Algoritmo ABC Aplicado à Atribuição de Índices para Quantização Vetorial Robusta de Imagens

Ana Paula Barros*, Clodomir Santana Jr.†, Danilo Almeida‡, Hugerles Silva§, Wamberto Queiroz¶, Waslon Lopes‖, Francisco Madeiro**
*Universidade de Pernambuco (UPE), Recife, e-mail: apbc@poli.br
†Universidade de Pernambuco (UPE), Recife, e-mail: clodomir@ieee.org
‡Universidade Federal de Campina Grande (UFCG), e-mail: danilo.almeida@ee.ufcg.edu.br
§Universidade Federal da Paraíba (UFPB), e-mail: hugerles.silva@ee.ufcg.edu.br
¶Universidade Federal de Campina Grande (UFCG), e-mail: wamberto@dee.ufcg.edu.br
‖Universidade Federal da Paraíba (UFPB), e-mail: waslon@ieee.org
**Universidade de Pernambuco (UPE), Recife, e-mail: madeiro@poli.br

***Resumo*—Nesse artigo é abordado o problema da Quantização Vetorial Robusta (QVR) no âmbito da transmissão de imagens por canal com desvanecimento rápido e ruído aditivo gaussiano branco convencional ou duplamente gatilhado (impulsivo). São apresentadas modificações no algoritmo *Artificial Bee Colony* (ABC) aplicado ao problema de otimização combinatorial de atribuição de índices na quantização vetorial robusta. O desempenho do ABC é comparado com outro algoritmo de otimização amplamente utilizado na literatura, o *Simulated Annealing* (SA). Resultados de simulação mostram que o algoritmo ABC modificado apresenta uma superioridade sobre o SA em termos de redução do índice de desordem em todos os tamanhos de dicionários testados, permitindo obter imagens reconstruídas com qualidade superior àquelas reconstruídas sem uso de QVR por canais ruidosos.**

***Palavras-chave*—Quantização vetorial robusta, otimização combinatorial, inteligência de exames, ruído impulsivo, desvanecimento generalizado, transmissão de imagens.**

I. Introdução

O crescente avanço das comunicações móveis e o aumento do número de acessos à internet proporcionam aos usuários o uso de uma imensa gama de recursos de mídias, entre eles, as imagens digitais, que têm grande relevância em inúmeras aplicações atualmente. Nesse contexto, é primordial o desenvolvimento de técnicas para melhorar a qualidade das reconstruções dessas imagens após sua transmissão através de canais ruidosos.

O processamento de imagens digitais pode demandar uma grande complexidade computacional. Em diversas aplicações, requisitos de largura de banda, para transmissão, e de espaço em memória, para armazenamento, devem ser reduzidos. Tal redução pode ser obtida por meio da compressão de sinais, que tem como objetivo diminuir o número de bits necessários para representá-los, mantendo um nível de qualidade adequado a uma dada aplicação.

Uma técnica amplamente utilizada nesses cenários é a Quantização Vetorial (QV) [1]. Trata-se de uma técnica de compressão com perdas, que permite a obtenção de elevadas taxas de compressão, porém apresenta uma alta sensibilidade aos erros introduzidos pelo canal de transmissão, em que o desempenho do quantizador pode ser bastante prejudicado [2].

A Quantização Vetorial Robusta (QVR) [2][3] é uma alternativa que visa reduzir a sensibilidade supracitada. Durante a transmissão de imagens por canais ruidosos, bits podem ser invertidos, gerando degradação na qualidade da imagem reconstruída. A QVR é uma técnica para organizar o dicionário por meio da adequada atribuição de índices aos seus vetores-código (vetores de reconstrução), tornando-o mais robusto aos erros de canal. Para isso, um dicionário inicial é obtido, por exemplo, com o algoritmo Linde-Buzo-Gray (LBG) [4]. No entanto, para determinação da configuração ótima de um dicionário de tamanho N, há $N!$ possíveis organizações dos vetores-código a serem avaliadas. Portanto, o problema da atribuição de índices para fins de QVR pode ser classificado como pertencente à classe de problemas NP-completos [5].

Dentre as alternativas para atacar o problema, podem ser citados os algoritmos de otimização, como o *Simulated Annealing* (SA) [6] e o *Variable Neighborhood Search* (VNS) [7], utilizados na atribuição de índices aos vetores-código na QVR, obtendo resultados satisfatórios em Canal Binário Simétrico (BSC, *Binary Symmetric Channel*). Contudo, nas comunicações móveis os canais são mais complexos que o BSC e o efeito do desvanecimento, provocado pelos múltiplos percursos de propagação dos sinais transmitidos, pode degradar significativamente o desempenho de sistemas de comunicações digitais.

