Universidade de Pernambuco (UPE)
Escola Politécnica de Pernambuco (POLI)
Instituto de Ciências Biológicas (ICB)

Programa de Pós-Graduação em Engenharia de Sistemas

Rodrigo Spencer Hartmann Calazans

Nova Técnica de Projeto de Quantizadores Vetoriais Baseada em Computação Inteligente

Dissertação de Mestrado

Recife, Junho de 2021.

Universidade de Pernambuco (UPE)
Escola Politécnica de Pernambuco (POLI)
Instituto de Ciências Biológicas (ICB)

Programa de Pós-Graduação em Engenharia de Sistemas

Rodrigo Spencer Hartmann Calazans

Nova Técnica de Projeto de Quantizadores Vetoriais Baseada em Computação Inteligente

Dissertação apresentada à Universidade de Pernambuco como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Sistemas.

Área de concentração: **Telemática**

Orientador: Prof. Dr. Francisco Madeiro Bernardino Junior

Recife, Junho de 2021.

Agradecimentos

Gostaria de agradecer aos familiares e amigos pela compreensão e incentivo que sempre estiveram presentes durante este período. Principalmente ao meus pais e irmãos que me fortalecem a todo instante.

À Camilla Medeiros com quem divido cada passo dado na vida e a quem admiro pelo grande caráter e força que possui.

Ao professor Francisco Madeiro por ter tornado esse caminho possível e engrandecedor. Além de um excelente profissional, o professor Madeiro carrega consigo uma grande humanidade.

À coordenação e à secretaria do programa pela disponibilidade e atenção dedicada a cada mestrando do programa.

Ao programa de pós-graduação em engenharia de sistemas (PPGES) que sempre atuou em prol de seus mestrandos com grande presteza.

Aos colegas do programa que sempre mantiveram o ambiente colaborativo e positivo. Em especial, gostaria de agradecer ao grupo de pesquisa que contribuiu bastante com debates e trocas de ideias.

À FACEPE (Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco) por ter financiado este projeto.

Resumo

A Quantização Vetorial (QV) é uma técnica que pode ser utilizada para compressão de imagens. O desempenho da QV, no que diz respeito à qualidade das imagens reconstruídas, depende da qualidade do dicionário utilizado. Nos últimos anos, a Inteligência de Enxames vem sendo empregada para o projeto de dicionários, que pode ser visto como um problema de otimização multidimensional. Algoritmos de enxame são técnicas bioinspiradas baseadas em população, em que os membros do enxame (partículas) buscam resolver um problema de forma descentralizada e eficiente. Este trabalho apresenta um novo algoritmo de projeto de dicionário, denominado Cuckoo-LBG, o qual é uma técnica que combina o algoritmo bioinspirado *Cuckoo Search* (CS) com o algoritmo de clusterização LBG (Linde-Buzo-Gray). A etapa de particionamento do algoritmo LBG envolve elevada complexidade computacional, a qual é um problema relevante em virtude do fato de que o algoritmo LBG, a cada iteração do Cuckoo-LBG, realiza o particionamento para cada partícula (dicionário) do enxame. Neste trabalho é avaliada uma alternativa para reduzir a complexidade supracitada. No trabalho são também avaliados os impactos de estratégias de inicialização para as partículas do enxame, em termos da qualidade dos dicionários projetados, avaliada por meio da qualidade das imagens reconstruídas. Precisamente, as estratégias de inicialização consistem em combinar dicionários obtidos por meio de algoritmos da literatura com dicionários constituídos por vetores aleatoriamente selecionados do conjunto de treino. Os resultados obtidos com o algoritmo Cuckoo-LBG são comparados com os obtidos por técnicas que se constituem em estado da arte. Os resultados de simulação apontam para os benefícios das estratégias de inicialização propostas.

Palavras-chave: Compressão de imagens, Quantização vetorial, *Cuckoo Search*, Computação bioinspirada, Projeto de dicionários

Abstract

Vector Quantization (VQ) is a technique used for image compression. The performance of VQ, regarding the quality of the reconstructed images, depends on the quality of the codebook used. In recent years, Swarm Intelligence has been used for the codebook design, which can be seen as a multidimensional optimization problem. Swarm algorithms, such as *Fish School Search* (FSS), are population-based bioinspired techniques, in which swarm members (particles) seek to solve a problem in a decentralized and efficient way. This work presents a new codebook design algorithm, called Cuckoo-LBG, which is a technique that combines the bioinspired *Cuckoo Search* (CS) algorithm with the LBG (Linde-Buzo-Gray) clustering algorithm. The partitioning step of the LBG algorithm involves high computational complexity, which is a relevant problem due to the fact that the LBG algorithm, at each iteration of the Cuckoo-LBG, performs the partitioning for each particle (codebook) of the swarm. In this work, an alternative to reduce the aforementioned complexity is evaluated. The work also evaluates the impacts of initialization strategies for the particles in the swarm, in terms of the quality of the designed codebooks, assessed by the quality of the reconstructed images. Precisely, the initialization strategies consist in combining codebooks obtained by initialization algorithms from the literature with codebooks constituted by vectors randomly selected from the training set. The results obtained with the Cuckoo-LBG algorithm are compared with those obtained by state-of-the-art techniques. The simulation results point out to the benefits of the proposed initialization strategies.

Keywords: Image compression,Vector quantization, *Cuckoo Search*, Bioinspired Computation, Codebook design

Lista de Abreviações e Siglas

ABC Colônia de Abelhas Artificiais (*Artificial Bee Colony)*

ACO Otimização por Colônia de Formigas (*Ant Colony Optimization)*

BT Busca Total

CS Busca do Cuco (*Cuckoo Search*)

DSICS Seleção dos Vetores-Código Iniciais Baseada em Dupla Ordenação) (*Double Sorting based Initial Codewords Selection*)

FA Algoritmo do Vagalume (*Firefly Algorithm)*

FSA Algoritmo da Busca Total (*Full Search Algorithm*)

FSS Busca por Cardume de Peixes (*Fish School Search*)

GLA *Generalized Lloyd Algorithm*

IEENNS Busca do Vizinho mais próximo de médias iguais variâncias iguais aprimorado (*Improved Equal-average Equal-variance Nearest Neighbor Search)*

LBG Linde-Buzo-Gray

MEIM Método de Inicialização de Entropia Máxima (*Maximum Entropy Initialisation Method*)

MSE Error Médio Quadrático (*Mean Squared Error*)

PDS Busca da Distorção Parcial (*Particial Distortion Search)*

PSNR Relação Sinal Ruído de Pico (*Peak Signal-to-Noise Ratio*)

PSO Otimização por Enxame de Partículas (*Particle Swarm Optimization*)

QV Quantização Vetorial

SMO Otimização por Macaco-aranha (*Spider Monkey Optimization*)

SSA Algoritmo da Busca por Esquilo *(Squirrel Search Algorithm)*

SSIM Medida do Índice de Similiridade Estrutural (*Structural Similarity Index Measure*)

UFPE Universidade Federal de Pernambuco

UPE Universidade de Pernambuco

VMP Vizinho mais Próximo

VQ *Vector Quantization*

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução e Motivação

Com o crescimento acelerado da produção e consumo de dados, os sistemas de telecomunicações recorrem a diferentes estratégias para armazenar e transmitir esta grande massa de dados. A compressão de sinais é uma alternativa para aumentar a eficiência na transmissão ou no armazenamento de informação. As técnicas de compressão de sinais se dividem predominantemente em dois grupos: a compressão sem perdas, utilizada por exemplo para o armazenamento de imagens médicas, e a compressão com perdas. Na compressão com perdas geralmente é possível obter maiores taxas de compressão [1].

A quantização vetorial (QV) é um método compressão de sinais com perdas [2]. A QV é amplamente utilizada em sistema de transmissão de voz [3] tendo aplicações também em saúde, na compressão de sinais médicos [4,5], marca d'água digital [6], reconhecimento de padrões [7,8], entre outros. No processo de codificação da QV, um conjunto limitado de vetores, denominado dicionário, é utilizado como referência para representar os vetores de entrada. Quando aplicada a uma imagem digital, a QV utiliza blocos não sobrepostos de *pixels* da imagem.

A qualidade do sistema de quantização vetorial está diretamente associada à qualidade do dicionário utilizado. Trabalho seminal na área de projeto de dicionário foi a introdução do algoritmo LBG (Linde-Buzo-Gray), também conhecido como GLA (*Generalized Lloyd Algorithm*) [9], por ser uma algoritmo de simples implementação e rápida convergência [10]. Diferentes técnicas de projeto de dicionário baseadas em agrupamentos têm sido propostas, avaliando, por exemplo, a qualidade dos sinais reconstruídos e número de iterações realizadas pelos algoritmos [11, 12]. No entanto o LBG apresenta limitações, são influenciados o desempenho e a velocidade de convergência pelo dicionário que inicializa o algoritmo e o tempo de execução é concentrado na etapa de particionamento.

Os algoritmos de enxame são técnicas metaheurísticas inspiradas na natureza, como no comportamento de seres coletivos, abelhas [13], formigas [14] e pássaros [15]. Algoritmos deste grupo também já foram aplicados ao projeto de dicionários de quantizadores vetoriais [16, 17].

Nos últimos anos uma nova família de algoritmos surgiu combinando a inteligência de enxame com técnicas de agrupamento como o LBG, [18–20]. A adaptabilidade das técnicas de enxame permite que eles sejam integrados a algoritmo LBG e sejam utilizadas para o projeto de quantizadores vetoriais. Recentemente, bons desempenhos dessas técnicas híbridas têm sido observados [21–23].

1.1 Objetivos

1.1.1 Objetivo Geral

Apresentar uma nova técnica de projeto de dicionário baseada em computação inteligente.

1.1.2 Objetivos Específicos

- Apresentar e avaliar o desempenho de uma nova técnica de computação inteligente aplicada ao projeto de dicionário para compressão de imagens baseadas em QV.
- Investigar técnicas de inicialização do novo algoritmo de computação inteligente aplicado ao projeto quantizadores vetoriais.
- Introduzir técnicas para redução do número de operações lógicas e/ou aritméticas do novo algoritmo de computação inteligente aplicado a projeto de dicionário.

1.2 Trabalhos Relacionados

A quantização vetorial é uma técnica de compressão de sinais. Particularmente, em se tratando de compressão de imagens, a qualidade das imagens reconstruídas está fortemente atrelada à qualidade do dicionário (quantizador vetorial) projetado. Em 1980 o algoritmo LBG foi proposto para o projeto de dicionário e a literatura mostra que diversos trabalho têm como base esta técnica [24–26].

O desempenho do algoritmo LBG, como outros baseados em QV, possui dependência em relação ao seu dicionário inicial, ou seja, do primeiros vetores-código escolhidos que darão início ao processo de partição [27].

A inicialização de técnicas de projeto de dicionário pode ser realizada de diversas maneiras, tais como: aleatória [28, 29], baseada em distância [30–33], baseada em ordenamento [34, 35], baseada em densidade [36, 37].

O algoritmo LBG pode levar a vetores-código com pouca "mobilidade", o que pode fazer com que o algoritmo fique preso em mínimos locais, comprometendo assim a exploração do espaço de busca [38]. Em grande medida a combinação de algoritmos de enxame ao LBG busca compensar essa limitação do algoritmo.

Nos últimos anos algoritmos bioinspirados combinados com o algoritmo LBG foram utilizados no contexto de projeto de dicionários, tais como: SSA-LBG (*Squirrel Search Algorithm* - LBG) [39] e CS-LBG (*Cuckoo Search* - LBG) [40].

O algoritmo CS-LBG aplica a técnica de enxame *Cuckoo Search* para o problema do projeto de dicionário no artigo [40] se utiliza o algoritmo LBG apenas para inicializar uma partícula de seu enxame e desse momento em diante segue com o algoritmo *Cuckoo Search*. Nesta dissertação, é apresentado um novo método de projeto de dicionário, denominado Cuckoo-LBG, que combina a otimização do algoritmo *Cuckoo Search* com o algoritmo LBG. A técnica ora proposta difere do algoritmo CS-LBG, em virtude do fato de que a primeira alterna ciclos de *Cuckoo Search* e de LBG, até que a condição de parada (baseada no limiar de distorção da melhor partícula) seja satisfeita. Além disso, na etapa LBG, todos os dicionários são atualizados, ou seja, o LBG é utilizado otimizando a busca local.

A complexidade computacional em algoritmos baseados em LBG se concentra predominantemente na etapa de do particionamento, etapa esta em que ocorre a busca pelo vizinho mais próximo (VMP). No algoritmo proposto nesta dissertação, assim como em outras técnicas de enxames que se combinam com o algoritmo LBG [21, 22], a complexidade computacional do

particionamento é elevada, sobretudo pelo fato de que nas técnicas supracitadas são projetados vários dicionários simultaneamente, ou seja, cada partícula do enxame corresponde a um dicionário. Neste trabalho uma das formas de aliviar a complexidade computacional é a utilização de técnicas eficientes de busca do VMP, cujo cenário tradicional de utilização é a etapa de codificação da quantização vetorial. Como exemplo de técnicas de busca do VMP pode-se citar *Partial Distance Search* (PDS) [41], *Improved Equal-average Equal-variance Nearest Neighbour Search* (IEENNS) [42].

