UNIVERSIDADE FEDERAL DO PARANÁ

IVAN LUCAS REIS SILVA

ALGORITMOS BASEADOS EM INTELIGÊNCIA DE ENXAMES APLICADOS À MULTILIMIARIZAÇÃO DE IMAGENS

CURITIBA

2018

IVAN LUCAS REIS SILVA

ALGORITMOS BASEADOS EM INTELIGÊNCIA DE ENXAMES APLICADOS À MULTILIMIARIZAÇÃO DE IMAGENS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Área de Concentração Sistemas Eletrônicos, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Leandro dos Santos Coelho

CURITIBA

2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

S586a Silva, Ivan Lucas Reis

Algoritmos baseados em inteligência de enxames aplicados à multilimiarização de imagens [recurso eletrônico] / Ivan Lucas Reis Silva – Curitiba, 2018.

Dissertação (mestrado) - Universidade Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica, Departamento de Elétrica no Setor de Tecnologia.
Orientador: Prof. Dr. Leandro dos Santos Coelho

1. Engenharia elétrica. 2. Processamento de imagem. I. Universidade Federal do Paraná. II. Coelho, Leandro dos Santos. III. Título.

CDD 621.367

Bibliotecária: Vilma Machado CRB9/1563

MINISTÉRIO DA EDUCAÇÃO
SETOR TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA ELÉTRICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **IVAN LUCAS REIS SILVA** intitulada: **Algoritmos baseados em inteligência de enxames aplicados à multilimiarização de imagens**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 20 de Agosto de 2018.

LEANDRO DOS SANTOS COELHO
Presidente da Banca Examinadora (UFPR)

ROBERTO ZANETTI FREIRE
Avaliador Externo (PUCPR)

LUIS HENRIQUE ASSUMPÇÃO LOLIS
Avaliador Interno (UFPR)

DEDICATÓRIA

À minha mãe, Ivone, que já se foi, mas continua sendo minha maior força e inspiração na vida.

Ao meu pai e irmão, Devanir e Iran, com quem compartilhei momentos de alegria, tristeza e ansiedade.

Aos meus amigos Silvio, Eldrey e Giovana, que sempre acreditaram em mim.

AGRADECIMENTOS

À Deus acima de tudo, pelo amor imenso e gratuito; pela vida, oportunidades e capacidades com as quais fui abençoado; por ter colocado pessoas iluminadas em meu caminho.

À minha família, especialmente ao meu pai e à minha mãe, por todo o apoio e encorajamento, assim como os sacrifícios feitos para que eu chegasse até aqui.

Ao meu orientador, o ProfessorLeandro dos Santos Coelho, pelo estímulo à pesquisa, por sua paciência e colaboração na elaboração da dissertação.

Aos membros da banca por terem gentilmente aceito o convite para participarem da avaliação deste trabalho.

RESUMO

O processamento de imagens é uma área que cresce à medida que as tecnologias de geração e armazenamento de informações digitais evoluem. Uma das etapas iniciais do processamento de imagem é a segmentação, onde a multilimiarização é uma das técnicas de segmentação mais simples. Um focorelevante de pesquisa nesta área é o projeto de abordagens visando a separação de diferentes objetos na imagem em grupos, por meio de limiares, para facilitar assim a interpretação da informação contida na imagem. Uma imagem perde informação, ou entropia, quando é limiarizada. A equação de limiarização multiníveis de Kapur calcula, a partir dos limiares escolhidos, qual a quantidade de informação que uma imagem apresentará após a limiarização. Assim, pela maximização da equação de multimiliarização de Kapur, é possível determinar os limiares que retornam uma imagem com valor maior de entropia. Quanto maior a quantidade de limiares, maior a dificuldade para encontrar a melhor solução, devido ao aumento significativo da quantidade de possíveis soluções. O objetivo desta dissertação é de apresentar um estudo comparativodecinco algoritmos de otimização (meta-heurísticas de otimização)da inteligência de enxame, incluindo Otimização por Enxame de Partículas (PSO), Otimização por Enxame de Partículas Darwiniano (DPSO), Otimização por Enxame de Partículas Darwiniano de Ordem Fracionária (FO-DPSO), Otimizador baseado no comportamento dos Lobos-cinza (GWO) e Otimizador inspirado no comportamento da Formiga-leão (ALO), de forma a avaliarqual deles obtém a melhor solução e convergência em termos da função objetivo relacionada a entropia da imagem. Uma contribuição desta dissertação é a aplicação de diferentes meta-heurísticas de otimização ao problema de multilimiarização de imagens, assim como o estudo do impacto das suas variáveis de controle (hiperparâmetros) para o problema em questão.Nesta dissertação são apresentados resultados paraquatro imagens diferentes, sendo duas imagens registradas por satélite (Rio Hunza e Yellowstone) e outras duas imagens teste (*benchmark*) obtidas do Centro de Engenharia Elétrica e Ciência da Computação do MIT (Massachussetts Institute of Technology). Os resultados são comparados considerando a média e o desvio padrão da entropia de cada imagem resultante. Com base nos resultados obtidos conclui-se que o algoritmo mais indicado para o problema de multilimiarização de imagens dos avaliados é o GWO, pelo seu desempenho superior em relação aos outros algoritmos e pelas entropias das imagens resultantes serem satisfatórias.

Palavras-chave: Segmentação de imagens. Multilimiarização. Inteligência de enxames. Otimização por enxame de partículas. Otimizador dos lobos-cinza. Otimizador formiga-leão.

ABSTRACT

Image processing is a field that grows as digital information storage and generation technologies evolution. One of the initial stages of image processing is segmentation procedure, where the multi level thresholding is one of the simplest segmentation approaches. A relevant research objective in this field is the design of approaches aimed at separating different objects in the image into groups, through thresholds, to facilitate the interpretation of the information contained in the image. An image loses information, or entropy, when it is thresholded. The Kapur multilevel thresholding equation calculates, from the chosen thresholds, how much information an image will present after the thresholding. Thus, by the maximization of the Kapur multilevel limiarization equation, it is possible to determine the thresholds that return an image with a larger value of entropy. The higher the amount of thresholds, the greater the difficulty in finding the best solution, due to the significant increase in the quantity of possible solutions. The objective of this dissertation is to present a comparative study between fiveoptimization metaheuristics of the swarm intelligence field, including Particle Swarm Optimization (PSO), Darwinian Particle Swarm Optimization (DPSO), Fractional Order Darwinian Particle Swarm Optimization (FO-DPSO), Grey Wolf Optimizer (GWO) and the Ant lion behavioral optimizer (ALO), in order to identify which one gets the best solution and convergence in terms of the objective function and the entropy of the image. A contribution of this dissertation is the application of different optimization metaheuristics to the problem of multilimizing of images, as well as the study of the impact of its control variables (hyperparameters) on the problem in question. Experiments are conducted with four images, two images being recorded by satellite (Hunza River and Yellowstone) and two other test(benchmark) images obtained from MIT's (Massachussetts Institute of Technology) Electrical Engineering and Computer Science Center. The results are compared considering the mean and standard deviation values of each resulting image entropy.Based on the results obtained it is concluded that the most suitable algorithm for the problem of multilevel thresholding of images is the GWO, for its superior performance in relation to the other tested algorithms and satisfactory entropies of the resulting images.

Key-words: Image segmentation. Multilevel thresholding. Kapur's entropy. Swarm intelligence. Particle swarm optimization. Grey wolf optimizer. Ant lion optimizer.

LISTA DE ILUSTRAÇÕES

Figura 1-Etapas do processamento de imagem 23
Figura 2-Imagem representada por uma matriz *f(x,y)* 24
Figura 3- Detalhe ampliado de uma imagem 25
Figura 4 - Histograma de imagem 26
Figura 5 - Imagem A com entropia zero e imagem B com entropia máxima 28
Figura 6- Histograma de níveis de cinza, divididos por um limiar único *T* 30
Figura 7- Histograma particionado em múltiplos limiares 31
Figura 8- Comportamento da caça dos lobos cinzentos 52
Figura 9 - Vetores de posição 2D e suas possíveis localizações subsequentes. 54
Figura 10 - Vetores de posição 3D e suas possíveis localizações seguintes. 55
Figura 11- Atualização de posição no GWO 56
Figura 12 - Formiga-leão em sua forma adulta 59
Figura 13 - Larva da formiga-leão 60
Figura 14 - Armadilha construída pela larva da formiga-leão 61
Figura 15 - Caminhada aleatória de uma formiga dentro de uma armadilha de formiga-leão 65
Figura 16 - Limites superior (c) e inferior (d) adaptativos 67
Figura 17 - Imagem Barcos 79
Figura 18 - Histograma da imagem 'Barcos' 80
Figura 19- Imagem Bárbara 80
Figura 20 - Histograma da imagem 'Bárbara' 80
Figura 21- Imagem Rio Hunza 81
Figura 22 - Histograma da imagem 'Rio Hunza' 81
Figura 23 - Imagem Yellowstone 82
Figura 24 - Histograma da imagem 'Yellowstone' 82
Figura 25 - Tempo de execução para diferentes quantidades de agentes do algoritmo ALO 85
Figura 26 - Entropia resultante para diferentes quantidades de agentes do algoritmo ALO 86
Figura 27 - Tempo de execução para diferentes quantidades de gerações do algoritmo ALO 87

Figura 28 - Entropia resultante para diferentes quantidades de gerações do algoritmo ALO 87
Figura 29 - Tempo de execução para diferentes quantidades de agentes do algoritmo GWO 88
Figura 30 - Entropia resultante para diferentes quantidades de agentes do algoritmo GWO 89
Figura 31 - Tempo de execução para diferentes quantidades de gerações do algoritmo GWO 90
Figura 32 - Entropia resultante para diferentes quantidades de gerações do algoritmo GWO 90
Figura 33 - Tempo de execução para as diferentes populações iniciais do algoritmo PSO 91
Figura 34 - Entropia das imagens resultantes para as diferentes populações iniciais do algoritmo PSO 92
Figura 35 - Tempo de execução para os diferentes pesos individuais utilizados pelo algoritmo PSO 92
Figura 36- Entropia das imagens resultantes para os diferentes pesos individuais utilizados pelo algoritmo PSO 93
Figura 37 - Tempo de execução para os diferentes pesos sociais utilizados pelo algoritmo PSO 93
Figura 38- Entropia das imagens resultantes para os diferentes pesos sociais utilizados pelo algoritmo PSO 94
Figura 39 - Tempo de execução para os diferentes fatores de inércia utilizados pelo algoritmo PSO 95
Figura 40- Entropia das imagens resultantes para os diferentes fatores de inércia utilizados pelo algoritmo PSO 95
Figura 41 - Tempo de execução para as diferentes populações mínimas do algoritmo DPSO 97
Figura 42- Entropia das imagens resultantes para as diferentes populações mínimas do algoritmo DPSO 97
Figura 43 - Tempo de execução para as diferenças entre populações mínimas e máximas do algoritmo DPSO 98
Figura 44- Entropia das imagens resultantes para as diferenças entre populações mínimas e máximas do algoritmo DPSO 98

Figura 45 - Tempo de execução para as diferentes enxames iniciais do algoritmo DPSO 99
Figura 46- Entropia das imagens resultantes para as diferentes enxames iniciais do algoritmo DPSO 100
Figura 47 - Tempo de execução para as diferentes enxames mínimos do algoritmo DPSO 100
Figura 48- Entropia das imagens resultantes para as diferentes enxames mínimos do algoritmo DPSO 101
Figura 49 - Tempo de execução para as diferenças entre enxames mínimos e máximos do algoritmo DPSO 102
Figura 50- Entropia das imagens resultantes para as diferenças entre enxames mínimos e máximos do algoritmo DPSO 102
Figura 51- Tempo de execução para as diferentes valores de estagnação do algoritmo DPSO 103
Figura 52- Entropia das imagens resultantes para as diferentes valores de estagnação do algoritmo DPSO 103
Figura 53 – Tempo de execução para as diferentes quantidades de gerações do algoritmo DPSO 104
Figura 54- Entropia das imagens resultantes para as diferentes quantidades de gerações do algoritmo DPSO 104
Figura 55- Tempo de execução para os diferentes coeficientes fracionários do algoritmo FO-DPSO 106
Figura 56- Entropia das imagens resultantes para os diferentes coeficientes fracionários do algoritmo FO-DPSO 107
Figura 57- Tempo de execução para os diferentes distâncias mínimas entre picos para o algorítmo de força-bruta 108
Figura 58- Entropia das imagens resultantes para os diferentes distâncias mínimas entre picos para o algorítmo de força-bruta 109
Figura 59- Tempo de execução para os diferentes janelas de suavização para o algorítmo de força-bruta 110
Figura 60- Entropia das imagens resultantes para os diferentes janelas de suavização para o algorítmo de força-bruta 110

LISTA DE TABELAS

Tabela 1 - Sumarização da literatura 41
Tabela 2 - Tempos de execução (em segundos) para a imagem Barbara..... 112
Tabela 3 - Tempos de execução (em segundos) para a imagem Barcos 112
Tabela 4 - Tempos de execução (em segundos) para a imagem Rio Hunza..... 112
Tabela 5 - Tempos de execução (em segundos) para a imagem Yellowstone 113
Tabela 6 - Entropias resultantes para a imagem Barbara 113
Tabela 7 - Entropias resultantes para a imagem Barcos..... 114
Tabela 8 - Entropias resultantes para a imagem Rio Hunza 114
Tabela 9 - Entropias resultantes para a imagem Yellowstone..... 114

LISTA DE SIGLAS

ACO - *Ant Colony Optimization*

ACOBSPPM -*Ant-Colony Optimization Based Binary Search Point Matching*

BA -*Bat Algorithm*

CHBMA -*Cooperative Honey Bee Mating Algorithm*

DE- *Differential Evolution*

DPSO -*Darwinian Particle Swarm Optimization*

FA - *Firefly Algorithm*

FASSO -*Fuzzy Adaptive Swallow Swarm Optimization*

FO-DPSO -*Fractional-Order Darwinian Particle Swarm Optimization*

GA -*Genetic Algorithm*

GOA - *Grasshopper Optimization Algorithm*

GWO - *Grey Wolf Optimizer*

HBMA -*Honey Bee Mating Algorithm*

HS - Hirschberg-Sinclair*Algorithm*

IGP -*Interior Gateway Protocol*

MFO -*Moth-Flame Optimization*

MRI - *Magnetic Resonance Imaging*

MSSIM - *Mean Structural Similarity*

PSO - *Particle Swarm Optimization*

PSOGSA -*Particle Swarm Optimization Gravitational Search Algorithm*

SCA -*Sine Cosine Algorithm*

SSO -*Simplified Swarm Optimization*

STA -*State Transition Algorithm*

SVM -*Support Vector Machine*

WOA - *Whale Optimization Algorithm*

SUMÁRIO

1. INTRODUÇÃO 16

1.1 PROBLEMATIZAÇÃO 18

1.2 OBJETIVOS 19

1.2.1 Objetivo Geral 19

1.2.2 Objetivos específicos 19

1.3 JUSTIFICATIVA 20

1.4 METODOLOGIA DE PESQUISA 21

1.5 ESTRUTURA DO DOCUMENTO 21

2. PROCESSAMENTO DIGITAL DE IMAGEM 22

2.1 FUNDAMENTOS DE IMAGENS DIGITAIS 23

2.1.1 Etapas fundamentais do processamento de imagem 23

2.1.2 Modelo de imagens 24

2.1.3 Amostragem e quantização 25

2.1.4 Histograma da imagem 26

2.1.5 Entropia em imagens 27

2.2Segmentação de imagens 28

2.2.1 Limiarização 29

2.2.2 Limiarização por entropia 31

3. REVISÃO DA LITERATURA 35

4. META-HEURÍSTICAS DE INTELIGÊNCIA DE ENXAME 43

4.1 FUNDAMENTOS DE INTELIGÊNCIA DE ENXAME 45

4.1.1Princípios biológicos da inteligência de enxame 46

4.1.2 Processo de auto-organização de um enxame 47

4.1.3 Categorização de comportamentos 48

4.1.4 Modulação de comportamentos auto-organizados 49

4.2 ALGORITMOS DE INTELIGÊNCIA DE ENXAME 50
4.2.1 Otimizador do lobo cinzento (*Grey Wolf Optimizer*, GWO) 50
4.2.2 Otimizador formiga-leão (*Ant Lion Optimizer*, ALO) 58
4.2.3 Algoritmo de otimização por enxame de partículas (*Particle Swarm Optimization*, PSO) 70
4.2.4 Algoritmo de otimização por enxame de partículas Darwiniano (*Darwinian Particle Swarm Optimization*, DPSO) 74
4.2.5 Algoritmo de otimização por enxame de partículas Darwiniano de ordem fracionária (*Fractional-Order Darwinian Particle Swarm Optimization,* FO-DPSO) 77
5. IMAGENS DE TESTE 79
6. ANÁLISE DE RESULTADOS 84
6.1 DETERMINAÇÃO DOS PARÂMETROS DE CONTROLE 84
6.1.1 Otimizador formiga leão (ALO) 84
6.1.2 Otimizador lobo cinzento (GWO) 88
6.1.3 Otimizador por enxame de partículas (PSO) 91
6.1.4 Otimizador por enxame de partículas Darwiniano (DPSO) 96
6.1.5 Otimizador por enxame de partículas Darwiniano de ordem fracionária (FO-DPSO) 105
6.1.6 Otimizador por força-bruta 108
7. COMPARAÇÃO ENTRE META-HEURÍSTICAS DE OTIMIZAÇÃO E ALGORITMO FORÇA-BRUTA 112
8. CONCLUSÃO E FUTURA PESQUISA 115
REFERÊNCIAS 117

1. INTRODUÇÃO

Diversos métodos são sugeridos na literatura para o processo de segmentação de imagens, tais como (CHEN *et al.*, 2018;WANG, B.; CHEN; CHENG, 2018). Um método simples e eficaz que vem sendo amplamente utilizado é o de limiarização e multilimiarização (MERZBAN; ELBAYOUMI, 2019). A segmentação de imagens é uma etapa relevante no processamento de imagens, pois uma segmentação ineficiente pode comprometer os próximos passos do processamento, como, por exemplo, a interpretação dos dados contidos na imagem.

A multilimiarização ou limiarização multinível é uma técnica que visa separar os diferentes objetos contidos na imagem. Entende-se que quanto melhor segmentados estão os objetos de interesse em uma imagem, mais informações essa imagem transmite. Para medir a quantidade de informação contida em uma imagem é utilizada a entropia da mesma, logo os limiares da multilimiarização que resultem em uma segmentação adequada podem ser obtidos quando a entropia da imagem é maximizada.A entropia é uma medida de desordem ou uma medida de informações heterogêneas e é sabido que a quantidade de informação não pode ser negativa (KARCI, 2016).

Multilimiarização é considerado um problema para o qual as abordagens clássicas de métodos de otimização pode ser complexas de aplicar com eficiência. Sendo assim, o emprego de *meta-heurísticas*se torna recomendado (HINOJOSA *et al.*, 2018). As meta-heurísticas podem obter soluções para problemas de otimização computacionalmente intratáveis obtidas por meio de métodos exatos, cuja formulação matemática não é completamente conhecida ou cujos requisitos são difíceis de serem atendidos usando abordagens convencionais de otimização, em contrapartida seus resultados não são necessariamente ótimos. O termo *heurística* deriva da palavra grega *heuristiken*, que significa encontrar ou descobrir. As *meta-heurísticas* buscam por soluções subótimas de boa qualidade, visando obter um compromisso entre a qualidade das soluções obtidas e o custo computacional para exploração do espaço de busca. No entanto, as meta-heurísticas podem não apresentar garantia de factibilidade e/ou otimalidade das soluções encontradas.

Uma meta-heurística pode ser definida como um ou mais heurísticascombinadas de forma a operar em sinergia na busca por soluções promissoras. As meta-heurísticas são geralmente aplicadas a problemas para os quais não existe um algoritmo capaz de resolver o problema ou quando estes requerem acentuadas quantidades de recursos computacionais, tornando inviável sua aplicação prática.

Uma vez que amaximização da função objetivo da entropia na multilimiarização é um problema cuja solução analítica é inviável(HINOJOSA *et al.*, 2018), o que justifica necessidade de utilização de meta-heurísticas de otimização em sua resolução. Exemplos de meta-heurísticas de otimização são os algoritmos bio-inspirados de inteligência de enxames, que emulam características da inteligência coletiva e social na resolução de problemas (ERTENLICE; KALAYCI, 2018, MAVROVOUNIOTIS; LI; YANG, 2017;KOLIAS; KAMBOURAKIS; MARAGOUDAKIS, 2011).

A Inteligência de enxames, é um conjunto de técnicas inspiradas no comportamento coletivo de sistemas auto-organizados, distribuídos, autônomos, flexíveis e dinâmicos. Estes sistemas são formados por uma população de agentes computacionais simples que possuem a capacidade de perceber e modificar o seu ambiente de maneira local. Esta capacidade torna possível a comunicação entre os agentes, que captam as mudanças no ambiente geradas pelo comportamento de seus congêneres. Embora não exista uma estrutura centralizada de controle que estabelece como os agentes devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre os agentes geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema (SERAPIÃO, 2009).

O algoritmo de Otimização por Enxame de Partículas (*do inglês,Particle Swarm Optimization* (PSO) (KENNEDY; EBERHART, 1995) é uma meta-heurística inspirada nos processos sócio-cognitivos observados e empregados em modelos abstratos da inteligência e aprendizado coletivo. O PSO, desde sua inicial proposição, têm sido influenciado pelos conceitos da computação evolucionária.

Existem outras abordagens de PSO que podem ser eficientes para problemas de otimização global. O algoritmo de Otimização por Enxame de Partículas Darwiniano (*do inglês, Darwinian Particle Swarm Optimization,* DPSO) (TILLETT; RAO; *et al.*,

2005) foi concebido visando uma cálculo paralelo utilizando o PSO, na tentativa de melhorar a sua velocidade de convergência e qualidade de soluções obtidas.

O algoritmo de Otimização por Enxame de Partículas Darwiniano de ordem fracionada (*do inglês, Fractional-Order Darwinian Particle Swarm Optimization*, FO-DPSO)(COUCEIRO*et al.*, 2012) é uma extensão do DPSOna qual o cálculo fracionário é utilizado para controlar a taxa de convergência do algoritmo. O algoritmo DPSO, embora apresente um desempenho superior quando comparado com o PSO para muitos problemas teste (*benchmarks*) de otimização global no domínio contínuo, apresenta também a desvantagem de ser mais complexo computacionalmente, O FO-DPSO por sua vez controla a taxa de convergência do DPSO utilizando cálculo fracionário (SOLTEIRO PIRES *et al.*, 2010).

O Otimizador do Lobo Cinzento (*Grey Wolf Optimizer* (GWO))(MIRJALILI; MIRJALILI; LEWIS, 2014)é baseado no comportamento social e hierárquico apresentado na alcatéia dos Lobos cinzentos (*Canis Lupus*). Quatro tipos de lobos são utilizados na simulação de liderança e hierarquia, sendo eles *alpha, beta, delta* e *ômega.* Adicionalmente, são implementados os três passos da caça ao alimento, sendo tais passos: procura pela presa, cerco sobre a presa e o ataque em si.

O Otimizador Formiga-Leão (*Ant Lion Optimizer*, ALO)(MIRJALILI, 2015) é inspirado no comportamento de caça das larvas de Formiga-leão, baseado na construção de armadilhas para atrair a presa. Os passos principais da caça são implementadosincluindo a movimentação aleatória das formigas, a construção de armadilhas, o aprisionamento das formigas nas armadilhas, captura da presa e a reconstrução das armadilhas.

1.1 PROBLEMATIZAÇÃO

Muitas meta-heurísticas têm sido avaliadas quanto ao seu desempenho em encontrar solução para problemas de otimização global (DAS*et al.*, 2011) e (SALCEDO-SANZ, 2016). O algoritmo de Otimização por Enxames de Partículas (PSO) foi proposto em (KENNEDY; EBERHART, 1995) e desde então pesquisadores procuram testá-lo em diferentes problemas e também tentam propor melhoramentos ao algoritmo, entre eles estão os algorítmos de Otimização por Enxames de Partículas Darwiniano (DPSO) e Otimização por Enxames de Partículas Darwiniano de Ordem Fracionária (FO-DPSO), publicados em (TILLETT; RAO*et al.*,

2005 e COUCEIRO *et al.*, 2012) respectivamente. O Otimizador do Lobo Cinzento (GWO) e o Otimizador Formiga-leão (ALO) são algoritmos que foram publicados recentemente em (MIRJALILI; MIRJALILI; LEWIS, 2014; MIRJALILI, 2015), respectivamente.

A maximização da função objetivo da limiarização multinível por entropia de imagens é desafiadora e pode ser considerada como um apropriado problema de otimização global para testar o desempenho desses algoritmos. Quanto maior o número de limiares na limiarização, mais complexa esse tipo de segmentação de imagem se torna.

O problema considerado nestadissertação é a maximização da função objetivo da limiarização multinível por entropia de imagens, para gerar uma segmentação em que os objetos de interesse na imagem estejam adequadamente separados.

1.2 OBJETIVOS

A seguir são mencionados os objetivo geral e os objetivos específicos desta dissertação.

1.2.1 Objetivo Geral

O objetivo geral desta dissertação é comparar o desempenho de diferentes meta-heurísticas de otimização baseadas em inteligência de enxames visando a multilimiarização baseada em informação de entropia de imagens.

1.2.2 Objetivos específicos

Os objetivos específicos desta dissertação são os seguintes:

- Projetar, implementar e validar o algoritmo de otimização baseado em Enxame de Partículas e suas variantes Darwiniana e de Ordem Fracionária;
- Projetar, implementar e validar o algoritmo de otimização baseado no comportamento dos Lobos Cinzentos;
- Projetar, implementar e validar o algoritmo de otimização baseado no comportamento da Formiga Leão;
- Aplicar as meta-heurísticasde otimização implementadas ao estudo de caso de multilimiarização de imagem;

- Definir os parâmetros de controle (hiperparâmetros) das meta-heurísticas de otimização utlizadas, de forma que seja possível a sua comparação e análise do desempenhodas mesmas em termos de convergência e qualidade das soluções obtidas;
- Medir o desempenho dos algoritmos no método de segmentação escolhido, em termos de tempo computacional e entropia das imagens resultantes, por meio de análise de desempenho frente a problemasteste apresentados na literatura recente.

1.3 JUSTIFICATIVA

A segmentação de imagem vem apresentando crescimento significativo e é utilizada em vários ramos do conhecimento. Tanto que, por exemplo, possui aplicação em diagnóstico de doenças, fotointerpretação na agricultura, reconhecimento de faces em imagens, controle automático de tráfego, sistemas de visão computacional para veículos autônomos e identificação de placas de veículos em equipamentos de monitoramento de trânsito, entre outros. Os modelos computacionais clássicos de segmentação de imagem não são, muitas vezes, suficientes na resolução de problemas nas mais diversas aplicações em que a segmentação de imagens pode atuar. Logo, há a necessidade de busca de soluções e alternativas mais eficazes. Nesse contexto, a utilização de meta-heurísticas de otimização vem sendo cada vez mais comum, com a geração de modelos que se comportem de forma inspirada em modelos inteligentes presentes na natureza. Uma inspiração no projeto de meta-heurísticas de otimização é o estudo de algoritmos de otimização de inteligência de enxames, que são algoritmos que possuem características de comportamento coletivo e social e que atuam na otimização de soluções para muitas áreas do conhecimento (MAVROVOUNIOTIS; LI; YANG, 2017).

Conforme mencionado, esta dissertação aborda cinco meta-heurísticas de otimização baseadas em inteligência de enxames e avalia seus desempenhos em segmentar as imagens teste escolhidas. O desempenho das meta-heurísticas de otimização é avaliado levando em consideração tantoo tempo de execução quanto a entropia das imagens resultantes. Essadissertação pode ser útil para determinar uma solução mais eficiente, em termos de desempenho computacional e entropia da imagem resultante,para a multilimiarização de imagens, apresentando evidências em favor da meta-heurística escolhida. Essa dissertação pode também ser

considerada um elo dentro da série de estudos sobre a utilização de meta-heurísticas de otimização aplicadas ao problema de multilimiarização de imagem.

1.4 METODOLOGIA DE PESQUISA

A metodologia de pesquisa pode variar conforme sua natureza. Como este trabalho tem o objetivo de comparar desempenho de meta-heurísticas de otimização em segmentação de imagem por limiarização multinível por entropia, ou seja, procura gerar conhecimentos para aplicação prática dirigida à solução de um problema específico e não universal, a natureza da pesquisa é aplicada.

Ela também pode ser considerada quantitativa, pois as imagens utilizadas para segmentação são digitalizadas por quantização e amostragem e são convertidas em matrizes de números, que são conjuntos de informações que podem ser classificados e analisados.

1.5 ESTRUTURA DO DOCUMENTO

O restante desta dissertação está organizado da seguinte forma.O capítulo 2 apresenta uma revisão da literatura sobre segmentação e limiarização de imagens e também sobre os algoritmos utilizados.

Depois, o capítulo 3 apresenta alguns conceitos fundamentais de processamento de imagem e também esclarece a respeito de segmentação de imagens e limiarização por entropia.O capítulo4 expõe algumas características das meta-heurísticas utilizadas, explica resumidamente a base dos algoritmos inspirados em inteligência de enxames e ainda apresenta os conceitos dos algoritmos escolhidos para a segmentação de imagem.

O capítulo5 apresenta as imagens teste utilizadas e os seus histogramas correspondentes.O capítulo 6 detalha os resultadosda busca pelos melhores parâmetros de controle dos algoritmos abordados da inteligência de enxames.

O capítulo7 apresenta a comparação entre os resultados obtidos a partir de meta-heurísticase os resultados obtidos a partir do algoritmo de força-bruta.O capítulo 8 é dedicado à conclusões obtidasnesta dissertação e à pesquisa futura.

2. PROCESSAMENTO DIGITAL DE IMAGEM

Por Processamento Digital de Imagens (PDI) entende-se a manipulação de uma imagem por computador de modo que a entrada e a saída do processo sejam imagens. O objetivo de se usar processamento digital de imagens é melhorar o aspecto visual de certas feições estruturais para o analista humano e fornecer outros subsídios para a sua interpretação, inclusive gerando produtos que possam ser posteriormente submetidos a outros processamentos (CÂMARA *et al.*, 1996). O resultado desse processo é a produção de outras imagens, estas já contendo informações específicas, extraídas e realçadas a partir das imagens "brutas".

A informação de interesse é caracterizada em função das propriedades dos objetos ou padrões que compõem a imagem. Portanto, extrair informação de imagens envolve o reconhecimento de objetos ou padrões. A maior parte dessa atividade requer aprimoradacapacidade de cognição por parte do intérprete, devido à complexidade dos processos envolvidos e à falta de algoritmos computacionais precisos o suficientepara realizá-lo de forma automática.

O sistema visual humano possui uma notável capacidade de reconhecer padrões. Contudo, ele dificilmente é capaz de processar o "enorme" volume de informação presente numa imagem. Vários tipos de degradações e distorções, inerentes aos processos de aquisição, transmissão e visualização de imagens, contribuem para limitar ainda mais essa capacidade do olho humano.

Um dos focos das pesquisas na área de processamento de imagens é o de remover essas barreiras, inerentes ao sistema visual humano, facilitando a extração de informações a partir de imagens.

Nas próximas seções deste capítulo são apresentados alguns fundamentos do processamento de imagens e o tipo de segmentação adotada nestadissertação.

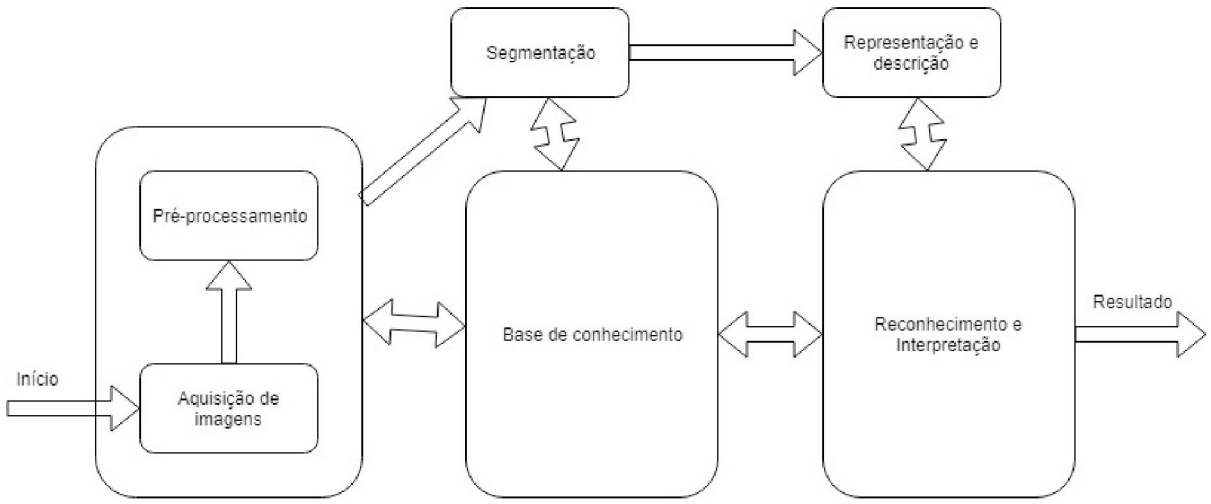
2.1 FUNDAMENTOS DE IMAGENS DIGITAIS

Alguns fundamentos do processamento de imagens serão discutidos nesta seção, pois são base para o entendimento da segmentação proposta.

2.1.1 Etapas fundamentais do processamento de imagem

As etapas fundamentais do processamento de imagem estão apresentadas na Figura 1.

Figura 1-Etapas do processamento de imagem



Fonte:Adaptado de (PEDRINI; SCHWARTZ, 2008), p. 5.

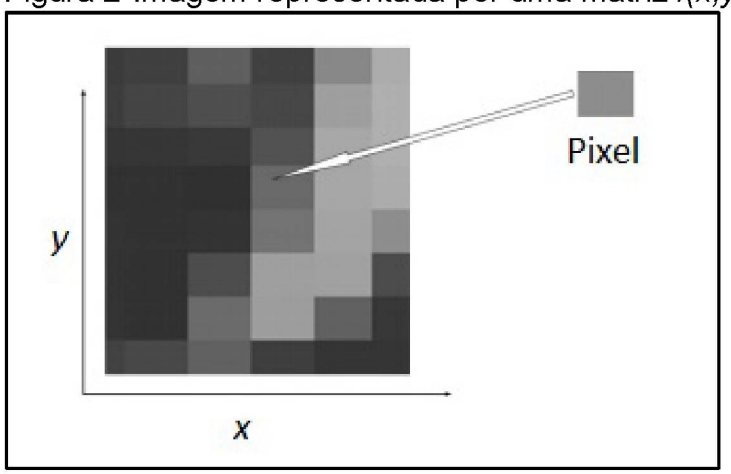
A aquisição de imagens é realizada por meio de um sensor de imageamento, que poderia ser uma câmera de televisão, por exemplo, resultando em uma imagem digital. O pré-processamento é o passo em que a imagem é tratada de forma a se aumentarem as chances de as etapas seguintes serem realizadas. Um exemplo de pré-processamento é a remoção de ruídos por meio de aplicação de filtros e o realce de contrastes. A segmentação é a divisão da imagem em partes menores que sejam objetos que tenham alguma relevância para a determinada aplicação. Essa etapa será detalhada na seção 2.2. A representação é a ação de representar uma imagem de uma forma que seja conveniente para o subsequente processamento

computacional. O processo de descrição procura extrair características da imagem que resultem em informações quantitativas, normalmente para classificação de objetos. O reconhecimento rotula um objeto de relevência na imagem e a interpretação atribui significado a um conjunto de objetos (GONZALEZ e WOODS, 2000;PEDRINI e SCHWARTZ, 2007).

