Applied Intelligence (2019) 49:2482–2514
https://doi.org/10.1007/s10489-018-1389-6

The color quantization problem solved by swarm-based operations

María-Luisa Pérez-Delgado[1]

Published online: 23 January 2019

Abstract
The objective of the color quantization problem is to reduce the number of different colors of an image, in order to obtain a new image as similar as possible to the original. This is a complex problem and several solution techniques have been proposed to solve it. Among the most novel solution methods are those that apply swarm-based algorithms. These algorithms define an interesting solution approach, since they have been successfully applied to solve many different problems. This paper presents a color quantization method that combines the Artificial Bee Colony algorithm with the Ant-tree for Color Quantization algorithm, creating an improved version of a previous method that combines artificial bees with the K-means algorithm. Computational results show that the new method significantly reduces computing time compared to the initial method, and generates good quality images. Moreover, this new method generates better images than other well-known color quantization methods such as Wu's method, Neuquant, Octree or the Variance-based method.

Keywords Color quantization · Artificial ants · Ant-tree algorithm · Artificial bee colony algorithm · Clustering

1 Introduction

Color quantization consists of reducing the number of colors of an image in such a way that the quality of the quantized image is acceptable. Let us consider an image with n pixels, $\{p_1, ..., p_n\}$, represented in the RGB color space so that each pixel is stored as a vector of three integers between 0 and 255 which represent the amount of red, green and blue, $p_i = (R_i, G_i, B_i)$. To solve the quantization problem, a quantized palette with q RGB colors must be defined, $C = \{c_1, ..., c_q\}$, where q is smaller than the number of colors of the original image. The quantized image is obtained by replacing each pixel of the original image by a color of the quantized palette. Garey et al. demonstrated that finding the optimal quantized palette is an NP-complete problem [19].

Electronic documents include many images that must be stored, transmitted and displayed. Current devices can display high quality images with many colors. Nevertheless, the quality of the image is a disadvantage for its storage and transmission, since more colors means more quality, but also implies more storage space and slower speed of transmission. Reducing the colors of an image not only allows it to be displayed on low-end devices, but also reduces the size of the image and this allows it to be stored and transmitted more efficiently.

Color quantization is a problem in itself, and also appears as a subproblem in other image processing contexts, such as texture analysis [17, 49], image segmentation [1, 48], image compression [6, 12, 55, 66], or content-based image retrieval [36, 50, 60, 61]. The broad field of application of this problem has led to a number of solution methods being proposed over the years.

This paper proposes a color quantization method which combines two swarm-based algorithms: the Ant-tree for Color Quantization (ATCQ) algorithm and the Artificial Bee Colony (ABC) algorithm.

The ATCQ algorithm was proposed by Pérez-Delgado [45]. This method is based on the Ant-tree algorithm [4], which imitates the self-assembly behavior of some types of natural ants. The ATCQ algorithm modifies some characteristics of the original Ant-tree algorithm to perform color quantization. Such modifications make it possible to obtain a solution with low computational cost and to handle the memory requirements associated with the large amount of pixels to be processed. The results published in [45] show that ATCQ is competitive with some well-known color quantization methods.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10489-018-1389-6) contains supplementary material, which is available to authorized users.

✉ María-Luisa Pérez-Delgado
mlperez@usal.es

[1] University of Salamanca, Escuela Politécnica Superior de Zamora, Av. Requejo, 33, C.P. 49022, Zamora, Spain

The ABC algorithm is based on the foraging behavior of honey bees [27]. Although it was designed to solve optimization problems, it has also been applied to other problems such as clustering [30, 68], artificial neural networks training [31] and data mining [53]. An overview of variants and applications of this algorithm is shown in [5, 32].

Ozturk et al. combined the ABC algorithm with the K-means algorithm to solve the color quantization problem [41]; this method will hereinafter be referred to as ABC+K-means. K-means is a very popular clustering algorithm used to divide a dataset into k clusters or groups [24]. The algorithm considers k initial centroids (selected randomly or with other methods) and applies an iterative process to improve them. Each iteration first assigns each point to the nearest centroid and then recalculates the centroids. The ABC + K-means algorithm generates good images, but with a high computational cost. Although the results reported in [41] do not include execution time information, the parameters used for the tests, combined with the application of the K-means method, indicate that ABC + K-means is not competitive with respect to time as compared to other color quantization methods.

The color quantization method proposed in this paper is based on ABC + K-means, but combines ABC and ATCQ algorithms to obtain a faster method. The idea for this approach arose from the results reported in [45], which show that ATCQ can obtain better images than K-means with less time consumption.

The rest of the article is organized as follows. Section 2 briefly describes some well-known color quantization methods. Sections 3 and 4 describe the ABC and the ATCQ algorithms, respectively, since they are the starting point to understand the algorithm proposed in this paper, which is described in Section 5. This section also describes the ABC+K-means algorithm, to compare it with the new method. Section 6 shows computational results and the paper ends with the conclusions.

2 Color quantization methods

Two main groups of methods that perform color quantization can be identified: splitting methods and clustering-based methods. The first group obtains solutions at a lower computational cost, but the second group can obtain better solutions.

Splitting methods divide the color space into boxes until q boxes are obtained; then, each one is represented in the quantized palette by a single color. Some splitting methods are Median-cut [22], Variance-based method [58], Binary splitting [39], Wu's methods [64, 65] or Octree [20].

The Median-cut method selects the box with more pixels and splits it along the longest axis at the median point. When q boxes are obtained, the average color of each one defines a color of the quantized palette.

The Variance-based method is similar to the previous one, but splits the box with the largest weighted variance along the axis with the least weighted sum of projected variances; the splitting point is the point that minimizes the marginal squared error.

The Binary splitting method takes the box with the largest distortion and splits it along the axis with the largest variance; the splitting point is the projection of the centroid to such axis.

Wu proposed an algorithm similar to the Variance-based method, which selects the box with the largest weighted variance and splits it along the axis that minimizes the sum of the variances of both sides of the splitting plane [64]. The same author proposed another method that applies dynamic programming. This method sorts the colors along their principal axis and then splits the color space taking into account this ordering [65].

The Octree algorithm creates a tree structure limited to 8 levels and 8 children per node. First, the pixels of the image are processed to create a tree which stores the colors of the image on the leaf nodes. Then, the tree is processed from the leaves towards the root node, merging the nodes with similar colors until q colors are obtained.

Some other solutions inspired by the methods described above appear in [10, 14, 25, 26, 54, 67].

The Variance-cut method is based on the Binary splitting method, but it selects the box with the greatest sum of squared error and splits it along the axis with the greatest variance at the mean point [10].

Sirisathitkul et al. proposed a method based on Median-cut that performs data partitioning along the color axis with the greatest variance [54]. To split a box, the colors are sorted along the principal axis (the axis with the highest variance of color distribution), and then the Euclidean distances between adjacent colors along such axis are computed. The average value of said distances defines the splitting point ant the splitting plane is perpendicular to the principal axis.

The solution described in [26] is based on the method proposed by Wu in [65], but computes a new value instead of the actual variance. This method does not calculate the actual variance of the selected box using all the points in that box; on the contrary, it only uses eight values corresponding to the centroids of eight sub-boxes defined by three planes that pass through the centroid of the selected box and are perpendicular to the three axes.

Clustering-based methods define groups or clusters of similar pixels that are represented by the same color in the quantized image.

K-means is the most representative clustering method applied for color quantization. In this case, the centroids considered by the algorithm are q colors that define the quantized palette. A drawback of this method is that the results are conditioned by the initial centroids. Several modifications have been proposed to reduce the effect of the initial conditions of such method when applied to color quantization [9, 23, 33, 52, 57]. For example, the Weighted sort-means method [9] applies K-means, but only uses a subset of pixels of the input image and each point of this subset is assigned a weight that is proportional to its frequency. In addition, the sort-means algorithm is applied to reduce the number of calculations performed by K-means to associate each item to a cluster.

The Fuzzy c-means algorithm is based on the same idea as K-means, but each point can belong to several clusters with a degree of membership. This method has also been applied to reduce the number of colors of an image [40, 51, 63].

K-means has also been combined with some swarm-based algorithms, such as the Particle Swarm Optimization algorithm [37, 38] or the ABC algorithm [41]. The Particle Swarm Optimization algorithm is an optimization method that considers a set of particles that represent solutions to the problem. The operations of the algorithm move the particles in the search space in order to improve the solution they represent. When this method is applied to solve the color quantization problem, each particle represents a quantized palette and the objective function associated with the problem is the mean squared error, which must be minimized. To improve the results, the K-means algorithm is applied in a probabilistic way to the particles. Section 5.1 describes the application of the ABC algorithm combined with K-means to reduce the colors of an image.