1.3 Estrutura da Dissertação

O trabalho foi organizado em capítulos conforme descrito a seguir. O Capítulo 2 apresenta os conceitos e terminologias empregadas no cenário de quantização vetorial assim como um breve resumo das técnicas utilizadas para as inicializações e finaliza com um breve resumo dos métodos de busca do VMP. O Capítulo 3 apresenta uma base teórica sobre técnicas de enxame assim como o algoritmo *Cuckoo Search* e a técnica proposta nesta dissertação, denominada *Cuckoo-LBG*, para o projeto de quantizadores vetoriais. A metodologia é abordada no Capítulo 4. No Capítulo 5 os resultados são apresentados e discutidos e o Capítulo 6 contém as conclusões.

2
Quantização Vetorial

2.1 Fundamentos

Utilizada para a compressão de sinais com perdas, a quantização vetorial permite obter altas taxas de compressão [1] e é uma generalização da técnica de quantização escalar. Na QV sinais de entrada são representados por blocos de dados sem sobreposição entre eles [43].

O sistema baseado em quantização vetorial consiste em três principais etapas: a codificação, a transmissão/armazenamento e a decodificação, como ilustrado na Figura 2.1. A Figura 2.2 exibe um bloco de dimensão 4×4 *pixels* destacado da imagem e os valores em uma escala de cinza de 256 níveis, sendo o zero o preto e 255 o branco.

Na etapa da codificação os dados de entrada são mapeados para um conjunto limitado de vetores, com a mesma dimensão da entrada. Este conjunto denomina-se dicionário. Os vetores que constituem o dicionário são chamados de vetores-código ou representantes. A busca do VMP é realizada nesta etapa da quantização vetorial. Para cada vetor de entrada o codificador determina qual dos vetores-código do dicionário tem maior semelhança ou similaridade, ou seja, qual dos vetores-código tem a menor distância. Em outras palavras, determina-se o vizinho mais próximo. O vetor do dicionário com a maior similaridade terá seu índice enviado pelo canal (se a aplicação envolver transmissão) ou armazenamento (se a aplicação envolver armazenamento em memória).

Na decodificação, o mesmo dicionário é utilizado para reconstruir o sinal, o decodificador simplesmente recupera o vetor-código correspondente ao índice recebido. O sinal reconstruído será uma versão degradada do original. A decodificação é, portanto, uma etapa de baixa complexidade. Quando aplicado à compressão de imagens, a QV utiliza blocos de *pixels* da imagem de entrada. Para blocos de tamanho 4×4 e 8×8 *pixels*, por exemplo, tem-se quantização vetorial de dimensão 16 e 64 respectivamente.

Uma medida muito utilizada para avaliar a similaridade de vetores é a distância euclidiana quadrática, dada por

$$d(\vec{x_m}, \vec{w_i}) = \sum_{j=1}^{K} (x_{m,j} - w_{i,j})^2, \tag{2.1}$$

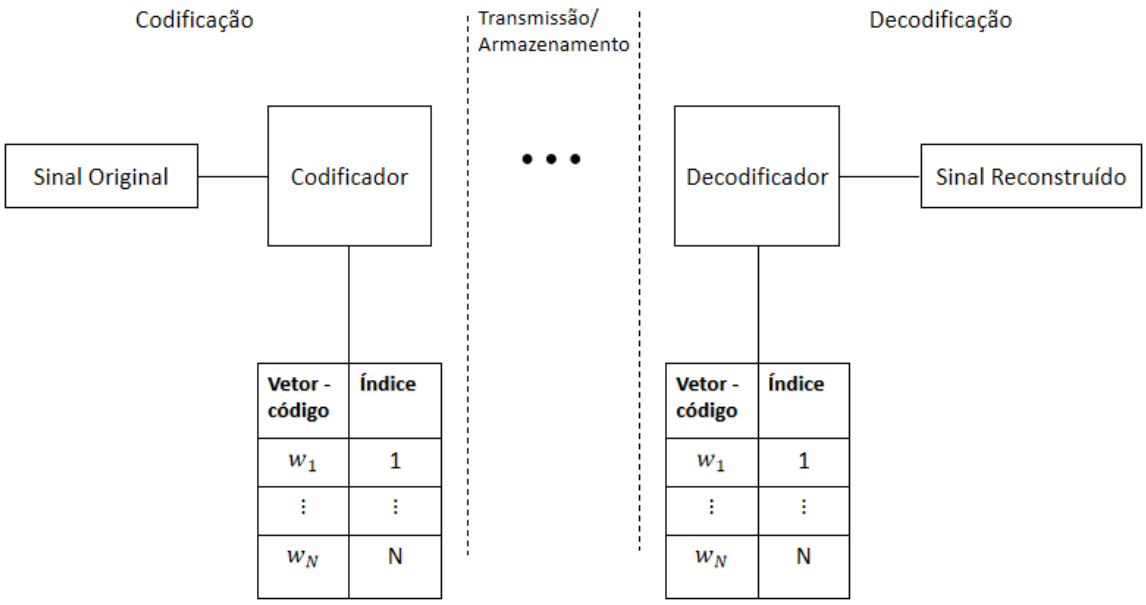
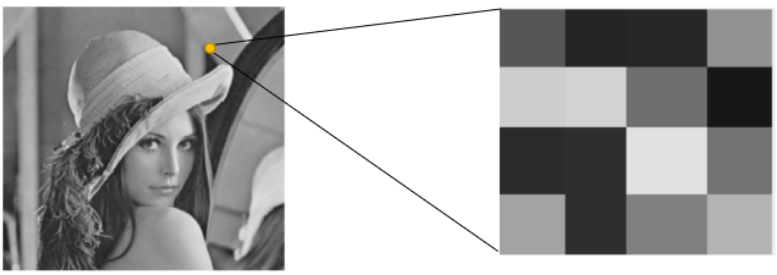


Figura 2.1: Sistema baseado em quantização vetorial.



86	37	39	145
205	210	111	23
43	46	225	114
163	46	128	180

Figura 2.2: Exemplo de um bloco de imagem 4×4 *pixels*.

em que $\vec{x_m}$ é o m-ésimo vetor de entrada, $x_{m,j}$ é a j-ésima componente de x_m e $w_{i,j}$ é a j-ésima componente do vetor $\vec{w_i}$.

Um quantizador Q de tamanho N e dimensão K mapeia vetores do conjunto de entrada $\vec{x_i} \in \mathbb{R}^K$ do espaço euclidiano, em um subconjunto finito W de mesma dimensão,

$$Q : \mathbb{R}^K \rightarrow W \tag{2.2}$$

em que W representa o dicionário, $W = \{\vec{w_i},\ i = 1, 2, .., N\}$ e $\vec{w_i}$ é o *i*-ésimo vetor-código. Para cada vetor-código é estabelecida uma região em sua vizinhança chamada de região de *Voronoi* representada por S. Para cada vetor de entrada é realizada uma avaliação, caso este vetor seja mais perto de um vetor-código do que qualquer outro listado no dicionário, então se diz que o vetor de entrada pertence a região de *Voronoi* definida por este vetor-código. Nessa etapa de busca do VMP é onde a QV concentra a maior parte de sua complexidade computacional. A versão quantizada de $\vec{x}$, denotada por $Q(\vec{x})$, é $\vec{w_i}$, se este vetor-código é o vizinho mais próximo de $\vec{x}$. Em outras palavras,

$$Q(\vec{x}) = \vec{w_i},\ se\ \vec{x} \in S_i, \tag{2.3}$$

assim definida:

$$S_i = \{\vec{x}\ :\ d(\vec{x}, \vec{w_i}) < d(\vec{x}, \vec{w_j})\ \forall\ i \neq j\}. \tag{2.4}$$

A imagem original é degradada pela quantização vetorial pois vetores de entrada pertencentes a uma região de *Voronoi* passam a serem substituídos pelo vetores representantes da região, o que leva a um erro na representação chamado de distorção, calculada por,

$$D = \sum_{i=1}^{N} \sum_{x_m \in S_i} d(\vec{x_m}, \vec{w_i}). \tag{2.5}$$

A taxa de codificação de um quantizador vetorial é calculada por $R = \frac{\log_2 N}{K}$, dada em bpp (*bits per pixel*), no cenário de codificação de imagem. A qualidade da imagem reconstruída portanto, está diretamente associada ao dicionário projetado utilizado.

Um algoritmo amplamente utilizado no projeto de dicionário para a QV é o LBG, que leva as iniciais de seus autores (*Linde, Buzo e Gray*) [9]. Nesse algoritmo iterativo, um dicionário inicial é fornecido como entrada e a cada iteração o dicionário é atualizado até que o seu critério de parada seja alcançado. Os passos do LBG são descritos a seguir:

Entradas: Limiar de distorção μ, dimensão do quantizador K, tamanho do dicionário N e o conjunto de treinamento.

- Inicialização: Um dicionário inicial W de tamanho N dimensão K é fornecido ou gerado, geralmente de forma aleatória, do conjunto de treino $X = \{\vec{x_1}, \vec{x_2}, ..., \vec{x_M}\}$.
- Particionamento: Nesta etapa é a realizada a busca do VMP e as regiões de *Voronoi* são formadas, S_i de acordo com a equação 2.4. Após a alocação de todos os vetores de entrada em seus grupos a distorção é calculada.
- Avaliação do critério de parada: O critério de parada é avaliado, caso não seja satisfeito o algoritmo prossegue. Este poderia ser por exemplo o número de iterações ou o limiar de distorção (Este último foi o critério utilizado neste trabalho), μ, definido pela diferença relativa da distorção anterior,D^{t-1}, com a atual,D^t, em que t representa a iteração

$$\mu = \frac{D^{t-1} - D^t}{D^{t-1}}. \tag{2.6}$$

- Atualização: Após a definição das regiões é realizado o cálculo do centróides de cada região. Este cálculo é realizado conforme indicado na Equação 2.7. O novo vetor-código da região passa a ser o centróide atual e o algoritmo se repete com um novo particionamento.

$$\vec{w_i} = \frac{1}{M_i} \sum_{\vec{x_m} \in S_i} \vec{x_m}, \tag{2.7}$$

em que x_m é o vetor de entrada pertencente região S_i e M_i é o número de vetores que compõem S_i.

2.2 Inicialização

O dicionário inicial é uma das entradas do algoritmo LBG. O processo doravante chamado de Inicialização consiste na apresentação deste dicionário inicial. Uma abordagem utilizada para gerar esse conjunto inicial de vetores é extrai-los do conjunto de treinamento aleatoriamente. É a chamada Inicialização Aleatória.

2.2.1 Hadamard

A inicialização de Hadamard foi introduzida por Chen e Li [35]. Essa técnica gera vetores-código iniciais por meio de um ordenamento dos M vetores de entrada, após terem sido levados ao domínio de Hadamard. A técnica se inicia aplicando a transformada de Hadamard a todos os vetores de entrada. Em seguida, são ordenados, de forma crescente, os vetores transformados considerando o valor de suas primeiras componentes, para então serem particionados em N grupos e um vetor-código inicial é extraído de cada grupo.

Considere um vetor de entrada $\vec{x} = (x_1, x_2, ..., x_K)$. Seja $H_{K\times K}$ a matriz do Hadamard de dimensão $K \times K$. A transformada do vetor de entrada é dado por $h(\vec{x}) = \vec{x} \cdot H_{K\times K}$. Após as transformadas de todos os vetores de entrada terem sido computadas, eles são ordenados considerando a primeira componente e divididos em N grupos de tamanhos iguais. O último passo é a seleção do vetor que se localiza no posição central de cada grupo como o vetor-código do dicionário inicial.

A transformada de Hadamard pode ser obtida através da multiplicação do vetor de entrada pela matriz de Hadamard. A matriz de Hadamard é uma matriz quadrada em que a dimensão é uma potência de 2, criada seguindo a seguinte regra de formação,

$$H_{2^m} = \frac{1}{\sqrt{2^m}} \begin{pmatrix} H_{2^{m-1}} & H_{2^{m-1}} \\ H_{2^{m-1}} & -H_{2^{m-1}} \end{pmatrix}, \tag{2.8}$$

em que m é um número inteiro não negativo e $H_1 = (1)$. Exemplos:

$$H_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.9}$$

$$H_4 = \frac{1}{\sqrt{4}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{2.10}$$

2.2.2 Estratégia de Grupos

Nesta inicialização, aqui chamada de Ma por ser o sobrenome de um dos seus autores, o objetivo é fazer uso dos parâmetros de variância e média dos M vetores do conjunto treino de dimensão K, para selecionar o dicionário inicial [44] . Os cálculos da média e variância são expressos nas equações 2.11 e 2.12 respectivamente, onde K representa a dimensão do vetor $\vec{x}$:

$$m_x = \frac{1}{K}\sum_{j=1}^{K} x_j, \tag{2.11}$$

$$v_x = \frac{1}{k}\sum_{j=1}^{K}(x_j - m_x)^2. \tag{2.12}$$

O artigo [44] tem como conjunto de treinamento as imagens *Lena, Peppers, Airplane, Splash, Girl* e *Lake*. Os autores demonstram que a maioria dos vetores de entrada apresentam variância muito baixa, um pequeno grupo apresenta variância intermediária e um grupo ainda menor com alta variância, quando os dados a serem quantizados são provenientes dessas imagens. Vetores de alta variância geralmente são vetores de bordas e por isso carregam muita informação.

