LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

Análise do poder do método PAR-Δ para diferentes valores de Δ

Alice Batista Nogueira
Departamento de Ciências da Computação, Universidade Federal de Alfenas
Av. Jovino Fernandes de Sales, 2600 - Santa Clara, Alfenas - MG, 37133-840, Brasil
alice.nogueira@sou.unifal-mg.edu.br

Iago Augusto Carvalho
Departamento de Ciências da Computação, Universidade Federal de Alfenas
Av. Jovino Fernandes de Sales, 2600 - Santa Clara, Alfenas - MG, 37133-840, Brasil
iago.carvalho@unifal-mg.edu.br

RESUMO

A avaliação empírica de algoritmos produz um grande conjunto de dados que precisam ser avaliados estatisticamente. Comumente, o conjunto gerado possui algumas entradas faltantes devido a inabilidade de um algoritmo em encontrar uma solução para determinado caso de teste. Estes dados faltantes limitam o uso de testes estatísticos, pois testes de hipótese não podem ser diretamente aplicados na avaliação destes dados. O método PAR-Δ emerge como um método de pré-processamento que realiza o *input* dos dados faltantes, criando um conjunto de dados completo que pode ser analizado estatisticamente. A literatura utiliza o método PAR-Δ com $\Delta = 10$. Entretanto, não existem justificativas para tal valor. Este trabalho avalia o poder do método PAR-Δ para valores de Δ entre 2 e 50. Os resultados demonstraram pouca variação para os diferentes valores de Δ, sendo que um teste estatístico não encontrou diferenças significativas para nenhum dos valores de Δ testados.

PALAVRAS CHAVE. PAR-Δ, Poder, Avaliação estatística, Programação Linear Inteira

Tópicos: Estatística e Modelos Probabilísticos, Programação Matemática.

ABSTRACT

The empirical evaluation of algorithms generates a large dataset that needs to be evaluated statistically. Commonly, the generated dataset has some missing entries due to the inability of an algorithm to find a solution for a given test case. These missing data strongly limit the use of statistical tests, as classical hypothesis tests cannot be directly applied to these datasets. The PAR-Δ method emerges as a pre-processing method that performs the *input* of missing data, creating a complete dataset that can be statistically analyzed. The literature uses the PAR-Δ method with $\Delta = 10$. However, there are no justifications for such a value. In this work, we evaluate the power of the PAR-Δ method for values of Δ between 2 and 50 employing a data set generated by an integer linear programming solver. The results showed little variation for different values of Δ, such that a statistical test did not find significant differences for the evaluated values of Δ.

KEYWORDS. Wireless Sensor Networks. Integer Linear Programs. Connectivity.

Paper topics: Statistics and Probabilistic Models, Mathematical Programming.

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

1. Introdução

O processo de projeto de um algoritmo geralmente leva em consideração alguma estrutura especial do problema em questão. Por exemplo, uma rede neural para reconhecimento de rostos em fotografias é treinada em conjuntos de dados com milhares de rostos diferentes das mais variadas posições [Gao et al., 2020], enquanto um algoritmo de busca local para um problema de roteamento de veículos emprega a estratégia de vizinhança mais eficaz para esta tarefa. problema [Arnold e Sörensen, 2019]. Portanto, um algoritmo pode ter um desempenho melhor que outro na resolução de um problema específico dependendo de como explora essas estruturas especiais. Consequentemente, surge a questão de decidir qual algoritmo é o melhor.

Esta questão pode ser respondida realizando experimentos computacionais em um conjunto de casos de teste, ou seja, instâncias de *benchmark*. Esta avaliação empírica resulta em grandes conjuntos de dados que precisam de ser avaliados através de testes estatísticos. Pode haver falta de entradas no conjunto de dados devido à incapacidade de um algoritmo de resolver um *benchmark* proposto. Esses valores faltantes impossibilitam o cálculo da média e do desvio padrão do conjunto de dados. Portanto, o uso de testes estatísticos é bastante restrito [Box et al., 2005].