Artificial Ants define another clustering method that has been applied to color quantization [21, 45]. The solution proposed in [21] uses a set of ants to define groups of similar pixels. At the beginning of the process, the pixels of the original image are spread on a grid. Then, the ants move on the grid to define the groups. With this purpose, each ant takes one pixel and moves it to another position on the grid where there are similar pixels. It should be noted that the process of creating the groups can be slow. The approach proposed in [45] is described in Section 4.

Artificial Neural Networks have also been applied to solve the color quantization problem [11, 16, 43, 44, 59]. These networks are systems that can learn to solve problems from examples. Several authors have used the self-organizing feature map (SOM) [34]. This network includes two layers: the input layer, which is used to present the input data, and the competitive layer, which includes a set of neurons that are trained with the input data. When an item of the data set is presented to the network, the most similar neuron to the input item is selected as the winner. Then, the network updates the weights of the winner and the neighboring neurons so that they are more similar to the input item.

Neuquant is a popular color quantization algorithm that trains a one-dimensional SOM with q neurons and uses the weights of the network to define the quantized palette [16]. The initial weights of the neurons are set to positions on the main diagonal of the RGB cube, and the network can be trained with a subset of pixels of the original image to accelerate the process (although the quality of the quantized image is reduced).

The Color importance-based SOM, proposed in [44], uses the frequency of the colors in the image to compute the color importance. Then, it applies the same operations as SOM, but uses the color importance of the winning neuron to determine the learning rate and the radius of neighborhood. Moreover, the network is trained with subsequences of data that include pixels at a fixed distance. The initial point of the subsequence used in iteration i is the i-th point of the sequence defined by all the pixels of the image, so that successive iterations consider different subsets of pixels.

The Frequency sensitive SOM (FS-SOM) [11] combines the SOM network with the frequency sensitive competitive learning. There are four differences between this method and the basic SOM: 1- the weights of the neurons are initialized with uniformly distributed weight vectors on the major diagonal of the input space; 2- the input item is selected according to a butterfly permutation sequence; 3- the learning rate is a function of the number of times a neuron wins; 4- after a predefined number of iterations, the neurons that have never won are updated based on the weights of the most frequently updated neurons.

When a SOM network is used, the number of neurons and the connections among them are defined before training the network. Nevertheless, the Growing neural gas is a self-organizing neural network that determines its own number of neurons and topology during the training process, based on the distribution of the input data [18]. The number of neurons of the network is progressively increased and the connections between the neurons are dynamically created and eliminated. This network model was applied to color quantization in [42]. In this case, the initial network includes two neurons whose weights are the colors of two random pixels of the image. At each iteration of the algorithm, a random pixel is presented to the network and the nearest neuron, A, and the second-nearest neuron are selected. If there is no connection between both neurons, the connection

is defined and its age is set to 0. In addition, the algorithm updates the error associated with A, the weights of A and its neighbors and the age of all the connections associated with A. The next operation removes the connections whose ages exceed a threshold and the neurons without connections. Moreover, when the current iteration is a multiple of a given parameter, a new neuron is inserted taking into account the two neurons with the maximum error. The last operation of the iteration reduces the error of all the neurons. The Growing neural gas network was used to define other solutions to the color quantization problem in [3, 13].

An interesting solution method based on competitive learning is proposed in [8]. In this article, the color quantization problem is solved by the Adaptive distributing units method proposed in [56]. This clustering method starts with a single unit (cluster), that is represented by the centroid of the data set, and applies an iterative process to define q units (clusters). Each iteration considers a random point of the set and selects the nearest unit, which is considered the winner unit. Then, this unit is moved towards the input point. Moreover, if its win count exceeds a predefined threshold, the winner unit is split and a new unit (cluster) is defined.

References [7] and [41] can be read to obtain more information about the color quantization methods.

3 The ABC algorithm

Honey bees are social insects that live in beehives. There are several types of bees in a beehive, each with a specific role in the search for food. The work of the bee colony is oriented to find good food sources and to take the nectar of such sources to the beehive. When the amount of nectar of a food source greatly decreases, that source is abandoned and replaced by another food source with more nectar. Based on the behavior of these bees, Karaboga proposed the ABC algorithm to solve optimization problems [27–29].

Let us consider an optimization problem defined on a D-dimensional space, with an objective function $f(x)$. The solution to the problem is a vector $x_k = (x_{k1}, ..., x_{kD})$ that minimizes (or maximizes) the objective function. To solve this problem by artificial bees, each food source represents a solution to the problem and the nectar associated with that food source represents the quality of the solution. The algorithm considers a set of F food sources, $\{x_1, ..., x_i, ..., x_F\}$, where x_i is a vector with D components, $x_i = (x_{i1}, ..., x_{iD})$. The fitness or amount of nectar associated with x_i can be computed as a function of $f(x_i)$ by (1).

$$fitness(x_i) = \begin{cases} \frac{1}{1+f(x_i)} & f(x_i) \geq 0 \\ 1 + abs(f(x_i)) & f(x_i) < 0 \end{cases} \quad (1)$$

Algorithm 1 ABC algorithm

```
for i = 1 to F do
    Initialize the food source i by (2)
    Set limit_i = 0
end for

for t = 1 to t_max do
    for each employed bee i do {***Apply employed operations***}
        Obtain a candidate food source, v_i, by applying (3)
        if (fitness(v_i) > fitness(x_i)) then
            Set x_i = v_i, limit_i = 0
        else
            Set limit_i = limit_i + 1
        end if
    end for
    for each onlooker bee do {***Apply onlooker operations***}
        Select a food source x_i depending on the probability given by (4)
        Obtain a candidate food source, v_i, by applying (3)
        if (fitness(v_i) > fitness(x_i)) then
            Set x_i = v_i, limit_i = 0
        else
            Set limit_i = limit_i + 1
        end if
    end for
    for each food source x_i do {***Apply scout operations***}
        if (limit_i > LI) then
            Replace x_i with a new solution generated by (2)
        end if
    end for
    Select the best solution so far
end for
```

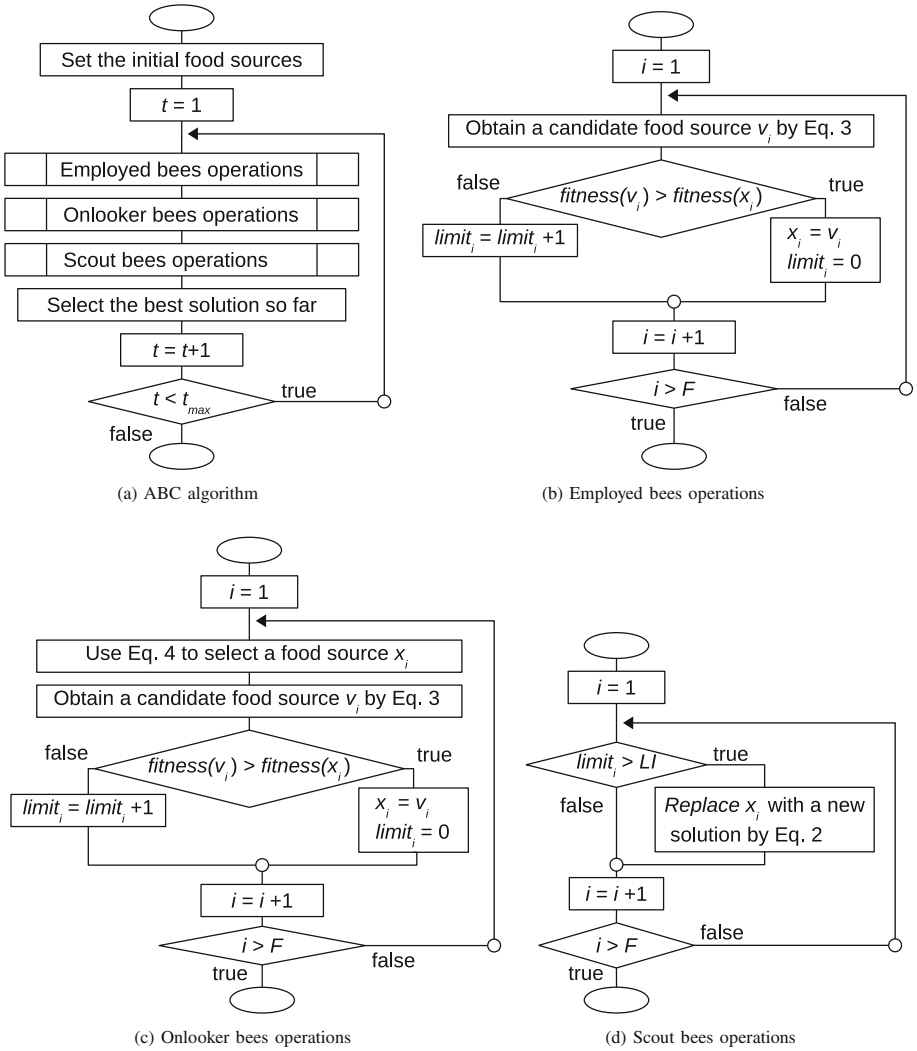
The swarm considered includes three types of bees: employed bees, onlooker bees and scout bees. Each employed bee is associated with a specific food source; it takes the nectar to the beehive and informs to the onlooker bees about the food source. Each onlooker bee selects a food source based on the information provided by the employed bees. The scout bees perform a random search for new food sources.

