema. Esse estudo é usado de base para discernir instâncias de difícil solução por Smith-Miles e Lopes (2012), que apresenta pela primeira vez o que veio a ser chamado de *Instance-Space Analysis* (***ISA***, em português, Análise de Espaço de Instância). Trata-se de um método para visualizar instâncias plotadas como pontos em um plano baseado em características (ou *features*) das instâncias, de modo que instâncias semelhantes fiquem próximas.

Smith-Miles et al. (2014) explicam como prever performance de um algoritmo dado apenas suas *features*, demonstram sua acurácia, e como os pontos fortes e fracos de um algoritmo podem ser medidos mais objetivamente. Smith-Miles,

Christiansen e Muñoz (2021) desenvolve esse trabalho demonstrando como determinar áreas nesse plano que algoritmos provavelmente irão resolver as instâncias nessas áreas da melhor maneira. Em "*Generating new test instances by evolving in instance space*", Smith-Miles e Bowly (2015) aplicam a *ISA* concorrentemente com um gerador de instâncias guiado pela *ISA*, a fim de melhor preencher áreas relevantes com menos densidade de instância na projeção. Esta monografia busca expor, embasar e justificar a utilidade do método.

1.1 Objetivos

Esse trabalho apresenta o funcionamento interno do *software* MATILDA (do inglês: *Melbourne Algorithm Test Instance Library with Data Analytics*), que é a implementação pública do método *ISA*. Afim de demonstrar a utilidade do método, é dado como exemplo resultados do estudo "*Revisiting where are the hard knapsack problems? via Instance Space Analysis*" (SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021), que gera instâncias do problema da mochila binária. Seu código fonte é público e há interface online disponibilizado pela Universidade de Melbourne.

1.2 Estruturação

Após esse capítulo de introdução, o Capítulo 2 explica o problema da mochila binária (BOYD; VANDENBERGHE, 2004). O Capítulo 3 explica a metodologia utilizada pelo *software* MATILDA. Resultados considerando o problema da mochila binária são reportados no Capítulo 4, enquanto o Capítulo 5 realiza a conclusão do trabalho.

2 PROBLEMA DA MOCHILA

2.1 Motivação

O problema da mochila binário foi escolhido por ser um problema de otimização clássico da literatura. A Universidade de Melbourne disponibiliza instâncias para demonstrar a ferramenta MATILDA. Existem aplicações relevantes atuais em diversos campos, como na geração de chaves criptográficas e na redução de desperdício no corte de matéria prima (KELLERER; PFERSCHY; PISINGER, 2004).

Apesar do objetivo do problema ser de fácil entendimento, sua solução é longe de ser trivial. Não é conhecido método que resolva-o em tempo polinomial, embora existam aproximações pseudo-polinomiais segundo Book (1975). Este problema é classificado como NP-completo, dado que pode ser resolvido com um algoritmo com *backtracking* de profundidade polinomial em relação à entrada (BOOK, 1975), e que pode ser reduzido a outro problema da mesma classe de complexidade, como Garey e Johnson (1979) demonstram ao reduzir para o problema de partições. Portanto, cada solução pode ter sua viabilidade verificada em tempo polinomial.

2.2 Definição

O objetivo consiste em escolher itens a serem inseridos na mochila de modo que o valor total dos itens seja o maior possível, e que não ultrapassem a capacidade de carga da mochila. Cada item tem seu valor e peso associado, ambos não negativos. Os itens não podem ser partidos, ou seja, não é admissível adicionar apenas frações de um item na mochila. Apresenta-se, a seguir, os conjuntos de parâmetros e variáveis, além do modelo matemático que formaliza o problema.

Parâmetros:

- C é a capacidade de carga da mochila;

- m é a quantidade de itens disponíveis;
- P_m é o conjunto de pesos $\{p_1,\ p_2,\ p_3,\ \cdots,\ p_m\}$ tal que o elemento p_i se refere ao peso do item i;
- V_m é o conjunto de valores $\{v_1,\ v_2,\ v_3,\ \cdots,\ v_m\}$ tal que o elemento v_i se refere ao valor do item i.

Variáveis:

- X_m é o conjunto de variáveis binárias $\{x_1, x_2, x_3, \cdots, x_m\}$ tal que o elemento $x_i = 1$ caso o item i tenha sido colocado na mochila, e $x_i = 0$ caso contrário.

Modelo:

$$\text{maximize} \sum_{i=1}^{m} v_i x_i \tag{2.1}$$

sujeito a

$$\sum_{i=1}^{m} p_i x_i \leq C \tag{2.2}$$

$$x_i \in \{0,\ 1\} \qquad i \in \{1, ..., m\}. \tag{2.3}$$

2.3 Aplicação

Uma aplicação prática do problema da mochila binária consiste na maximização de lucro de algum comerciante em viagem que decide comprar mercadoria para revenda no seu retorno. Nesta situação hipotética, o vendedor decide por importar *smartphones* para seu país de origem dado seu alto valor e baixas dimensões, assim será desconsiderado o problema de organizar os produtos dentro da bagagem.

A única restrição imposta é pela linha aérea, que aplica uma multa caso sua bagagem ultrapasse o limite de peso de C. Pode-se extrair uma lista dos 10

dispositivos mais populares da loja de origem, portanto m é a soma dos estoques destes produtos. Cada item i tem seu peso p_i dado pela massa total do respectivo dispositivo, caixa, manuais e acessórios inclusos, e o valor v_i é o preço do produto no país de origem do comerciante, subtraindo o investimento do preço original no exterior.

Outros contextos os quais pode-se ver a presença do problema da mochila binária são o empacotamento de objetos em contêineres (OLIVEIRA; GAMBOA; SILVA, 2023), planejamento de transporte de produtos em caminhões (PEREIRA; MATEUS; URRUTIA, 2022) e alocação de funções virtuais em redes que utilizam tecnologia 4G, 5G e 6G (ARAUJO; SOUZA; MATEUS, 2023).