A abordagem mais comum na literatura para lidar com este problema é ignorar os dados faltantes. Embora comumente utilizada, esta estratégia negligencia erroneamente informações sobre o comportamento dos algoritmos avaliados, penalizando os algoritmos eficientes que foram capazes de resolver todos os casos de teste.

Felizmente, existem vários métodos para pré-processar conjuntos de dados com entradas faltantes [Skillings e Mack, 1981; Wittkowski, 1988; Hansen et al., 2010; Kerschke et al., 2017; Carvalho, 2019]. Neste trabalho nós estamos interessados no método PAR-Δ [Bischl et al., 2016; Kerschke et al., 2017]. Este método realiza o *input* dos dados faltantes como um valor Δ vezes pior em relação ao pior resultado obtido por um algoritmo para um caso de teste específico.

A literatura utiliza o método PAR-Δ com $\Delta = 10$. Entretanto, nenhum trabalho da literatura fornece justificativas para a utilização deste valor. Neste trabalho estamos interessados em investigar o método PAR-Δ com diferentes valores de Δ. Nós investigamos a aplicação do método utilizando valores de Δ dentre 2 e 50 e avaliamos o seu poder quando aplicado junto ao teste de Friedman. Esta metodologia é realizada para avaliar um conjunto de dados proveniente da resolução de modelos de programação linear inteira [Gleixner et al., 2019].

O restante do trabalho está organizado da seguinte forma. A Seção 2 relata os trabalhos relacionados, enquanto a Seção 3 apresenta formalmente o método PAR-Δ. Em sequência, a Seção 4 exibe o método de geração de dados e a Seção 5 relata os experimentos computacionais. Finalmente, a Seção 6 conclui nossa pesquisa e apresenta possíveis direções para trabalhos futuros.

2. Trabalhos relacionados

Trabalhos anteriores na literatura compararam testes estatísticos abrangentes e procedimentos post-hoc para avaliação de dados de experimentos computacionais. O primeiro e mais importante deles foi o trabalho de Demšar [2006]. O autor avaliou o teste t, ANOVA, teste de Wilcoxon, teste de Friedman e teste de Sign, além de cinco procedimentos post-hoc para comparação com controle: teste de Dunnett, teste de Bonferroni-Dunn, teste de Holm, o teste de Hochberg e o teste de Hommel. Concluiu que os testes não paramétricos (Wilcoxon e Friedman) foram os testes estatísticos abrangentes mais eficazes. Além disso, uma análise experimental de poder demonstrou que o teste de Hommel tem maior poder do que os demais procedimentos post-hoc estudados.

Garcia e Herrera [2008] ampliaram o estudo de Demšar [2006] considerando testes de comparações múltiplas, nomeadamente o teste de Nemenyi, o procedimento de Holm, o procedimento estático de Shaffer e o Procedimento dinâmico de Bergmann-Hommel. Uma análise experimental do teste de pder demonstrou que o procedimento dinâmico de Bergmann-Hommel é o mais

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

poderoso entre os testes avaliados. No entanto, este teste é difícil de entender e computacionalmente caro. Assim, os autores incentivaram o uso do procedimento estático de Shaffer.

García et al. [2010] também ampliaram o trabalho de Demšar [2006] avaliando um grande número de testes estatísticos abrangentes e procedimentos post-hoc para comparação com um controle. Os autores descobriram que o teste omnibus Friedman Aligned Rank e o teste Quade têm maior poder do que o teste de Friedman, especialmente quando comparam menos de 5 algoritmos diferentes. Em relação aos procedimentos post-hoc concluiu-se que os procedimentos de Hommel e Rom foram os mais poderosos. No entanto, são também os mais difíceis de serem aplicados e compreendidos. Assim, os autores recomendaram a utilização do teste de Finner no caso geral e do teste de Li em alguns casos específicos.

Pereira et al. [2015] compararam o poder do teste de Friedman e do teste de Iman-Davenport. Concluíram que o teste de Iman-Davenport tem maior poder que o teste de Friedman e classificaram os procedimentos post-hoc em três grupos diferentes de acordo com seu poder.

