UNIVERSIDADE FEDERAL DE VIÇOSA
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

USO DA META-HEURÍSTICA BORDER COLLIE OPTIMIZATION NA RESOLUÇÃO DO PROBLEMA DO CAIXEIRO-VIAJANTE

BOLSISTA: Artur Souza Papa
ORIENTADOR: Marcus Henrique Soares Mendes

Relatório Final, referente ao período de 07/2021 a 07/2022, apresentado à Universidade Federal de Viçosa, como parte das exigências do PIBIC/CNPq.

FLORESTAL
MINAS GERAIS - BRASIL
SETEMBRO 2022

UNIVERSIDADE FEDERAL DE VIÇOSA
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RESUMO

USO DA META-HEURÍSTICA BORDER COLLIE OPTIMIZATION NA RESOLUÇÃO DO PROBLEMA DO CAIXEIRO-VIAJANTE

O Problema do Caixeiro-Viajante (PCV) é um dos clássicos problemas de otimização combinatória, consistindo em determinar em um grafo ponderado G= (N,M), onde N = 1,...,n representa o conjunto de vértices do grafo e M = 1,...,m o conjunto de arestas, um ciclo hamiltoniano de menor custo. O PCV é um dos problemas de otimização combinatória mais intensamente pesquisados e possui diversas aplicações reais e muitas variantes que por sua vez também possuem várias aplicações reais.

A importância do PCV pode ser atribuída a três características combinadas: 1) grande número de aplicações práticas; 2) relação com outras problemas e muitas variantes e 3) grande dificuldade de solução exata. O PCV é NP-Difícil.

Devido à complexidade do PCV é interessante o uso de heurísticas e meta-heurísticas para solucioná-lo, a fim de se obter boas soluções num tempo razoável. Afinal essas estratégias têm sido utilizadas com sucesso para solucionar diversos problemas complexos de otimização, inclusive no próprio problema do caixeiro-viajante e nas suas variantes.

Recentemente na literatura, em junho de 2020, foi proposta uma nova meta-heurística intitulada *Border Collie Optimization* (BCO) que forneceu resultados bem competitivos na resolução de trinta e cinco problemas de otimização contínuos quando comparadas a outras meta-heurísticas.

Diante desse contexto é propício estender a aplicabilidade do BCO aos problemas de otimização combinatória, especificamente ao PCV, o que pode ser feito por meio da incorporação de mecanismos capazes de lidar com parâmetros discretos.

Assim, nesta pesquisa pretende-se implementar um algoritmo baseado no BCO que seja capaz de resolver problemas de otimização combinatória, com foco no PCV.

Data: __/___/_____

Assinatura do Orientador

Assinatura do Bolsista

SUMÁRIO

1 INTRODUÇÃO

O Border Collie Optimization (BCO) é uma meta-heurística baseada em enxame (swarm). Uma meta-heurística baseada em enxame inspira-se no comportamento social de insetos ou animais. Em um enxame, cada indivíduo tem sua própria inteligência e comportamento. O comportamento combinado dos indivíduos torna o enxame uma ferramenta poderosa para resolver problemas complexos. Em geral, as meta-heurísticas baseadas em enxame são fáceis de implementar e exigem menor número de parâmetros. Além disso, operadores complexos como mutação, elitismo e crossover usados na evolução algoritmos não são necessários para implementar enxames. (1)

O BCO imita o comportamento de pastoreio dos cães Border Collie, que é uma raça afetuosa, inteligente e enérgica. Os cães Border Collie são extremamente inteligentes, atléticos e podem ser facilmente treinados. O pastoreio é uma habilidade inerente que eles possuem. Mesmo quando um filhote é apresentado ao pastoreio pela primeira vez, ele demonstra um imenso controle sobre as ovelhas.

Os cães Border Collie adotam uma abordagem diferente para o pastoreio. Em vez de se aproximarem por trás, eles pastoreiam ovelhas dos lados e pela frente. Eles seguem, principalmente, três técnicas de pastoreio: gathering (reunião), stalking (perseguição) e eyeing (observação). No gathering, os cães Border Collie controlam as ovelhas pelos lados e pela frente. Eles tendem a reuni-los e direcioná-los para a fazenda. No stalking, os cães Border Collies adotam poucos movimentos parecidos com de lobo quando ele vai controlar as ovelhas. Eles se agacham abaixando suas cabeças, colocam seus traseiros alto e seus rabos para baixo. No eyeing, os cães Border Collie imitam o comportamento dos lobos de seleção da vítima. Isso é chamado de “dar uma olhada” ou de observação. Quando as ovelhas se perdem, esses cães inteligentes olham fixamente nos seus olhos. Isso exerce pressão psicológica para que o grupo se mova na direção correta.