After selecting random initial food sources, employed and onlooker bees exploit these sources, which causes them to become exhausted. When the source associated with an employed bee is exhausted, it becomes a scout bee, and then it must look for another food source.

Algorithm 1 and Fig. 1a show the main operations of the ABC algorithm. The first operation sets the initial food sources randomly. The initial value of the component j of the food source x_i is computed by (2), where γ is a random value in [0, 1] and $[x_j^{\min}, x_j^{\max}]$ is the range of valid values for x_{ij}.

$$x_{ij} = x_j^{\min} + \gamma(x_j^{\max} - x_j^{\min}) \quad (2)$$

Each food source x_i has a variable $limit_i$ associated with it. Such variable is set to 0 when the source begins to be exploited and it increases during the iterations of the



(a) ABC algorithm (b) Employed bees operations

(c) Onlooker bees operations (d) Scout bees operations

Fig. 1 Diagram of ABC algorithm

algorithm; when it reaches a predefined value LI, the source x_i is considered exhausted and it is abandoned.

Once the initial food sources have been selected, an iterative process is applied $t_{\max}$ times. Each iteration performs operations related to each type of bee. First, employed bees operations are performed (Fig. 1b). Each employed bee is exploiting a food source, and it also looks for new food sources near to the current one; if it finds a source with more nectar, such source replaces the current one.

The employed bee which is exploiting x_i selects a new food source v_i in the neighborhood of x_i by applying (3), where ϕ_{ij} is a random value in $[-1, 1]$; x_k is a randomly selected food source, with $k \neq i$, and j is a component randomly selected. If the fitness of v_i is better than the fitness of x_i, the bee forgets the current source (x_i) and remembers the new one (v_i). Each time a bee selects a neighboring source whose fitness is worse than the fitness of x_i, $limit_i$ increases by one, to reflect the fact that the current source cannot be replaced by a better source.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3}$$

Then, onlooker bees operations are applied to perform more local search around the best solutions (Fig. 1c). These bees perform a probabilistic selection of a food source among the sources currently exploited by the employed bees, based on the information provided by those bees. The probability associated with select x_i is computed by (4). Once the onlooker bee makes its selection, it performs the same operations as an employed bee to decide if a neighboring food source is better than the current one.

$$p_i = \frac{fitness(x_i)}{\sum_{j=1}^{F} fitness(x_j)} \tag{4}$$

After onlooker bees, scout bees operations are applied to abandon solutions not improved after a certain number of trials (Fig. 1d). This prevents the algorithm from being trapped in local optima.

When $limit_i$ reaches the value LI, the source x_i is abandoned. At that moment, the employed bee associated with x_i becomes a scout bee ant it selects a new random food source by applying (2).

The last operation of each iteration of the algorithm stores the source with the best fitness found so far. When the algorithm ends, such source represents the solution to the problem.

4 The ATCQ algorithm

The ATCQ algorithm adapts the Ant-tree algorithm to perform color quantization. The Ant-tree algorithm is a clustering method inspired by the self-assembly behavior of some types of real ants [4]. These ants connect their bodies and build structures that allow them to cross empty spaces or avoid obstacles. The ants begin to construct the structure from a fixed support and they move on the structure until they connect to other ants already connected.

The ATCQ algorithm creates a tree structure to define a solution to the color quantization problem. Each pixel p_i of the original image is represented by an ant, h_i. These ants are connected to define a tree structure, where each subtree connected to the root node defines a color of the quantized palette. The algorithm ends when all the ants have been connected to the tree; at this moment, the number of children of the root, q, defines the number of colors of the quantized palette. To define the quantized image, the pixels associated with the ants connected to a subtree are represented by the color of such subtree.

There are three types of nodes in the tree. The first level of the structure only includes the root node or support, a_0. The second level includes the q children of a_0, (S_1, S_2, ..., S_q); these nodes are the roots of the q subtrees of ants created by the algorithm. The other levels of the tree include the ants that are progressively connected to the structure.

The number of children of each node is limited to $L_{\max}$. Since the number of children of a_0 determines the number of colors of the quantized palette, another limit is defined for this node, $Q_{\max}$ ($q \leq Q_{\max}$).

The root of subtree c, S_c, stores three values, denoted as nc_c, sum_c, and e_c. nc_c is the number of ants associated with this subtree, and sum_c is the sum of the RGB colors of such ants; the color of this subtree is computed as a function of both values: $RGB_c = sum_c/nc_c$.

Let d_{ic} denote the similarity between ant h_i and the color of subtree c computed when this ant is included into this subtree ($d_{ic} = Sim(h_i, RGB_c)$). e_c is the sum of the d_{ic} values for all the ants associated with subtree c (5).

$$e_c = \sum_{h_i \in c} d_{ic} \tag{5}$$

The value of e_c is used to compute the threshold T_c for the subtree c, according to (6), where $\alpha \in (0, 1]$. This threshold is used to decide if an ant is similar enough to a subtree, which allows it to connect to the subtree. It is also used to decide when a new subtree should be created (when the ant does not reach the threshold).

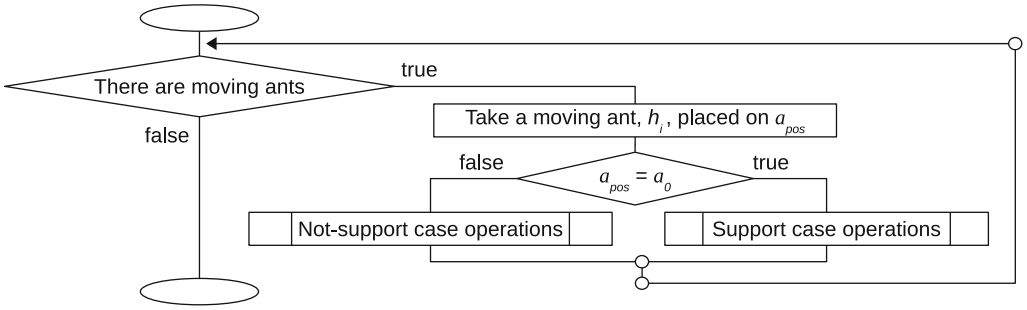
$$T_c = \frac{e_c}{nc_c}\alpha \tag{6}$$

