Recent Applications of Swarm-Based Algorithms to Color Quantization

María-Luisa Pérez-Delgado

Abstract Nowadays, images are very important elements in everyday communication. Current devices include high-quality displays that allow showing images with many colors. Nevertheless, the size of these images is a major issue when the speed of transmission and the storage space must be taken into consideration. The color quantization process reduces the number of different colors used to represent an image while trying to make the new image similar to the original. In addition to enabling visualization with low-end devices and efficient image storage and transmission, color reduction is related to other operations applied to images, such as segmentation, compression, texture analysis, watermarking and content-based image retrieval. The color quantization problem is complex, since the selection of the best colors to represent the image is a NP-complete problem. The complexity and interest of the problem have led to several solution approaches over the years. Recently, several interesting solutions have been proposed that apply swarm-based algorithms. Said algorithms are based on a population of individuals that cooperate to solve a problem. This chapter focuses on the swarm-based solutions proposed for the color quantization problem and shows that these novel methods can generate better images than those obtained by classical solution approaches.

Keywords Color quantization · Swarm-based methods · Clustering

1 Introduction

Images are very important elements in human communication. Currently available devices can define, capture, store and display high-quality images. Nevertheless, the human eye can only distinguish a limited range of colors. For this reason, some operations performed on images may consider a limited number of different colors

M.-L. Pérez-Delgado (✉)
Escuela Politécnica Superior de Zamora, University of Salamanca, Av. Requejo 33, C. P. 49022 Zamora, Spain
e-mail: mlperez@usal.es

D. J. Hemanth et al. (eds.), *Recent Advances on Memetic Algorithms and its Applications in Image Processing*, Studies in Computational Intelligence 873,
https://doi.org/10.1007/978-981-15-1362-6_5

to represent the images. When an image is represented with fewer colors, its quality is reduced as well the space required to store the image. This second feature accelerates the transmission of said image in case such operation is required.

Determining the optimal palette to reduce the colors of an image is a difficult problem [1]. For this reason, several solutions have been proposed that apply two different approaches:

- Splitting methods, which consider the color cube and apply an iterative process that divides that cube into smaller boxes. The process ends when the quantity of boxes is equal to the number of colors that the method must define. Then, a color is selected to represent each box and the set of colors defines the quantized palette. Some popular methods of this type are those proposed in [2–6].
- Clustering-based methods, which apply techniques proposed to divide a set of items into several groups or clusters according to the features of said items. When the items to be clustered are the pixels of an image, the groups are defined so that they include pixels with a similar color. Some clustering methods that have been applied to reduce the colors of an image are *K*-means, competitive learning, neural networks and swarm-based algorithms.

Swarm-based methods constitute a set of innovative techniques to solve difficult problems. The characteristics of these methods allow them to obtain good results for different types of problems. This chapter describes several methods of this kind recently proposed to deal with the color quantization problem. The description begins with the definition of the problem to be solved. Following, the main features of the swarm-based methods are presented. After this, several color quantization methods that use swarm-based solutions are described. The chapter ends with the conclusions.

2 Problem Definition

A digital image is composed by a set of N pixels represented using a certain color space. When the RGB color space is considered, each pixel $\boldsymbol{p}_i$ of the image, with $1 \leq i \leq N$, is represented by three integer values between 0 and 255, $\boldsymbol{p}_i = (R_i, G_i, B_i)$, which indicate the amount of red, green and blue of the pixel. Therefore, this color space allows us to use 256^3 different colors (i.e., more than 16 million colors).

The color quantization process tries to reduce the number of different colors of the image without losing basic information. To achieve this goal, a color quantization method performs two operations. First, it selects a small number of different colors, q, to define a new palette, called quantized palette, $\boldsymbol{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_q\}$, where $\boldsymbol{c}_j = (R_j, G_j, B_j)$. Next, the colors of this palette are used to determine the new image, called quantized image. To this effect, each pixel $\boldsymbol{p}_i$ of the initial image is replaced by another pixel $\boldsymbol{P}_i$, which takes one of the q colors from the quantized palette.

3 Overview of Swarm-Based Methods

In nature, there are some groups of animals that show some intelligence when dealing with certain problems. Although each individual performs very simple operations, the group itself can perform complex operations. For example, this type of behavior is observed when a group of birds or fish moves.

Swarm-based methods define a group of computational algorithms that solve complex problems by imitating these groups of biological individuals. The behavior of different individuals has been imitated over the years: ants [7], particles [8], bacteria [9], frogs [10], fish [11], bees [12], cuckoos [13], bats [14], fireflies [15], wolves [16], etc.

Swarm-based methods share some common features:

- Each individual of the population has associated with some basic behaviors.
- Although each individual only performs simple tasks, the combination of all of their operations allows the swarm to solve complex problems.
- There is not a central control in these groups.
- There is no individual access to the complete status of the swarm.
- It is not possible to do an effective division of the work to be carried out.
- Most of these methods were initially designed to solve an optimization problem. Therefore, there is an objective function associated with the problem to be solved and the solution method attempts to minimize (or maximize) that function.
- The solution method combines exploration and exploitation. The method not only exploits the information already known related to solutions previously analyzed, but also explores new solutions.
- Local and global searches are used to find a solution. The local search analyzes new solutions close to a given one, while the global search analyzes new solutions distant from that solution given.

Several researchers have defined memetic algorithms by combining a swarm-based algorithm with another method. In this case, the swarm-based method performs the global search in the solution space and the other method applies local search to improve the solution obtained by the hybrid method. Some interesting methods of this type use the particle swarm optimization algorithm [17–19], the artificial bee colony algorithm [20–22] and the firefly algorithm [23–25].

4 Swarm-Based Methods Applied to Color Quantization

This section describes several recent color quantization methods based on swarm intelligence. The methods that have been published including more details and results are described here more extensively, while the others are listed in a table.

Some of the solutions evaluated use the *K*-means method. This is a well-known clustering method that defines *k* clusters of similar items. The algorithm begins considering *k* initial values called centroids, which can be selected randomly or by other techniques. After this, each item of the set is associated with the nearest centroid. Then, this algorithm calculates the average value of every cluster and these values are used as the new centroids. This process is applied iteratively during a preset number of iterations or until a predefined error is reached [26].

4.1 Artificial Ants

Several ant-based algorithms have been proposed that mimic different behaviors of natural ants. The ant-based methods used to solve the color quantization problem mimic two of these behaviors. This section first describes the solutions that build a tree of ants to quantize the image and then considers other methods in which the ants move on a grid to form groups of similar pixels.

4.1.1 The Ant-Tree for Color Quantization Method

Pérez-Delgado [27] describes a color quantization method called Ant-tree for color quantization (ATCQ), which adapts the Ant-tree algorithm to reduce the colors used in an image. The Ant-tree algorithm mimics the self-assembly behavior of some species of biological ants and applies it as a clustering technique [28].

The pixels of the original image are represented by ants that build a tree structure. The pixel $\boldsymbol{p}_i$ is represented by the ant h_i, with $1 \leq i \leq N$. Before applying the operations of the algorithm, the tree just includes the root node, a_0, where all these ants wait to move through the tree until they become connected to it. The children of a_0 are included in the second level of the tree as the operations progress. The number of children, q, is initially equal to 0 and it can increase to a predefined maximum, $Q_{\max}$. When the algorithm ends, q will be the size of the quantized palette defined by the tree of ants. Every child S_c of a_0, with $1 \leq c \leq q$, is likewise the root of a subtree of ants and has three variables attached to it: nc_c, $\boldsymbol{sum}_c$ and D_c. The variable nc_c represents the number of ants connected to the subtree until this moment and $\boldsymbol{sum}_c$ is the sum of the colors of all those ants. The color associated with S_c is $\boldsymbol{sum}_c/nc_c$. On the other hand, D_c is computed by Eq. 1, where the value d_{ic} represents the similarity between the ant h_i and the color of the node S_c when this ant was associated with the subtree whose root is S_c, d_{ic} = Sim (h_i, $\boldsymbol{sum}_c/nc_c$).

$$D_c = \sum_{h_i \in c} d_{ic} \tag{1}$$

The following box summarized the operations of the ATCQ method:

```
ATCQ algorithm
while there are moving ants
  Take a moving ant, $h_i$, which is on a node denoted $a_{pos}$
  if $h_i$ is on the root of the tree
    Apply operations to process an ant placed on the root of the tree
  else
    Apply operations to process an ant not placed on the root of the tree
end-while
Generate the quantized image
```

The algorithm applies an iterative process that allows all the ants to be connected as nodes of the tree. Before applying this process, the ants are sorted by increasing average similarity, because the authors of the Ant-tree method suggested that better results are obtained with sorted data. At the beginning of the algorithm, the ants define a sorted list that is on the root of the tree. As the operations progress, the ants go down on the nodes of that structure until they connect to the tree, becoming new nodes.