3 METODOLOGIA

3.1 Motivação

O Capítulo 1 apresenta o conceito de instância, que consiste em um conjunto de parâmetros de entrada necessários para a definição de variáveis utilizadas no processo de tomada de decisão. Quando se plota pontos representando instâncias em um espaço cartesiano, baseado-se em vetores de características derivadas destes parâmetros, este espaço é chamado de espaço de instâncias Smith-Miles e Tan (2012).

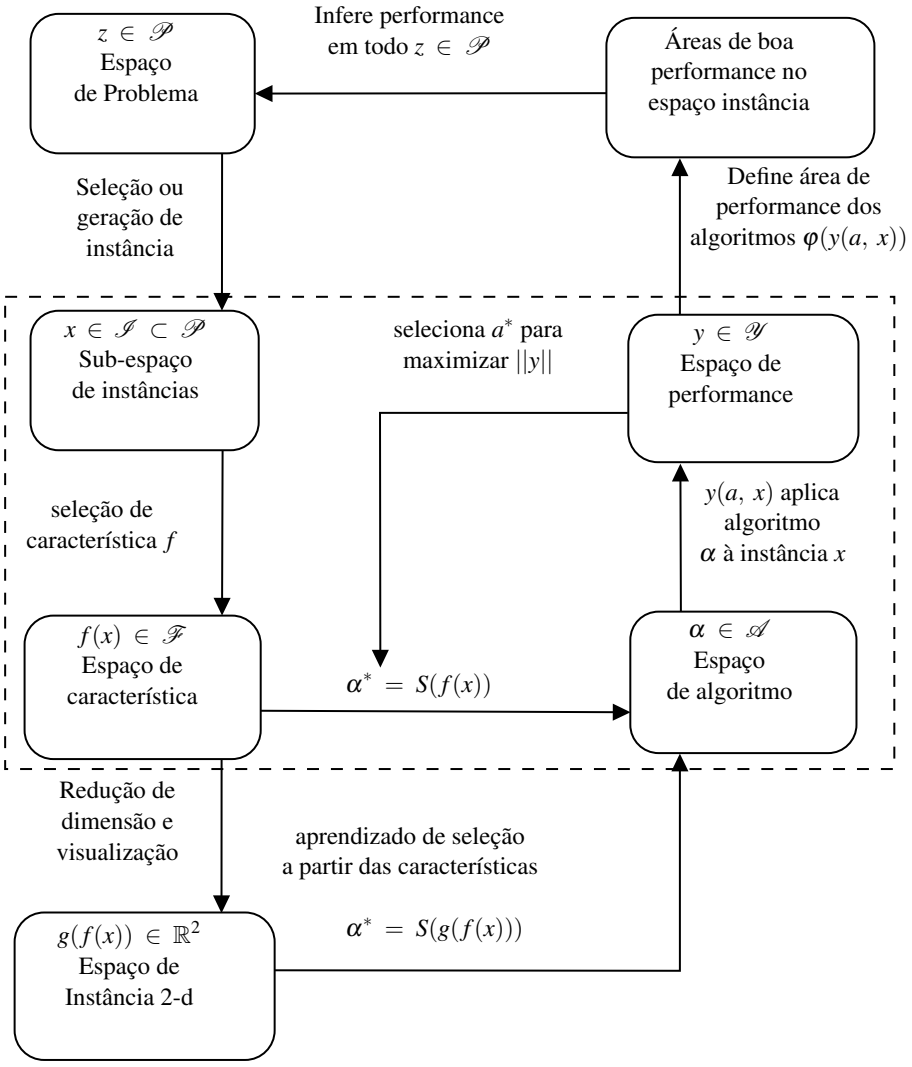
Não é produtivo gerar instâncias aleatórias indiscriminadamente dado que "[...] problemas aleatórios geralmente não se assemelham a problemas reais" (HOOKER, 1995, tradução autoral). Com uma boa representação do espaço de instância é mais fácil analisar as instâncias e o desempenho de algoritmos objetivamente (SMITH-MILES et al., 2014). Como demonstram Smith-Miles e Bowly (2015), essa representação permite até guiar a geração de instâncias com maior diversidade, que por sua vez expande a análise de pontos fortes e fracos de algoritmos, de forma mais consistentemente.

3.2 Estrutura

O Diagrama 3.1 mostra a estrutura da *ISA*, que evoluiu da seleção de algoritmo proposta por Rice (1976), mostrada entre linhas pontilhadas. Este diagrama é constituído de 6 espaços e conjuntos, porém Smith-Miles e Bowly (2015) os chamam apenas de espaços. O primeiro é $\mathscr{P}$, o *espaço de problema*, que contém todas as instâncias possíveis de um problema, incluindo o *subconjunto de instâncias* $\mathscr{I}$, que se tem os resultados computacionais e características chamados metadados. Este é o subconjunto de instâncias que foram testadas pelo usuário, e seus metadados que são os dados de entrada para MATILDA. Desses metadados, as características são medíveis no *espaço de características* $\mathscr{F}$, e as performances dos

algoritmos usados para teste estão no *espaço de performance* $\mathscr{Y}$. As performances em $\mathscr{Y}$ estão mapeadas para cada instância em $\mathscr{I}$ com um respectivo algoritmo. O conjunto de todos os algoritmos testados é chamado de *espaço de algoritmo* $\mathscr{A}$. Como pode se notar, é uma estrutura iterativa, robusta o suficiente para evoluir com geração de novas instâncias.