Derrac et al. [2011] apresentaram um guia prático sobre como aplicar procedimentos não paramétricos para classificar metaheurísticas. Os autores demonstraram seu uso e compararam seu poder. Recentemente, uma nova pesquisa realizada por Carrasco et al. [2020] ampliou este guia prático apresentando as mais novas tendências neste campo. O leitor é encaminhado a esses trabalhos para uma discussão mais profunda sobre como classificar metaheurísticas.

Bossek et al. [2020] propuseram uma nova metodologia para avaliação de algoritmos. A proposta deles analisa duas ou mais métricas conhecidas em otimização multiobjetivo para construir uma fronteira de Pareto para cada um dos algoritmos avaliados. Então, o Hypervolume [Zitzler e Thiele, 1998], uma métrica bem conhecida para avaliar problemas de otimização multiobjetivo, é usado para decidir o melhor algoritmo.

Existem outros métodos na literatura que realizam o *input* de dados de maneira semelhante ao PAR-Δ. Dentre eles, podemos destacar o Ranking Bi-Objetivo Lexicográfico [Carvalho, 2019], o teste de Skillings–Mack [Skillings e Mack, 1981] e o teste de Wittkowski [Wittkowski, 1988]. Uma pequena revisão sobre estes métodos foi realizada por Carvalho e Moreira [2023].

Carvalho e Moreira [2023] avaliaram o poder de diferentes métodos de *input* de dados para completar conjuntos com dados faltantes, dentre eles o método PAR-Δ. Os autores observaram que o método PAR-Δ, com $\Delta = 10$, obtém bons resultados quando existem dados faltantes para menos de 30% dos benchmarks avaliados.

Apesar do número de trabalhos na literatura sobre avaliação de experimentos computacionais, não existe nenhum estudo que avalia o método PAR-Δ. Deste modo, nosso objetivo é avaliar o poder deste método, testando valores de Δ entre 2 e 50.

3. O método PAR-Δ

As pontuações *PAR*-Δ [Bischl et al., 2016; Kerschke et al., 2017] consideram o número de instâncias resolvidas por um algoritmo e a qualidade de seus resultados. A entrada é um vetor bidimensional $\mathbf{G}_{m,n}$ em que cada coluna representa um algoritmo e cada linha denota um caso de teste. Uma célula $g_{ij} \in \mathbf{G}$ corresponde ao resultado obtido pelo algoritmo j no benchmark i. Mais especificamente, cada célula g_{ij} contém o valor de dualidade quando estamos avaliando um algoritmo determinístico/exato ou a qualidade média da solução de múltiplas execuções para algoritmos estocásticos.

O método de pré-processamento *PAR*-Δ altera os valores g_{ij} como

$$g_{ij} = \begin{cases} g_{ij}, & \text{se } g_{ij} \text{ não está faltando} \\ \Delta \cdot max_g, & \text{se } g_{ij} \text{ está faltando} \end{cases},$$

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

para todo $g_{ij} \in \mathbf{G}$. Nesta equação, max_g representa o pior resultado de um algoritmo para determinado caso de teste e Δ representa um fator multiplicativo. Existem várias pontuações PAR-Δ, mas a mais amplamente aplicada é $\Delta = 10$, que é chamada de pontuação PAR10.

A pontuação PAR-Δ final de um algoritmo é definida como $\sum_{j=1}^{m} g_{ij}$, que é a soma de todas as suas pontuações, uma para cada benchmark. Pode-se argumentar que o algoritmo α tem um desempenho melhor que o algoritmo β se a pontuação PAR-Δ de α for menor que a de β (e.g., Bischl et al. [2016]). No entanto, outros estudos empregaram um teste estatístico abrangente para inferir se um algoritmo difere estatisticamente de outros (e.g., Kadioglu et al. [2011]).