2.1.2 Modelo de imagens

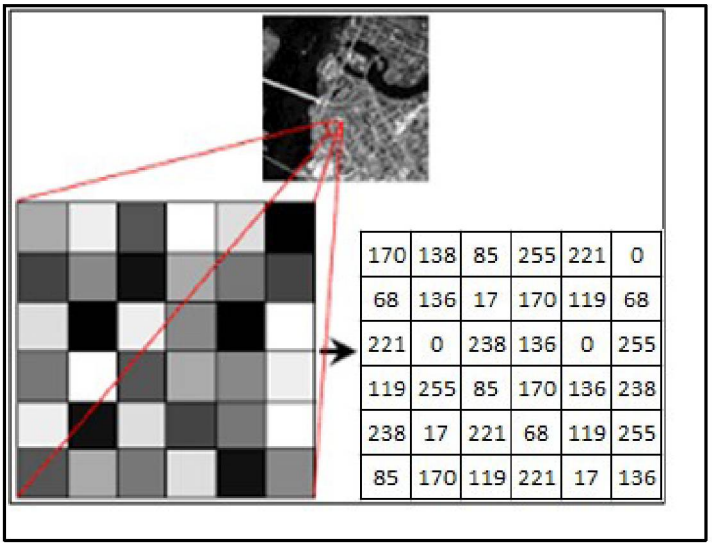
SegundoRUSS (2002) uma imagem monocromática, com somente tons de cinza, pode ser representada por uma função de níveis de cinza ou de intensidade luminosa, definida como $f(x, y)$. A amplitude das coordenadas (x, y) fornece o brilho, ou a intensidade da imagem naquele ponto. Como mostra a Figura 2, uma matriz bidimensional também pode ser utilizada para representar uma imagem, em que cada elemento (x, y) da matrizrepresenta um *pixel* da imagem. Cada *pixel* da imagem contém o valor de intensidade ou nível de cinza igual a $f(x, y)$ correspondente. A Figura 3 mostra um detalhe de uma imagem aérea, na qual cada *pixel* contém seu respectivo nível de cinza, e apresenta a matriz correspondente do detalhe ampliado, cujos elementos são os valores da intensidade de cada *pixel*.

Figura 2-Imagem representada por uma matriz *f(x,y)*



Fonte: Tecnologia radiológica, 2015.

Figura 3- Detalhe ampliado de uma imagem



Fonte: Wordpress, 2015.

2.1.3 Amostragem e quantização

SegundoPEDRINI eSCHWARTZ (2007), para que a função $f(x, y)$, apresentada na seção anterior, possa ser tratada por processamento computacional, ela precisa ser digitalizada tanto em amplitude como espacialmente. A digitalização espacial é a amostragem, na qual são escolhidos o número de linhas e colunas da matriz que representará a imagem. A de amplitude é a quantização, que também pode ser denominada de quantização em níveis de cinza.

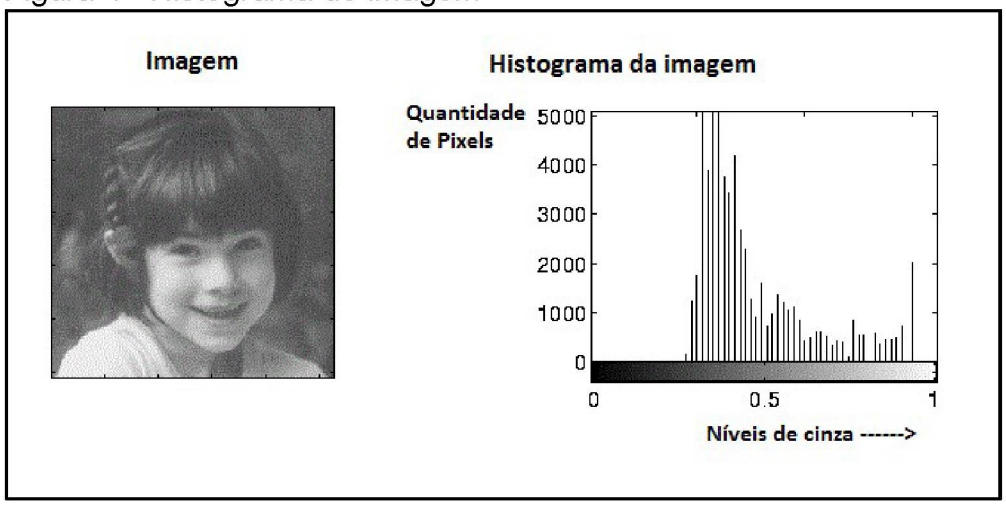
Normalmente, a qualidade da imagem se dá pela amostragem com que é realizada. Quanto maior o número de linhas e colunas da matriz que representa a imagem, maior é a qualidade. O tamanho da matriz é comumente conhecido como resolução da imagem. Logo quanto maior a resolução da imagem, maior é a qualidade. A amostragem insuficiente de uma imagem pode fazer com que os contornos presentes na imagem tenham aparência serrilhada (GONZALEZ; WOODS, 2000).

Quanto maior a quantização, ou quanto maior a quantidade de níveis de cinza para representar uma imagem, mais fiel é a representação da imagem digital com relação à imagem real. Uma imagem com quantização insuficiente pode gerar falsos contornos, que em uma representação da imagem com quantização suficiente, estariam em níveis de cinza distintos. Como a imagem está insuficientemente quantizada, esses contornos acabam se tornando um contorno único, que é diferente do contorno presente na imagem real (PEDRINI; SCHWARTZ, 2007).

2.1.4 Histograma da imagem

O histograma é a forma mais comum de se representar a distribuição dos tons de cinza em uma imagem. A representação é feita por meio de um gráfico em que o eixo das abcissas são os níveis de cinza e o das ordenadas são quantidades de *pixels*. O histograma é a relação entre intensidade e quantidade de *pixels*, ou seja, cada ponto do gráfico representa a quantidade, em toda a imagem, de *pixels* que contêm a intensidade da abcissa do ponto da curva (GONZALEZ; WOODS, 2000). A Figura 4 mostra uma imagem e seu histograma.

Figura 4 - Histograma de imagem



Fonte:Adaptado de (HOMEPAGES, 2015).

Uma imagem possui um único histograma, porém um histograma não representa somente uma imagem. Geralmente, diferentes imagens podem ser representadas pelo mesmo histograma. O histograma ainda pode ser considerado como uma distribuição de probabilidades discreta. Nesse caso, o número de *pixels* para uma determinada intensidade é usado para calcular a probabilidade de se encontrar um *pixel* com aquele valor de intensidade. Assim o histograma $p(f)$ pode ser definido como

$$p(i) = \frac{n_i}{n} \tag{2.1}$$

onde n_i corresponde ao número de ocorrências da intensidade i e n é o número total de *pixels* na imagem f (GONZALEZ; WOODS, 2000; PEDRINI; SCHWARTZ, 2007).

2.1.5 Entropia em imagens

SHANNON (1948) introduziu a Teoria da Informação, que apresenta um novo modelo matemático para sistemas de comunicação. A intenção do pai da teoria da informação era descobrir as leis que governam os sistemas usados em comunicação e manipulação da informação. Seu objetivo era também definir, em medidas quantitativas, a informação e a capacidade de sistemas de transmitir, amazenar e processar informação. Um dos frutos de seus estudos foi o princípio fundamental da teoria da informação que estabelece que a geração de informação pode ser modelada como um processo probabilístico.

Em seu trabalho, Shannonfoi quem difundiu primeiramente o conceito de entropia na informação, como uma forma de medição quantitativa da informação, baseando-se na expressão de entropia de Boltzmann (1896) da termodinâmica. Para Shannon, a quantidade de informação transmitida por um canal de comunicação é inversamente relacionada com a previsibilidade da mensagem. Uma moeda com duas caras, por exemplo, não transmite informação, pois a mensagem originada pelo lançamento da moeda é sempre cara, a probabilidade é 1, ou 100%, não há incerteza (GONZALEZ; WOODS, 2000).

Uma imagem pode ser considerada como o resultado de um processo estocástico. A probabilidade p_i correspondeà probabilidade de um *pixel* assumir uma intensidade, ou nível de cinza, i, tal que $i = 0, 1, \ldots, L_{máx}$, em que $L_{máx}$ é valor máximo da escala de cinza. A função densidade de probabilidade pode ser obtida pela função do histograma (2.1) em que $\sum_{i=0}^{L_{máx}} p_i = 1$ e p_i é a probabilidade do *i*-ésimo tom ser utilizado novamente (PEDRINI; SCHWARTZ, 2007).

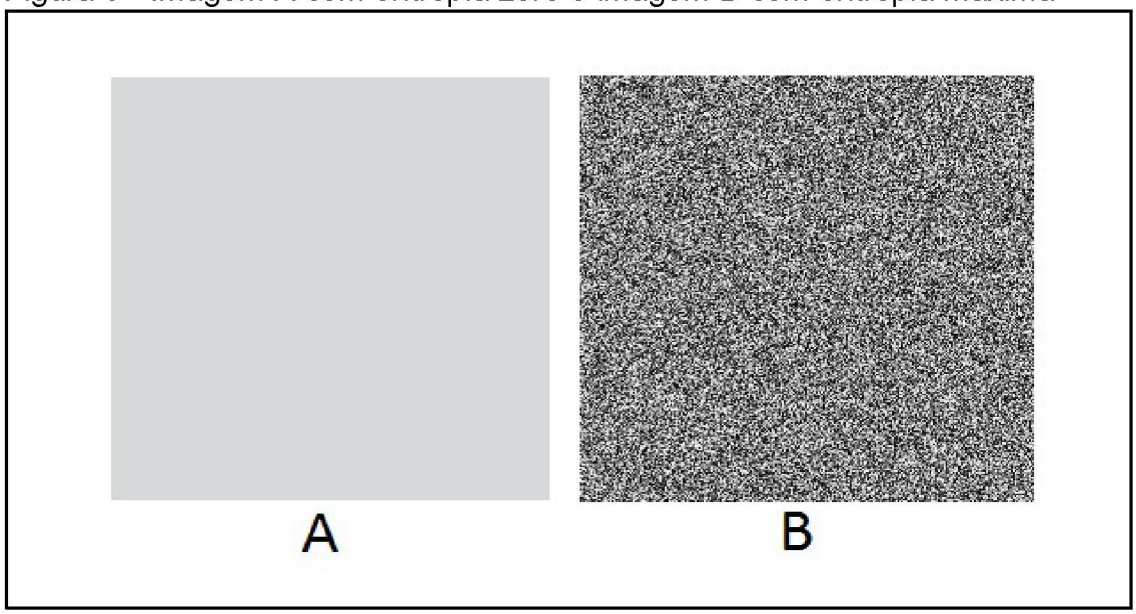
A entropia *H* de uma imagem pode ser calculada como

$$H = -\sum_{i=0}^{L_{máx}} p_i \log p_i \qquad (2.2)$$

e seu valor é sempre positivo. A base do logaritmo na Equação 2.3 determina a unidade com que se mede a entropia. Utilizando-se um logaritmo de base r, a medida da entropia é em unidades *r*-árias. Se a base for 2, a unidade resultante é binária, ou em bits. Quando todos os *pixels* da imagem possuem o mesmo nível de cinza, o valor da entropia é zero, o menor possível. Já, quando a entropia é máxima, é porque a imagem contém a mesma quantidade de *pixels* para todos os níveis de cinza (BOVIK, 2009).

Na Figura 5, a imagem A apresentada possui entropia zero, já a imagem B possui entropia máxima, considerando as imagens de tamanho igual a 256x256 *pixels*.

Figura 5 - Imagem A com entropia zero e imagem B com entropia máxima



Fonte: Adaptado de (PEDRINI; SCHWARTZ, 2008), p. 29.

2.2Segmentação de imagens

A análise de imagens é uma área do processamento de imagens em que há a extração de informação de uma imagem. Normalmente, a primeira etapa em análise de imagens é a segmentação de imagens. Essa etapa divide uma imagem em partes menores que se diferenciam entre si, que são objetos de interesse ou de

relevância para uma determinada aplicação. Uma das etapas seguintes à segmentação de imagem é a interpretação de imagem, que é um processo complexo. A subdivisão de forma efetiva da imagem pode auxiliar esse passo seguinte da análise de imagens (WANG, WEIWEI; WU, 2017).

A segmentação da imagem deve ser realizada até um nível em que os objetos de interesse estejam isolados. Há várias técnicas de segmentação de imagens e geralmente elas são baseadas nas propriedades dos níveis de cinza: descontinuidade e similaridade. A primeira abordagem é a subdivisão da imagem levando-se em conta as mudanças abruptas dos níveis de cinza. Já a segunda abordagem procura agrupar pontos da imagem cujos valores de nível de cinza sejam similares (PEDRINI; SCHWARTZ, 2007).

2.2.1 Limiarização

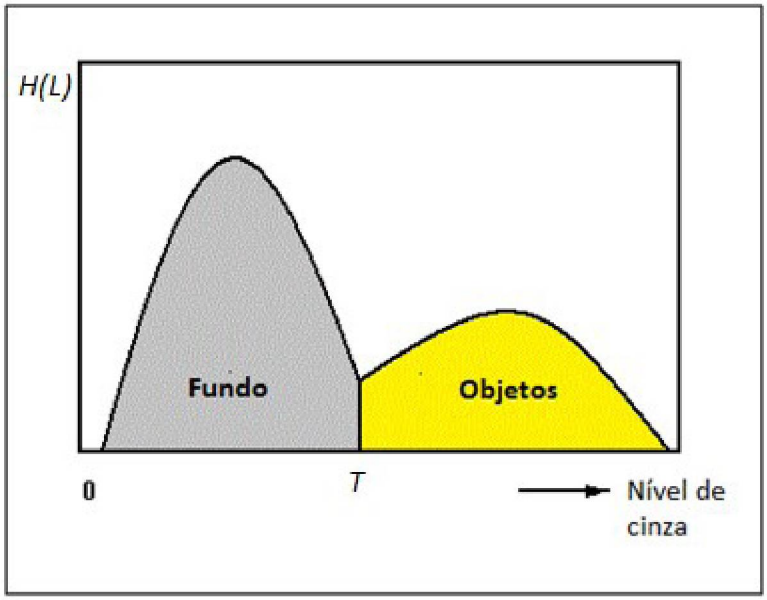
Segundo GONZALEZ e WOODS (2000), a limiarização é uma das principais técnicas de segmentação de imagem. Ela é baseada na similaridade dos valores de nível de cinza da imagem. Um histograma dos níveis de cinza $H(L)$de uma imagem $f(x,y)$ é apresentado na Figura 6, de forma que os *pixels* dos objetos e os do fundo tenham seus níveis de cinza divididos em dois grupos dominantes. Uma forma de diferenciar os objetos do fundo é estabelecendo um limiar Tque separe os dois grupos. Cada ponto (x,y) da imagem, tal que $f(x,y) > T$, é classificado como um ponto que pertence ao objeto e cada ponto em que $f(x,y) \leq T$ é identificado como um ponto do fundo. A imagem limiarizada $g(x,y)$ é então definida como

$$g(x,y) = \begin{cases} 0, & \text{se } f(x,y) \leq T \\ 1, & \text{se } f(x,y) > T \end{cases} \tag{2.3}$$

Essa limiarização é definida também como binarização, porque os *pixels* com níveis de cinza de valor maior que o limiar T recebem o valor 1, correspondente ao fundo da imagem, enquanto os que de valor menor recebem o valor 0, que corresponde

aos objetos. Ou seja, a imagem resultante tem apenas dois valores de intensidade, 1 (cor branca) e 0 (cor preta).

Figura 6- Histograma de níveis de cinza, divididos por um limiar único *T*



Fonte: Adaptado de (3D-DOCTOR, 2008).

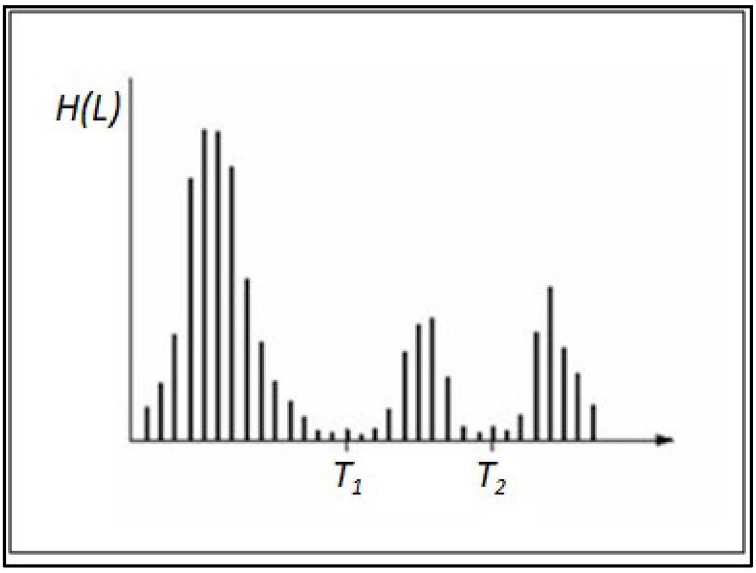
A Figura 7 apresenta os níveis de cinza dos *pixels* de uma imagem $f(x,y)$, de forma que essas intensidades estão divididas em três grupos. Para separar esses grupos, dois limiares T_1 e T_2 podem ser escolhidos. Esse histograma pode corresponder a uma imagem em que, sobre um fundo escuro, dois tipos de objetos são iluminados. Essa limiarização é denominada limiarização multiníveis e pode ser definida como

$$g(x,y) = \begin{cases} l_1, & se\ f(x,y) \leq T_1 \\ l_2, & se\ T_1 < f(x,y) \leq T_2 \\ l_3, & se\ f(x,y) > T_2 \end{cases} \quad (2.4)$$

tal que, para cada intervalo, há a especificação deuma intensidade l correspondente. Esse é um caso mais geral da binarização. Quanto em mais grupos

os níves de cinza de uma imagem se concentrarem, mais limiares *T* são necessários para separá-los (PEDRINI; SCHWARTZ, 2007).

Figura 7- Histograma particionado em múltiplos limiares



Fonte: Adaptado de (GONZALES; WOODS, 2000), p. 316.

A utilização de apenas um limiar na segmentação de toda uma imagem nem sempre é adequada, levando em consideração que a imagem pode possuir valores de intensidades próximos entre os tipos de objetos ou ainda entre um tipo de objeto e o fundo. Para situações assim, resultados melhores podem ser alcançados por meio da limiarização com múltiplos valores de limiares (PEDRINI; SCHWARTZ, 2007).

2.2.2 Limiarização por entropia

Conforme apresentado na seção anterior, há a necessidade de calcular os valores de limiar na limiarização. Esse cálculo deve ser feito de modo que os limiares façam uma “boa” separação entre os tipos de objetos presentes na imagem e entre os tipos de objetos e o fundo da imagem. Entre os melhores métodos de cálculo de limiares, as limiarizações baseada em entropia têm atraído a atenção de pesquisadores, dentre as quais o método de limiarização de (KAPUR; SAHOO; WONG, 1985) baseado em entropia foi considerado com desempenho superior pelos pesquisadores (HARNRNOUCHE; DIAF; SIARRY, 2010; AKAY, 2013) comparado com outras técnicas de limiarização (BHANDARI *et al.*, 2014).

2.2.2.1 Método de limiarização binível de Kapur baseado em entropia

PUN (1980)propôs um método de limiarização em que se procura a maximizar a entropia de uma imagem. Nesse método a distribuição de probabilidade dos níveis de cinza do histograma deve estar separada em duas classes distintas: uma representando o objeto e a outra, o fundo da imagem. As entropias $H_b(T)$ e $H_w(T)$ associadas, respectivamente, aos *pixels* do fundo da imagem e do objeto, são definidas por

$$H_b(T) = -\sum_{i=0}^{T} p_i \log p_i$$
$$H_w(T) = -\sum_{i=T+1}^{L-1} p_i \log p_i \qquad (2.5)$$

Sendo $\sum_{i=0}^{L-1} p_i = 1$, $p_{i=}\frac{n_i}{n}$, tal que p_ié a probabilidade de a intensidade $i$ser encontrada na imagem, n_ié o número de *pixels* com intensidade i, n é o número de *pixels* presente na imagem eLé o número de intensidades da imagem. É feita a análise da escala de intensidades com o objetivo de determinar um valor de limiar *T* de forma que $T = H_b(T) + H_w(T)$ seja de valor máximo, resultando em umaapropriada separação entre o fundo da imagem e o tipo de objeto ((PEDRINI; SCHWARTZ, 2007), ou seja,

$$T = \arg\max[H_b(T) + H_w(T)] \qquad (2.6)$$

KAPUR; SAHOO e WONG (1985)propuseram que as entropias entropias $H_b(T)$e $H_w(T)$, diferentemente dePUN (1980), sejam calculadas por

$$H_b(T) = -\sum_{i=0}^{T} \frac{p_i}{p_1(T)} \log \frac{p_i}{p_1(T)} \tag{2.7}$$

$$H_w(T) = -\sum_{i=T+1}^{L-1} \frac{p_i}{p_2(T)} \log \frac{p_i}{p_2(T)} \tag{2.8}$$

sendo $p_1(T) = \sum_{i=0}^{T} p_i \; p_2(T) = \sum_{i=T+1}^{L-1} p_i$e $p_1(T) + p_2(T) = 1$. Como nos trabalhos de (PUN, 1980), é realizado um processo de escolha do limiar T com objetivo de maximizar a equação2.6.

2.2.2.2 Método de limiarização multiníveis de Kapur baseado em entropia

Na seção anterior, foi considerado o método de Kapur (1985) apenas para um único limiar$T$, porém esse método pode também ser estendido para múltiplos limiares (SATHYA; KAYALVIZHI, 2011), em que, na imagem, há vários tipos de objetos relevantes, conforme segue

$$\begin{aligned}
H_0 &= -\sum_{i=0}^{t_1-1} \frac{p_i}{w_0} \log \frac{p_i}{w_0}, \quad w_0 = \sum_{i=0}^{t_1-1} p_i \\
H_1 &= -\sum_{i=t_1}^{t_2-1} \frac{p_i}{w_1} \log \frac{p_i}{w_1}, \quad w_1 = \sum_{i=t_1}^{t_2-1} p_i \\
H_2 &= -\sum_{i=t_2}^{t_3-1} \frac{p_i}{w_2} \log \frac{p_i}{w_2}, \quad w_2 = \sum_{i=t_2}^{t_3-1} p_i \\
H_j &= -\sum_{i=t_j}^{t_{j+1}-1} \frac{p_i}{w_j} \log \frac{p_i}{w_j}, \quad w_j = \sum_{i=t_j}^{t_{j+1}-1} p_i \\
H_m &= -\sum_{i=t_m}^{L-1} \frac{p_i}{w_m} \log \frac{p_i}{w_m}, \quad w_m = \sum_{i=t_m}^{L-1} p_i
\end{aligned} \tag{2.9}$$

onde $t_1 \ldots \; t_m$ são os múltiplos limiares. Como Pun (1980, 1981), os limiares $t$que melhor separem os tipos de objetos entre si e os tipos de objetos do fundo da imagem são encontrados com a maximização da soma das entropias, dado pela equação 2.11, ou seja

$$t = \arg\max\left[\sum_{i=0}^{m} H_i\right] \tag{2.10}$$

tal que té um vetor que contém os múltiplos limiares $t_1 \ldots t_m$ que maximizam a equação 2.11 (BHANDARI *et al.*, 2014).

O foco da otimização nesta dissertação é maximizar a função objetivo representada pela equação 2.11por meiode meta-heurísticas de otimização. Estas serão detalhadas no capítulo 4.

3. REVISÃO DA LITERATURA

Uma abordagem integrada à segmentação de imagens é apresentada por (BHALERAO; WILSON, 2001) que combina informações de região e fronteira usando a estimativa máxima a posteriori e a teoria da decisão. O algoritmo emprega estimativa iterativa, orientada a decisão, realizada em uma nova representação multi-resolução.

O uso de uma técnica de multi-resolução garante tanto a robustez em relação ao ruído quanto a eficiência da computação, enquanto o processo de decisão e estimativa baseado em modelo é flexível e espacialmente local, evitando suposições sobre homogeneidade global ou tamanho e número de regiões. Uma avaliação comparativa do método em relação a métodos únicos somente para região e somente para fronteira é apresentada e mostra que eles produzem segmentações precisas com índices de sinal e ruído baixos.

A mistura finita é uma ferramenta de modelagem probabilística flexível e poderosa, podendo ser usada para fornecer um grupo (*cluster*) baseado em modelo no campo de reconhecimento de padrões. No entanto, a aplicação de misturas finitas à segmentação de imagens apresenta algumas limitações, tais como a estimativa do número de componentes ainda ser uma questão aberta e o agrupamento de dados baseado em mistura não considerar informações espaciais, o que é importante para que regiões suaves sejam obtidas nos resultados da segmentação. Na pesquisa de (YANG, XIANGYU; KRISHNAN, 2004), a informação espacial é usada como um conhecimento prévio do número de componentes. A informação espacial não indica o valor do número de componentes, em vez disso, ela fornece algumas informações indiretas sobre esse valor. Um algoritmo baseado em maximização de expectativa é desenvolvido para estimar a densidade da mistura usando a informação indireta. Os resultados experimentais com dados simulados da mistura Gaussiana bidimensional mostram que o algoritmo proposto é capaz de estimar o número de componentes com precisão sem usar nenhum critério de seleção do modelo. Os resultados da segmentação de imagens mostram que o algoritmo proposto possui melhor desempenho na geração de regiões suaves nos resultados da segmentação em

comparação com os algoritmos comuns que utilizam os critérios de seleção do modelo para estimar o número de componentes.

Uma forma alternativa de avaliação de desempenho para algoritmos de segmentação de imagem é apresentada por POLAK; ZHANG e PI (2009), definindo uma nova medida de erro para a qualificação do algoritmo. Tal medida de erro é baseada em comparações objeto-a-objeto entre a imagem segmentada e uma imagem de referência considerada como verdadeira, levando em consideração o tamanho, a forma e a posição de cada objeto. Comparada com as medidas de erro existentes o método proposto funciona no nível do objeto e é sensível tanto à sub-segmentação quanto à sobre-segmentação.

LO *et al.*(2011) apresentam em sua pesquisa um método de segmentação de imagens a partir das características de textura invariantes de escala e rotação a partir da transformada de *complex dual tree wavelet* para a extração de características. O desempenho da segmentação de imagem utilizando esse novo método é então comparado com diferentes técnicas sobre diferentes bases de imagens, demonstrando um desempenho satisfatório sobre imagens em geral e um desempenho particularmente satisfatório sobre imagens que contenham objetos com texturas escalonadas e rotacionadas.

Um método também utilizado para realizar a segmentação de imagens é a máquina de vetor de suporte (*do inglês, Support Vector Machine* (SVM)*)*, graças à sua utilidade em inúmeras áreas, tais como classificação de imagens, categorização de texto e reconhecimento de escrita manual. Um dos principais problemas de pesquisa é como melhorar a eficiência do modelo SVM original sem deteriorar o seu desempenho de classificação. Por isso, YU; WONG e WEN (2011) propõem e sua pesquisa um algoritmo baseado em SVM modificando as propriedades dos vetores de suporte e uma estratégia de poda para preservar vetores de suporte, ao mesmo tempo em que elimina vetores de treinamento redundantes. Os experimentos realizados mostram que a aproximação proposta pode reduzir a quantidade de vetores de treinamento na entrada, preservando os vetores de suporte, o que leva a uma redução significativa no custo computacional obtendo níveis de precisão similares, apresentando sucesso em aplicações de segmentação de imagens.

Em sua pesquisa, SONG *et al.*(2015) propõem a utilização do algoritmo de otimização baseado no comportamento dos lobos cinzas (GWO) para resolver o problema de inversão da curva de dispersão das ondas de superfície. A estratégia proposta é analisada utilizando dados da campo com e sem ruído embutido. Para a verificação, os resultados obtidos com o GWO foram comparados com o algorítmo genético (GA), o algoritmo híbrido entre o otimizador por enxame de partículas e o algoritmo de busca gravitacional (*Particle Swarm Optimization and Gravitational Search Algorithm*, PSOGSA), e com o algoritmo baseado em informação de gradiente. Os resultados, tanto dos dados sintetizados quanto dos dados reais demonstraram que o GWO apresentou um apropriadoequilíbrio na exploração do espaço de busca, o que resultou em uma boa evasão de ótimos locais e uma convergência rápida. SONG *et al.* (2015) enfatizam que as vantagens do GWO são de que ele é um algoritmo simples, flexível, robusto e fácil de implementar, tendo também menos parâmetros de controle para serem ajustados que muitas meta-heurísticas.

A correspondência por padrão de pontos (do inglês *Point Pattern Matching, PPM)* é a tarefa de emparelhar os pontos em duas imagens de uma mesma cena. Existem muitas abordagens na literatura para a correspondência por padrão de pontos, tal como em ZHU; LIANG e YAN (2014), onde os autores propõem a utilização de um algoritmo PPM robusto para a decomposição de *QR-codes*. No entanto, a desvantagem reside na alta complexidade dos algoritmos. Para superar esta desvantagem, Sreeja e Sankar (2015) propõemum algoritmo baseado na busca binária pelo ponto de correspondência utilizando otimizador baseado no comportamento da formiga-leão (*do inglês, Ant-Colony Optimization Based Binary Search Point Matching* -ACOBSPPM). De acordo com essa abordagem, as bordas da imagem são armazenadas na forma de padrões de pontos. Os resultados experimentais mostram que o algoritmo ACOBSPPM é eficiente quando comparado com as abordagens existentes de correspondência de padrões de pontos em termos de complexidade de tempo e de precisão.

O método de limiarização de Otsu(OTSU, 1979) é uma aproximação clássica na segmentação de imagens. Embora o método de aproximação bidimensional (2D) de Otsu apresente um desempenho melhor que o original na segmentação de imagens corrompidas por ruído, ele é sensível ao ruído do tipo *Salt&Pepper* (Sal e Pimenta).

Na tentativa de resolver esse problema, SHA; HOU e CUI (2016) apresentam um método de Otsu bidimensional robusto, construindo o histograma bidimensional baseado na imagem suavizada tanto por filtros de média e mediana, em contraste com o método original que utiliza apenas o filtro da média. O vetor de limiares ótimos é determinado por duas buscas unidimensionais nas duas dimensões do histograma bidimensional, em adição foi inserido um pós processamento por região para lidar com os *pixels* de ruídos e bordas. Nos resultados experimentais, comparado com o método tradicional de Otsu bidimensional, o método proposto apresentou uma melhora significativa quanto aos ruídos nos casos Gaussiano e *Salt&Pepper*.

Entre outros métodos, a entropia cruzada mínima é implementada por sua efetividade e simplicidade, tal método é eficiente e fornece excelentes resultados em casos de bi-limiarização, mas sua avaliação fica computacionalmente custosa quando extendida para a multi-limiarização. Por isso, em sua pesquisa, PARE *et al.* (2016) adotaram uma técnica de multi-limiarização baseada no algoritmo baseado em pássaros cucopara renderizar o algoritmos de entropia cruzada mínima multinível de forma mais prática, reduzindo sua complexidade. Os resultados experimentais baseados em resultados qualitativos e diferentes parâmetros de fidelidade mostraram que a aproximação proposta seleciona valores de limiarização ótimos de forma mais eficiente e precisa quando comparado com outras técnicas, também produz imagens segmentadas de alta qualidade.

Na tentativa de reduzir o tempo necessário, JIANG*et al.*(2016) sugerem em sua pesquisa a utilização de um algoritmo de multi-limiarização baseado no comportamento das abelhas (do inglês, *Honey Bee Mating Algorithm*, HBMA), e no aprendizado cooperativo, o que aumentou consideravelmente a capacidade de pesquisa do algoritmo. No algoritmo adotado, foi adotada uma nova estratégia de inicialização de população para tornar a pesquisa mais eficiente, o que foi comprovado nos resultados em comparação com outros algoritmos utilizados, tais como algumas variações do PSO.

Segundo KHAIRUZZAMAN e CHAUDHURY (2017) a multilimiarização é uma das áreas de maior importância no campo da segentação de imagem. Entretanto, a complexidade computacional da multilimiarização aumenta exponencialmente com o aumento do número de limiares, então para superar esse problema é proposta uma

aproximação GWO. Tal meta-heurística é aplicada ao problema de multilimiarização utilizando funções de entropia de Kapur e Otsu entre classes. O desempenho do método proposto é comparado então com versões do otimizador por enxame de partículas e com BFO (*Bacterial Foraging Optimization*) baseados em métodos de multilimiarização. A qualidade das imagens segmentadas é computada utilizando o índice*Mean Structural Similarity* (MSSIM). Os resultados experimentais mostram que o método proposto é mais estável e elabora soluções de de maior qualidade do que os métodos utilizando PSO e BFO, entretanto o método proposto se mostrou mais rápido que o BFO mas mais lento que o método baseado em PSO.

A determinação de limiares ótimos para a segmentação de imagens obteve cada vez mais atenção nos últimos anos devido ao fato de ter muitas aplicações. Os métodos mais utilizados para determinar um limiar único para imagem, tal como Otsu e Kapur, podem facilmente ser extendidos para casos de multilimiarização, entretanto o consumo de tempo aumenta consideravelmente nessa aplicação. Para evitar esse problema, AZIZ *et al.*(2017) examinam os algoritmos WOA e MFO para determinar os diversos limiares para a segmentação de imagens. Os resultados foram comparados com outros cinco algoritmos baseados em inteligênicia de enxames, sendo eles: SCA (*Sine Cosine Algorithm*), HS (Hirschberg–Sinclair *Algorithm*), SSO (*Simplified Swarm Optimization*), FASSO (*Fuzzy Adaptive Swallow Swarm Optimization*) e FA (*Firefly Algorithm*). Osresultados apresentadosmostraram que os métodos propostos tiveram melhor desempenho que os outros algoritmos baseados em inteligência de enxames, também que o MFO obteve melhores resultados que os obtidos pelo WOA.

NAIDU*et al.*(2016) propuseram em sua pesquisa um algoritmo de multi-limiarização de imagens baseado no algoritmo FA com o objetivo de maximizar a entropia proposta por Shannon, de forma similar à entropia *fuzzy*. O algoritmo proposto do testado em um conjunto padrão de imagens os resultados foram comparados com métodos baseados em entropia Shannon utilizando os métodos de otimização DE (*Differential Evolution*), PSO (*Swarm Optimization*) e BA (*Bat Algorithm*). O método proposto na pesquisa exibiu um melhor desempenho na função objetivo, no índice de similaridade estrutural, na proporção entre o pico do sinal e o ruído, no erro de classificação e no tempo de execução computacional do que os métodos utilizados como base de comparação.

Em sua pesquisa, HAN *et al.*(2017) propõem a utilização de um histograma normalizado de uma imagem adaptado por funções de distribuição normal linearmente combinadas onde cada distribuição normal representa uma classe de *pixels*, sendo os parâmetros como a média, variância e pesos da função de adaptação indeterminados. Transformando a função de adaptação em um problema de otimização não linear e não convexo, o algoritmo de transição de estados (*State Transition Algorithm*,STA) é utilizado para escolher os parâmetros ótimos da função de adaptação. A efetividade da aproximação de multi-limiarização utilizando STA proposta foi testada e apresentou resultados competitivos tanto em termos de otimização quanto na segmentação de imagens quando comparada com os métodos de Otsu, PSO, GAe DE.

Na pesquisa de SAHOO eCHANDRA (2017) o GWO é utilizado com o foco de identificar lesões no colo do útero à partir imagens de contraste CT-Scan melhoradas para fornecer uma discriminação confiável e objetiva entre lesões benignas e malignas. O GWO tradicicionalmente é aplicável em problemas de otimização contínuos de objetivo simples, mas em sua pesquisa Sahoo e Chandra (2017) propõe duas aproximações diferentes para algoritmos GWO com representação bináriapara otimização multi-objetivo. Uma é uma abordagem escalarizada para GWO multi-objetivo (MOGWO) e a outra é uma GWO com classificação não dominada (NSGWO). Estes são usados para a seleção de características baseadas no wrapper que seleciona um subconjunto de característica textural ideal para melhor classificação das lesões do colo do útero. Os resultados das abordagens propostas são comparados com meta-heurísticas clássicas da literatura, tais como GA e algoritmo de dos vagalumes (*do inglês, firefly algorithm*, FA) para otimização multi-objetivo. Com melhor diversificação e intensificação, o GWO obtém soluções de Pareto, que dominam as soluções obtidas por GA e FA quando avaliadas nos casos de lesão do colo do útero utilizado. Outras experiências foram realizadas em conjuntos de dados de expressão de genes de microarrays de alta dimensão coletados on-line. Os resultados demonstram que o método proposto é significativamente melhor do que outros métodos selecionando genes relevantes para diagnóstico de câncer de alta dimensão e multi-categoria.