O procedimento para a geração do dicionário inicial segue os seguintes passos:

1. Calcular a variância de todos os vetores-treino.
2. Ordenar os vetores-treino em ordem ascendente de variância.
3. Separar os vetores ordenados em 3 grupos A, B e C na proporção 17:2:1, ou seja, 84% pertencem ao grupo A, 10% ao grupo B e os demais ao grupo C.
4. Para os Grupos A e B:
   - Calcular a média de cada vetor pertencente ao grupo.
   - Ordenar de forma crescente de acordo com o valor da média.
   - Subdividir o grupo A em $N/2$ e o B em $N/4$ subgrupos.
   - De cada subgrupo extrair o vetor do meio como vetor-código.
5. Selecionar aleatoriamente $N/4$ vetores do Grupo C.

2.2.3 *Subtractive Clustering*

Nesse método de inicialização se faz uso da ideia de que cada vector de entrada é um possível centróide [36]. Para selecionar os melhores candidatos do conjunto de entrada é feito o cálculo da densidade de cada vetor. Essa medida é obtida por meio da equação 2.13 para um vetor de entrada x_i.

$$D_i = \sum_{j=1}^{M} \exp\left(-\frac{\|\vec{x_i} - \vec{x_j}\|^2}{(\frac{r_a}{2})^2}\right), \tag{2.13}$$

em que r_a é um valor positivo e representa o raio da vizinhança e $\vec{x_j}$ é um outro vetor também do conjunto de entrada. O vetor $\vec{c_1}$ que possuir a maior densidade, D_{c_1} será escolhido como o primeiro vetor-código do dicionário. Após essa etapa, as densidades são atualizadas de acordo com a equação

$$D_i = D_i - D_{c_1} \cdot \exp\left(-\frac{\|\vec{x_i} - \vec{c_1}\|^2}{(\frac{r_b}{2})^2}\right), \tag{2.14}$$

em que r_b é uma constante positiva que define uma vizinhança de menor densidade. O segundo vetor-código será aquele que possuir a maior densidade a partir da equação 2.14 e o processo se repete até formar o dicionário inicial.

2.2.4 KATSA

A principal ideia da técnica proposta por Katsavounidis, Kuo e Zhang [30] é a distância entre os vetores de entrada. Vetores distantes apresentam maior probabilidade de pertencerem a diferentes regiões de Voronoi e, portanto, são selecionados como vetores-código do dicionário inicial.

O procedimento adotado nesta inicialização pode ser descrito pelos passos seguintes:

1. É calculada a norma de todos os vetores de treino $\vec{x_m}$. O vetor de maior norma, $\vec{c_1}$, será escolhido como o primeiro do dicionário.
2. São calculadas as distâncias dos demais vetores treino para o primeiro vetor-código, $\vec{c_1}$. Aquele que apresentar a maior distância será tido como o segundo vetor do dicionário, $\vec{c_2}$.
3. Para os demais vetores de entrada são calculadas as distâncias para os vetores-código, a menor distância encontrada será chamada de distância do dicionário daquele vetor. O vetor do conjunto de entrada que apresentar a maior distância de dicionário será incorporado ao dicionário. O processo se repete até atingir o tamanho desejado, N.

2.2.5 MEIM

A inicialização MEIM (*Maximum Entropy Initialisation Method*) foi proposta por Nyeck e Tosser-Roussey [45]. Os autores observaram que em algumas técnicas de inicialização os vetores-código iniciais não participavam de forma equivalente no processo de quantização, ou seja, alguns vetores iniciais eram representantes de uma quantidade muito maior de vetores de entrada do que outros vetores iniciais, em alguns casos, alguns dos vetores iniciais poderiam nem serem utilizados. Com o objetivo de uniformizar a quantidade de vetores de entrada por região de Voronoi e acelerar a convergência do quantizador, a técnica busca maximizar a entropia do dicionário. A entropia do dicionário é dada por

$$E = -\sum_i p_i \log_2(p_i), \tag{2.15}$$

em que p_i é a frequência relativa do i-ésimo vetor-código i no conjunto treino. O aumento da entropia se reflete na criação de regiões de Voronoi de tamanhos próximos. A técnica segue as seguintes etapas:

1. Um dicionário inicial W, de dimensão K, é fornecido de tamanho N, $W = \{\vec{w_1}, ..., \vec{w_N}\}$. O particionamento de Y_0 é representado por $P(W) = \{S_1, ..., S_N\}$, em que S_i é um *cluster*. A quantidade de vetores em cada *cluster*, N_i, é inicialmente um.
2. Seleciona um vetor do conjunto de entrada $\vec{x} = \{x_1, ..., x_K\}$
3. Testa a condição: $N_i.d(\vec{x}, \vec{w_i}) < N_j.d(\vec{x}, \vec{w_j})$ para $i, j = \{1, ..., N \mid i \neq j\}$. Caso verdadeira, $\vec{x} \in S_i$. Ressalta-se que $d(\vec{x}, \vec{w_i})$ é a distância euclidiana.
4. O processo se repete a partir da etapa 2 para todos os vetores de entrada.
5. Após a alocação completa dos vetores o cálculo dos centróide é feito a partir de expressão:

$$\vec{c_i} = \frac{1}{N_i} \sum \vec{x_j} \epsilon S_i, \tag{2.16}$$

em que N_i é o número de vetores do conjunto de treino pertencentes à região de Voronoi S_i.Tem-se, assim, o dicionário inicial $C = \{\vec{c_1}, ..., \vec{c_N}\}$.

2.2.6 DSICS

O método Seleção dos Vetores-Código Iniciais Baseado em Dupla Ordenação DSICS (do inglês, *Double Sorting based Initial Codewords Selection*) foi introduzido por Hu e colaboradores [34]. Esse método combina duas ideias centrais, a de ordenação dos vetores de entrada baseada na distância destes à origem e a ordenação dos vetores de entrada considerando a soma de suas componentes.

Dado o conjunto de entrada, $X = \{\vec{x_1}, ..., \vec{x_M}\}$, em que $\vec{x_i}$ é um vetor de dimensão K, duas cópias desse conjunto são feitas, C_1 e C_2. Utilizando os vetores em C_1, calcula-se a distância euclidiana destes até a origem e na sequência os vetores são ordenados de maneira crescente a partir de suas distâncias.

Em C_2 os vetores têm suas componentes somadas. O valor da soma de cada vetor é utilizado para ordena-los de forma crescente. Ambos os grupos C_1 e C_2 são divididos igualmente em N subgrupos c_{1i} e c_{2i} respectivamente.

A seleção dos vetores inicias é feita da seguinte forma: Caso haja vetores em comum num mesmo subgrupo i, ou seja, $c_{1i} \cap c_{2i} \neq \emptyset$ o vetor ocupante da posição central é escolhido como o representante do subgrupo. Se a interseção dos subgrupos for vazia, $c_{1i} \cap c_{2i} = \emptyset$, o vetor selecionado será o central do subgrupo c_{1i}.

2.3 Busca do Vizinho Mais Próximo

A busca pelo vizinho mais próximo pode ser entendida como a busca, no contexto da QV, por um vetor do dicionário que melhor represente um dado vetor de entrada.

Supondo um vetor de entrada e uma métrica de similaridade, como a distância euclidiana, uma técnica de busca do vizinho mais próximo percorre o dicionário e retorna com aquele vetor que apresentar a maior similaridade ou, no caso, menor distância em relação ao vetor de entrada. Caso o conjunto possua muitos elementos ou estes sejam vetores multidimensionais, a tarefa de encontrar um vizinho se torna cada vez mais desafiadora.

Os algoritmos de busca do vizinho mais próximo buscam reduzir a quantidade de operações lógicas/aritméticas neste processo de busca. Na etapa de codificação da QV a seleção dos representantes é por meio da similaridade entre os vetores e é nesta etapa que as técnicas de busca do vizinho mais próximo podem ser utilizadas para aumentar a eficiência da codificação, em termos de tempo, reduzindo a complexidade computacional.

2.3.1 *Partial Distortion Search*

O algoritmo de busca por distorção parcial [41] reduz o número de operações aritméticas na medida que interrompe o cálculo da distância entre o vetor de entrada e o vetor-código. Se no cálculo da distância o valor acumulado até uma determinada componente $j < K$, a distância parcial for maior do que a menor distância até então calculada, o cálculo da distância atual é interrompido e reinicia o processo, novo cálculo da distância, para o próximo vetor-código.

Considerando d e d_{min} a distância atual e mínima respectivamente, $\vec{x}$ um vetor de entrada de dimensão K e $\vec{w_i}$ o i-ésimo vetor do dicionário, o cálculo da distância entre $\vec{w_i}$ e $\vec{x}$ é interrompido se em algum $j < K$, o valor de $d > d_{min}$, em que j é a j-ésima componente.

2.3.2 *Improved Equal-average Equal-variance*

O algoritmo IEENNS é integrante da família dos algoritmos de busca do vizinho mais próximo que deriva da ENNS [46]. Neste algoritmo as informações da média e variância das componentes dos vetores são utilizadas para estabelecer critérios de rejeição do vetor-código. Os cálculos das médias e variâncias dos vetores de entrada são realizados antes de iniciar a etapa de codificação.

Durante a cada etapa de particionamento, as médias e variância dos vetores-código são calculadas. Em seguida, estes vetores são ordenados de acordo com suas médias, em ordem crescente. A busca se inicia quando se calcula a distância euclidiana quadrática d_{min} de um vetor de entrada $\vec{x}$ para aquele vetor-código w_i que possui a média de suas componentes mais próxima da média de $\vec{x}$. Considerando m_x e m_i as médias das componentes dos vetores de entrada e do vetor-código respectivamente e K a dimensão dos vetores, w_i é descartado da busca se satisfizer a Inequação 2.17:

$$m_i \geq m_x + \sqrt{d_{min}/K} \text{ ou } m_i \leq m_x - \sqrt{d_{min}/K}. \tag{2.17}$$

O busca segue alternando o sentido que percorre a lista de vetores-código, ora acima da posição do vetor w_i e ora abaixo. Caso a condição da Inequação 2.17 não seja atendida o método segue utilizando os valores do desvio padrão com a seguinte avaliação

$$K(m_x - m_i)^2 + (\sigma_x - \sigma_i)^2 \geq d_{min}, \tag{2.18}$$

em que σ_x e σ_i representam o desvio padrão dos vetores de entrada e do vetor-código, respectivamente. Se nenhuma das condições supracitadas forem satisfeitas, a busca do VMP é realizada aplicando o método PDS.

3
Algoritmo Cuckoo-LBG

3.1 Fundamentos

A Inteligência de Enxames ou a Inteligência Coletiva é uma subárea da Inteligência Computacional. As técnicas de enxame começaram a surgir na década de 80 e têm como uma das suas principais aplicações problemas de otimização [47]. Tais técnicas possibilitam a resolução de problemas de alta complexidade ou aqueles que não apresentam ainda uma solução analítica. A principal inspiração dos algoritmos de enxame é a natureza, sendo a maioria dos algoritmos resultados das observações dos comportamentos de animais coletivos como peixes, como é o caso do algoritmo *Fish School Search* (FSS) [48], insetos, a exemplo de *Firefly Algorithm* (FA) [14, 49], pássaros, como é o exemplo do algoritmo *Particule Swarm Optimization* (PSO) [15] e mamíferos, como exemplo o *Spider Monkey Optimization* (SMO) [50].

Uma das ideias em que se baseia a Inteligência Coletiva é que grupos de indivíduos, apesar de realizarem tarefas simples, quando em conjunto, são capazes de alcançar resultados que os indivíduos sozinhos seriam incapazes de obter [51]. No enxame, a cada indivíduo, que quando abstraído pelo modelo é denominados de partícula, são atribuídas tarefas e ao fim delas uma avaliação é feita utilizando uma métrica, figura de mérito. As partículas do enxame têm suas posições atualizadas a cada iteração do algoritmo e trocam informações entre si sobre avaliação de sua posição atual e o meio. Ao final da movimentação de todo o enxame, a partícula melhor avaliada tem sua posição armazenada e o processo se repete até que o critério de parada do algoritmo seja alcançado. Para explorar o espaço de busca os algoritmos procuram equilibrar a exploração local e global, desta maneira, buscam evitar ficarem presos em mínimos (ou máximos) locais para aumentar a eficiência da busca na região explorada.