4. Obtenção do conjunto de dados

O conjunto de dados utilizados neste estudo foi gerado de forma experimental. Nesta seção, nós descreremos o problema utilizado como base para geração dos dados na seção

4.1. Programação linear inteira

Nosso problema base é o clássico Problema de Programação Linear Inteira (MIP), um tradicional problema NP-Difícil. Um MIP pode ser definido como a função objetivo (1) e as restrições em (2)–(4). Ele usa variáveis de decisão $x \in \mathbb{Z}^n$ e $y \in \mathbb{R}^k$. Nesta formulação, $\mathbf{c}, \mathbf{d}$ e $\mathbf{b}$ são vetores de valores inteiros, e $\mathbf{A}$ e $\mathbf{B}$ são matrizes cujos coeficientes são inteiros.

$$\min \quad \mathbf{c}^T x + \mathbf{d}^T y \tag{1}$$
$$\mathbf{A}x + \mathbf{B}y \leq \mathbf{b} \tag{2}$$
$$x \in \mathbb{Z}^n \tag{3}$$
$$y \in \mathbb{R}^k \tag{4}$$

Os MIPs possuem diversas aplicações, como planejamento de agendamento de atividades ou processos [Stewart et al., 2023; Goubaa et al., 2020], projeto, manutenção e otimização de redes de comunicação [Carvalho e Ribeiro, 2020; Chagas et al., 2018], e aprendizado de máquina [Gambella et al., 2020; Bengio et al., 2020]. Neste trabalho, resolvemos vários MIPs usando a implementação do algoritmo de *branch-and-bound* de um dos solucionadores comerciais mais conhecidos da indústria, o IBM ILOG CPLEX Optimization Studio (CPLEX)[1].

4.2. Instâncias de benchmark

Nós utilizamos um total de 240 instâncias de benchmark obtidas do site da Mixed Integer Programming Library (MIPLIB) de 2017 Gleixner et al. [2019], que é o conjunto de instâncias mais conhecido para MIP na literatura. Todos eles pertencem ao *Benchmark set*[2], que pode ser resolvido adequadamente pelos algoritmos atuais em um período de tempo razoável. As instâncias representam problemas de otimização clássicos como cobertura de conjunto, empacotamento e mochila, entre outros. Além disso, essas instâncias possuem condições comuns a muitos problemas de otimização combinatória, como precedência entre tarefas e agregação de decisões. O tamanho dos problemas varia de 34 variáveis e 3 restrições a mais de um milhão de cada.

4.3. Geração dos dados

As instâncias acima descritas foram resolvidas utilizando o algoritmo de *branch-and-bound* implementado pelo CPLEX. Para gerar um conjunto de dados diverso, nós variamos a *chave de ênfase da Programação Inteira Mista*, um dos parâmetros disponíveis na configuração

[1] `https://www.ibm.com/produtos/ilog-cplex-optimization-studio`
[2] https://miplib.zib.de/tag_benchmark.html

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

do CPLEX. Este parâmetro possui 5 valores possíveis: (i) *balanced*; (ii) *feasibility*; (iii) *optimality*; (iv) *best bound*; e (v) *hiddenfeas*. Quando definido como *balanced*, o CPLEX inicialmente se concentra em encontrar soluções viáveis e depois tenta provar sua otimalidade. Por outro lado, usando *feasibility*, o CPLEX calcula muitas soluções viáveis enquanto resolve o MIP, diminuindo a velocidade da prova de otimalidade. Usando *optimality*, o comportamento do CPLEX é o oposto: ele enfatiza o cálculo da solução ótima e não se preocupa em calcular encontrar um grande número de soluções viáveis. Ao definir o parâmetro para *best bound*, o CPLEX coloca quase todo o seu esforço para provar a otimalidade de uma solução viável, melhorando o valor do melhor limite enquanto a detecção de soluções viáveis se torna quase acidental. Finalmente, usando *hiddenfeas*, o CPLEX se concentra em computar soluções viáveis de alta qualidade que de outra forma seriam muito difíceis de encontrar. No restante do texto, denotaremos respectivamente como $CPLEX_1, CPLEX_2, CPLEX_3, CPLEX_4$ e $CPLEX_5$ o algoritmo usando *balanced, feasibility, optimality, best bound* e *hiddenfeas*.