Uma importante aplicação da segmentação de imagem é a segmentação de imagens do fígado em imagens médicas, conforme mencionado por ZIDAN *et al.*

(2016). Em sua pesquisa o algoritmo baseado no comportamento da formiga Leão (*do inglês, Ant Lion Optimizer -ALO*) é utilizado como uma técnica de agrupamento para executar o processo de segmentação em imagens de ressonância magnética (*Magnetic Resonance Imaging*, MRI). O algoritmo ALO é combinado com uma imagem estatística do fígado para segmentar o fígado inteiro. A região segmentada do fígado é melhorada por meiode operações morfológicas. Então, a técnica de agrupamento de mudança média divide o fígado segmentado em várias regiões de interesse. Por meio do ALO, a técnica de agrupamentocalcula os valores de diferentes grupos (*clusters*) na imagem. Uma imagem estatística do fígado é usada para obter a região potencial que pode existir. Alguns *pixels* que representam os grupos necessários são apanhados para obter o fígado segmentado inicial. Em seguida, o fígado segmentado é aumentado usando operações morfológicas. Finalmente, a técnica de agrupamento de mudança média divide o fígado em diferentes regiões de interesse. Um conjunto de 70 imagens por ressonância magnética, foi utilizado para segmentar o fígado e testar a abordagem proposta. O índice de semelhança estrutural valida o sucesso da abordagem. Os resultados experimentais mostraram que a precisão geral da abordagem proposta resulta em 94,49% de precisão.

Tabela 1 - Sumarização da literatura

Referência	Método proposto
(BHALERAO; WILSON, 2001)	Estimativaiterativa orientada a decisão, realizada em uma representação multi-solução.
(YANG, XIANGYU; KRISHNAN, 2004)	Maximização da estimativa.
(POLAK; ZHANG; PI, 2009)	Medida de erro baseada em comparações objeto-a-objeto entre a imagem segmentada e uma imagem de referência considerada como verdadeira.
(LO *et al.*, 2011)	Segmentação de imagens a partir das características de textura de escala e rotação.
(YU; WONG; WEN, 2011)	Utilização da máquina de vetor de suporte (SVM).
(SONG *et al.*, 2015)	Utilização do Otimizador Lobo cinzento (GWO) comparando com o algoritmo genético (GA) e com o

	otimizador por enxame de partículas e o algoritmo de busca gravitacional (PSOGSA).
(ZHU; LIANG; YAN, 2014)	Utilização de correspondência por padrão de pontos (PPM).
(SREEJA; SANKAR, 2015)	Utilização de um algoritmo baseado na busca binária pelo ponto de correspondência utilizando otimizador baseado no comportamento da formiga-leão (ACOBSPPM).
(SHA; HOU; CUI, 2016)	Utilização de um método de Otsu bidimensional robusto.
(PARE *et al.*, 2017)	Utilização da pesquisa de Cuckoo (*Cuckoo search*).
(JIANG, YUNZHI *et al.*, 2016)	Utilização de um algoritmo baseado no enxame de abelhas.
(KHAIRUZZAMAN; CHAUDHURY, 2017)	Utilização do algoritmo otimizador do lobo cinzento (GWO) comparando sua perfirmance com o algoritmo BFO.
(AZIZ; EWEES; HASSANIEN, 2017)	Utilização dos algoritmos baseados no comportamento de baleias (WOA) e das mariposas (MFO).
(NAIDU; RAJESH KUMAR; CHIRANJEEVI, 2016)	Utilização do algoritmo baseado nos vagalumes (FA).
(HAN *et al.*, 2017)	Utilização do algoritmo de transição de estados (STA).
(SAHOO; CHANDRA, 2017)	Utilização do algoritmo otimizador do lobo cinzento (GWO).
(ZIDAN *et al.*, 2016)	Utilização do algoritmo otimizador da formiga leão (ALO).

4. META-HEURÍSTICAS DE INTELIGÊNCIA DE ENXAME

Meta-heurística é um método flexível para resolver diferentes formasdeproblemas de otimização, ou seja, pode-se dizer que é um procedimentode ampla aplicabilidade para determinarsoluções de um problema de otimização, mesmo com informações incompletas ou imperfeitas ou limitações na capacidade computacional. Uma meta-heurística é normalmente utilizada quando os valores máximos ou mínimos de uma determinada função não são facilmente encontrados de formaanalítica (BIANCHI *et al.*, 2009).

As meta-heurísticas se tornaram populares nos últimos 20 anos. Pode-se mencionar as meta-heurísticas Algoritmo Genético (*Genetic Algorythm* (GA)), Otimizador Colônia de Formiga (*Ant Colony Optimizer* (ACO)) e Otimização de Enxame de Partículas (*Particle Swarm Optimizer* (PSO)) são bem conhecidas, não apenas entre cientistas da computação, mas, entre cientistas de diferentes áreas do conhecimento, uma vez que tais técnicas de otimização são empregadas em vários campos de estudo (MIRJALILI; MIRJALILI; LEWIS, 2014).

A popularidade das meta-heurística pode ser atribuída aos seguintes fatores (MIRJALILI; MIRJALILI; LEWIS, 2014):

- *Simplicidade*: São inspiradas em conceitos simples, observados na natureza, tais como fenômenos físicos, comportamentos de animais ou conceitos da evolução. A simplicidade permite a simulação de diferentes conceitos, propor novas meta-heurísticas, juntar mais de uma meta-heurística ou melhorar as meta-heurísticas existentes.
- *Flexibilidade*: As meta-heurísticas conseguem ser aplicadas aos mais variados problemas, sem necessidade de alterações em sua estrutura, já que tratam os problemas como modelos caixas-pretas.
- *Prevenção de máximos locais*: As meta-heurísticas possuem habilidades superiores para prevenir os máximos locais, já que sua natureza estocástica permite evitar estagnação em soluções locais e procuraamplano espaço de busca um problemas de otimização global.

Existem 3 classes de meta-heurísticas. A primeira classe compreende os algoritmos evolutivos, que são inspirados em conceitos de evolução na natureza. O mais conhecido dessa classe é o Algoritmo Genético, proposto por Holland, nos anos 1960, que simula conceitos da evolução Darwiniana. A otimização é realizadaevoluindo uma solução gerada com transições estocásticas. Cada nova população é geradapela combinação e mutação de indivíduos da geração anterior. Como os melhores indivíduos têm chance maior de participar da geração de nova população, espera-se que os indivíduos (soluções candidatas) melhorem no curso das gerações. Alguns exemplos de algoritmos evolutivos são: Evolução Diferencial (*Diferential Evolution* (DE)), Programação Evolutiva (*Evolutionary Programming* (EP)), Estratégia Evolutiva (*Evolutionary Strategy* (ES)), Programação Genética (*Genetic Programming* (GP)) e Otimizador Baseado em Biogeografia (*Biogeography-Based Optimizer* (BBO)) (MIRJALILI; MIRJALILI; LEWIS, 2014).

A segunda classe de meta-heurísticas é a das técnicas baseadas na física. Essas otimizações mimetizam leis da física. Os agentes de busca se comunicam e se movem dentro de um espaço de busca de acordo com leis da física, tais como força gravitacional, queda de raios, força eletromagnética, inércia, entre outros. Alguns exemplos de otimizadores dessa classe são: Busca Local Gravitacional (*Gravitational Local Search,*GLSA), Grande Crise do *Big Bang* (*Big-Bang Big-Crunch* (BBBC)), Busca de Sistema Carregado (*Charged System Search*, CSS), Otimização do Espaço Curvado (*Curved Space Optimization,* CSO), entre outros (MIRJALILI; MIRJALILI; LEWIS, 2014).

A terceira classe é a denominada inteligência de enxame, na qual, há as meta-heurísticas baseadas no comportamento social de enxames, rebanhos, alcateias, entre outros grupos de criaturas da natureza. O mecanismo é semelhante ao do mecanismo dos algoritmos baseados na física, mas os agentes de busca simulam a inteligência coletiva e social das criaturas. O PSO é a técnica de exame mais presente na literatura. Ela é baseada no comportamento social de pássaros em bando. O algoritmo PSO emprega múltiplas partículas que perseguem a posição das melhores partículas e suas melhores posições obtidas até o momento (MIRJALILI; MIRJALILI; LEWIS, 2014).

Umalgoritmo clássicode inteligência de enxame é o Otimizador Colônia de Formiga (*Ant Colony Optimizer*, ACO), baseado na inteligência social das formigas para encontrar o caminho mais curto entre o ninho e a fonte de alimento. Outro exemplo de algoritmo desse grupo é o denominado Colônia Artificial de Abelhas (Artificial Bee Colony, ABC), que mimetiza o comportamento coletivo das abelhas para encontrar fontes de alimento (MIRJALILI; MIRJALILI; LEWIS, 2014).

4.1 FUNDAMENTOS DE INTELIGÊNCIA DE ENXAME

A Inteligência de enxame (*Swarm Intelligence*) teve sua concepção a partir da percepção das habilidades sociais dos insetos para resolver seus problemas diários (BONABEAU; DORIGO; THERAULAZ, 1999). A inteligência de enxame foi inspirada por estudos em neurociências, psicologia cognitiva e ciências comportamentais, o conceito de inteligência de enxamefoi introduzido no domínio da computação e inteligência artificial em 1989 como uma solução inovadora para a resolução de problemas e inteligência distribuída.

Os algoritmos baseados em inteligência de enxame têm sido amplamente estudados e aplicados para resolver vários problemas científicos e de engenharia, tais como projeto de robôs, despacho econômico de sistemas de energia e problemas de identificação de dobra de proteína. Estes algoritmos têm desfrutado de sucesso nas aplicações devido à sua simplicidade, robustez e flexibilidade. Os algoritmos baseados em inteligência de enxame atuam sobre uma população de possíveis soluções aplicando o princípio de diversidade de indivíduos (agentes) e da sobrevivência de indivíduos mais fortes e bem adaptados ao ambiente, que se reproduzem por meiode operadores que imitam os conceitos genéticos, gerandodescendentes mais fortes que se aproximam da solução do problema (SERAPIÃO, 2009).

Um enxame é formado por uma população de agentes homogêneos e simples que executam tarefas simples interagindo localmente entre si, sem um controle central. Mesmo esses agentes sendo simples e com capacidades limitadas, a capacidade de

resolver problemas se dá atravez do compartilhamento de informação e do seu comportamento colaborativo (MARINI; WALCZAK, 2015).

4.1.1Princípios biológicos da inteligência de enxame

Um exemplo de modelo que serve como inspiração na elaboração de algoritmos bio-inspirados é a colônia de formigas, foi observado que durante a procura de alimento para a colônia as formigas propagam os resultados sobre as trilhas encontradas influenciando o ambiente, espalhando uma substância química conhecida como feromônio no ambiente de busca, o feromônio é o método de comunicação entre os indivíduos da colônia (KHAN; BAIG, 2017).

Outro exemplo, menos utilizado, é a colônia de libélulas, que apresentam comportamentos de enxames tanto na procura de alimento quanto na migração. O comportamento do enxame de libélulas à procura de alimento é caracterizado pela formação de um pequeno grupo de indivíduos movendo-se localmente mudando de posição de forma abrupta. Enquanto o comportamento do migratório do enxame de libélulas é caracterizado por um grupo de indivíduos movendo-se na mesma direção por longas distâncias (RANJINI; MURUGAN, 2017).

No entanto, a complexidade de tais comportamentos e estruturas coletivas não reflete a simplicidade do comportamento individual de um inseto. Entretanto, como é apontado por SEELEY (2002), a complexidade de um indivíduo, inseto, em termos cognitivos ou de habilidades de comunicação podem ser altas em um sentido absoluto, mesmo não sendo altas o suficiente para supervisionar um grande sistema ou para explicar a complexidade de todos os comportamento na escala da colônia. Em muitos casos, um inseto não é capaz de encontrar uma solução eficiente para um problema da colônia sozinho, enquanto a sociedade à qual ele pertence consegue encontrar, como um todo, uma solução facilmente.

Por trás de toda essa “organização sem um organizador” existem vários mecanismos escondidos que permitem à sociedade de insetos, cujos membros lidam apenas com informações parciais sobre o ambiente, encontrarem soluções de problemas complexos (GARNIER; GAUTRAIS; THERAULAZ, 2007).

Os exemplos citados anteriormente contam apenas com os mecanismos que conhecidamente ocorrem com insetos sociáveis, entretanto é importante notar que vários outros sistemas biológicos compartilham as mesmas propriedades coletivas, tais como as colônias de lobos cinzentos (EMARY; ZAWBAA; HASSANIEN, 2016) e as baleias (MIRJALILI; LEWIS, 2016).

4.1.2 Processo de auto-organização de um enxame

As decisões coletivas tomadas em um exame contam com o componente de auto organização. A auto organização é um grupo de mecanismos dinâmicos onde estruturas aparecem no nível global de um sistema à partir das interações entre os componentes dos níveis mais baixos, sem serem codificadas explicitamente no nível individual. A auto organização depende de quatro regras básicas, sendo elas:

- *Realimentação positiva*: resulta da execução de regras gerais comportamentais simples que promovem a criação de estruturas;
- *Realimentação negativa*: serve para contrabalançar a realimentação positiva e leva à estabilização do padrão de comportamento coletivo, podendo ser resultado de algum for externo, como mudanças no ambiente ou aparecimento de componentes não pertencentes ao grupo;
- *Amplificação da flutuação pela realimentação positiva*: amplifica o comportamento estocástico dos componentes do grupo, sendo crucial para encontrar novas soluções e explorar caminhos alternativos;
- *Múltipla interação direta entre os indivíduos*:produz resultados "aparentemente" determinísticos.

Há também algumas regras que a auto organização possui:

- Sistemas auto organizados são dinâmicos;
- Sistemas auto-organizados exibem propriedades emergentes, exibem propriedades que são mais complexas que a simples contribuição de cada agente. Tais propriedades aparecem por causa das combinações não lineares de interações entre os membros da colônia;

- Assim como as propriedades emergentes, interações não lineares levam o sistema auto organizado a bifurcações, soluções diferentes mas com qualidades semelhantes;
- Sistemas auto organizados podem sermulti estáveis. Multi-estabilidade significa que, para um dado conjunto de parâmetros, o sistema pode alcançar diferentes estados estáveis, dependendo das condições iniciais e das variações estocásticas(GARNIER; GAUTRAIS; THERAULAZ, 2007).

4.1.3 Categorização de comportamentos

A partir do processo de auto organização podem surgir uma vasta quantidade de comportamentos utilizados na resolução de problemas. Tais comportamentos podem ser desconstruídos, segundo GARNIER; GAUTRAIS; THERAULAZ (2007), em quatro funções: coordenação, cooperação, deliberação e colaboração. Tais funções não são mutuamente exclusivas, mas contribuem entre si para a realização de diversas tarefas na colônia. A seguir tem-se a definição de cada uma dessas funções.

- *Coordenação* é a organização apropriada no espaço e no tempo das atividades necessárias para a resolução de um problema, levando a distribuições espaço-temporais específicas para as atividades de cada indivíduo.
- *Cooperação* ocorre quando os indivíduos completam juntos uma atividade que não poderia ser feito por um indivíduo sozinho. Os indivíduos devem combinar esforços para solucionar um problema que está elém de suas capacidades individuais.
- *Deliberação* se refere aos mecanismos que recebem várias oportunidades, esse mecanismo resulta em uma escolha coletiva por pelo menos uma das oportunidades.
- *Colaboração* significa que diferentes atividades podem ser executadas simultaneamente por grupos de indivíduos especializados. Tal especialização pode contar com a diferenciação comportamental ou morfológica e ser influenciada pela idade dos indivíduos.

4.1.4 Modulação de comportamentos auto-organizados

Conforme mencionado anteriormente, a organização de comportamentos coletivos surge a partir de quatro funções que emergem das atividade de uma densa cadeia de interações, tais interações ocorrem entre os membros da colônia e entre eles e o ambiente no qual estão localizados. Considerando que tanto a colônia quanto o ambiente evoluem com o tempo, eles podem ser considerados como sistemas dinâmicos acoplados. Entretanto, dentro de uma colônia algumas características precisam se manter constantes, por exemplo, a queda na umidade pode ser fatal para as baratas, que para evitar a morte mantém localmente o nível de humidade (DAMBACH; GOEHLEN, 1999).

A única forma de uma colônia adaptar seu comportamento coletivo é atravéz da modulação do comportamento de seus indivíduos. A "modulação" significa a probabilidade de um dado comportamento ocorrer varia de acordo com a perturbação do ambiente. Cada indivíduo é capaz de variar levemente o seu comportamento em resposta às variações no ambiente, tal variação afeta a rede de interações, e por consequência a estrutura global por meio de um balanceamento entre realimentações negativas e positivas.

Dois tipos de perturbações causam a modulação do comportamento dos indivíduos de uma colônia. O primeiro é produzido por mudanças no ambiente, compreendendo parâmetros climáticos, como temperatura e umidade, e ecológicos, como distribuição de alimento ou a presença de um predador. O segundo é produzido dentro da colônia e está diretamento ligado à colônia ou aos seus componentes, tal como o tamanho da colônia e a distribuição morfológica entre os indivíduos (GARNIER; GAUTRAIS; THERAULAZ, 2007).

Uma propriedade interessante dos sistemas auto-organizados são a robustez, a habilidade do sistema ser executado sem falha sob uma acentuadavariação de condições. Outra propriedade é a flexibilidade, esta uma habilidade do sistema se adaptar prapidamente à uma mudança de condições ou requisitos. A robustez é o resultado da interação entre os indivíduos pertencentes à colônia, garantindo que

mesmo que um indivíduo falhe em concluir a sua atividade, essa falha será compensada pelos outros indivíduos. O que também promove a estabilidade dos padrões produzidos onde os comportamentos individuais são, em sua maioria, probabilísticos. A estabilidade também pode depender da modulação das regras comportamentais individuais por algum fator produzido pelo ambiente ou até pela própria atividade da colônia.

4.2 ALGORITMOS DE INTELIGÊNCIA DE ENXAME

Nesta seção tem-se os algoritmos de inteligência de enxame utilizados nessa dissertação.

4.2.1 Otimizador do lobo cinzento (*Grey Wolf Optimizer*, GWO)

Nesta seção é descrito o Otimizador do Lobo Cinzento. Esse algoritmo foi inspirado na organização e comportamento de caça dos lobos cinzentos e proposto porMIRJALILI; MIRJALILI; LEWIS (2014).

Os lobos cinzentos (*Canis lupus*) são mamíferos da família dos canídeos conhecidos por viverem em grupos, chamados de alcateias. O que chama a atenção para essa espécie e que serve de inspiração para o algoritmo GWO é a forma como o grupo se organiza hierarquicamente e, ainda, sua estratégia para caçar suas presas(MECH, 1999).

Esses canídeos são superpredadores, ou seja, estão no topo da cadeia alimentar. Vivem em grupos que variam de 5 a 12 indivíduos e se organizam respeitando uma hierarquia dominante bem definida. Os líderes são um macho e uma fêmea, chamados de alfas. Eles são responsáveis por tomar decisões quanto à caça, local para dormir, hora de acordar, etc. São os únicos com permissão para acasalar. Vale ressaltar que o alfa não é, necessariamente, o mais forte da alcateia, mas, sim, o melhor em termos de gerenciamento do grupo. Isso mostra que, entre os lobos

cinzentos, organização e disciplina são mais importantes que força física (MECH, 1999).

O segundo nível na hierarquia dos lobos cinzentos é o beta. Os betas auxiliam os alfas a tomarem decisões e em outras atividades. Os betas devem respeitar os alfas, mas comandam os lobos dos níveis mais baixos. Desempenham o papel de conselheiros dos líderes e disciplinadores do resto do grupo, reforçando os comandos dos alfas. O beta pode ser macho ou fêmea e é o melhor candidato para substituir o alfa, caso este faleça ou fique muito velho(MECH, 1999).

O nível mais baixo na hierarquia dos lobos cinzentos é chamado de ômega. Desempenham o papel de bode expiatório. Obedecem a todos os lobos dominantes e são os últimos a comer. Apesar de estarem no nível mais baixo, foi observado que a presença dos ômegas é importante para manter a estabilidade dos níveis mais altos, uma vez que a perda dos indivíduos ômega pode acarretar brigas e disputas entre os lobos das classes mais altas. Os ômegas também desempenham a função de cuidar dos filhotes da alcateia (MECH, 1999).

Se um lobo não for alfa, beta ou ômega, ele é denominado subordinado ou delta. São subordinados aos alfas e betas, mas dominam os ômegas. Neste grupo, estão presentes: os lobos de escolta, que vigiam os limites do território da alcateia e alertam o grupo em caso de perigo; os lobos sentinela, que protegem e garantem a segurança do grupo; os lobos idosos, que já foram alfa ou beta e foram substituídos; os caçadores; que ajudam os alfas e betas a caçar e providenciar alimento para a alcateia; e, por fim, os lobos cuidadores que cuidam dos lobos fracos, feridos ou doentes (MECH, 1999).

Além da organização hierárquica dos lobos cinzentos, a caça em grupo é outro comportamento social interessante dessa espécie.A caça pode ser dividida em três fases. A primeira é a em que os lobos rastreiam, seguem e se aproximam da presa. Na fase seguinte, eles perseguem, cercam e assediam a presa. Por último, atacam à presa (MURO *et al.*, 2011). O comportamento de caça é mostrado na Figura 8, onde: (A) é o rastreamento e a aproximação; (B-D) são a perseguição, o assédio e o cerco, respectivamente; (E) é a situação estacionária e o ataque.

Figura 8- Comportamento da caça dos lobos cinzentos

Fonte: Mirjalili et al., 2014.

A organização social e a técnica de caça são modeladas matematicamente para realizar a otimização no algoritmo do lobo cinzento.Para modelar a organização social, a solução mais apta é considerada o lobo alfa (α). A segunda e terceira melhores soluções são chamadas de beta (β) e delta (δ), respectivamente. O resto das soluções candidatas são consideradas ômega (ω). No algoritmo GWO, a caça (otimização) é guiada por alfa, beta e delta. Os ômegas seguem esses três lobos (MIRJALILI; MIRJALILI; LEWIS, 2014).

Um dos atos dos lobos, durante a caça, é o cerco da presa. A descrição bio-inspirada desse comportamento é obtidapela equações 4.1 e 4.2 dadas por:

$$\vec{D} = |\vec{C}\vec{X}_p(t) - \vec{X}(t)| \quad (4.1)$$

$$\overrightarrow{X(t+1)} = \vec{X}_p(t) - \vec{A}\vec{D} \quad (4.2)$$

onde $t$indica a iteração atual, $\vec{A}$ e $\vec{C}$ são coeficientes vetoriais, $\vec{X}_p$ é o vetor de posição da presa, e o$\vec{X}$indica o vetor de posição de um lobo cinzento. (MIRJALILI et al., 2014)

Os vetores $\vec{A}$ e $\vec{C}$ são calculados como

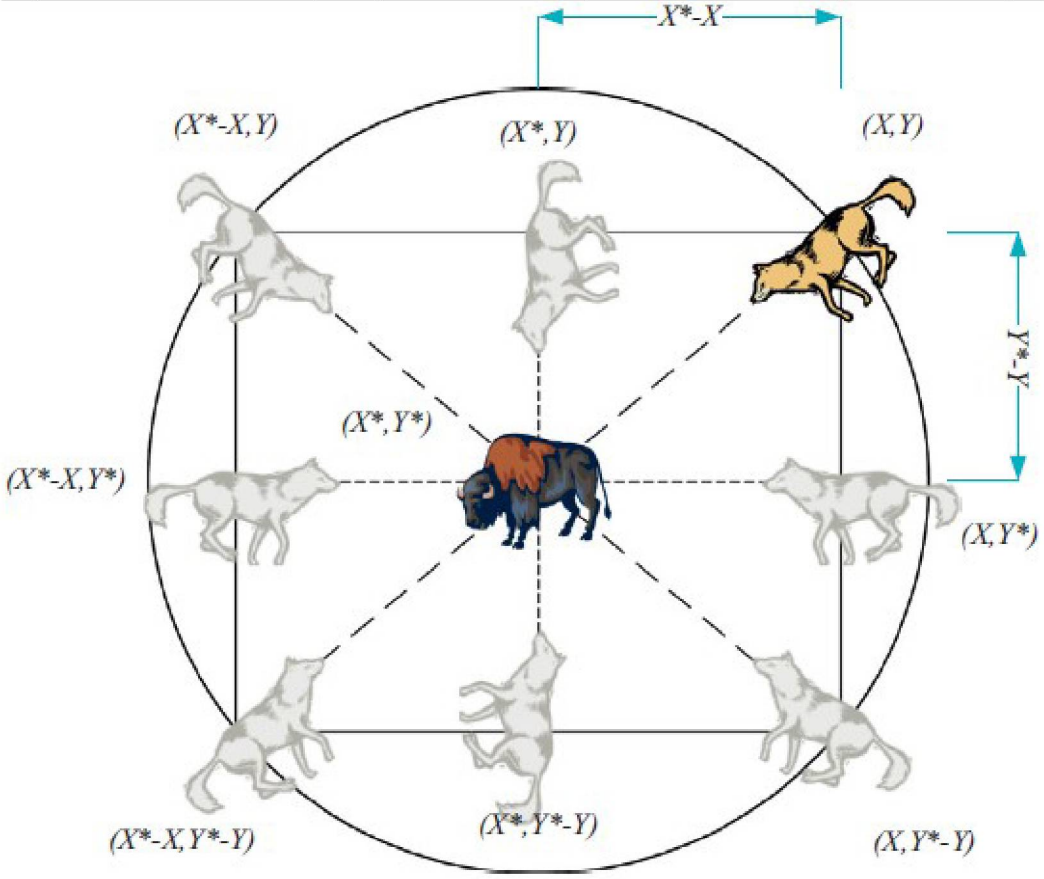
$$\vec{A} = 2\vec{a}\vec{r}_1 - \vec{a} \quad (4.3)$$

$$\vec{C} = 2\vec{r}_2 \quad (4.4)$$

tal que os componentes de $\vec{a}$ são diminuídos de forma linear de 2 a 0, durante o curso das iterações e r_1 e r_2 são números aleatórios gerados com distribuição uniforme no intervalo $[0, 1]$ (MIRJALILI; MIRJALILI; LEWIS, 2014).

Os efeitos das equações 4.1 e 4.2, um vetor de posição bidimensional e alguns dos possíveis vizinhos são ilustrados na Figura 9. O lobo cinzento na posição (X, Y) pode atualizar sua posição de acordo com a posição da presa (X^*, Y^*). Diferentes locais em torno da presa podem ser alcançados, em relação à posição atual, ajustando o valor dos vetores $\vec{A}$ e $\vec{C}$. Por exemplo, $(X - X^*, Y)$ pode ser alcançado ajustando $\vec{A} = (1, 0)$ e $\vec{C} = (1, 1)$ (MIRJALILI; MIRJALILI; LEWIS, 2014).

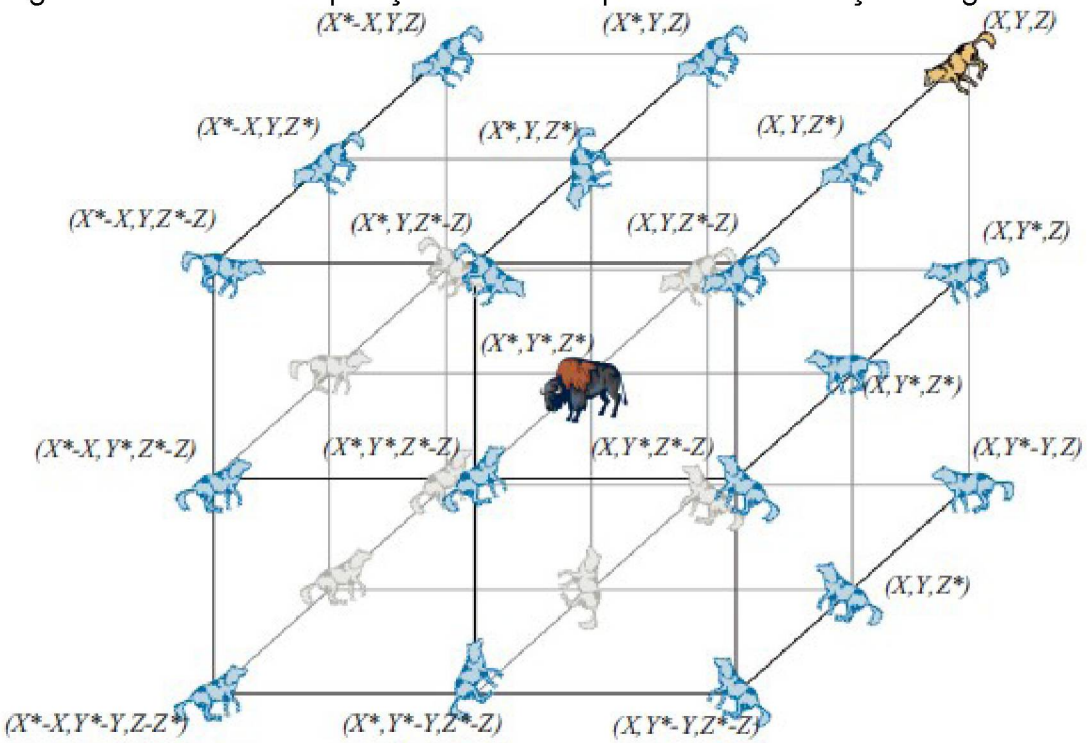
Figura 9 - Vetores de posição 2D e suas possíveis localizações subsequentes.



Fonte: Mirjalili et al., 2014.

As possíveis posições de um lobo cinzento em um espaço tridimensional são demonstradas na Figura 10. Os lobos podem atualizar suas posições com as equações 4.1 e 4.2 e esse conceito pode ser estendido a um espaço de busca com n dimensões, assim os lobos se moveriam em hipercubos ou hiperesferas em volta da melhor solução até o momento (MIRJALILI; MIRJALILI; LEWIS, 2014).

Figura 10 - Vetores de posição 3D e suas possíveis localizações seguintes.



Fonte: Mirjalili *et al.*, 2014.

Para modelar a caça, leva-se em consideração que a caça é, normalmente, guiada pelo alfa e, ocasionalmente, o beta e delta podem participar da caça. Os lobos cinzentos têm a habilidade de localizar e cercar a presa, mas, em um espaço de busca abstrato, não se sabe onde está o ótimo (presa). Para simular o comportamento de caça dos lobos cinzentos, considera-se que o alfa (melhor solução candidata), o beta e o delta conhecem a localização potencial da presa. Então, as três melhores soluções, até o momento, são salvas e os outros agentes de busca (ômegas) são forçados a atualizar suas posições de acordo com os melhores agentes de busca. Para isso, (MIRJALILI; MIRJALILI; LEWIS, 2014), propõem as seguintes equações:

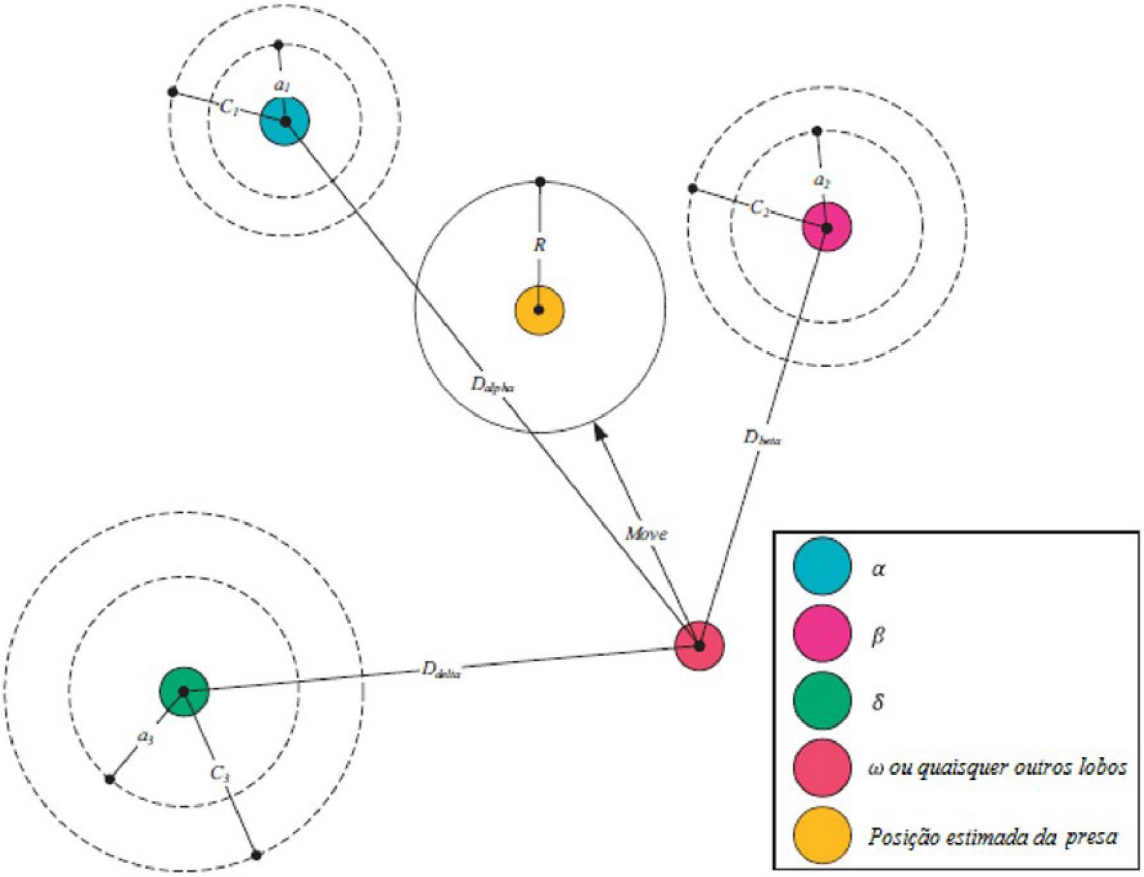
$$\vec{D}_\alpha = |\vec{C}_1\vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2\vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3\vec{X}_\delta - \vec{X}| \tag{4.5}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1(\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2(\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3(\vec{D}_\delta) \tag{4.6}$$

$$\vec{X}(T+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{4.7}$$

Esse modelo é ilustrado na Figura 11, que mostra a atualização de posição dos agentes de busca de acordo com alfa, beta e delta, em um espaço de busca bidimensional. Basicamente, alfa, beta e delta estimam a localização da presa e os outros lobos se posicionamaleatoriamente em volta dela (MIRJALILI; MIRJALILI; LEWIS, 2014).

Figura 11- Atualização de posição no GWO



Fonte: Mirjalili et al., 2014.

O processo de caça dos lobos ocorre quando a presa para de se mover e os lobos a atacam. Para modelar matematicamente a aproximação dos lobos em direção à presa, o valor de $\vec{a}$ é diminuído. O alcance de flutuação de $\vec{A}$ também é afetado com a diminuição de $\vec{a}$, pois $\vec{A}$ é um numéro geradoaleatoriamentecom distribuição uniforme geradono intervalo $[-2a, 2a]$, onde $a$diminui de 2 a 0 à medida em que as iterações ocorrem. Quando valores aleatórios de $\vec{A}$ estão em $[-1, 1]$, a próxima posição de um agente de busca pode ser em qualquer lugar entre sua posição atual

e a posição da presa. Resumindo, se $|A| < 1$, o lobo ataca a presa (MIRJALILI; MIRJALILI; LEWIS, 2014).

