Algoritmo FSS Modificado Aplicado à Quantização Vetorial Robusta de Imagens

Wesley Vieira de Santana
Universidade de Pernambuco
Recife, Brasil
wvs1@poli.br

Waslon Terllizzie Araújo Lopes
Universidade Federal da Paraíba
João Pessoa, Brasil
waslon@ieee.org

Francisco Madeiro
Universidade de Pernambuco
Recife, Brasil
madeiro@poli.br

***Resumo*—No presente trabalho, é apresentada uma versão modificada do algoritmo de Busca por Cardume (FSS – *Fish School Search*) aplicado ao problema de Quantização Vetorial Robusta (QVR). No processo de otimização da QVR, a figura de mérito utilizada é o índice de desordem considerando distâncias de Hamming iguais a 1 e 2. Os resultados de simulação mostram que o algoritmo FSS apresentou melhor desempenho quando comparado ao *Simulated Annealing* (SA) para várias condições de transmissão de imagens por Canal Binário Simétrico.**

***Palavras-chaves*—Transmissão de imagens, quantização vetorial robusta, inteligência de enxame, otimização combinatorial, FSS, distância de *Hamming*.**

I. Introdução

O avanço das comunicações digitais tem motivado pesquisas em busca de esquemas de codificação eficientes para diminuir a ocupação da largura de banda na transmissão ou minimizar a utilização dos recursos de armazenamento de um sistema. A compressão de sinais (voz, imagem, áudio e vídeo) visa diminuir a quantidade de *bits* na sua representação digital.

A Quantização Vetorial (QV) é uma relevante técnica de compressão de sinais utilizada em muitos sistemas de comunicação digital [1]–[4]. A QV fundamenta-se na Teoria da Distorção Versus Taxa, formulada por Shanon, na qual é possível encontrar um melhor desempenho de compressão realizando a codificação de blocos de amostras, ou seja, vetores, em vez de escalares [5].

Contudo, a QV tem sensibilidade ao ruído introduzido pelo canal de comunicação, levando a erros na decodificação. Os erros do canal podem comprometer o desempenho do sistema de QV. Formas para atenuar este tipo de problema têm sido alvo de pesquisas, e.g. [6], [7].

Os impactos provocados por canais ruidosos no desempenho de um sistema de QV podem ser amenizados utilizando a Quantização Vetorial Otimizada para Canal (QVOC) [7], [8] ou a Quantização Vetorial Robusta (QVR) [6], [9], [10]. Na QVOC, o projeto do dicionário de vetores-códigos é realizado levando em consideração a distorção de um canal específico. Na QVR, o projeto do dicionário de vetores-códigos é realizado considerando um canal sem ruído, e, após a conclusão do projeto do dicionário, é realizada uma atribuição de índices (AI) aos vetores-códigos do dicionário através de um algoritmo de otimização. Este processo de organização dos índices minimiza os impactos do ruído do canal, tornando o dicionário mais robusto aos erros [9].

Pesquisas têm explorado algoritmos que realizem a tarefa de organização dos índices do dicionário com o objetivo de torná-lo robusto. Farvadin [11] em 1990 propôs a utilização do *Simulated Annealing* (SA) para a organização dos índices de um dicionário. Melhorias foram propostas para o SA, e.g. [12], [13]. Zeger e Gersho [14] apresentaram o *Binary Switching Algorithm* (BSA). O algoritmo *Variable Neighborhood Search* (VNS) também foi utilizado como alternativa para atacar este problema, obtendo resultados satisfatórios comparados ao SA [15].

Algoritmos de inteligência computacional (IC), particularmente da área de computação bioinspirada, podem ser utilizados para resolver o problema de AI, como o algoritmo Colônia Artificial de Abelhas (ABC – *Artificial Bee Colony*) [6] e algoritmos evolucionários [16], [17].

Este trabalho tem como foco a QVR para transmissão de imagens. É apresentada uma modificação do algoritmo Busca por Cardume (FSS – *Fish School Search*) [18] para a realização da organização dos índices do dicionário. Neste estudo, foram consideradas distâncias de Hamming iguais a 1 e 2 como critérios para otimização da função objetivo, ou seja, o índice de desordem do dicionário. Para avaliar as imagens reconstruídas foram consideradas as medidas objetivas, Relação Sinal-Ruído de Pico (PSNR – *Peak Signal-to-Noise Ratio*) e Similaridade Estrutural (SSIM – *Structural Similarity*) [19].

Este trabalho está organizado da seguinte forma: a Seção II apresenta a problemática e fundamentos da QVR, a Seção III faz uma breve abordagem do algoritmo FSS, a Seção IV apresenta as modificações feitas no FSS para adaptá-lo ao problema de otimização combinatorial da QVR, a Seção V apresenta os resultados e a Seção VI contém as conclusões.

II. Quantização Vetorial Robusta

Um Quantizador Vetorial pode ser definido como um mapeamento de um vetor K-dimensional pertencente a um espaço euclidiano $\mathbb{R}^K$ para um subconjunto finito dado por $Q : \mathbb{R}^K \rightarrow Y$, em que $Y = \{y_0, y_1, y_2, ..., y_{N-1}\}$, sendo $y_i \in \mathbb{R}^K$, para $0 \leq i \leq N-1$. O conjunto Y é chamado de dicionário e os N elementos são os vetores-códigos. O tamanho do dicionário é $N = 2^b$, sendo b um inteiro positivo.

Cada um dos N índices que representam os vetores-códigos do dicionário podem ser apresentados na forma binária por $\{0,1\}^b$, em que b é o comprimento da palavra-código.