Each iteration of the algorithm takes an ant h_i which is on the node a_{pos} of the tree. h_i is considered a moving ant since it is not connected to the structure. The algorithm executes different operations depending on the node where the ant is.

- If the ant is on the root of the tree, two different situations can happen.
  - In case the second level of the tree is empty, it is created the first node of this level and the ant connects to it.
  - In other case, the node S_c of the second level most similar to the ant is determined. Then, the similarity between the ant and the color of said node is compared to a threshold computed by Eq. 2, where $\alpha \in (0, 1]$. If the ant and the color of S_c are sufficiently similar (if $d_{ic} \geq T$), h_i moves to the root of the subtree c; in other case, it must be included in a new subtree. In the special case in which the tree already includes $Q_{\max}$ nodes in the second level, no more subtrees can be created, so that the ant moves to the subtree c.

$$T = \frac{D_c}{nc_c}\alpha \tag{2}$$

When a node S_j of the second level of the tree is created, the algorithm initializes the variables associated with that node: nc_j and D_j are set to 1 and 0, respectively, and $\boldsymbol{sum}_j$ takes the color of h_i.

When an ant moves on the root of an existing subtree c, the information related to the root of said subtree is updated: The color of the ant h_i is added to $\boldsymbol{sum}_c$, d_{ic} is added to D_c and nc_c increases by one.

- If the ant is not on the root of the tree, there are three different possible situations:
  - When a_{pos} has no child, h_i becomes the first one.
  - When a_{pos} has exactly two children and the second one has never been disconnected from the structure, this child and all its descendants are disconnected and return to the root of the general tree. Next, h_i becomes the new second child of a_{pos}.
  - In all other cases, a child of a_{pos}, denoted $a+$, is chosen to determine if h_i connects to the tree or moves down into the structure. If $a+$ and h_i are not similar enough ($\text{Sim}(h_i, a+) < T$) and a_{pos} can include at least one more child, h_i becomes a new child of a_{pos}; in other case, h_i moves on $a+$.

The process concludes when all ants have been linked to the structure. At this moment, the quantized palette generated by the ants includes q colors, defined by the children of a_0 ($\boldsymbol{sum}_1/nc_1, \ldots, \boldsymbol{sum}_q/nc_q$). To define the quantized image, the pixels of the original image represented by all the ants included in the subtree c are replaced in the new image by the color $\boldsymbol{sum}_c/nc_c$.

4.1.2 The Iterative ATCQ

Based on the ATCQ algorithm, [29] describes another color quantization method, called ITATCQ.

In essence, ITATCQ applies iteratively the operations of ATCQ. The first iteration of ITATCQ performs the same operations as ATCQ and builds a tree. The following iterations use the tree defined in the previous iterations. Before performing a new iteration, all the ants are disconnected from the structure and only the root node and its children remain in the tree. Then, the ATCQ operations are applied again to connect all the ants to the tree.

The main differences between both methods, ATCQ and ITATCQ, are these:

- ITATCQ performs a preset number of iterations, each of which defines a quantized palette that allows obtaining a quantized image. As iterations progress, the quantized image improves.
- ITATCQ does not consider sorted data. This algorithm does not use pixels sorted by average similarity. On the contrary, the pixels are processed in the same order they are read from the original image, that is, from left to right and from top to bottom. This feature reduces the execution time of the quantization process.
- To accelerate the process, the disconnection of ants performed during ATCQ operations is eliminated. In this case, each iteration of ITATCQ processes each

ant once, and this reduces the execution time. When this disconnection process is eliminated, it can be used a tree with just three levels connecting the ants in the third level of the structure.

Next box shows the steps of the ITATCQ algorithm:

```
ITATCQ algorithm
for $t$=1 to $TMAX$
  while there are moving ants
    Take a moving ant, $h_i$, which is on a node denoted $a_{pos}$
    if $h_i$ is on the root of the tree
      Apply operations to process an ant placed on the root node of the tree
    else
      Connect $h_i$ to $a_{pos}$
    end-if
  end-while
  if $t<TMAX$
    Move all the ants back to the node $a_0$
  end-if
end-for
Generate the quantized image
```

Thus, ITATCQ can obtain better images than ATCQ with few iterations. Figure 1 shows an image with 74,738 colors and the quantized image with 256 colors that is obtained when applying ATCQ and ITATCQ.

4.1.3 ATCQ Combined with Binary Splitting

Pérez-Delgado and Román [30] proposed a solution that combines the ATCQ and the binary splitting algorithms. The second algorithm is a fast color quantization method that generates good images [6]. The following box details the operations of the resulting method:

```
ATCQ algorithm with Binary Splitting
Apply Binary Splitting
Define the nodes in the second level of the tree of ants
Apply ATCQ to connect the ants
Generate the quantized image
```