As técnicas de enxames foram aplicadas à resolução de uma gama problemas em diversas áreas como engenharia [52, 53], saúde [54, 55] e economia [56]. Devido à versatilidade dos algoritmos de enxames eles são combinados com outros algoritmos em abordagens híbridas como o caso do *Interwined K-means and PSO*, *Firefly Algorithm-LBG* (FA-LBG), *modified Firefly Algorithm - LBG* (m-FA-LBG), *Fish School Search-LBG new breed* (FSS-LBG-NB) [18, 21, 22]. Esses dois últimos trabalhos aplicaram as técnicas híbridas para o projeto de quantizadores vetoriais. Neste trabalho é proposta uma abordagem, também híbrida, para o projeto de dicionário a partir do algoritmo, LBG e do *Cuckoo Search*, chamada de Cuckoo-LBG.

3.2 *Cuckoo Search*

O comportamento de pássaros da família *Cuculidae*, conhecidos popularmente por cucos, é a inspiração para o algoritmo de busca *Cuckoo Search* [57]. Especificamente o comportamento relativo ao seu hábito de reprodução. Pássaros cuco exibem um atitude parasitária chamada de parasistimo de ninho. Nesse tipo de parasistimo, as fêmeas da espécie põem seus ovos em ninhos de outras espécies, fazendo com que as fêmeas hospedeiras choquem e alimentem o filhote cuco [58].

Para ter sucesso em sua estratégia reprodutiva, algumas espécies de cuco conseguem até mimetizar seus ovos com os existentes no ninho hospedeiro. Em geral, o tempo para um ovo cuco eclodir é menor do que é para seus "irmãos", assim, quando nasce este filhote seu primeiro movimento instintivo é expulsar os demais ovos do ninho, assim aumenta a porção de alimentos destinados a si. É possível que o ovo estrangeiro seja detectado pela espécie hospedeira, o que resulta em dois possíveis cenários. No primeiro, o ninho é abandonado e um novo é construído noutro lugar, no segundo cenário, o ovo cuco é lançado para fora do ninho.

O algoritmo *Cuckoo Search* realiza a movimentação dos pássaros por meio do *Levy Flight* (do inglês, Voo de Levy). *Levy Flight* se baseia na distribuição de probabilidade de Lévy, um matemático francês (1886-1971), em que há uma combinação de passos curtos e longos [59]. Os passos do Voo de Levy são obtidos de sua distribuição, que se assemelha com uma distribuição exponencial. Mantegna desenvolveu um método [60] para se obter o tamanho do passo da distribuição de Levy por meio de distribuição uniforme. Os passos podem sem obtidos pelo método segundo o trabalho de *Qi et. al* [61]. Alguns estudos demonstram que algumas espécies de animais se movimentam segundo essa metodologia [62–64]. O voo de Levy foi aplicado em estudos para otimizar métodos de busca em algoritmos [62, 65, 66].

No *Cuckoo Search* três regras precisam ser atendidas:

1. Cada pássaro pode depositar apenas um ovo por ninho e este ninho é escolhido de maneira aleatória,
2. Os melhores ninhos, maior valor de *fitness*, sobrevivem para a próxima geração,
3. A quantidade inicial de ninhos é fixa e existe uma probabilidade p_a dos ovos depositados serem descobertos. Caso isso ocorra, o ninho é abandonado ou o ovo estrangeiro é derrubado do ninho.

É importante destacar que o conceito de ninho, cuco e ovos é o mesmo, e todos eles são metáforas para a solução do problema. Essa última regra na prática indica que uma porcentagem, p_a, dos n ninhos é substituída por novos.

A equação da atualização do movimento para um cuco, $\vec{x_i}$, é

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \alpha \cdot L(s, \lambda), \tag{3.1}$$

em que t representa a iteração atual, α é um fator de escala do passo, s é o tamanho do passo e $L(s, \lambda)$ é dado por

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) sen(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \tag{3.2}$$

em que $1 < \lambda \leq 3$ e Γ é a função gamma. No artigo [67], os autores propõem uma alternativa de movimentação, descrita pela equação 3.3.

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \alpha \cdot s \otimes H(p_a - \epsilon) \otimes (\vec{x}_i^t - \vec{x}_j^t), \tag{3.3}$$

em que $\otimes$ é uma operação de multiplicação entre os elementos em posições correspondentes, $H(u)$ é a função de *Heaviside*, também conhecida como função degrau, descrita em 3.4 com

descontinuidade em a, $\epsilon \in [0, 1]$ e $\vec{x_j}$ é um vetor escolhido aleatoriamente.

$$H(x - a) = \begin{cases} 0, & x < a \\ 1 & x \geq a \end{cases}. \tag{3.4}$$

O pseudo-código do algoritmo do *Cuckoo Search* é exibido em Algoritmo 1. A Figura 3.1 apresenta o diagrama de blocos do algoritmo.

3.2.1 Pseudocódigo

Algoritmo 1: Cuckoo Search via Lévy Flights

Entrada: $f(\vec{x})$ - função objetivo;
$\vec{x} = (x_1, ..., x_K)$ - vetor representando um cuco de dimensão K;
n - número de cucos gerados inicialmente;
t_{max} - Máximo de iterações;
p_a - fração de cucos descobertos;
início
  enquanto $t < t_{max}$ **faça**
    selecionar um cuco, $\vec{x_i}$ e realizar um voo de Levy;
    avaliar o seu *fitness* (F_i);
    escolher um ninho, $\vec{x_j}$ tal que $i \neq j$;
    se $F_i > F_j$ **então**
      substituir x_j pela nova solução
    fim
    Uma fração p_a dos piores ninhos são abandonados e novos ninhos são criados;
    Ordenar os cucos pelo *fitness* e armazenar o pássaro de maior *fitness*;
  fim
fim

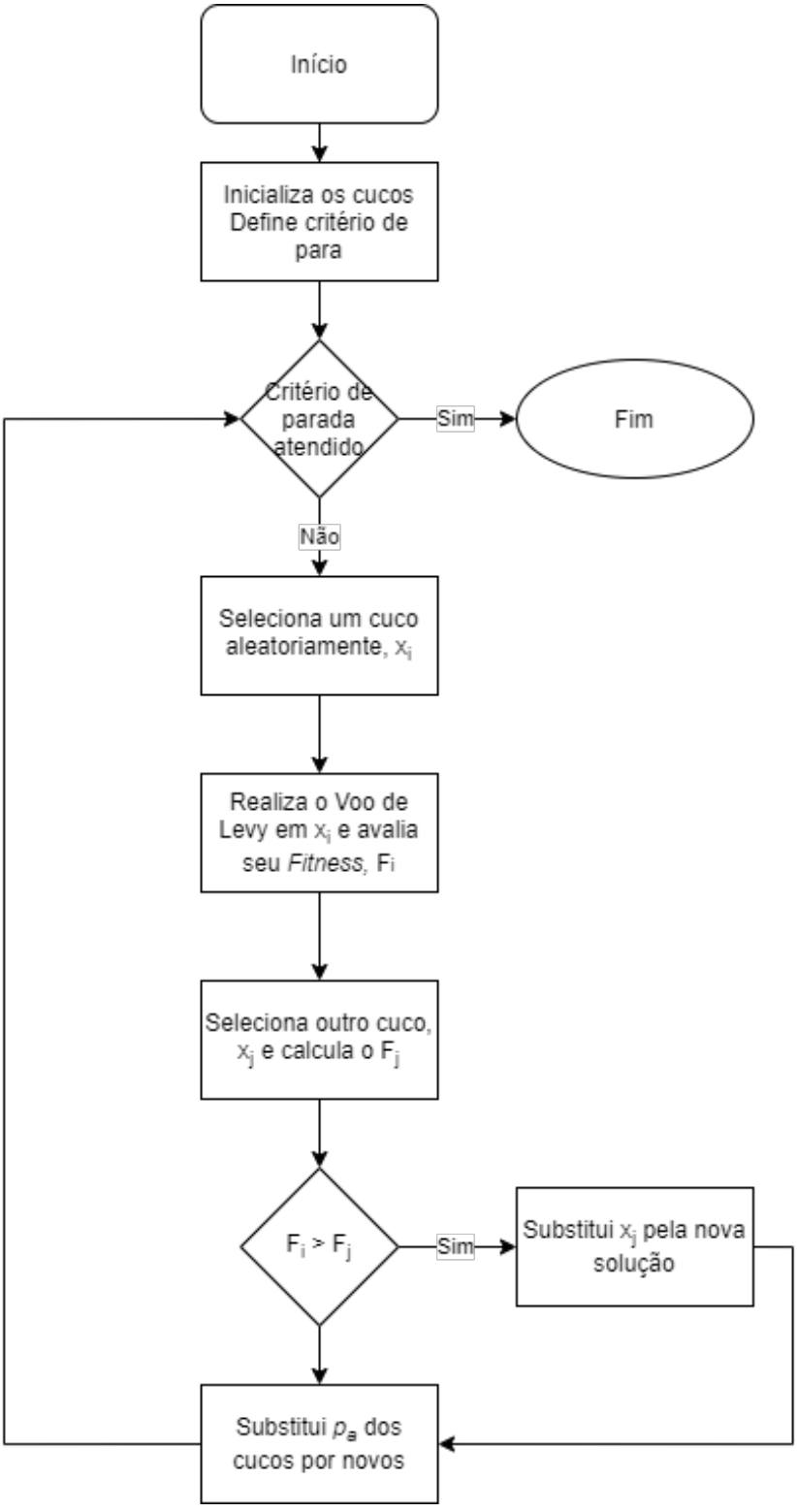


Figura 3.1: Diagrama de Blocos do Algoritmo *Cuckoo Search*.

3.3 Técnica Proposta: Cuckoo-LBG

O algoritmo Cuckoo-LBG é uma técnica proposta para o projeto de dicionário no contexto da quantização vetorial. O Cuckoo-LBG tem como objetivo produzir dicionários que levem a imagens reconstruídas com qualidade melhor que as obtidas usando dicionários projetados com

técnicas que são estado da arte.

Nos algoritmos de enxame, cada partícula é uma possível solução, que atualiza sua posição na tentativa de melhorar a qualidade de sua solução. No algoritmo proposto cada dicionário é uma partícula e a função objetivo corresponde ao inverso da distorção média, ou seja, $1/D$. Portanto, a técnica de enxame é aplicada orientada à maximação da função objetivo.

No algoritmo Cuckoo-LBG cada dicionário é uma matriz de dimensão $N \times K$, em que N e K são o tamanho e a dimensão do dicionário respectivamente. Um dicionário, W, é composto por vetores-código $\vec{w}_i$ de dimensão K, o número de vetores-código é igual ao tamanho do dicionário, N.

No algoritmo proposto, os cucos iniciais são gerados de acordo com as estratégias de inicialização apresentadas na Seção 2.2 utilizando os vetores do conjunto de entrada. Caso a inicialização utilizada não seja especificamente mencionada, deve se considerar como uma inicialização aleatória para todos n cucos.

Após a inicialização, todos os dicionários são particionados e a distorção média de cada um deles é calculada. Na sequência, os dicionários são atualizados seguindo a Equação 3.5 e o valor da distorção média, de cada partícula, novamente computado.

$$\vec{w}_i^{t+1} = \vec{w}_i^t + \alpha \cdot s \otimes H(p_a - \epsilon) \otimes (\vec{w}_i^t - \vec{w}_j^t). \tag{3.5}$$

Um novo voo é realizado para uma fração p_a dos dicionário com os piores desempenhos. Neste caso, a equação de atualização, Equação 3.5 é alterada para

$$\vec{w}_i^{t+1} = \vec{w}_i^t + \alpha \cdot s \cdot (\vec{w}_i^t - \vec{w}_{best,i}^t), \tag{3.6}$$

em que $\vec{w}_{best,i}^t$ é uma vetor-código do dicionário com o melhor desempenho na iteração atual.

Uma iteração do Cuckoo-LBG estará completa quando os dicionário tiverem seus valores atualizados pelo CS e o algoritmo LBG tiver sido aplicado a cada dicionário, a busca do vizinho mais próximo ter sido realizada e a distorção média ter sido calculada. Ao fim de uma iteração, o cuco que possuir o maior *fitness* seguirá para a próxima iteração e, ao fim da execução, o melhor cuco será o dicionário final projetado pelo algoritmo. A Figura 3.2 apresenta o diagrama de blocos do algoritmo Cuckoo-LBG.