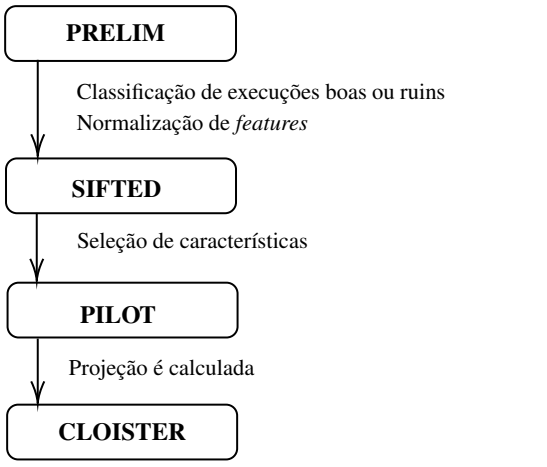
Figura 3.1 – Estrutura de Análise de Espaço de Instância, com Estrutura de Seleção de Algoritmo de Rice (1976) original no quadro pontilhado
Fonte: (SMITH-MILES; BOWLY, 2015, tradução nossa)



3.3 Funcionamento

A construção do espaço instâncias é dividida em 4 métodos principais: *Preparation for Learning of Instance Meta-Data*[1] (***PRELIM***), *Selection of Instance Features to Explain Difficulty*[2] (***SIFTED***), *Projecting Instances with Linearly Observable Trends*[3] (***PILOT***) e *Correlated Limits of the Instance Space's Theoretical or Experimental Regions*[4] (***CLOISTER***) (SMITH-MILES; MUÑOZ, MARIO ANDRÉS, 2023).

Figura 3.2 – Ordem de execução dos algoritmos



No início do *PRELIM*, todas performances das instâncias são classificadas como "boas" ou "ruins" para seus respectivos algoritmos, usando uma porcentagem tolerância dada pelo usuário a ser comparada com o melhor algoritmo para a instância. Por exemplo, Smith-Miles, Christiansen e Muñoz (2021) utiliza como métrica de performance o tempo de execução do algoritmo para cada instância, e configura a tolerância para 20%. Isto faz com que, para cada instância, se o tempo de execução de um dos algoritmos leva mais que 20% a mais que o melhor tempo

[1] Preparação para aprendizado de metadados de instância
[2] Seleção de características de instância para explicar dificuldade
[3] Projeção de instâncias com tendências linearmente observáveis
[4] Limites correlacionados das regiões teórica ou experimental do espaço de instância

para aquela instância, aquele algoritmo é classificado como “ruim” para aquela instância, ou “bom” caso contrário.

Para reduzir o efeito de *outliers*, os valores das características são limitados na faixa de diferença de até 5 intervalos interquartil da mediana. Um intervalo interquartil é definido ordenando os valores dos dados, encontrando os pontos superiores e inferiores em que cada engloba 25% dos dados entre eles e a mediana, e calculando a diferença entre estes pontos. Todos os valores de uma característica fora desta faixa de 10 intervalos tem seus valores truncados para o extremo mais próximo da faixa.

Afim de estabilizar a variância e normalizar os dados, uma transformação *Box-Cox* de um parâmetro é aplicada a cada característica e métrica de performance, deixando-as mais semelhante a uma distribuição normal, para evitar problemas derivados de assimetria. Para detalhes da transformação, sugere-se a leitura de *An Analysis of Transformations*, proposta por Box e Cox (1964). Para garantir média 0 e desvio padrão de 1, também é aplicada a transformada Z, concluindo o *PRELIM*.

Jong (1980) define algoritmo genético como um programa que dado possíveis soluções chamadas “agentes”, através de uma seleção dessas que tiveram o melhor resultado, mistura as estratégias utilizadas para gerar novos agentes, repetindo até atingir um objetivo. O *software* MATILDA utiliza de um algoritmo genético durante o *SIFTED* para selecionar o subconjunto de características menos relacionadas entre si, e mais relacionadas à performance. Cada agente desse algoritmo é um subconjunto de características, construído com uma base estática das características com maior correlação com cada algoritmo em teste, e características que possuem correlação no mínimo moderada com pelo menos um algoritmo (SMITH-MILES; MUÑOZ, MARIO ANDRÉS, 2023).

O algoritmo de clusterização *k-means* é utilizado para separar as características de acordo com suas dissimilaridades, e uma característica aleatória é esco-

lhida dentro de cada *cluster* para ser parte do subconjunto, a variação entre agentes surge dessa escolha. As instâncias nesse espaço hiper-dimensional com as características de um subconjunto como vetores ortogonais são projetadas para um plano cartesiano através do processo de *Principal Component Analysis*[5] (***PCA***), um método que visa simultaneamente extrair as informações importantes dos dados e simplificar sua apresentação (ABDI; WILLIAMS, 2010).

A avaliação dos agentes é feita baseada em um modelo de Floresta Aleatória, um algoritmo de aprendizado de máquina para classificação, treinada com esse dados de entrada para decidir se uma instância é boa ou não com um algoritmo. Quanto menor o erro *out-of-bag*, uma medida da Floresta Aleatória baseada em testar uma fração do *input* comparando com as definições de algoritmos "bons" feitos no *PRELIM*, melhor será classificado o agente (SMITH-MILES; MUÑOZ, MARIO ANDRÉS, 2023).

O *PILOT* utiliza um algoritmo de otimização chamado Broyden-Fletcher-Goldfatb-Shanno (BFGS), cujos detalhes podem ser encontrados em *The Convergence of a Class of Double-rank Minimization Algorithms* (BROYDEN, 1970), de forma que os valores das características e das performances aumentam linearmente de um lado ao outro do espaço de instância (SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021). Como potencialmente existem infinitas soluções ótimas e o BFGS não é determinístico, o algoritmo é executado diversas vezes, e o melhor resultado é determinado pelo que tiver melhor preservação topológica, ou seja, o que tiver melhor correlação entre as distâncias no espaço hiper-dimensional e no plano. Para mais detalhes desse método sugere-se o estudo do artigo (SMITH-MILES; MUÑOZ, MARIO ANDRÉS, 2023).

A construção do espaço-instância é concluída com *CLOISTER*, quando delimita o espaço onde instâncias podem existir utilizando os limites superiores e inferiores das características, removendo combinações improváveis por correlação

[5] Análise de Componente Principal

forte entre características. Estes limites podem ser extraídos teoricamente, ou dos metadados. Por fim, esses limites são projetados no plano cartesiano.