O algoritmo branch-and-bound do CPLEX resolveu as instâncias de benchmark usando os cinco valores diferentes para a chave de ênfase da chave de ênfase da Programação Inteira Mista. Os resultados completos obtidos neste experimento estão publicamente disponíveis em um repositório Mendeley. Este repositório será oculto aqui para não entregar a identidade dos autores do artigo. Entretanto, a referência a este repositório será inserido na versão final.

5. Experimento computacional

Os experimentos computacionais foram realizados em um Intel Xeon E5405 de núcleo único com clock de $2,0$ GHz e 32 GB de RAM, rodando GNU/Linux Ubuntu 20.04. Usamos o IBM ILOG CPLEX Optimization Studio C++ versão 12.9. O código foi compilado com o GNU g++ 8.2.0 e o tempo máximo de execução foi definido para 60 segundos. Os testes estatísticos foram realizados utilizando a linguagem de programação Python versão 3.1.1. Além disso, o método PAR-Δ foi desenvolvido também em Python versão 3.1.1 e codificado sem o auxílio de bibliotecas externas relevantes.

5.1. Medida de poder

Formalmente, o poder (ou erro do tipo 2) de um teste estatístico é definido como a probabilidade de o teste rejeitar corretamente uma hipótese nula falsa quando uma hipótese alternativa é verdadeira. Como não somos capazes de medir diretamente o poder de um teste, devemos avaliar indicativos deste poder.

Neste trabalho, empregamos uma medida de replicabilidade de resultados como indicativo do poder de um teste estatístico quando o PAR-Δ é utilizado como método de pré-processamento. Este método, proposto por Bouckaert [2004], é utilizado juntamente com o número de rejeições da hipótese nula. O autor definiu a replicabilidade de um teste estatístico como a probabilidade de que dois testes estatísticos com o mesmo número de amostras produzam os mesmos resultados, ou seja, que ambos os experimentos aceitem ou rejeitem a hipótese nula. O autor também formulou um estimador imparcial ideal desta probabilidade denotado como $R(e)$. Este estimador é definido na Equação (5), onde δ é o número total de experimentos. Além disso, e_i representa o resultado do i-ésimo experimento, tal que $e_i = 1$ se a hipótese nula não for rejeitada e $e_i = 0$ se a hipótese nula for rejeitada. O estimador também emprega uma função booleana $I(e_i == e_j)$, que retorna 1 se $e_i = e_j$ e 0 caso contrário. Quanto *maior* $R(e)$, mais *mais replicável* o teste estatístico. Destacamos que $R(e)$ varia entre $0,5$ e 1.

$$R(e) = \sum_{i=0}^{\delta-1} \sum_{j=i+1}^{\delta} \frac{2 \cdot I(e_i == e_j)}{\delta^2 - \delta} \tag{5}$$

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

5.2. O experimento

Este experimento utilizou o método PAR-Δ variando os valores de Δ como todos os valores inteiros entre 2 e 50. Os diferentes métodos PAR-Δ foram aplicados no conjunto de dados completo descrito na Seção 4 e o conjunto de dados obtidos foram utilizados como entrada para um teste de Friedman. Então, o poder deste teste de Friedman foi calculado utilizando a medida $R(e)$.

O resultado deste experimento é reportado na Figura 1. Esta figura mostra, no eixo X, os diferentes valores de Δ. Já no eixo Y, representa-se o valor de $R(e)$. Para cada valor de Δ exibe-se o valor médio encontrado para 100 repetições do método PAR-Δ e seu desvio padrão. Além disso, o gráfico também plota uma linha de regressão linear (em laranja) das médias encontradas. Como resultados, encontrou-se uma pequena diferença entre os valores de Δ avaliados, sendo que a diferença percentual entre o melhor e pior valor de delta foi de apenas 2 pontos no valor de $R(e)$. Além disso, a regressão linear apresentou uma equação quase estável, sendo ligeiramente decrescente com o aumento do valor de Δ.