As principais características do BCO são as seguintes: 1) possui mecanismos de exploração e explotação do espaço de busca, que é essencial para o sucesso das meta-heurísticas; 2) possui mecanismo de feedback, sendo que o eyeing está relacionado com feedback negativo e que o gathering e o stalking estão relacionados com feedback positivo; 3) possui habilidade de sair de ótimos locais; 4) possui poucos parâmetros independentes e; 5) possui fácil implementação e guarda a solução ótima.

2 BORDER COLLIE OPTIMIZATION

Existem três técnicas principais que fazem com que o BCO funcione de maneira eficiente, sendo elas:

- **Gathering**: Os Border Collies controlam as ovelhas de lado e frente. Tendem a reuni-los e a dirigi-los em direção à fazenda. Isto é conhecido como *gathering*.

- **Stalking**: Os Border Collies adotam alguns movimentos semelhantes aos dos lobos quando se trata de controlar as ovelhas. Agacham-se abaixando a cabeça, colocam os quartos traseiros bem alto e pousam as suas caudas. Este comportamento é chamado perseguição.
- **Eyeing**: Border Collies imitam o comportamento de seleção da vítima dos lobos. Isso se chama encarar ou *eyeing*. Quando a ovelha se perde, esses cães inteligentes encará-los nos olhos. Isso exerce pressão psicológica no bando para se mover na direção correta.

Além das três técnicas citadas, também é válido dizer que a população é separada em quatro componentes, sendo eles: o cão líder, o cão à esquerda e à direita e as ovelhas.

2.0.1 CÁLCULO DA VELOCIDADE E ACELERAÇÃO

A velocidade de todos os três cães, no momento (*t+1*) é calculada usando as seguintes equações:

$$V_f(t+1) = \sqrt{V_f(t)^2 + 2 \times Acc_f(t) \times Pop_f(t)} \quad (1)$$

$$V_{ri}(t+1) = \sqrt{V_{ri}(t)^2 + 2 \times Acc_{ri}(t) \times Pop_{ri}(t)} \quad (2)$$

$$V_{le}(t+1) = \sqrt{V_{le}(t)^2 + 2 \times Acc_{le}(t) \times Pop_{le}(t)} \quad (3)$$

Figura 1: Velocidade dos cães

As equações (1), (2) e (3), *Vf(t + 1)*, *Vri(t + 1)* e *Vle(t + 1)* representam a velocidade no tempo *(t + 1)* para o líder, cão à direita e à esquerda respectivamente. Da mesma forma, *Vf(t)*, *Vri(t)* e *Vle(t)* representam velocidade no tempo *(t)* para o cão líder, da direita e esquerda. *Accf(t)*, *Accri(t)* e *Accle(t)* significam aceleração no *tempo(t)* para o cão líder, cão da direita e cão da esquerda, respectivamente. *Popf(t)*, *Popri(t)* e *Pople(t)* são os posições do cão líder, cão direito e cão esquerdo no momento *(t)*, respectivamente. A equação (1) atualiza a velocidade do cão líder. As equações (2) e (3) atualizam as velocidades dos cães da direita e esquerda, respectivamente. (1)

A velocidade das ovelhas é atualizada utilizando as três técnicas de pastoreio.

Gathering: As ovelhas que estão mais perto do cão líder, movem-se na direção do cão líder. Por isso, essas ovelhas são apenas reunidas. Elas são escolhidos com base em seus valores de fitness.

$$D_g = (fit_f - fit_s) - ((\frac{fit_{le} + fit_{ri}}{2}) - fit_s) \quad (4)$$

Figura 2: Função do fitness

Em (4), se o valor de *Dg* é positivo, isso indica que a ovelha está próxima do cão líder. Neste caso a velocidade da ovelha é atualizada utilizando a seguinte equação.

$$V_{sg}(t+1) = \sqrt{V_f(t+1)^2 + 2 \times Acc_f(t) \times Pop_{sg}(t)} \tag{5}$$

Figura 3: Velocidade das ovelhas em gathering

Em (5), a velocidade da ovelha, *Vsg* é diretamente influenciada pela velocidade do cão líder no momento *(t + 1)* e aceleração do cão líder, no instante *(t)*. *Popsg* é a localização atual das ovelhas a serem coletadas.