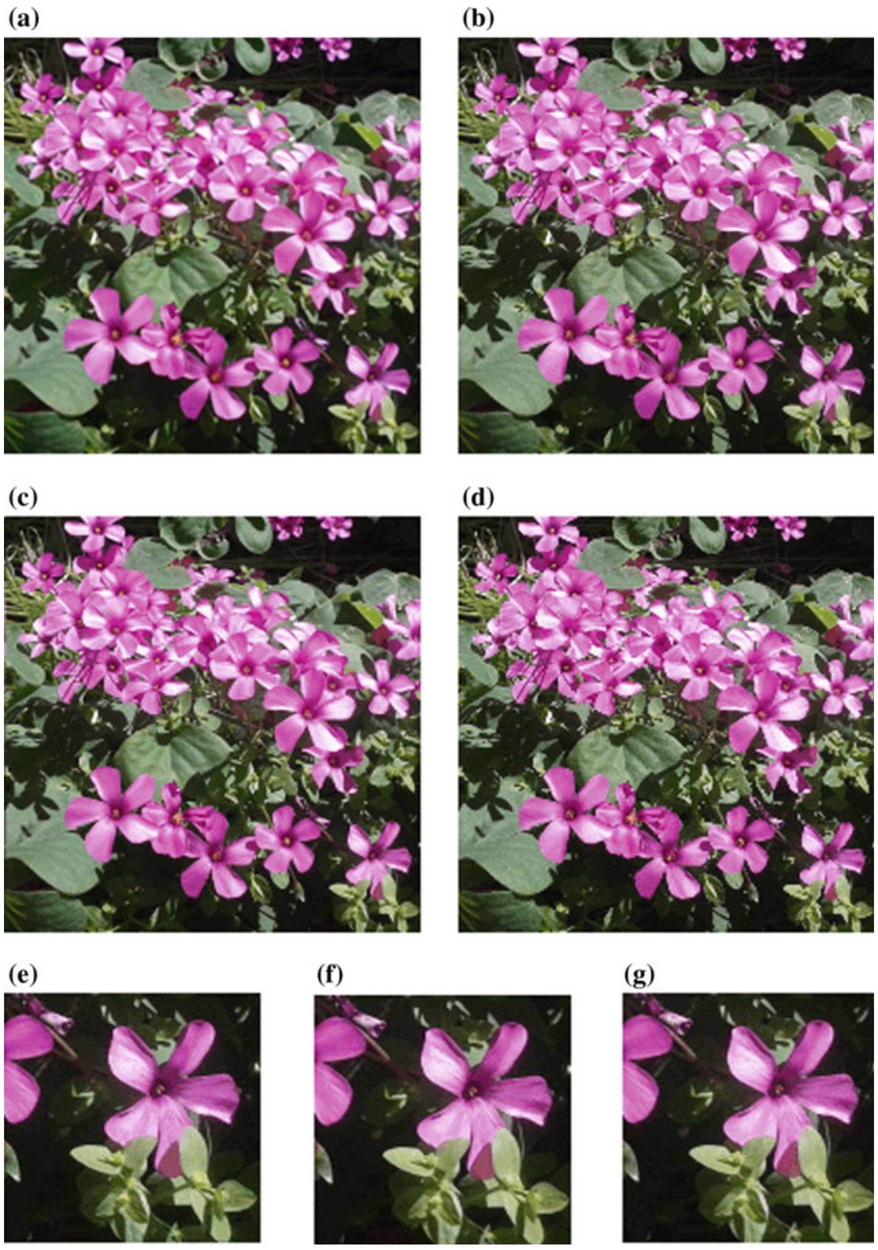

Fig. 1 Original image of clover with 74,738 different colors (**a**), quantized images with 256 colors obtained by ATCQ (**b**), ITATCQ at iteration 10 (**c**) and ATCQ combined with binary splitting (**d**). Subfigures **e**, **f** and **g** show the detail of the lower right corner of **b**, **c** and **d**, respectively

First, the binary splitting method is applied to define an initial color palette with q elements. This method creates a binary tree with q leaves. A leaf l_k, with $1 \leq k \leq q$, represents a subset Y_k with N_k pixels of the original image. This leaf defines a color of the quantized palette by $\boldsymbol{M}_k/N_k$, where $\boldsymbol{M}_k$ is the sum of the colors of the subset Y_k. The color of this leaf is used to represent all the pixels of the subset Y_k in the quantized image. More details about the binary splitting method are not included here, as it is beyond the scope of this chapter; the reader should consult [6] for a complete description.

The leaves of the tree defined by the binary splitting method are used to define the initial nodes of the second level of the tree which is used by ATCQ. Each leaf of the first tree will serve to set initial values for a node of the second tree. In this way, the three variables related to the node S_j of the second tree are set from the values of the leaf l_j of the first tree, with $1 \leq j \leq q$. With this purpose, the variables are initialized as follows: $nc_j = N_j$, $\boldsymbol{sum}_j = \boldsymbol{M}_j$ and D_j is defined by Eq. 3.

$$D_j = \sum_{\mathbf{p}_i \in Y_j} \mathrm{Sim}\left(\mathbf{p}_i, \frac{\mathbf{M}_j}{N_j}\right) \tag{3}$$

When all these nodes have been included in the tree, the ATCQ operations are applied in order to connect all the ants. When the connection process ends, the quantized image is defined in the same way as described for the ATCQ algorithm. Figure 1 shows the clover image reduced to 256 colors by this method.

An advantage of this proposal is that it allows generating good quality quantized images faster than other methods. The results published in [30] show that the final image obtained by the mixed method is always better than the image obtained by binary splitting and ATCQ when applied independently.

4.1.4 Other Color Quantization Methods that Use Artificial Ants

Table 1 shows other ant-based color quantization methods proposed in the literature.

Table 1 Other color quantization proposals based on the use of artificial ants

References	Comment
Ghanbarian et al. [31]	This solution uses the ant-based clustering algorithm described by Handl and Meyer [32] that allows creating topic maps In this case, a set of ants moves over the image. Each ant picks up a pixel, moves to another place with similar pixels and drops the pixel. This behavior of the ants allows defining groups of similar pixels
Hu et al. [33]	They apply the ant-based clustering method described by Lumer and Faieta [34], which later inspired the proposal of Handl and Meyer [32]. In this case, the ants move on a grid where the pixels of the image are randomly distributed before the process begins This article does not include numerical results. There is another article by the same authors that is not cited here because its content is very similar

4.2 *Particle Swarm Optimization*

Omram et al. [35] used the particle swarm optimization (PSO) algorithm to reduce the colors of an image and applied K-means to improve the solutions represented by the particles.

This method uses a population of M particles. The particle i, with $1 \leq i \leq M$, has a position $\boldsymbol{C}_i$ and a velocity $\boldsymbol{v}_i$, which are modified during the iterations of the algorithm. The position defines a quantized palette with q RGB colors: $\boldsymbol{C}_i = (\boldsymbol{c}_{i1}, \ldots, \boldsymbol{c}_{iq})$. The velocity, $\boldsymbol{v}_i = (\boldsymbol{v}_{i1}, \ldots, \boldsymbol{v}_{iq})$, also includes q elements, each of them with three components that take values in an interval $[v_{\min}, v_{\max}]$. During the iterations of the algorithm, the best position found by the particle is stored as its personal best position, $\boldsymbol{b}_i$. In addition, the best position found by the population defines the global best position, $\boldsymbol{g}$.

The objective function associated with the problem is used to measure the quality or fitness of a quantized palette. The function selected in this case is the mean squared error (MSE), given by Eq. 4.

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N} \|\boldsymbol{p}_k - \boldsymbol{P}_k\|^2 \tag{4}$$

The following box shows the main operations of the algorithm:

```
PSO algorithm with $K$-means applied to color quantization
Initialize the population
for $t$ = 1 to TMAX
    Apply $K$-means to each particle in a probabilistic way
    Compute the fitness of each particle
    Update the personal best position of each particle
    Update the best solution of the population
    Update the velocity of each particle by Eq. 5
    Update the position of each particle by Eq. 6
end-for
Generate the quantized image
```

It can be observed that two new operations are added to the general scheme of the PSO algorithm: the first operation of the loop and the last operation of the algorithm.

The first operation of the algorithm defines the initial population of particles. With this purpose, the initial position of a particle includes q pixels randomly selected from the original image and its initial velocity takes random values in [$v_{\min}$, $v_{\max}$].

After the initialization stage, the algorithm applies TMAX iterations, where TMAX is a predefined number. The first operation of every iteration applies the K-means algorithm to each particle in a probabilistic way, in order to improve the position of the particle. K-means is used if a random variable takes a value larger than a predefined limit. In this case, the current position of the particle defines the initial centroids used by K-means, while the final centroids calculated by this method define the new position of the particle.

The second operation of each iteration computes the fitness of each particle. Before computing the fitness of the particle i, it is necessary to determine the value $\boldsymbol{P}_k$ corresponding to each pixel $\boldsymbol{p}_k$ of the initial image. In this case, $\boldsymbol{P}_k$ takes the value of the element $\boldsymbol{c}_{ij}$ of $\boldsymbol{C}_i$ which is closest to $\boldsymbol{p}_k$.

If the new fitness of the particle i is smaller than the one associated with the personal best position of this particle, $\boldsymbol{b}_i$ takes the current value of $\boldsymbol{C}_i$. Moreover, when the personal best position of any particle is better than the global best position, $\boldsymbol{g}$ is updated to store that position.

The next operation updates the velocity of each particle by Eq. 5, where $\boldsymbol{\varepsilon}_1$ and $\boldsymbol{\varepsilon}_2$ take random values in [0, 1], while ω, ϕ_1 and ϕ_2 are non-negative values that determine the relative influence of each addend.

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + \phi_1 \boldsymbol{\varepsilon}_1 [\mathbf{b}_i(t) - \mathbf{C}_i(t)] + \phi_2 \boldsymbol{\varepsilon}_2 [\mathbf{g}(t) - \mathbf{C}_i(t)] \tag{5}$$

The last operation of the iteration updates the position of each particle by Eq. 6.

$$\mathbf{C}_i(t+1) = \mathbf{C}_i(t) + \mathbf{v}_i(t+1) \tag{6}$$

When the iterations of the algorithm conclude, $\boldsymbol{g}$ represents the quantized palette that must be used to generate the new image. To obtain this image, the color of each pixel of the original image is replaced by the closest color of $\boldsymbol{g}$.

Figure 2 shows the results obtained when 10 iterations of this method are applied to the clover image.