Com os operadores apresentados até aqui, os agentes de busca atualizam suas posições de acordo com alfa, beta e delta para atacar a presa. No entanto, com esses operadores, o algoritmo GWO é suscetível à estagnação em soluções locais (MIRJALILI; MIRJALILI; LEWIS, 2014).

Para evitar a estagnação e ampliar a exploração, o autor lança mão da divergência entre lobos e presa. Para modelar a divergência, matematicamente, o autor utiliza valores aleatórios de $\vec{A}$maiores que 1 ou menores que -1 para obrigar os agentes de busca a divergir da presa. Isso aumenta a exploração global do GWO. Em outras palavras, se $|A| > 1$, o lobo cinzento se afasta da presa e procura por uma presa com maior aptidão (MIRJALILI; MIRJALILI; LEWIS, 2014).

Outro componente do GWO que favorece a exploração é o $\vec{C}$, como se vê na equação 4.5. O vetor $\vec{C}$contém valores aleatórios gerados no intervalo$[0, 2]$ e atribui valores aleatórios de peso às presas, para, de forma estocástica, acentuar $(C > 1)$ ou atenuar $(C < 1)$ o efeito da presa em definir a distância, na equação 4.5 (MIRJALILI; MIRJALILI; LEWIS, 2014).

Esses componentes ajudam o GWO a mostrar um comportamento mais aleatório dentro da otimização, favorecendo exploração ampla e prevenção ótima local. O Cnão diminui de forma linear, como o A, com isso os valores de $\vec{C}$acentuam o espaço de exploração não apenas nas iterações iniciais, mas, também, nas iterações finais, sendo útil no caso de estagnação local, em especial, nas iterações finais do processo iterativo (MIRJALILI; MIRJALILI; LEWIS, 2014) e (YANG, BO *et al.*, 2017).

O vetor $\vec{C}$ pode ser considerado como o efeito dos obstáculos que os lobos cinzentos encontram na natureza, durante o processo de caça. Dependendo da posição do lobo, o $\vec{C}$ dá à presa um peso gerado aleatoriamente, podendo dificultar ou facilitar o alcance da presa, pelo lobo (MIRJALILI; MIRJALILI; LEWIS, 2014).

Para mostrar a habilidade teórica do GWO para resolver problemas de otimização, o autor destaca os seguintes pontos:

- A hierarquia social proposta auxilia o GWO a salvar as melhores soluções até o momento, no curso das iterações;
- O mecanismo de cerco proposto define uma vizinhança circular em volta das soluções e podem ser estendidas a maiores dimensões como uma hiperesfera;
- O parâmetros aleatórios$A$ e C ajudam soluções candidatas a terem hiperesferas com diferentes raios;
- O método de caça proposto permite que soluções candidatas localizem a provável posição da presa;
- Exploração ampla (*exploration*) e exploração específica (*exploitation*) são garantidas pelos valores gerados adaptativamente de a e A;
- Com A decrescente, metade das iterações é dedicada à exploração ampla ($|A| \geq 1$) e a outra metade dedicada à exploração específica ($|A| < 1$);
- O GWO tem apenas dois parâmetros de controle para serem ajustados (a e C) (MIRJALILI; MIRJALILI; LEWIS, 2014).

A seguir, o pseudo-código do algoritmo GWO:

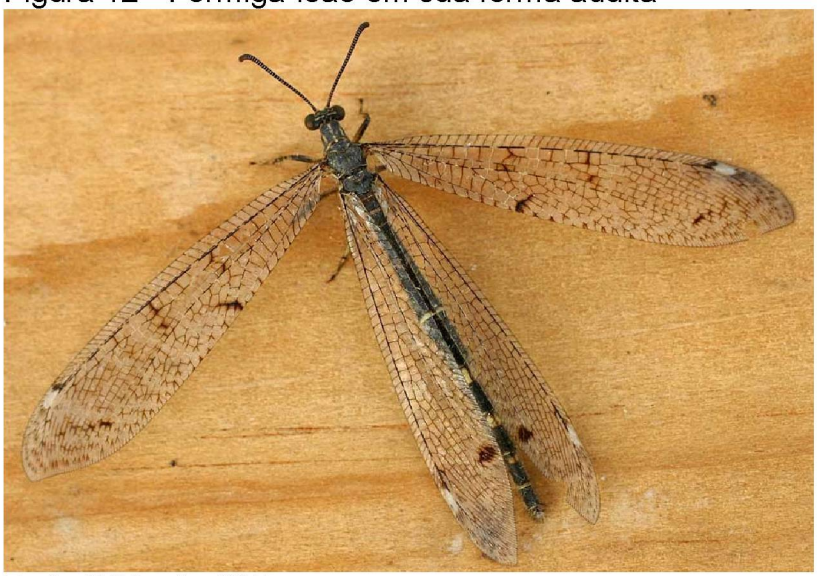
$Inicie\ a\ população\ de\ lobos\ cinzentos\ X_I(i = 1, 2, \ldots, n)$
$Inicie\ a, A\ e\ C$
$Calcule\ a\ aptidão\ de\ cada\ agente\ de\ busca$
$X_\alpha = o\ melhor\ agente\ de\ busca$
$X_\beta = o\ segundo\ melhor\ agente\ de\ busca$
$X_\delta = o\ terceiro\ melhor\ agente\ de\ busca$
$\boldsymbol{enquanto}\ (t < número\ máximo\ de\ iterações)$
    $\boldsymbol{para}\ cada\ agente\ de\ busca$
        $Atualize\ a\ posição\ do\ agente\ de\ busca\ atual\ usando\ a\ equação\ 06$
    $\boldsymbol{finalize\ para}$
    $Atualize\ a, A\ e\ C$
    $Calcule\ a\ aptidão\ de\ todos\ os\ agentes\ de\ busca$
    $Atualize\ X_\alpha,\ X_\beta\ e\ X_\delta$
    $t = t + 1$
$\boldsymbol{finalize\ enquanto}$
$retorne\ X_\alpha$

4.2.2 Otimizador formiga-leão (*Ant Lion Optimizer*, ALO)

O algoritmo ALO foi proposto por Sayedali Mirjalili (2015). É baseado no mecanismo de caça das larvas da Formiga-leão, *Antlion,* em inglês (MIRJALILI, 2015).

Formiga-leão é um inseto da ordem Neuróptera, e da família *Myrmeleontidae* (do grego, *mýrmex* significa formiga e *léon*, leão) (LEVIN, 2001). A Figura 12 apresenta a imagem do formiga-leão na forma adulta.

Figura 12 - Formiga-leão em sua forma adulta



Fonte: Wikipedia, 2015.

Sua larva, inspiração desse algoritmo, tem uma morfologia peculiar, bem diferente de sua forma adulta. A imagem da larva do formiga-leão é apresentada na Figura 13. Possui corpo fusiforme, com abdome robusto, tórax com 3 pares de pernas, prototórax formando um “pescoço” móvel para sua cabeça grande, quadrada e achatada, que acomoda 1 par de grandes mandíbulas em formas de foice, usadas para capturar as presas, na maior parte das vezes, formigas, que caem em sua armadilha (LEVIN, 2001).

Figura 13 - Larva da formiga-leão

Fonte: Wikipedia, 2015.

Para capturar suas presas, a larva da Formiga-leão constrói armadilhas, Figura14, em forma de cone, com diâmetro em torno de 7,5cm e 5cm de profundidade, em terrenos arenosos e se posiciona em seu centro com o corpo enterrado na areia, ficando à mostra, apenas, suas mandíbulas. As armadilhas são construídas com ângulo crítico de repouso, isto é, a maior inclinação para que os grãos de areia não deslizem pela parede da armadilha até seu centro. No entanto, qualquer perturbação mínima, como por exemplo, a presença de uma formiga, pode causar um deslizamento, arrastando a presa para o centro da armadilha (BOTZ *et al.*, 2003).

Figura 14 - Armadilha construída pela larva da formiga-leão

Fonte: Wikipedia, 2015.

Uma vez dentro da armadilha, a larva da formiga-leão tenta capturar a presa, enquanto esta tenta escapar. Para aumentar a chance de captura, a larva de formiga-leão arremessa grãos de areia em direção à presa para ela continuar deslizando para o centro da armadilha. Assim que a larva de formiga-leão consegue agarrar sua presa, ela a leva para dentro da areia e a consome.Em seguida, arremessa os restos para fora da areia e remonta sua armadilha (LEVIN, 2001); (BOTZ *et al.*, 2003).

O ALO mimetiza a relação entre a larva de formiga-leão dentro da armadilha. Para modelar tais interações, formigas precisam se mover sobre um espaço de busca e larvas de formiga-leão são permitidas a caçá-las e aumentarem sua aptidão, usando armadilhas. Uma vez que as formigas se movem de forma estocástica, na natureza, uma caminhada aleatória é escolhida para simular o movimento das formigas, onde

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \dots, cumsum(2r(t_n) - 1)] \qquad (4.8)$$

tal que$cumsum$ calcula a soma cumulativa, n é o número máximo de iteração, $t$mostra o passo de uma caminhada aleatória (iteração, nesse estudo), e $r(t)$ é a função estocástica, definida por

$$r(t) = \begin{cases} 1 \; se \; rand > 5 \\ 0 \; se \; rand \; \leq 5 \end{cases} \tag{4.9}$$

onde$t$ é o passo da caminhada aleatória e $rand$é um número aleatório gerado com distribuição uniforme gerado no intervalo $[0, 1]$(MIRJALILI, 2015).

As posições das formigas são salvas e utilizadas, durante a optimização, na seguinte matriz

$$M_{Ant} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & \dots & A_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \dots \\ A_{n,1} & A_{n,2} & \dots & \dots & A_{n,d} \end{bmatrix} \tag{4.10}$$

SendoM_{Ant}a matriz para salvar a posição de cada formiga. Já A_{ij} mostra o valor da *j*-ésima variável (dimensão) da *i*-ésima formiga, n é o número de formigas e d é o número de variáveis (MIRJALILI, 2015).

As formigas são semelhantes às partículas na Otimização de Enxames de Partículas (PSO) e aos indivíduos no Algoritmo Genético. A posição de uma formiga refere-se aos parâmetros para uma solução particular (MIRJALILI, 2015).

Para avaliar cada formiga, uma função de aptidão é utilizada durante a otimização e os valores de aptidão de todas as formigas são armazenados na seguinte matriz

$$M_{OA} = \begin{bmatrix} f([A_{1,1}, A_{1,2}, \dots, A_{1,d}]) \\ f([A_{2,1}, A_{2,2}, \dots A_{2,d}]) \\ \vdots \\ f([A_{n,1}, A_{n,2}, \dots, A_{n,d}]) \end{bmatrix} \tag{4.11}$$

onde M_{OA} é a matriz para salvar a aptidão de cada formiga, A_{ij}mostra o valor da *j*-ésima dimensão da *i*-ésima formiga, n é o número de formigas e f é a função objetivo (MIRJALILI, 2015).

Em adição às formigas, assume-se que larvas de formiga-leão estão escondidas em algum lugar do espaço de busca. As equações 4.12 e 4.13 representam as matrizes que são utilizadas para salvar suas posições e seus valores de aptidão, respectivamente,

$$M_{Antlion} = \begin{bmatrix} AL_{1,1} & AL_{1,2} & \dots & \dots & AL_{1,d} \\ AL_{2,1} & AL_{2,2} & \dots & \dots & AL_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{n,1} & AL_{n,2} & \dots & \dots & AL_{n,d} \end{bmatrix} \tag{4.12}$$

onde$M_{Antlion}$é a matriz para salvar a posição das larvas de formiga-leão, AL_{ij} mostra o valor da*j*-ésima dimensão da *i*-ésima larva de formiga-leão, n é o número de larvas e d é o número de variáveis (MIRJALILI, 2015). Neste caso adota-se

$$M_{OAL} = \begin{bmatrix} f([AL_{1,1}, AL_{1,2}, \dots, AL_{1,d}]) \\ f([AL_{2,1}, AL_{2,2}, \dots, AL_{2,d}]) \\ \vdots \\ f([AL_{n,1}, AL_{n,2}, \dots, AL_{n,d}]) \end{bmatrix} \tag{4.13}$$

de forma que M_{OAL} é a matriz para salvar o valor de aptidão de cada larva de formiga-leão, AL_{ij} mostra o valor da *j*-ésima dimensão da *i*-ésima larva de formiga-leão, n é o número de larvas e f é a função objetivo (MIRJALILI, 2015).

Durante a otimização, as seguintes condições são aplicadas (MIRJALILI, 2015):

- As formigas se movem pelo espaço de busca, usando diferentes caminhadas aleatórias;
- Caminhadas aleatórias são aplicadas a todas as dimensões de formigas;
- Caminhadas aleatórias são afetadas pelas armadilhas de larvas de formiga-leão;
- Larvas de formiga-leão podem construir armadilhas proporcionais às sua aptidões (quanto maior a aptidão, maior a armadilha);

- Cada formiga pode ser pega por uma larva de formiga-leão em cada iteração e pela elite (larva com maior aptidão);
- O alcance da caminha aleatória é reduzida adaptativamente para simular formigas deslizando em direção às larvas de formiga-leão;
- Se uma formiga ficar com o valor de aptidão maior que o de uma larva de formiga-leão, significa que ela foi capturada e puxada para baixo da areia, pela larva;
- Uma larva de formiga-leão se reposiciona para a última presa capturada e constrói nova armadilha para aumentar sua chance de capturar outra presa, depois de cada caça.

As caminhadas aleatórias das formigas são baseadas na equação 4.8 e suas posições são atualizadas a cada passo da otimização. Uma vez que o espaço de busca tem um limite de variáveis, aquela equação não pode ser usada para atualizar a posição das formigas. Para manter as caminhadas aleatórias dentro do espaço de busca, elas são normalizadas (normalização min-max), tal que

$$X_i^t = \frac{(X_i^t - a_i)(d_i - c_i^t)}{(d_i^t - a_i)} + c_i \tag{4.14}$$

em que a_i é o mínimo da caminhada aleatória da *i*-ésima variável, b_i é o máximo da caminhada aleatória na *i*-ésima variável, c_i^t é o mínimo da *i*-ésima variável na *t*-ésima iteração e d_i^t indica o máximo da *i*-ésima variável na *t*-ésima iteração. Essa equação deve ser aplicada em cada iteração para garantir que as caminhadas aleatórias ocorram dentro do espaço de busca (MIRJALILI, 2015).

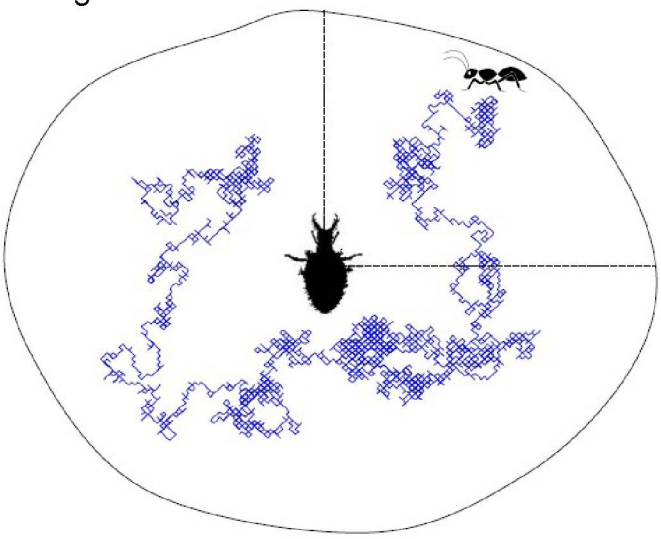
As caminhadas aleatórias são afetadas pelas armadilhas de formiga leão. Para modelar isso, matematicamente, são propostas as seguintes equações

$$\begin{aligned} c_i^t &= Antlion_j^t + c^t \\ d_i^t &= Antlion_j^t + d^t \end{aligned} \tag{4.15}$$

onde c^té o mínimo de todas as variáveis na *t*-ésima iteração, d^t indica o vetor incluindo o máximo de todas as variáveis na *t*-ésima iteração, c_j^t é o mínimo de todas as variáveis para a *i*-ésima formiga, $Antalion_j^t$ mostra a posição *j*-ésima formiga-leão na *t*-ésima iteração (MIRJALILI, 2015).

Essas equações mostram que as formigas andam aleatoriamente em uma hiper esfera definida pelos vetores c e d em volta de uma formiga-leão selecionada. Esse comportamento é ilustrado de forma conceitual pela Figura 15, que representa um espaço de busca bi-dimensional.

Figura 15 - Caminhada aleatória de uma formiga dentro de uma armadilha de formiga-leão



Fonte: Mirjalili, 2015.

Para construir as armadilhas, a habilidade de caça da formiga leão é modelada usando um operador de roleta (usado nos algoritmos genéticos). Como mostrado na Figura 15, as formigas caem apenas na armadilha de uma formiga-leão selecionada. O algoritmo ALO é requerido para utilizar o operador de roleta para selecionar as formigas-leão com base em suas aptidões, durante a otimização. Esse mecanismo aumenta a chance de seleção de formigas-leão com maiores aptidões, para capturar as formigas (MIRJALILI, 2015).

O operador de roleta, também conhecido como seleção proporcional à aptidão, é um operador genético, usado nos algoritmos genéticos, que mimetiza o jogo da roleta dos casinos, em que a função de aptidão atribui um valor de aptidão para as possíveis soluções. O nível de aptidão é usado para associar a probabilidade de

seleção de um indivíduo (solução), numa determinada população. A probabilidade de seleção é dada por

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4.16}$$

de forma que p_i é a probabilidade de seleção de um indivíduo, em uma dada população, f_i é a aptidão do indivíduo e N é o número de indivíduos na população (BÄCK, 1996);(ANGELOVA; ATANASSOV; PENCHEVA, 2012).

Até aqui, foi estabelecido que formigas-leão constroem armadilhas proporcionais às suas aptidões e as formigas se movem aleatoriamente e caem na armadilha de uma formiga-leão. No entanto, ainda há mais uma ação da formiga-leão para capturar sua presa: A formiga-leão arremessa areia, quando percebe que uma formiga caiu na armadilha, para ela deslizar até o centro e não conseguir escapar (MIRJALILI, 2015; BOTZ *et al.*, 2003).

Para modelar esse comportamento, matematicamente, o raio da hiper-esfera das caminhadas aleatórias das formigas diminui, de forma adaptativa, como segue

$$c^t = \frac{c^t}{I} \tag{4.17}$$

$$d^t = \frac{d^t}{I} \tag{4.18}$$

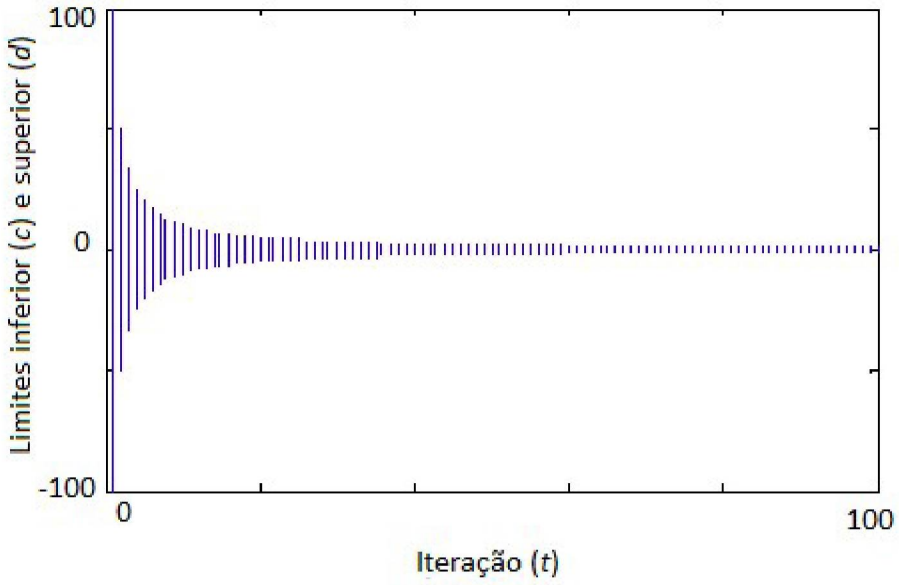
tal que I é uma razão, c^t é o mínimo de todas as variáveis na *t*-ésima iteração, e d^t indica o vetor incluindo o máximo de todas as variáveis na *t*-ésima iteração.

$$I = 10^w \frac{t}{T} \tag{4.19}$$

onde t é a iteração corrente, T é o número máximo de iterações, w uma constante baseada na iteração corrente ($w = 2$, quando $t > 0{,}1T$; $w = 3$, quando $t > 0{,}5T$; $w = 4$, quando $t > 0{,}75T$, $w = 5$, quando $t > 0{,}9T$; e $w = 6$, quando $t > 0{,}95T$). Basicamente, a constante w pode ajustar o nível de acurácia da exploração. A Figura 16 ilustra esse comportamento, onde as equações supracitadas encolhem o

raio da posição das formigas em cada atualização e mimetizam o processo de deslizamento da formiga dentro da armadilha.

Figura 16 - Limites superior (c) e inferior (d) adaptativos



Fonte: Mirjalili, 2015.

O estágio final da caça ocorre quando a formiga chega ao fundo da armadilha e é pega pela mandíbula da formiga-leão. Depois disso, a formiga-leão puxa a formiga para dentro da areia e a consome. Em seguida, reconstrói sua armadilha e se prepara para a próxima presa. Para mimetizar esse processo, assume-se que a captura ocorre quando a aptidão da formiga for maior que o da sua formiga-leão correspondente. Uma formiga-leão é, então, requisitada para atualizar sua posição para a última posição da formiga capturada para aumentar sua chance de capturar a nova presa. Esse modelo é definido como segue

$$Antlion_j^t = Ant_i^t \quad if\ f(Ant_i^t) > f(Antlion_j^t) \tag{4.20}$$

onde t mostra a iteração corrente, $Antlion_j^t$ mostra a posição da *j*-ésima formiga-leão selecionada na *t*-ésima iteração, e Ant_i^t indica a posição da *i*-ésima formiga na *t*-ésima iteração (MIRJALILI, 2015).

O elitismo é uma característica importante dos algoritmos evolucionários que permite-lhes manter as melhores soluções obtidas em qualquer estágio do processo de otimização. Na ALO, a melhores formigas-leão obtidas em cada iteração são salvas e consideradas elite. Uma vez que a elite é a formiga-leão com maior aptidão, ela será capaz de afetar o movimento de todas as formigas, durante as iterações. Por isso, assume-se que cada formiga anda aleatoriamente em volta de uma formiga-leão selecionada pela roleta e pela elite, simultaneamente, conformesegue

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{4.21}$$

ondeR_A^té a caminhada aleatória em torno da formiga-leão selecionada pela roleta, na *t*-ésima iteração, e R_E^t é a caminhada estocásticaem torno da elite, na *t*-ésima iteração, e Ant_i^t indica a posição da *i*-ésima formiga, na *t*-ésima iteração (MIRJALILI, 2015).

Com todos os operadores apresentados acima, o algoritmo pode, agora, ser definido. O algoritmo ALO é definido como uma função de três parâmetros que aproxima o ótimo global para otimização de problemas, expresso como

$$ALO(A, B, C) \tag{4.22}$$

tal que, A é uma função que gera as soluções iniciais aleatórias, B manipula a população inicial providenciada pela função A, e C retorna verdadeiro quando o critério fim é satisfeito. As funções A, B e C, respectivamente, são definidas como

$$\emptyset \xrightarrow{A} \{M_{Ant}, M_{OA}, M_{Antlion}, M_{OAL},\} \tag{4.23}$$

$$\{M_{Ant}, M_{Antlion}\} \xrightarrow{B} \{M_{Ant}, M_{Antlion}\} \tag{4.24}$$

$$\{M, M\} \xrightarrow{C} \{verdadeiro, falso\} \tag{4.25}$$

tal que, M_{Ant} é a matriz da posição das formigas, $M_{Antlion}$ inclui a posição das formigas-leão, M_{OA} contém as aptidões das formigas correspondentes e M_{OAL} contém as aptidões das formigas-leão (MIRJALILI, 2015).

O pseudo-código é definido da seguinte maneira:

Inicie a primeira população de formigas e formigas-leão aleatoriamente
Calcule a aptidão das formigas e formigas-leão
Encontre a melhor formiga-leão e assume-a como Elite (ótimo determinado)
***enquanto** o critério fim não é satisfeito*
    ***para** cada formiga*
        Selecione uma formiga-leão usando roleta
        Atualize c e d
        Crie uma caminhada aleatória e a normalize
        Atualize a posição da formiga
    finalize para
    Calcule a aptidão de todas as formigas
    Realoque uma formiga-leão com sua formiga correspondente se esta tiver maior aptidão
    Atualize Elite se uma formiga-leão tiver aptidão maior que Elite
finalize enquanto
***Retorne** Elite*

No algoritmo ALO, as matrizes de formiga-leão e de formiga são inicializadas aleatoriamente usando a função A. Em cada iteração, a função B atualiza a posição de cada formiga em relação a uma formiga-leão selecionada pelo operador de roleta e a elite. Os limites de atualização de posição são, primeiramente, definidas proporcionalmente ao número vigente de iterações. A posição atualizada é, então, conseguida por duas caminhadas aleatórias, em torno da formiga-leão selecionada e da elite. Quando todas as formigas caminham aleatoriamente, elas são avaliadas pela função de aptidão. Se alguma formiga se tornar mais apta que a alguma formiga-leão, suas posições são consideradas como as novas posições para as formigas-leão, na iteração seguinte. A melhor formiga-leão é encontrada durante a otimização (elite) e substituída, se necessário. Esses passos iteram até que a função C retorne $falsa$ (MIRJALILI, 2015).

O algoritmo ALO foi desenvolvido em ambiente computacional Matlab e disponibilizado como uma otimização de fonte-aberta. Ele permite que seus usuários definam o número de agentes de busca, número máximo de iterações,

número de variáveis, limites inferior e superior das variáveis e o nome da função objetivo, facilmente (MIRJALILI, 2015).

Segundo o autor, teoricamente, o algoritmo ALO consegue aproximar o ótimo global da otimização dos problemas pelas seguintes razões:

- Exploração do espaço de busca é garantida pela seleção aleatória de formigas-leão e caminhadas aleatórias das formigas em volta delas;
- Exploração do espaço de busca é garantida pelo encolhimento adaptativo dos limites das armadilhas das formigas-leão;
- Alta probabilidade de resolver estagnações de ótima locais, graças ao uso de caminhadas aleatórias e da roleta;
- O ALO é um algoritmo baseado em população, então a prevenção de ótimo local é intrinsicamente alto;
- A intensidade de movimento das formigas é diminuída de forma adaptativa, durante o curso das iterações, o que garante a convergência do algoritmo;
- Calcular caminhadas aleatórias para todas as formigas e dimensões promove diversidade na população;
- Formigas-leão são realocadas para a posição das melhores formigas, durante a otimização, logo, áreas promissoras do espaço de busca são salvas;
- Formigas-leão guiam formigas para áreas promissoras no espaço de busca;
- A melhor formiga-leão é salva e comparada com a melhor formiga-leão até então (elite);
- O algoritmo ALO tem poucos parâmetros para ajustar;
- O algoritmo ALO é um algoritmo livre de gradiente e considera o problema como uma caixa preta (MIRJALILI, 2015).

4.2.3 Algoritmo de otimização por enxame de partículas (*Particle Swarm Optimization*, PSO)

O algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization* (PSO)) foi desenvolvido no ano em meados dos anos 1990 no *Research Triangle Intistute* na Carolina do Norte (EUA), por Jim Kennedy e Russell Eberhart. O primeiro

artigo sobre PSO foi *Particle Swarm Optimization* e foi publicado no ano de 1995.O PSO possui conceitos derivados do campo de vida artificial e engenharia. Seu desenvolvimento foi inspirado no comportamento social de bandos de pássaros, cardumes de peixes e na teoria de “enxames”. Possui correlação com os paradigmas dos algoritmos genéticos e também outrosalgoritmos evolutivos. sAnteriormente ao desenvolvimento do PSO outros estudos com diversas simulações computacionais e diferentes interpretações sobre comportamento social de bandos de pássaros ou cardumes de peixes haviam sido desenvolvidos, esses modelos estavam fortemente baseados na distância entre indivíduos no grupo (KENNEDY; EBERHART, 1995).

Uma motivação para desenvolver o algoritmo foi o interesse em modelar comportamento social dos humanos, que difere da modelagem de animais, pois adentramos a uma área adicional, de conhecimento cognitivo, de maneira que dois humanos podem estar alinhados tendo a mesmas crenças e atitudes. Quando tratamos de animais o interesse se dá simplesmente no meio físico, o objeto de estudo é posição que o indivíduo ocupa no meio, a maneira que se movimentam para buscar alimentos, se proteger de predadores, entre outros (KENNEDY; EBERHART, 1995).

Os desenvolvedores do PSO são um psicologista social e um engenheiro eletricista e chegam à conclusão que PSO é útil para ambos os campos de estudo e concluem o motivo de comportamento social ser tão presente no reino animal, é porque isso otimiza. Em contrapartida uma boa maneira de solucionar problemas de otimização em engenharia é usando a modelagem de comportamentos social (KENNEDY; EBERHART, 1995).

O PSO mostrou-se um algoritmo simples e efetivo dentre um grande número de funções. Seu comportamento mostrou-se realístico, com “vida”, fato que não ocorreu para versões anteriores em que os movimentos pareciam artificiais (KENNEDY; EBERHART, 1995).

(VENTER; SOBIESZCZANSKI-SOBIESKI, 2003) abordaram o PSO e discutiram suas características a fim de propor melhoramentos ao algoritmo. Os operadores e parâmetros serão apresentados a partir do trabalho apresentado pelos autores supracitados.

O algoritmo é automatizado seguindo a ordem a seguir:

1. Inicialmente, uma população é iniciada atribuindo aleatoriamente a cada partícula sua velocidade e sua localização em um dado espaço.
2. É calculado o vetor de velocidade para cada uma das partículas do enxame.
3. Atualiza-se a posição de cada partícula com base no seu vetor de posição anterior e no seu vetor de velocidade atualizado no passo anterior.
4. Caso a convergência não seja atingida, se retorna ao passo 2. (VENTER; SOBIESZCZANSKI-SOBIESKI, 2003).

Para atualizar a posição de cada partícula, segue-se

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i}\Delta t \tag{4.26}$$

onde, x_{k+1}^{i} é a posição da partícula i na iteração $k+1$ e v_{k+1}^{i} é o vetor de velocidade correspondente. Um passo de unidade de tempo Δt é utilizado (VENTER; SOBIESZCZANSKI-SOBIESKI, 2003).

A equação para atualizar a velocidade de cada partícula depende do algoritmo PSO usado em cada passo. Uma equação comumente adotada é a seguinte:

$$v_{k+1}^{i} = w v_{k}^{i} + c_1 r_1 \frac{(p^{i} - x_{k}^{i})}{\Delta t} + c_2 r_2 \frac{(p_{k}^{g} - x_{k}^{i})}{\Delta t} \tag{4.27}$$

tal que, r_1 e r_2 são números aleatóriosindependentes entre 0 e 1, p^{i} é a melhor posição encontrada pela partícula i, até o momento, e p_{k}^{g} é a melhor posição no enxame no tempo k.Há três parâmetros de controle no problema: a inércia da partícula w e dois parâmetros de "confiança" c_1 e c_2. As velocidades são limitadas pelos parâmetros $v_{máx}$, velocidade máxima, e $v_{mín}$, velocidade mínima (VENTER; SOBIESZCZANSKI-SOBIESKI, 2003).

A inércia controla as propriedades de exploração, com valores maiores, facilita um comportamento mais amplo, enquanto que, com valores menores, um comportamento local é facilitado. Os parâmetros de confiança indicam quando uma partícula acredita em si mesma, em c_1, e quanto de confiança ela tem no enxame, em c_2 (VENTER; SOBIESZCZANSKI-SOBIESKI, 2003).

O enxame inicial é gerado de tal forma que as partículas se distribuam aleatoriamente pelo espaço de busca, cada uma com um vetor de velocidade inicial aleatório. As seguintes equações são usadas para obter a posição e a velocidade inicial, respectivamente,

$$x_0^i = x_{min} + r_3(x_{max} - x_{min}) \tag{4.28}$$
$$v_0^i = \frac{x_{min} + r_4(x_{max} - x_{min})}{\Delta t}$$

ondex_0^i é o vetor de posição inicial, v_0^i é o vetor de velocidade inicial da partícula i, r_3 e r_4 são números "independentes"gerados entre 0 e 1, x_{min} é o vetor dos limites mínimos e x_{max} é o vetor dos limites máximos para as variáveis em questão (VENTER; SOBIESZCZANSKI-SOBIESKI, 2003).

O pseudo-código do PSO é definido da seguinte maneira:

Inicie PSO
    Inicie enxame de partículas
    Avalie partículas e atualize melhor solução global
    ***Enquanto** iterações*
    Move partículas
    Avalia partículas
        Atualiza melhor solução
        Atualize melhor solução global
Enquanto (iterações)
Fim enquanto
Fim PSO
Retorne Elite

4.2.4 Algoritmo de otimização por enxame de partículas Darwiniano (*Darwinian Particle Swarm Optimization*, DPSO)

Um problema comumente encontrado no PSO e outros algoritmos de otimização é o fato de ficar preso em um ótimo local. Na tentativa de superar esse problema muitos autores têm sugerido outros ajustes aos parâmetros do PSO combinando a lógica *Fuzzy* (FAPSO) onde o peso da inércia w é dinamicamente ajustado utilizando as regras "SE-SENÃO" do *Fuzzy* (HARNRNOUCHE; DIAF; SIARRY, 2010) ou aproximações gaussianas (GPSO) onde a constante da inércia w não é mais necessária e as constantes ρ_1 , ρ_2 e ρ_3 são substituídas por números aleatórios com distribuição Gaussiana (JIANG, M.; LUO; YANG, 2007). Mais recentemente (SOLTEIRO PIRES *et al.*, 2010) utilizou cálculo fractional para controlar a taxa de convergência do PSO. Os autores rearranjaram a equação de velocidade original para modificar a ordem da velocidade derivativa. Alternativamente, muitos autores têm considerado incorporar seleção, mutação e cruzamento, tal como na Evolução Diferencial, no algoritmo PSO. O objetivo foi aumentar a diversidade da população tanto evitando que as partículas se movam muito próximas umas das outras, quanto auto-adaptando parâmetros tais como o fator de constrição, constantes de aceleração, ou constantes de inércia (GHAMISI *et al.*, 2012).

Na procura por um modelo de seleção natural utilizando o algoritmo PSO, o PSO Darwiniano (DPSO) foi formulado (TILLETT; M. RAO; *et al.*, 2005), no qual vários enxames de soluções de teste podem existir em qualquer momento. Cada enxame se comporta individualmente tal qual o algoritmo PSO comum, com regras governando a coleção de enxames que são projetadas para simular a seleção natural.

Apesar das similaridades entre PSO e Algoritmos Genéticos (GAs), como a população gerada de forma aleatória, função de avaliação de *fitness*, atualização da população, a procura por soluções ótimas com técnicas estocásticas e a não garantia de sucesso; o PSO não utiliza operadores genéticos, como mutação e cruzamento, portanto continua não sendo considerada uma técnica evolucionária. Por outro lado, o PSO Darwiniano (DPSO) extende o PSO para determinar se a

seleção natural (Princípio Darwiniano de sobrevivência do mais adaptado) pode melhorar a habilidade do PSO de escapar de um ótimo local.