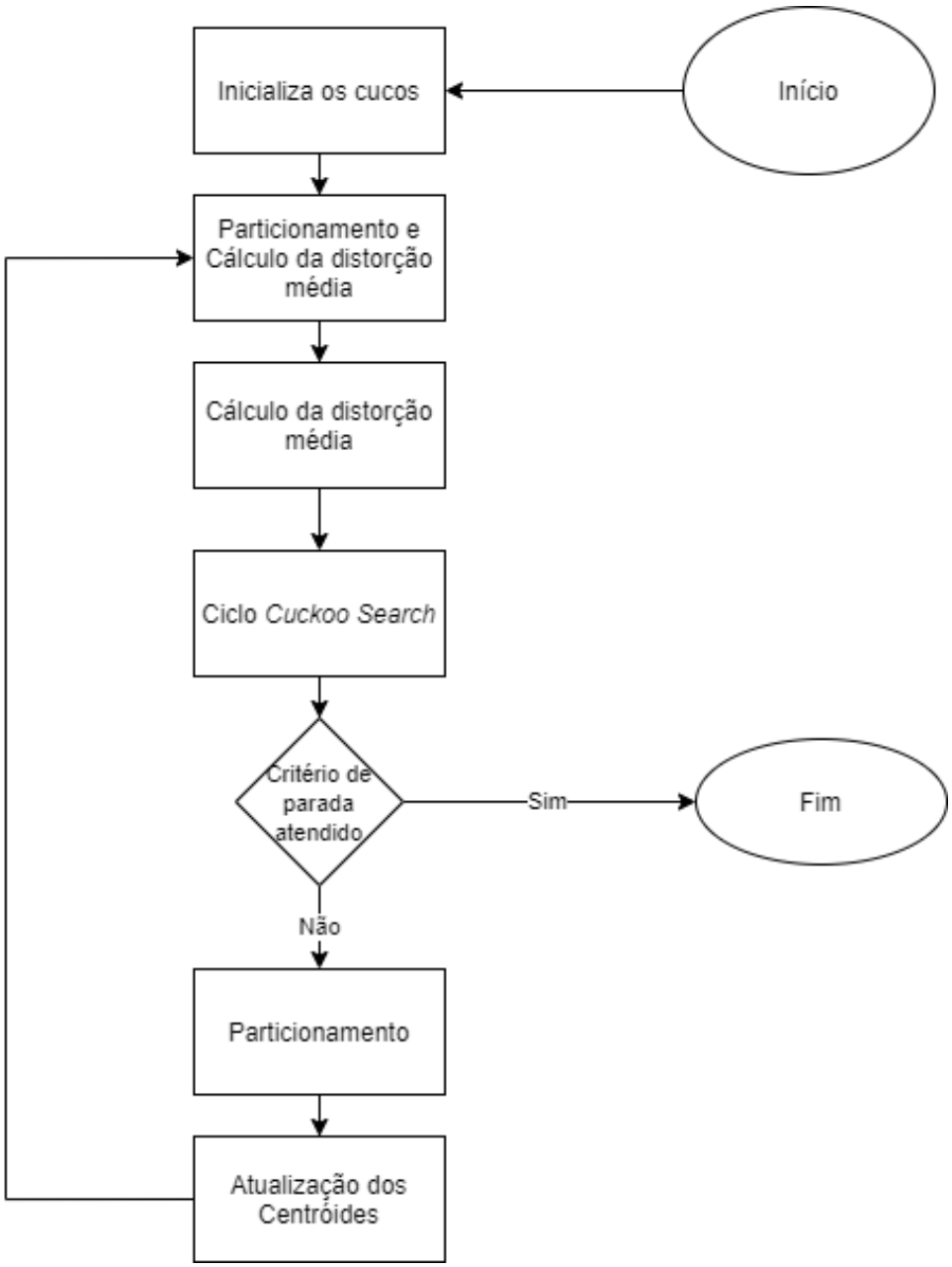


Figura 3.2: Diagrama de Blocos do Algoritmo Cuckoo-LBG.

4
Metodologia

Este capítulo apresenta a metologia aplicada ao desenvolvimento do trabalho, relativa a implementações computacionais e à análise dos resultados.

Todas as execuções computacionais foram realizadas na mesma máquina no sistema operacional *Windows 10*. Os resultados obtidos são apresentados em termos de valores de PSNR (em dB) e SSIM [68] das imagens reconstruídas para as estratégias de inicialização consideradas. Foram utilizadas quatro imagens monocromáticas para as implementações computacionais, sendo elas: *Barbara, Clock, Lena e Peppers* no formarto PGM (*do inglês, Portable Gray Map*), de tamanho 256×256, *pixels* codificadas a 8,0 *bits* por *pixel*. A dimensão do quantizador foi $K = 16$, ou seja, foram quantizados blocos de 4×4 *pixels*, e dicionário de tamanho $N = 32, 64, 128, 256, 512$. O limiar utilizado como critério de parada do algoritmo de projeto de dicionário foi $\mu = 0,001$. Para cada imagem foram projetados 30 dicionários gerados para cada tamanho de dicionário, N, com diferentes abordagens de inicialização para efeito de comparação. As técnicas de inicialização escolhidas foram: Seleção Aleatória, Inicialização de Hadamard, Subtrective Clustering, Inicialização de quantizadores proposta por Ma *et al.*, DSCIS, MEIM e Inicialização KATSA.

Para cada técnica de enxame considerada, foram utilizadas dez partículas, razão pela qual em cada execução da técnica são inicializados dez dicionários, em que cada dicionário corresponde a uma partícula. Após o treinamento, o dicionário com melhor desempenho é escolhido como o dicionário final. Diferentes arranjos foram projetados para a inicialização desses dicionários.

As inicializações são realizadas considerando grupos de técnicas. Conforme a Figura 4.1 cada letra utilizada na sigla representa uma inicialização Por exemplo, se tivermos dez dicionários na execução, a estratégia MS equivale a: um dicionário inicializado pela técnica Ma (M), outro pela Subtractive Clustering (S) e os demais são inicializados aleatoriamente (A). A figura 4.1 exibe um diagrama para simbolizar a estratégia MS, onde os dez dicionários são representados pelos retângulos com a técnica utilizada para sua inicialização descrita em seu interior por meio de sua sigla. A Tabela 4.1 contém na coluna da extrema esquerda o nome do conjunto de inicializações propostas no trabalho e na primeira linha da tabela, a forma abreviadas das técnicas de inicialização utilizadas para compor estes conjuntos.

Especificações da Máquina utilizada: Windows 10, processador Intel(R) Core (TM) i5-

A	A	A	A	A	A	A	A	M	S

Figura 4.1: Ilustração de um conjunto de inicializações com 10 dicionários.

Tabela 4.1: Conjuntos de Inicializações Avaliadas

	Aleatória	DSICS	MEIM	KATSA	MA	S.B.	HAD.
Aleatória	10	-	-	-	-	-	-
DsMe	8	1	1	-	-	-	-
DsKt	8	1	-	1	-	-	-
MeKt	8	-	1	1	-	-	-
MH	8	-	-	-	1		1
SH	8	-	-	-		1	1
MS	8	-	-	-	1	1	-
DsMeKt	7	1	1	1	-	-	-
MHS	7	-	-	-	1	1	1
Todos	4	1	1	1	1	1	1

8250U CPU @1.60GHz 1.80GHz e 8 GB de memória RAM. As implementações foram realizadas na linguagem de programação C/C++.

Relação Sinal-Ruído de Pico (PSNR, *Peak signal to noise ratio*) está definida na equação 4.1, em dB. A métrica foi utilizada para avaliar o desempenho dos dicionários projetados pelas técnicas.

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left[\frac{v_p^2}{\mathrm{MSE}} \right], \tag{4.1}$$

em que v_p é o valor de pico. Para uma imagem de 8 bit por pixel, $v_p = 255$.

O Erro Médio Quadrado (MSE, do inglês *Mean Squared Error*) é calculado por

$$\mathrm{MSE} = \frac{1}{T_1 \times T_2} \sum_{l=0}^{T_1-1} \sum_{c=0}^{T_2-1} [I(l,c) - I'(l,c)]^2, \tag{4.2}$$

em que $I(l,c)$ e $I'(l,c)$ representam a intensidade do pixel nas posições de linha l e coluna c da imagem original e da imagem reconstruída, respectivamente. T_1 e T_2 são as dimensões (número de *pixels* por linha e por coluna da imagem). Utilizou-se também, para avaliação de desempenho, a medida SSIM, dada por

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1) + \left(2\sigma_{x_y} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4.3}$$

em que μ_x e μ_y, representam a média da janela x e y ambas de tamanho $m \times n$, respectivamente. σ_x^2 e σ_x^2 são os valores da variância de x e y, σ_{x_y} corresponde a covariância entre x e y e por fim C_i são variáveis utilizadas para estabilizar a divisão e evitar denominadores próximos de zero, em que $C_1 = (0,01 \times L)^2$ e $C_2 = (0,03 \times L)^2$, em que L por padrão é dado por $L = 2^{\#bits\,por\,pixel} - 1$.

(a) Barbara.

(b) Clock.

(c) Lena.

(d) Peppers.

Figura 4.2: Imagens utilizadas como o conjunto de treinamento

5
Sistemas Simulados e Resultados

Os resultados obtidos por meio das simulações para os dicionários projetados a partir de técnicas baseadas em quantização vetorial serão apresentados neste capítulo. O capítulo foi dividido em 3 seções. A Seção 5.1 exibe os resultados do desempenho do algoritmo Cuckoo-LBG para as imagens reconstruídas utilizando a técnica proposta. É realizada uma comparação de desempenho, em termos de PSNR e SSIM das imagems reconstruídas, com outras técnicas híbridas (PSO-LBG, FSS-LBG-NB, FA-LBG) e com o algoritmo LBG. Na Seção 5.2 são apresentados os resultados do uso de diferentes estratégias de inicialização aplicadas ao projeto de quantizadores vetoriais com o algoritmo Cuckoo-LBG. Os resultados em termos do tempo de execução com o uso das técnicas de busca do vizinho mais próximo estão na Seção 5.3.

5.1 Resultado do Algoritmo Cuckoo-LBG

Os valores médios de PSNR (dB) da imagem *Barbara* são apresentados na Tabela 5.1. Para os tamanhos de dicionário variando de 32 até 256, as técnicas híbridas obtiveram resultados semelhantes, porém para o $N = 512$ o FSS-LBG-NB e o FA-LBG foram superiores. Em todos os casos o Cuckoo-LBG foi superior em termos de PSNR quando comparado com algoritmo LBG observa-se que a superioridade do Cuckoo-LBG sobre LBG aumenta com o tamanho do dicionário, N.

A Tabela 5.2 é referente aos resultados da imagem *Clock*. Para esta imagem a técnica Cuckoo-LBG leva aos maiores valores de PSNR para todos os tamanhos exceto para $N = 512$. O ganho de PSNR das imagens reconstruídas, obtido com o uso de dicionários Cuckoo-LBG em substituição aos dicionários LBG, para o tamanho $N = 256$ é de 1,30 dB e de 1,53 dB para $N = 512$.

A Tabela 5.3 apresenta os resultados para a imagem *Lena*. Para $N \geq 256$, a substituição de dicionários LBG por dicionários Cuckoo-LBG leva a ganhos de PSNR superiores a 0,60 dB. Para N = 512, particularmente, a substituição supracitada leva a um ganho de 1,14 dB.

Conforme se observa na Tabela 5.4, para a imagem *Peppers*, a superioridade do Cuckoo-LBG sobre o LBG, em termos de PSNR das imagens reconstruídas, é observado para todas os valores de N. Precisamente, essa superioridade aumenta com o tamanho do dicionário.

Dentre as técnicas de enxame, o melhor desempenho é apresentado por FA-LBG. Para $N \leq 128$, as técnicas de enxame têm desempenho próximo.

Tabela 5.1: Valores médios de PSNR (dB) da imagem Barbara.

N	Técnica				
	LBG	FA-LBG	PSO-LBG	FSS-LBG-NB	Cuckoo-LBG
32	24,75	**24,81**	24,79	24,79	**24,80**
64	25,67	**25,80**	**25,80**	**25,80**	25,79
128	26,68	**26,88**	26,82	**26,85**	26,84
256	27,76	**28,13**	28,02	**28,09**	28,08
512	29,06	**29,88**	29,64	**29,84**	29,63

Tabela 5.2: Valores médios de PSNR (dB) da imagem Clock.

N	**Técnica**				
	LBG	FA-LBG	PSO-LBG	FSS-LBG-NB	Cuckoo-LBG
32	26,23	26,65	**26,70**	26,68	**26,72**
64	27,23	27,81	**27,90**	27,84	**27,97**
128	28,29	29,17	**29,22**	29,04	**29,25**
256	29,50	**30,76**	30,72	30,36	**30,80**
512	30,87	**32,60**	**32,48**	32,05	32,40

Tabela 5.3: Valores médios de PSNR (dB) da imagem Lena.

N	Técnica				
	LBG	FA-LBG	PSO-LBG	FSS-LBG-NB	Cuckoo-LBG
32	26,61	**26,67**	**26,68**	**26,67**	**26,67**
64	27,74	**27,89**	**27,89**	**27,89**	27,88
128	28,83	**29,20**	**29,19**	29,17	29,14
256	29,90	**30,71**	30,64	**30,67**	30,52
512	31,11	**32,79**	**32,70**	32,61	32,25

Tabela 5.4: Valores médios de PSNR (dB) da imagem Peppers.