Algorithm 2 ATCQ algorithm

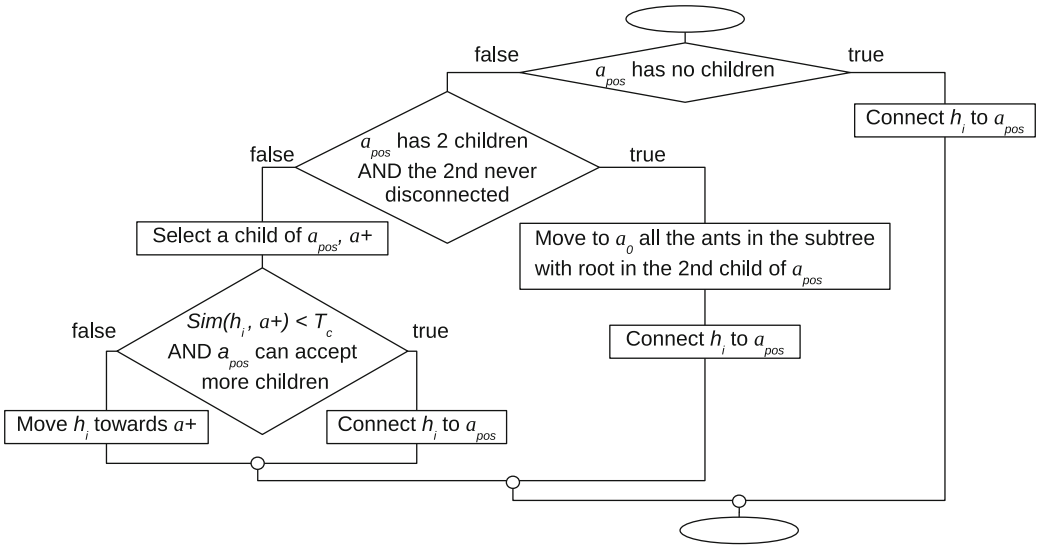
PARAMETERS: $tree$, q, α
while there are moving ants **do**
  Take a moving ant, h_i, now on a_{pos}
  if ($a_{pos} = a_0$) **then** {** *if the selected ant is on the support* **}
    if ($q = 0$) **then** {**** *the first subtree is created* ****}
      Set $q = 1$
      Create the child S_1 of a_0
      Set $nc_1 = 1$, $sum_1 = h_i$, $e_1 = 0$
      Connect h_i to S_1
    else
      {**** *selection of the subtree c most similar to* h_i ****}
      Set $c = 1$, d_{ic} = Sim(h_i, RGB_1)
      for j=2 to q **do**
        if (d_{ic} <Sim(h_i, RGB_j)) **then**
          Set $c = j$, d_{ic} = Sim(h_i, RGB_c)
        end if
      end for

      {**** *decision for* h_i ****}
      if ($d_{ic} < T_c$) **and** ($q < Q_{max}$) **then** {****** *a new subtree is created* ******}
        Set $q = q + 1$
        Create the child S_q of a_0
        Set $nc_q = 1$, $sum_q = h_i$, $e_q = 0$
        Connect h_i to S_q
      else {****** *a subtree is modified* ****** }
        Move ant h_i towards node S_c
        Set $nc_c = nc_c + 1$, $sum_c = sum_c + h_i$, $e_c = e_c + d_{ic}$
      end if
    end if
  else { ** *if the selected ant is not on the support* **}
    if no child connected to a_{pos} **then**
      Connect h_i to a_{pos}
    else if two ants connected to a_{pos} and the second one never disconnected **then**
      Move to a_0 the subtree with root in the second child of a_{pos}
      Connect h_i to a_{pos}
    else
      Select a node connected to a_{pos}, a^+
      if (Sim(h_i, a^+)< T_c) **and** (a_{pos} has less than L_{max} children) **then**
        Connect h_i to a_{pos}
      else
        Move h_i towards a^+
      end if
    end if
  end if
end while

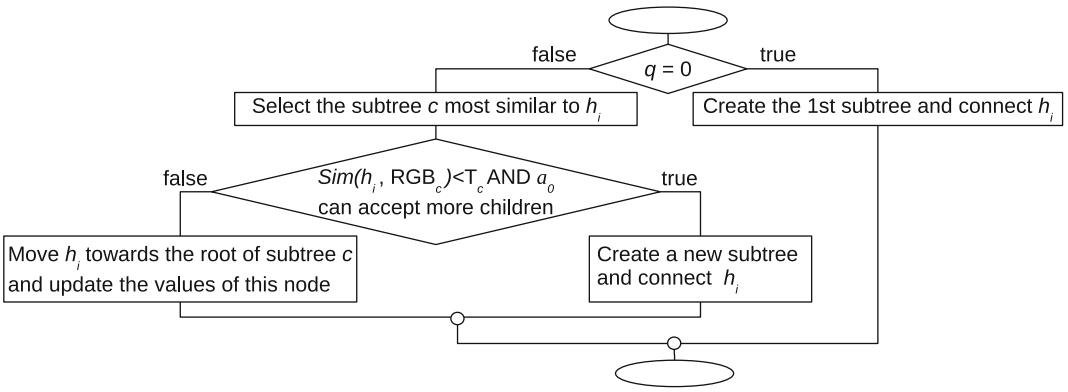
The initial tree only includes the support (q is 0) and all the ants are on such node. Then, an iterative process is applied to connect all the ants to the tree. Each ant moves on



(a) ATCQ algorithm



(b) Not-support case operations



(c) Support case operations

Fig. 2 Diagram of ATCQ algorithm

the structure until it connects to the tree, so that it becomes a new node. Algorithm 2 and Fig. 2a describe the operations applied to build the tree of ants. At each iteration, a moving ant h_i is selected (an ant that is not connected to the tree), which is on some node denoted by a_{pos}. The operations applied to h_i depend on the value of a_{pos} (Fig. 2b and c):

- If the selected ant is on the support ($a_{pos} = a_0$), the subtree c most similar to h_i is determined. If the similarity between c and h_i does not reach the threshold for the subtree ($d_{ic} < T_c$) and a_0 can accept more children, a new subtree is created to connect h_i; otherwise, h_i moves towards the subtree c.

When a new subtree is created, q increases by one, the root of the new subtree, S_q, is created and initial values are given to the three variables associated with S_q; sum_q takes the color associated with h_i, e_q is set to 0 and

nc_q is set to 1. Moreover, h_i connects as the first child of S_q.

When h_i moves to an existing subtree c, it is placed on the root of such subtree, S_c. In addition, nc_c, sum_c and e_c are updated to reflect the inclusion of the new ant into this subtree; nc_c increases by one, the color of h_i is added to sum_c and the similarity between h_i and the subtree c, d_{ic}, is added to e_c.

When the first ant is processed, the tree does not include any subtree ($q = 0$); in this case, the first subtree is created and h_i connects as the first child of such subtree.

- If the selected ant is not on the support ($a_{pos} \neq a_0$), the operations depend on the number of children of a_{pos} (these operations are based on the variant of Ant-tree called Ant-tree no-thresholds [45]):
  - If a_{pos} has no children, h_i connects to a_{pos}.
  - If a_{pos} has two children and the second one has never been disconnected from the tree, the subtree with root in that second child is disconnected and replaced with h_i. All the disconnected ants are moved back to the support, giving them another opportunity to move to a better subtree.
  - In other cases, a child of a_{pos} is selected, denoted by a^+. If the similarity between h_i and a^+ does not reach the value T_c defined for the subtree that includes to a_{pos}, and a_{pos} can accept more children, h_i connects to a_{pos}; otherwise, h_i moves towards a^+. a^+ can be a random child or the child most similar to h_i; in the first case the algorithm consumes less time, whereas in the second one better images can be obtained.

The final value of q determines the number of colors of the quantized palette. The colors of this palette are the colors associated with the q subtrees: $\{RGB_1, ..., RGB_q\} = \{sum_1/nc_1, ..., sum_q/nc_q\}$. The quantized image is obtained by representing the pixels associated with each subtree by the color of such subtree.

The general algorithm can be modified to eliminate the disconnection of ants from the tree (lines 35 to 35 of Algorithm 2). In this case, a 3-level tree can be used, where all the ants of a subtree are connected as children of a node in the second level. This modification requires less computing time although the quantized image might be worse. For more details, [45] can be consulted.

5 Color quantization based on ABC

To solve the color quantization problem by artificial bees, the F food sources represent quantized palettes. The food source x_i includes q colors: $x_i = (x_{i1}, ..., x_{iq})$, where x_{ik} is an RGB color, $x_{ik} = (R_{ik}, G_{ik}, B_{ik})$.

Since the ABC algorithm is an optimization method, it is necessary to define an objective function for the color quantization problem. This function must measure the quality of the quantized palette associated with a food source.

This section first describes the method proposed by Ozturk et al. and then presents the new method proposed in this article.

5.1 Artificial bees with K-means for color quantization: ABC + K-means

The color quantization method proposed by Ozturk et al. in [41] combines the ABC algorithm with the K-means algorithm. It performs the basic operations of ABC shown in Algorithm 1 and, moreover, it applies K-means to the food source exploited by a bee.

The initial value of each food source (each quantized palette) is determined by randomly selecting q pixels of the original image. When a food source is abandoned and it must be replaced by another food source, the same mechanism is applied.