Para cada algoritmo é treinado uma *Support Vector Machine*[6] (***SVM***), que é um classificador que denota uma área onde se espera que boa performance do algoritmo (SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021). Como estas áreas são calculadas por algoritmo, é possível ocorrer interseção. Nesses casos, o algoritmo com maior precisão é selecionado. Se alguma área não houver um algoritmo predito que tenha um bom desempenho, o algoritmo com melhor performance média é selecionado.

[6] Máquina de Vetor de Suporte

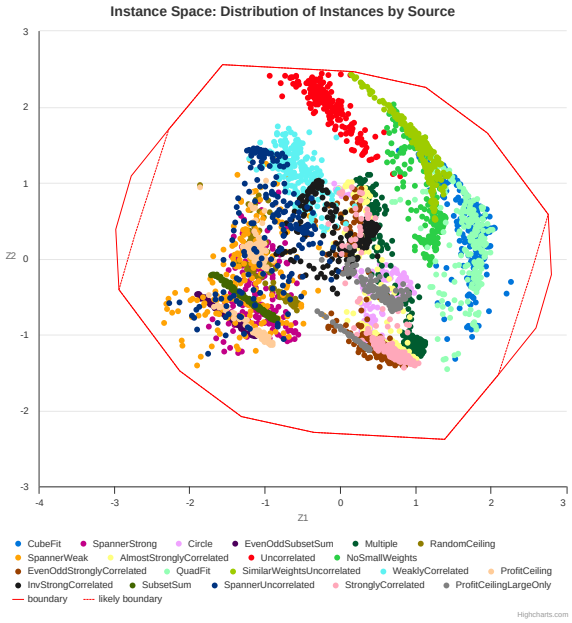
4 RESULTADOS

Como exemplo de utilização, serão apresentados resultados obtidos por Smith-Miles, Christiansen e Muñoz (2021) para o problema da mochila binária. Para detalhes sobre as instâncias, suas características e os algoritmos usados de entrada, sugere-se a consulta ao material original "Revisiting *where are the hard knapsack problems?* via Instance Space Analysis" (SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021). Os gráficos são extraídos pela interface gráfica disponibilizada por Smith-Miles e Muñoz (2019).

Foram usados 20 classes instâncias como fontes de bases de dados. A maioria dessas classes são derivadas dos trabalhos de Martello, Pisinger e Toth (1999) e Pisinger (2005). Para cada classe foram geradas 200 instâncias, com 1.000 itens cada. Para cada instância foram calculadas 23 características, detalhadas no Apêndice A. Os algoritmos testados foram *EXPKNAP* (PISINGER, 1995), *MINKNAP* (PISINGER, 1997) e *COMBO* (MARTELLO; PISINGER; TOTH, 1999). Todos encontram solução ótima em todas as instâncias, portanto a métrica de performance escolhida foi o tempo de execução.

A Figura 4.1 evidencia que cada conjunto de instância falta diversidade, e que a utilização de apenas uma dessas fontes para análise de uma heurística é insuficiente, dado que diferentes fontes cobrem distintas porções do espaço de instância. Usando as instâncias, as performances e a projeção da Figura 4.1 como base, foram designados pontos alvo no espaço-instância, e executados algoritmos genéticos desenvolvidos neste estudo, e outros 4 guiados pelos parâmetros disponíveis no Apêndice B a fim de criar instâncias próximas aos pontos alvo. Consulte o material de Smith-Miles, Christiansen e Muñoz (2021) para informações sobre a implementação.

Figura 4.1 – Espaço de Instância: Distribuição de Instâncias por suas Fontes. Fonte: Smith-Miles e Muñoz (2019)



Recalculando a projeção com as informações das novas instâncias geradas, tem-se as instâncias originais na nova projeção na Figura 4.2 e as novas instâncias na Figura 4.3.

Figura 4.2 – Distribuição de Instâncias Originais por suas Fontes, em projeção com Instâncias Novas. Fonte: Smith-Miles e Muñoz (2019)

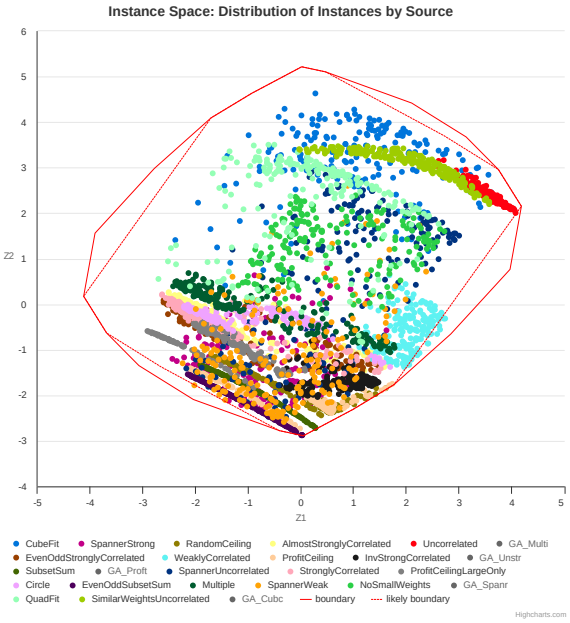
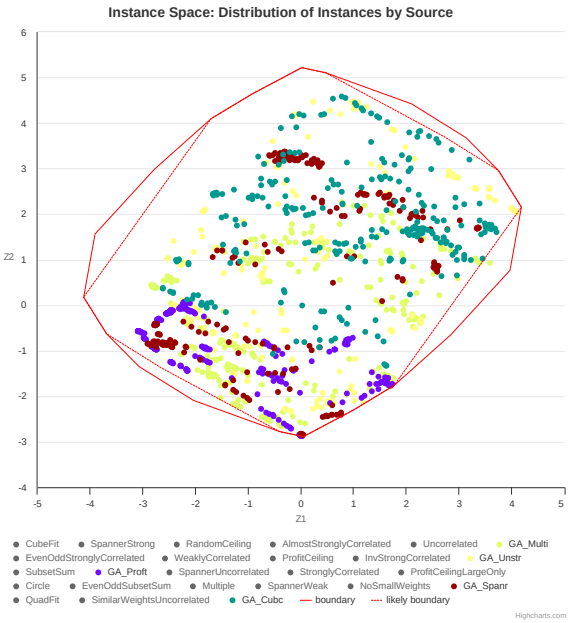