A ideia é rodar vários algoritmos PSO paralelos simultaneamente, cada um com um enxame diferente, dentro do mesmo problema de teste e com um mecanismo de seleção simples sendo aplicado. Quando a pesquisa tender para um ótimo local, a pesquisa nessa área é simplesmente descartada e então outra área é procurada. Nessa aproximação, a cada passo, os enxames que dão melhores resultados são recompensados (extendendo o tempo de vida da partícula ou é gerado um novo descendente) e os enxames stagnados são punidos (reduz o tempo de vida do enxame ou deleta partículas). Para analizar o estado geral de cada enxame, o *fitness* de todas as partículas é avaliado e a vizinhança e a posição com o melhor resultado individual de cada partícula são atualizados. Se uma nova solução global é encontrada, uma nova partícula é gerada. Uma partícula é deletada se o enxame falhar na procura por um estado com melhor *fitness* em um número de passos definido (GHAMISI *et al.*, 2012).

Loop principal:

Enquanto não encontra condição de parada
    Para cada enxame na coleção
        Evolua o enxame
        Permite ao enxame criar ou remover partículas
        Remove os enxames "falhos"
    Fim para
Retorna o enxame com o melhor resultado

Algoritmo de evolução do enxame:

Para cada partícula no enxame
    Atualiza o fitness da partícula
    Atualiza a melhor posição da partícula
    Move a partícula
Se o resultado do enxame melhorar
    O enxame é recompensado
Se o resultado do enxame não melhorar
    O enxame é punido
Retorna o enxame

Algumas regras são seguidas para se deletar um enxame, deletar partículas, e criar um novo enxame ou novas partículas:

I. Quando a população do enxame cai abaixo de um limite mínimo, o enxame é deletado;
II. E, a partícula com o pior desempenho em um enxame é deletada quando a quantidade de passos (Contador de passos SC_C^{max}) supera um limiar máximo sem uma melhora no resultado do *fitness*.

Após a deleção da partícula, ao invéz de ser armazenado como zero, o contador é reiniciado para um valor aproximado do limiar, de acordo com a equação 4.29, onde N_{kill} é o número de partículas deletadas do enxame dentro de um período onde não houve melhora no *fitness*. Para criar um novo enxame, um enxame não pode ter nenhuma partícula deletada em momento algum e o número máximo de enxames não pode ser excedido. Entretanto, o novo enxame é criado apenas com a probabilidade $p = f/NS$, o de f é um número aleatório entre [0,1] e NS é o número de enxames. Esse fator evita a criação de novos enxames quando já existe um grande número de enxames sendo processados (GHAMISI *et al.*, 2012), tal que

$$SC_c(N_{kill}) = SC_C^{max}\left[1 - \frac{1}{N_{kill}+1}\right] \quad (4.29)$$

O enxame 'pai' não é afetado e metade das particulas do mesmo são selecionadas de forma aleatória para comporem o enxame 'filho', a outra metade das partículas do enxame 'filho' é selecionada, também de forma aleatória, de um membro da coleção de enxames. Se o número inicial da populaççao do enxame não é obtida, então o restante das partículas é aleatóriamente inicializada e adicionada ao novo enxame. Uma partícula é criada sempre que um enxame atingir um ótimo global e a quantidade máxima definida de partículas não foi atingida. Assim como o PSO, alguns parâmetros também precisam ser ajustados de forma a rodar o algoritmo de forma eficiente:

I. A população inicial do enxame;
II. A população máxima e mínima do enxame;
III. O número inicial de enxames;
IV. O número máximo e mínimo de enxames;
V. O limiar de estagnação.

Em problemas de estimação previamente estudados por (DEL VALLE *et al.*, 2008) e estratégias de exploração robótica (ALRASHIDI; EL-HAWARY, 2009) o DPSO foi

bem sucedido em comparação com o PSO, exibindo um desempenho superior (GHAMISI *et al.*, 2012).

4.2.5 Algoritmo de otimização por enxame de partículas Darwiniano de ordem fracionária (*Fractional-Order Darwinian Particle Swarm Optimization,* FO-DPSO)

O algoritmo FO-DPSO foi apresentado por (COUCEIRO, M; GHAMISI, 2015) é uma extensão do DPSO no qual o cálculo fracionário é utilizado para controlar a taxa de convergência. O cálculo fracional tem atraído a atenção de diversos pesquisadores (TENREIRO MACHADO *et al.*, 2010); (IONESCU; SABATIER; MACHADO, 2012) sendo aplicado em várias áreas, tais como engenharia, matemática computacional, mecânica dos fluídos, entre outros (COUCEIRO, MICAEL S.; FONSECA FERREIRA; TENREIRO MACHADO, 2010). A implementação discreta no tempo da definição de Grünwald-Letnikov baseada no conceito da diferencial fracional com $\alpha \in C$ do sinal $x(t)$, é dada por:

$$D^{\alpha}[x(t)] = \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^{k} \Gamma(\alpha+1) x(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \quad (4.30)$$

onde Γé a função gama, *T* é o período de amostragem e*r* é a ordem de truncagem (GHAMISI *et al.*, 2012).

Uma importante propriedade revelada por Grünwald-Letnikov é que enquanto uma derivada de ordem inteira apenas implica em uma séria finita, a ordem fracionária requer um infinito número de termos. Sendo assim, derivativos inteiros são operadores 'locais' enquanto derivativos fracionários possuem, implicitamente, a 'memória' de todos os eventos passados. Contudo, a influência dos eventos passados diminui com o tempo (GHAMISI *et al.*, 2012).

As características reveladas pelo cálculo fracionário fazem dessa ferramenta matemática muito bem situada para descrever fenômenos tais como irreversibilidade e caos por causa de sua propriedade de memória inerente (GHAMISI *et al.*, 2012) e (WANG, YUAN-YUAN *et al.*, 2019).

Considerando a influência inercial da equação básica do PSO, assumindo *T* = 1 e similaridades com o trabalho de (SOLTEIRO PIRES *et al.*, 2010), a pressão a seguir pode ser definida:

$$D^{\alpha}[v_{t+1}^{n}] = \rho_1 r_1(\breve{g}_t^n - x_t^n) + \rho_2 r_2(\breve{x}_t^n - x_t^n) + \rho_3 r_3(\breve{n}_t^n - x_t^n) \quad (4.31)$$

Os resultados preliminares dos testes realizados por (COUCEIRO; FONSECA FERREIRA; TENREIRO MACHADO, 2010) apresentaram resultados similares para $r \geq 4$. Além disso, os requisitos computacionais aumentam linearmente de acordo com *r*, isto é, o FO-DPSO apresenta uma complexidade em memória de $\theta(r)$. Por conseguinte, utilizando apenas os primeiros $r = 4$ termos da diferencial dada por 4.30 e a função básica do PSO pode ser reescrita por:

$$v_{t+1}^{n} = \alpha_t^n + \frac{1}{2}\alpha_{t-1}^n + \frac{1}{6}\alpha(1-\alpha)v_{t-2}^n + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)v_{t-3}^n + \rho_1 r_1(\breve{g}_t^n - x_t^n) + \rho_2 r_2(\breve{x}_t^n - x_t^n) + \rho_3 r_3(\breve{n}_t^n - x_t^n) \quad (4.32)$$

O DPSO é então considerado como sendo um caso de FO-DPSO quando $\alpha = 1$ (sem 'memória'). Consequentemente, o valor de α afeta largamente as particulas inerciais. Com um pequeno α, as partículas vão ignorar as suas atividades anteriores, ignorando assim a dinâmica do sistema e ficando sucetíveis a ficarem presas em soluções locais. Por outro lado, com um grande α, as partículas vão apresentar um comportamento mais diversificado permitindo explorar novas soluções, melhorando assim o desempenho à longo prazo. Entretanto, se o nível de exploração é muito alto, então o algoritmo pode levar muito tempo para encontrar a solução global. Baseado nos resultados experimentais de (COUCEIRO; GHAMISI, 2015).

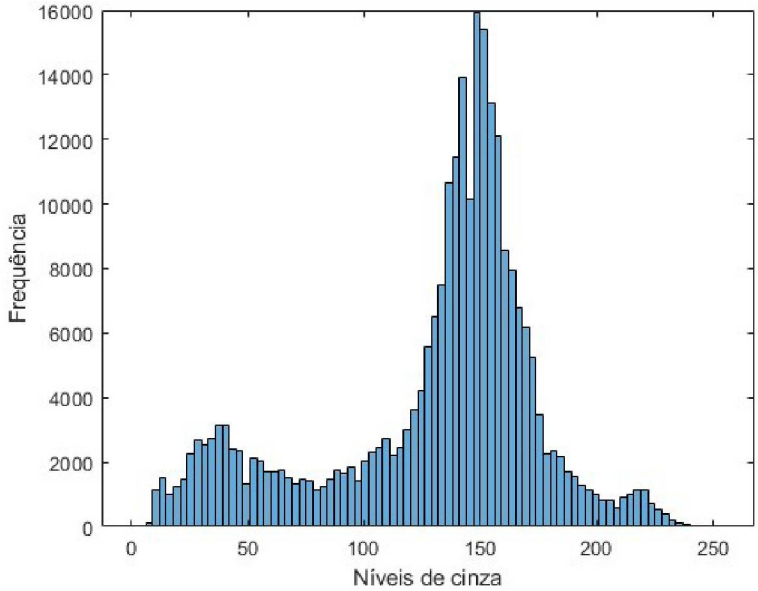
5. IMAGENS DE TESTE

Nestecapítulo as imagens utilizadas para a análise do desempenho dos algoritmos estudados são apresentadas.As imagens utilizadas nos experimentos são apresentadas nas Figuras 17, 19, 21 e 23 e seus respectivos histogramas, nas Figuras 18, 20, 22 e 24, respectivamente. As primeiras duas imagens possuem resolução de 512x512 *pixels*, com 256 níveis de cinza obtidas a partir do USC-SIPI (University of Southern California - *Signal and Image Processing Institute*). Enquanto as imagens seguintes possuem resolução de 2400x2400 e 3274x3274 respectivamente, também com 256 níveis de cinza, tendo sido capturadas via satélite.

Figura 17 - Imagem Barcos

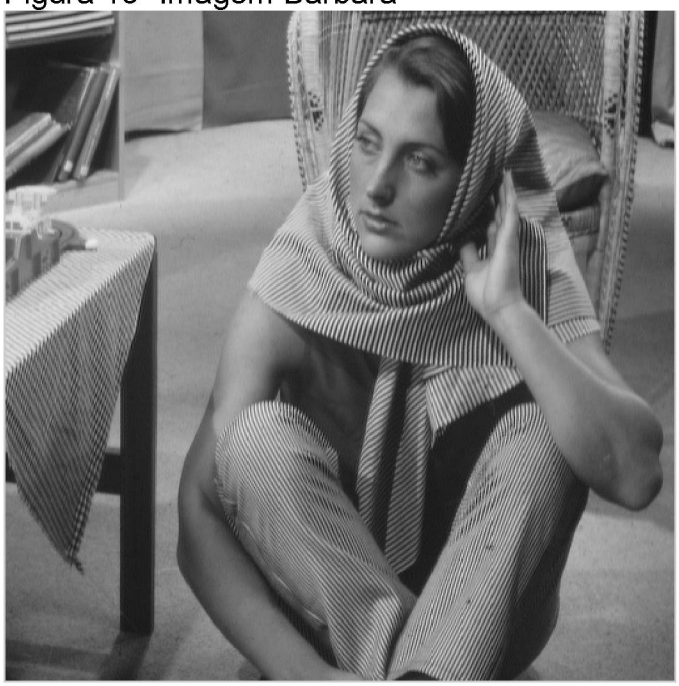
Fonte: SIPI, 2015.

Figura 18 - Histograma da imagem 'Barcos'



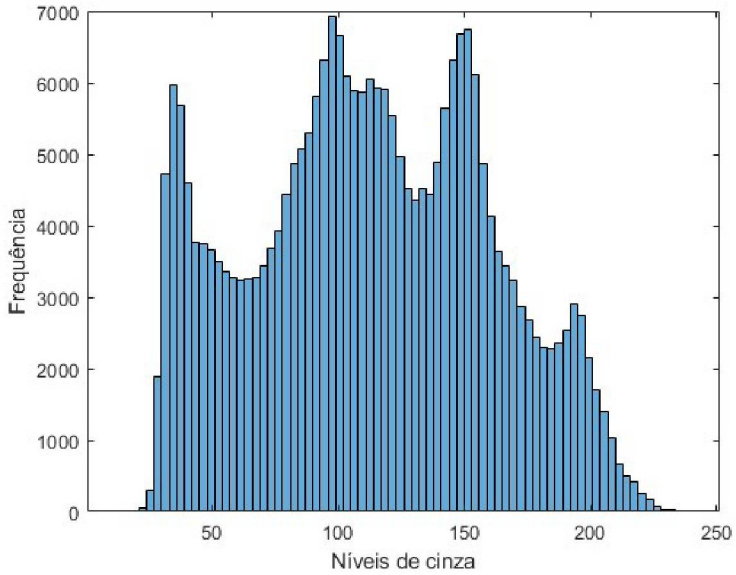
Fonte: O autor, 2018.

Figura 19- Imagem Bárbara



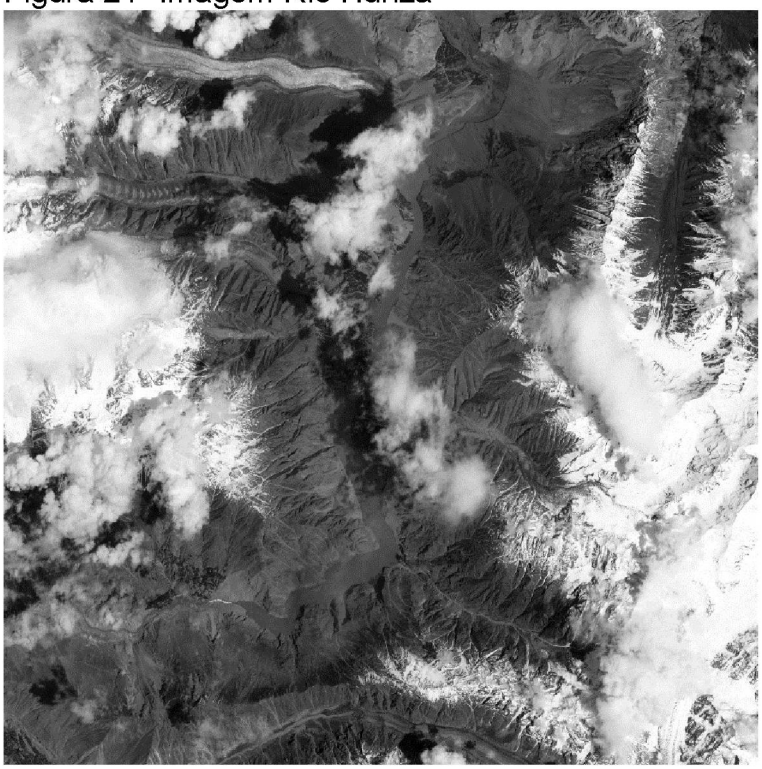
Fonte: EECS, 2015.

Figura 20 - Histograma da imagem 'Bárbara'



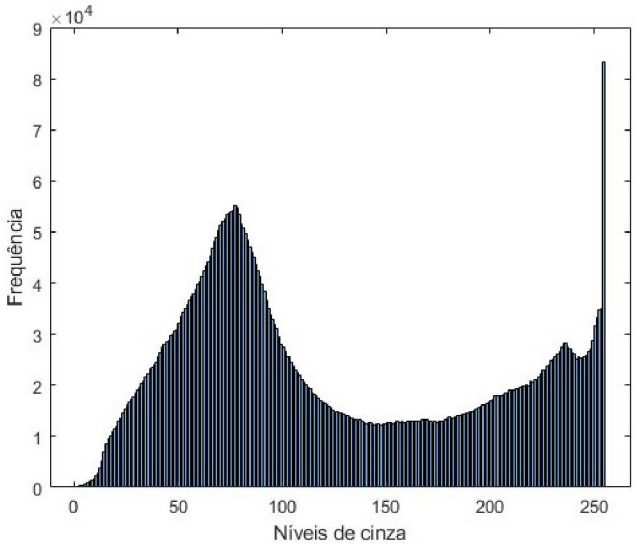
Fonte: O autor, 2018.

Figura 21- Imagem Rio Hunza



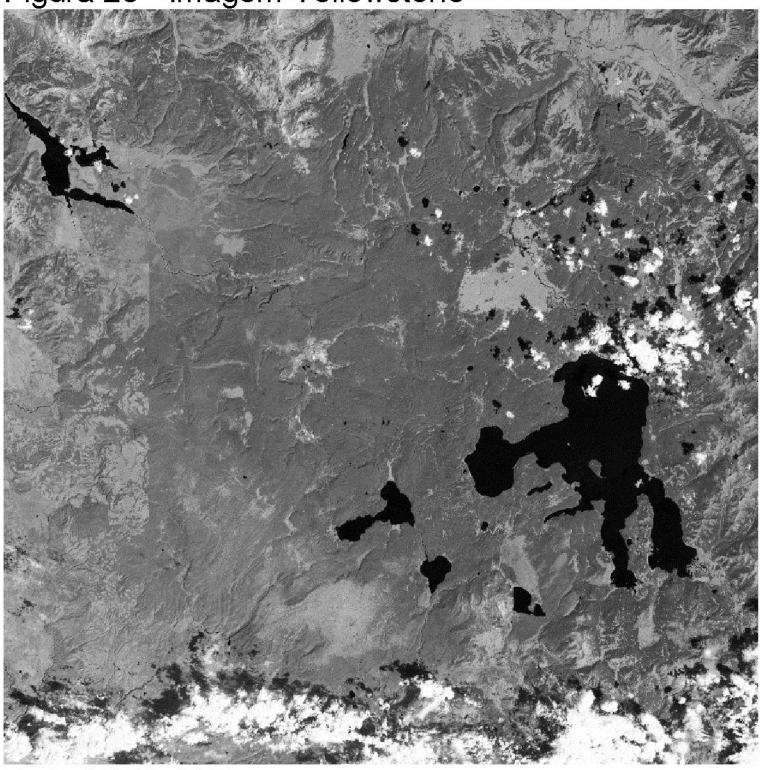
Fonte: NASA, 2015.

Figura 22 - Histograma da imagem 'Rio Hunza'



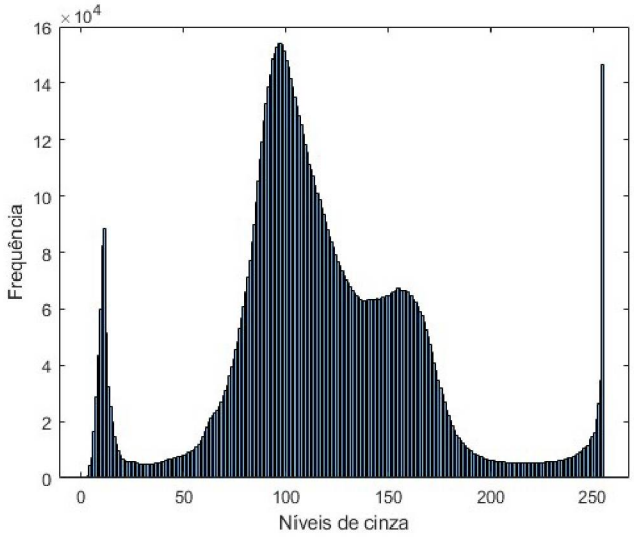
Fonte: O autor, 2018.

Figura 23 - Imagem Yellowstone



Fonte: NASA, 2015.

Figura 24 - Histograma da imagem 'Yellowstone'



Fonte: O autor, 2018.

No histograma apresentado na Figura 18 pode-se observar a existência de três grupos de objetos, sendo um grupo mais explícito, contido entre os níveis 100 e 200 aproximadamente, e outros dois podendo ser o fundo da imagem e alguns detalhes.

No histograma apresentado na Figura 20 a divisão entre grupos se apresenta de forma mais clara, explicitando a existência de quatro grupos bem definidos de objetos na imagem correspondente.

Nos histogramas apresentados nas Figuras 22 e 24 pode-se observar a existência de grupos de objetos diferentes, mas a divisão entre tais grupos não é evidente, por causa da existência de níveis (de cinza) próximos com intensidades semelhantes.

6. ANÁLISE DE RESULTADOS

Nesta seção, os resultados experimentais são apresentados. Os experimentos foram realizados em ambiente computacional MATLAB e o objetivo é comparar o desempenho das meta-heurísticasna determinação dos limiares que maximizam a expressão de limiarização multinível de Kapur,descritopela equação 3.11, para diferentes imagens.

Os códigos computacionais das meta-heurísticas estão disponíveis nos endereços de *internet*[1]que acompanham suas referências. O código (*script*) da função objetivo foi gerado como uma função do ambiente computacional MATLAB e possui como entrada os valores de limiares fornecidos pelas meta-heurísticas e o vetor do histograma da imagem. A saída da função é o valor de entropia, calculada a partir da equação 3.11.Cada algoritmo foi executado 30 vezes, com os mesmos parâmetros de controle, para cada caso de teste. Esses parâmetros foram definidos na próxima seção deste trabalho.

6.1 DETERMINAÇÃO DOS PARÂMETROS DE CONTROLE

Para determinar os parâmetros de controle a serem utilizados nos algoritmos de inteligência de enxema foram executados testes variando os valores dos mesmos. Tais testes visam identificar o valor mais apropriado para cada parâmetro de controle tendo em vista a influência do parâmetro no tempo de execução do algoritmo e na entropia resultante nas imagens teste.

6.1.1 Otimizador formiga leão (ALO)

Para o algoritmo otimizador formiga leão foram testadas variações em dois parâmetros, sendo eles: quantidade de agentes e quantidade de gerações. Adotando os seguintes valores para os parâmetros:

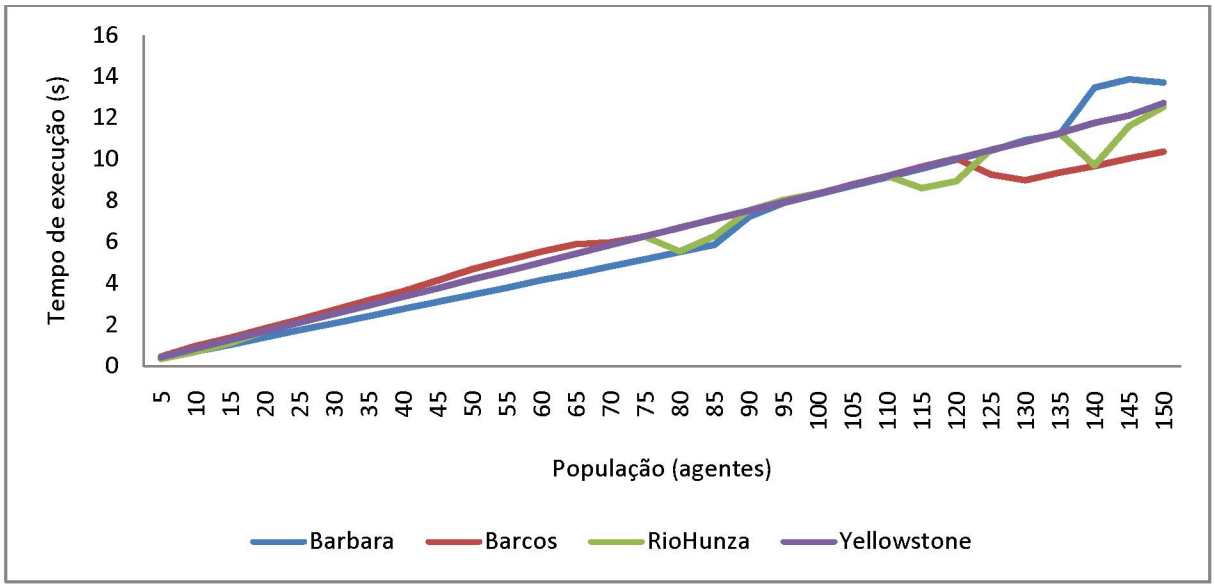
---

[1] *https://www.mathworks.com/matlabcentral/fileexchange/31570-finding-dominant-peaks-and-valleys-of-an-image-histogram*
https://www.mathworks.com/matlabcentral/fileexchange/29517-segmentation
http://www.alimirjalili.com/GWO.html
http://www.alimirjalili.com/ALO.html

- *Quantidade de agentes*: 5 a 150, com um passo de 5 entre cada valor (sendo assim foram testados os valores: 5, 10, 15, ..., 140, 145 e 150);
- *Quantidade de gerações*: 10 a 300, com um passo de 10 entre cada valor.

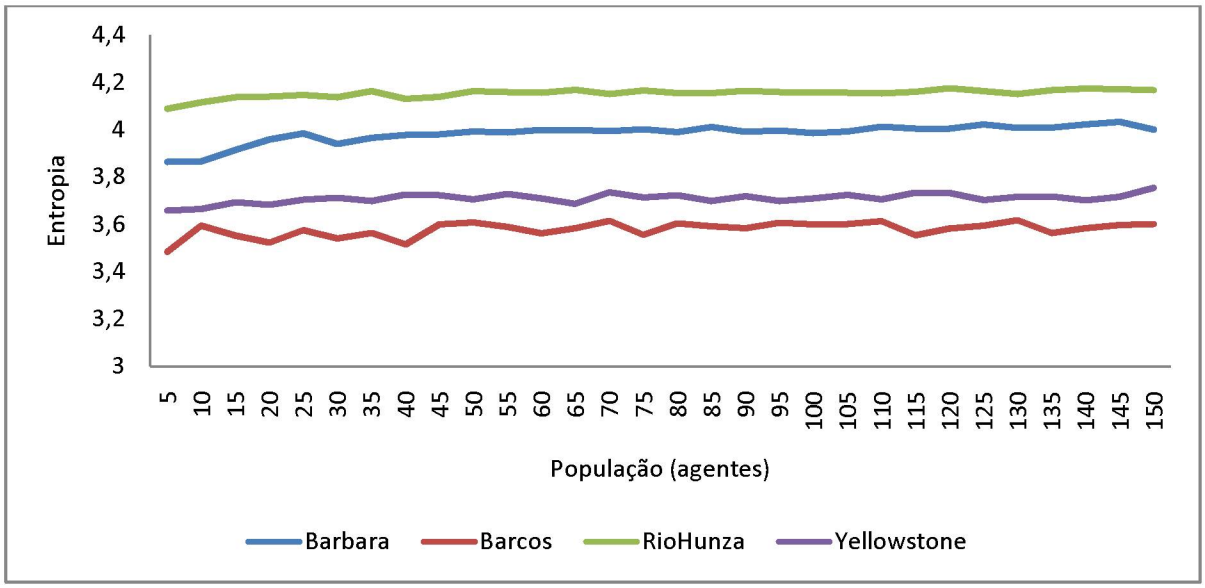
A Figura 25 apresenta a variação do tempo de execução para as diferentes quantidades de agentes adotadas.

Figura 25 - Tempo de execução para diferentes quantidades de agentes do algoritmo ALO



Na Figura 26 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de agentes adotadas.

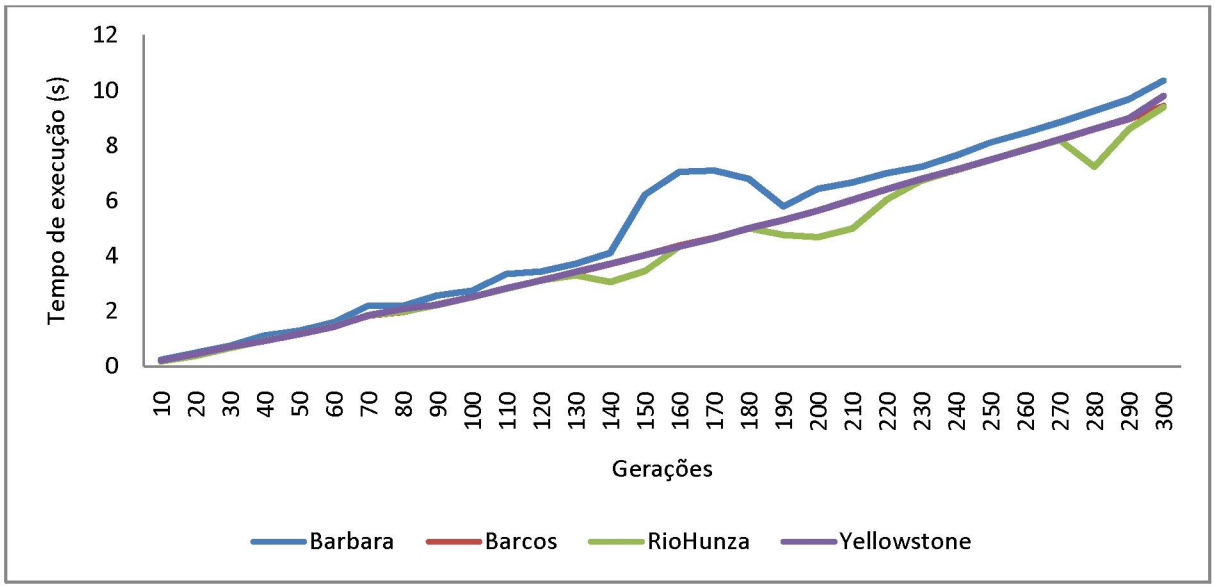
Figura 26 - Entropia resultante para diferentes quantidades de agentes do algoritmo ALO



A partir dos resultados obtidos nas Figuras25 e 26 pode-se concluir que o tempo de execução é diretamente afetado pela quantidade de agentes, mas a entropia sofre pouca variação, principalmente para mais de 10 agentes, sendo assim a quantidade de agentes escolhida é 10.

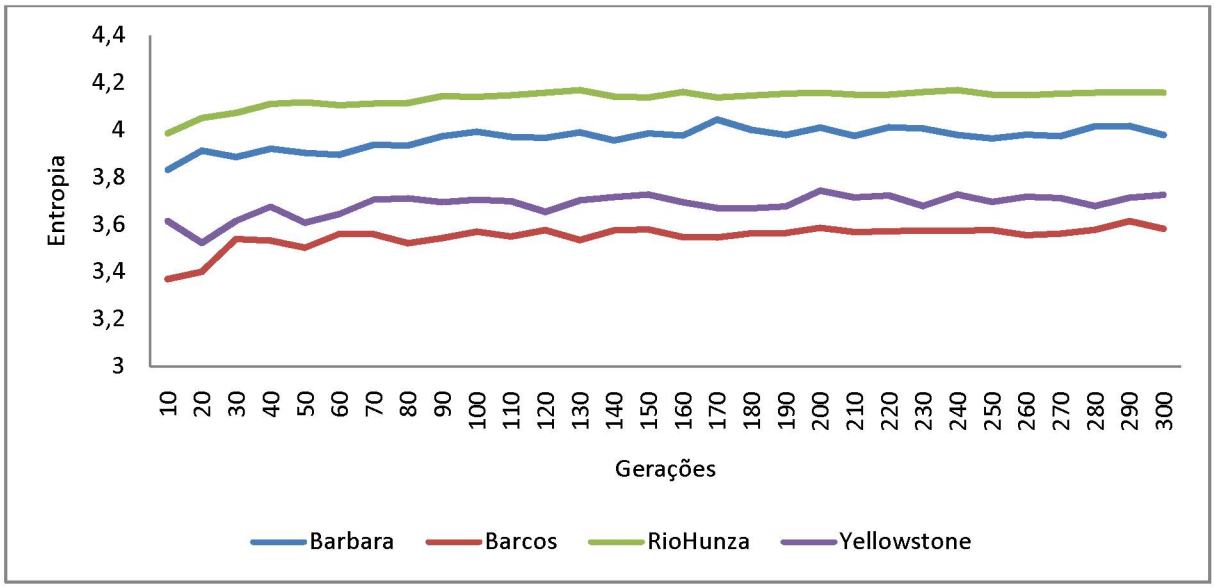
A Figura 27 apresenta a variação do tempo de execução para as diferentes quantidades de gerações adotadas.

Figura 27 - Tempo de execução para diferentes quantidades de gerações do algoritmo ALO



Na Figura 28 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de gerações adotadas.

Figura 28 - Entropia resultante para diferentes quantidades de gerações do algoritmo ALO



Pelos resultados obtidos nas Figura 27 e 28 nota-se que o tempo de execução é diretamente afetado pela quantidade de gerações, mas a entropia sofre pouca variação, principalmente para mais de 30 gerações, sendo assim a quantidade de gerações escolhida é 30.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador formiga leão foram escolhidos:

- *Quantidade de agentes*: 10;
- *Quantidade de gerações*: 30.

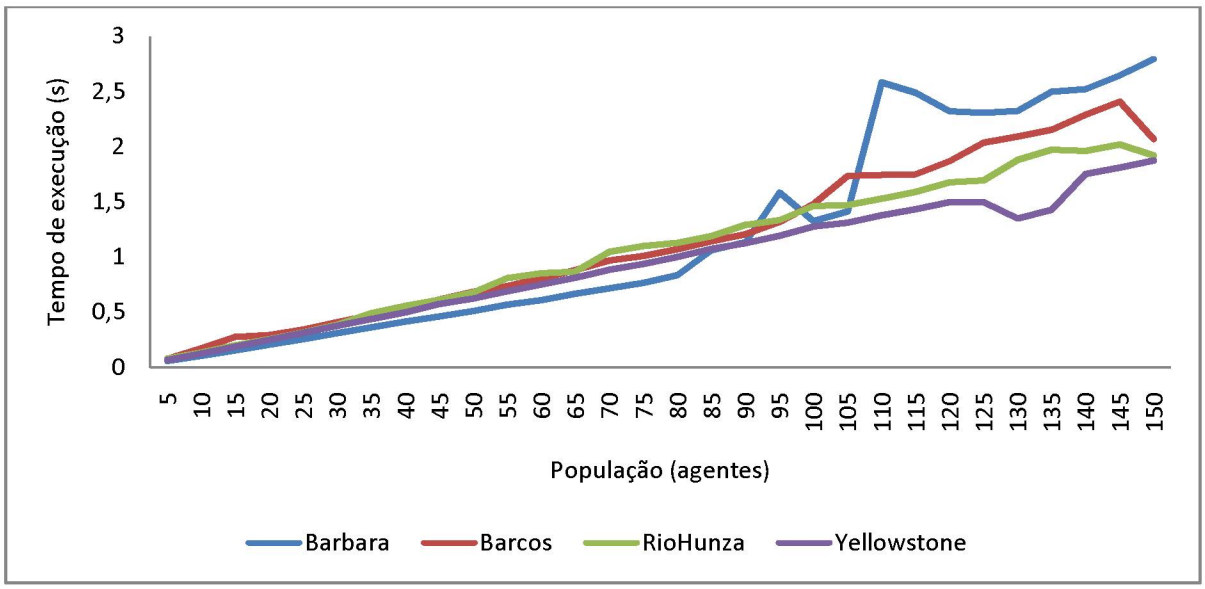
6.1.2 Otimizador lobo cinzento (GWO)

Para o algoritmo otimizador lobo cinzento foram testadas variações em dois parâmetros, sendo eles: quantidade de agentes e quantidade de gerações. Adotando os seguintes valores:

- *Quantidade de agentes*: 5 a 150, com um passo de 5 entre cada valor;
- *Quantidade de gerações*: 10 a 300, com um passo de 10 entre cada valor.