Stalking: As ovelhas que estiverem próximas do cão da esquerda e da direita, precisam ser seguidas pelos lados para se manterem no caminho certo. Estas são as ovelhas que possuem o valor de *Dg* negativo. As equações para a atualização da velocidade das ovelhas perseguidas estão apresentadas abaixo.

$$v_{ri} = \sqrt{(V_{ri}(t+1)tan(\theta_1))^2 + 2 \times Acc_{ri}(t) \times Pop_{ri}(t)} \tag{6}$$

$$v_{le} = \sqrt{(V_{le}(t+1)tan(\theta_2))^2 + 2 \times Acc_{le}(t) \times Pop_{le}(t)} \tag{7}$$

$$V_{ss}(t+1) = \frac{v_{le} + v_{ri}}{2} \tag{8}$$

Figura 4: Velocidade das ovelhas em stalking

Em (8), a velocidade da ovelha perseguida, *Vss* depende da velocidade dos cachorros da esquerda e da direita. Vale notar que os valores de $\theta 1$ varia de (1-89) graus e $\theta 2$ varia de (91 - 179) graus. Os valores de $\theta 1$ e $\theta 2$ são escolhidos de maneira aleatória.(1)

Eyeing: As ovelhas que estão totalmente desgarradas são as aquelas que precisam do *eyeing*. *Eyeing* e implementado, quando em iterações consecutivas, o fitness de um indivíduo não melhora. Nesse caso, o cão com o pior fitness é o responsável por ir atrás da ovelha e dar-lhe uma encarada. Tal tipo de ação são representadas pelas seguinte fórmulas.

$$V_{se}(t+1) = \sqrt{V_{le}(t+1)^2 - 2 \times Acc_{le}(t) \times Pop_{le}(t)} \tag{9}$$

Figura 5: Velocidade das ovelhas em eyeing

$$V_{se}(t+1) = \sqrt{V_{ri}(t+1)^2 - 2 \times Acc_{ri}(t) \times Pop_{se}(t)} \tag{10}$$

Figura 6: Velocidade das ovelhas em eyeing

Em (9), *Vle(t + 1)* e *Accle(t)* são a velocidade e aceleração do cão à esquerda quando este tem o pior fitness dentre os três cães. Em (10), *Vri(t + 1)* e *Accri(t)* são a velocidade e aceleração do cão à direita quando este tem o pior fitness dentre os três cães. É considerado o

cão com o menor fitness porque se assume que este está mais próximo da ovelha.(1)

Aceleração dos cães e das ovelhas: A equação para atualizar a aceleração é derivada da função mais comum utilizada na física.

$$Acc_i(t+1) = \frac{(V_i(t+1) - V_i(t))}{Time_i(t)} \qquad (11)$$

Figura 7: Aceleração dos cães e das ovelhas

A aceleração de todos os cães e ovelhas, *Accf(t + 1), Accri(t + 1), Accri(t + 1), Accri(t + 1), Accss(t + 1)* e *Accse(t)* são atualizados usando (11). i ϵ {f,le,ri,sg,ss para se}. (1)

2.0.2 CÁLCULO DO TEMPO

O tempo *(T)* de travessia é atualizado para cada indivíduo usando o seguinte equação.

$$Time_i(t+1) = \text{Avg} \sum_{i=1}^{d} \frac{(V_i(t+1) - V_i(t))}{Acc_i(t+1)} \qquad (12)$$

Figura 8: Cálculo do tempo para cães e ovelhas

2.0.3 POSIÇÃO DOS CÃES E OVELHAS

As localizações de cães e ovelhas são inicializadas com variáveis. Os cães - cão líder, cão esquerdo e cão direito - são nomeados assim com base em suas posições. O cão líder controla o rebanho de frente. O indivíduo com melhor condicionamento físico (*fitf*) é, portanto, designado como o cão líder ou cão na frente do rebanho, em cada iteração. Eles são responsáveis principalmente pelo *gathering*.