Figura 4.3 – Distribuição de Instâncias Novas por suas Fontes. Fonte: Smith-Miles e Muñoz (2019)



O programa também permite visualização da área que cada algoritmo testado melhor performa, visualização dos vetores de características, e das instâncias fáceis e difíceis. Estão representadas, respectivamente, nas Figuras 4.4, 4.5 e 4.6. São disponibilizados gráficos de todas as características selecionadas na fase de *SIFTED*, porém para fins demonstrativos basta analisar característica *GreedyUnusedCapacity*, uma métrica referente ao espaço inutilizado pela solução de um algoritmo guloso, detalhada no Item 23 do Apêndice A , junta das demais visualizações de dificuldade e algoritmo selecionado. Pode-se induzir que embora esteja fracamente relacionada à dificuldade de uma instância, tem grande relevância no algoritmo de melhor performance. Além disso, percebe-se a ausência do algoritmo *EXPKNAP* na Figura 4.4, o que vai de acordo com Pisinger e Saidi (2017), que afirmam que o *MINKNAP* e o *COMBO* são algoritmos estado-da-arte.

Figura 4.4 – Instâncias por algoritmo que melhor as resolve. Fonte: Smith-Miles e Muñoz (2019)

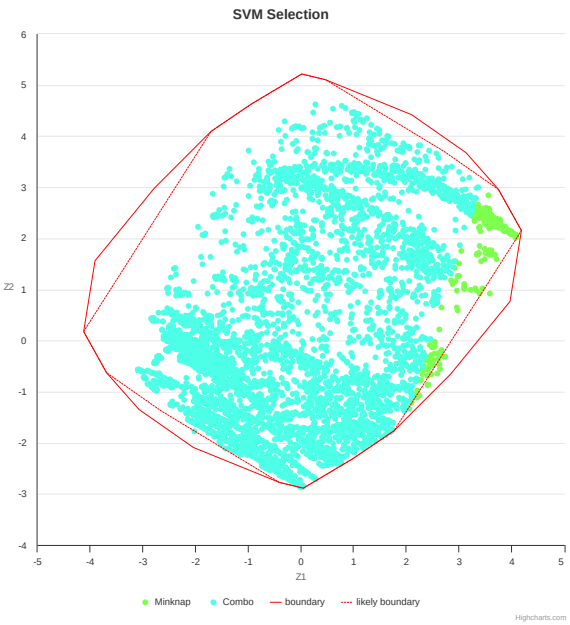


Figura 4.5 – Instâncias marcadas por valor da característica "*GreedyUnusedCapacity*". Fonte: Smith-Miles e Muñoz (2019)

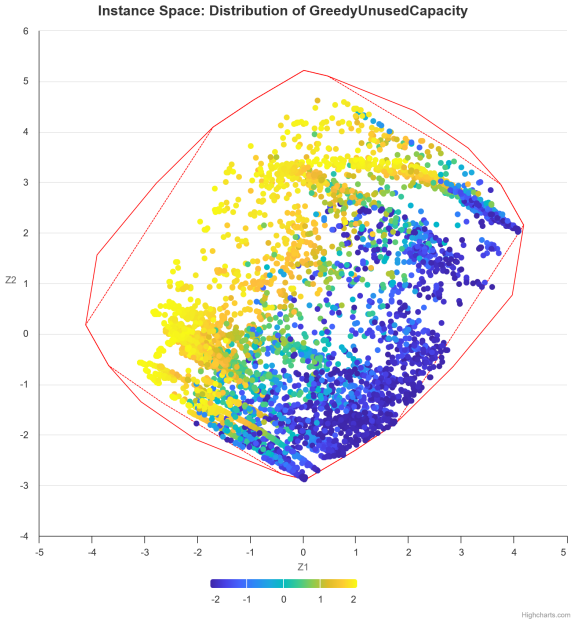
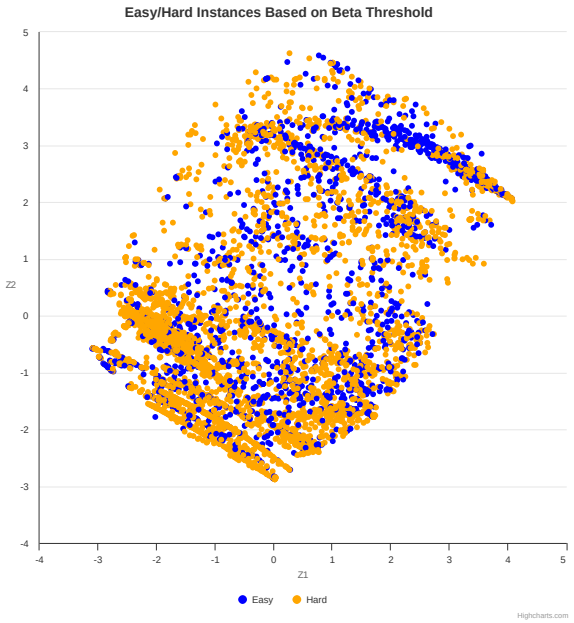


Figura 4.6 – Instâncias fáceis e difíceis. Fonte: Smith-Miles e Muñoz (2019)



Essas figuras são apenas uma fração do que é disponibilizado pela MATILDA, apresentadas com a finalidade de ilustrar que tipo de análises podem ser feitas a partir da projeção. A leitura de Smith-Miles, Christiansen e Muñoz (2021) é fortemente recomendada para exploração extensiva das informações apresentadas na projeção específica dessas instâncias.