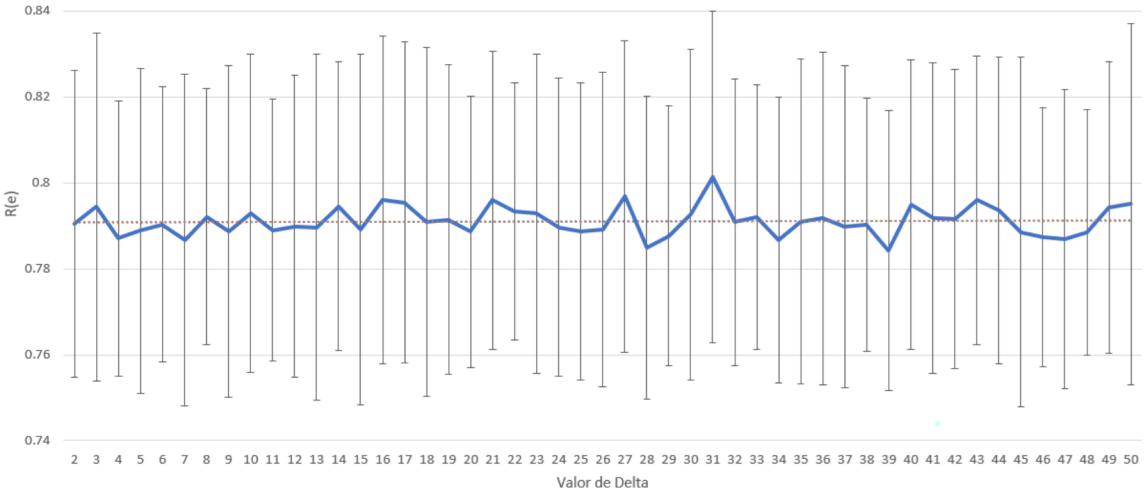


Figura 1: Valores de $R(e)$ obtidos para cada valor de Δ avaliado

Para avaliar se existe uma diferença significativa ou não na variação dos valores de Δ, nós utilizamos outro teste de Friedman. Este teste recebeu como entradas todos os $R(e)$ gerados com os diferentes valores de Δ, assim como apresentados na Figura 1. Este teste tinha como hipótese nula que não existe diferença significativa entre os diversos valores de Δ. Além disso, sua hipótese alternativa é que existe uma diferença significativa entre pelo menos dois dos valores de Δ avaliados.

Com um p-valor maior que 0.05, a hipótese nula foi aceita. Desta forma, pode-se concluir que, utilizando o experimento delineado neste artigo, não existe diferença significativa em utilizar quaisquer valores de Δ no método PAR-Δ.

6. Conclusões

Este trabalho avaliou o método de pré-processamento PAR-Δ, quando utilizado junto ao teste de Friedman, com valores de Δ variando entre 2 e 50. Um experimento computacional concluiu que não existe diferença no poder deste teste estatístico para os diferentes valores de Δ avaliados.

Este trabalho parcial fornece um primeiro *insight* sobre o comportamento do método PAR-Δ. Entretanto, novos experimentos são necessários para se confirmar se realmente não existe diferença para este método com diferentes valores de Δ. Provavelmente esta pequena variação se dá pela natureza não-paramétrica do teste de Friedman, sendo que seus rankings não consideram a magnitude das variações numéricas dos dados de entrada.

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

Trabalhos futuros neste tema poderão explorar outras métricas de reprodutibilidade, como a métrica $R(p)$ proposta por Demšar [2006]. Além disso, trabalhos futuros também poderão explorar maneiras mais inteligentes de amostragem do conjunto de dados gerados, como aquelas utilizadas recentemente por Carvalho e Moreira [2023]. Estas métricas de reprodutibilidade e técnicas de amostragem poderão ser utilizadas neste mesmo conjunto de dados ou em outros conjuntos de dados pertinentes obtidos a partir da avaliação de algoritmos heurísticos, meta-heurísticos ou aproximativos.

Referências

Arnold, F. e Sörensen, K. (2019). Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research*, 105:32–46.

Bengio, Y., Lodi, A., e Prouvost, A. (2020). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., et al. (2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58.