Canais sujeitos a ruído impulsivo e desvanecimento generalizado η - μ [8] podem caracterizar uma ampla variedade de cenários de transmissão [9], incluindo desvanecimento em pequena e larga escala, com ausência de linha de visada entre o transmissor e o receptor, englobando, como casos especiais, por exemplo, as distribuições: Rayleigh, Weibull, Nakagami-m e Hoyt.

Este trabalho apresenta uma adaptação do algoritmo *Artificial Bee Colony* (ABC) [10] para promover a atri-

buição de índices (organização) aos vetores-código do dicionário, com o fim de minimizar o impacto dos erros na transmissão de imagens por canal sujeito a ruído impulsivo e desvanecimento. Neste estudo, adotou-se o esquema de modulação em amplitude e quadratura M-ário (M-QAM), com $M = 64$, estimação perfeita do ganho do canal no receptor e o modelo de ruído aditivo Gaussiano branco duplamente gatilhado (G^2AWGN, *Double Gated Additive White Gaussian Noise*) [11], que engloba como casos particulares outros modelos de ruído mais simples, incluindo o ruído aditivo Gaussiano branco (AWGN, *Additive White Gaussian Noise*). Para avaliar a qualidade da imagem reconstruída, duas medidas de distorção foram consideradas: a relação sinal-ruído de pico (PSNR, *Peak Signal-to-noise Ratio*) e a similaridade estrutural (SSIM, *Structural Similarity*) [12].

O restante do artigo está organizado da seguinte forma: a Seção II apresenta os fundamentos da QV e da QVR, uma breve abordagem do algoritmo ABC, originalmente usado para otimização contínua, a Seção III descreve as modificações realizadas no algoritmo ABC para que ele seja aplicado ao problema de otimização combinatorial (para fins de QVR), a Seção IV contempla os resultados de simulação e a Seção V apresenta a conclusão.

II. Referencial Teórico

A. Quantização Vetorial

A QV é definida como um mapeamento Q de um vetor x pertencente ao espaço euclidiano K-dimensional, $\mathbb{R}^K$, em um vetor pertencente a um subconjunto finito chamado dicionário W, ou seja, $Q : \mathbb{R}^K \rightarrow W$, em que $W = \{w_i; i = 1, 2, ..., N\}$ é o conjunto de vetores-código K-dimensionais, também chamados de vetores de reconstrução ou protótipos, em que N é o tamanho do dicionário, isto é, a quantidade de vetores-código. Cada índice i, correspondente a um vetor-código w_i, pode ser representado por uma palavra-binária $b_i \in \{0, 1\}^b$, em que $b = \log_2 N$ corresponde ao número de *bits* usados para codificar o índice de cada vetor do dicionário. A taxa de codificação do quantizador vetorial, que mede o número de *bits* por componente do vetor, é definida por $R = \frac{\log_2 N}{K}$.

B. Quantização Vetorial Robusta

Na transmissão de imagens, o transmissor e o receptor contém uma cópia do dicionário, assim, o transmissor envia apenas os índices (palavras-código) dos blocos codificados. Durante esse processo, os erros do canal podem inverter bits correspondentes aos índices transmitidos. Alterado um índice, são afetados K pixels da imagem reconstruída. A degradação na imagem pode ser minimizada com atribuição de índices com grande/pequena distância de Hamming a vetores-código com grande/pequena distância euclidiana [3].

Para medir o grau de organização do dicionário, ou seja, sua robutez aos erros do canal, pode-se calcular o índice de desordem (Id) [3], que é definido em sua forma simplificada como

$$Id = \sum_{i=1}^{N} \sum_{j \in H^1(i)} \|w_i - w_j\|^2, \quad (1)$$

em que $H^1(i)$ é o conjunto de todas as palavras-código com distância de Hamming unitária em relação à palavra binária (i) [3].

C. Algoritmo ABC

Algoritmos bioinspirados baseados em populações e meta-heurísticas vem sendo usados para resolver problemas de busca e otimização em vários domínios de problemas para os quais soluções robustas são difíceis ou impossíveis de encontrar. Inspirados em processos naturais, foram desenvolvidos modelos computacionais com base no conceito de inteligência coletiva.

Inteligência Coletiva ou Inteligência de Colônias, também referenciada como Inteligência de Enxames (IE), é a denominação aplicada à tentativa de desenvolvimento de algoritmos para solução distribuída de problemas inspirando-se no comportamento de colônias de insetos sociais e outras sociedades de animais [13]. Um enxame também pode ser considerado como qualquer coleção de agentes ou indivíduos que interagem.

As propriedades principais de um sistema de inteligência de enxame são [14]: proximidade – os agentes devem ser capazes de interagir; qualidade – os agentes devem ser capazes de avaliar seus comportamentos; diversidade – permite ao sistema reagir a situações inesperadas; estabilidade – nem todas as variações ambientais devem afetar o comportamento de um agente; adaptabilidade – capacidade de adequação a variações ambientais.