N	Técnica				
	LBG	FA-LBG	PSO-LBG	FSS-LBG-NB	Cuckoo-LBG
32	26,07	**26,15**	**26,15**	**26,15**	**26,15**
64	27,36	**27,47**	**27,51**	27,47	27,45
128	28,52	**28,84**	**28,83**	**28,83**	28,78
256	29,65	**30,49**	**30,42**	30,40	30,27
512	30,93	**32,72**	**32,55**	32,42	32,22

5.1.1 Resultados Relativos aos Valores de SSIM dos Algoritmos de Projeto de Dicionário para as Imagens Reconstruídas

A Tabela 5.5 contempla os valores médios da métrica SSIM para cada imagem reconstruída. Para a imagem *Barbara*, a técnica Cuckoo-LBG é levemente superior em todos os tamanhos quando comparada ao algoritmo LBG

Para a imagem *Clock* a substituição da técnica LBG por Cuckoo-LBG aumentou o valor do SSIM médio das imagens reconstruídas, para todos os tamanhos de dicionário projetados. É possível destacar, para a imagem *Clock*, que a técnica proposta exibiu os maiores valores de SSIM, quando comparada com as demais técnicas, para os tamanhos 64, 128 e 256. Quando observados os valores de SSIM médios, relativos à imagem *Lena*, entre as técnicas híbridas de projeto de dicionário, a Cuckoo-LBG apresentou os maiores resultados para os tamanhos de $N = \{64, 128, 256\}$, ficando em segundo dentre as técnicas de projeto de dicionário avaliadas. Na imagem *Peppers* o algoritmo LBG, exceto para $N = 32$, alternou entre a primeira e segunda técnica com melhor resultado em termos de SSIM médio das imagens reconstruídas.

Tabela 5.5: Valores de SSIM do Conjunto de Imagens.

	N / Método	32	64	128	256	512
Barbara	LBG	0,6801	0,7326	0,7856	0,8349	0,8784
	Cuckoo-LBG	**0,6815**	0,7353	0,7877	**0,8355**	**0,8797**
	FA-LBG	**0,6819**	**0,7365**	**0,7891**	**0,8366**	**0,8799**
	PSO-LBG	0,6807	0,7355	0,7862	0,8332	0,8747
	FSS-LBG-NB	0,6809	**0,7360**	**0,7881**	0,8351	0,8788
	N / Método	32	64	128	256	512
Clock	LBG	0,8382	0,8678	0,8941	0,9173	0,9374
	Cuckoo-LBG	**0,8452**	**0,8753**	**0,8993**	**0,9214**	0,9384
	FA-LBG	0,8451	0,8736	0,8982	**0,9207**	**0,9406**
	PSO-LBG	0,8438	0,8733	0,8969	0,9182	0,9358
	FSS-LBG-NB	**0,8454**	**0,8745**	**0,8988**	0,9206	**0,9399**
	N / Método	32	64	128	256	512
Lena	LBG	0,7791	**0,8225**	**0,8584**	**0,8891**	**0,9161**
	Cuckoo-LBG	0,7799	**0,8213**	**0,8566**	**0,8870**	0,9135
	FA-LBG	**0,7802**	0,8211	0,8561	0,8850	0,9143
	PSO-LBG	0,7794	0,8189	0,8526	0,8804	0,9084
	FSS-LBG-NB	**0,7802**	0,8209	0,8549	0,8856	**0,9165**
	N / Método	32	64	128	256	512
Peppers	LBG	0,7699	**0,8169**	**0,8559**	**0,8882**	**0,9158**
	Cuckoo-LBG	**0,7704**	0,8154	0,8540	**0,8853**	0,9137
	FA-LBG	**0,7704**	**0,8169**	**0,8543**	0,8851	0,9146
	PSO-LBG	0,7686	0,8141	0,8503	0,8791	0,9085
	FSS-LBG-NB	**0,7711**	0,8148	0,8538	**0,8853**	**0,9166**

5.2 Estratégias de Inicializações

Diversas estratégias de inicialização foram utilizadas em cada imagem e os valores médios de PSNR (dB) são apresentados nesta seção. O algoritmo de treinamento foi o mesmo para cada conjunto de inicialização, Cuckoo-LBG. A Tabela 5.6 apresenta os resultados em dB para a imagem *Barbara*. As inicializações obtiveram desempenho próximos à inicialização aleatória em todos os tamanhos de dicionário projetados. As estratégias **MS**, **MSH** e **Todos** apresentaram ganhos sobre a inicialização aleatória maiores ou iguais a 0,12 dB no tamanho $N = 512$.

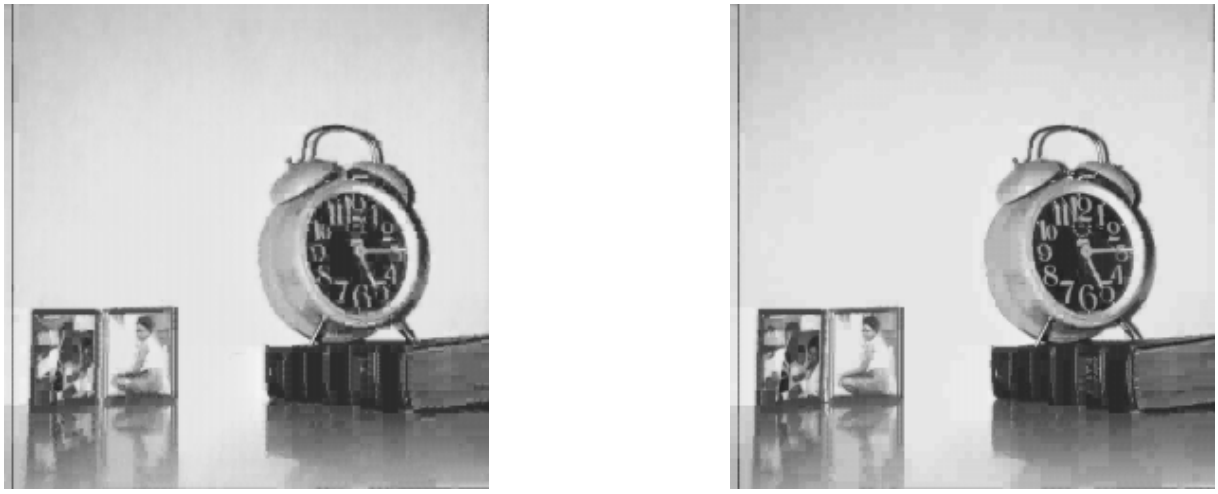
Para a imagem *Clock* os valores médios de PSNR estão presentes na Tabela 5.7. Dentre as estratégias de inicialização utilizadas, a de melhor desempenho, para tamanhos de dicionário menores ou igual a 256, foi a **DsKt**, exceto para $N = 128$. Para todas as estratégias observa-se que a superioridade sobre a inicialização **Aleatória**, em termos de PSNR, da imagem *Clock*, aumenta com N. Em particular, para $N = 512$, o ganho de PSNR obtido na substituição da inicialização **Aleatória** por **MSH**, é de 2,66 dB. A Figura 5.1 exibe duas imagens *Clock* reconstruídas utilizando as inicializações **Aleatória** e **MSH**, ambas com um dicionário de tamanho 512.

A Tabela 5.8 se refere aos resultados das inicializações para a imagem *Lena*. A inicialização **MS** apresenta praticamente o mesmo desempenho quando comparada à inicialização **Aleatória** para o tamanho de dicionário $N \leq 128$. Pode-se destacar que o maior ganho relativo à **Aleatória** ocorreu para a inicialização **MSH** com $N = 512$, com diferença de 0,55 dB.

A estratégia que utiliza sete inicializações aplicadas neste trabalho, ou seja, a estratégia denominada **Todos**, apresentou os melhores resultados em valores de PSNR, como demonstra a Tabela 5.9, quando observada a contribuição da estratégia nos diferentes tamanhos de dicionário.

As técnicas **MS** e **Todos** apresentam resultados de PSNR próximos entre si para a imagem *Peppers*. O maior ganho, quando comparado à inicialização **Aleatória**, está na **MSH** para o dicionário de tamanho 512, correspondendo a 0,51 dB.

A Tabela 5.10 contempla as diferenças absolutas, em termos de PSNR, dos conjuntos de inicializações em relação à inicialização aleatória para todas as imagens. Os valores positivos na tabela demonstram que a técnica em questão obteve resultado superior à técnica de inicialização aleatória. De acordo com a tabela, as estratégias de inicialização com as maiores contribuições são **MSH** e **Todos**.



(a) Aleatória.

(b) MSH.

Figura 5.1: Imagens Clock reconstruídas com $N = 512$ com estratégias de inicialização diferentes.

Tabela 5.6: Valores médios de PSNR utilizando o Cuckoo-LBG aplicado a diferentes estratégias de inicialização para a imagem Barbara.

N	Conjuntos de Inicializações									
	DsMe	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**	**Aleat.**
32	24,79	**24,80**	**24,80**	**24,80**	**24,79**	**24,79**	**24,79**	**24,79**	24,78	**24,80**
64	25,79	25,79	25,78	25,78	**25,80**	25,79	**25,80**	**25,81**	**25,80**	25,79
128	**26,84**	26,83	**26,85**	26,83	**26,85**	26,78	**26,84**	**26,85**	**26,84**	**26,84**
256	28,07	28,06	28,06	28,05	**28,10**	**28,09**	**28,10**	28,09	28,07	28,08
512	29,55	29,70	29,69	29,70	29,72	29,74	**29,76**	29,75	**29,78**	29,63

Tabela 5.7: Valores médios de PSNR utilizando o Cuckoo-LBG aplicado a diferentes estratégias de inicialização para a imagem Clock.

N	Conjuntos de Inicializações									
	DsMe	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**	**Aleat.**
32	26,75	**26,83**	**26,80**	26,78	26,76	26,76	26,75	**26,83**	**26,80**	26,72
64	27,99	**28,16**	28,14	**28,15**	27,98	28,07	28,07	28,11	28,14	27,97
128	29,31	29,58	**29,60**	**29,63**	29,44	29,48	**29,52**	**29,58**	29,51	29,25
256	30,84	**31,58**	31,47	**31,54**	31,29	31,15	31,18	**31,31**	**31,50**	30,80
512	32,61	33,70	34,66	34,09	**35,01**	33,48	34,00	**35,06**	34,26	32,40

Tabela 5.8: Valores médios de PSNR utilizando o Cuckoo-LBG aplicado a diferentes estratégias de inicialização para a imagem Lena

N	Conjuntos de Inicializações									
	DsMe	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**	**Aleat.**
32	**26,67**	26,65	26,67	**26,67**	26,65	**26,66**	**26,67**	26,61	**26,67**	**26,67**
64	27,87	27,88	27,87	27,86	27,85	27,89	**27,90**	27,86	**27,91**	27,88
128	29,12	29,14	29,13	29,14	29,10	29,16	**29,18**	29,08	**29,20**	29,14
256	30,52	30,69	30,60	30,65	30,51	**30,71**	**30,72**	30,54	30,70	30,52
512	32,28	32,56	32,53	32,60	32,63	**32,77**	32,71	**32,80**	32,70	32,25

Tabela 5.9: Valores médios de PSNR utilizando o Cuckoo-LBG aplicado a diferentes estratégias de inicialização para a imagem Peppers.

N	Conjuntos de Inicializações									
	DsMe	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**	**Aleat.**
32	**26,17**	**26,16**	26,15	**26,16**	**26,17**	**26,16**	**26,17**	**26,16**	**26,17**	26,15
64	27,45	**27,47**	**27,48**	27,47	**27,48**	27,46	27,43	**27,47**	27,46	27,45
128	28,80	28,79	28,81	28,79	28,79	28,85	**28,86**	28,79	**28,87**	28,78
256	30,30	30,42	30,43	30,39	30,33	30,45	**30,49**	30,34	**30,49**	30,27
512	32,16	32,54	32,48	32,42	32,51	32,64	32,63	**32,73**	**32,70**	32,22

Tabela 5.10: Diferenças do valores de PSNR das estratégias de inicialização em relação à inicialização aleatória

	N	**DsMe**	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**
Barbara	32	-0,01	**0,00**	**0,00**	**0,00**	-0,01	-0,01	-0,01	-0,01	-0,02
	64	**0,01**	0,00	-0,01	0,00	**0,02**	0,00	**0,01**	**0,02**	**0,01**
	128	**0,00**	-0,01	**0,01**	-0,01	**0,01**	-0,06	**0,00**	**0,01**	**0,01**
	256	-0,01	-0,02	-0,02	-0,03	**0,02**	**0,01**	**0,02**	**0,01**	-0,01
	512	-0,08	0,07	0,06	0,07	0,09	0,12	**0,13**	**0,13**	**0,15**
Clock	N	**DsMe**	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**
	32	0,03	**0,11**	**0,08**	0,06	0,04	0,04	0,03	**0,11**	**0,08**
	64	0,02	**0,19**	0,18	**0,19**	0,01	0,10	0,10	0,14	0,17
	128	0,06	0,33	**0,35**	**0,38**	0,19	0,23	0,27	0,32	0,26
	256	0,04	**0,78**	0,67	**0,74**	0,49	0,35	0,37	0,51	0,69
	512	0,21	1,30	2,26	1,69	**2,61**	1,08	1,59	**2,66**	1,86
Lena	N	**DsMe**	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**
	32	**0,00**	-0,01	**0,00**	**0,00**	-0,02	-0,01	**0,01**	-0,06	**0,00**
	64	-0,02	-0,01	-0,02	-0,02	-0,04	**0,00**	**0,02**	-0,03	**0,02**
	128	-0,02	0,01	-0,01	0,00	-0,04	0,02	**0,04**	-0,06	**0,06**
	256	0,00	0,16	0,08	0,12	-0,01	**0,19**	**0,20**	0,02	0,18
	512	0,03	0,31	0,28	0,35	0,38	**0,52**	0,46	**0,55**	0,45
Peppers	N	**DsMe**	**DsKt**	**MeKt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**Todos**
	32	**0,02**	**0,01**	0,00	**0,01**	**0,02**	**0,01**	**0,02**	**0,01**	**0,02**
	64	0,00	**0,02**	**0,03**	**0,02**	**0,03**	0,01	-0,02	**0,02**	0,01
	128	0,02	0,01	0,03	0,02	0,01	**0,08**	**0,08**	0,01	**0,09**
	256	0,03	0,14	0,15	0,12	0,05	0,18	**0,22**	0,07	**0,21**
	512	-0,05	0,32	0,26	0,20	0,30	0,42	0,42	**0,52**	**0,49**

5.3 Técnicas de Busca do VMP

Esta seção contém os resultados do tempo de execução médio, em segundos, para as técnicas de Busca Total (BT ou FSA, do inglês, *Full Search Algorithm*), Busca por Distorção Parcial (PDS, do inglês *Partical Distortion Search*) e IEENNS (do inglês, *Improve Equal-average Equal-variance Nearest Neighbour Search*).