A QVR visa diminuir, por meio do processo de AI, a sensibilidade dos erros provocados por um canal ruidoso sem a necessidade de adição de *bits* na codificação. Os erros do canal podem modificar o índice enviado pelo codificador, forçando o decodificador escolher de forma errada um vetor-código diferente do codificado, provocando, no caso de transmissão de imagens, em geral, bloqueamentos espúrios [13], [20].

Os efeitos provocados pelo ruído do canal podem resultar em uma distorção relevante do sinal reconstruído no receptor. Uma atribuição adequada dos índices aos vetores que compõem o dicionário pode fazer com que os vetores decodificados estejam próximos, em média, dos vetores originais. Desta forma, a distorção média introduzida pelos erros do canal pode ser minimizada [21].

A degradação do sinal é atenuada com a atribuição de índices com grande/pequena distância de Hamming a vetores-código com grande/pequena distância euclidiana. A distorção total do sistema de QV é a soma dos erros de quantização e dos erros nos índices transmitidos. É representada matematicamente como

$$U_\pi = E[d(x, y_i)] + E[d(y_i, y_j)], \tag{1}$$

em que U_π é a distorção total, $E[d(x, y_i)]$ representa o operador valor esperado da distorção da quantização vetorial e $E[d(y_i, y_j)$ é o valor esperado da distorção da transmissão dos índices [21].

De forma a otimizar o desempenho do sistema de codificação para um dado dicionário, o valor da distorção provocada pelos erros nos índices deve ser minimizado considerando as possíveis permutações. Na equação (1) observa-se que $E[d(x, y_i)$ é a distorção esperada dado um vetor de entrada por um vetor-código do dicionário. Este valor é independente das atribuições dos índices aos vetores-códigos, ou seja, representa a distorção proveniente do projeto do dicionário. Para fins de minimização da distorção devido as permutações dos índices, é suficiente minimizar

$$D_\pi = \min_{\pi \in S_N} E[d(y_i, y_j)]. \tag{2}$$

O valor de $E[d(y_i, y_j)]$ representa a distorção esperada dado o vetor-código transmitido em relação ao vetor-código recebido, assumindo que $d(y_i, y_j)$ é a função de distorção quadrática entre eles.

Considerando cada índice binário $q \in \{0,1\}^b$ representando um vetor-código, m sendo um inteiro, $1 \leq m \leq b$, pode-se obter um conjunto de índices que fazem vizinhança com q obtendo os índices que possuem distância Hamming m em relação a q. Para uma palavra binária q de comprimento b o número de índices com distância Hamming m é dado por

$$\begin{pmatrix} b \\ m \end{pmatrix}. \tag{3}$$

Supondo $b = 3$, os conjuntos para o índice "1" em decimal seriam: $V^1(1) = \{0, 3, 5\}$, $V^2(1) = \{2, 4, 7\}$ e $V^3(1) = \{6\}$.

Considerando um Canal Binário Simétrico (BSC – *Binary Symmetric Channel*) com probabilidade de erro p e utilizando a distribuição de Bernouli. A probabilidade de uma palavra-código de comprimento b seja transmitida e recebida no decodificador por uma outra palavra-código particular de distância Hamming m, é dada por

$$q_m = \begin{pmatrix} b \\ m \end{pmatrix} p^m (1-p)^{b-m}. \tag{4}$$

Desta forma, a distorção para um vetor-código em particular y_k, em relação à permutação π, é dada por

$$C_{\pi(y_k)} = \sum_{m=1}^{b} q_m d(y_k, y_{\pi^{-1}(V^m(\pi(k)))}). \tag{5}$$

A função custo, que mede a contribuição total da distorção provocada pelos erros de canal, a qual deve ser minimizada, é dada por

$$D_\pi = \sum_{k=0}^{N-1} C_{\pi(y_k)}. \tag{6}$$

Esta função é chamada de índice de desordem do dicionário [21].

As Figuras 1 e 2 apresentam uma transmissão do índice 1, em sua representação binária, de um dicionário projetado e do dicionário correspondente organizado por meio da técnica de atribuição de índices, respectivamente. No cenário apresentado, considera-se que o canal poderá provocar até um erro de *bit* por símbolo, ou seja, o dicionário organizado da Figura 2 levou em consideração apenas distância Hamming 1 no processo de AI. Nessas ilustrações, observa-se que o dicionário organizado apresenta uma distorção menor que o original para a transmissão particular do índice 1 dos dicionários, o que representará uma minimização da distorção gerada na imagem reconstruída.

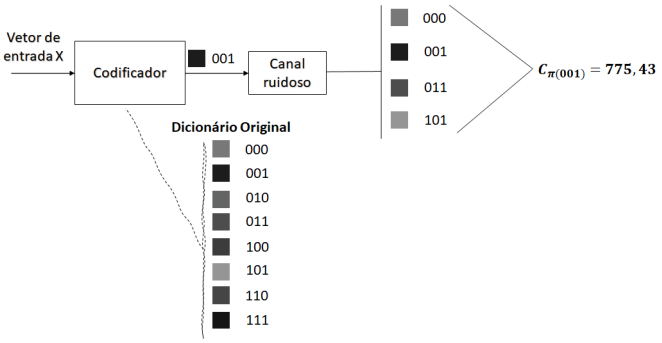

Figura 1: Transmissão do índice 1 de um vetor-código de dicionário original de tamanho 8 por um canal ruidoso.