Uma característica dos algoritmos de enxame é que a obtenção de um comportamento inteligente ocorre por meio de dois conceitos fundamentais: auto-organização e divisão de trabalho [15]:

a) auto-organização se baseia em quatro propriedades básicas: realimentação (*feedback*) positiva, *feedback* negativo, flutuações e interações mútiplas.

b) divisão de trabalho consiste nas diferentes tarefas simultâneas realizadas por indivíduos especializados, tornando o desempenho mais eficiente e permitindo que o enxame responda às condições alteradas no espaço de busca.

Vários grupos de animais tiveram suas características mapeadas em algoritmos de otimização, os quais tem sido amplamente utilizados em grande número de problemas, em diversos domínios. Por exemplo, o comportamento coletivo de aves e peixes foi a principal inspiração para os algoritmos *Particle Swarm Opitimization* (PSO) [16] e *Fish School Search* (FSS) [17], respectivamente.

Além do PSO e do FSS, existem outros algoritmos de enxames, como o *Artificial Bee Colony* (ABC) [18]. Na literatura encontram-se diversos exemplos de aplicação do ABC em áreas como: agrupamento de dados, roteamento, treinamento de redes neurais, implantação de rede de sensores sem fios e análise de imagens.

O ABC tem como fonte de inspiração o comportamento de abelhas na busca por fontes de alimento. Salvo poucas variações, o modelo consiste, basicamente, de três componentes essenciais: abelhas empregadas, observadoras e escoteiras. As empregadas são responsáveis por explorar o espaço de busca e, uma vez que localizam o alimento, compartilham sua localização com uma abelha observadora, que explora a vizinhança da fonte de alimento, procurando por fontes melhores. A escoteira, terceiro tipo de abelha, é chamada quando uma fonte de alimento não pode ser melhorada após sucessivas tentativas (T), parâmetro definido pelo usuário. A escoteira deve buscar uma nova fonte de alimento, em lugar aleatório, dentro do espaço de busca.

A inicialização de cada solução candidata no ABC pode ser definida por

$$x_{i,d} = x_{d_{min}} + rand(0,1)(x_{d_{max}} - x_{d_{min}}), \qquad (2)$$

sendo $x_{i,d}$ a posição da fonte de alimento i, $rand$(0,1) é a função que gera o número aleatório no intervalo [0,1] usando a distribuição uniforme, $x_{d_{min}}$ e $x_{d_{max}}$ são, respectivamente, os limites inferior e superior da dimensão d no espaço de busca, que para o problema de QVR todas as dimensões tem os mesmos limites, em que 1 é o limite inferior e o tamanho do dicionário é o limite superior. Vale salientar que esse mesmo procedimento de geração das soluções iniciais é utilizado pelas abelhas escoteiras.

Após a inicialização, as abelhas empregadas são acionadas:

$$v_{i,d} = x_{i,d} + rand(-1,1)(x_{i,d} - x_{j,d}), \qquad (3)$$

em que v é uma fonte de alimento candidata, i e j são fontes de alimento selecionadas de 1, 2, ..., SN em que SN é o número de fontes de alimento que indica a quantidade de abelhas empregadas e observadoras, d pode assumir qualquer valor inteiro no conjunto [1, 2, ..., D] sendo D o número de dimensões, x é a posição atual dada a fonte de alimento, e $rand$(-1,1) é a função que gera o número aleatório no intervalo [-1,1] usando a distribuição uniforme.

A equação que retorna o valor que representa a qualidade de uma fonte de alimento é chamada de função objetivo. As abelhas empregadas e observadoras só atualizam a localização da fonte de alimento se a fonte candidata for melhor que a atual (melhor *fitness*), usando o que se chama de abordagem gananciosa. A função objetivo pode ser dada por

$$fitness_i = \begin{cases} 1/(1+f_i), & \text{se } f_i \geq 0. \\ 1 + abs(f_i), & \text{caso contrário.} \end{cases} \qquad (4)$$

em que f_i é o valor retornado da função objetivo da fonte de alimento x_i e abs é uma função que retorna o valor absoluto de um número.

Depois de todas as abelhas empregadas executarem sua função, as abelhas observadoras são acionadas. Essas abelhas operam de maneira semelhante à operação das anteriores, porém a escolha da fonte de alimento não é feita de forma aleatória [19]. As observadoras escolhem as fontes de alimento com base em uma probabilidade de seleção dada pela Equação 5, que apresenta como é determinada a probabilidade de a abelha observadora selecionar uma fonte de alimento para explorar. A ideia desse mecanismo é selecionar com mais frequência as fontes de alta qualidade

$$p_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}, \qquad (5)$$

em que $fitness_i$ é a qualidade da solução para uma determinada fonte de alimento x e SN o número de fontes de alimento. Quando as abelhas empregadas ou as observadoras não conseguem melhorar uma fonte de alimento, um contador de tentativas é incrementado e quando esse contador atinge o limite especificado por T, a abelha escoteira é acionada para gerar uma nova fonte de alimento.