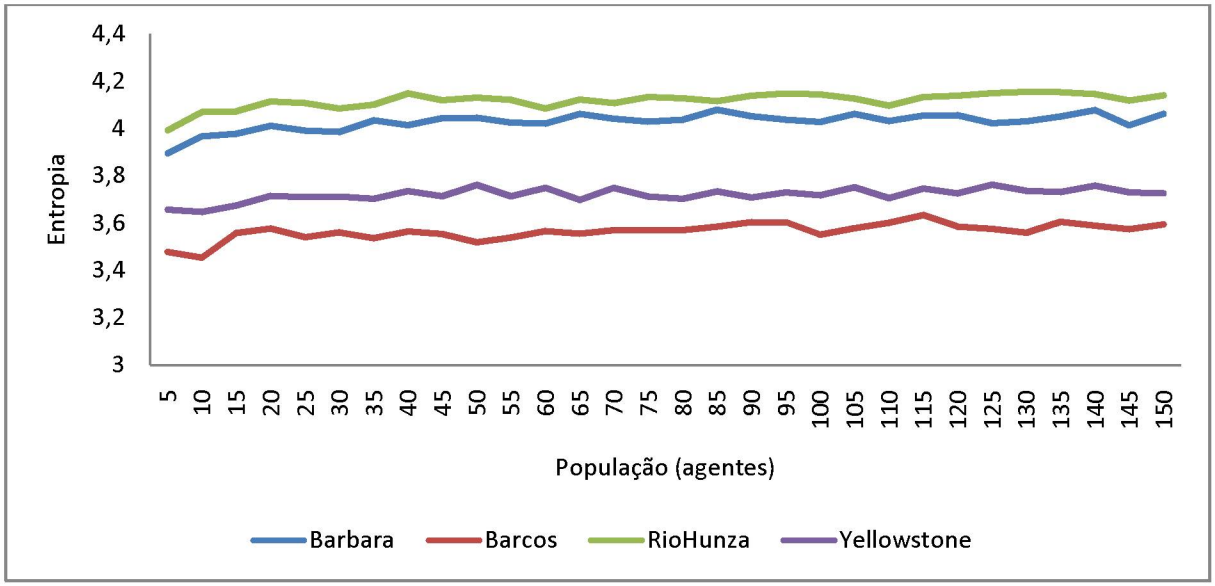
A Figura 29 apresentaa variação do tempo de execução para as diferentes quantidades de agentes adotadas.

Figura 29 - Tempo de execução para diferentes quantidades de agentes do algoritmo GWO



Na Figura 30 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de agentes adotadas.

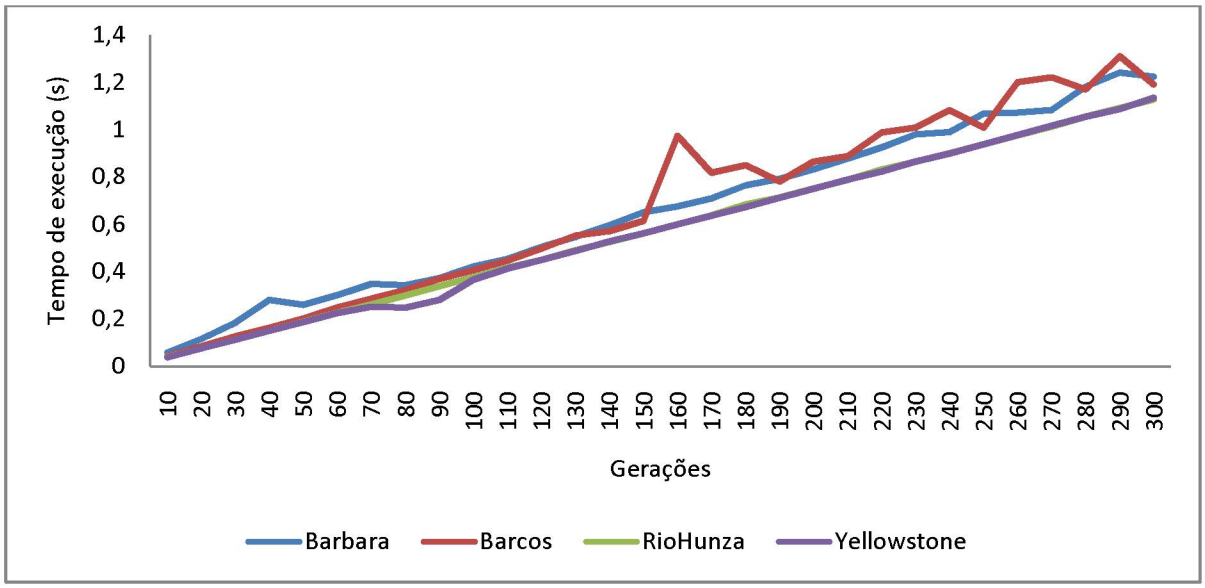
Figura 30 - Entropia resultante para diferentes quantidades de agentes do algoritmo GWO



Pelos resultados obtidos nas Figuras 29 e 30 observa-se que o tempo de execução é diretamente afetado pela quantidade de agentes, mas a entropia sofre pouca variação, principalmente para mais de 15 agentes, sendo assim a quantidade de agentes escolhida é 15.

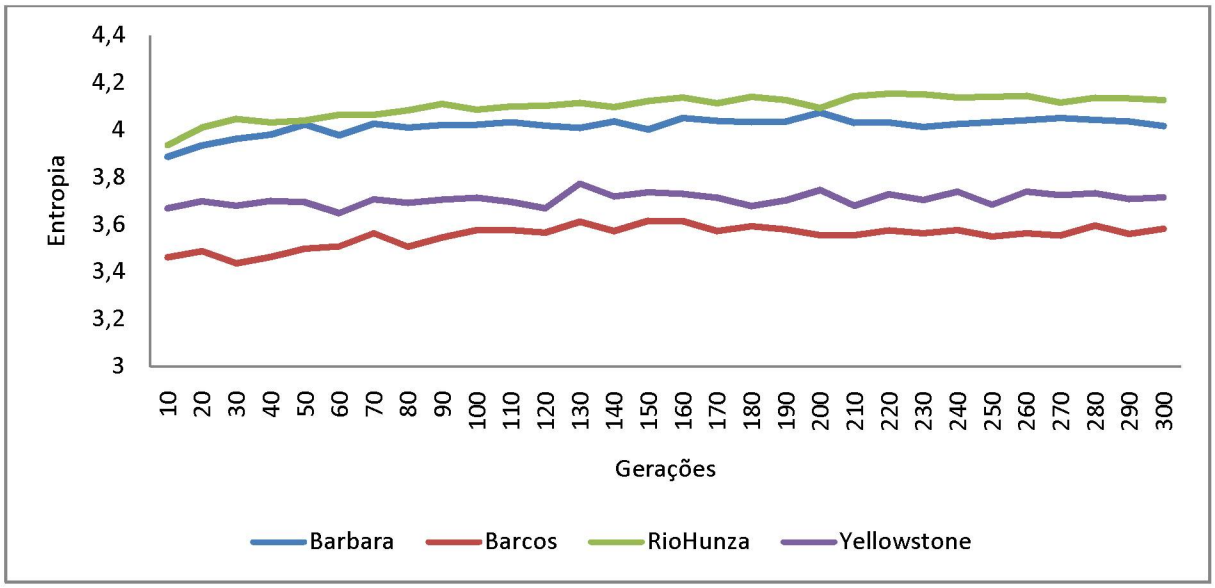
A Figura 31 apresenta a variação do tempo de execução para as diferentes quantidades de gerações adotadas.

Figura 31 - Tempo de execução para diferentes quantidades de gerações do algoritmo GWO



Na Figura 32 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de gerações adotadas.

Figura 32 - Entropia resultante para diferentes quantidades de gerações do algoritmo GWO



O tempo de execução é diretamente afetado pela quantidade de gerações, mas a entropia sofre pouca variação, principalmente para mais de 20 gerações, sendo assim a quantidade de gerações escolhida é 20, conforme pode ser observado nas Figuras 31 e 32.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador lobo cinzento foram escolhidas as seguintes quantidades:

- *Quantidade de agentes*: 15;
- *Quantidade de gerações*: 20.

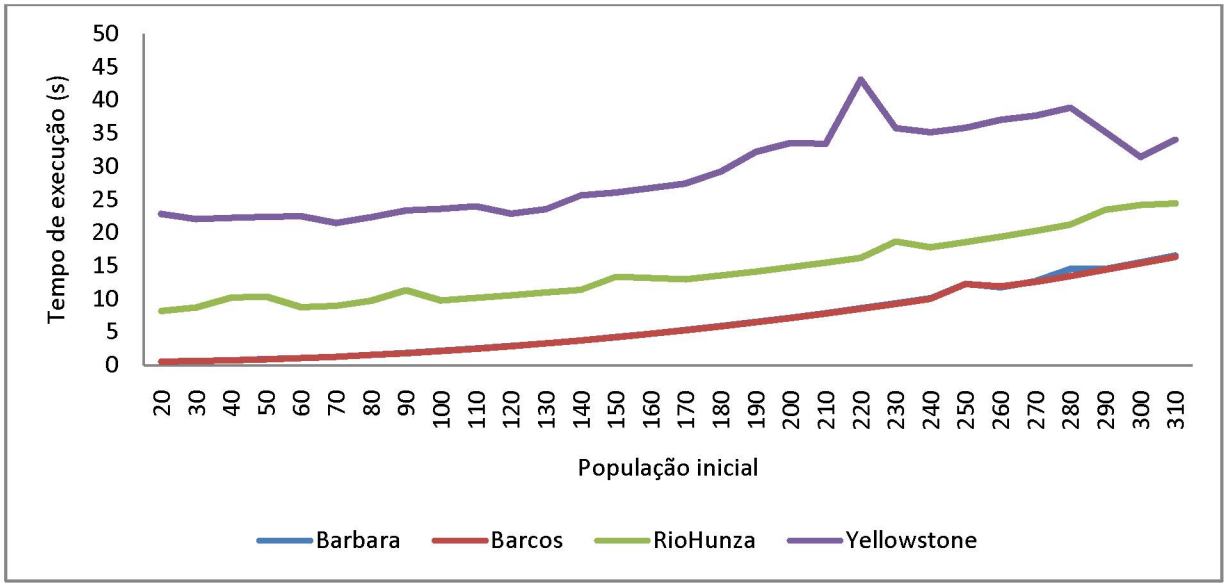
6.1.3 Otimizador por enxame de partículas (PSO)

Para o algoritmo otimizador por enxame de partículas foram testadas variações em quatro parâmetros, sendo eles: população inicial, peso individual, peso social e fator de inércia. Adotando os seguintes valores:

- *População inicial*: 20 a 310, com um passo de 10 entre cada valor;
- *Peso individual*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Peso social*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Fator de inércia*: 0,1 a 3, com um passo de 0,1 entre cada valor.

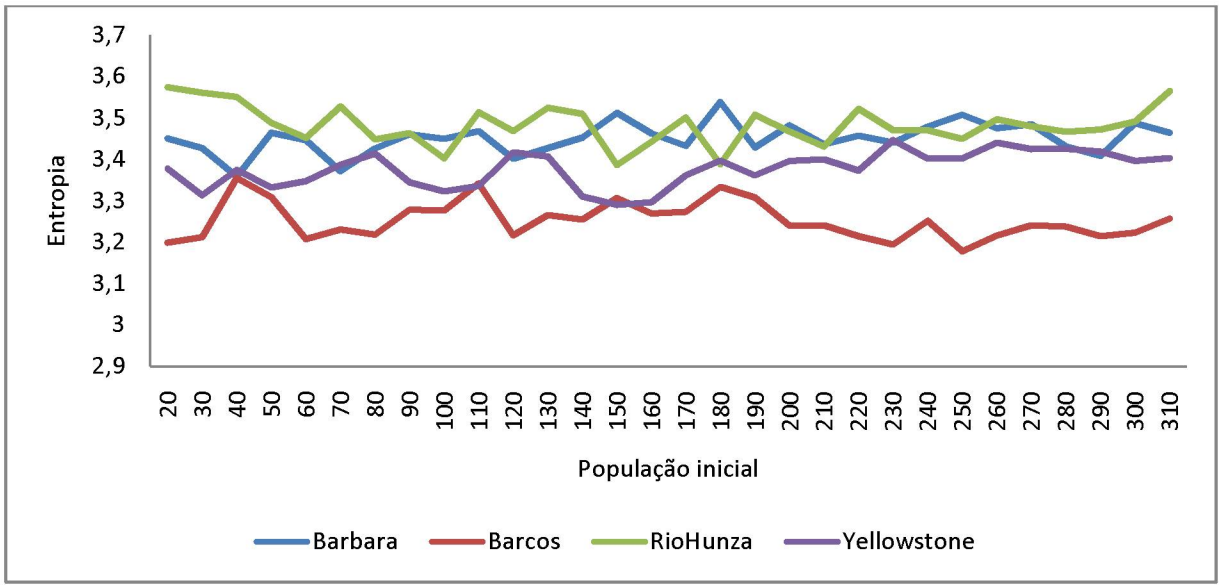
A Figura 33 apresenta a variação do tempo de execução para as diferentes populações iniciais adotadas.

Figura 33 - Tempo de execução para as diferentes populações iniciais do algoritmo PSO



Na Figura 34 é apresentada a variação da entropia das imagens resultantes para as diferentes populações iniciais adotadas.

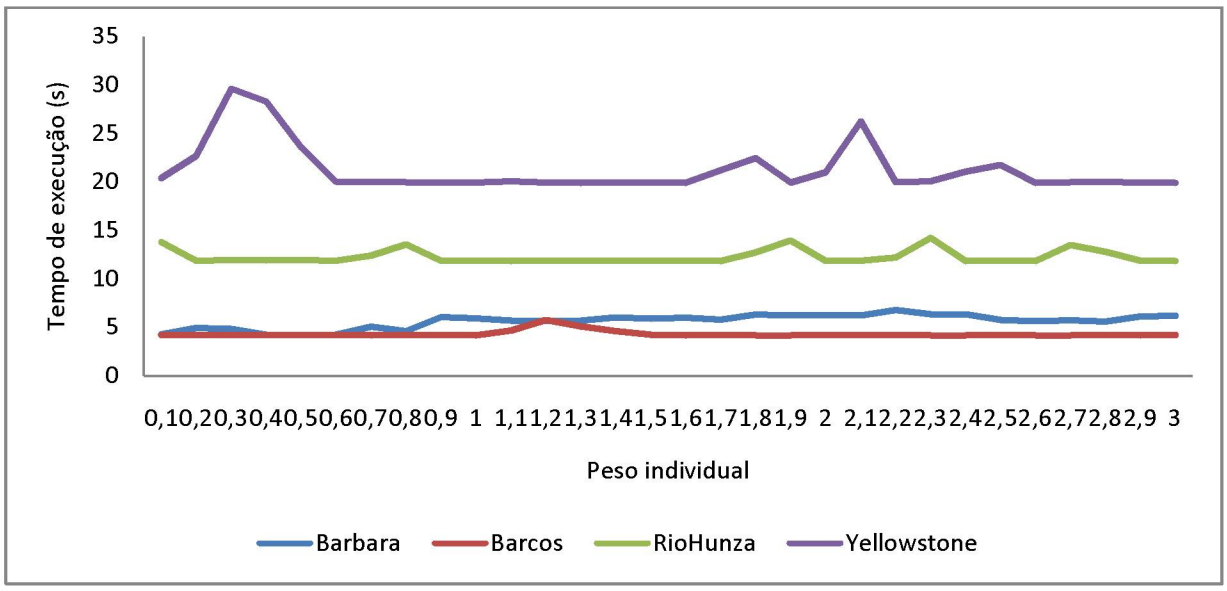
Figura 34 - Entropia das imagens resultantes para as diferentes populações iniciais do algoritmo PSO



Nota-se pelas Figuras 33 e 34 que o tempo de execução é diretamente afetado pela população inicial, mas a entropia sofre variação pouco acentuada, principalmente para mais de 40 partículas, sendo assim a população inicial escolhida é 40.

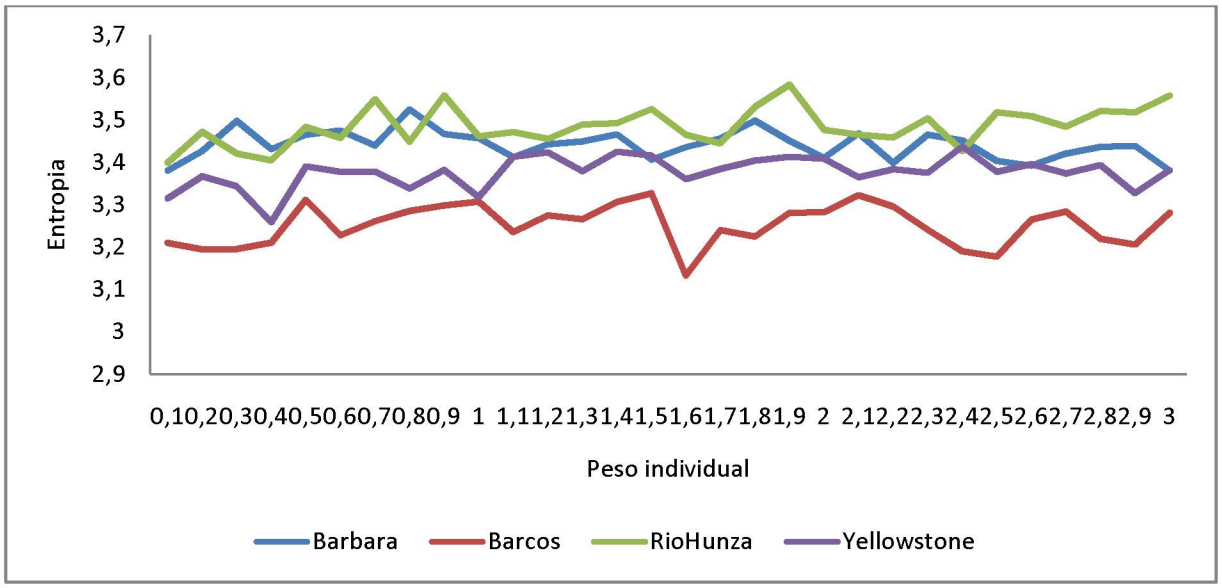
A Figura 35 apresenta a variação do tempo de execução para os diferentes pesos individuais adotados.

Figura 35 - Tempo de execução para os diferentes pesos individuais utilizados pelo algoritmo PSO



Na Figura 36 é apresentada a variação da entropia das imagens resultantes para os diferentes pesos individuais adotados.

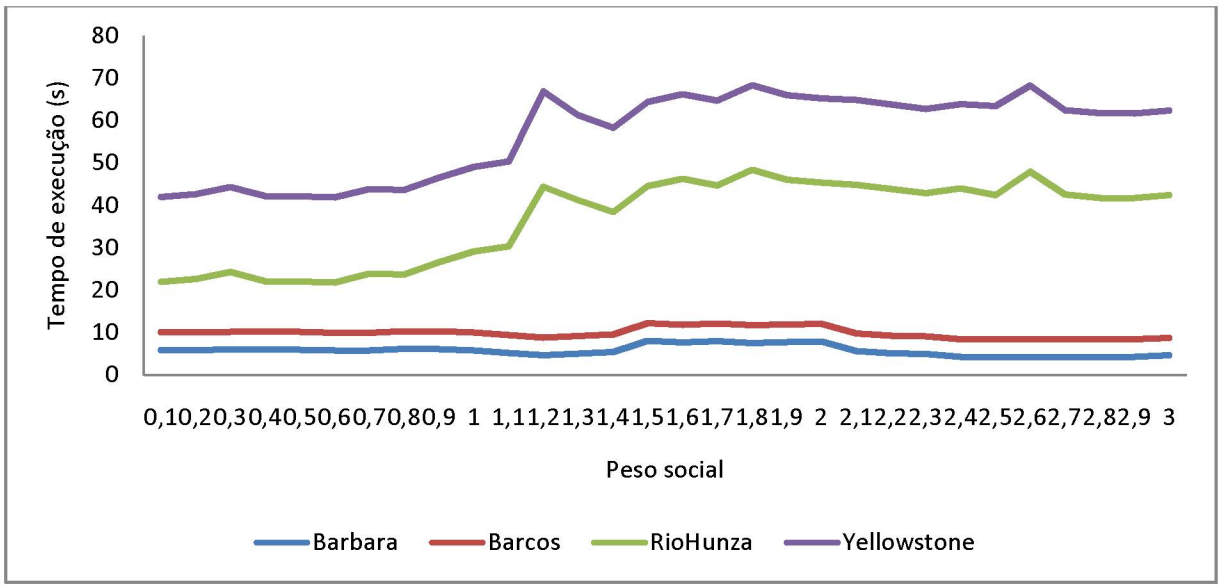
Figura 36- Entropia das imagens resultantes para os diferentes pesos individuais utilizados pelo algoritmo PSO



Observa-se pelas Figuras 35 e 36 quetanto o tempo de execução quanto a entropia resultante são pouco afetados pelos diferentes pesos individuais aplicados, sendo assim o peso individual das particulas escolhido é 0,8, pois o mesmo possui resultados semelhantes em otimização do tempo de execução e entropia.

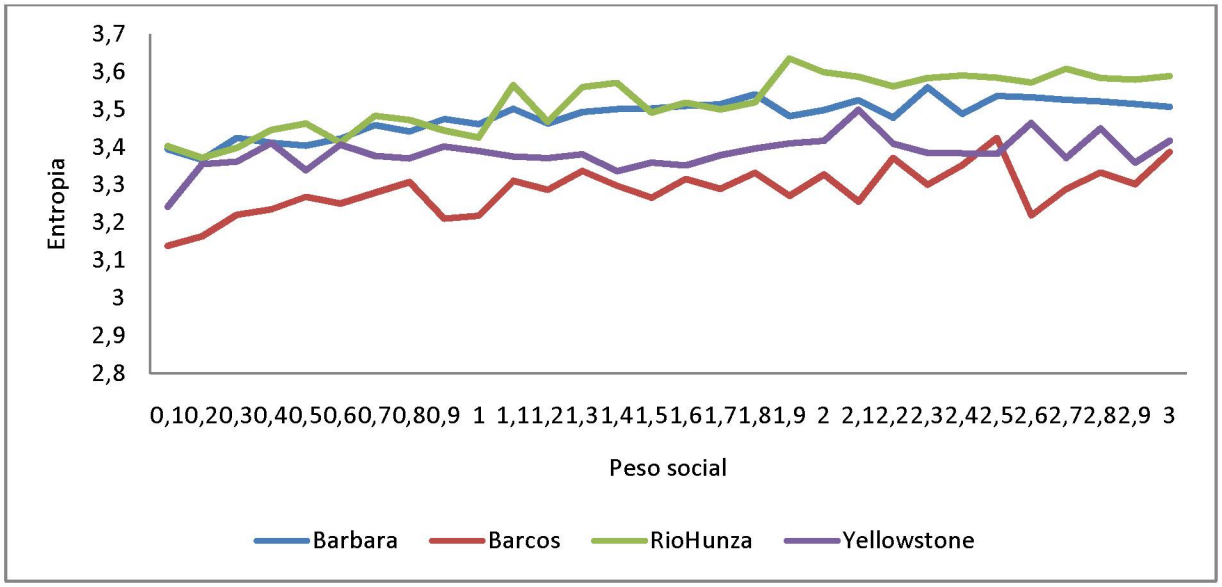
A Figura 37 apresenta a variação do tempo de execução para os diferentes pesos sociais adotados.

Figura 37 - Tempo de execução para os diferentes pesos sociais utilizados pelo algoritmo PSO



Na Figura 38 é apresentada a variação da entropia das imagens resultantes para os diferentes pesos sociais adotados.

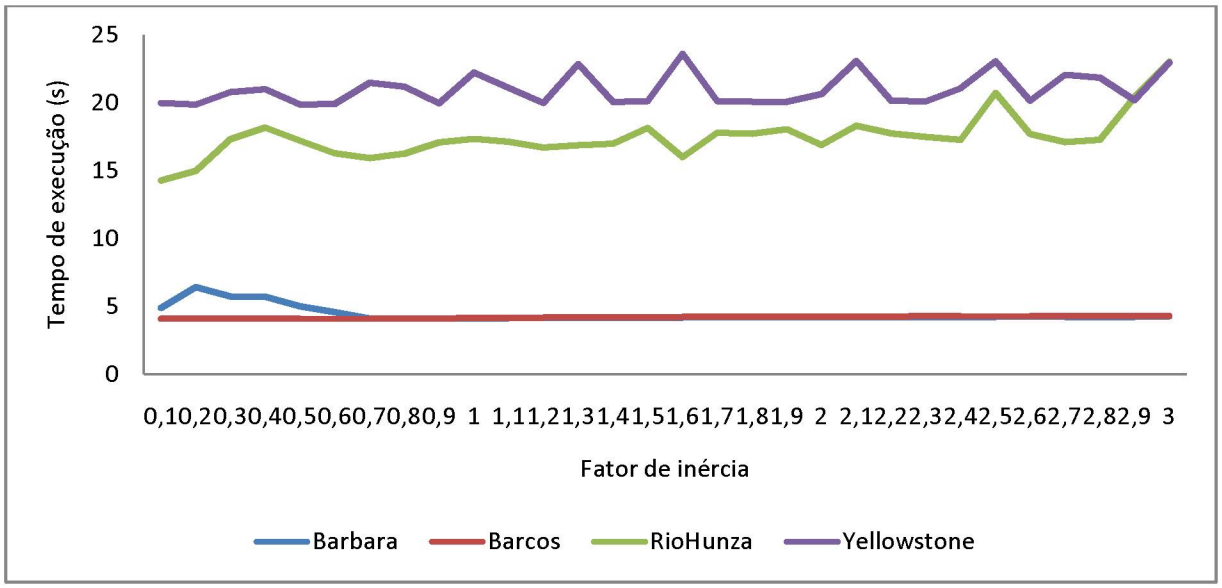
Figura 38- Entropia das imagens resultantes para os diferentes pesos sociais utilizados pelo algoritmo PSO



A partir dos resultados obtidos nasFiguras 37 e 38 pode-se concluir que o tempo de execução é diretamente afetado pelos diferentes pesos sociais aplicados, mas a entropia sofre pouca variação, principalmente para pesos sociais maiores que 0,8, sendo assim o peso social escolhido é 0,8.

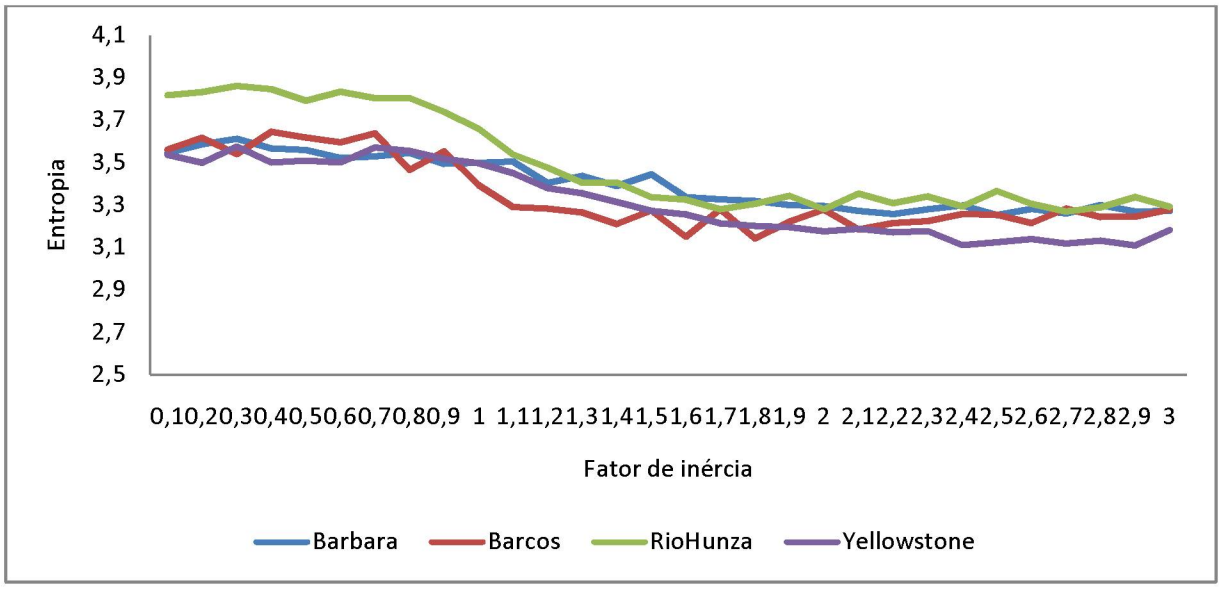
A Figura 39 apresenta a variação do tempo de execução para os diferentes fatores de inércia adotados.

Figura 39 - Tempo de execução para os diferentes fatores de inércia utilizados pelo algoritmo PSO



Na Figura 40 é apresentada a variação da entropia das imagens resultantes para os diferentes fatores de inércia adotados.

Figura 40- Entropia das imagens resultantes para os diferentes fatores de inércia utilizados pelo algoritmo PSO



A partir dos resultados apresentados nas Figuras 39 e 40 pode-se concluir que a entropia é diretamente afetada pelos diferentes fatores de inércia aplicados, mas o tempo de execução sofre pouca, ou nenhuma, variação, o fator de inércia com o melhor resultado tanto em relação ao tempo de execução quanto à entropia resultante é 0.7, sendo assim este é o valor escolhido.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador por enxame de partículas foram escolhidos:

- *População inicial*: 40;
- *Peso individual*: 0,8;
- *Peso social*: 0,8;
- *Fator de inércia*: 0,7.

6.1.4 Otimizador por enxame de partículas Darwiniano (DPSO)

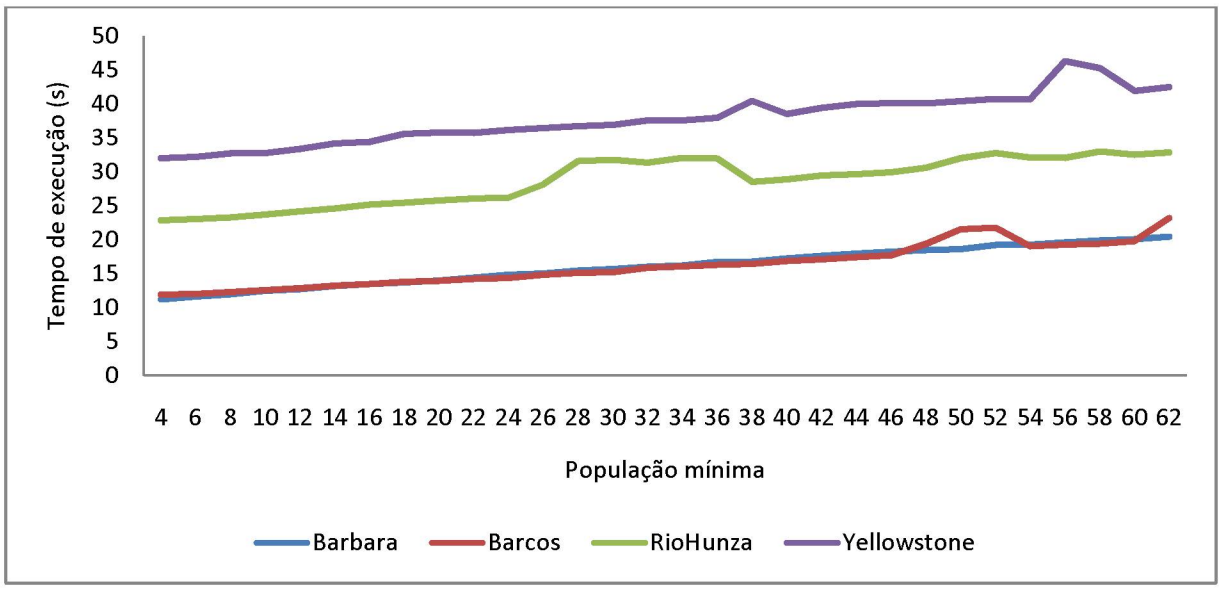
Para o algoritmo otimizador por enxame de partículas darwiniano foram testadas variações em onze parâmetros, sendo eles: população inicial, população mínima, variação das populações, enxames iniciais, enxames mínimos, variação da quantidade de enxames, tempo de estagnação, peso individual, peso social, fator de inércia e quantidade de gerações. Adotando os seguintes valores:

- *População inicial*: 20 a 310, com um passo de 10 entre cada valor;
- *População mínima*: 4 a 62, com um passo de 2 entre cada valor;
- *Variação das populações*: 10 a 155, com um passo de 5 entre cada valor;
- *Enxames iniciais*: 2 a 31, com um passo de 1 entre cada valor;
- *Enxames mínimos*: 2 a 31, com um passo de 1 entre cada valor;
- *Variação da quantidade de enxames*: 2 a 31, com um passo de 1 entre cada valor;
- *Tempo de estagnação*: 2 a 31, com um passo de 1 entre cada valor;
- *Peso individual*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Peso social*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Fator de inércia*: 0,1 a 3, com um passo de 0,1 entre cada valor.
- *Quantidade de gerações*: 10 a 300, com um passo de 10 entre cada valor.

Devido ao algoritmo ser derivado do otimizador por enxame de partículas, ficou claro nos testes executados que a variação dos parâmetros de controle em comum (população inicial, peso individual, peso social e fator de inércia) possui o mesmo comportamento em ambos os algoritmos. Por esse motivo os valores dos parâmetros de controlecomuns adotados para ambas as meta-heurísticas de otimização são os mesmos.

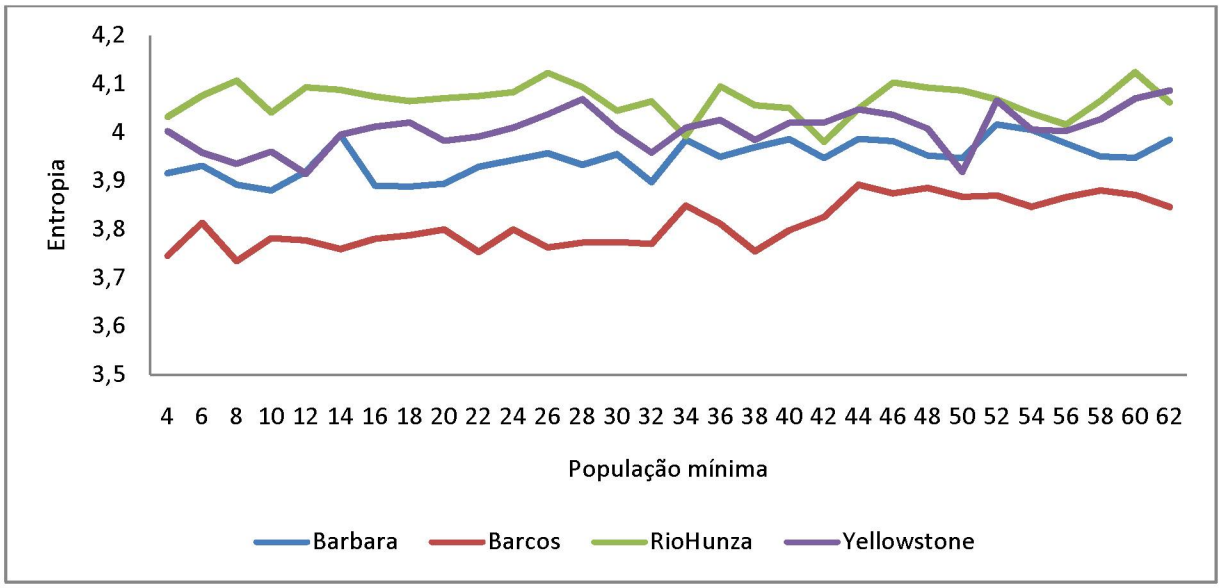
A Figura 41 apresenta a variação do tempo de execução para as diferentes populações mínimas adotadas.

Figura 41 - Tempo de execução para as diferentes populações mínimas do algoritmo DPSO



Na Figura 42 é apresentada a variação da entropia das imagens resultantes para as diferentes populações mínimas adotadas.

