ConBRepro
X CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO
EVENTO ON-LINE
02 a 04 de dezembro 2020

Desenvolvimento de uma metaheurística híbrida para o problema de agrupamento de dados

Felipe Kauai
Departamento de Administração Geral e Aplicada, PPGMNE - GTAO, Universidade Federal do Paraná, Curitiba, Brasil.
José Eduardo Pécora Junior
Departamento de Administração Geral e Aplicada, PPGMNE - GTAO, Universidade Federal do Paraná, Curitiba, Brasil.
Leonardo Silva de Lima
Departamento de Administração Geral e Aplicada, PPGMNE - GTAO, Universidade Federal do Paraná, Curitiba, Brasil.

Resumo: O problema de agrupamento de dados é uma área de pesquisa científica bastante ativa e encontra aplicações em vários problemas práticos dentro do campo da engenharia. A estrutura de problemas de agrupamento é conhecida por ser difícil de resolução através de métodos exatos, mesmo para instâncias consideradas pequenas. Assim, métodos heurísticos e metaheurísticos são frequentemente propostos como alternativas para obtenção de soluções consideradas razoáveis dentro de intervalos de tempo aceitáveis. Neste artigo, uma versão híbrida da metaheurística conhecida como *Black Hole Algorithm* é desenvolvida a partir da inserção de buscas locais em pontos estratégicos do algoritmo. Experimentos computacionais foram realizados em instâncias geradas artificialmente e instâncias geométricas amplamente utilizadas na literatura. Os resultados computacionais indicam que o algoritmo proposto é bastante eficaz e possui potencial para investigações mais aprofundadas.

Palavras-chave: Metaheurísticas, Análise de agrupamento, Otimização, Black Hole Algorithm

Development of a hybrid metaheuristic for clustering data problems

Abstract: The problem of cluster analysis is an active research field which has several potential applications in practical problems arising in the field of engineering. The structure of clustering problems is known to be had to solve by exact methods even for small-sized instances. Thus, many heuristics and metaheuristics are frequently proposed as alternative approaches to find reasonably good solutions within acceptable time intervals. In this work, a hybrid version of the Black Hole Algorithm metaheuristic is developed by inserting local searches at strategic places of the algorithm. Computational experiments run over artificially generated instances, as well as well-known geometric instances from the literature, are reported. Results indicate that the newly proposed algorithm is quite efficient and has potential to be developed further in future research.

Keywords: Metaheuristics, Cluster Analysis, Optimization, Black Hole Algorithm

1. Introdução

Uma das técnicas heurísticas mais populares para o problema de agrupamento é o algoritmo de k-médias (FRÄNTI; SIERANOJA, 2018), o qual opera através do agrupamento de pontos dentro de um determinado número de *clusters* tal que a soma das distâncias quadráticas entre cada ponto e seu centróide mais próximo é mínima. Entretanto, o desempenho do algoritmo depende fortemente da solução inicial fornecida e convergências em ótimos locais são frequentemente verificadas. Para evitar tais inconvenientes metaheurísticas podem ser utilizadas como alternativa.

Algoritmos metaheurísticos podem ser classificados basicamente de duas maneiras diferentes: algoritmos baseados em solução única e algoritmos baseado em populações (SAMSUDDIN *et al*., 2018). Metaheurísticas de solução única operam sobre uma única solução fornecida e a desenvolvem até o ponto que nenhuma solução melhor pode ser encontrada. Um exemplo clássico para este tipo de algoritmos é o *Simulated Annealing* (KIRKPATRICK *et al*., 1983). Por outro lado, metaheurísticas com base em populações otimizam populações de soluções. Um exemplo típico para essa última categoria são os Algoritmo Genéticos, os quais operam de acordo com a teoria de evolução de Darwin (HOLLAND, 1992).

O algoritmo do buraco negro, ou *Black Hole Algorithm* (BHA) em inglês, é um algoritmo baseado em populações e fora desenvolvido de acordo com o funcionamento de buracos negros no universo (HATAMLOU, 2013). O algoritmo opera através da inicialização de uma população inicial de soluções randômicas (estrelas) e a melhor solução dentre elas é escolhida como o buraco negro. Iterativamente, as estrelas gravitam em direção ao buraco negro e duas situações são possíveis: uma estrela adquire uma função objetivo melhor que a do buraco negro, e nesse caso essa nova estrela se torna o buraco negro, ou uma estrela atravessa o horizonte de eventos (*event horizon*) e morre, e nesse caso uma nova estrela é alocada randomicamente no espaço de soluções.