To conclude this section, Table 2 gives a list of some other articles that apply PSO to reduce the colors of an image.

4.3 Artificial Bees

This section describes two proposals that use the artificial bee colony (ABC) algorithm [12].

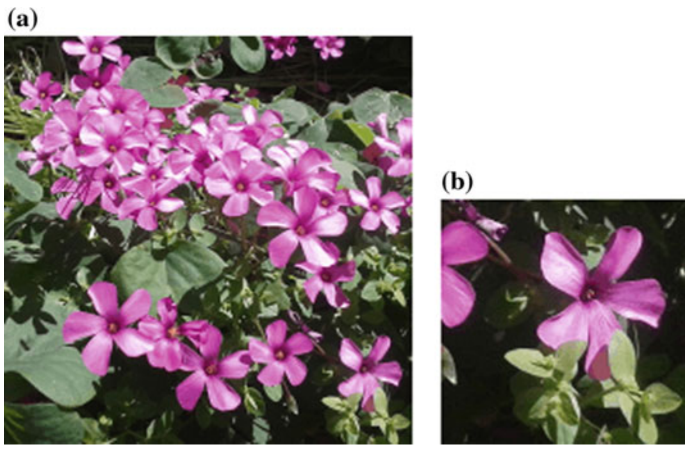

Fig. 2 Clover image quantized to 256 colors by PSO combined with K-means (**a**) and lower right corner of this image (**b**)

Table 2 Other color quantization proposals based on the PSO algorithm

References	Comment
Wang et al. [36]	They use PSO without considering K-means. The coefficient ω of Eq. 5 is computed based on the number of iterations of the algorithm and a predefined range for the parameter The objective function is the MSE, which is computed based on the color-levels histogram
Alamdar and Haratizadeh [37]	It combines the operations of PSO, genetic algorithms and a well-known clustering method called fuzzy c-means The objective function is the one associated with fuzzy c-means
Kaur et al. [38]	This solution uses the LAB color space. The objective function computes the difference between two colors based on the chroma, lighting and hue perceptual difference
Barman et al. [39]	This proposal is similar to that of Omran et al. [35], but it uses new coefficients that replace ϕ_1, ε_1, ϕ_2 and ε_2 in Eq. 5
Kubota et al. [40]	It combines PSO with a vector error diffusion method and uses MSE as the objective function

4.3.1 Artificial Bee Colony Combined with *K*-Means

Ozturk et al. [41] proposed a solution that applies the ABC algorithm together with the K-means algorithm. This method applies K-means in a similar way to the proposal of [35] and also considers the same objective function.

In this case, it is considered a set of B food sources that represent solutions to the problem. Each food source $\boldsymbol{C}_i$ represents a quantized palette with q elements, $\boldsymbol{C}_i = (\boldsymbol{c}_{i1}, \ldots, \boldsymbol{c}_{iq})$, where $\boldsymbol{c}_{ik}$ is an RGB color, $1 \leq i \leq B$ and $1 \leq k \leq q$. The fitness or amount of nectar of this source represents the quality of the palette and is computed by Eq. 4. The algorithm uses a set of bees that work in the solution space to select the best palette. A limit is associated with each food source to determine when that source is considered exhausted. This variable takes the value 0 when a new food source is selected and increases as the algorithm proceeds; when the variable reaches a predefined value *L*, the source is abandoned by the bees.

The solution defined by Ozturk et al. applies the basic steps of the ABC algorithm and includes the probabilistic application of *K*-means to improve the food sources:

```
ABC algorithm with K-means applied to color quantization
Initialize the food sources
for $t$ = 1 to TMAX
    Apply employed bees (with probabilistic application of K-means)
    Apply onlooker bees (with probabilistic application of K-means)
    Apply scout bees
    Select the best solution so far
end-for
Generate the quantized image
```

The population of bees includes three types of bees according to the different tasks these perform. Each employed bee exploits a food source of the current set of sources considered by the population. The onlooker bees take into account the information provided by the employed bees in order to further exploit a certain food source of the current set. The scout bees are in charge of finding new food sources.

To define the initial set of food sources (initial palettes), random pixels are taken from the original image to define the *q* elements of each source. After this, the operations associated with each group of bees are applied iteratively.

The first step applies the operations corresponding to the employed bees. Each of the *B* food sources is exploited by an employed bee. The employed bee that is exploiting the food source $\boldsymbol{C}_i$ selects a candidate food source $\boldsymbol{C'}_i$ according to Eq. 7, where $\phi_{ir} \in [-1, 1]$ is a random value, *r* is a random component of the food sources and $\boldsymbol{C}_k$ is a random food source different from $\boldsymbol{C}_i$. If the fitness of the candidate source is worse than that of $\boldsymbol{C}_i$, the candidate source replaces $\boldsymbol{C}_i$; in other case, the limit of $\boldsymbol{C}_i$ increases by one.

$$\mathbf{C}'_{ir} = \mathbf{C}_{ir} + \phi_{ir}(\mathbf{C}_{ir} - \mathbf{C}_{kr}) \tag{7}$$

The second step applies the operations of the onlooker bees. Such operations are similar to those of the employed bees, although the onlooker bees are not linked to a specific food source. For this reason, the first operation selects a food source from the current set. An onlooker bee chooses the source $\boldsymbol{C}_i$ with a probability defined by Eq. 8, where fit_i represents the fitness of the food source *i*, with $1 \leq i \leq B$.

$$\text{probability}_i = \frac{\text{fit}_i}{\sum_{j=1}^{B} \text{fit}_j} \tag{8}$$

The third step applies the operations of the scout bees. These operations consist of replacing the food sources whose limit has reached the value *L*. Said sources are replaced by new sources defined in the same way described for the initialization step of the algorithm.

Once the operations of the three types of bees have been performed, the source with the lowest fitness found by the bees represents the best quantized palette.

When the iterative process concludes, the quantized image is obtained by applying the same operations described for [35] in Sect. 4.2.

To calculate the fitness of a food source by Eq. 4, the operations are the same as those described for the method proposed in [35]. However, when considering a source currently exploited by a bee, a random variable is used to decide whether *K*-means should be applied to the source before computing its fitness. If *K*-means is applied, the current position of the source defines the initial centroids used by this method, while the final centroids define the new position of the source.

4.3.2 Artificial Bee Colony Combined with ATCQ

Pérez-Delgado [42] also applied the ABC algorithm, but the *K*-means method was discarded because it consumes a lot of time.

The following box summarizes the steps of the proposed method, which includes the basic operations of the ABC algorithm but uses the ATCQ algorithm to calculate the fitness of the current food sources:

ABC algorithm with ATCQ applied to color quantization

```
Initialize the food sources
for $t$ = 1 to TMAX
    Apply employed bees (ATCQ is applied to the current food sources)
    Apply onlooker bees (ATCQ is applied to the current food sources)
    Apply scout bees
    Select the best solution so far and store the corresponding tree
end-for
Generate the quantized image
```

The initialization step defines the initial position of each particle as the proposal described in Sect. 4.3.1, but sets the initial velocity to 0.

This proposal applies two different methods to calculate the fitness of a food source, depending on whether the source belongs to the current set exploited by the bees or if it is a candidate source. To calculate the fitness of a candidate source, the same operations described in Sect. 4.3.1 are applied (excluding the application of K-means). On the contrary, to calculate the fitness of a source $\boldsymbol{C}_i$ of the current set, the ATCQ algorithm is applied. With this purpose, q nodes are included in the second level of the tree and each element of the food source $\boldsymbol{C}_{\boldsymbol{i}}$ is used to initialize a node. The initial values of the node S_j of the second level, with $1 \leq j \leq q$, are set as follows: $\boldsymbol{sum}_j = \boldsymbol{c}_{ij}$, $nc_j = 1$, $D_j = 0$. Once all these nodes have been created, the ATCQ operations to build the tree of ants are carried out. Then, the colors of the nodes of the second level of the tree are used to update each element of the position of the source: $\boldsymbol{c}_{ij} = \boldsymbol{sum}_j/nc_j$, with $1 \leq j \leq q$. Finally, the information of the tree is used to compute the fitness of the source, taking into account that the value $\boldsymbol{sum}_j/nc_j$ represents the element $\boldsymbol{P}_{\boldsymbol{k}}$ of the Eq. 4 for all the pixels of the initial image represented by the ants that belong to the subtree j.