O Algoritmo 1 ilustra o pseudo-código para o algoritmo ABC considerando problemas de maximização.

III. Algoritmo ABC para fins de QVR

Como o ABC original foi projetado para problemas de otimização contínua, foi necessário realizar modificações para que ele fosse aplicado à QVR (otimização combinatorial). As mudanças apresentadas neste trabalho foram baseadas na versão do ABC para otimização binária proposta por Santana Jr *et al.* [10].

A primeira mudança necessária foi feita na inicialização do algoritmo. Nesse caso, em vez de usar a Equação 2 para gerar as posições das fontes de alimentação, utiliza-se o esquema descrito por

$$x_i = rand\_perm(), \qquad (6)$$

em que $rand\_perm$ é uma função que retorna uma permutação aleatória do conjunto de índices disponíveis.

O Algoritmo 2 mostra o mecanismo utilizado pelas abelhas empregadas e observadoras.

Novamente, o que difere o processo das abelhas empregadas do processo das observadoras é a forma de seleção da fonte de alimentação. Neste trabalho, foi mantida a estratégia proposta pelo ABC original (seleção aleatória ou pela probabilidade).

O novo parâmetro max_flips proposto em [10] determina o número máximo de dimensões que podem ser alteradas em uma fonte de alimentação. Esse parâmetro é importante para controle da convergência do enxame, evitando a convergência prematura, *ceil* é uma função que retorna o próximo número inteiro e a função *swap* troca a posição do vetor v_i que contém o valor de $x_{j,d}$ e a posição que tinha o valor de $x_{j,d}$ passa a ter o valor de $v_{i,d}$. Essa função garante que as fontes de alimento candidatas são uma mutação válida das fontes atuais. Por fim, a abelha escoteira passa a utilizar a Equação 6 para gerar novas fontes de alimento quando necessário.

A função objetivo utilizada é definida para maximizar a redução do índice de desordem, ou seja

$$F(x) = \max\{Id(x_{inicial}) - Id(x_{novo})\}, \qquad (7)$$

Algoritmo 1: Pseudocódigo ABC

```
Inicialize todas as fontes de alimento aplicando a Equação 2;
Calcule o fitness das fontes (Equação 4);
while Critério de parada não for atendido do
    # Fase das Abelhas Empregadas
    for cada abelha empregada i = 1, ..., SN do
        Selecione uma fonte de alimentação aleatória j da lista de fontes de alimento;
        Encontre uma fonte de alimento candidata na vizinhança da fonte selecionada (Equação 3);
        Calcule o fitness da fonte candidata;
        if fitness da fonte candidata > fitness da fonte atual then
            Atualize a posição da fonte atual para a fonte de alimento candidata;
        else
            Incremente o numero de tentativas da fonte de alimento i;
    Calcule a probabilidade de seleção das fontes de alimento com a Equação (5);
    #Fase das Abelhas Observadoras
    for cada abelha observadora do
        for cada fonte de alimento i = 1, ..., SN do
            faça r = rand(0, 1);
            if r > p_i then
                Envie a observadora i para a i-ésima fonte de alimento;
                Encontre uma fonte de alimento candidata aplicando a Equação 3;
                Calcule o fitness da fonte candidata;
                if fitness da fonte candidata > fitness da fonte atual then
                    Atualize a posição da fonte atual para a fonte de alimento candidata;
                else
                    Incremente o número de tentativas da fonte de alimento i;
    #Fase das Abelhas Escoteiras
    if número de tentativas de alguma fonte de alimento ≥ limite de tentativas (T) then
        Produza uma nova fonte de alimento usando a Equação 2;
        Calcule o fitness da fonte produzida;
Retorne a melhor fonte encontrada;
```

em que $x_{inicial}$ é o conjunto original de índices do dicionário e x_{novo} representa uma nova ordenação desse conjunto de índices.

IV. Experimentos e Resultados

As simulações utilizaram dicionários de tamanho $N = 32, 64, 128, 256$ e 512 com dimensão $K = 16$

Algoritmo 2: Processo Utilizado pelas Abelhas Empregadas e Observadoras.

```
Faça j como sendo uma fonte de alimento selecionada de acordo com a regra de seleção do tipo de abelha;
Faça num_dim = ceil(max_flips · D);
Selecione num_dim dimensões aleatórias da fonte de alimento j usando uma distribuição uniforme;
for dimensão d selecionada do
    v_i = swap(v_{i,d}, x_{j,d});
Retorne v_i;
```

(blocos de 4×4 *pixels*). Para conjunto de treino dos dicionários foi usada a imagem Goldhill codificada a 8 *bpp* (256 níveis de cinza), no formato *Portable Gray Map* (PGM) de 256×256 *pixels*. O algoritmo utilizado para projetar o dicionário foi o LBG.