Além do problema de agrupamento, para o qual o BHA fora desenvolvido, o algoritmo tem sido aplicado com sucesso em outros problemas de otimização (HATAMLOU, 2018; WARNANA, 2018; PASHAEI *et al.*, 2019). Desde sua concepção, entretanto, poucos trabalhos acadêmicos replicaram ou exploraram potenciais hibridizações do algoritmo, mesmo embora sua implementação seja razoavelmente simples e sua eficácia potencialmente interessante. Assim, o presente trabalho tem por objetivo estender a ideia do BHA e desenvolver uma hibridização do algoritmo com buscas locais em pontos estratégicos. O algoritmo, a partir de agora chamado de BHA-LS, é testado em um conjunto de instâncias geradas artificialmente de pequeno, médio e grande porte.

Este artigo está organizado da seguinte maneira. A Seção 2 descreve o problema de agrupamento e uma formulação matemática é apresentada. Na Seção 3, o BHA original é discutido e o algoritmo proposto é desenvolvido em detalhes. Na Seção 4, os experimentos computacionais são apresentados e os resultados obtidos discutidos. Por fim, a Seção 5 conclui o trabalho com indicações de direções futuras de investigação sobre o tema.

2. Descrição matemática do problema

A análise de agrupamento tem por objetivo agrupar um conjunto de objetos em subgrupos disjuntos tal que a similaridade desses objetos dentro de cada subgrupo é máxima. Usualmente, para avaliar a similaridade entre objetos a distância Euclideana é utilizada.

Assim, o problema pode ser formulado da seguinte maneira: dado um conjunto de N objetos e *M* subgrupos, o objetivo é alocar cada objeto em um dos M subgrupos, tal que a soma das distâncias quadráticas, entre cada par de objetos, é minimizada. Nascimento *et al.* (2010) fornece uma formulação matemática linearizada com base em uma representação não linear proposta por Rao (1971). A formulação linear pode ser escrita como segue:

$$min\ F(Y) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} y_{ij} \qquad (1.0)$$

sujeito a:

$$\sum_{k=1}^{M} x_{ik} = 1, \qquad i = 1, \dots, N \qquad (1.1)$$

$$\sum_{i=1}^{N} x_{ik} \geq 1, \qquad k = 1, \dots, M \qquad (1.2)$$

$$x_{ik} \in \{0,1\} \qquad i = 1, \dots, N, \qquad k = 1, \dots, M \qquad (1.3)$$

$$y_{ij} \geq x_{ik} + x_{jk} - 1, \qquad i = 1, \dots, N, \qquad j = i + 1, \dots, N, \qquad k = 1, \dots, M \qquad (1.4)$$

$$y_{ij} \geq 0, \qquad i, j = 1, \dots, N, \qquad (1.5)$$

A função objetivo tem por objetivo minimizar a distância d_{ij} entre todos os objetos que pertencem ao mesmo *cluster*. Restrições (1.1) garantem que um objeto é alocado somente em um *cluster*. Restrições (1.2) garantem que um *cluster* tem no mínimo um objeto, i.e. é não vazio. Restrições (1.3) garantem que x_{ik} é estritamente binário. Restrições (1.4) e (1.5) garantem que y_{ij} é 1 se, e somente se, x_{ik} e x_{jk} são 1, e 0 caso contrário. O algoritmo desenvolvido nesse trabalho é adaptado para resolver essa formulação.

3. O algoritmo *Black Hole* e a hibridização proposta

O algoritmo de buraco negro é uma metaheurística com base em população, desenvolvida com inspiração no fenômeno de buracos negros na natureza. Assim, na inicialização, uma população de soluções, chamada população de estrelas, é randomicamente alocada no espaço de busca. A cada iteração, o valor da função objetivo é computado, e a melhor solução é selecionada para ser o buraco negro. As estrelas gravitam em direção ao buraco negro usando a seguinte fórmula:

$$x_i(t+1) = x_i(t) + r * (x_{BN} - x_i(t)), \qquad i = 1, \dots, P \qquad (2.0)$$

onde $x_i(t)$ e $x_i(t+1)$ são soluções que representam as localizações da i-ésima estrela nas iterações t e $t+1$, respectivamente. A solução x_{BN} é a localização atual do buraco negro e r é um número aleatório no intervalo $[0,1]$. O parâmetro P representa o tamanho da população e é definido na seção de experimentos computacionais.