Para a imagem *Barbara*, de acordo com a Tabela 5.11(a), para todos os tamanhos de N, a técnica IEENNS reduz em mais de dois terços o tempo de execução quando comparado a BT. Particularmente, em $N = 512$, a redução está acima de 91%.

Na Tabela 5.11(b), são apresentados os resultados do tempo de execução utilizando a técnica IEENNS na imagem *Clock*. Observa-se que esta técnica leva a uma economia de tempo superior a 74% para $N = 32$. A economia de tempo de execução aumenta com o tamanho do dicionário utilizado.

A técnica de busca eficiente para o vizinho mais próximo IEENNS apresentou uma redução, no tempo de execução, para a imagem *Lena*, quando comparada a BT, como exibido na Tabela 5.11(c). Para $N = 512$ a redução foi de 64,75 segundos em média.

Conforme mostra a Tabela 5.11(d), para a imagem *Peppers* a técnica de busca IEENNS apresentou os menores tempos de execução. Em relação a BT a maior economia da IEENNS se deu em $N = 512$, com uma redução de 87,4%.

A Figura 5.2 apresenta os gráficos, para cada imagem, do tempo em segundos obtidos aplicando as técnicas de busca do VMP: BT, PDS e IEENNS.

Tabela 5.11: Tempo de execução em segundos para as técnicas de busca do VMP.

	32	64	128	256	512
BT	3,040	7,003	14,767	32,238	122,852
PDS	1,989	3,555	6,605	13,006	27,892
IEENS	0,997	1,802	3,204	6,081	9,921

(a) Tempo em segundos para a Imagem *Barbara*.

	32	64	128	256	512
BT	4,683	10,276	24,375	50,750	89,532
PDS	2,380	4,454	8,529	15,495	25,044
IEENS	1,208	1,865	3,279	6,297	10,593

(b) Tempo em segundos para a Imagem *Clock*.

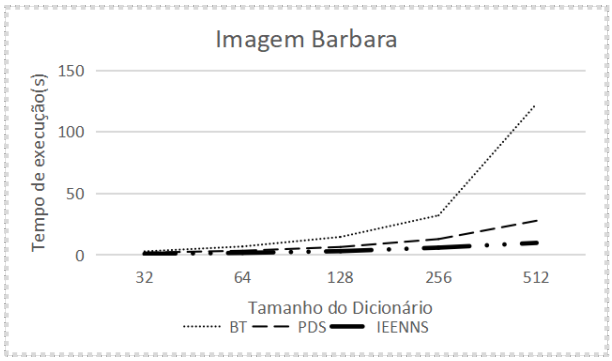
	32	64	128	256	512
BT	3,399	7,337	15,927	32,318	75,015
PDS	1,961	3,576	6,918	14,568	26,379
IEENS	0,991	1,786	3,284	5,557	10,265

(c) Tempo em segundos para a Imagem *Lena*.

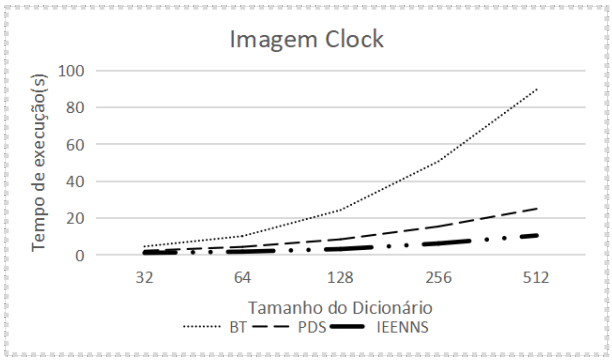
	32	64	128	256	512
BT	3,315	7,897	17,383	39,771	84,130
PDS	1,962	3,548	7,016	13,750	26,458
IEENS	0,940	1,558	2,769	5,044	10,597

(d) Tempo em segundos para a Imagem *Peppers*.

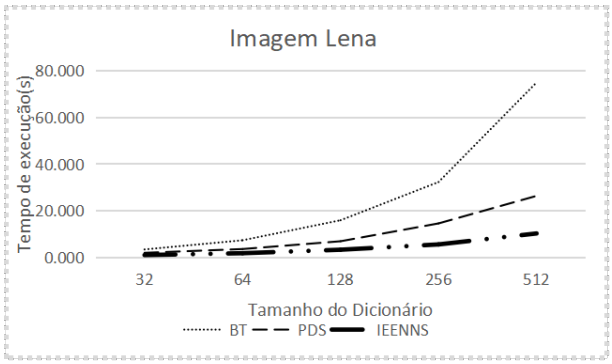
A Tabela 5.12 exibe os valores médios de SSIM para as imagens reconstruídas aplicando diferentes estratégias de inicialização. Em destaque estão os maiores valores de SSIM numa determinada linha, em outras palavras, qual a inicialização que alcançou o maior valor de SSIM para um dado tamanho N. A última coluna, com um "*" no cabeçalho, exibe a diferença entre o valor destacado em negrito e o valor calculado utilizando a inicialização aleatória. A inicialização **MH** apresenta os maiores valores em cinco cenários implementados e o maior aumento em relação à inicialização aleatória ocorreu para *Clock* em $N = 512$, no valor de 0,0047 também utilizando a estratégia **MH**.



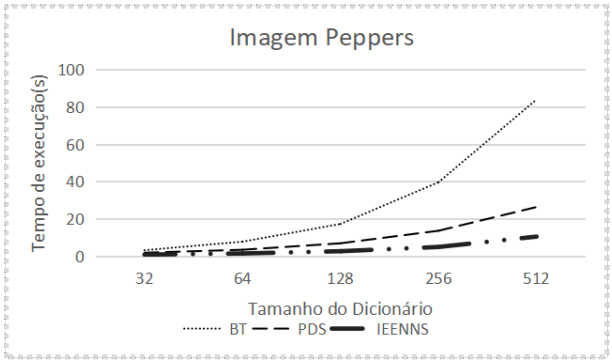
(a) Tempo em segundos das técnicas de busca do VMP para a imagem Barbara.



(b) Tempo em segundos das técnicas de busca do VMP para a imagem Clock.



(c) Tempo em segundos das técnicas de busca do VMP para a imagem Lena.



(d) Tempo em segundos das técnicas de busca do VMP para a imagem Peppers.

Figura 5.2: Desempenho das técnicas de busca do VMP.

Tabela 5.12: Valores de SSIM para as Imagens Reconstruídas Aplicando Diferentes Inicializações.

Imagem	***N***	**DsMe**	**DsKt**	**Mekt**	**DsMeKt**	**SH**	**MH**	**MS**	**MSH**	**TODOS**	**Aleatória**	*
Barbara	32	**0,6821**	0,6820	0,6809	0,6817	0,6804	0,6802	0,6806	0,6813	0,6802	0,6815	0,0006
	64	0,7360	**0,7369**	0,7348	0,7341	0,7312	0,7288	0,7324	0,7324	0,7323	0,7353	0,0016
	128	0,7873	**0,7881**	0,7874	0,7854	0,7867	0,7765	0,7869	0,7863	0,7863	0,7877	0,0004
	256	**0,8360**	0,8274	0,8282	0,8320	0,8273	0,8267	0,8275	0,8280	0,8284	0,8355	0,0005
	512	0,8796	0,8689	0,8746	0,8705	0,8596	0,8690	0,8583	0,8597	0,8599	**0,8797**	-0,0001
Clock	32	0,8456	0,8421	0,8434	0,8431	0,8458	**0,8462**	0,8450	0,8420	0,8446	0,8452	0,0010
	64	0,8754	0,8681	0,8720	0,8691	**0,8762**	0,8751	0,8759	0,8692	0,8736	0,8753	0,0009
	128	0,9005	0,8950	0,8942	0,8951	0,8995	**0,9032**	0,9030	0,8982	0,8998	0,8993	0,0039
	256	0,9212	0,9160	0,9158	0,9151	0,9238	**0,9243**	0,9217	0,9187	0,9186	0,9214	0,0029
	512	0,9396	0,9392	0,9381	0,9378	0,9376	**0,9431**	0,9395	0,9354	0,9397	0,9384	0,0047
Lena	32	0,7795	0,7790	**0,7803**	0,7801	0,7790	0,7752	0,7757	0,7745	0,7770	0,7799	0,0004
	64	0,8212	0,8208	0,8204	0,8209	**0,8237**	0,8182	0,8191	0,8229	0,8185	0,8213	0,0024
	128	0,8551	0,8541	0,8520	0,8490	0,8556	0,8538	0,8544	0,8536	0,8536	**0,8566**	0,0000
	256	0,8857	0,8815	0,8734	0,8782	0,8851	0,8845	0,8844	0,8804	0,8851	**0,8870**	0,0000
	512	0,9132	0,9046	0,9063	0,9075	0,9011	**0,9147**	0,9141	0,9014	0,9137	0,9135	0,0012
Peppers	32	0,7697	**0,7707**	0,7699	0,7706	0,7694	0,7681	0,7676	0,7636	0,7685	0,7704	0,0003
	64	0,8139	0,8151	0,8154	0,8142	0,8154	0,8101	0,8103	**0,8171**	0,8109	0,8154	0,0017
	128	0,8543	0,8520	0,8500	0,8513	**0,8547**	0,8513	0,8511	0,8490	0,8510	0,8540	0,0007
	256	**0,8861**	0,8716	0,8747	0,8761	0,8849	0,8852	0,8852	0,8738	0,8846	0,8853	0,0008
	512	0,9141	0,9034	0,9084	0,9051	0,9074	0,9140	**0,9144**	0,9138	0,9098	0,9137	0,0007

6
Conclusões

Nesta dissertação foi apresentado um novo método híbrido para o projeto de dicionários para a compressão de sinais baseada em quantização vetorial. Precisamente, o método consiste em uma combinação da técnica de enxame *Cuckoo Search* e do algoritmo LBG (Linde-Buzo-Gray). A técnica proposta é denominada Cuckoo-LBG e foi aplicada no contexto da compreensão de imagens baseada em quantização vetorial. A nova técnica se mostrou superior ao algoritmo LBG para todas as imagens avaliadas em termos de PSNR (*Peak Signal-to-Noise Ratio*) da imagens reconstruídas.

Uma outra novidade deste trabalho foi a proposta e avaliação de diferentes estratégias de inicialização de algoritmos de enxames combinados com o LBG. Cada estratégia de inicialização das partículas de enxame consiste em combinar técnicas da literatura com inicializações aleatórias. Por exemplo, na estratégia MSH um conjunto de dez dicionários iniciais é formado por sete dicionários aleatórios, um utilizando a técnica aqui designada por Ma (o primeiro dos quatro autores do método), um utilizando a técnica *Subtractive Clustering* e o último correspondente a uma inicialização que faz uso da transformada de Hadamard. Aplicando a estratégia MSH, a imagem reconstruída *Clock*, apresentou em média, para dicionário com tamanho $N = 512$ e dimensão $K = 16$ (correspondente a blocos de 4×4 *pixels*) projetado com o Cuckoo-LBG, um ganho de 2,66 dB em comparação à inicialização aleatória.

Nesta dissertação são apresentadas nove estratégias de inicialização, as quais são comparadas com a inicialização aleatória. Utilizadas no algoritmo Cuckoo-LBG, várias das estratégias supracitadas permitiram obter dicionários com qualidade superior aos obtidos com uso de inicialização aleatória, contribuindo, assim, para a obtenção de imagens reconstruídas com melhor qualidade em termos de PSNR.