III. Algoritmo FSS

O FSS é um algoritmo de inteligência de enxame inspirado no comportamento alimentar de cardumes, apresentado em 2008 por Bastos-Filho *et al.* [18]. O FSS apresenta bons resultados para problemas de otimização multimodal, devido

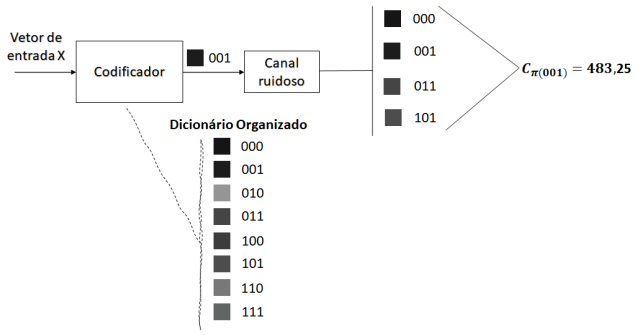

Figura 2: Transmissão do índice 1 de um vetor-código de dicionário organizado de tamanho 8 por um canal ruidoso.

ao mecanismo de expansão ou contração do cardume o que evita os mínimos locais [22]. Os movimentos realizados pelo cardume são: deslocamento individual, movimento coletivo instintivo e movimento coletivo volitivo.

A. Deslocamento Individual

Cada peixe do cardume executa uma busca local por melhores posições em sua vizinhaça. Sua posição é atualizada se após o movimento individual o seu *fitness* melhorar. O operador de deslocamento individual é apresentado matematicamente como

$$x_p^{t+1} = x_p^t + (step_{ind} \cdot rand), \tag{7}$$

em que x_p^{t+1} é a posição futura do peixe após sofrer o deslocamento individual, x_p^t é a posição atual do peixe, $step_{ind}$ é o valor de passo individual do peixe, sendo este valor multiplicado por um valor aleatório gerado a partir de uma distribuição uniforme entre $[-1, 1]$.

Após o deslocamento individual do cardume, realiza-se a operação de alimentação, o que definirá o peso de cada peixe. Este peso representa o sucesso do peixe em sua procura por melhores soluções. Quanto mais pesado for o peixe melhor é a qualidade da das soluções encontradas. No início das interações do algoritmo, atribui-se a cada peixe um peso o qual será atualizado no decorrer das interações. A equação que define o peso do peixe é dada por

$$W_p^{t+1} = W_p^t + \frac{\Delta f_p}{\max(|\Delta f|)}, \tag{8}$$

sendo W_p^{t+1} o peso do peixe depois da alimentação, W_p^t é o peso atual do peixe, Δf_p representa a variação do *fitness* do peixe na interação atual em relação a anterior e $\max(|\Delta f|)$ é o valor da maior variação de *fitness* do cardume.

B. Movimento Coletivo Instintivo

Este movimento visa mover o cardume em direção dos peixes que obtiveram melhores resultados após o deslocamento individual.

$$x_p^{t+1} = x_p^t + \frac{\sum_{p=1}^{N}(\Delta x_p^{ind} \cdot \Delta f_p)}{\sum_{p=1}^{N} \Delta f_p}, \tag{9}$$

em que x_p^{t+1} é a posição do peixe depois do movimento coletivo, x_p^t a posição atual do peixe, N é o total de peixes no cardume, Δx_p^{ind} é o deslocamento executado pelo peixe, Δf_p é a variação do *fitness* do peixe da interação atual em relação à anterior.

C. Movimento Coletivo Volitivo

Este é o último movimento executado pelo cardume. A idéia desse movimento é aproximar ou afastar os peixes do baricentro do cardume. O baricentro do cardume é dado por

$$Bari_{(t)} = \frac{\sum_{p=1}^{N}(x_p^t \cdot W_p^t)}{\sum_{p=1}^{N} W_p^t}, \tag{10}$$

sendo N o total de peixes no cardume, x_p^t a posição atual do peixe e W_p^t o peso do peixe.

De forma geral, se o peso do cardume aumenta, então os peixes movem-se em direção do baricentro, ou seja, contração do cardume:

$$x_p^{t+1} = x_p^t - step_{vol} \frac{x_p^t - Bari_{(t)}}{dist(x_p^t, Bari_{(t)})}. \tag{11}$$

Do contrário, os peixes se afastam do baricentro, ou seja, expansão do cardume:

$$x_p^{t+1} = x_p^t + step_{vol} \frac{x_p^t - Bari_{(t)}}{dist(x_p^t, Bari_{(t)})}, \tag{12}$$

em x_p^{t+1} é a posição do peixe depois da execução do movimento coletivo volitivo, x_p^t é a posição atual do peixe, $step_{vol}$ é um parâmetro que controla a granularidade da pesquisa que segue a mesma função do $step_{ind}$ e a função $dist(\cdot, \cdot)$ retorna a distância euclidiana entre os dois elementos. A contração do cardume contribui para a convergência do algoritmo, enquanto a expansão ajuda a prevenir mínimos locais.

O Algoritmo 1 ilustra a sequência de passos do FSS.

IV. Algoritmo FSS aplicado a QVR

O FSS tem seu foco em problemas de otimização contínua. Considerando que a QVR trata-se de uma otimização combinatorial, faz-se necessário realizar modificações no FSS para adequá-lo ao problema. Este trabalho apresenta mudanças no algoritmo que foram inspiradas nas versões binárias *Improved Binary Fish School Search* (IBFSS) proposto por Carneiro e Bastos-Filho [23] e na versão *Simplified Binary Fish School Search* (SBFSS) proposta por Santana Jr. *et al.* [24].

O primeiro passo é a inicialização do cardume, sendo que cada peixe x_p deve começar com uma combinação de N índices distintos, combinados a partir do conjunto $x = \{1, 2, 3, ..., N\}$, sendo N o valor do tamanho do dicionário. Para o problema apresentado neste trabalho, todas as permutações dos índices são viáveis. Desta forma, a inicialização de um peixe pode ser descrita como

$$x_p^{t_0} = rand_{perm}(x), \tag{13}$$

em que $rand_{perm}()$ é uma função que retorna uma permutação aleatória do conjunto de índices possíveis.