5 CONCLUSÃO

As análises permitidas pela MATILDA abrem possibilidades promissoras para o campo de desenvolvimento de heurísticas. Agora é possível mais facilmente entender os pontos fortes e fracos de uma heurística, e decidir quais usar automaticamente através de características das instâncias. O modelo de *SVM*, que prediz o melhor algoritmo, também pode ser aplicado fora do contexto acadêmico para reduzir custo computacional de problemas grandes, uma vez que as características muitas vezes são mais fáceis de computar.

Todavia, deve-se tomar cuidado na etapa de seleção de instâncias, métrica de performance e características a serem analisadas. Se forem escolhidas de forma não crítica, sem variedade adequada, a análise do espaço de instância não será de grande ajuda. Por exemplo, como mencionado no Capítulo 4, Smith-Miles, Christiansen e Muñoz (2021) usam apenas instâncias com 1.000 itens, pois tiveram essa cautela. A complexidade de tempo de um algoritmo já é medida em relação ao tamanho do *input*, logo se estivesse presentes nos metadados instâncias com diferentes quantidades de itens e características relacionadas a essas quantidades, utilizar tempo de execução como métrica de performance iria poluir os resultados com uma informação já conhecida: a dificuldade de uma instância tende a ser proporcional ao tamanho da entrada.

A interface *online* é de fácil uso, o que permite maior acessibilidade. Esta facilidade de acesso é crucial para popularizar mais estudos utilizando esta técnica, e já permitiu trabalhos utilizando o *software*.

REFERÊNCIAS

ABDI, H.; WILLIAMS, L. J. Principal component analysis. **WIREs Computational Statistics**, v. 2, n. 4, p. 433–459, 2010. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.101>.

ARAUJO, S.; SOUZA, F.; MATEUS, G. Modeling and analysis of different reconfiguration strategies for virtual network function placement and chaining with service classes identification. **IEEE Latin America Transactions**, v. 21, n. 3, p. 475–482, 2023.

BALAS, E.; ZEMEL, E. An Algorithm for Large Zero-One Knapsack Problems. **Operations Research**, v. 28, n. 5, p. 1130–1154, out. 1980. ISSN 0030-364X, 1526-5463. Disponível em: <https://pubsonline.informs.org/doi/10.1287/opre.28.5.1130>.

BOOK, R. V. Richard m. karp. reducibility among combinatorial problems. complexity of computer computations, proceedings of a symposium on the complexity of computer computations, held march 20-22, 1972, at the ibm thomas j. watson center, yorktown heights, new york, edited by raymond e. miller and james w. thatcher, plenum press, new york and london 1972, pp. 85–103. **The Journal of symbolic logic**, v. 40, n. 4, p. 618–619, 1975. ISSN 0022-4812.

BOX, G. E. P.; COX, D. R. An analysis of transformations. **Journal of the Royal Statistical Society. Series B (Methodological)**, [Royal Statistical Society, Wiley], v. 26, n. 2, p. 211–252, 1964. ISSN 00359246. Disponível em: <http://www.jstor.org/stable/2984418>.

BOYD, S.; VANDENBERGHE, L. **Convex Optimization**. Cambridge, England: Cambridge University Press, 2004.

BROYDEN, C. G. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. **IMA Journal of Applied Mathematics**, v. 6, n. 1, p. 76–90, 03 1970. ISSN 0272-4960. Disponível em: <https://doi.org/10.1093/imamat/6.1.76>.

CHUNG, C.-S.; HUNG, M. S.; ROM, W. O. A hard knapsack problem. **Naval Research Logistics**, v. 35, n. 1, p. 85–98, fev. 1988. ISSN 0894069X, 15206750. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/1520-6750(198802)35:1<85::AID-NAV3220350108>3.0.CO;2-D>.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability: a guide to the theory of NP-completeness**. San Francisco: W. H. Freeman, 1979. (A Series of books in the mathematical sciences). ISBN 9780716710448.

HALL, N. G.; POSNER, M. E. Performance Prediction and Preselection for Optimization and Heuristic Solution Procedures. **Operations Research**, v. 55,

n. 4, p. 703–716, ago. 2007. ISSN 0030-364X, 1526-5463. Disponível em: <https://pubsonline.informs.org/doi/10.1287/opre.1070.0398>.

HOOKER, J. N. Testing heuristics: We have it all wrong. **Journal of heuristics**, Springer, v. 1, n. 1, p. 33–42, 1995. ISSN 1381-1231.

JONG, K. D. Adaptive system design: A genetic approach. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 10, n. 9, p. 566–574, Sep. 1980. ISSN 2168-2909.

KELLERER, H.; PFERSCHY, U.; PISINGER, D. **Knapsack Problems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. ISBN 9783642073113.

MARTELLO, S.; PISINGER, D.; TOTH, P. Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem. **Management Science**, v. 45, n. 3, p. 414–424, mar. 1999. ISSN 0025-1909, 1526-5501. Disponível em: <https://pubsonline.informs.org/doi/10.1287/mnsc.45.3.414>.

MARTELLO, S.; TOTH, P. Upper Bounds and Algorithms for Hard 0-1 Knapsack Problems. **Operations Research**, v. 45, n. 5, p. 768–778, out. 1997. ISSN 0030-364X, 1526-5463. Disponível em: <https://pubsonline.informs.org/doi/10.1287/opre.45.5.768>.

OLIVEIRA, O.; GAMBOA, D.; SILVA, E. An introduction to the two-dimensional rectangular cutting and packing problem. **International Transactions in Operational Research**, v. 30, n. 6, p. 3238–3266, 2023.

PEREIRA, A. H.; MATEUS, G. R.; URRUTIA, S. A. Valid inequalities and branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. **European Journal of Operational Research**, v. 300, n. 1, p. 207–220, 2022.

PISINGER, D. An expanding-core algorithm for the exact 0–1 knapsack problem. **European Journal of Operational Research**, v. 87, n. 1, p. 175–187, nov. 1995. ISSN 03772217. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/0377221794000133>.