Os indivíduos com os 2º e 3º melhores valores de fitness são escolhidos como cães esquerdo e direito. Um método de seleção de torneio é aplicado para escolher o cão esquerdo e direito. Esses cães principalmente participar na perseguição e observação do rebanho. Seus valores de fitness são referidos como (*fitle*) e (*fitri*), respectivamente. a população restante é composta por ovelhas, cujos valores de fitness são menores do que os dos cães. O fitness da ovelha é referido como (*fits*).

A posição dos cães é atualizada usando a equação básica da física de deslocamento.(1)

$$Pop_f(t+1) = V_f(t+1) \times Time_f(t+1) + \frac{1}{2}Acc_f(t+1) \times Time_f(t+1)^2 \quad (13)$$

$$Pop_{le}(t+1) = V_{le}(t+1) \times Time_{le}(t+1) + \frac{1}{2}Acc_{le}(t+1) \times Time_{le}(t+1)^2 \quad (14)$$

$$Pop_{ri}(t+1) = V_{ri}(t+1) \times Time_{ri}(t+1) + \frac{1}{2}Acc_{ri}(t+1) \times Time_{ri}(t+1)^2 \quad (15)$$

Figura 9: Posição dos cães

Vale ressaltar nesta equação que *Popf* se trata da posição do cão líder, enquanto que *Pople* e *Popri* se referem ao cão da esquerda e ao da direita respectivamente.

3 OBJETIVOS

O objetivo geral deste trabalho é propor o uso da meta-heurística *Border Collie Optimization* para resolver o *Problema do Caixeiro-Viajante* de maneira a visualizar se o algoritmo também é eficiente para resolver problemas discretos.

Os objetivos mais específicos envolvem a validação do algoritmo principal para problemas contínuos, a discretização do mesmo para o **PCV** e a comparação dos resultados obtidos.

4 MATERIAL E MÉTODOS

Nesta seção são apresentadas as configurações dos testes realizados, assim como os algoritmos das melhorias propostas.

Todos os testes realizados utilizaram a linguagem de programação **Python** na versão 3.9.7, juntamente com a ferramenta **Jupyter notebooks**.

4.0.1 IMPLEMENTAÇÃO DA HEURÍSTICA DO VIZINHO MAIS PRÓXIMO E BORDER COLLIE OPTIMIZATION

Antes de ser testada a heurística do Border Collie Optimization foi aplicado ao **PCV** a heurística do vizinho mais próximo e o vizinho mais próximo *k-opt*

Além disso, vale dizer que ambos os algoritmos foram testados no problema *bays29.tsp*, retirados do site tsplib95.

4.0.2 VIZINHO MAIS PRÓXIMO

O algoritmo do vizinho mais próximo foi, na ciência da computação, um dos primeiros algoritmos utilizados para determinar uma solução para o problema do caixeiro viajante. Se trata de uma heurística gulosa, que gera rapidamente um caminho curto mas geralmente não o ideal.(2)

A seguir estão os passos do algoritmo:

- Escolha um vértice arbitrário como vértice atual
- Descubra a aresta de menor peso que seja conectada ao vértice atual e a um vértice não visitado V;
- Faça o vértice atual ser V;
- Marque V como visitado;
- Se todos os vértices no domínio estiverem visitados, encerre o algoritmo;
- Se não vá para o passo 2.

Seguindo este algoritmo e, aplicando-o ao problema *bays29.tsp* foi obtido um valor de **2258**

4.0.3 VIZINHO MAIS PRÓXIMO 2-OPT

Já para o algoritmo 2-opt, elimina-se 2 arestas não adjacentes, reconecta-as usando duas outras arestas e verifica-se se houve melhora. Este processo é repetido para todos os pares de arestas. A melhor troca é então realizada.(3)

- Remover 2 arestas da solução H obtendo uma solução H';
- Construir todas as soluções viáveis contendo H';
- Escolher a melhor solução dentre as encontradas e guardar;
- Escolher outro conjunto de 2 arestas ainda não selecionado e retornar ao primeiro passo, caso contrário, pare.

Aplicando este algoritmo ao mesmo problema citado anteriormente, foi obtido o resultado de **2082**.

4.0.4 BORDER COLLIE OPTIMIZATION

Para a meta-heurística do Border Collie Optimization, foi utilizado o algoritmo proposto no artigo para os problemas contínuos.