Algoritmo 1 Pseudocódigo FSS

```
Gere as posições aleatórias iniciais dos peixes dentro do espaço de pesquisa
Determine o peso inicial dos peixes
Use a função fitness para avaliar o fitness inicial dos peixes
enquanto critério de parada não é alcançado faça
    para cada peixe do cardume faça
        Execute o deslocamento individual. Equação (7)
    fim para
    Use a função fitness para avaliar o fitness dos peixes
    Atualize o melhor peixe
    para cada peixe do cardume faça
        Execute o operador de alimentação. Equação (8)
    fim para
    para cada peixe do cardume faça
        Execute o movimento coletivo instintivo. Equação (9)
    fim para
    para cada peixe do cardume faça

        Execute o movimento coletivo volitivo. Equação (10)
    fim para
    Use a função fitness para avaliar o fitness dos peixes
    Atualize o melhor peixe
    Atualize os step_ind e step_vol
fim enquanto
Retorne o melhor peixe encontrado
```

Inicializado o cardume, realizam-se os movimentos de deslocamento individual, coletivo instintivo e coletivo volitivo do cardume, com as adaptações para o problema da QVR.

A. Deslocamento Individual

Para o deslocamento individual foi retirado o parâmetro $step_{ind}$ conforme proposto em [24]. Para realizar o deslocamento individual são escolhidas duas dimensões aleatórias a e b as quais sejam valores inteiros dos intervalos, $1 \leq a \leq N$, $1 \leq b \leq N$ e $a \neq b$, sendo N o tamanho da dimensão do peixe. Dessa forma o operador de deslocamento individual passa a ser:

$$x_p^{t+1} = swap(a, b, x_p^t), \tag{14}$$

sendo $swap(a, b, x)$ a função que permuta os valores das dimensões a e b do vetor x. Exemplificando, considerando o peixe $x^t = \{2, 1, 3, 4\}$ para um $N = 4$, supondo que foram escolhidos aleatoriamente as dimensões $a = 1$ e $b = 3$, a nova posição do peixe após aplicar a equação (14) será $x^{t+1} = \{3, 1, 2, 4\}$.

Após executado o operador de deslocamento individual, caso o peixe tenha melhorado o seu *fitness* o peixe assume a nova posição, do contrário, permanece na posição anterior. Foi proposta a criação de um vetor auxiliar M_p de comprimento N para cada peixe o qual identificará as dimensões que foram permutadas:

$$M_{p,d} = \begin{cases} 1, & \text{se} \quad x_{p,d}^{t+1} \neq x_{p,d}^t \\ 0, & \text{se} \quad x_{p,d}^{t+1} = x_{p,d}^t, \end{cases} \tag{15}$$

em que $M_{p,d}$ é valor relativo ao vetor auxiliar M_p na dimensão d, $x_{p,d}^{t+1}$ é o valor na dimensão d do peixe após o deslocamento individual e $x_{p,d}^t$ é o valor na dimensão d do peixe antes do operador de deslocamento individual.

O operador de alimentação não sofreu alteração, portanto, basta aplicar a equação (8).

B. Movimento Coletivo Instintivo

Este movimento leva o cardume em direção às dimensões dos peixes com melhores *fitness* na interação. Para executar este operador, cria-se o vetor instintivo I que vai conter os valores mais promissores de cada dimensão. Para tanto, são utilizados os vetores auxiliares M_p da equação (15). Para cada dimensão dos peixes, verifica o valor do índice quando o vetor M_p da referida dimensão for igual a 1. O vetor I assumirá em cada dimensão o valor do peixe que obteve a maior variação do *fitness*. Quando em uma dimensão o vetor I não assumir nenhum valor, então será atribuído o valor referente do melhor peixe global. O Algoritmo 2 descreve o processo de criação do vetor I.

Algoritmo 2 Vetor

```
para cada dimensão d faça
    Inicia Δ_{f|dimensao} = 0
    para cada peixe p faça
        se M_{p,d} == 1 então
            se Δ_{f(p)} > Δ_{f|dimensao} então
                I_d = x_{p,d}
                Δ_{f(p)} = Δ_{f|dimensao}
            fim se
        fim se
    fim para
    se Δ_{f|dimensao} = 0 então
        I_d = melhor_{peixe(d)}
    fim se
fim para
```

Com o vetor I, pode-se realizar o deslocamento comparando cada dimensão de x_p com I, selecionando todas as dimensões que $x_{p,d} \neq I_d$. Em seguida, sorteia uma porcentagem destas dimensões selecionadas para realizar a permutação, de forma que não haja repetição. A porcentagem é definida pelo parâmetro $step_{inst}$ que compreende em um intervalo de $[0, 1]$ [23]. Após sorteada a dimensão que sofrerá a permutação, definida por a, deve-se encontrar a dimensão em x_p que contém o valor de I na dimensão sorteada. Esta dimensão encontrada no vetor x_p é definida por b. Com os índices das dimensões a e b, executa-se a permutação aplicando a equação (14).

Após realizadas todas as permutações verifica-se o novo *fitness*. Caso seja superior ao anterior o peixe assume a nova posição. O Algoritmo 3 mostra este processo.

Algoritmo 3 Movimento Coletivo Instintivo

para cada peixe p **faça**
  Verifica a quantidade de dimensões selecionadas $Qtde_{dif}$ com $x_{p,d} \neq I_d$
  Crie uma lista $DimensoesSelecionadas$ com as dimensões selecionadas para o sorteio
  $permutacoes = step_{inst} \cdot Qtde_{dif}$
  $j = 1$
  para $j \leq permutacoes$ **faça**
    $a = Random(DimensoesSelecionadas)$
    Encontre a dimensão b onde $x_{p,b} = I_a$
    Aplique a equação (14) utilizando os valores a e b
    $j = j + 1$
  fim para
fim para

C. Movimento Coletivo Volitivo

Neste movimento é empregada uma estratégia semelhante ao movimento coletivo. Para executar este operador, cria-se o vetor baricentro B, o que irá permitir realizar a expansão ou a contração do cardume. Para tanto, são utilizados os vetores auxiliares M_p criados conforme a equação (15). Para cada dimensão dos peixes, verifica o valor do índice quando o vetor M_p da referida dimensão for igual a 1. O vetor B assumirá em cada dimensão o valor do peixe que obteve a maior variação de peso, calculado no operador de alimentação. Quando em uma dimensão o vetor B não assumir nenhum valor, então será atribuído o valor referente ao peixe com maior peso. O algoritmo 4 descreve o processo de criação do vetor baricentro B.