Figura 42- Entropia das imagens resultantes para as diferentes populações mínimas do algoritmo DPSO

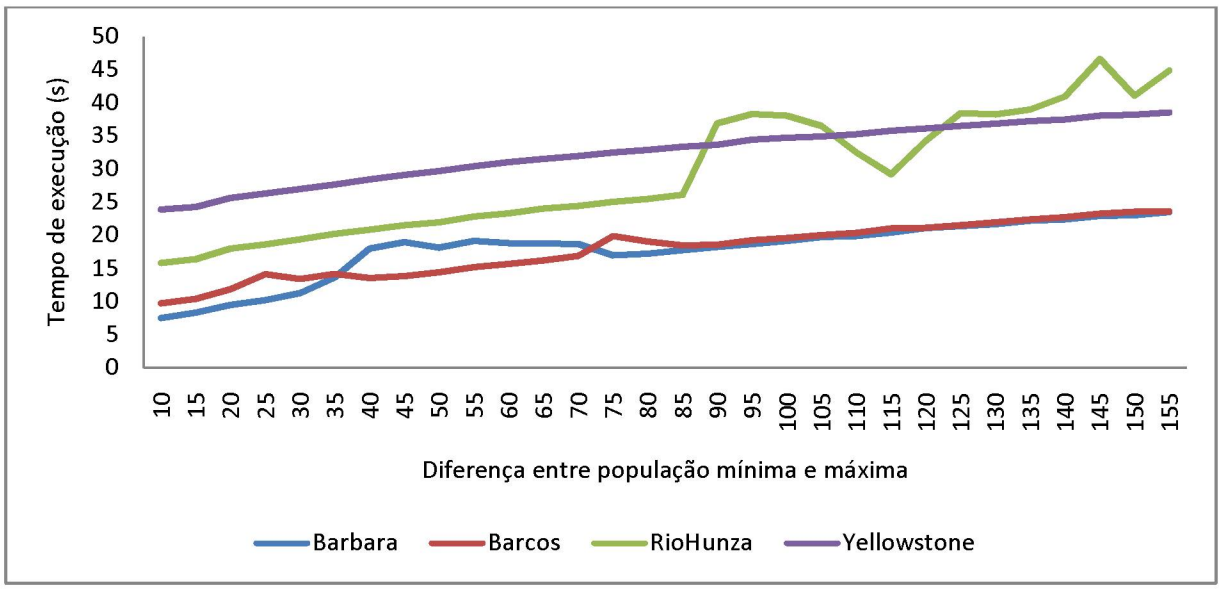


Pelos resultados das Figuras 41 e 42 pode-se concluir que o tempo de execução é diretamente afetado pela população mínima, mas a entropia sofre pouca variação,

principalmente para mais valores maiores que 6, sendo assim a população mínima escolhida é 6.

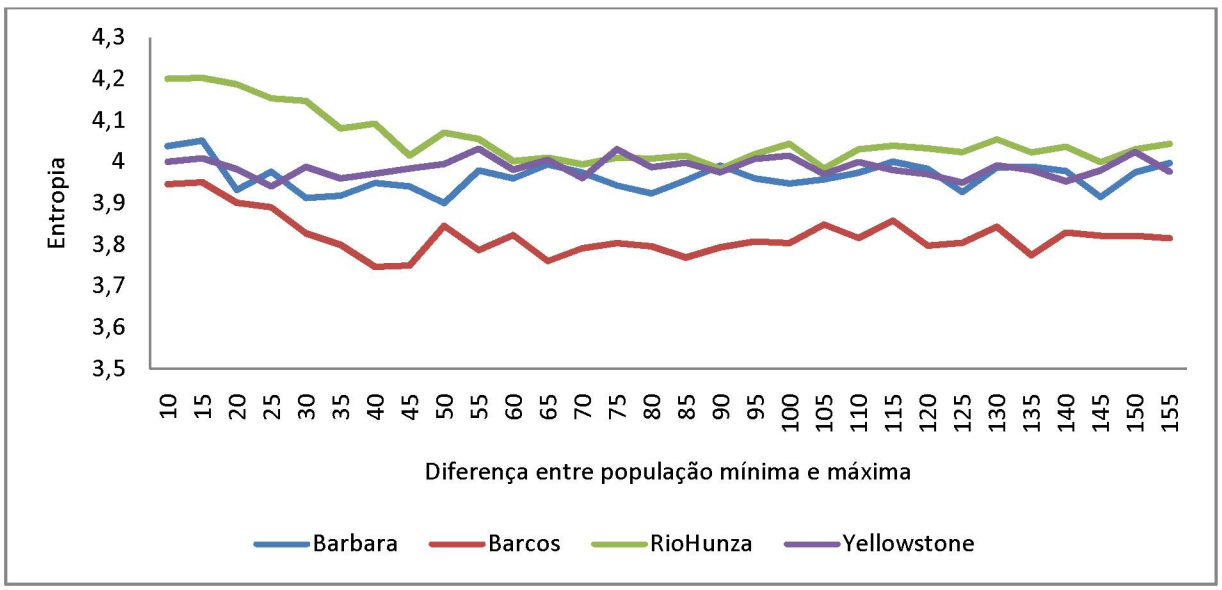
A Figura 43 apresenta a variação do tempo de execução para a variação entre as populações mínima e máxima adotadas.

Figura 43 - Tempo de execução para as diferenças entre populações mínimas e máximas do algoritmo DPSO



Na Figura 44 é apresentada a variação da entropia das imagens resultantes para a variação entre as populações mínima e máxima adotadas.

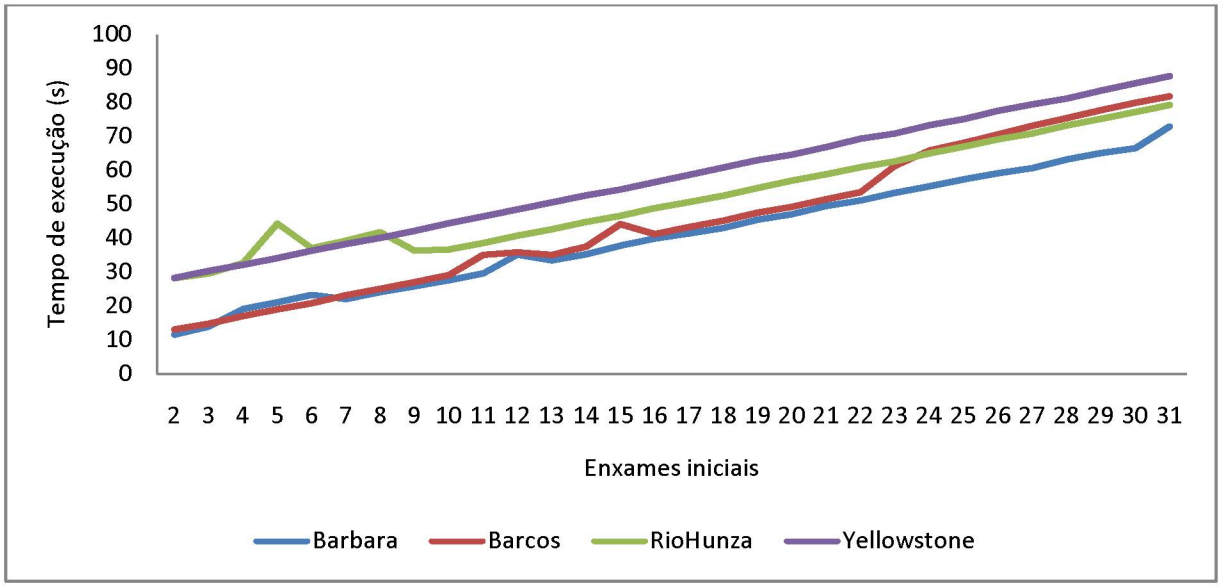
Figura 44- Entropia das imagens resultantes para as diferenças entre populações mínimas e máximas do algoritmo DPSO



A partir dos resultados obtidos nas Figuras 43 e 44 pode-se concluir que o tempo de execução é diretamente afetado pela variação entre as populações mínima e máxima, mas a entropia sofre pouca variação, sofrendo uma piora no resultado para valores maiores que 35, sendo assim a variação entre as populações mínima e máxima escolhida é 15, pois possui o melhor resultado de entropia para todas as imagens teste.

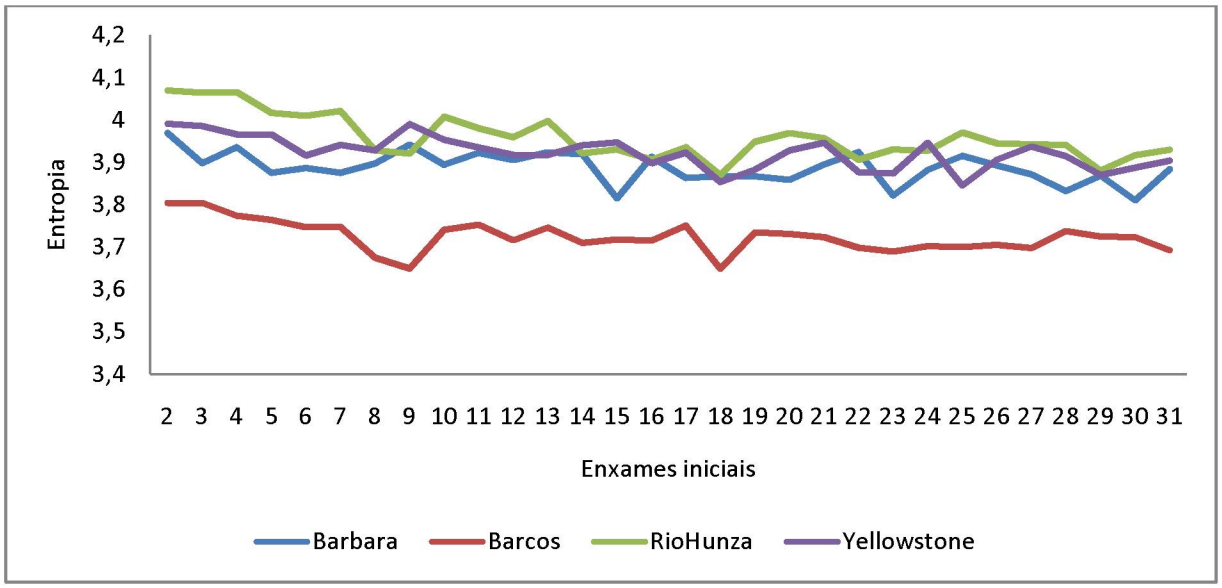
A Figura 45 apresenta a variação do tempo de execução para as diferentes quantidades de enxames iniciais adotadas.

Figura 45 - Tempo de execução para as diferentes enxames iniciais do algoritmo DPSO



Na Figura 46 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de enxames iniciais adotadas.

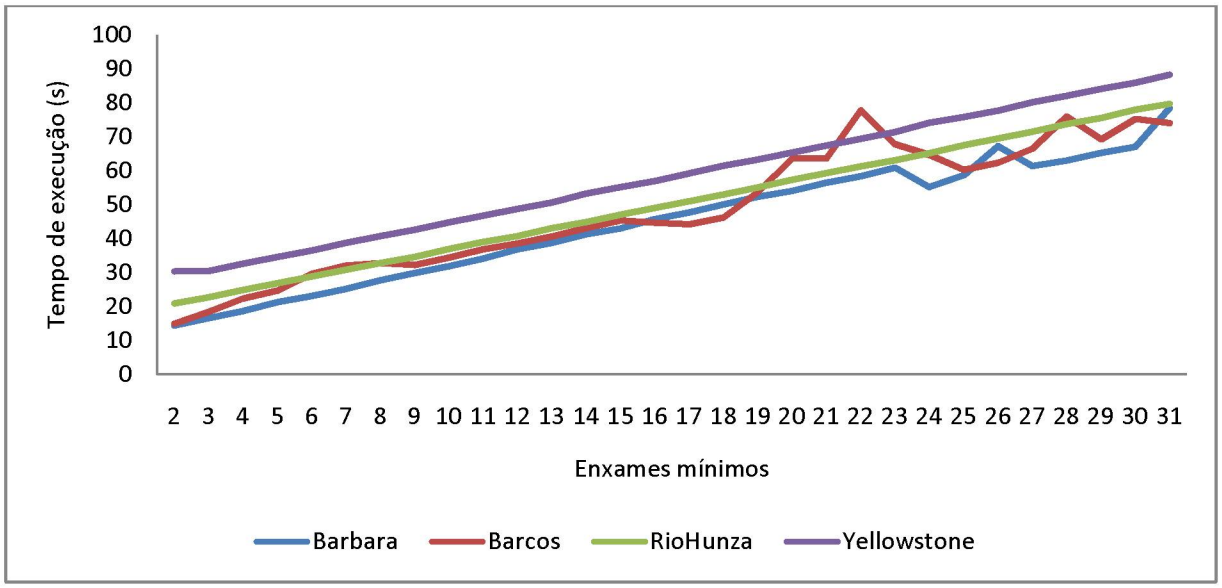
Figura 46- Entropia das imagens resultantes para as diferentes enxames iniciais do algoritmo DPSO



A partir dos resultados obtidos nas Figuras 45 e 46 pode-se concluir que o tempo de execução é diretamente afetado pela variação das diferentes quantidades de enxames iniciais, mas a entropia sofre pouca variação, sendo assim a quantidade de enxames iniciais escolhida é 2, pois possui o melhor resultado de entropia para todas as imagens teste.

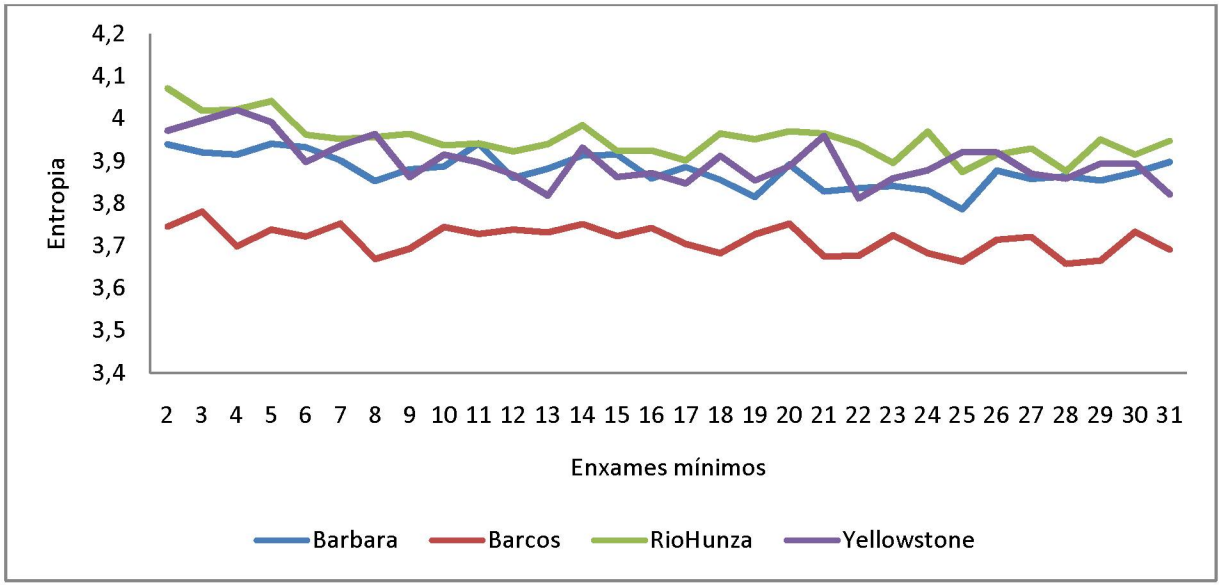
A Figura 47 apresenta a variação do tempo de execução para as diferentes quantidades de enxames mínimos adotadas.

Figura 47 - Tempo de execução para as diferentes enxames mínimos do algoritmo DPSO



Na Figura 48 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de enxames mínimos adotadas.

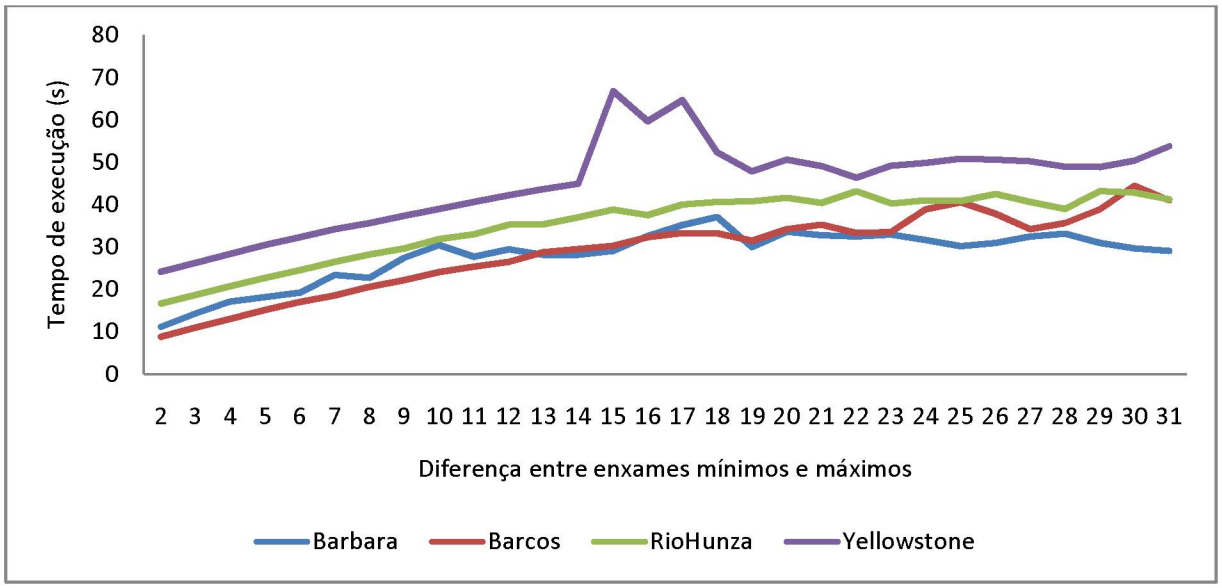
Figura 48- Entropia das imagens resultantes para as diferentes enxames mínimos do algoritmo DPSO



Pode-se concluir que o tempo de execução é diretamente afetado pela variação das diferentes quantidades de enxames mínimos, mas a entropia sofre pouca variação, principalmente para valores maiores que 5 sendo assim a quantidade de enxames mínimos escolhida é 3, pois possui um dos melhores resultados de entropia e tempo de execução para todas as imagens teste, a partir dos resultados obtidos nas Figuras 47 e 48.

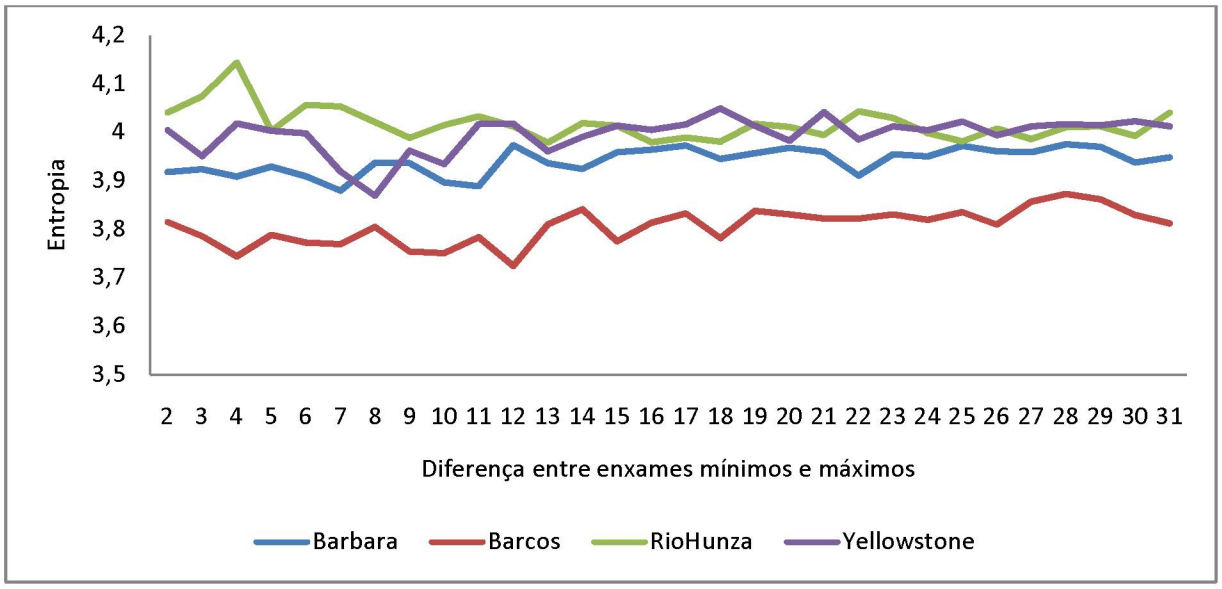
A Figura 49 apresenta a variação do tempo de execução para a variação entre a quantidade de enxames mínima e máxima adotadas.

Figura 49 - Tempo de execução para as diferenças entre enxames mínimos e máximos do algoritmo DPSO



Na Figura 50 é apresentada a variação da entropia das imagens resultantes para a variação entre a quantidade de enxames mínima e máxima adotadas.

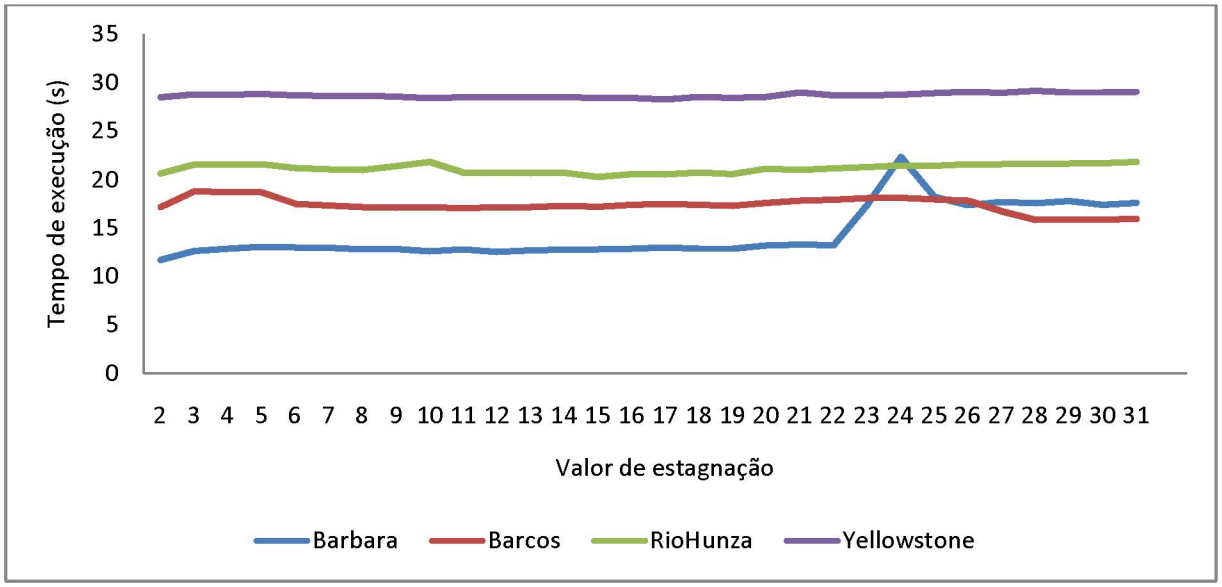
Figura 50- Entropia das imagens resultantes para as diferenças entre enxames mínimos e máximos do algoritmo DPSO



A partir dos resultados obtidos nas Figuras 49 e 50 pode-se concluir que o tempo de execução é diretamente afetado pela variação entre a quantidade de enxames mínima e máxima, mas a entropia sofre pouca variação, sendo assim a variação entre a quantidade de enxames mínima e máxima escolhida é 4, pois possui um dos melhores resultados de entropia e tempo de execução para todas as imagens teste.

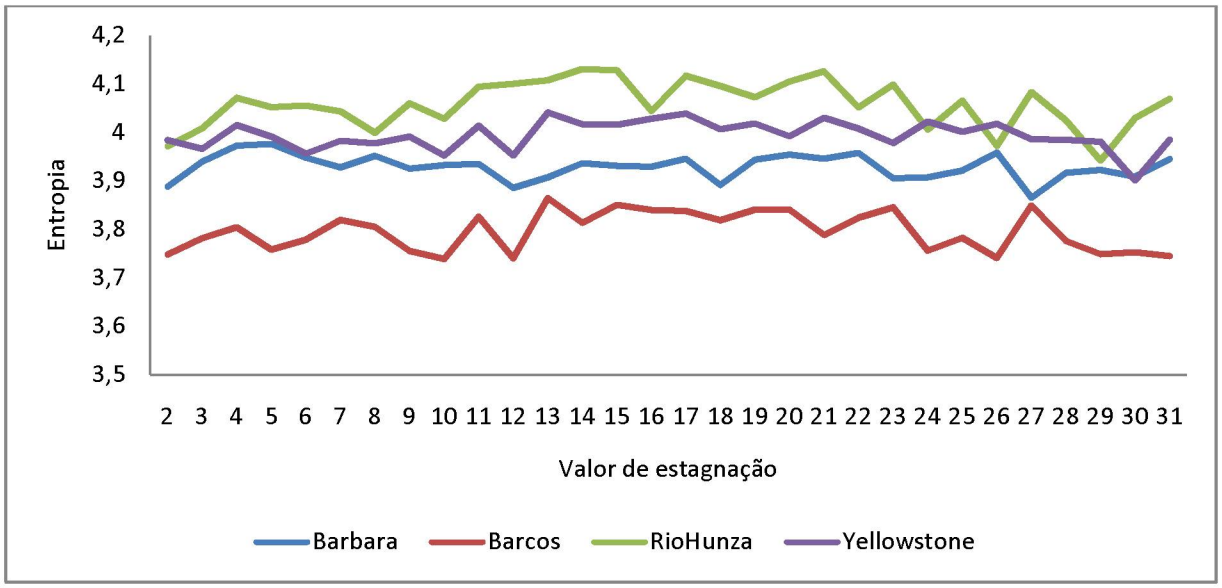
A Figura 51 apresenta a variação do tempo de execução para os diferentes valores de estagnação adotados.

Figura 51- Tempo de execução para as diferentes valores de estagnação do algoritmo DPSO



Na Figura 52 é apresentada a variação da entropia das imagens resultantes para os diferentes valores de estagnação adotados.

Figura 52- Entropia das imagens resultantes para as diferentes valores de estagnação do algoritmo DPSO

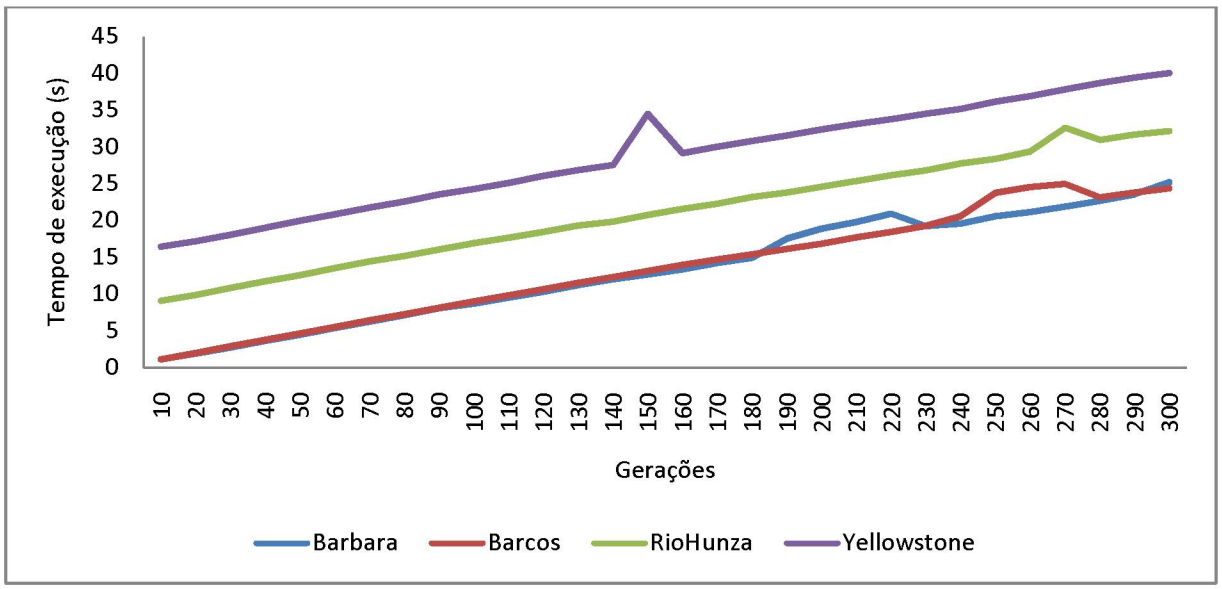


A partir dos resultados mostrados nas Figuras 51 e 52 pode-se concluir que tanto o tempo de execução quanto a entropia são pouco afetados pela variação do valor de

estagnação, sendo assim valor de estagnaçãoescolhido é 15, pois possui um dos melhores valores de entropia para todas as imagens teste.

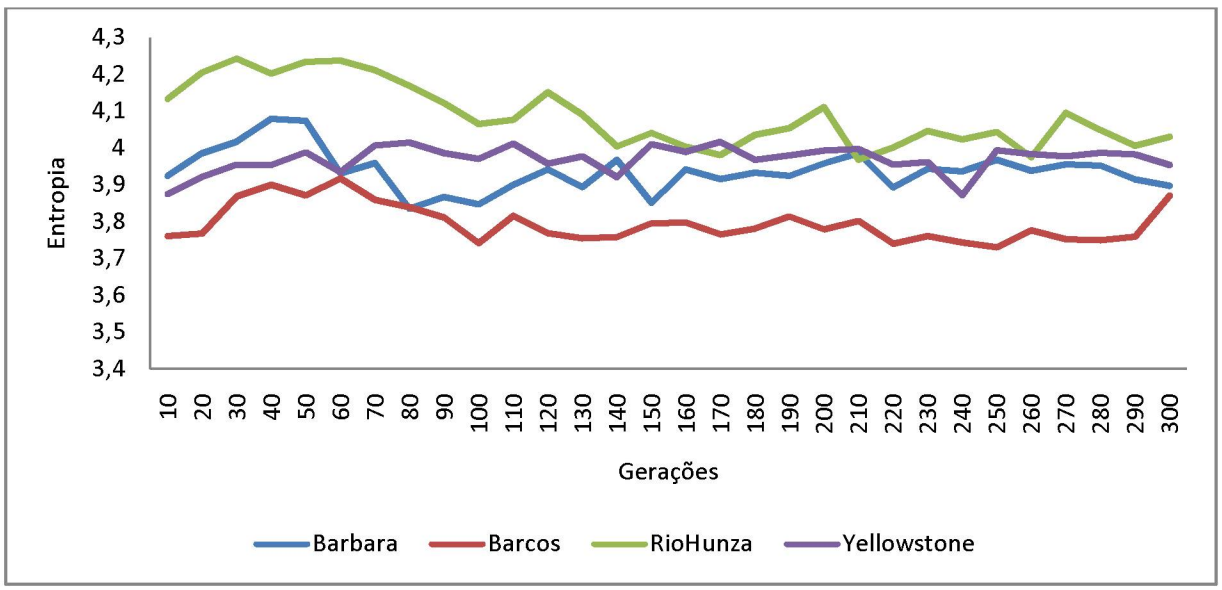
A Figura 53 apresenta a variação do tempo de execução para as diferentes quantidades de gerações adotadas.

Figura 53 – Tempo de execução para as diferentes quantidades de gerações do algoritmo DPSO



Na Figura 54 é apresentada a variação da entropia das imagens resultantes para as diferentes quantidades de gerações adotadas.

Figura 54- Entropia das imagens resultantes para as diferentes quantidades de gerações do algoritmo DPSO



A partir dos resultados obtidos nas Figuras 53 e 54 observa-se que o tempo de execução é diretamente afetado pela quantidade de gerações, e a entropia sofre uma piora para valores maiores que 100 ou menores que 20, sendo assim a quantidade de gerações escolhida é 50, pois possui um dos melhores resultados de entropia e tempo de execução para todas as imagens teste.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador por enxame de partículas darwiniano foram escolhidos:

- *População inicial*: 40;
- *População mínima*: 6;
- *Variação das populações*: 15;
- *Enxames iniciais*: 2;
- *Enxames mínimos*: 5;
- *Variação da quantidade de enxames*: 4;
- *Tempo de estagnação*: 15;
- *Peso individual*: 0,8;
- *Peso social*: 0,8;
- *Fator de inércia*: 0,7;
- *Quantidade de gerações*: 50.

6.1.5 Otimizador por enxame de partículas Darwiniano de ordem fracionária (FO-DPSO)

Para o algoritmo otimizador por enxame de partículas darwiniano foram testadas variações em onze parâmetros, sendo eles: população inicial, população mínima, variação das populações, enxames iniciais, enxames mínimos, variação da quantidade de enxames, tempo de estagnação, peso individual, peso social, fator de inércia, quantidade de gerações e coeficiente fracionário. Adotando os seguintes valores:

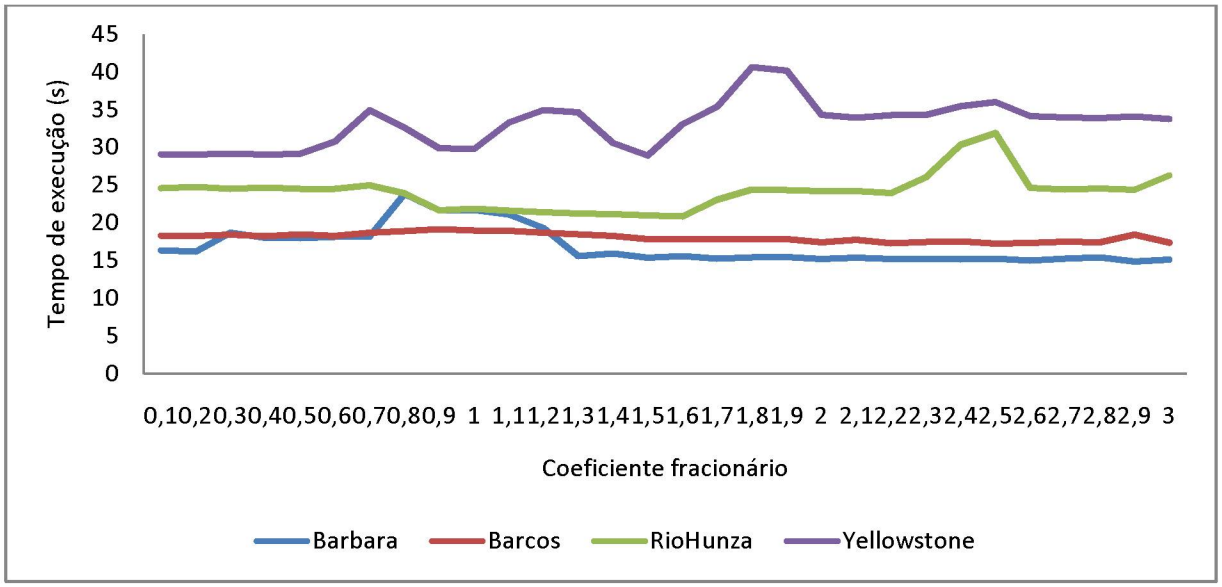
- *População inicial*: 20 a 310, com um passo de 10 entre cada valor;
- *População mínima*: 4 a 62, com um passo de 2 entre cada valor;
- *Variação das populações*: 10 a 155, com um passo de 5 entre cada valor;
- *Enxames iniciais*: 2 a 31, com um passo de 1 entre cada valor;

- *Enxames mínimos*: 2 a 31, com um passo de 1 entre cada valor;
- *Variação da quantidade de enxames*: 2 a 31, com um passo de 1 entre cada valor;
- *Tempo de estagnação*: 2 a 31, com um passo de 1 entre cada valor;
- *Peso individual*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Peso social*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Fator de inércia*: 0,1 a 3, com um passo de 0,1 entre cada valor;
- *Quantidade de gerações*: 10 a 300, com um passo de 10 entre cada valor;
- *Coeficiente fracionário*: 0,1 a 3, com um passo de 0,1 entre cada valor.

Devido ao algoritmo ser derivado do otimizador por enxame de partículas darwiniano, nota-se nos testes executados que a variação dos parâmetros de controle em comum (população inicial, população mínima, variação das populações, enxames iniciais, enxames mínimos, variação da quantidade de enxames, tempo de estagnação, peso individual, peso social, fator de inércia e quantidade de gerações) possui o mesmo comportamento em ambos os algoritmos.