PISINGER, D. A Minimal Algorithm for the 0-1 Knapsack Problem. **Operations Research**, v. 45, n. 5, p. 758–767, out. 1997. ISSN 0030-364X, 1526-5463. Disponível em: <https://pubsonline.informs.org/doi/10.1287/opre.45.5.758>.

PISINGER, D. Where are the hard knapsack problems? **Computers & Operations Research**, v. 32, n. 9, p. 2271–2284, set. 2005. ISSN 03050548. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S030505480400036X>.

PISINGER, D.; SAIDI, A. Tolerance analysis for 0–1 knapsack problems. **European journal of operational research**, Elsevier B.V, v. 258, n. 3, p. 866–876, 2017. ISSN 0377-2217.

RICE, J. R. The algorithm selection problem. In: RUBINOFF, M.; YOVITS, M. C. (Ed.). Elsevier, 1976, (Advances in Computers, v. 15). p. 65–118. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0065245808605203>.

SMITH-MILES, K. et al. Towards objective measures of algorithm performance across instance space. **Computers & Operations Research**, Elsevier Ltd, OXFORD, v. 45, p. 12–24, 2014. ISSN 0305-0548.

SMITH-MILES, K.; BOWLY, S. Generating new test instances by evolving in instance space. **Computers & Operations Research**, Elsevier BV, v. 63, p. 102–113, nov. 2015. Disponível em: <https://doi.org/10.1016/j.cor.2015.04.022>.

SMITH-MILES, K.; CHRISTIANSEN, J.; MUÑOZ, M. A. Revisiting *where are the hard knapsack problems?* via instance space analysis. **Computers & Operations Research**, Elsevier Ltd, v. 128, p. 105184, 2021. ISSN 0305-0548.

SMITH-MILES, K.; LOPES, L. Measuring instance difficulty for combinatorial optimization problems. **Computers & Operations Research**, Elsevier Ltd, OXFORD, v. 39, n. 5, p. 875–889, 2012. ISSN 0305-0548.

SMITH-MILES, K.; MUÑOZ, M. **Matilda - University of Melbourne**. 2019. Disponível em: <https://matilda.unimelb.edu.au/matilda/>.

SMITH-MILES, K.; MUÑOZ, MARIO ANDRÉS. Instance space analysis for algorithm testing: Methodology and software tools. **ACM computing surveys**, v. 55, n. 12, p. 1–31, 2023. ISSN 0360-0300.

SMITH-MILES, K.; TAN, T. T. Measuring algorithm footprints in instance space. In: **2012 IEEE Congress on Evolutionary Computation**. [S.l.]: IEEE, 2012. p. 1–8. ISBN 1467315109. ISSN 1089-778X.

SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. **ACM computing surveys**, Association for Computing Machinery, Inc, v. 41, n. 1, p. 6, 2009. ISSN 0360-0300.

WOLPERT, D.; MACREADY, W. No free lunch theorems for optimization. **IEEE transactions on evolutionary computation**, IEEE, v. 1, n. 1, p. 67–82, 1997. ISSN 1089-778X.

APÊNDICE A – Características

Tabela 1 – Tabela de Características

Feature Name	Bound	Description
1. Dominant Pairs	$[0,1]$	Proportion of item pairs i,j for which $p_i \geqslant p_j$ and $w_i \leqslant w_j$ OR $p_j \geqslant p_i$ and $w_j \leqslant w_i$, excluding identical items. Similar to Hall and Posner (2007).
2. Smaller Better Pairs	$[0,1]$	Proportion of item pairs i,j for which $p_i/w_i > p_j/w_j$ and $w_i \leqslant w_j$ OR $p_j/w_j > p_i/w_i$ and $w_j \leqslant w_i$.
3. Capacity Fraction	$[0,1]$	Capacity of the knapsack divided by the sum of the weights of all items.
4. Capacity Fraction Distance From $1/2$	$[0,1]$	$\lvert\text{Capacity Fraction} - 0.5\rvert \times 2$
5.Correlation Coefficient	$[-1,1]$	Correlation coefficient of item weights and profits, viewed as points (w_i,p_i) in $\mathbb{R}^2$
6. Approximation Gap	$[0,1]$	Difference between the linear relaxation upper bound and greedy algorithm lower bound (similar to Hall and Posner (2007)) divided by the profit assigned to the most profitable item.
7.Coefficient of Variation of Weights	$[0,1]$	Standard deviation of item weights divided by average item weight, as in Hall and Posner (2007). Normalised as per (1) with $D=1$.
8. Coefficient of Variation of Profits	$[0,1]$	Standard deviation of item profits divided by average item profit, as in Hall and Posner (2007). Normalised as per (1) with $D=1$.
9. Coefficient of Variation of Efficiencies	$[0,1]$	Standard deviation of item efficiencies divided by average item efficiency. Normalised as per (1) with $D=5$.
10. Possible Item Fix Proportion	$[0,1]$	Proportion of items which can be determined to be included or excluded in the optimal packing of the knapsack by fixing them as in or out of the knapsack and comparing the value of the linear relaxation solution of the resulting problem with the value of the greedy solution of the original problem.
11. Subset Sum Likeness	$[0,1]$	Proportion of items with efficiency within 0.05% as the split item. Intended to measure the similarity of the instance to a Subset Sum instance.
12. Even–Odd Likeness	$[0,10]$	Equal to Subset Sum Likeness, but multiply by 10 if the weights of the items that have the split item's efficiency have a greatest common divisor greater than 1. Intended to measure the similarity of the instance to an Even–Odd Subset Sum instance, or a Profit Ceiling instance with sufficiently large capacity.
13. Modified Balas-Zemel Measure	$[0,1]$	Modified version of the measure defined in Balas and Zemel (1980) using the value of the greedy solution in the place of the value of the integer optimum solution: $\frac{\text{Linear Relaxation value}-\text{Greedy Solution value}}{(1/2)max_{i\in\{1,\dots,N\}}\lvert p_i-(w_i\times\text{Efficiency of Split Item})\rvert}$Normalised as per (1) with $D=500$.
14. Maximum Cardinality	$[0,1]$	Upper bound, defined in Martello and Toth (1997) Eq. (11), on the number of items which may potentially be included in the optimal solution, divided by n.
15. Minimum Cardinality	$[0,1]$	Lower bound, defined in Martello and Toth (1997), on the number of items which may potentially be included in the optimal solution found by trying to add items in descending order of weight, divided by n.
16. Cardinality Gap	$[0,1]$	Difference between Maximum Cardinality and Minimum Cardinality.
17. First Weight	$[0,1]$	Weight of the least heavy item divided by the weight of the heaviest item. Similar feature used in Chung et al. (1988).
18. First Profit	$[0,1]$	Profit of the least profitable item divided by the weight of the most profitable item.
19. Polyfit Linear	$[0,1]$	Consider the items as defining points (w_i,p_i) in $\mathbb{R}^2$, and find the linear equation $P(w)=aw+b$ which best fits these points in a least-squares sense. Define $\bar{p}=\frac{1}{n}\sum_{i=1}^{n}p_i$. Then the value of the feature is $\max\left\{1-\frac{\sum_{i=1}^{n}(p_i-P(w_i))^2}{\sum_{i=1}^{n}(p_i-\bar{p})^2},0\right\}$
20. Polyfit Quadratic	$[0,1]$	As Polyfit Linear, but considering a quadratic equation $P(w)=aw^2+bw+c$.
21. Polyfit Cubic	$[0,1]$	As Polyfit Linear, but considering a cubic equation $P(w)=aw^3+bw^2+cw+d$.
22. Before/After Split Ratio	$[0,1]$	Let the number of items with efficiency strictly better than that of the split item be a, and the number of items with efficiency equal to or worse than that of the split item be b. If $a>b$ then the feature is defined as $\frac{\arctan(\frac{a}{b}-1)}{\pi}+\frac{1}{2}$. Otherwise it is defined as $\frac{\arctan(1-\frac{b}{a})}{\pi}+\frac{1}{2}$
23. Greedy Unused Capacity	$[0,1]$	If the instance has no item pairs i,j such that $w_i \neq w_j$ then this feature defaults to 0. Otherwise it is defined as the capacity unused by the greedy solution, divided by the smallest non-zero difference between any two items' weights, then normalised as per (1) with $D=100$.