Algoritmo 4 Vetor Baricentro

para cada dimensão d **faça**
  Inicia $\Delta_{peso|dimensao} = 0$
  para cada peixe p **faça**
    se $M_{p,d} == 1$ **então**
      se $\Delta_{peso(p)} > \Delta_{peso|dimensao}$ **então**
        $B_d = x_{p,d}$
        $\Delta_{peso(p)} = \Delta_{peso|dimensao}$
      fim se
    fim se
  fim para
  se $\Delta_{peso|dimensao} = 0$ **então**
    $B_d = PeixeMaiorPeso_d$
  fim se
fim para

Com o vetor baricentro B, pode-se realizar a contração ou expansão do cardume. Quando o peso do cardume aumenta na interação atual, o cardume aproxima-se do baricentro, ocasionando a contração, sendo assim, compara-se cada dimensão de x_p com B, selecionando todas as dimensões em que $x_{p,d} \neq B_d$. Em seguida, sorteia uma porcentagem destas dimensões selecionadas para realizar a permutação, de forma que não haja repetição. Esta porcentagem é definida pelo parâmetro $step_{vol}$ que compreende em um intervalo de $[0, 1]$, sendo este parâmetro decrementado linearmente [23]. Após sorteada a dimensão a que sofrerá a permutação, deve-se encontrar a dimensão em x_p que contém o valor de I na dimensão sorteada, sendo esta dimensão encontrada no vetor x_p definida por b. Com os valores de a e b, é realizada a permutação aplicando a equação (14).

Quando o peso do cardume não aumenta em relação à interação anterior, o cardume deve se afastar do baricentro. O processo é o mesmo do caso de contração. Contudo, selecionam-se as dimensões que $x_{p,d} = B_d$.

Após realizadas todas as permutações, verifica-se o novo *fitness*. Caso seja superior ao anterior o peixe assume a nova posição. Os algoritmos 5 e 6 mostram este processo para o caso de contração e expansão do cardume, respectivamente.

Algoritmo 5 Movimento Coletivo Volitivo (contração)

para cada peixe p **faça**
  Verifica a quantidade de dimensões selecionadas $Qtde_{dif}$ que $x_{p,d} \neq B_d$
  Crie uma lista $DimensoesSelecionadas$ com as dimensões selecionadas para o sorteio
  $permutacoes = step_{vol} \cdot Qtde_{dif}$
  $j = 1$
  para $j \leq permutacoes$ **faça**
    $a = Random(DimensoesSelecionadas)$ //Não pode repetir
    Encontre a dimensão b onde $x_{p,b} = I_a$
    Aplique a equação (14) utilizando os valores a e b
    $j = j + 1$
  fim para
fim para

Algoritmo 6 Movimento Coletivo Volitivo (expansão)

para cada peixe p **faça**
  Verifica a quantidade de dimensões selecionadas $Qtde_{dif}$ que $x_{p,d} = B_d$
  Crie uma lista $DimensoesSelecionadas$ com as dimensões selecionadas para o sorteio
  $permutacoes = step_{vol} \cdot Qtde_{dif}$
  $j = 1$
  para $j \leq permutacoes$ **faça**
    $a = Random(DimensoesSelecionadas)$ //Não pode repetir
    Encontre a dimensão b onde $x_{p,b} = I_a$
    Aplique a equação (14) utilizando os valores a e b
    $j = j + 1$
  fim para
fim para

V. RESULTADOS

Para comparar o desempenho do FSS aplicado à QVR, foi escolhido o algoritmo SA [11]. Nas simulações, foram utilizados dicionários de tamanho $N = 128$, 256 e 512 com dimensões $K = 16$, isto é, blocos de imagens com 4×4 *pixels*. Os dicionários originais foram projetados utilizando o algoritmo *Linde-Buzo-Gray* (LBG) e para conjunto de treino foi escolhida a imagem *Lena*, codificada a 8 *bpp* (256 níveis de cinza), no formato *Portable Gray Map* (PGM) de 256×256 *pixels*.

O algoritmo FSS foi configurado com os parâmetros de tamanho de cardume, $step_{inst}$, $step_{vol}$ e número de interações conforme apresentado na Tabela I. Estes valores foram baseados em [23] e [24], ajustados de acordo com experimentos prévios. Na Tabela II são apresentados os valores configurados para os parâmetros temperatura inicial (T_0), temperatura final (T_f) e constante de resfriamento (β) do SA, baseando-se em [21] e também ajustados conforme experimentos prévios.

Tabela I: Parâmetros do FSS.

Parâmetros	$N = 128$	$N = 256$	$N = 512$
Tamanho do cardume	30	30	30
$step_{inst}$	0,1	0,1	0,1
$step_{vol}$ *inicial*	0,75	0,75	0,75
$step_{vol}$ *final*	0,25	0,25	0,25
Interações	3000	4000	4000

Tabela II: Parâmetros do SA.