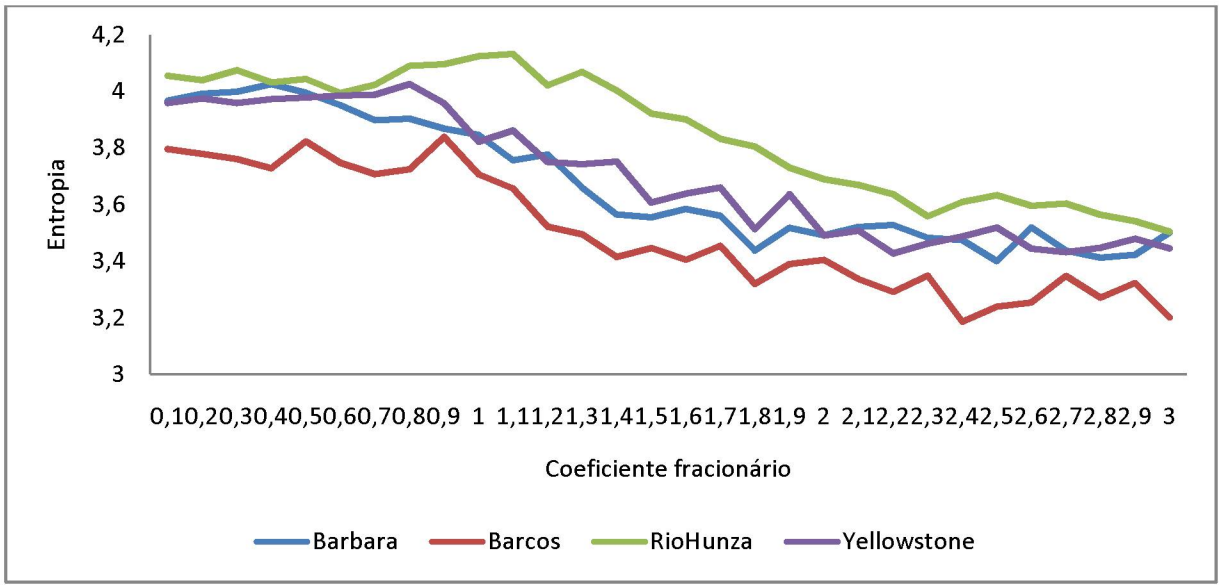
A Figura 55 apresenta a variação do tempo de execução para os diferentes coeficientes fracionários adotados.

Figura 55- Tempo de execução para os diferentes coeficientes fracionários do algoritmo FO-DPSO



Na Figura 56 é apresentada a variação da entropia das imagens resultantes para os diferentes coeficientes fracionários adotados.

Figura 56- Entropia das imagens resultantes para os diferentes coeficientes fracionários do algoritmo FO-DPSO



A partir dos resultados das Figuras 55 e 56 pode-se concluir que a entropia resultante é diretamente afetada pelo aumento do coeficiente fracionário, enquanto o tempo de execução é pouco afetado, sendo assim o coeficiente fracionário escolhido é 0.5, pois possui um dos melhores resultados de entropia e tempo de execução para todas as imagens teste.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador por enxame de partículas darwiniano de ordem fracionária foram escolhidos:

- *População inicial*: 40;
- *População mínima*: 6;
- *Variação das populações*: 15;
- *Enxames iniciais*: 2;
- *Enxames mínimos*: 5;
- *Variação da quantidade de enxames*: 4;
- *Tempo de estagnação*: 15;
- *Peso individual*: 0,8;
- *Peso social*: 0,8;
- *Fator de inércia*: 0,7;
- *Quantidade de gerações*: 50;

- *Coeficiente fracionário*: 0,5.

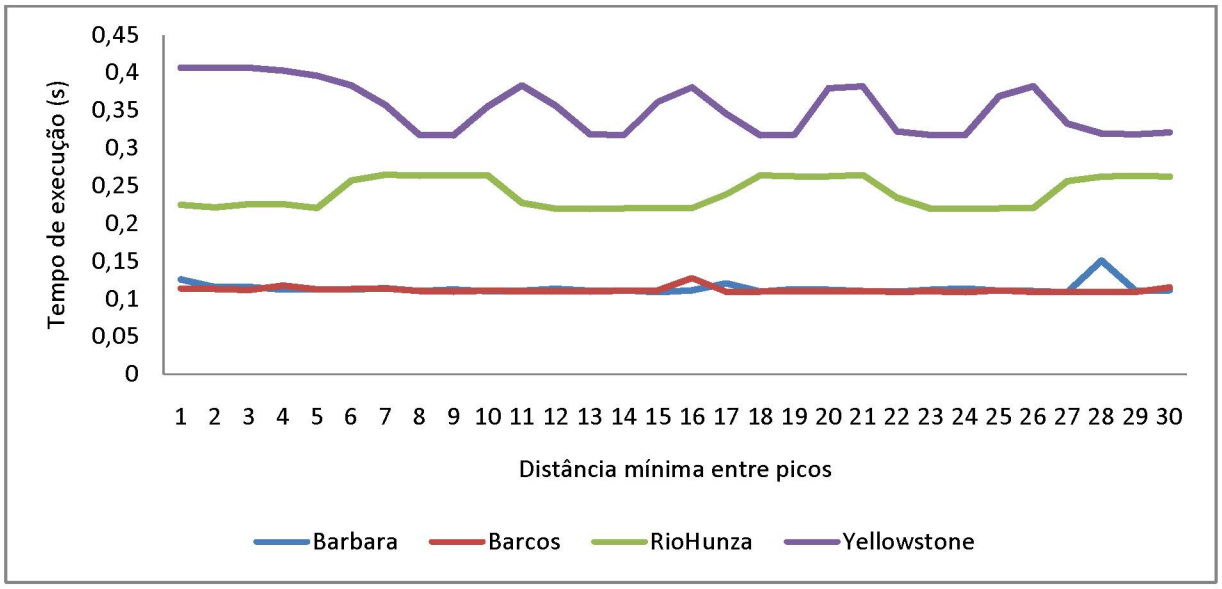
6.1.6 Otimizador por força-bruta

Para o algoritmo otimizador por força-bruta foram testadas variações em dois parâmetros, sendo eles: população inicial, população mínima, variação das populações, enxames iniciais, enxames mínimos, variação da quantidade de enxames, tempo de estagnação, peso individual, peso social, fator de inércia, quantidade de gerações e coeficiente fracionário. Adotando os seguintes valores:

- *Distância mínimos entre picos*: 1 a 30, com um passo de 1 entre cada valor;
- *Janela de suavização*: 1 a 30, com um passo de 1 entre cada valor.

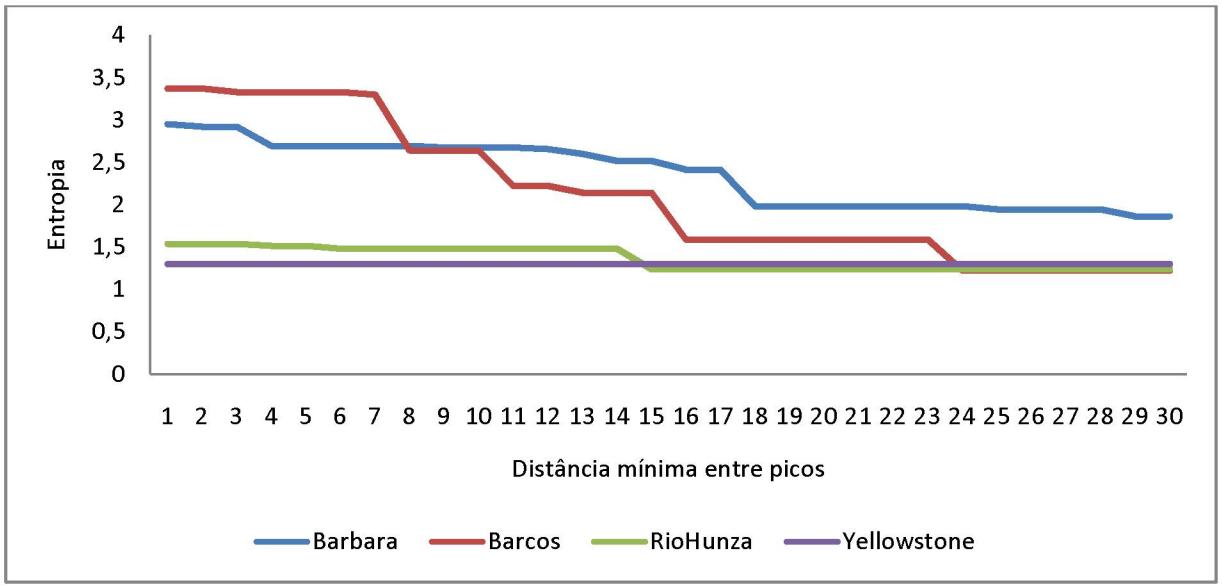
A Figura 57 apresenta a variação do tempo de execução para as diferentes distâncias mínimas entre picos adotadas.

Figura 57- Tempo de execução para os diferentes distâncias mínimas entre picos para o algorítmo de força-bruta



Na Figura 58 é apresentada a variação da entropia das imagens resultantes para as diferentes distâncias mínimas entre picos adotadas.

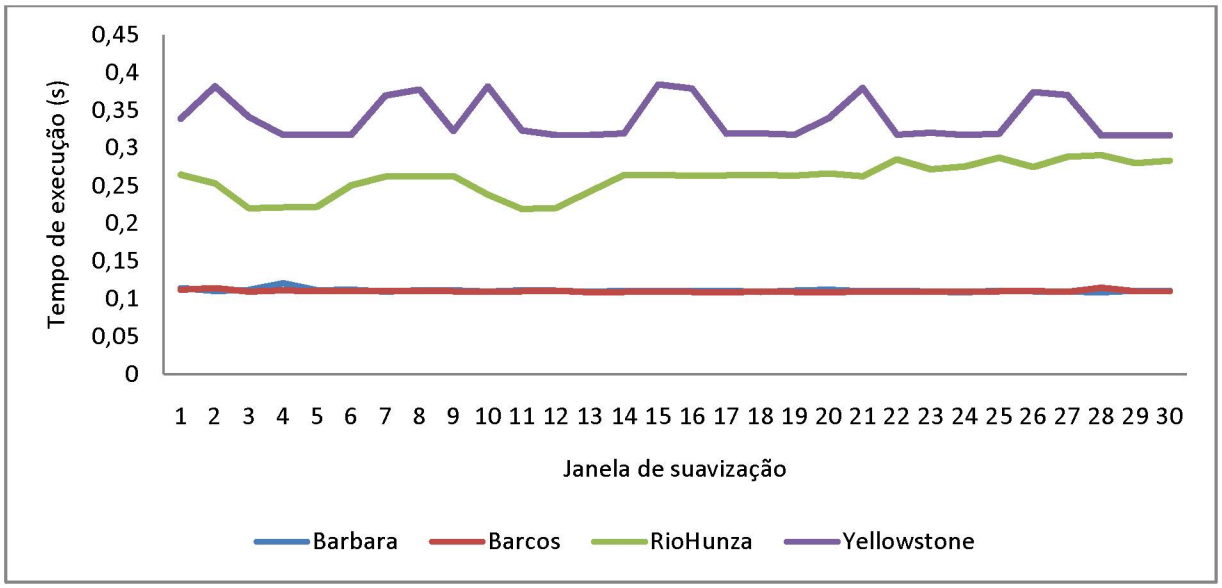
Figura 58- Entropia das imagens resultantes para os diferentes distâncias mínimas entre picos para o algorítmo de força-bruta



A partir dos resultados apresentados nas Figuras 57 e 58 pode-se concluir que a entropia resultante é diretamente afetada pelo aumento da distância mínima entre os picos, enquanto o tempo de execução é pouco afetado, sendo assim a distância mínima entre picos escolhidoa é 3, pois possui o melhor resultado de entropia e um dos melhores tempos de execução tempo de execução para quase todas as imagens teste.

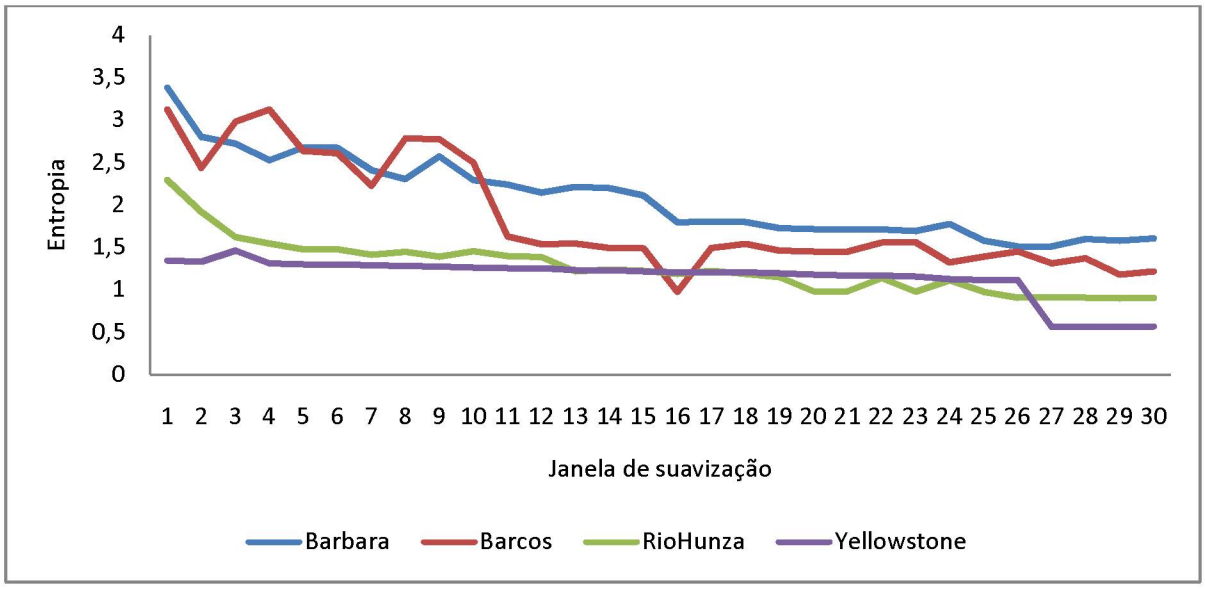
A Figura 59 apresenta a variação do tempo de execução para as diferentes janelas de suavização adotadas.

Figura 59- Tempo de execução para os diferentes janelas de suavização para o algorítmo de força-bruta



Na Figura 60 é apresentada a variação da entropia das imagens resultantes para as diferentes janelas de suavização adotadas.

Figura 60- Entropia das imagens resultantes para os diferentes janelas de suavização para o algorítmo de força-bruta



Pode-se concluir que a entropia resultante é diretamente afetada pelo aumento da distância janela de suavização, enquanto o tempo de execução é pouco afetado, sendo assim a janela de suavização escolhidoa é 1, pois possui o melhor resultado de entropia e um dos melhores tempos de execução tempo de execução para todas as imagens teste, a partir dos resultados obtidos nas Figuras 59 e 60.

Sendo assim, os seguintes valores para os parâmetros de controle do algoritmo otimizador por força-bruta foram adotados:

- *Distância mínimos entre picos*: 3;
- *Janela de suavização*: 1.

7. COMPARAÇÃO ENTRE META-HEURÍSTICAS DE OTIMIZAÇÃO E ALGORITMO FORÇA-BRUTA

Após a determinação dos melhores parâmetros de controle para cada algoritmo utilizado, os algoritmos foram executados novamente, apenas com os parâmetros definidos no capítulo 6. Cada algoritmo de otimização foi executado trinta vezes, com os mesmos parâmetros de controle e a mesma quantidade de avaliações da função objetivo, para cada uma das quatro imagens. Sendo assim, nas Tabelas de 2 a 9 são apresentadas as entropias resultantes e os tempos de execuções computacionais em segundos.

Tabela 2 - Tempos de execução (em segundos) para a imagem Barbara

Barbara	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	1,13	8,12	9,09	0,45	0,15	1,99
Mínimo	0,72	2,51	2,65	0,19	0,04	0,12
Média	0,79	6,43	7,08	0,21	**0,07**	0,20
Desvio padrão	0,08	1,37	1,62	0,05	0,04	0,34

Tabela 3 - Tempos de execução (em segundos) para a imagem Barcos

Barcos	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	1,13	8,47	14,45	0,24	0,08	0,54
Mínimo	0,76	6,02	6,84	0,19	0,04	0,12
Média	0,80	7,32	9,39	0,20	**0,05**	0,17
Desvio padrão	0,07	0,57	1,65	0,01	0,01	0,08

Tabela 4 - Tempos de execução (em segundos) para a imagem Rio Hunza

Rio Hunza	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	13,30	21,89	24,63	0,51	0,08	0,75
Mínimo	9,27	16,69	18,28	0,19	0,04	0,24
Média	11,31	18,40	20,48	0,25	**0,04**	0,28
Desvio padrão	0,92	1,05	1,35	0,06	0,01	0,09

Tabela 5 - Tempos de execução (em segundos) para a imagem Yellowstone

Yellowstone	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	27,37	32,24	33,11	0,28	0,07	0,85
Mínimo	20,00	25,85	28,91	0,23	0,04	0,34
Média	22,47	27,35	30,15	0,24	**0,04**	0,46
Desvio padrão	1,87	1,23	0,97	0,01	0,01	0,11

Nas Tabelas 2 a 5 são apresentados os tempos de execução dos algoritmos testados para as imagens teste. Percebe-se que o algoritmo GWO apresenta os menores tempos médio e máximo de execução e também o menor desvio padrão entre as execuções, permanecendo com tempo de execução semelhante para as quatro imagens teste, sendo assim considerado o algoritmo com o melhor desempenho computacional entre os utilizados.

O algoritmo PSO e seus derivado (DPSO e FODPSO) apresentam a maior variação de tempo de execução entre as imagens utilizadas, apresentando tempos maiores para as imagens de satélite (Rio Hunza e Yellowstone), sendo que o algoritmo PSO em si é o que apresenta a maior variação, levando em média até 28 vezes mais tempo para processar a imagem Yellowstone do que para processar as imagens Barbara e Barcos.

Percebe-se também que o algoritmo FODPSO apresenta os maiores tempos médio e mínimo de execução e também o maior desvio padrão entre as execuções para todas as imagens teste, sendo considerado assim o algoritmo com o pior desempenho computacional entre os utilizados, em termos de tempo de execução necessário. O algoritmo de força bruta apresenta resultados semelhantes aos do algoritmo ALO, com exceção da imagem Yellowstone, onde necessita do dobro do tempo computacional do ALO para realizar o processamento da imagem.

Tabela 6 - Entropias resultantes para a imagem Barbara

Barbara	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	4,04	4,18	4,22	4,13	4,15	4,36
Mínimo	3,17	3,33	3,45	3,56	3,73	4,36
Média	3,59	3,96	3,94	3,89	3,98	**4,36**
Desvio padrão	0,20	0,21	0,21	0,15	0,11	0,00

Tabela 7 - Entropias resultantes para a imagem Barcos

Barcos	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	4,04	4,00	4,06	4,01	3,96	4,54
Mínimo	3,30	3,30	3,03	3,01	3,15	4,54
Média	3,81	3,88	3,85	3,48	3,59	**4,54**
Desvio padrão	0,17	0,15	0,23	0,28	0,23	0,00

Tabela 8 - Entropias resultantes para a imagem Rio Hunza

Rio Hunza	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	4,27	4,29	4,30	4,28	4,22	2,51
Mínimo	3,67	3,71	3,74	3,66	3,80	2,51
Média	3,96	**4,19**	4,18	4,08	4,05	2,51
Desvio padrão	0,17	0,14	0,13	0,11	0,11	0,00

Tabela 9 - Entropias resultantes para a imagem Yellowstone

Yellowstone	PSO	DPSO	FODPSO	ALO	GWO	Força Bruta
Máximo	4,14	4,08	4,09	4,13	4,01	1,56
Mínimo	3,31	3,43	3,37	3,14	3,36	1,56
Média	3,71	**3,98**	3,95	3,64	3,70	1,56
Desvio padrão	0,25	0,12	0,16	0,24	0,18	0,00

Nas Tabelas 6 a 9 são apresentadas as entropias das imagens resultantes dos algoritmos testados para as imagens teste. Percebe-se que o algoritmo de força bruta apresenta os melhores resultados de entropia resultante para as imagens teste Barbara e Barcos, mas apresenta os piores resutados para as imagens aeroespaciais, pelo fato de as imagens aeroespaciais possuírem maior resolução, resultando em histogramas mais complexos, com uma quantidade maior de máximos e mínimos locais. Percebe-se também que todos os algoritmos baseados em inteligência de enxame utilizados apresentam resultados semelhantes para as quatroimagens teste utilizadas.

8. CONCLUSÃO E FUTURA PESQUISA

Foi proposta nesta dissertação o estudo do desempenho de cinco meta-heurísticas de otimização aplicadas à multilimiarização de imagens, em termos de desempenho computacional e entropia das imagens resultantes. Para limitar o espectro de abordagem da dissertação, foram escolhidas apenas quatro imagens teste, sendo duas delas já utilizadas como *benchmark* em estudos anteriores, enquanto as outras duas imagens capturas aeroespaciais. As meta-heurísticas implementadas foram comparadas com o método de força bruta, baseando a comparação no desempenho dos algoritmos em termos de tempo computacional e entropia da imagem resultante.

Os objetivos desta dissertação, tanto geral quanto específicos foram atendidos. As meta-heurísticas de otimização baseadas em inteligência de enxame foram implementadas e validadas com sucesso para a aplicação de multilimiarização de imagem. Realizou-se também um estudo sobre os parâmetros de controle das meta-heurísticas de otimização utilizadas, conforme apresentado no capítulo 6, sendo possível analizar o efeito de cada parâmetro de controle para a aplicação de multilimiarização de enxame.

Com base nos resultados apresentados no capítulo 7 pode-se dizer que existe uma clara tendência em ser recomendada a utilização do GWO, por apresentar entropias das imagens resultantes próximas dos melhores resultados obtidos, tanto com as demais meta-heurísticas quanto com o algoritmo de força bruta, combinado com o melhor desempenho computacional em todas as imagens teste utilizadas. Apesar de o algorítmo GWO não apresentar o melhor resultado de entropia resultante média para nenhuma das imagens teste utilizadas, o mesmo apresenta sempre o menor tempo computacional médio, sendo entre 0,04 e 0,07 segundos para cada imagem processada, o que o torna o algoritmo mais indicado principalmente para aplicações que necessitam da segmentação de um grande volume de imagens, como o processamento de vídeo em tempo real, por exemplo.

Para uma pesquisa posterior, podem ser consideradas meta-heurísticas de otimização diferentes para a aplicação em multilimiarização de imagens, tais como o otimizador baseado na Colônia de Formigas (do inglês *Ant Colony Optimization,*

ACO(DORIGO; MANIEZZO; COLORNI, 1996)) e o algoritmo baseado nos Vagalumes (do inglês *Firefly Algorithm, FA)* propostpo em YANG (2010).

Diferentes métodos de validação também podem ser aplicados, como a utilização das imagens resultantes como entrada para algoritmos de classificação de imagens (GHONGE; MITTAL, 2017) e reconhecimento de padrões(GONZÁLEZ; BIANCONI; FERNÁNDEZ, 2016).

Para futura pesquisa, sugere-se a adoção de abordagens multiobjetivo de otimização ou mesmo meta-heurísticas de otimização adaptativas para problemas de multilimiarização de imagens. Outra abordagem seria a pesquisa de parâmetros de inicialização das meta-heurísticas de otimização diferentes para cada classe de imagens.

REFERÊNCIAS

AKAY, B. A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Applied Soft Computing Journal*, v. 13, n. 6, p. 3066–3091, 2013.

ALRASHIDI, M.R.; EL-HAWARY, M.E. A Survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 4, p. 913–918, 2009.

ANGELOVA, M.; ATANASSOV, K.; PENCHEVA, T. Purposeful model parameters genesis in simple genetic algorithms. *Computers & Mathematics with Applications*, v. 64, n. 3, p. 221–228, 2012.

AZIZ, M. A. E.; EWEES, A. A.; HASSANIEN, A. E. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications*, v. 83, p. 242–256, 2017.

BÄCK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

BHALERAO, A.; WILSON, R. Unsupervised image segmentation combining region and boundary estimation. *Image and Vision Computing*, v. 19, n. 6, p. 353–368, 2001.

BHANDARI, A. K.*et al.* Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. *Expert Systems with Applications*, v. 41, n. 7, p. 3538–3560, 2014.

BIANCHI, L.*et al.* A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, v. 8, n. 2, p. 239–287, 2009.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

BOTZ, J. T.*et al.* Effects of slope and particle size on ant locomotion : implications for choice of substrate by antlions. *Entomological Society*, v. 76, n. 3, p. 426–435, 2003.

BOVIK, A. C. *The Essential guide to image processing*. Academic Press, 2009.

CâMARA, G.*et al.*Spring: Integrating remote sensing and gis by object-oriented data modelling. *Computers and Graphics*, v. 20, n. 3, p. 395–403, 1996.

CHEN, J.*et al.* A novel image segmentation method based on fast density clustering algorithm. *Engineering Applications of Artificial Intelligence*, v. 73, p. 92–110, 2018.

COUCEIRO, M; GHAMISI, P. *Fractional order Darwinian particle swarm optimization: applications and evaluation of an evolutionary algorithm*. Springer International Publishing, 2015.

COUCEIRO, M. S. *et al.*Introducing the fractional order robotic Darwinian PSO. Signal, Image, and Video Processing, v. 6, n. 3, p. 343–350.

COUCEIRO, M. S.; FONSECA FERREIRA, N. M.; TENREIRO MACHADO, J. A. Application of fractional algorithms in the control of a robotic bird. *Communications in Nonlinear Science and Numerical Simulation*, v. 15, n. 4, p. 895–910, 2010.

DAMBACH, Martin; GOEHLEN, Britta. Aggregation density and longevity correlate with humidity in first- instar nymphs of the cockroach (Blattella germanica L., Dictyoptera). *Journal of Insect Physiology*, v. 45, n. 5, p. 423–429, 1999.

DAS, S.*et al.*Real-parameter evolutionary multimodal optimization-A survey of the state-of-the-art.*Swarm and Evolutionary Computation*, v. 1, n. 2, p. 71–88, 2011

DEL VALLE, Y. *et al.* Particle Swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, v. 12, n. 2, p. 171–195, 2008.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, v. 26, n. 1, p. 29–41, 1996.

EMARY, E.; ZAWBAA, H. M.; HASSANIEN, A. E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, v. 172, p. 371–381, 2016. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2015.06.083>.

ERTENLICE, Okkes; KALAYCI, Can B. A survey of swarm intelligence for portfolio optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, v. 39, p. 36–52, 2018.

GARNIER, S.; GAUTRAIS, J.; THERAULAZ, G. The biological principles of swarm intelligence. *Swarm Intelligence*, v. 1, n. 1, p. 3–31, 2007.

GHAMISI, P.*et al.* An efficient method for segmentation of images based on fractional calculus and natural selection. *Expert Systems with Applications*, v. 39, n. 16, p. 12407–12417, 2012.

GHONGE, N. P.; MITTAL, D. Evaluation of imaging patterns of extra-cranial cysticercosis in North India. *Radiology of Infectious Diseases*, v. 4, n. 2, p. 49–57, 1 jun. 2017.

GONZÁLEZ, E.; BIANCONI, F.; FERNÁNDEZ, A. An investigation on the use of local multi-resolution patterns for image classification. *Information Sciences*, v. 361–362, p. 1–13,

GONZALEZ, R C; WOODS, R E. *Processamento digital de imagens*. Addison Wesley Brasil, 2000.

HAN, J.*et al.* A new multi-threshold image segmentation approach using state transition algorithm. *Applied Mathematical Modelling*, v. 44, p. 588–601, 2017.

HARNRNOUCHE, K.; DIAF, M.; SIARRY, P. A comparative study of various meta-

heuristic techniques applied to the multilevel thresholding problem. *Engineering Applications of Artificial Intelligence*, v. 23, n. 5, p. 676–688, 2010.

HINOJOSA, S.*et al.* Unassisted thresholding based on multi-objective evolutionary algorithms. *Knowledge-Based Systems*, v. 159, n. June, p. 221–232, 2018.

IONESCU, C. M.; SABATIER, J.; MACHADO, J. A Tenreiro. Special issue: Fractional signals and systems. *Signal, Image and Video Processing*, v. 6, n. 3, p. 341–342, 2012.

JIANG, M.; LUO, Y. P.; YANG, S. Y. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, v. 102, n. 1, p. 8–16, 2007.

JIANG, Y.*et al.* A cooperative honey bee mating algorithm and its application in multi-threshold image segmentation. *Information Sciences*, v. 369, p. 171–183, 2016.

KAPUR, J N; SAHOO, P K; WONG, A K C. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, v. 29, n. 3, p. 273–285, 1985.

KARCI, Ali. Fractional order entropy: New perspectives. *Optik*, v. 127, n. 20, p. 9172–9177, 2016.

KENNEDY, J; EBERHART, R. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, v. 4, p. 1942–1948, 1995.

KHAIRUZZAMAN, A. K. M.; CHAUDHURY, S. Multilevel thresholding using grey wolf optimizer for image segmentation. *Expert Systems with Applications*, v. 86, p. 64–76, 2017.

KHAN, S.; BAIG, A. R. Ant colony optimization based hierarchical multi-label classification algorithm. *Applied Soft Computing*, v. 55, p. 462–479, 2017.

KOLIAS, C.; KAMBOURAKIS, G.; MARAGOUDAKIS, M. Swarm intelligence in intrusion detection: A survey. *Computers & Security*, v. 30, n. 8, p. 625–642, 2011.

LEVIN, S. A. (Ed.). Subject Index. *Encyclopedia of Biodiversity*. New York: Elsevier, p. 1021–1103, 2001.

LO, E. H.S. *et al.* Image segmentation from scale and rotation invariant texture features from the double dyadic dual-tree complex wavelet transform. *Image and Vision Computing*, v. 29, n. 1, p. 15–28, 2011.

MARINI, F.; WALCZAK, B. Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems*, v. 149, p. 153–165, 2015.

MAVROVOUNIOTIS, M.; LI, C.; YANG, S. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, v. 33, n. December 2016, p. 1–17, 2017a.

MECH, L D. Alpha status, dominance, and division of labor in wolf packs. *Canadian Journal of Zoology*, v. 77, n. 8, p. 1196–1203, 1999.

MERZBAN, M. H.; ELBAYOUMI, M. Efficient solution of Otsu multilevel image thresholding: A comparative study. *Expert Systems with Applications*, v. 116, p. 299–309, 2019.

MIRJALILI, Seyedali. The ant lion optimizer. *Advances in Engineering Software*, v. 83, p. 80–98, 2015.

MIRJALILI, S. LEWIS, A. The whale optimization algorithm. *Advances in Engineering Software*, v. 95, p. 51–67, 2016.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. *Advances in Engineering Software*, v. 69, p. 46–61, 2014.

MURO, C. *et al.*Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural Processes*, v. 88, n. 3, p. 192–197, 2011.

NAIDU, M. S.R.; RAJESH KUMAR, P.; CHIRANJEEVI, K. Shannon and Fuzzy entropy based evolutionary image thresholding for image segmentation. *Alexandria Engineering Journal*, v. 57, n. 3, p. 1643–1655, 2016.

OTSU, N. A Threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 9, n. 1, p. 62–66, 1979.

PARE, S. *et al.* A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve. *Applied Soft Computing Journal*, v. 47, p. 76–102, 2016.

PARE, S. *et al.* An efficient method for multilevel color image thresholding using cuckoo search algorithm based on minimum cross entropy. *Applied Soft Computing Journal*, v. 61, p. 570–592, 2017.

PEDRINI, H; SCHWARTZ, W R. *Análise de imagens digitais: Princípios, algoritmos e aplicações*. Editora Thomson Learning, 2007.

POLAK, M.; ZHANG, H.; PI, M. An evaluation metric for image segmentation of multiple objects. *Image and Vision Computing*, v. 27, n. 8, p. 1223–1227, 2009.

PUN, T. A new method for grey-level picture thresholding using the entropy of the histogram. *Signal Processing*, v. 2, n. 3, p. 223–237, 1980.

RUSS, J. C. *Image processing handbook,* 4th. ed. Boca Raton, FL, USA: CRC Press, 2002.

SAHOO, A.; CHANDRA, S. Multi-objective grey wolf optimizer for improved cervix lesion classification. *Applied Soft Computing Journal*, v. 52, p. 64–80, 2017.

SALCEDO-SANZ, S. Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. *Physics Reports*, v. 655, p. 1–70, 2016.

SATHYA, P. D.; KAYALVIZHI, R. Optimal segmentation of brain MRI based on adaptive bacterial foraging algorithm. *Neurocomputing*, v. 74, n. 14–15, p. 2299–2313, 2011.

SEELEY, T. D. When is self-organization used in biological systems? *Biological Bulletin*, v.

202, n. 3, p. 314–318, 2002.

SERAPIÃO, A. B. de S. Fundamentos de otimização por inteligência de enxames: uma visão geral. *SBA: Controle & Automação*, v. 20, n. 3, p. 271–304, 2009.

SHA, C.; HOU, J.; CUI, H. A robust 2D Otsu's thresholding method in image segmentation. *Journal of Visual Communication and Image Representation*, v. 41, p. 339–351, 2016.

SHANNON, C.E. E. A Mathematical theory of communication. *Bell System Technical Journal*, v. 27, n. 3, p. 379–423, 1948.

SOLTEIRO PIRES, E. J. *et al.*Particle swarm optimization with fractional-order velocity. *Nonlinear Dynamics*, v. 61, n. 1–2, p. 295–301, 2010.

SONG, X.*et al.* Grey Wolf Optimizer for parameter estimation in surface waves. *Soil Dynamics and Earthquake Engineering*, v. 75, p. 147–157, 2015.

SREE RANJINI, S. R.; MURUGAN, S. Memory based hybrid dragonfly algorithm for numerical optimization problems. *Expert Systems with Applications*, v. 83, p. 63–78, 2017.

SREEJA, N. K.; SANKAR, A. Ant colony optimization based binary search for efficient point pattern matching in images. *European Journal of Operational Research*, v. 246, n. 1, p. 154–169, 2015.

TENREIRO MACHADO, J. A. *et al.*Some applications of fractional calculus in engineering. *Mathematical Problems in Engineering*. v. 2010, Article ID 639801, 34 pages, 2010.

TILLETT, J.; RAO, T M; *et al.* Darwinian particle swarm optimization. *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, Pune, India, p. 1474–1487.

VENTER, G.; SOBIESZCZANSKI-SOBIESKI, J. Particle Swarm Optimization. *AIAA Journal*, v. 41, n. 8, p. 1583–1589, 2003.

WANG, B.; CHEN, L.L.; CHENG, J. New result on maximum entropy threshold image segmentation based on P system. *Optik*, v. 163, p. 81–85, 2018.

WANG, W.; WU, C. Image segmentation by correlation adaptive weighted regression. *Neurocomputing*, v. 267, p. 426–435, 2017.

WANG, Y.-Y.*et al.* Fractional-order Darwinian PSO-based feature selection for media-adventitia border detection in intravascular ultrasound images. *Ultrasonics*, v. 92, p. 1–7,2018

YANG, B.*et al.* Grouped grey wolf optimizer for maximum power point tracking of doubly-fed induction generator based wind turbine. *Energy Conversion and Management*, v. 133, p. 427–443, 2017.

YANG, X.; KRISHNAN, S. M. Image segmentation using finite mixtures and spatial information. *Image and Vision Computing*, v. 22, n. 9, p. 735–745, 2004.

YANG, X. -S. Firefly algorithm, stochastic test functions and design optimisation,*International Journal of Bio-Inspired Computation*,v. 2, n. 2, p. 78–84, 2010.

YU, Z.; WONG, H. S.; WEN, G. A modified support vector machine and its application to image segmentation. *Image and Vision Computing*, v. 29, n. 1, p. 29–40, 2011.

ZHU, M.; LIANG, D.; YAN, P. A point pattern matching algorithm based on QR decomposition. *Optik*, v. 125, n. 14, p. 3485–3490, 2014.

ZIDAN, A.*et al.*Antlion optimization based segmentation for MRI liver images.Proceedings of the 10thInternational Conference on Genetic and Evolutionary Computing,Fuzhou City, Fujian Province, China,2016.