Fonte: Revisiting where are the hard knapsack problems? via Instance Space Analysis(SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021, pág. 6, tabela 2)

São citados na tabela Hall e Posner (2007), Balas e Zemel (1980), Martello e Toth (1997) e Chung, Hung e Rom (1988).

$$\text{normalised feature} \leftarrow \frac{\tan^{-1}(\text{raw feature value}/D)}{\pi/2} \tag{1}$$

APÊNDICE B – Algoritmos Genéticos

Tabela 2 – Tabela de Algoritmos

Generation Procedure	Parameters and Generation Procedure
Profit Ceiling-like	• knapsack capacity fraction in $[0, 1]$ • m, minimum allowed weight/ R in $[0, 1]$ • parameter to round to multiplies of in $\{2, 3, 4, 5\}$ (d in Pisinger (2005)) • binary variable defining whether to create a Profit Ceiling or Random Ceiling-type instance • f, fraction of items to have weights such that their weight is equal to their profit, in $[0, 1]$ Generate item profits as per Profit Ceiling and Random Ceiling classes, but generate weights with the following procedure. For each item, with probability f assign it a weight greater than m such that its profit will be equal to its weight; choose from any of the acceptable weights with equal probability. With probability $1 - f$, assign it a weight greater than m such that its profit will not be equal to its weight in the same way.
Cubic Fit	• knapsack capacity fraction in $[0, 1]$ • four values $a_1, \ldots, a_4$ in $[0, 1]$ For $j = 1, \ldots, 4$ set $b_j = (2.88a_j^3 - a.32b_j^2 + 2.44a_j)R$ to shift the curve in favour of profits closer to $R/2$. Find the unique cubic function $f(x)$ which passes through the points $(0, b_1)$, $(R/3, b_2)$, $(2R/3, b_3)$ and (R, b_4). Then generate item weights with a uniform random distribution between 1 and R and assign each item's profit as $p_i = \max\{1, \min\{R, \lceil f(w_i) \rceil\}\}$.
Spanner	• knapsack capacity fraction in $[0, 1]$ • maximum multiplier to be applied to the spanner items m in $\{5, 6, \ldots, 15\}$ • two pairs of parameters (a_1, d_1) and (a_2, d_2), all in $[0, 1]$, which determine the weight and profit of the spanner items For $j = 1, 2$ generate the weight and profit of the two spanner items as follows: $w_j = \left\lceil \frac{d_j}{m} R \cos(\frac{\pi}{2} a_j) \right\rceil$ $p_j = \left\lceil \frac{d_j}{m} R \sin(\frac{\pi}{2} a_j) \right\rceil$ Then generate the spanner instance as in the Spanner-type instance classes.
Multiple Strongly Correlated-like	• knapsack capacity fraction in $[0, 1]$ • Three parameters k_1 in $[0, 1]$, k_2 in $[0, 1]$ and d in $\{3, 4, \ldots, 10\}$ • f, fraction of items which will have weight divisible by d For each item, with probability f assign it a weight divisible by d (with an equal chance for any acceptable weight) and with probability $1 - f$ assign it a weight not divisible by d in the same way. Then assign the profit of each item as in Pisinger (2005): if mod $(w_i, d) = 0$ then $p_i = \lfloor \frac{R}{2} k_1 \rfloor + w_i$, and $p_i = \lfloor \frac{R}{2} k_2 \rfloor + w_i$ otherwise.

Fonte: Revisiting where are the hard knapsack problems? via Instance Space Analysis(SMITH-MILES; CHRISTIANSEN; MUÑOZ, 2021, pág. 14, tabela 3)