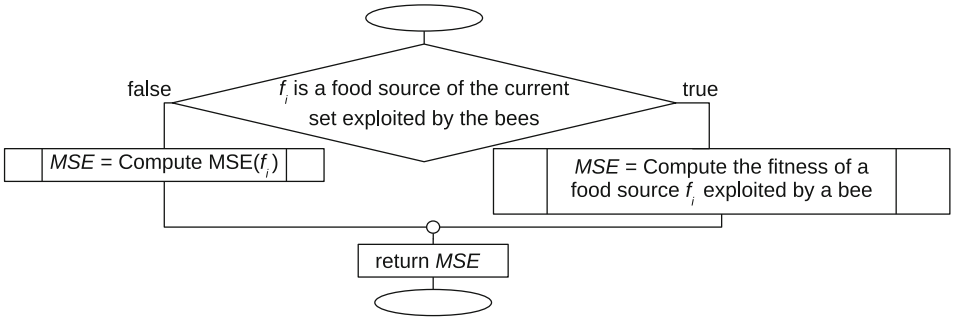
The objective function considered is the mean squared error (MSE), whose value must be minimized. The MSE of an image is computed by (7), where p_r is a pixel of the original image and p'_r is the pixel in the same position, but in the quantized image. Therefore, to compute $f(x_i)$ the values p'_r, with $r = 1, ..., n$ must be determined from the quantized palette x_i.

$$MSE = \frac{1}{n} \sum_{r=1}^{n} ||p_r - p'_r||^2 \tag{7}$$

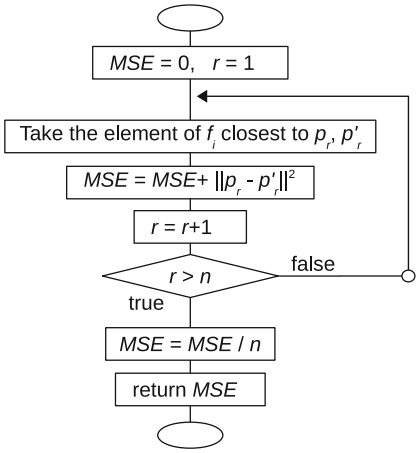
Algorithm 3 and Fig. 3b show the operations performed to compute the MSE value of a food source f_i (f_i represents to x_i or v_i). This algorithm determines the color of the palette f_i closest to each pixel of the original image and then computes MSE. Let $d(a, b)$ denote the Euclidean distance between two pixels a and b in the RGB color space. For each pixel of the original image, p_r, the element of f_i with minimum distance is determined; that value represents the color p'_r corresponding to p_r in the quantized image. Once this value is identified, the term $||p_r - p'_r||^2$ can be computed.

Algorithm 3 Compute the MSE for a food source f_i for ABC + K-means

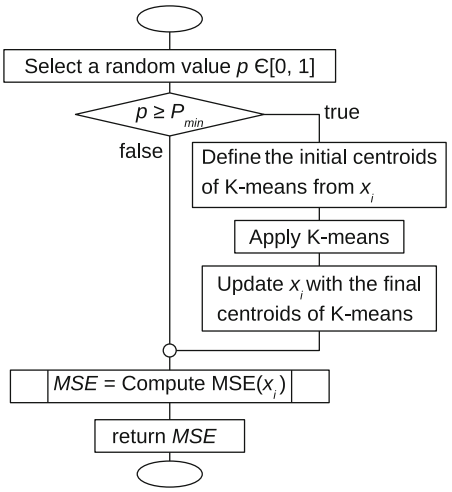
```
1: PARAMETERS: f_i
2: Set MSE = 0
3: for r = 1 to n do
4:     Set p'_r = f_i1; dist_min = d(p_r, f_i1)
5:     for k = 2 to q do
6:         if (d(p_r, f_ik) < dist_min) then
7:             Set p'_r = f_ik, dist_min = d(p_r, f_ik)
8:         end if
9:     end for
10:    Set MSE = MSE + ||p_r - p'_r||^2
11: end for
12: Set MSE = MSE/n
13: return MSE
```



(a) Compute fitness(f_i)



(b) Compute MSE(f_i)



(c) Compute the fitness of a food source x_i exploited by a bee

Fig. 3 Diagram that represents the operations performed to compute the fitness of a food source f_i when ABC + K-means is applied

The operations performed to compute the fitness of a food source depend on the type of source (Fig 3a). In any case, the MSE value is calculated, but for the sources belonging to the current set (any food source x_i, with $i = 1, ..., F$), the K-means algorithm can be applied previously.

When the fitness of the food source x_i associated with an employed bee must be computed, the K-means algorithm is applied to such source in a probabilistic way. If a random variable reaches a prefixed threshold $P_{\min}$, K-means is applied taking the values of x_i as initial centroids; when the algorithm ends, x_i represents the centroids adjusted by K-means. This process improves x_i since it is more adjusted to the data set. After this, the MSE of x_i is computed by Algorithm 3. Algorithm 4 and Fig. 3c show all the operations described to compute the fitness of x_i.

When an onlooker bee selects a food source, the same operations of Algorithm 4 are applied. Nevertheless, when a bee (employed or onlooker) explores a candidate source v_i, the fitness of such source is computed by Algorithm 3 (K-means is not applied).

Algorithm 4 Compute the fitness of a food source x_i exploited by a bee (employed or onlooker) – ABC + K-means

1: **PARAMETERS**: x_i
2: Select a random value p in [0, 1]
3: **if** $(p \geq P_{min})$ **then**
4: Apply K-means to x_i for a few iterations
5: **end if**
6: Compute the MSE for food source x_i
7: **return** MSE

The last step of each iteration of the ABC algorithm stores the food source with the best fitness found so far. Such food source is the quantized palette used to generate the quantized image at the end of the algorithm. The final image is built by replacing each pixel p_r of the original image by the element of the quantized palette closest to p_r (these values can be computed in the same way described in Algorithm 3).

5.2 Artificial bees with ATCQ for color quantization: ABC + ATCQ

Since K-means is a time consuming method, this paper proposes to replace such algorithm by the ATCQ algorithm when ABC is applied to color quantization.

Algorithm 5 and Fig. 4 describe the operations performed by ABC + ATCQ to compute the fitness of a food source x_i exploited by an employed bee or an onlooker bee.

As described in Section 4, the initial tree considered by ATCQ only includes the support. Nevertheless, to combine ATCQ with ABC, the tree considered also includes q nodes in the second level, $\{S_1, ..., S_q\}$, and these nodes take as initial color an element of x_i, that is $sum_j = x_{ij}$, for $j = 1, ...q$. Since the color of the subtree j is computed as sum_j/nc_j, nc_j must be set to 1 for every j value. The other parameter associated with the node S_j is set to 0 ($e_j = 0$). Therefore, when ATCQ is applied, the tree initially includes nodes in the first and the second level: the support node (where all the ants are) and q children of the support with an initial color.

Once the nodes in the second level of the tree have been defined and initial values have been assigned, the ATCQ algorithm is applied (Algorithm 2). When the execution of ATCQ ends, a new value has been defined for x_i that is more representative of the pixels of the original image. Therefore, the new value of each element x_{ij} of x_i is set to the color

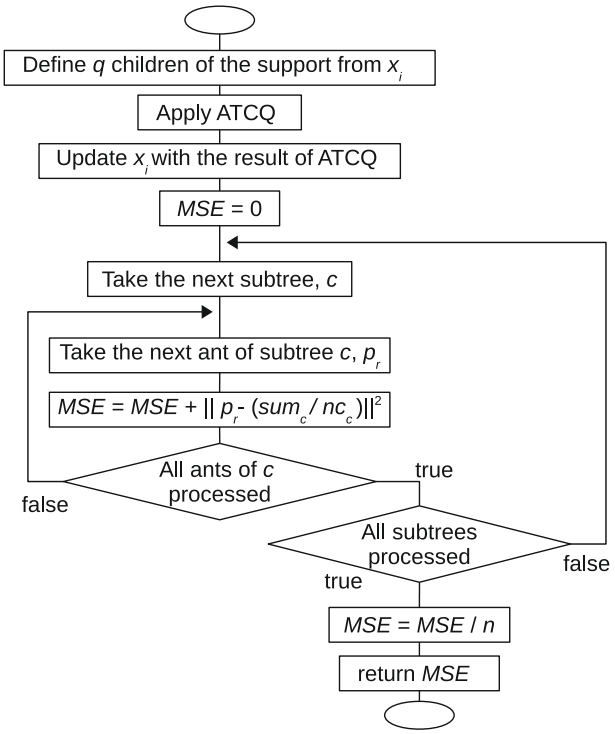


Fig. 4 Diagram that represents the operations performed to compute the fitness of a food source x_i exploited by a bee when ABC + ATCQ is applied

of a subtree. Moreover, the resulting tree is used to compute the MSE of x_i, since the color of the subtree c defines the value p'_r for every pixel p_r associated with such subtree (8).

$$MSE = \frac{1}{n}\sum_{c=1}^{q}\sum_{p_r \in c} ||p_r - \frac{sum_c}{nc_c}||^2 \quad (8)$$

To compute the fitness of a candidate source v_i, the same operations described for ABC + K-means are performed (Algorithm 3 and Fig. 3b) to allow a more exhaustive exploration of the search space. In that case, the fitness of v_i is computed without previously improving this solution by applying ATCQ. Therefore, (8) cannot be applied because no tree has been created. For this reason, Algorithm 3 is applied to compute the fitness of v_i according to (7).