Parâmetros	$N = 128$	$N = 256$	$N = 512$
T_0	50000	50000	50000
T_f	1	1	1
β	0,98	0,98	0,98

A função objetivo empregada é descrita na equação (6), para distância Hamming 1 e 2. Para cada tamanho de dicionário foram gerados cinco dicionários com distância Hamming 1 e mais cinco com distância Hamming igual a 2. As simulações foram realizadas em um computador com CPU Intel Core i5 7200U, 8 GB de memória RAM e SSD de 240 GB rodando o Windows 10. A linguagem utilizada para implementar o algoritmo FSS foi Python (versão 3.7).

As Tabelas III e IV apresentam as médias e os desvios padrão dos índices de desordem dos dicionários encontrados. Observa-se que o SA obteve resultados superiores ao FSS para $N = 128$ e distâncias Hamming 1 e 2. Enquanto que $N = 256$ o FSS apresentou melhor desempenho que SA para Hamming 1 e 2. Para $N = 512$ e distância Hamming igual a 2, o SA obteve média levemente superior ao FSS. Contudo, pode-se afirmar que um desvio padrão elevado no SA faz com que alguns dicionários projetados tenham um índice de desordem maior do que FSS, o que prejudica o desempenho do sistema. Em outras palavras, quando dois algoritmos apresentarem médias de índice de desordem próximas, deve-se optar por aquele que tenha menor desvio padrão.

Para avaliar a robustez dos dicionários organizados pelos algoritmos, a imagem *Lena* foi transmitida 10 vezes por um canal BSC sob diversos valores de probabilidade de erro de *bit*, considerando o melhor dicionário gerado por cada algoritmo. A avaliação da qualidade das imagens reconstruídas para cada dicionário foi realizada em termos de PSNR e SSIM.

Tabela III: Média e desvio padrão do valor do Índice de desordem, com distância Hamming = 1, encontrado pelos algoritmos.

N	**FSS**	**SA**
128	96186,40 ± 1223,36	95902,20 ± 1301,52
256	178169,00 ± 349,29	178900,80 ± 1042,59
512	376051,20 ± 2415,86	380271,00 ± 5239,02

Tabela IV: Média e desvio padrão do valor do Índice de desordem, com distância Hamming = 2, encontrado pelos algoritmos.

N	**FSS**	**SA**
128	79054,40 ± 149,55	78552,60 ± 170,23
256	181990,80 ± 662,67	182525,00 ± 324,33
512	469895,60 ± 799,72	468905,80 ± 2829,51

As Figuras 3 a 8 apresentam os valores médios de PSNR e SSIM em função da probabilidade de erro de *bit* do canal. Observa-se que em todos os tamanhos, os dicionários organizados apresentaram superioridade ao dicionário original. Para todos os tamanhos, dicionários organizados com distância *Hamming* igual a 2 pelo FSS, para probabilidade de erro de bit superior a 0,01, obtiveram desempenho superior aos organizados com *Hamming* igual a 1. Considerando o tamanho 128, os resultados alcançados pelos dicionários processados por FSS e SA ficaram muito próximos, sendo o FSS levemente superior. Para os tamanhos 256 e 512, o FSS apresenta superioridade ao SA quando compara os resultados para Hamming igual a 1 e 2. No tamanho 512, a escala utilizada pelo SSIM evidência a superiodade do FSS em relação ao SA.

As Figuras 9 e 10 apresentam as imagens *Lena* após serem submetidas à transmissão pelo canal BSC com probabilidade de erro de bit 0,01 e 0,1, respectivamente, para os dicionários de tamanho 512. Pode-se observar que as imagens reconstruídas a partir dos dicionários organizados obtiveram um desempenho de qualidade de imagem superior ao dicionário original, apresentando menos bloqueamentos espúrios.

VI. CONCLUSÕES

Neste trabalho verifica-se que a versão do FSS apresentada para fins de QVR representa uma alternativa satisfatória ao problema de organização de dicionários. Com o aumento do tamanho dos diconários o FSS apresenta melhor desempenho comparado ao SA. Considerar distância Hamming igual a 2, no cálculo do índice de desordem, é relevante e resulta em valores de qualidade não desprezíveis quando comparados a distância Hamming igual a 1, sobretudo com o aumento do tamanho dos dicionários e probabilidades de erro de *bit* superiores a 0,01.

Para trabalhos futuros, pretende-se avaliar a robustez dos dicionários organizados pelo FSS para outros tipos de canais, como canais com desvanecimento. Pretende-se também analisar a complexidade computacional das técnicas propostas.

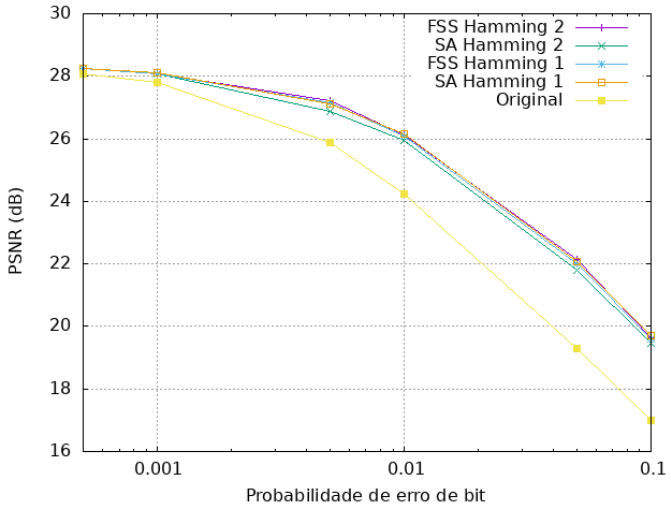

Figura 3: PSNR da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 128.