Bossek, J., Kerschke, P., e Trautmann, H. (2020). A multi-objective perspective on performance assessment and automated selection of single-objective optimization algorithms. *Applied Soft Computing*, 88:105901.

Bouckaert, R. R. (2004). Estimating replicability of classifier learning experiments. In *Proceedings of the twenty-first international conference on Machine learning*, p. 15. ACM.

Box, G. E., Hunter, J. S., e Hunter, W. G. (2005). *Statistics for experimenters: design, innovation, and discovery*, volume 2. Wiley-Interscience, New York. ISBN 978-0470570913.

Carrasco, J., García, S., Rueda, M., Das, S., e Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665.

Carvalho, I. A. (2019). On the statistical evaluation of algorithmic's computational experimentation with infeasible solutions. *Information Processing Letters*, 143:24–27.

Carvalho, I. A. e Moreira, A. F. (2023). Analysis of power in preprocessing methodologies for datasets with missing values. *Communications in Statistics-Simulation and Computation*, p. 1–15.

Carvalho, I. A. e Ribeiro, M. A. (2020). An exact approach for the minimum-cost bounded-error calibration tree problem. *Annals of Operations Research*, 287(1):109–126.

Chagas, R. J., Valle, C. A., e da Cunha, A. S. (2018). Exact solution approaches for the multi-period degree constrained minimum spanning tree problem. *European Journal of Operational Research*, 271(1):57–71.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

Derrac, J., García, S., Molina, D., e Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Gambella, C., Ghaddar, B., e Naoum-Sawaya, J. (2020). Optimization problems for machine learning: a survey. *European Journal of Operational Research*.

Gao, G., Yu, Y., Yang, M., Chang, H., Huang, P., e Yue, D. (2020). Cross-resolution face recognition with pose variations via multilayer locality-constrained structural orthogonal procrustes regression. *Information Sciences*, 506:19–36.

García, S., Fernández, A., Luengo, J., e Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064.

Garcia, S. e Herrera, F. (2008). An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677–2694.

Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P. M., Jarck, K., Koch, T., Linderoth, J., Lübecke, M., Mittelmann, H. D., Ozyurt, D., Ralphs, T. K., Salvagnin, D., e Shinano, Y. (2019). MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. Technical report, Optimization Online. URL `http://www.optimization-online.org/DB_HTML/2019/07/7285.html`.

Goubaa, A., Khalgui, M., Li, Z., Frey, G., e Zhou, M. (2020). Scheduling periodic and aperiodic tasks with time, energy harvesting and precedence constraints on multi-core systems. *Information Sciences*, 520:86–104.

Hansen, N., Auger, A., Finck, S., e Ros, R. (2010). Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA. URL `https://hal.inria.fr/inria-00462481`.

Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., e Sellmann, M. (2011). Algorithm selection and scheduling. In *International Conference on Principles and Practice of Constraint Programming*, p. 454–469. Springer.

Kerschke, P., Kotthoff, L., Bossek, J., Hoos, H. H., e Trautmann, H. (2017). Leveraging tsp solver complementarity through machine learning. *Evolutionary Computation*, p. 1–24. .

Pereira, D. G., Afonso, A., e Medeiros, F. M. (2015). Overview of friedman's test and post-hoc analysis. *Communications in Statistics-Simulation and Computation*, 44(10):2636–2653.

Skillings, J. H. e Mack, G. A. (1981). On the use of a friedman-type statistic in balanced and unbalanced block designs. *Technometrics*, 23(2):171–177.

Stewart, R., Raith, A., e Sinnen, O. (2023). Optimising makespan and energy consumption in task scheduling for parallel systems. *Computers & Operations Research*, 154:106212.

Wittkowski, K. M. (1988). Friedman-type statistics and consistent multiple comparisons for unbalanced designs with missing data. *Journal of the American Statistical Association*, 83(404):1163–1170.

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br

Zitzler, E. e Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, p. 292–301. Springer.

DOI: 10.59254/sbpo-2024-193505
https://proceedings.science/p/193505?lang=pt-br