A medida que as estrelas gravitam em direção ao buraco negro, algumas delas podem alcançar um valor de função objetivo melhor do que aquele do buraco negro e, assim, o buraco negro é substituído pela nova estrela. Ainda, se uma estrela entra no domínio do

buraco negro, isto é, o horizonte de eventos, a estrela é absorvida, e uma nova estrela aleatória é alocada no espaço de busca. Esse último fenômeno garante que a população se mantenha constante com respeito ao número de potenciais candidatos. O raio do horizonte de eventos é calculado da seguinte maneira:

$$R_i = \frac{f_{BN}}{f_i}, \tag{2.1}$$

onde f_{BN} é o valor da função objetivo do buraco negro, f_i é o valor da função objetivo da estrela i e R_i é o raio do horizonte de eventos do buraco negro. Quando a distância de uma estrela ao buraco negro é menor que R_i, então a estrela colapsa e uma nova é randomicamente gerada. Como descrito em Soto *et al.*, (2018) a morte de uma estrela é calculada através da geração de um número real aleatório, a partir de uma distribuição uniforme, no intervalo [0,1]. Se o número gerado for menor que R_i, computado para cada estrela individualmente, então uma nova estrela nascerá no lugar da estrela que colapsou.

No presente trabalho introduzimos o conceito de posições relativas para que a natureza estocástica do cálculo acima citado seja suprimida. Esse conceito permite controlar soluções que estão muito próximas do buraco negro, mas que, no entanto, possuem custos maiores. O raio do horizonte de eventos é calculado como em (2.1) e a posição relativa (PR) da estrela i é calculada como segue:

$$PR_i = \frac{f_i}{\sum_{j=1, j\neq i}^{P} f_j}\,. \tag{3.0}$$

Dessa forma, a distância relativa da estrela i para o buraco negro é obtida da seguinte maneira:

$$d_{i,bn} = \frac{PR_i - R}{R}\,. \tag{3.1}$$

Se $d_{i,bn}$ é menor que um parâmetro Δ, chamado de ponto de não retorno, então a estrela morre e uma nova é gerada aleatoriamente em seu lugar. Note que a diversificação do sistema pode ser parcialmente controlada através do ajuste de Δ. Quando uma estrela colapsa sobre si mesma e a formação de um buraco negro estelar é iniciada, o material da estrela morta é progressivamente comprimido em um objeto progressivamente mais denso. Seguindo esse princípio, quando uma solução candidata é selecionada como o buraco negro, introduzimos uma pesquisa local bastante rápida, chamada de *Densificação*, a qual deverá aumentar a densidade da estrela, ou seja, reduzir o seu custo. O algoritmo proposto pode ser lido no seguinte pseudocódigo:

Algoritmo 1 *Densificação*

Input: solução do buraco negro
1: RC = selecionar um *cluster* aleatório
2: **for** i = 1 to N **do**
3: **if** i está contido em RC **then**
4: **for** j = 1 to M **do**
5: retirar objeto de RC e alocar em j
6: calcular função objetivo
7: **if** função objetivo > função objetivo do buraco negro **then**
8: actualizar configuração da solução do buraco negro
9: RC = j
10: **end if**
11: **end for**
12: **end if**
13: **end for**
Output: solução do buraco negro otimizada

Um critério de parada para o algoritmo original do BHA pode ser a imposição de um número máximo de iterações independente do estado do sistema, ou um número máximo de iterações sem melhoria. Neste trabalho, a última opção é escolhida para a inserção de um segundo tipo de pesquisa local no algoritmo.

Físicos nos ensinam que buracos negros não têm vida eterna, uma vez que eles emitem partículas subatômicas. Esse processo é conhecido como radiação de Hawking e seu nome é uma homenagem ao físico Stephen Hawking. Assim, se nenhuma fonte de energia é fornecida ao buraco negro, este teria sua energia rotacional eventualmente esgotada, desaparecendo da existência. Como um paralelo, no nosso algoritmo, após um certo número de iterações sem melhoria, i.e. o buraco negro é isolado do sistema de estrelas, um segundo tipo de pesquisa local é desenvolvida de maneira a otimizar a solução do buraco negro e fazer com que o sistema seja ativo novamente. Essa última pesquisa local recebe o nome de *Radiação* e o pseudocódigo é apresentado a seguir:

Algoritmo 2 *Radiação*

Input: solução candidata do buraco negro
1: **for** i = 1 to ΩN **do** (Ω é um número real menor que 1)
2: RN = selecionar um nó aleatório no intervalo [1, N]
3: **for** j = 1 to M **do**
4: **if** RN está alocado em j **then**
5: **for** k = 1 to M\j **do**
6: insira RN em k
7: calcular função objetivo
8: **if** função objetivo > função objetivo candidata **then**
9: atualizar configuração da solução candidata
10: **end if**
11: **end for**
12: **end if**
13: **end for**
14: **end for**
Output: solução do buraco negro otimizada

O algoritmo BHA-LS consistem em gerar uma população aleatória de estrelas na inicialização, impor o número de iterações sem melhoria para operação, gravitar as estrelas em direção ao buraco negro e gerar novas estrelas de acordo com as distâncias relativas, introduzidas anteriormente. A cada vez que um novo buraco negro é encontrado o algoritmo *Densificação* é chamado. Também, a cada vez que o buraco negro permanece preso em um estado sub ótimo por um certo número de iterações, o algoritmo *Radiação* é chamado. O pseudocódigo para o algoritmo geral é apresentado a seguir:

Algoritmo 3 *BHA-LS*

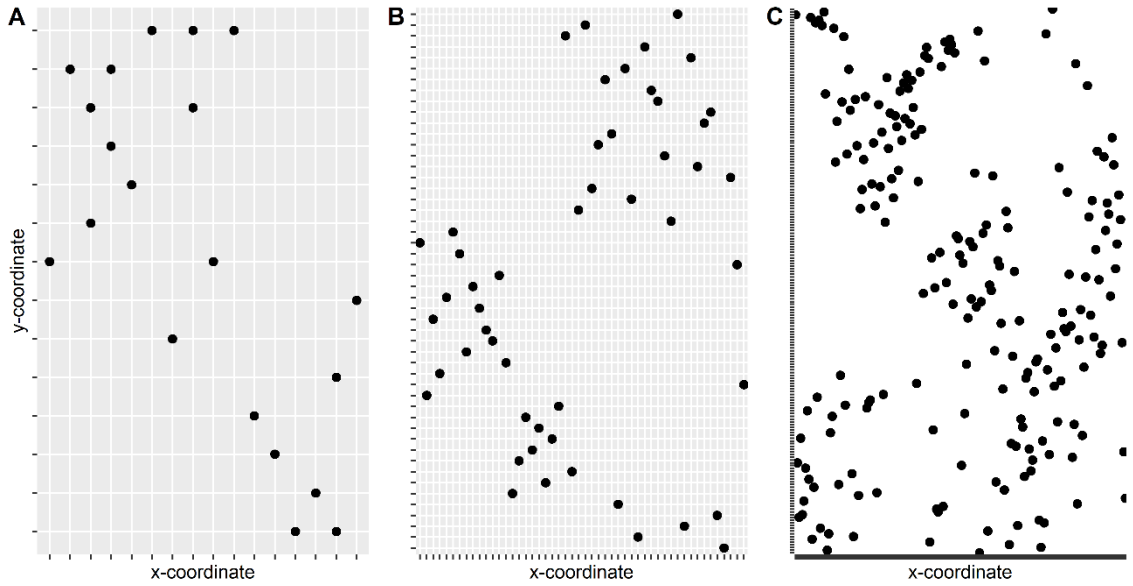
Input: conjunto de dados
1: δ = inicializar o número máximo de iterações sem melhoria
2: contador = 0
3: encontrar buraco negro dentre as estrelas
4: **while** contador < δ **do**
5: gravitar as estrelas em direção ao buraco negro de acordo com (2.0)
6: calcular função objetivo das estrelas
7: **if** novo buraco negro é encontrado **then**
8: chamar *Densificação*
9: atualizar configuração do buraco negro
10: **else**
11: calcule posições relativas de acordo com (3.0)
12: **for** i to P **do**
13: calcule distância relativa da estrela i de acordo com (3.1)
14: **if** $PR_i < \Delta$ **then**
15: gerar uma estrela aleatória no lugar de i
16: **end if**
17: **end for**
18: repetir 11-17 até que não haja mais estrelas
19: **end if**
20: **if** um múltiplo β de δ fora alcançado **then**
21: chamar *Radiação*
22: **end if**
23: **end while**
Output: solução final

4. Experimentos computacionais e resultados

4.1 Descrição do conjunto de dados

O desempenho do algoritmo proposto fora avaliado através da resolução de três grupos de cinco instâncias bidimensionais geradas artificialmente. O primeiro grupo é composto por 5 instâncias com 20 pontos e 2 *clusters* (C2N20), o qual classificamos como instâncias de pequeno porte; o segundo grupo com 5 instâncias de 50 pontos e 4 *clusters* (C4N50), o qual classificamos como instâncias de médio porte e o terceiro grupo com 5 instâncias de 200 pontos e 10 *clusters* (C10N200), o qual classificamos como grande porte. O primeiro e o segundo grupo foram resolvidos até o ótimo global, enquanto o terceiro grupo é composto por instâncias com soluções parciais. A figura 1, a seguir, mostra a distribuição dos dados no plano, para uma instância de cada grupo.