It should be noted that when any component of a color is modified, it is always limited to take values in [0, 255].

Algorithm 5 Compute the fitness of a food source x_i exploited by a bee (employed or onlooker) – ABC + ATCQ

1: **PARAMETERS**: x_i
2: **for** $j = 1$ to q **do** {***create nodes in the second level of the tree* $tree_i$**}
3: Create the child S_j of node a_0
4: Set $sum_j = x_{ij}, nc_j = 1, e_j = 0$
5: **end for**
6: ATCQ($tree_i, q, \alpha_i$) {***apply ATCQ algorithm***}
7: **for** $j = 1$ to q **do** {***update* x_i *with the result of ATCQ***}
8: Set $x_{ij} = sum_j/nc_j$
9: **end for**
10: {***compute the MSE of this solution***}
11: Set $MSE = 0$
12: **for** each subtree c of $tree_i$ **do**
13: **for** each pixel p_r in this subtree **do**
14: Set $MSE = MSE + ||p_r - (sum_c/nc_c)||^2$
15: **end for**
16: **end for**
17: Set $MSE = MSE/n$
18: **return** MSE

In this case, the last step of the ABC algorithm, which stores the best food source found so far, stores the tree associated with that source. This allows the best quantized palette to be stored and use it to generate the quantized image when the algorithm concludes. This image is generated in the same way as described in Section 4 for the ATCQ algorithm.

It should be noted that the original ATCQ algorithm and the variant that is combined with ABC have some differences. In addition, the objective of ATCQ is not the same in both cases.

When ATCQ is applied independently, the initial tree only includes the support node. The initial quantized palette is empty and new colors are added as the pixels of the image are processed. When all the ants (pixels) have been connected to the structure, the operations of the algorithm

conclude. At this moment, the colors of the nodes in the second level of the tree define the colors of the quantized palette. The size of this palette is influenced by the features of the original image and the value selected for the α parameter. This size is a value between 1 and $Q_{\max}$. Moreover, the quantized image can be easily defined by replacing each ant (pixel) that belongs to a subtree by the color of that subtree.

When ATCQ is combined with ABC, the initial tree includes the support and the nodes in the second level. The initial quantized palette includes q colors, corresponding to the palette associated with a food source x_i. The colors of the palette are updated as the ants connect to the structure and the final palette is used to update the value of x_i. Moreover, the resulting tree structure is also used to compute the fitness corresponding to the new value defined for x_i. This fitness is quickly computed based on the color of each subtree and the colors of the ants (pixels) in such subtree. In this case it is not necessary to generate the quantized image; the objective is to improve the value of x_i and then calculate the fitness of that new value.

The original ATCQ is applied once to an image in order to generate a quantized image. However, in the proposed method, ATCQ is applied once to each food source x_i whose fitness should be evaluated. Each of these ATCQ applications is independent of the others and its objective is to improve the value of x_i and calculate its fitness, instead of generating a quantized image.

ABC + ATCQ performs a more exhaustive exploration of the solution space than the original ATCQ. ABC considers several solutions and selects the best. On the other hand, ATCQ can improve each of these solutions. The quantized image obtained by the original ATCQ is conditioned by the α parameter, since this parameter influences the number of colors of the final palette. Nevertheless, when ATCQ is combined with ABC, the final palette depends on the initial palette considered to build the tree. For this reason, the application of ATCQ in the proposed method can generate quantized palettes different from those obtained when applied independently.

The operations of ABC and ATCQ contribute to improve the final solution. First, the bees select a set of good candidate solutions. Next, this set is improved by applying the ATCQ algorithm to each solution. Then, the solutions improved by ATCQ are used to define the new set of candidate solutions considered by the bees. In this way, ABC and ATCQ cooperate to improve the solutions as the iterations of the ABC + ATCQ algorithm progress.

ABC + ATCQ and ABC + K-means are different because ATCQ and K-means are different. There are two main differences between ATCQ and K-means. First, ATCQ is not an iterative method. Second, ATCQ updates the colors of the palette as the ants are processed.

K-means and ATCQ are applied to an initial palette and generate a new palette, but the operations are different in both cases:

- At each iteration, K-means considers a set of initial centroids (the colors of a quantized palette) and applies two operations. First, the method determines the closest centroid for each pixel in the image. Next, the average color of the set of pixels associated with each centroid is computed. The values computed in the second operation define the new centroids, that is, the new quantized palette. This process is applied a certain number of iterations.

  Therefore, each iteration of K-means considers an initial palette and each color of such palette is recalculated at the end of the iteration.
- ATCQ updates the colors of the palette as the ants (pixels) are connected to the tree. That is, when an ant is associated with a subtree, the color of this ant is used immediately to update the color of such subtree. In this way, when an ant is processed, the colors of the subtrees represent the values computed based on all the ants previously processed. When the last ant is connected to the tree, the current colors of the subtrees define the quantized palette. As a consequence, ATCQ does not require the second operation used by K-means to compute the final palette. Moreover, each final color of the palette is computed from the initial color and the colors of all the ants connected to the corresponding subtree.

Once a new quantized palette has been obtained, the procedure to calculate its fitness is different in ABC + K-means and ABC + ATCQ. ABC + K-means computes the MSE value by Algorithm 3, and this requires to process again all the pixels of the image to determine the closest color in the quantized palette. On the contrary, ABC + ATCQ does not compute this value because it considers that the best color of the palette for each pixel is the color of the subtree that includes said pixel.

Therefore, ABC + ATCQ is faster than ABC + K-means because ATCQ operates faster than K-means. This is because ATCQ is not an iterative method, it defines the final palette as the tree is built and takes advantage of the information associated with the tree to compute the fitness of the palette.

6 Results and discussion

The ABC + ATCQ and ABC + K-means algorithms were coded in C language and executed on a PC running under Linux operating system with an I7 processor (2 GHz) and 16 GBytes of RAM.