O algoritmo ABC foi configurado com os parâmetros de limite de tentativas (T), número de fontes de alimento (SN), limite de inversão (max_flips) e a dimensão (D) é o espaço de busca que é igual ao tamanho do dicionário (N). Esses valores foram baseados nos valores recomendados em [10] e foram sendo ajustados em experimentos prévios.

Tabela I
Parâmetros e percentual médio de redução do índice de desordem do ABC modificado

N	T	max_flips	SN	Avaliações	ΔId
32	40	0,10	15	100000	58,18%
64	50	0,05	15	100000	63,45%
128	80	0,05	16	100000	65,99%
256	150	0,05	15	180000	67,98%
512	200	0,01	16	400000	70,06%

O algoritmo escolhido para ser comparado com o algoritmo ABC proposto foi o SA [6] o qual foi configurado com os parâmetros utilizados em [3], temperatura inicial (t_0), temperatura final (t_f) e constante de resfriamento β.

Tabela II
Parâmetros e percentual médio de redução do índice de desordem do SA

N	t_0	t_f	β	Avaliações	ΔId
32	10000	1	0,95	100000	57,96%
64	10000	1	0,95	100000	63,20%
128	10000	1	0,95	100000	65,17%
256	50000	1	0,97	180000	67,35%
512	50000	10	0,97	400000	69,15%

Para os algoritmos selecionados, o critério de parada adotado foi o número de avaliações de *fitness*. A função objetivo empregada é descrita na Equação 7 e para cada dicionário foram executadas 30 simulações independentes. Os otimizadores foram implementados utilizando a Linguagem de programação Julia [20] e as simulações foram executadas em um computador com CPU Intel Core i7-4500U, 8GB de memória RAM e SSD de 240GB rodando Ubuntu 19.04 64 bit.

As Figuras 1 a 5 apresentam as melhores curvas de convergência encontradas pelos algoritmos para todos os dicionários. O ABC proposto conseguiu superar levemente os resultados do SA, o que pode ser observado particularmente pela pequena distância entre as curvas ao final do treinamento, para os dicionários de tamanho 128, 256 e 512.

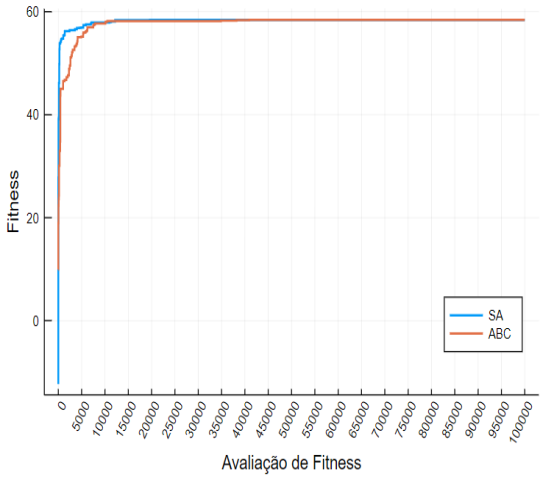

Figura 1. Curva de convergência dos algoritmos em termos de função $fitness$ para dicionário com 32 dimensões.

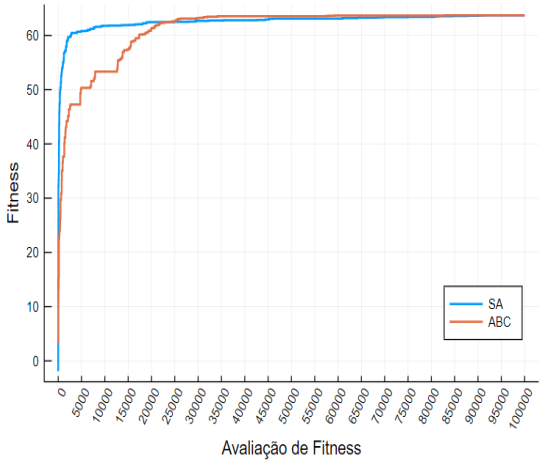

Figura 2. Curva de convergência dos algoritmos em termos de função $fitness$ para dicionário com 64 dimensões.

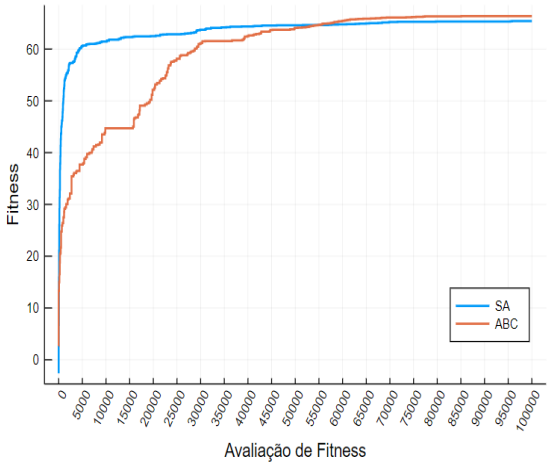

Figura 3. Curva de convergência dos algoritmos em termos de função $fitness$ para dicionário com 128 dimensões.