Figura 1. A – Instância C2N20_005. B - C4N50_005. C - C10N200_005



Para testar o desempenho do algoritmo sobre problemas de agrupamento com diferentes arranjos geométricos, as instâncias *Flame*, *R15* e *Aggregation*, foram utilizadas. Essas instâncias são amplamente utilizadas na literatura e estão disponíveis publicamente no seguinte endereço eletrônico: http://cs.joensuu.fi/sipu/datasets/.

4.2 Experimentos computacionais e resultados

O algoritmo BHA-LS possui 4 parâmetros, Δ, δ, β e Ω. O ponto de não retorno, Δ, foi configurado em 0.25. O número de iterações sem melhoria, δ, foi configurado em 100. O número de iterações antes de iniciar *Radiação*, foi de 25. Por último, Ω foi configurado em 0.25. Durante pré-teste foi verificado que maiores valores de Ω não contribuíam para a melhoria da qualidade de soluções e só aumentavam o tempo de execução computacional.

O algoritmo proposto foi implementado em Java e executado em um computador Intel Core i5-7200U CPU @3.50 GHz e 8 GB de memória. Resultados foram calculados com base em 100 execuções do algoritmo para cada instância. O desvio da solução obtida com a solução obtida através da execução com o modelo exato (Gap) é calculado da seguinte maneira:

$$Gap = \frac{Solução\ exata - Solução\ encontrada}{Solução\ exata} x\ 100$$

Assim, se o Gap é negativo, temos a porcentagem média que a solução encontrada afastou-se da solução exata. Se, no entanto, o Gap é positivo, temos a porcentagem média de melhoria em relação à solução exata disponível. As tabelas 1 e 2 mostram os resultados obtidos para as instâncias artificiais estudadas. Note-se que as instâncias C10N200 tiveram todas um *Gap* médio positivo. Isso decorre do fato de que o algoritmo exato não foi capaz de convergir para a solução ótima nessas instâncias.

Tabela 1. Resultados encontrados para as instâncias C2N20 e C4N50. Resultados são baseados em 100 execuções do BHA-LS por instância

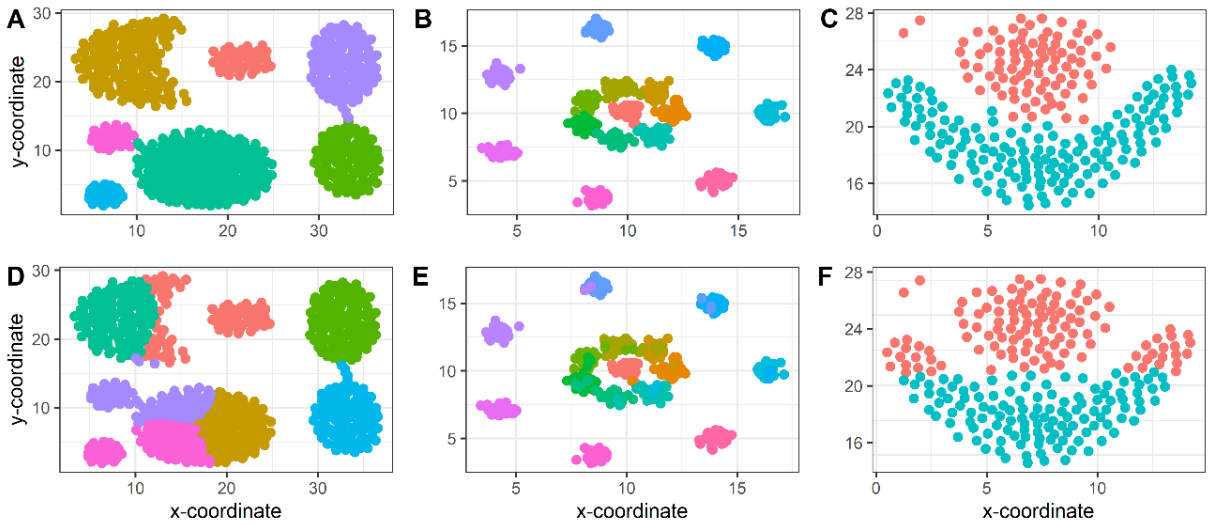
Instâncias	Função objetivo ótima	Gap médio (%)	Número de ótimos encontrados	Tempo de processamento médio (s)
C2N20_001	8348.6836	-0.34	71	0.0013
C2N20_002	8009.9048	-0.05	73	0.0014
C2N20_003	8201.6025	-1.37	79	0.0020
C2N20_004	6891.7427	-3.75	75	0.0016
C2N20_005	8410.6553	-1.08	78	0.0015
C4N50_001	32359.9023	-1.07	34	0.1675
C4N50_002	30562.3027	-1.87	33	0.1689
C4N50_003	27175.5898	-0.81	43	0.1688
C4N50_004	27483.0547	-1.97	25	0.1639
C4N50_005	29114.4160	-1.78	57	0.1525