Table 1 Results for palette size 32 (it: iteration of the algorithm; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	1	122.93	126.65	2.3	925	122.62	125.14	2.2	2228	122.82	128.30	3.4	9713
	5	119.26	120.79	0.8	4405	118.68	119.42	0.6	11653	118.85	119.90	0.7	48617
	10	118.68	119.88	0.8	8886	118.10	118.69	0.3	23324	118.51	119.32	0.4	96084
	15	118.40	119.41	0.7	13308	117.89	118.42	0.3	35064	118.51	119.16	0.4	144936
	20	118.40	119.29	0.6	17870	117.89	118.33	0.3	46257	118.51	119.16	0.4	191970
Girl	1	85.12	98.92	8.5	900	91.81	108.42	11.8	2120	91.41	114.07	15.4	9375
	5	82.09	83.91	1.1	4526	82.49	83.10	0.4	11658	82.98	84.54	1.0	47842
	10	82.07	82.77	0.6	9274	81.83	82.21	0.3	23180	82.49	83.44	0.7	96247
	15	81.96	82.41	0.4	13965	81.55	81.89	0.3	34562	82.49	83.06	0.4	145519
	20	81.78	82.18	0.3	18863	81.53	81.76	0.2	45784	82.49	82.95	0.3	191388
Plane	1	96.48	119.48	13.9	902	97.70	131.21	18.1	2498	112.66	132.30	10.8	9636
	5	73.74	87.24	11.0	4547	72.75	90.14	11.8	11875	88.00	113.93	14.2	47581
	10	68.04	75.46	4.4	8860	67.17	74.63	3.4	23626	76.33	95.58	17.4	94878
	15	66.86	70.30	2.1	13294	66.03	69.85	2.1	35426	74.31	86.24	10.1	144361
	20	65.34	67.08	1.3	17651	64.80	67.00	1.6	46937	74.31	84.08	9.0	190270
Peppers	1	238.14	245.55	4.6	765	236.33	241.63	3.9	2091	238.77	249.81	5.7	9440
	5	227.49	231.96	1.9	4416	226.82	230.28	1.7	11315	230.33	231.70	1.4	48562
	10	227.33	229.84	1.6	8915	226.82	228.97	1.4	22663	229.53	230.50	0.8	95845
	15	227.33	229.39	1.3	13229	226.82	228.87	1.4	33971	229.37	230.21	0.5	144727
	20	227.33	229.27	1.3	17584	226.82	228.57	1.4	45167	229.16	230.05	0.6	192514
Lake	1	208.91	215.82	3.9	971	207.61	213.71	4.9	2182	209.36	218.53	7.3	9579
	5	204.99	205.97	0.9	4824	204.16	205.13	1.1	11171	203.75	205.51	0.8	47169
	10	204.25	205.02	0.5	9420	203.51	204.18	0.7	22049	202.91	204.40	0.8	93833
	15	203.18	204.28	0.6	14060	202.36	203.71	0.6	33555	202.91	203.98	0.8	142806
	20	202.69	203.92	0.6	18598	202.21	203.15	0.6	44617	202.14	203.75	0.9	189882
Mandrill	1	385.77	394.10	4.9	911	382.61	385.65	2.4	2369	388.62	397.25	5.9	8553
	5	376.32	378.60	1.8	4846	374.71	376.66	1.0	11527	375.40	377.17	1.2	47219
	10	375.83	376.88	0.9	9461	374.32	375.97	1.1	23081	374.05	375.82	1.0	94599
	15	375.35	376.30	0.5	14144	374.32	375.51	0.7	34365	374.05	375.29	0.8	142024
	20	375.35	376.17	0.5	18932	374.32	375.14	0.6	46320	373.64	374.96	0.8	190425
EPSZ	1	116.53	121.62	2.5	905	115.31	119.23	2.9	2262	117.57	123.76	3.6	8714
	5	114.42	116.29	1.4	4454	113.54	114.40	0.7	11741	113.95	115.92	1.2	46659
	10	113.98	115.90	1.4	8687	113.24	114.01	0.4	23211	113.70	115.10	0.8	95114
	15	113.98	115.90	1.4	12854	113.09	113.89	0.5	34540	113.70	114.75	0.7	140276
	20	113.98	115.90	1.4	17127	113.09	113.87	0.5	46163	113.70	114.58	0.6	188657
Machine	1	75.93	79.08	2.8	908	73.83	76.83	3.3	2635	76.75	81.27	2.8	9354
	5	73.32	74.66	0.8	4455	72.17	73.12	1.0	11787	72.98	75.19	1.6	49653
	10	73.32	74.14	0.5	8715	71.82	72.72	1.1	23245	72.78	74.14	1.0	98205
	15	73.32	74.07	0.5	12922	71.82	72.68	1.1	34699	72.78	73.98	0.8	144245
	20	73.32	74.07	0.5	17164	71.82	72.68	1.1	45832	72.65	73.42	0.7	190904

Table 1 (continued)

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*
Text	1	36.73	40.04	2.1	916	37.08	38.42	1.3	2585	39.35	42.33	2.4	9752
	5	34.77	36.35	1.0	4613	34.68	35.24	0.6	12247	35.39	37.14	1.3	52071
	10	34.59	35.32	0.7	9204	34.33	34.82	0.4	22687	35.34	36.35	0.7	107015
	15	34.54	35.23	0.7	13545	33.87	34.70	0.4	32575	35.31	35.99	0.5	164685
	20	34.26	35.19	0.7	17942	33.87	34.70	0.4	42533	35.31	35.97	0.5	227195
Hydrant	1	175.98	182.59	3.7	858	177.27	180.17	2.2	2403	180.71	191.80	5.4	8969
	5	169.24	172.34	1.7	4480	169.30	170.36	0.7	11688	171.77	173.70	2.1	46998
	10	169.24	171.02	1.2	9146	169.30	169.76	0.3	23127	170.31	171.85	1.3	94090
	15	169.24	170.78	0.9	13507	169.30	169.73	0.2	33359	170.31	171.39	0.6	139419
	20	169.24	170.78	0.9	17776	169.30	169.71	0.2	44364	170.31	171.05	0.5	187319
Sign	1	68.42	73.69	4.5	844	65.15	71.12	4.5	2311	67.98	73.25	4.4	9958
	5	62.15	64.19	1.5	4489	59.99	62.08	1.8	11858	61.57	63.87	2.1	48604
	10	61.36	62.90	1.2	8813	59.25	61.43	1.6	23344	61.10	62.77	1.6	95220
	15	61.36	62.83	1.1	13165	59.25	61.40	1.6	34577	61.10	62.23	0.9	144644
	20	61.36	62.73	1.0	17310	59.25	61.40	1.6	44763	61.10	62.18	0.8	192479
Bikes	1	110.17	117.63	5.1	937	110.89	115.64	3.9	2565	115.41	123.80	6.3	8734
	5	103.29	106.30	1.7	4672	103.95	105.53	1.4	11838	105.95	109.13	2.3	45516
	10	103.29	104.71	0.8	9197	102.67	103.88	0.7	23032	105.08	106.71	1.0	92597
	15	103.29	104.20	0.9	13739	102.57	103.71	0.8	33369	104.16	106.05	1.0	139219
	20	103.29	104.09	0.8	18233	102.57	103.71	0.8	44777	104.16	105.65	1.1	186310

Twelve images were analyzed: six images commonly used to test color quantization methods [62], and six new images proposed by the author of this article [46]. Four palette sizes were considered to quantize these images: $q = \{32, 64, 128, 256\}$. The following parameters were used for the tests: $F = 5, t_{\max} = 20, LI = 8$, for the ABC algorithm; $P_{\min} = 0.1$, and 5 iterations for the K-means algorithm; $L_{\max} = 32$, $Q_{\max} = q$, $\alpha = \{0.25, 0.30, 0.35, 0.40, 0.45\}$, for the ATCQ algorithm. Each α value was associated with a food source to which ATCQ was applied (the selected values were tested in [45], producing a good result for the global set of palette sizes considered).

Since the ABC + K-means algorithm is very slow, the referred values were considered for the tests rather than those used in the tests reported in [41].

Both variants of the ATCQ algorithm were used: the general form of the algorithm, which generates a multilevel tree; and the variant without disconnection of ants, which generates a 3-level tree.

Tables 1, 2, 3 and 4 show the results of 20 independent tests performed for each problem and palette size. These tables show the MSE value (minimum, average and standard deviation) and the average execution time for several iterations of each algorithm (iterations 1, 5, 10, 15 and 20).

Figures 5 and 6 show the best images obtained by ABC + ATCQ for $q = 256$ (the variant of ATCQ that generates a 3-level tree was considered). The best images obtained for all the palette sizes by both variants of ABC + ATCQ are included in Online Resource 1.

It can be observed that ABC + K-means consumes much more time than ABC + ATCQ for the same number of iterations, generating images of similar quality after the same number of iterations. On the other hand, when the results with similar computing times are compared, the images obtained by the new method are always better than those generated by ABC + K-means.

When the ATCQ algorithm builds a 3-level tree, the computing time is considerably reduced with respect to the multilevel tree case. Moreover, the MSE value is similar, on average, for both trees. Therefore, a good option is to combine ABC with the ATCQ algorithm that generates a 3-level tree.