This solution stores the tree corresponding to the best palette found during the iterations, since said tree allows to generate the new image at the end of the operations. The same operations described for the ATCQ algorithm are used to obtain this image.

When both bee-based solutions described in this chapter are compared, they can obtain quantized images of similar quality (Fig. 3), but the solution proposed by Pérez-Delgado consumes much less time (see Sect. 4.6). The proposal of Pérez-Delgado is faster because it leverages the tree of ants to calculate the fitness of the sources and discards the use of K-means, which is slower than ATCQ.

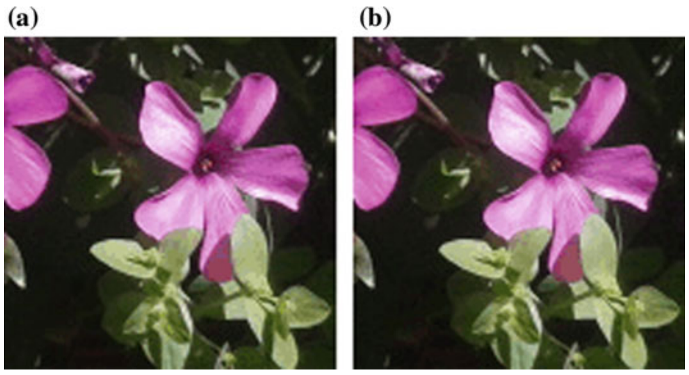

Fig. 3 Detail of the clover image reduced to 256 colors by ABC combined with K-means (**a**) and ABC combined with ATCQ (**b**)

4.4 Firefly Algorithm

Pérez-Delgado [43] combined the operations of the firefly algorithm [15] with those of the ATCQ algorithm to define a new color quantization method. This method uses a population of F fireflies and each firefly has a tree of ants associated with it. Three elements define the firefly i: the position ($\boldsymbol{C}_i$), the brightness (L_i) and the tree of ants (tree$_i$). The position represents a color palette, $\boldsymbol{C}_i = (\boldsymbol{C}_{i1}, \ldots, \boldsymbol{C}_{iqi})$, where qi is the number of colors of this palette, $\boldsymbol{C}_{ik}$ is an RGB color and $1 \leq i \leq F$. The brightness of this position represents the quality or fitness of the quantized palette. The tree of ants includes qi nodes in the second level, ($S_{i1}, \ldots, S_{i\ qi}$), with $qi \leq Q_{\max}$. The three values associated with the node S_{ij} are denoted $\boldsymbol{sum}_{ij}$, nc_{ij} and D_{ij}.

During the execution of the algorithm, the fireflies move toward other brighter fireflies, which represent better palettes. In addition, the ATCQ algorithm is applied to a firefly not only to improve its position but also to calculate its brightness. The author of the method proposed two different objective functions (Eqs. 9 and 10), computed based on the information of the tree associated with a firefly.

$$g_1(\text{tree}_i) = \frac{1}{\frac{1}{N}\sum_{j=1}^{qi}\sum_{h_k \in j}\left\| h_k - \frac{\textbf{sum}_{ij}}{nc_{ij}} \right\|^2} \tag{9}$$

$$g_2(\text{tree}_i) = \sum_{j=1}^{qi} D_{ij} \tag{10}$$

The following box enumerates the operations of the algorithm.

```
Firefly algorithm combined with ATCQ for color quantization
Initialize the population
    Determine the best firefly of the initial population
    for $t$ = 1 to TMAX
    Move all the fireflies toward brighter partners according to Eq. 11
    Move the brightest firefly according to Eq. 12
    Apply ATCQ to every firefly and compute their fitness
    Update the best solution
end-for
Generate the quantized image
```

The first operation defines the initial position of the fireflies. To perform this operation, F different values of the α parameter are considered, $\{\alpha_1, \ldots, \alpha_F\}$, with $\alpha_i \in (0,1]$, $1 \leq i \leq F$, and the ATCQ algorithm is applied once with each of these values. The F trees obtained are used to define the initial position and brightness of the F fireflies. When ATCQ is executed with α_i, the tree tree$_i$ is defined, which includes qi nodes in the second level. The colors of these nodes define the initial position of the firefly i: $\boldsymbol{C}_i(0) = (\boldsymbol{sum}_{i1}/nc_{i1}, \ldots, \boldsymbol{sum}_{i\ qi}/nc_{i\ qi})$. In addition, the tree is used to define the brightness of the firefly (by Eqs. 9 or 10).

The second operation of the algorithm selects the brightest firefly as the initial global best firefly, denoted gb, and stores the tree associated with it, tree$_{\text{gb}}$. The best firefly of the population defines the quantized palette that will be used to generate the quantized image.

After this, an iterative process tries to improve the initial solution defined by the population of fireflies. This process applies the basic steps of the firefly algorithm, but also uses the ATCQ algorithm to calculate the brightness of the fireflies.

Each firefly moves toward all the fireflies that represent better solutions than it. The movement toward each one of said fireflies depends on their brightness. If firefly k is brighter than i, Eq. 11 determines the movement of firefly i toward k at iteration t. In this equation, β_0 is the attractiveness at distance 0, γ is the light absorption coefficient, r_{ik} is the distance between fireflies i and k and $\boldsymbol{\varepsilon}_i$ takes random values in [−5, 5].

$$\mathbf{C}_i(t) = \mathbf{C}_i(t-1) + \beta_0 e^{-\gamma\, r_{ik}^2}[\mathbf{C}_k(t-1) - \mathbf{C}_i(t-1)] + \boldsymbol{\varepsilon}_i \tag{11}$$

The brightest firefly moves randomly according to Eq. 12, where $\boldsymbol{\varepsilon}$ takes random values in [−1, 1].

$$\mathbf{C}_{\text{brightest}}(t) = \mathbf{C}_{\text{brightest}}(t-1) + \boldsymbol{\varepsilon} \tag{12}$$

After moving all the fireflies, the ATCQ algorithm is applied again to all of them. It should be noted that the initial application of ATCQ for a firefly considers a tree with just its root node, but the following applications for the same firefly consider a tree also including nodes in its second level. In this case, the current position of firefly i defines the initial value of the nodes in the second level of tree$_i$ and the number of nodes of that level. First, all the ants are disconnected from the tree and then the initial values of each node S_{ij} are set as follows: $\boldsymbol{sum}_{ij} = \boldsymbol{c}_{ij}$, $nc_{ij} = 1$ and $D_{ij} = 0$, with $1 \le j \le qi$. Next, ATCQ is applied to reconnect these ants, and the resulting tree is used to calculate the brightness of the firefly i by Eqs. 9 or 10. To conclude the operation, the colors of the nodes in the second level of the resulting tree are used to update the position of the firefly: $\boldsymbol{c}_{ij} = \boldsymbol{sum}_{ij}/nc_{ij}$, with $1 \le j \le qi$.

If the brightness of any firefly of the current population is better than that of the best solution stored so far, said firefly defines the new best solution and the tree associated with it is stored as the new tree$_{\text{gb}}$.

The last operation of the algorithm uses the tree tree$_{\text{gb}}$ to define the quantized image and the quantized palette, in the same way described for the ATCQ algorithm in Sect. 4.1.1. Figure 4 shows the result obtained by this method when applied to the image included in Fig. 1a.