Algorithm 1 Border Collie Optimization

Initialize
$Pop_i \rightarrow$ A random population of n individuals having d dimensions each, 3 dogs and $(n-3)$ sheep;
$Acc_i \rightarrow$ Random acceleration for each of the n individuals having d dimensions;
$Time_i \rightarrow$ Random time for each of the n individuals;
$V_i \rightarrow$ Zero velocity for n individuals having d dimensions;
$k = 0$;
while $t <$ *Max_Iterations* **do**
  Eyeing $= 0$
  $fit_t =$ Calculate fitness of n individuals
  if $fit_t < fit_{t-1}$ **then**
    $k = k + 1$
  end if
  if $k = 5$ **then**
    Eyeing $= 1$
    $k = 0$
  end if
  LeadDog = Individual with best fitness (fit_f))
  $R =$ *Random Number*[2, 3]
  if $R = 2$ **then**
    RightDog = Individual with 2^{nd} best fitness (fit_{ri})
    LeftDog = Individual with 3^{rd} best fitness (fit_{le})
  else
    LeftDog = Individual with 2^{nd} best fitness (fit_{le})
    RightDog = Individual with 3^{rd} best fitness (fit_{ri})
  end if
  Sheep = Rest of the individuals excluding top three (fit_s)
  Update velocity of dogs (using (1), (2) & (3))
  while $i >$ 3 *and* $i <= n$ **do**
    if *Eyeing* $= 1$ **then**
      Update velocity of sheep (using (9))
    else
      if $D_g > 0$ **then**
        Update velocity of sheep (using (5))
      else
        Update velocity of sheep (using (8))
      end if
    end if
  end while
  Update Acceleration of n individuals (using (11))
  Update Time of n individuals (using (12))
  Update Population of Dogs (using (13), (14) & (15))
  while $i >$ 3 *and* $i <= n$ **do**
    if *Eyeing* $= 1$ **then**
      Update Population of sheep (using (18))
    else
      Update Population of sheep (using (16) & (17))
    end if
  end while
end while

Figura 10: Border Collie Optimization

4.0.5 DISCRETIZAÇÃO

Para que fosse possível a utilização do algoritmo Border Collie Optimization para resolver o **PCV** foi necessário fazer a mudança no algoritmo para que este passasse a lidar com problemas discretos, esse processo se chama **discretização**.

Para este algoritmo a discretização se deu restrigindo os resultados para que estes ficassem limitados à quantidade de cidades existentes no problema. Ou seja, se temos um problema com 30 cidades, nossos valores a serem discretizados ficarão restritos ao intervalo de [0-29]. Também é válido dizer que os valores de custo de uma cidade para outra não sofrerão nenhuma mudança dentro do algoritmo e serão influenciados apenas pela rota gerada pelo própria metaheurística.

5 RESULTADOS E DISCUSSÃO

Nessa seção serão discutidos os resultados obtidos pelo Border Collie Optimization, tais como comparação com demais heurísticas, além de ser discutida a utilização do mesmo para resolver o ***Problema do Caixeiro Viajante***

5.0.1 BENCHMARK

As funções de *benchmark* são uma ótima maneira para testar a eficiência de um algoritmo, sendo estes testes feitos com diferentes parâmetros. Vale dizer que as funções utilizadas foram retiradas do artigo ***Grey Wolf Optimizer***.(4)

5.0.2 BENCHMARKS PARA PROBLEMAS CONTÍNUOS

Para que o problema fosse testado, foram utilizadas 8 funções de benchmark, sendo estas mostradas na tabela, seguidas da dimenção, intervalo e *fmin*, sendo o último o valor ideal para a função.

Tabela 1 – Funções de benchmark.

Function	**Dim**	**Range**	**fmin**
α	β	γ	δ
$\sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$\sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	30	$[-10, 10]$	0
$max_i\|x_i\|, 1 <= i <= n$	30	$[-100, 100]$	0
$\sum_{i=1}^{n}([x_i + 0.5)])^2$	30	$[-100, 100]$	0
$\sum_{i=1}^{n} ix_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0

$F_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ 30 [-50,50] 0

Figura 11: Função de benchmark

$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$ 4 [-5,5] 0.00030

Figura 12: Função de benchmark

5.0.3 PARÂMETROS PARA O BENCHMARK

- **Número de execuções:** 30
- **Número máximo de iterações:** 200
- **Tamanho da população:** 30