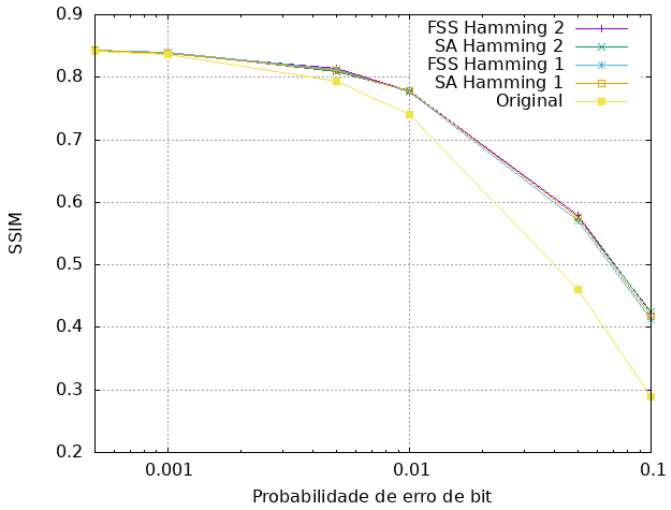

Figura 4: SSIM da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 128.

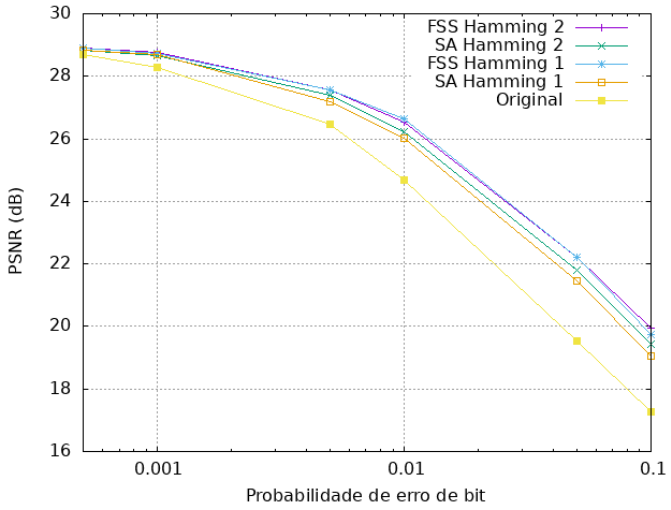

Figura 5: PSNR da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 256.

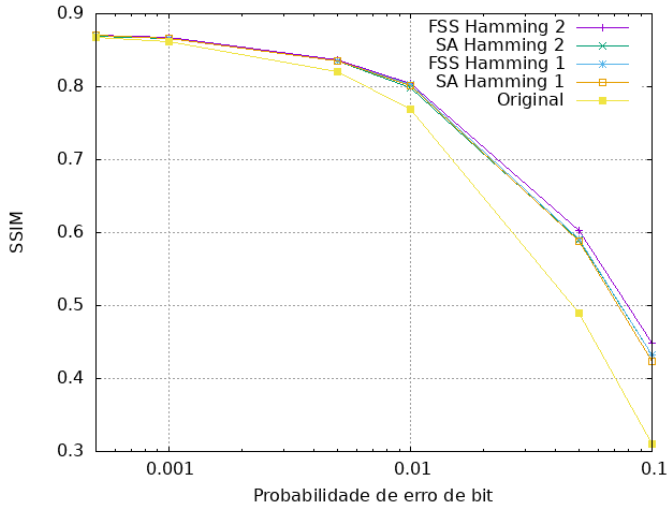

Figura 6: SSIM da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 256.

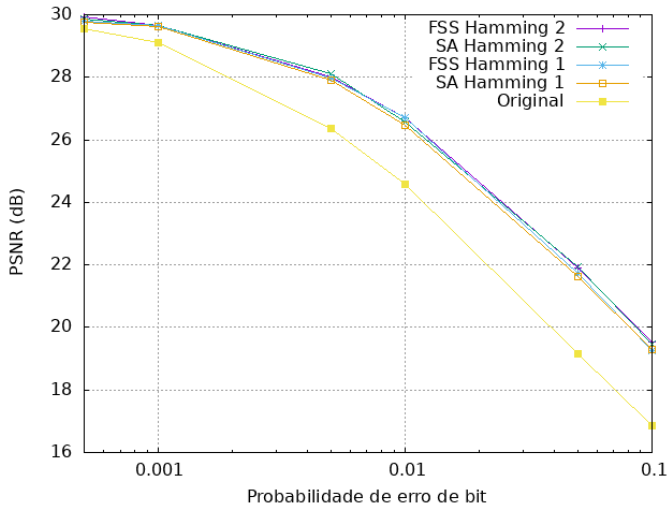

Figura 7: PSNR da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 512.

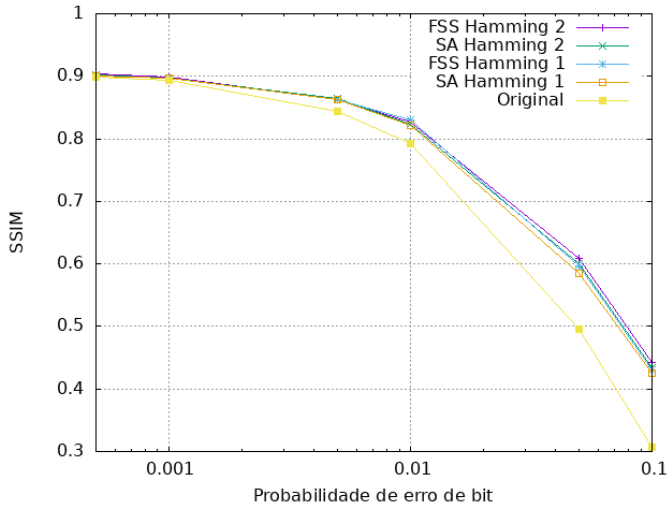

Figura 8: SSIM da imagem *Lena* reconstruída em função da probabilidade de erro de *bit* do canal utilizando um dicionário de tamanho 512.