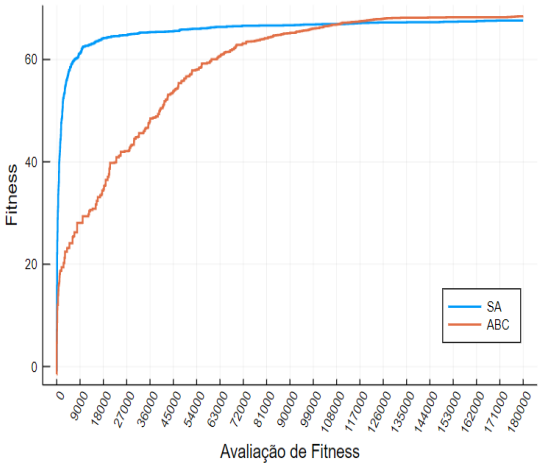

Figura 4. Curva de convergência dos algoritmos em termos de função $fitness$ para dicionário com 256 dimensões.

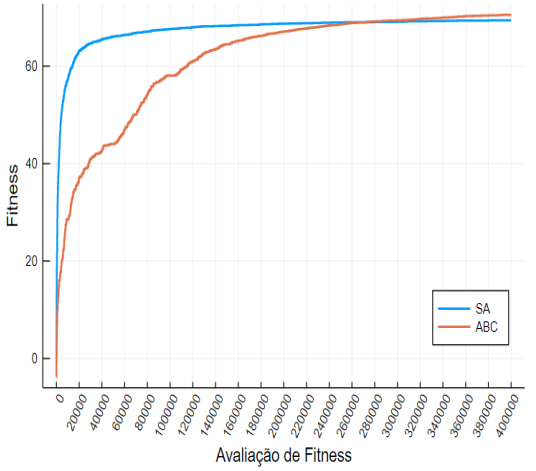

Figura 5. Curva de convergência dos algoritmos em termos de função $fitness$ para dicionário com 512 dimensões.

A Tabela III apresenta as médias e desvios padrão para o *fitness* e o tempo necessário até a execução de uma chamada da função objetivo. Como pode ser observado, os valores médios de redução do índice de desordem do ABC modificado foram levemente melhores do que os do SA. Em relação ao tempo de execução, o SA apresentou os menores tempos para todos os tamanhos de dicionários.

Outra análise que pode ser feita é em relação ao tempo que cada algoritmo leva para executar as chamadas das funções objetivo. Esse tempo é um indicativo da complexidade dos operadores aplicados pela técnica. Por exemplo, técnicas com poucos laços e operações aritméticas de baixo custo computacional apresentam um tempo menor que as técnicas com muitos laços e operações sob vetores e estruturas de dados complexas. Nesse sentido, pode ser visto que o ABC teve um desempenho um pouco mais lento do que o SA.

Tabela III
Resultados de fitness e tempo de avaliação dos algoritmos ABC e SA (média e desvio padrão)

Dimensão		ABC	SA
32	Fitness	58,18 ± 0,14	57,96 ± 0,36
	Tempo	2,98 ± 0,14	2,10 ± 0,13
64	Fitness	63,45 ± 0,15	63,20 ± 0,22
	Tempo	5,86 ± 0,28	4,64 ± 0,27
128	Fitness	65,99 ± 0,29	65,17 ± 0,19
	Tempo	16,48 ± 0,87	12,05 ± 0,66
256	Fitness	67,98 ± 0,32	67,35 ± 0,21
	Tempo	52,90 ± 3,45	36,55 ± 1,83
512	Fitness	70,06 ± 0,27	69,15 ± 0,16
	Tempo	221,52 ± 3,31	194,96 ± 3,46

Para avaliar a robustez do dicionário organizado, a imagem Goldhill foi usada como conjunto de teste sendo transmitida 50 vezes por canal com esquema de modulação em amplitude e quadratura M-ário (M-QAM) com $M = 64$, modelo de ruído AWGN ou ruído G^{2}AWGN e com desvanecimento η - μ, sob diferentes valores de μ, considerando $\eta = 1{,}1$ e duração dos pulsos e surtos impulsivos, assim como suas probabilidades de ocorrência iguais a 0,5. No modelo de ruído G^{2}AWGN o valor da relação sinal-ruído impulsivo (SNI), definida como a razão entre a potência do sinal e a potência do ruído impulsivo, considerada na simulação foi de 10 dB, para o modelo de ruído AWGN a componente do ruído de fundo (SNR) adotou valores entre 0 dB e 20 dB. Assumiu-se estimação perfeita do ganho do canal no receptor e também entrelaçamento perfeito, que minimiza os erros em rajada entre codificador e decodificador. As amostras foram geradas a partir do método de Monte Carlo.