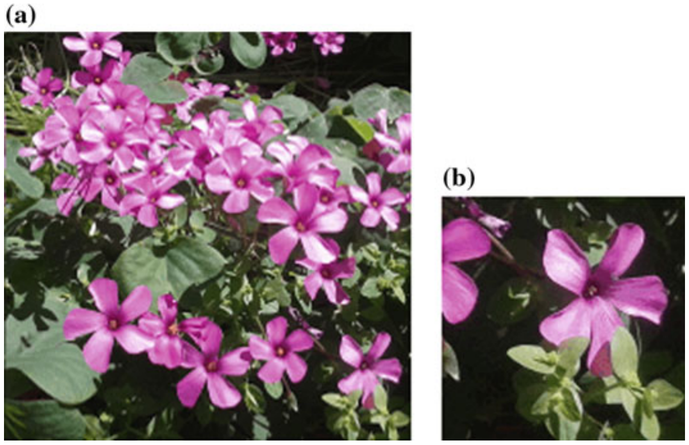

Fig. 4 Clover image reduced to 256 colors by the method that combines the Firefly and ATCQ algorithms (**a**) and detail of the lower right part of the quantized image (**b**)

Table 3 Other color quantization proposals based on the firefly algorithm

Reference	Comment
Jitpakdee et al. [44]	The solution obtained by the firefly algorithm defines the initial centroids used to apply *K*-means

Although the solution described in this section applies the ATCQ method in a similar way to the one described in Sect. 4.3.2 when it is combined with the ABC algorithm, there is a difference. In this case, ATCQ is used to generate the initial population, and this allows obtaining palettes with different number of elements for each firefly. Instead, when ATCQ is combined with ABC, all the palettes have the same size, since they are defined before applying ATCQ. It must be taken into account that the value of the α parameter of the ATCQ algorithm influences the size of the palette obtained by this method. This means that the limit $Q_{\max}$ defined for the palette size is not reached when small values of α are considered.

To conclude this section, Table 3 describes another solution for color quantization based on the firefly algorithm.

4.5 *The Shuffled Frog-Leaping Method*

The shuffled frog-leaping algorithm was proposed to solve optimization problems [10]. Certainly, the authors of the method describe it as a memetic meta-heuristic in [45]. Pérez-Delgado [46] presents a proposal that applies said method to reduce the colors of an image. The main steps of this proposal are described in the following box:

```
Shuffled frog-leaping algorithm for color quantization
Initialize the population
for $t$ = 1 to TMAX
    Compute the fitness of the frogs
    Create the memeplexes
    Improve each memeplex
    Shuffle the frogs of the memeplexes
    Improve the best frog
end-for
Generate the quantized image
```

The population of frogs includes R individuals. The frog i has a position $\boldsymbol{C}_i$ that represents a palette with q RGB colors, $\boldsymbol{C}_i = (\boldsymbol{c}_{i1}, \ldots, \boldsymbol{c}_{iq})$, where $1 \leq i \leq R$. The fitness of this position, which determines its quality, is calculated by the MSE (Eq. 4).

The operations performed to calculate the fitness of a frog r are the following. First, each pixel $\boldsymbol{p}_k$ of the initial image is associated with the element j of $\boldsymbol{C_r}$ closest to it, with $1 \leq j \leq q$. This operation divides the pixels into q groups. Second, the average color of each group is computed. Next, the color of group j is used to define the new value of $\boldsymbol{c}_{rj}$. Last, the MSE value is calculated using Eq. 4, where $\boldsymbol{P}_k$ is replaced by the element of $\boldsymbol{C}_r$ associated with $\boldsymbol{p}_k$ in the first operation.

To initialize the population, q pixels randomly selected from the original image define the initial position of each frog. After this, the iterative process applied by this algorithm separates the frogs into independent groups called memeplexes, improves the solution defined by each memeplex and, after that, regroups all the frogs.

Each iteration begins by calculating the fitness of all the frogs. Then, the frogs are sorted by decreasing fitness and divided into groups. The next operation processes each memeplex independently. The local search applied to a memeplex attempts to improve the frogs of said memeplex.

Let w denote the worst frog in a memeplex and $\boldsymbol{C}_w$ its position. A candidate position $\mathbf{C}'_w$ is computed for this frog by Eqs. 13 and 14, where $\boldsymbol{C}_b$ is the position of the best frog in the memeplex and ρ is a random value between 0 and 1. The result of Eq. 14 is limited to the interval [−DMAX, DMAX], where DMAX is a predefined constant.

$$\mathbf{C}'_w = \mathbf{C}_w + \mathbf{D} \tag{13}$$

$$\mathbf{D} = \rho(\mathbf{C}_b - \mathbf{C}_w) \tag{14}$$

If the fitness of $\mathbf{C}'_w$ is better than that of $\boldsymbol{C}_w$, the candidate position replaces $\boldsymbol{C}_w$; in other case, a second candidate is computed by Eqs. 13 and 15, where $\boldsymbol{C}_g$ represents the current position of the best frog in the population and the result of

Eq. 15 is also limited to the interval [−DMAX, DMAX]. If the second candidate improves the current position, it becomes the new current position. Otherwise, the worst frog is assigned a new position selected in the same way as when the initial population is defined.

$$\mathbf{D} = \rho(\mathbf{C}_g - \mathbf{C}_w) \tag{15}$$

The operations described for each memeplex are applied a preset number of iterations. When the processing of the memeplexes concludes, all the frogs are gathered in a single group (shuffle operation).

The algorithm adapted to reduce the colors of an image adds an operation to the original algorithm, whose objective is to improve the best frog in the population. With this purpose, the fitness of said frog is recalculated before concluding each iteration. This operation updates the position of the frog and allows the algorithm to obtain better results.

The quantized palette corresponding to the solution of the problem is that represented by the position of the best frog found by the algorithm, $\boldsymbol{C}_g$.

To accelerate the process, this method can work on a reduced set of pixels of the original image. Those pixels are sampled and the reduced set of pixels is used to calculate the fitness of the frogs, which speeds up this operation. In this case, the complete set of pixels is only used to define random positions and to generate the quantized image. For this reason, the colors of the best palette defined by the frogs are recomputed before generating the quantized image, so that they better represent the colors of the initial image. The colors are recomputed by applying the same operations described in the third paragraph of this section (excluding the computation of the MSE value). Said operations not only allow updating the value of $\boldsymbol{C}_g$, but also determine the color of the new palette that will be used to represent each pixel of the quantized image.

Figure 5 shows the lower right part of clover image reduced to 256 colors with this method by using different sampling factors. The first case considers all the pixels of the original image (160,000 pixels), while the other cases correspond to samples with 16,000, 1600 and 160 pixels, respectively.

4.6 A Brief Comparative Among Color Quantization Methods

The clover image has been used in the previous subsections to illustrate the results of the methods described in this chapter. Table 4 shows the MSE of the quantized image obtained by each method, together with the execution time. The table also shows the results generated by *K*-means and several color quantization methods not based on swarms (the article that introduced each of these methods is cited). *K*-means and Neuquant are clustering-based methods, while the others are splitting methods.

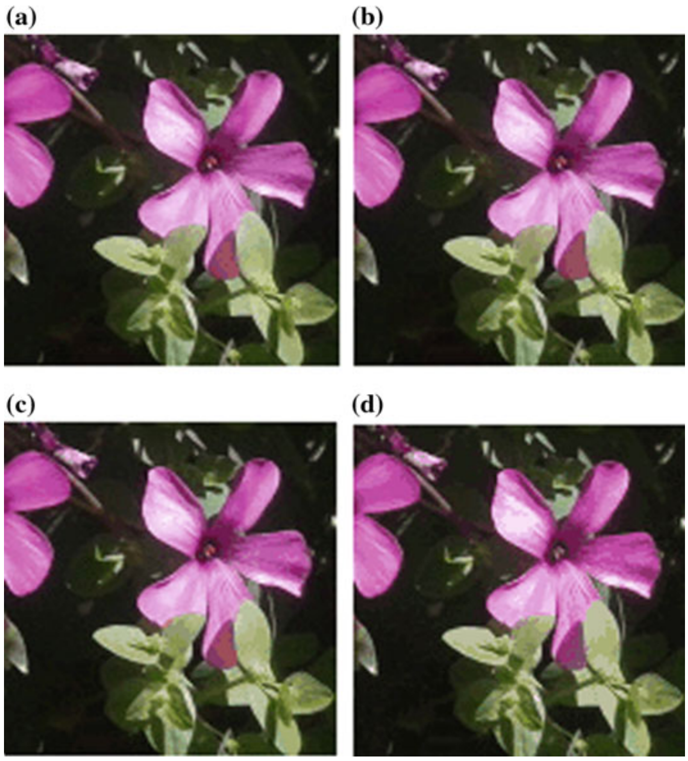

Fig. 5 Detail of the clover image reduced to 256 colors by the shuffled frog-leaping algorithm considering different sets of pixels to apply the algorithm: **a** the image is not sampled and 160,000 pixels are used, **b** sample with 16,000 pixels, **c** sample with 1600 pixels, **d** sample with 160 pixels