Tabela 2. Resultados encontrados para as instâncias 10CN200. Resultados são baseados em 100 execuções do BHA-LS por instância

Instâncias	Melhor função objetivo factível	Gap médio (%)	Tempo de processamento médio (s)
C10N200_001	248272.781250	40.11	3.52
C10N200_002	210943.421875	29.64	3.96
C10N200_003	282760.937500	39.25	3.40
C10N200_004	282364.250000	42.57	3.56
C10N200_005	311180.500000	45.79	4.06

A figura 2 apresenta o resultado médio obtido a partir de 100 execuções do algoritmo nas instâncias geométricas. Os gráficos superiores (A, B e C) representam a solução ótima das instâncias *Aggregation*, *R15* e *Flame*, respectivamente. Os gráficos inferiores (D, E e F), apresenta a configuração obtida pelo BHA-LS nas instâncias *Aggregation*, *R15* e *Flame*, respectivamente.

Figura 2. Gráficos na parte superior da figura (A, B e C) apresentam as configurações ótimas das instâncias *Aggregation*, *R15* e *Flame*, respectivamente. Gráficos na parte inferior da figura (D, E e F) apresentam as soluções media obtidas pelo BHA-LS nas instâncias *Aggregation*, *R15* e *Flame*, respectivamente



4.3 Conclusões

O algoritmo proposto apresentou resultados bastante interessantes nas instâncias analisadas e uma investigação mais aprofundada será necessária para avaliar seu desempenho em instâncias mais complexas da literatura. Ademais, a inserção de pesquisas locais mais sofisticadas poderá ser explorada em futuros trabalhos, de maneira a construir diferentes possibilidades de implementação, as quais podem, em última análise, melhorar a eficácia e desempenho do algoritmo.

Referências

FRÄNTI, P.; SIERANOJA, S. K-means properties on six clustering benchmark datasets. **Applied Intelligence**, v. 48, n. 12, p. 4743-4759, 2018.

HATAMLOU, A. Black hole: A new heuristic optimization approach for data clustering. **Information sciences**, v. 222, p. 175-184, 2013.

HATAMLOU, A. Solving travelling salesman problem using black hole algorithm. **Soft Computing**, v. 22, n. 24, p. 8167-8175, 2018.

HOLLAND, J. H. Genetic algorithms. **Scientific American**, v. *267*, n. 1, p. 66-73, 1992.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671-680, 1983.

NASCIMENTO, M. C.; TOLEDO, F. M.; CARVALHO, A. C. Investigation of a new GRASP-based clustering algorithm applied to biological data. **Computers & Operations Research**, v. 37, n. 8, p. 1381-1388, 2010.

PASHAEI, E.; PASHAEI, E.; AYDIN, N. Gene selection using hybrid binary black hole algorithm and modified binary particle swarm optimization. **Genomics**, v. 111, n. 4, p. 669-686, 2019.

RAO, M. R. Cluster analysis and mathematical programming. **Journal of the American statistical association**, v. 66, n. 335, p. 622-626, 1971.

SAMSUDDIN, S.; OTHMAN, M. S.; YUSUF, L. M. A review of single and population-based metaheuristic algorithms solving multi depot vehicle routing problem. **International Journal of Software Engineering and Computer Systems**, v. 4, n. 2, p. 80-93, 2018.

WARNANA, D. D. (2018). Black hole algorithm for determining model parameter in self-potential data. **Journal of Applied Geophysics**, v. *148*, p. 189-200, 2018.