(a) Dicionário original. (b) SA com Hamming 1. (c) FSS com Hamming 1. (d) SA com Hamming 2. (e) FSS com Hamming 2.

Figura 9: Imagem *Lena* reconstruída após transmissão por canal BSC com probabilidade de erro de *bit* igual a 0,01 e $N = 512$.

(a) Dicionário original. (b) SA com Hamming 1. (c) FSS com Hamming 1. (d) SA com Hamming 2. (e) FSS com Hamming 2.

Figura 10: Imagem *Lena* reconstruída após transmissão por canal BSC com probabilidade de erro de *bit* igual a 0,1 e $N = 512$.

REFERÊNCIAS

[1] P. Yu, H. Ren, Z. Le, e W. Hu, "Fast statistical estimation with vector quantization in compressed digital RoF system," in *2018 Asia Communications and Photonics Conference (ACP)*, 2018, pp. 1–3.

[2] P. K. Shah, R. P. Pandey, e R. Kumar, "Vector quantization with codebook and index compression," in *2016 International Conference System Modeling Advancement in Research Trends (SMART)*, 2016, pp. 49–52.

[3] Z. Wang, X. Xu, X. Qiang, e K. Li, "Learning vector quantization-aided detection for MIMO systems," *IEEE Communications Letters*, pp. 1–1, 2020.

[4] Y. Li, C. Chang, e H. Mingxing, "High capacity reversible data hiding for VQ-compressed images based on difference transformation and mapping technique," *IEEE Access*, vol. 8, pp. 32 226–32 245, 2020.

[5] R. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, nº. 2, pp. 4–29, 1984.

[6] A. P. Barros, C. Santana, D. Almeida, H. Silva, W. Queiroz, W. Lopes, e F. Madeiro, "Algoritmo ABC aplicado à atribuição de índices para quantização vetorial robusta de imagens," in *Anais do 14 Congresso Brasileiro de Inteligência Computacional*. Curitiba, PR: ABRICOM, 2019, pp. 1–8.

[7] F. Ferreira, H. Leitão, W. Lopes, e F. Madeiro, "Hybrid firefly-lindebuzo-gray algorithm for channel-optimized vector quantization codebook design," *Integrated Computer-Aided Engineering*, vol. 24, pp. 1–18, 05 2017.

[8] C. Fonseca, F. Ferreira, e F. Madeiro, "Vector quantization codebook design based on Fish School Search algorithm," *Applied Soft Computing*, vol. 73, pp. 958–968, 2018.

[9] E. Lima, G. Melo, W. Lopes, e F. Madeiro, "Um novo algoritmo para atribuição de indices: Avaliação em quantização vetorial de imagens," *Trends in Computational and Applied Mathematics*, vol. 10, nº. 2, pp. 167–177, 2009.

[10] J. Galvão, E. Lima, e F. Madeiro, "Atribuição de Índices para QV robusta usando o algoritmo Variable Neighborhood Search," *Trends in Computational and Applied Mathematics*, vol. 18, nº. 2, 2017.

[11] N. Farvardin, "A study of vector quantization for noisy channels," *IEEE Transactions on Information Theory*, vol. 36, nº. 4, pp. 799–809, 1990.

[12] T. N. Tuan e N. Q. Trung, "Improving the simulated annealing algorithm for the index assignment method to enhance the robustness of communication systems," *Journal of Research and Development on Information and Communication Technology*, pp. 13–20, 2014.

[13] T. N. Tuan, N. Q. Trung, e T. N. Khanh, "Improving the simulated annealing algorithm for source codeword index assignment by using the mechanism of tabu search algorithm," in *2016 International Conference on Advanced Technologies for Communications (ATC)*, 2016, pp. 91–96.

[14] K. Zeger e A. Gersho, "Pseudo-Gray coding," *IEEE Transactions on Communications*, vol. 38, nº. 12, pp. 2147–2158, 1990.

[15] J. Galvão, E. Lima, e F. Madeiro, "Atribuição de índices para QV robusta usando o algoritmo variable neighborhood search," *TEMA (São Carlos)*, vol. 18, nº. 2, pp. 197–214, 2017.

[16] L. Tianhao e Y. Songyu, "Evolutionary algorithm based index assignment algorithm for noisy channel," *Journal of Systems Engineering and Electronics*, vol. 15, nº. 3, pp. 431–435, 2004.

[17] J. Pan, F. McInnes, e M. Jack, "Application of parallel genetic algorithm and property of multiple global optima to VQ codevector index assignment for noisy channels," *Electronics Letters*, vol. 32, nº. 4, pp. 296–297, 1996.

[18] C. J. A. Bastos Filho, F. B. de Lima Neto, A. J. C. C. Lins, A. I. S. Nascimento, e M. P. Lima, "A novel search algorithm based on fish school behavior," in *2008 IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 2646–2651.

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

[20] R. A. Azevedo, F. Madeiro, W. T. A. Lopes, e E. A. O. Lima, "A quasi random symbol interleaving technique applied to image transmission by noisy channels," *IEEE Latin America Transactions*, vol. 14, nº. 3, pp. 1078–1085, 2016.

[21] W. T. A. Lopes, "Diversidade em modulação aplicada à transmissão de imagens em canais com desvanecimento." Tese de Doutorado em Engenharia Elétrica, Universidade Federal de Campina Grande, 2003.

[22] C. Bastos-Filho e D. Nascimento, "An enhanced fish school search algorithm," in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, 2013, pp. 152–157.

[23] R. F. Carneiro e C. J. A. Bastos-Filho, "Improving the binary fish school search algorithm for feature selection," in *2016 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2016, pp. 1–6.

[24] C. J. Santana, C. J. A. Bastos-Filho, M. Macedo, e H. Siqueira, "SBFSS: Simplified binary fish school search," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2595–2602.