Table 4 Numerical results of the application of several methods to reduce the clover image to 256 colors [Time: execution time (milliseconds)]

Method	MSE	Time	Method	MSE	Time
Median-cut [2]	278.91	10	ATCQ	59.91	173
Variance-based [4]	91.09	692	Iterative ATCQ	51.39	1960
Octree [3]	69.32	236	ATCQ with binary splitting	50.08	323
Wu's method [5]	61.43	39	PSO with *K*-means	50.61	226,635
Binary splitting [6]	56.25	22	ABC with *K*-means	51.36	324,101
Neuquant [47]	60.83	316	ABC with ATCQ	49.46	33,803
K-means	58.65	7069	Firefly with ATCQ	51.73	10,463
			Shuffled frog-leaping algorithm	51.02	36,603

The iterative methods have been executed during 10 iterations, except *K*-means, for which only 5 iterations have been applied, since it consumes a lot of time. In addition, 6 particles, frogs and fireflies have been used and the same number of food sources has been considered to apply the ABC algorithm.

The frogs-based solution was applied to all the pixels of the initial image. Nevertheless, if the image is sampled, the execution time is considerably reduced and the error increases a bit. For example, when a sample with 16,000 pixels is used, the error obtained is 55.80 and the algorithm takes 3747 ms.

The results indicate that the swarm-based methods can generate images with less error than other methods, although in general they consume more time.

Table 5 Other proposals for color quantization based on swarm methods

Swarm method	References	Comment
Cuckoo search	[48]	They combine cuckoo search and *K*-means methods. The CMC distance is used as objective function (this distance is computed from the chroma, lighting and hue perceptual difference between two colors)
Artificial fish swarm	[49]	They adapt some parameters of the original algorithm together with some behaviors to apply it for color reduction
	[50]	This proposal combines the artificial fish swarm algorithm with the fuzzy c-means method
Bacterial foraging optimization	[51]	This solution applies the algorithm to images represented in the CIE Luv color space and uses the CMC distance as the fitness function
	[52]	They use the LAB color space and consider the CMC distance as the objective function
	[53]	It applies an initial operation for indexing the image before applying Bacterial foraging optimization. The histogram of the original image is used to generate an indexed image which is used to select only the most frequent colors
	[54]	They consider the RGB color space and use the CMC distance as the fitness function
	[55]	They consider the HSI color space and use the Euclidean distance as the fitness function
	[56]	This article applies the Bacterial foraging optimization method to images stored using different formats (PNG, BMP, JPEG)
Grey wolf optimization	[57]	The algorithm is applied to images represented in the RGB color space and the MSE is used as objective function A penalty term is included in the algorithm to avoid the creation of colors in the quantized palette that are not used to represent any pixel

4.7 *Other Swarm-Based Methods*

Table 5 summarizes other color quantization methods inspired by swarms.

5 Conclusions

Swarm-based methods define a new approach to solve difficult problems. This chapter describes several proposals to solve the color quantization problem that use swarms. The iterative nature of these methods makes it possible to define an initial solution with few iterations, although a better image can be obtained if the number of iterations increases. Therefore, the more time the algorithm uses to find a solution, the better the quality of the quantized image.

References

1. Garey, M., Johnson, D., Witsenhausen, H: The complexity of the generalized Lloyd-max problem (corresp.). IEEE Trans. Inf. Theory **28**(2), 255–256 (1982)
2. Heckbert, P.: Color image quantization for frame buffer display. In: Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques, New York, USA. ACM, pp. 297–307 (1982). https://doi.org/10.1145/800064.801294
3. Gervautz, M., Purgathofer, W.: A simple method for color quantization: octree quantization. Graphic Gems, pp. 287–293. Academic Press, New York (1990)
4. Wan, S.J., Prusinkiewicz, P., Wong, S.K.M.: Variance-based color image quantization for frame buffer display. Color Res Appl **15**(1), 52–58 (1990)
5. Wu, X.: Efficient statistical computations for optimal color quantization. In: Arvo, J. (ed.) Graphics Gems, vol II. Morgan Kaufmann, pp. 126–133 (1991). https://doi.org/10.1016/B978-0-08-050754-5.50035-9
6. Orchard, M.T., Bouman, C.A.: Color quantization of images. IEEE Trans. Signal Process. **39** (12), 2677–2690 (1991). https://doi.org/10.1109/78.107417
7. Mullen, R.J., Monekosso, D., Barman, S., Remagnino, P.: A review of ant algorithms. Expert Syst. Appl. **36**(6), 9608–9617 (2009). https://doi.org/10.1016/j.eswa.2009.01.020
8. Kenedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference of Neural Networks, vol 4, pp 1942–1948 (1995). IEEE. https://doi.org/10.1109/ICNN.1995.488968
9. Das, S., Biswas, A., Dasgupta, S., Abraham, A.: Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. In: Foundations of Computational Intelligence. Studies in Computational Intelligence, vol. 3. Springer, Berlin, p 23–55 (2009)
10. Eusuff, M.M., Lansey, K.: Optimization of water distribution network design using the shuffled frog leaping algorithm. J. Water Resour. Plan. Manag. **129**(3), 210–225 (2003)
11. Li, X.L., Lu, F., Tian, G.H., Qian, J.X.: Applications of artificial fish school algorithm in combinatorial optimization problems. J. Shandong Univ. (Eng. Sci.) **5**, 15 (2004)
12. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report TR-06. Erciyes University, Engineering Faculty, Computer Engineering Department (2005)

13. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of the 2009 World Congress on Nature and Biologically Inspired Computing, pp. 210–214. IEEE (2009)
14. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization. Studies in Computational Intelligence, vol. 284. Springer, Berlin, pp. 65–74 (2010). https://doi.org/10.1007/978-3-642-12538-6_6
15. Yang, X.S., He, X.: Firefly algorithm: recent advances and applications. Int. J. Swarm Intell. **1**, 36–50 (2013). https://doi.org/10.1504/IJSI.2013.055801
16. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
17. Liu, D., Tan, K.C., Goh, C.K., Ho, W.K.: A multiobjective memetic algorithm based on particle swarm optimization. IEEE Trans. Syst. Man Cybern. Part B **37**(1), 42–50 (2007)
18. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Ann. Oper. Res. **156**(1), 99–127 (2007). https://doi.org/10.1007/s10479-007-0224-y
19. Wang, H., Moon, I., Yang, S., Wang, D.: A memetic particle swarm optimization algorithm for multimodal optimization problems. Inf. Sci. **197**, 38–52 (2012)
20. Fister, I., Žumer, J.B.: Memetic artificial bee colony algorithm for large-scale global optimization. In: IEEE Congress on Evolutionary Computation, Brisbane, Australia, pp. 1–8, 10–15 June 2012. IEEE (2012)
21. Bansal, J.C., Sharma, H., Arya, K.V., Nagar, A.: Memetic search in artificial bee colony algorithm. Soft Comput. **17**(10), 1911–1928 (2013). https://doi.org/10.1007/s00500-013-1032-8
22. Kumar, S., Sharma, V.K., Kumari, R.: An improved memetic search in artificial bee colony algorithm. Int. J. Comput. Sci. Inform. Technol. **5**(2), 1237–1247 (2014)
23. Fister, I., Yang, X.S., Fister, I., Brest, J.: Memetic firefly algorithm for combinatorial optimization. In: Filipic, B., Silc, J. (eds.) BIOMA 2012: Bioinspired Optimization Methods and Their Applications. Jozef Stefan Institute, Ljubljana, Slovenia (2012). https://arxiv.org/pdf/1204.5165.pdf. Accessed 25 July 2019
24. Fister, I., Yang, X.S., Brest, J.: Memetic self-adaptive firefly algorithm. In: Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M. (eds.) Swarm Intelligence and Bio-inspired Computation, pp. 73–102. Elsevier, Amsterdam (2013)
25. Gálvez, A., Iglesias, A.: New memetic self-adaptive firefly algorithm for continuous optimisation. Int. J. Bio-Inspired Comput. **8**(5), 300–317 (2016). https://doi.org/10.1504/IJBIC.2016.079570
26. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
27. Pérez-Delgado, M.-L.: Colour quantization with Ant-tree. Appl. Soft Comput. **36**, 656–669 (2015). https://doi.org/10.1016/j.asoc.2015.07.048
28. Azzag, H., Monmarche, N., Slimane, M., Venturini, G.: AntTree: a new model for clustering with artificial ants. In: Proceedings of the 2003 Congress on Evolutionary Computation, vol 4, pp. 2642–2647. IEEE (2003) https://doi.org/10.1109/CEC.2003.1299421
29. Pérez-Delgado, M.-L.: An iterative method to improve the results of Ant-tree algorithm applied to color quantisation. Int. J. Bio-inspired Comput. **12**(2), 87–114 (2018). https://doi.org/10.1504/IJBIC.2018.094199
30. Pérez-Delgado, M.-L., Román, J.A.: A two-stage method to improve the quality of quantized images. J. Real-time Image Process. (2018). https://doi.org/10.1007/s11554-018-0814-8
31. Ghanbarian, A.T., Kabir, E., Charkari, N.M.: Color reduction based on ant colony. Pattern Recognit. Lett. **28**(12), 1383–1390 (2007). https://doi.org/10.1016/j.patrec.2007.01.019
32. Handl, J., Meyer, B.: Improved ant-based clustering and sorting in a document retrieval interface. In: Proceeding of the Seventh International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 2439, pp. 913–923. Springer, Berlin (2002)
33. Hu, X., Xiong, N., Cui, S., Hui, W., Wang, J.: A color clustering algorithm for cloth image. In: Proceedings of the IEEE Asia-Pacific Conference on Services Computing, pp. 1500–1505. IEEE (2008). https://doi.org/10.1109/APSCC.2008.78

34. Lumer, E.D., Faieta, B.: Diversity and adaptation in populations of clustering ants. In: Cliff, D., Husbands, P., Meyer, J.A., Wilson, S.W. (eds.) Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3, SAB 94, pp. 501–508. MIT Press (1994)
35. Omran, M., Engelbrecht, A.P., Salman, A.: A color image quantization algorithm based on particle swarm optimization. Inform (Slovenia) **29**(3), 261–269 (2005)
36. Wang, Z., Sun, X., Zhang, D. A swarm intelligence based color image quantization algorithm. In: Proceedings of the 2007 1st International Conference on Bioinformatics and Biomedical Engineering, pp. 592–595. IEEE (2007). https://doi.org/10.1109/ICBBE.2007.155
37. Alamdar, F, Haratizadeh, S.: Color quantization with clustering by F-PSO-GA. In: Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 3, pp. 233–238. IEEE (2010). https://doi.org/10.1109/ICICISYS.2010.5658548
38. Kaur, R., Gupta, S., Sandhu, P.S.: Optimization color quantization in L* A* B* color space using particle swarm optimization. In: Proceedings of the International Conference on Intelligent Computational Systems (ICICS 2011), Bangkok, Thailand, 8–9 July 2011(2011a)
39. Barman, D., Hasnat, A., Sarkar, S., Rahaman, M.A.: Color image quantization using gaussian particle swarm optimization (CIQ-GPSO). In: Proceedings of the 2016 International Conference on Inventive Computation Technologies, Coimbatore, India, 26–27 August 2016 (2016)
40. Kubota, R., Tamukoh, H., Kawano, H., Suetake, N., Cha, B., Aso, T. (2016) A color quantization based on vector error diffusion and particle swarm optimization considering human visibility. In: Bräun, T., McCane, B., Rivera, M., Yu, X. (eds.) Image and video technology. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 9431, pp 332–343. Springer, Cham
41. Ozturk, C., Hancer, E., Karaboga, D.: Color image quantization: a short review and an application with artificial bee colony algorithm. Inform **25**(3), 485–503 (2014)
42. Pérez-Delgado, M.-L.: The color quantization problem solved by swarm-based operations. Appl. Intell. **49**(7), 2482–2514 (2019). https://doi.org/10.1007/s10489-018-1389-6
43. Pérez-Delgado, M.-L.: Artificial ants and fireflies can perform colour quantisation. Appl. Soft Comput. **73**, 153–177 (2018). https://doi.org/10.1016/j.asoc.2018.08.018
44. Jitpakdee, P., Aimmanee, P., Uyyanonvara, B.: A hybrid approach for color image quantization using k-means and firefly algorithms. World Acad. Sci. Eng. Technol. **77**, 133–139 (2013)
45. Eusuff, M.M., Lansey, K., Pash, F.: Shuffled frog-leaping algorithm. A memetic meta-heuristic for discrete optimization. Eng. Optim. **38**(2), 129–154 (2006)
46. Pérez-Delgado, M.-L.: Color image quantization using the shuffled-frog leaping algorithm. Eng. Appl. Artif. Intell. **79**, 142–158 (2019). https://doi.org/10.1016/j.engappai.2019.01.002
47. Dekker, A.H. Kohonen neural networks for optimal colour quantization. Network: Comput. Neural. Syst. **5**(3), 351–367 (1994). https://doi.org/10.1088/0954-898X_5_3_003
48. Kad, E.S., Kaur, E.S.: A note on quantization using cuckoo search using self information: a short review and an application with real images. Int. J. Comput. Sci. Inf. Technol. **6**(4), 3712–3715 (2015)
49. Yazdani, D., Nabizadeh, H., Kosari, E.M., Toosi, A.N.: Color quantization using modified artificial fish swarm algorithm. In: Wang, D., Reynolds, M. (eds.) AI 2011: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol 7106, pp. 382–391. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-25832-9_39
50. El-Said, S.A.: Image quantization using improved artificial fish swarm algorithm. Soft. Comput. **19**(9), 2667–2679 (2015). https://doi.org/10.1007/s00500-014-1436-0
51. Khullar, S., Verma, C.: Bacteria foraging optimization based color quantization. An Int. J. Eng. Sci. Inaug. Iss. (2010). http://ijoes.vidyapublications.com/paper/JSI/7.pdf. Accessed 15 May 2019
52. Kaur, R., Girdhar, A., Gupta, S.: Color image quantization based on bacteria foraging optimization. Int. J. Comput. Appl. **25**(7), 33–42 (2011)

53. Dua, R.L., Gupta, N.: Fast color image quantization based on bacterial foraging optimization. In: Proceedings of the Fourth International Conference on Advances in Recent Technologies in Communication and Computing, pp. 100–102. IET (2012)
54. Gupta, S., Sharma, V., Mohan, N., Singh Sandhu, P: Color reduction in RGB based on bacteria foraging optimization. In: International Conference on Computer and Communication Technologies, Phuket, pp. 174–177, 26–27 May 2012 (2012)
55. Kumar, D., Chopra, V.: Image quantization using HSI based on bacteria foraging optimization. Int. J. Inf. Technol. Knowl. Manag. **5**(2), 335–343 (2012)
56. Kumar, D.: Implementation of bacteria foraging optimization for color image quantization and its evaluation for various file formats. Int. J. Comput. Sci. Commun. Eng. **2**(1), 36–42 (2013)
57. Schaefer, G., Agarwal, P., Celebi, M.: Effective colour reduction using grey wolf optimisation. In: Tavares, J., Natal Jorge, R. (eds.) VipIMAGE 2017. Proceedings of the VI ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing. Lecture Notes in Computational Vision and Biomechanics, vol. 27. Springer, Cham, pp. 170–178 (2017)