No trabalho apresentou-se, ainda, uma alternativa de aceleração do algoritmo proposto, a qual consiste na acomodação do algoritmo de busca eficiente do VMP (Vizinho mais Próximo) IEENNS (*Improved Equal-average Equal-variance Nearest Neighbor Search*) na etapa de particionamento do algoritmo LBG. A economia de tempo de execução do Cuckoo-LBG obtida com a substituição da BT (Busca Total) pelo IEENNS aumenta como tamanho do dicionário.

6.1 Sugestões para Trabalhos Futuros

Como trabalhos futuros podem sem citados:

- Avaliação do algoritmo Cuckoo-LBG em outras aplicações, tais como segmentação de imagens e compressão de nuvens de pontos 3D.
- Apresentação de novas técnicas de inicialização de algoritmos de enxames aplicados à quantização vetorial.
- Apresentação de novas técnicas de enxames baseadas no algoritmo CS (*Cuckoo Search*) para o projeto de dicionários, como por exemplo, combinação do CS com o algoritmo *Fuzzy k-means* ou com versões modificadas do algoritmo LBG.

Referências Bibliográficas

[1] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[2] A. Gersho e R. M. Gray, *Vector quantization and signal compression*. Springer Science & Business Media, 2012, vol. 159.

[3] F. Harchli, Z. En-Naimani, A. Es-Safi, e M. Ettaouil, "Vector quantization for speech compression by a new version of PRSOM," *International Journal on Artificial Intelligence Tools*, vol. 27, nº. 03, p. 1850013, 2018.

[4] P. N. T. Ammah e E. Owusu, "Robust medical image compression based on wavelet transform and vector quantization," *Informatics in Medicine Unlocked*, vol. 15, p. 100183, 2019.

[5] S. Chandra, A. Sharma, e G. Singh, "A comparative analysis of performance of several wavelet based ECG data compression methodologies," *IRBM*, 2020.

[6] A. Tiwari e M. Sharma, "An efficient vector quantization based watermarking method for image integrity authentication," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Springer, 2018, pp. 215–225.

[7] M. Naveenkumar e S. Domnic, "Vector quantization based pairwise joint distance maps (VQ-PJDM) for 3d action recognition," *Procedia Computer Science*, vol. 133, pp. 27–36, 2018.

[8] Z. Cheng, F. Yang, B. Gao, P. Yu, Q. Yang, J. Tian, e X. Lu, "Partial discharge pattern recognition of XLPE cable based on vector quantization," *IEEE Transactions on Magnetics*, vol. 55, nº. 6, pp. 1–4, 2019.

[9] Y. Linde, A. Buzo, e R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, nº. 1, pp. 84–95, 1980.

[10] K.-C. Hu, C.-W. Tsai, e M.-C. Chiang, "A highly efficient method for improving the performance of GLA-based algorithms," *Journal of Visual Communication and Image Representation*, vol. 50, pp. 290–302, 2018.

[11] V. K. Dehariya, S. K. Shrivastava, e R. Jain, "Clustering of image data set using k-means and fuzzy k-means algorithms," in *2010 International Conference on Computational Intelligence and Communication Networks*. IEEE, 2010, pp. 386–391.

[12] M. A. Hossan e M. A. Gregory, "Speaker recognition utilizing distributed DCT-II based mel frequency cepstral coefficients and fuzzy vector quantization," *International Journal of Speech Technology*, vol. 16, nº. 1, pp. 103–113, 2013.

[13] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Computer Engineering Department, Tech. Rep., 2005.

[14] M. Dorigo, M. Birattari, e T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, nº. 4, pp. 28–39, 2006.

[15] R. Eberhart e J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[16] M.-H. Horng, "Vector quantization using the firefly algorithm for image compression," *Expert Systems with Applications*, vol. 39, nº. 1, pp. 1078–1091, 2012.

[17] M.-H. Horng e T.-W. Jiang, "Image vector quantization algorithm via honey bee mating optimization," *Expert Systems with Applications*, vol. 38, nº. 3, pp. 1382–1392, 2011.

[18] H. A. Atabay, M. J. Sheikhzadeh, e M. Torshizi, "A clustering algorithm based on integration of K-means and PSO," in *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*. IEEE, 2016, pp. 59–63.

[19] D. Karaboga e C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing*, vol. 11, nº. 1, pp. 652–657, 2011.

[20] H. Emami e F. Derakhshan, "Integrating fuzzy k-means, particle swarm optimization, and imperialist competitive algorithm for data clustering," *Arabian Journal for Science and Engineering*, vol. 40, nº. 12, pp. 3545–3554, 2015.

[21] V. Severo, H. Leitão, J. Lima, W. Lopes, e F. Madeiro, "Modified firefly algorithm applied to image vector quantisation codebook design," *International Journal of Innovative Computing and Applications*, vol. 7, nº. 4, pp. 202–213, 2016.

[22] C. Fonseca, F. A. Ferreira, e F. Madeiro, "Vector quantization codebook design based on fish school search algorithm," *Applied Soft Computing*, vol. 73, pp. 958–968, 2018.

[23] C. Ping-Yi, J.-T. Tsai, J.-H. Chou, W.-H. Ho, H.-Y. Shi, e S.-H. Chen, "Improved PSO-LBG to design VQ codebook," in *The SICE Annual Conference 2013*. IEEE, 2013, pp. 876–879.

[24] T. Dong, J. Wang, M. Yang, K. Yi, e N. Zheng, "Affine LBG for codebook training of univariate linear representation," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 46–50.

[25] S. D. Thepade, R. H. Garg, S. A. Ghewade, P. A. Jagdale, e N. M. Mahajan, "Performance assessment of assorted similarity measures in gray image colorization using LBG vector quantization algorithm," in *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, 2015, pp. 332–337.

[26] D. P. Tripathi e U. R. Jena, "Vector codebook design using gravitational search algorithm," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016, pp. 553–558.

[27] G. Patanè e M. Russo, "The enhanced LBG algorithm," *Neural networks*, vol. 14, nº. 9, pp. 1219–1237, 2001.

[28] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, nº. 14. Oakland, CA, USA, 1967, pp. 281–297.

[29] D. Steinley, "Local optima in k-means clustering: what you don't know may hurt you." *Psychological methods*, vol. 8, nº. 3, p. 294, 2003.

[30] I. Katsavounidis, C.-C. J. Kuo, e Z. Zhang, "A new initialization technique for generalized lloyd iteration," *IEEE Signal Processing Letters*, vol. 1, nº. 10, pp. 144–146, 1994.

[31] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[32] M. M.-T. Chiang e B. Mirkin, "Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads," *Journal of Classification*, vol. 27, nº. 1, pp. 3–40, 2010.

[33] F. Cao, J. Liang, e L. Bai, "A new initialization method for categorical data clustering," *Expert Systems with Applications*, vol. 36, nº. 7, pp. 10 223–10 228, 2009.

[34] K.-C. Hu, C.-H. Chen, C.-W. Tsai, e M.-C. Chiang, "An enhanced initialization method for codebook generation," in *2015 IEEE International Conference on Consumer Electronics-Taiwan*. IEEE, 2015, pp. 92–93.

[35] S. Chen e F. Li, "Initial codebook method of vector quantisation in Hadamard domain," *Electronics Letters*, vol. 46, nº. 9, pp. 630–631, 2010.

[36] B. Mirzaei, H. Nezamabadi-pour, e D. Abbasi-moghadam, "An effective codebook initialization technique for LBG algorithm using subtractive clustering," in *2014 Iranian Conference on Intelligent Systems (ICIS)*. IEEE, 2014, pp. 1–5.

[37] D. Steinley e M. J. Brusco, "Initializing k-means batch clustering: A critical evaluation of several techniques," *Journal of Classification*, vol. 24, nº. 1, pp. 99–121, 2007.

[38] P. Fränti e S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, nº. 12, pp. 4743–4759, 2018.

[39] M. Minu e R. A. Canessane, "An efficient squirrel search algorithm based vector quantization for image compression in unmanned aerial vehicles," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. IEEE, 2021, pp. 789–793.

[40] K. Chiranjeevi e U. R. Jena, "Image compression based on vector quantization using cuckoo search optimization technique," *Ain Shams Engineering Journal*, vol. 9, nº. 4, pp. 1417–1431, 2018.

[41] C.-D. Bei e R. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Transactions on Communications*, vol. 33, nº. 10, pp. 1132–1133, 1985.

[42] S. Baek, B. Jeon, e K.-M. Sung, "A fast encoding algorithm for vector quantization," *IEEE Signal Processing Letters*, vol. 4, nº. 12, pp. 325–327, 1997.

[43] A. J. Hussain, A. Al-Fayadh, e N. Radi, "Image compression techniques: A survey in lossless and lossy algorithms," *Neurocomputing*, vol. 300, pp. 44–69, 2018.

[44] X. Ma, Z. Pan, Y. Li, e J. Fang, "High-quality initial codebook design method of vector quantisation using grouping strategy," *IET Image Processing*, vol. 9, nº. 11, pp. 986–992, 2015.

[45] A. Nyeck e A. Tosser-Roussey, "Maximum entropy initialisation technique for image coding vector quantiser design," *Electronics Letters*, vol. 28, nº. 3, pp. 273–274, 1992.

[46] S.-W. Ra e J.-K. Kim, "A fast mean-distance-ordered partial codebook search algorithm for image vector quantization," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, nº. 9, pp. 576–579, 1993.

[47] A. Iglesias, A. Gálvez, e P. Suárez, "Swarm robotics–a case study: bat robotics," in *Nature-Inspired Computation and Swarm Intelligence*. Elsevier, 2020, pp. 273–302.

[48] C. J. Bastos Filho, F. B. de Lima Neto, A. J. Lins, A. I. Nascimento, e M. P. Lima, "Fish school search," in *Nature-inspired algorithms for optimisation*. Springer, 2009, pp. 261–277.

[49] X.-S. Yang *et al.*, "Firefly algorithm," *Nature-Inspired Metaheuristic Algorithms*, vol. 20, nº. 2008, pp. 79–90, 2008.

[50] J. C. Bansal, H. Sharma, S. S. Jadon, e M. Clerc, "Spider monkey optimization algorithm for numerical optimization," *Memetic Computing*, vol. 6, nº. 1, pp. 31–47, 2014.

[51] Y. Tan, *GPU-based parallel implementation of swarm intelligence algorithms*. Morgan Kaufmann, 2016.

[52] N. Nedjah, P. J. A. de Oliveira *et al.*, "Simultaneous localization and mapping using swarm intelligence based methods," *Expert Systems with Applications*, vol. 159, p. 113547, 2020.

[53] T.-H. Nguyen e J. J. Jung, "Swarm intelligence-based green optimization framework for sustainable transportation," *Sustainable Cities and Society*, p. 102947, 2021.

[54] O. Shindi, J. Kanesan, G. Kendall, e A. Ramanathan, "The combined effect of optimal control and swarm intelligence on optimization of cancer chemotherapy," *Computer Methods and Programs in Biomedicine*, vol. 189, p. 105327, 2020.

[55] J. H. Joloudari, H. Saadatfar, A. Dehzangi, e S. Shamshirband, "Computer-aided decision-making for predicting liver disease using PSO-based optimized SVM with feature selection," *Informatics in Medicine Unlocked*, vol. 17, p. 100255, 2019.

[56] W. Zhou, M. Chen, Z. Yang, e X. Song, "Real estate risk measurement and early warning based on pso-svm," *Socio-Economic Planning Sciences*, p. 101001, 2020.

[57] X.-S. Yang e S. Deb, "Cuckoo search via Lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. Ieee, 2009, pp. 210–214.

[58] R. B. Payne e M. D. Sorensen, *The cuckoos*. Oxford University Press, 2005, vol. 15.

[59] P. Barthelemy, J. Bertolotti, e D. S. Wiersma, "A Lévy flight for light," *Nature*, vol. 453, nº. 7194, pp. 495–498, 2008.

[60] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of levy stable stochastic processes," *Physical Review E*, vol. 49, nº. 5, p. 4677, 1994.

[61] Q. Qi, Y. Tian, e L. Han, "An improved image segmentation algorithm based on the maximum class variance method," in *MATEC Web of Conferences*, vol. 309. EDP Sciences, 2020, p. 03029.

[62] A. M. Reynolds e M. A. Frye, "Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search," *PloS One*, vol. 2, nº. 4, p. e354, 2007.

[63] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, nº. 2, pp. 1830–1844, 2007.

[64] ——, "Cooling down Lévy flights," *Journal of Physics A: Mathematical and Theoretical*, vol. 40, nº. 41, p. 12299, 2007.

[65] Y. Ling, Y. Zhou, e Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," *IEEE Access*, vol. 5, pp. 6168–6186, 2017.

[66] M. F. Shlesinger, "Search research," *Nature*, vol. 443, nº. 7109, pp. 281–282, 2006.

[67] X.-S. Yang e S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications*, vol. 24, nº. 1, pp. 169–174, 2014.

[68] Z. Wang, A. C. Bovik, H. R. Sheikh, e E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, nº. 4, pp. 600–612, 2004.