5.0.4 RESULTADOS BENCHMARK

Para medirmos o resultado foram utilizadas 4 métricas, sendo elas a **Optima** que representa o melhor valor dentre as 30 execuções, o pior valor dentre as 30 execuções, o valor mínimo da média e o desvio padrão.

```
Optima,Max,MinAverage,StandardDeviation

1.7576676458380436e-30,1.291901517125992,0.0478482043379997,0.24397892760983542
```

Figura 13: Resultados da primeira função de benchmark

```
Optima,Max,MinAverage,StandardDeviation

1.863811050002416e-27,13521613.35809433,466282.9934060994,2467225.531911579
```

Figura 14: Resultados da segunda função de benchmark

Dessa maneira, pode-se observar que o BCO obteve resultados bastante eficientes quando comparados com outros algoritmos como o ***Ant Colony Optimization*** e ***Particle Swarm Optimization***, como pode se notar na figura abaixo vemos os resultados de outros algoritmos de otimização utilizando as mesmas métricas citadas anteriormente.

ACO [3]	DE [16]	GA [11]	GWO [5]	HHO [58]	PSO [4]	WOA [52]
0.6865	0.1594	0.1215	8.7974e-09	7.8464e-42	2.5120e+04	1.6795e-26
0.2055	0.1307	0.0926	7.8904e-09	3.7385e-41	3.7996e+03	1.0894e-25
0.1774	0.0026	0.0023	5.5000e-10	2.2900e-56	1.7194e+04	9.3700e-36
1.1064	0.5430	0.4239	3.3100e-08	2.5200e-40	3.6666e+04	7.7100e-25
8.3400	0.0054	0.0030	0.0313	0.0265	0.0011	0.0121
1.9961	1.5333	1.0115	7.1530e-06	2.1364e-22	4.9682e+03	1.6111e-19
0.6610	0.7384	0.4304	3.3021e-06	4.8463e-22	3.2034e+04	5.6800e-19
0.8265	0.4643	0.0810	1.9500e-06	3.4900e-28	49.0195	1.2700e-23
4.8314	3.9380	1.9236	1.9500e-05	1.9900e-21	2.2669e+05	3.9400e-18

Figura 15: Resultados de outros algoritmos de otimização

5.0.5 RESULTADOS DISCRETIZAÇÃO

Enquanto que para os problemas contínuos o algoritmo do BCO se mostrou ser bastante eficiente, para o **PCV** por outro lado este se mostrou ter resultados não favoráveis quando comparado a outros algoritmos, como o do **vizinho mais próximo** e o **vizinho mais próximo 2-opt**. Diferentemente do primeiro teste de benchmark, para este tivemos como métrica apenas o valor mínimo (melhor rota) dentre as iterações do algoritmo e o valor máximo (pior rota), sendo estes valores **4300** e **6883** respectivamente. Assim, pod-se notar um desempenho pior quando comparado com os outros dois algoritmos citados acima, haja vista que o primeiro obteve um valor mínimo de **2258** e o segundo um valor mínimo de **2082**.

6 CONCLUSÃO

Neste trabalho, foi proposto o uso de uma nova meta-heurística para resolver o **Problema do Caixeiro-Viajante**, de forma a obter um resultado eficiente ainda não visto dentre as soluções já encontradas. Além de podermos fazer o uso desta heurística para resolver problemas discretos, o que abre a possibilidade do uso desse algoritmo para outros tipos de problemas.

A discretização do algoritmo se mostrou ineficiente, atigindo resultados não esperados e que não fora ótimos para a resolução do problema. Futuramente, pode-se pensar em resolver o **PCV** novamente, entretanto é necessário utilizar uma abordagem diferente para discretizar o problema.

Posto isso, apesar de não ter sido tirado resultados bons com a abordagem discreta, a abordagem contínua se mostrou ser eficiente, além de apresentar resultados melhores quando comparados a outros algoritmos de otimização. Por fim, trabalhos futuros poderiam focar em uma abordagem discreta diferente para outros tipos de problemas ñão-contínuos ou utilizar-se do BCO para resolver problemas contínuos.

REFERÊNCIAS

1 DUTTA, T. et al. Border collie optimization. **IEEE Access**, IEEE, v. 8, p. 109177–109197, 2020.

2 NN. <https://pt.wikipedia.org/wiki/Algoritmo_do_vizinho_mais_pr%C3%B3ximo>. (Accessed on 30/09/2022).

3 TSP. <https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf>. (Accessed on 30/09/2022).

4 MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. **Advances in engineering software**, Elsevier, v. 69, p. 46–61, 2014.