As curvas nas Figuras 6 e 7 correspondem a valores médios de 50 transmissões realizadas através do canal. São apresentados valores de PSNR e SSIM em função da relação sinal-ruído (SNR, *Signal-to-noise Ratio*). Nelas percebe-se que tanto a PSNR quanto a SSIM aumentam com a SNR, para os diferentes valores de μ, para os dois dicionários (original e organizado). Para μ fixo, são obtidos valores maiores de PSNR e SSIM para o dicionário organizado (com QVR) de tamanho $N = 256$, tanto para o modelo de ruído AWGN quanto para o G^{2}AWGN.

Nas Figuras 8 e 9 são apresentadas as curvas de PSNR e SSIM para os dicionários: Original, Organizado ABC e Organizado SA, para o tamanho de dicionário de $N = 512$. Nota-se que para o modelo de ruído G^{2}AWGN (impulsivo) o desempenho do dicionário organizado pelo algoritmo ABC modificado é levemente superior.

A Figura 10 permite realizar uma comparação visual das imagens reconstruídas no receptor para os dicionários: original, organizado pelo algoritmo ABC modificado e organizado pelo algoritmo SA, para o tamanho de dicionário de $N = 512$. Nota-se que o uso da QVR permite uma melhor qualidade na imagem em todos os dicionários organizados, os quais apresentaram vetores-código (blocos de pixels) mais próximos dos vetores que seriam resultados da decodificação de índices sem erros.

A Tabela IV mostra os valores de PSNR e SSIM para os dicionários de tamanhos 256 e 512, Original e Organizados pelos algoritmos ABC modificado e SA, para a condição de canal de $M = 64$, $\eta = 1{,}1$, $\mu = 1{,}0$, o produto $\alpha\beta p1p2 = 0{,}5$, SNR = 20 dB e SNI = 10 dB.

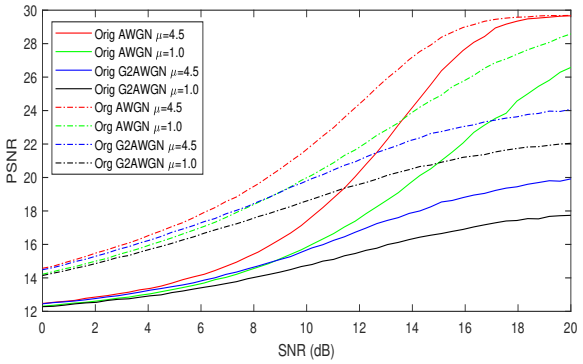

Figura 6. Valores de PSNR médios em função da SNR em canais com ruído AWGN ou G^{2}AWGN e desvanecimento η - μ, para os dicionários Original e Organizado, de $N = 256$.

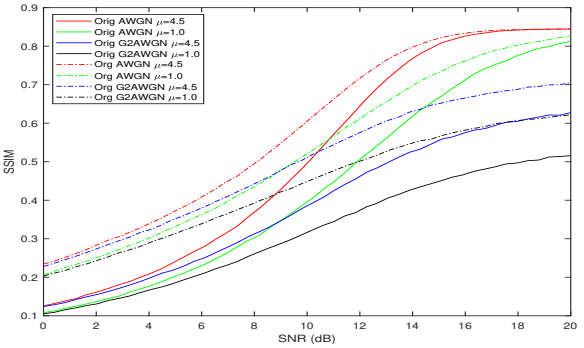

Figura 7. Valores de SSIM médios em função da SNR em canais com ruído AWGN ou G^{2}AWGN e desvanecimento η - μ, para os dicionários Original e Organizado, de $N = 256$.

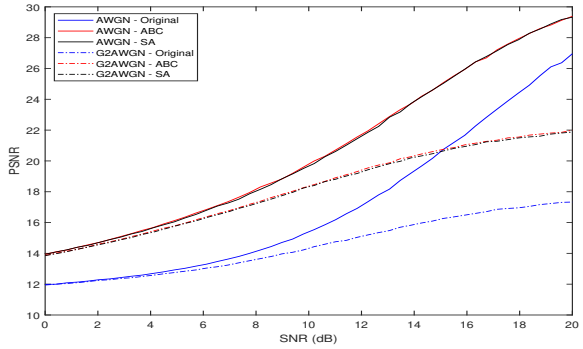

Figura 8. Valores de PSNR médios em função da SNR em canais com ruído AWGN ou G^{2}AWGN e desvanecimento η - μ, para os dicionários original, organizado com algoritmo ABC modificado e organizado com algortimo SA, de $N = 512$.

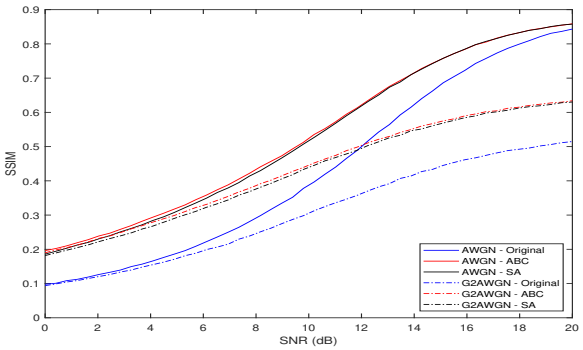

Figura 9. Valores de SSIM médios em funçãoo da SNR em canais com ruído AWGN ou G^{2}AWGN e desvanecimento η - μ, para os dicionários original, organizado com algoritmo ABC modificado e Organizado com algoritmo SA, de $N = 512$.

Tabela IV
VALORES DE PSNR E SSIM MEDIDOS APÓS TRANSMISSÃO DA IMAGEM NO CANAL

Dicionário	N	PSNR (dB)	SSIM
Original	256	21,69	0,6965
Organizado ABC	256	25,48	0,7558
Organizado SA	256	25,46	0,7504
Original	512	21,72	0,7122
Organizado ABC	512	26,24	0,7853
Organizado SA	512	25,58	0,7755

V. Conclusão

Com base nos resultados apresentados neste trabalho, verifica-se que a versão do algoritmo ABC modificado, para fins de QVR, constitui-se em alternativa satisfatória ao problema de atribuição de índices na organização de dicionários permitindo obter imagens reconstruídas de melhor qualidade quando comparada à QV convencional nas transmissão de imagens através de canal sujeito a ruído impulsivo e desvanecimento generalizado η - μ, que é um canal mais realístico que o BSC, utilizado em outros trabalhos. Como trabalhos futuros, pretende-se avaliar a

(a) Imagem Goldhill (b) Original

(c) Organizado ABC (d) Organizado SA

Figura 10. (a) Imagem Goldhill usada para projetar os dicionários aplicados à QVR, (b) Imagem reconstruída no receptor com o dicionário original (sem QVR), com $N = 512$, (c) Imagem reconstruída no receptor com o dicionário organizado pelo ABC (com QVR), com $N = 512$, (d) Imagem reconstruída no receptor com o dicionário organizado pelo SA (com QVR), com $N = 512$.

robustez dos dicionários organizados em canais com outros cenários de desvanecimento, como por exemplo, desvanecimento α - μ, sujeitos a outros tipos de ruído e utilizando técnicas capazes de entrelaçamento (*interleaving*) de bits. A pesquisa também pode ser estendida para imagens coloridas, assim como para outros tipos de sinais, como voz e vídeo.

Agradecimentos

Esse trabalho contou com apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), da Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001.

Referências

[1] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

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

[3] W. T. A. Lopes, "Diversidade em modulaçao aplicada a transmissão de imagens em canais com desvanecimento," *Tese (Doutorado em Engenharia Elétrica), Universidade Federal de Carnpina Grande*, 2003.

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

[5] E. Lima, G. Melo, W. Lopes, and F. Madeiro, "Um novo algoritmo para atribuição de índices: Avaliação em quantização vetorial de imagens," *TEMA - Tendências em Matemática Aplicada e Computacional*, vol. 10, no. 2, pp. 167–177, 2009.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[7] N. Mladenović, D. Urošević, and S. Hanafi, "Variable neighborhood search for the travelling deliveryman problem," *4OR*, vol. 11, no. 1, pp. 57–73, 2013.

[8] M. D. Yacoub, "The κ-μ distribution and the η-μ distribution," *IEEE Antennas and Propagation Magazine*, vol. 49, no. 1, pp. 68–81, 2007.

[9] H. S. Silva, M. S. de Alencar, W. J. de Queiroz, R. de A Coelho, and F. Madeiro, "Bit error probability of M-QAM under impulsive noise and fading modeled by using Markov chains," *Radioengineering*, vol. 27, no. 4, pp. 1183–1190, 2018.

[10] C. J. Santana Jr, M. Macedo, H. Siqueira, A. Gokhale, and C. J. Bastos Filho, "A novel binary artificial bee colony algorithm," *Future Generation Computer Systems*, vol. 98, pp. 180–196, 2019.

[11] H. S. Silva, M. S. de Alencar, W. J. de Queiroz, D. B. T. Almeida, and F. Madeiro, "Closed-form expression for the bit error probability of the M-QAM for a channel subjected to impulsive noise and Nakagami fading," *Wireless Communication and Mobile Computing*, vol. 2018, pp. 1–9, 2018.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and 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.

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[14] M. M. Millonas, "Swarms, phase transitions, and collective intelligence," *In C.G. Langton (Ed.), Artificial Life III, pp. 417-445. Reading, MA: Addison-Wesley*, 1994.

[15] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," tech. rep., Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

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

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

[18] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[19] M. Mernik, S.-H. Liu, D. Karaboga, and M. Črepinšek, "On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation," *Information Sciences*, vol. 291, pp. 115–127, 2015.

[20] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.