Table 2 Results for palette size 64 (it: iteration of the algorithm; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC+K-means			
image	it.	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	1	75.13	77.02	1.4	1404	75.33	77.06	1.3	2903	77.48	79.30	2.0	15882
	5	72.74	73.36	0.5	7403	72.67	73.28	0.6	15389	72.96	74.02	0.6	88830
	10	72.61	72.88	0.3	14684	72.43	72.87	0.4	30833	72.70	73.25	0.4	179037
	15	72.20	72.72	0.3	22117	71.94	72.43	0.3	46810	72.42	72.92	0.4	271766
	20	72.20	72.64	0.3	29150	71.94	72.27	0.2	63255	72.29	72.85	0.4	364491
Girl	1	52.21	54.25	1.6	1581	52.29	54.92	5.0	3337	53.19	58.66	7.7	20668
	5	49.51	50.42	0.6	7750	49.67	50.21	0.4	16662	50.50	51.25	0.6	93725
	10	49.18	49.81	0.4	15417	49.14	49.57	0.2	33327	49.56	50.62	0.7	186012
	15	49.01	49.29	0.2	23222	48.98	49.34	0.3	49368	49.37	50.25	0.4	277755
	20	48.69	49.15	0.3	31213	48.21	49.10	0.4	66374	49.37	50.08	0.5	374537
Plane	1	55.01	62.82	7.9	1558	54.49	63.19	7.5	3283	55.99	74.22	14.0	18240
	5	47.31	51.20	2.4	7331	47.01	50.71	2.9	17155	48.02	56.06	8.3	88169
	10	44.65	47.72	1.6	14716	44.77	47.31	1.9	34367	46.98	51.08	3.6	179802
	15	42.83	45.37	1.1	22118	42.91	44.89	1.6	51955	46.98	49.93	2.6	276006
	20	42.36	43.59	0.9	29810	41.04	43.36	1.5	69079	46.98	49.55	1.9	370540
Peppers	1	142.53	146.99	3.3	1506	139.82	144.62	2.7	3203	145.54	149.49	3.6	18623
	5	135.10	136.85	1.0	7716	134.17	136.10	2.3	15865	134.93	137.82	3.0	93227
	10	133.62	135.54	0.9	15361	133.50	134.98	2.4	31701	134.11	136.01	2.2	186451
	15	133.62	135.05	0.7	23095	133.00	134.74	2.3	47449	133.13	135.36	2.2	279700
	20	133.62	134.58	0.6	30679	132.86	134.48	2.2	62694	133.13	134.93	1.2	374858
Lake	1	139.12	142.94	2.4	1539	137.22	142.11	2.9	3166	139.74	144.76	4.0	18145
	5	131.96	133.67	0.9	7848	132.15	133.37	0.9	15746	131.89	134.33	1.9	91437
	10	130.97	132.02	0.7	15578	130.75	131.77	0.6	32160	131.48	132.60	0.9	185888
	15	130.94	131.59	0.5	23219	130.28	131.07	0.5	47943	130.83	132.15	0.8	277014
	20	130.94	131.49	0.4	31040	130.28	130.90	0.4	64436	130.83	132.00	0.9	370754
Mandrill	1	242.83	246.66	2.5	1436	242.85	245.39	2.6	3048	242.85	247.07	4.3	18851
	5	238.07	238.93	0.5	7534	237.06	238.23	0.5	15924	236.69	238.05	0.9	86578
	10	236.66	237.79	0.6	15254	237.01	237.45	0.4	32105	235.42	236.82	0.8	180339
	15	236.66	237.34	0.4	22945	237.01	237.33	0.3	48120	235.42	236.48	0.6	270455
	20	236.32	237.15	0.4	30422	236.20	236.95	0.4	63508	235.42	236.38	0.6	358280
EPSZ	1	63.33	66.24	1.9	1425	62.02	65.17	2.1	3024	64.27	68.71	2.7	11078
	5	61.74	62.26	0.4	7009	60.57	61.15	0.3	16372	62.12	63.23	0.6	56960
	10	61.24	61.72	0.4	14427	60.57	60.92	0.2	32380	61.78	62.44	0.5	113921
	15	60.91	61.66	0.5	21867	60.57	60.92	0.2	48275	61.68	61.94	0.2	169744
	20	60.91	61.66	0.5	28998	60.57	60.92	0.2	63502	61.68	61.94	0.2	226264
Machine	1	44.08	46.07	1.0	1477	43.19	44.71	1.3	3337	46.48	47.80	1.5	17568
	5	42.13	43.43	0.8	7353	41.81	42.62	0.8	16474	42.88	44.33	1.1	86901
	10	41.91	42.86	0.6	14615	41.80	42.42	0.6	31785	42.88	43.91	0.8	181372
	15	41.91	42.79	0.7	21935	41.80	42.40	0.6	46815	42.88	43.71	0.7	275604
	20	41.91	42.79	0.7	29265	41.80	42.35	0.5	61369	42.67	43.44	0.5	375965

Table 2 (continued)

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*
Text	1	21.83	23.53	1.2	1374	21.37	22.77	1.0	3593	23.75	25.67	1.8	16890
	5	20.19	21.01	0.7	7278	20.15	20.98	0.5	17799	21.34	22.54	0.8	97597
	10	20.08	20.64	0.4	14717	20.15	20.68	0.4	34092	21.08	21.97	0.7	190165
	15	20.08	20.62	0.4	22052	20.15	20.61	0.4	48425	20.68	21.64	0.5	282779
	20	20.08	20.52	0.3	29253	20.15	20.59	0.4	62651	20.68	21.60	0.5	376677
Hydrant	1	105.84	112.27	3.3	1396	105.01	107.87	2.4	3342	109.08	115.88	3.9	23229
	5	101.84	103.04	1.0	7552	101.16	102.43	0.7	16171	101.85	104.69	1.6	100932
	10	100.83	101.89	0.8	15102	101.03	101.70	0.5	32158	101.85	103.53	0.8	190187
	15	100.56	101.53	0.7	22645	101.03	101.58	0.5	47246	101.85	103.32	0.7	279730
	20	100.56	101.42	0.7	29889	100.92	101.57	0.5	62274	101.85	103.05	0.6	369070
Sign	1	34.78	36.91	1.3	1558	34.95	36.61	1.7	3380	37.73	40.57	1.8	19220
	5	33.16	33.70	0.5	7401	32.59	33.14	0.5	17031	33.84	35.07	0.6	90541
	10	32.70	33.12	0.4	14802	32.21	32.77	0.4	32538	33.68	34.57	0.6	181609
	15	32.56	32.99	0.4	22129	32.16	32.75	0.4	47376	33.24	34.08	0.7	269914
	20	32.56	32.95	0.4	29446	32.16	32.72	0.4	62193	33.24	34.07	0.8	359969
Bikes	1	62.69	66.38	2.7	1406	63.87	66.10	2.0	3377	66.73	71.51	4.0	16819
	5	57.77	58.92	0.8	7309	57.10	58.50	1.1	16852	59.61	61.01	1.1	92629
	10	56.76	57.40	0.4	15035	56.81	57.22	0.3	33546	58.48	59.53	0.6	186780
	15	56.53	57.02	0.4	22393	56.30	56.96	0.3	48934	57.97	58.75	0.6	285968
	20	56.53	56.93	0.3	29818	56.30	56.91	0.3	64324	57.61	58.44	0.5	376842

ABC+K-means can obtain better images if the number of iterations of K-means increases, but this also makes the method more time-consuming. To analyze this effect, Fig. 7 compares the results of ABC + K-means for the Plane image when 1, 3, 5 and 7 iterations of K-means are applied. It can be clearly observed that the execution time of ABC + K-means increases with the number of iterations of K-means. Although the quality of the images also improves when more iterations of K-means are performed, the effect is not as clear as when time is compared.

Figure 7 also shows the execution time and the MSE results of both variants of ABC + ATCQ. This figure clearly shows that ABC + K-means consumes more time than ABC + ATCQ, even when only one iteration of K-means is applied. In addition, the MSE results are worse for ABC + K-means even when 7 iterations of K-means are considered. For the palette with 128 colors, the initial iterations of ABC + K-means with 7 iterations of K-means obtain MSE values very similar to those of ABC + ATCQ, although the first method consumes much more time than the second. Therefore, ABC + ATCQ can generate better images than ABC + K-means and requires less time than said method.

When K-means or ATCQ are combined with ABC to solve the color quantization problem, such methods improve the solution represented by a food source. Obviously, the combination of both methods increases the execution time, but also improves the quality of the quantized image. Computational results show that the basic ABC algorithm (the ABC algorithm applied without considering K-means or ATCQ) reduces the execution time, but generates worse quantized images than ABC + ATCQ and ABC + K-means. To analyze this effect, Fig. 8 compares the results obtained for the Plane image by ABC + ATCQ, the basic ABC algorithm and ABC + K-means with 1 iteration of K-means (the other results of ABC + K-means represented in Fig. 7 are not included to allow a better visualization of the figures correspondig to the execution time). Figure 8 clearly shows that the basic ABC algorithm generates the worst quantized image. As far as the execution time is concerned, this figure shows that ABC always consumes less time than ABC + K-means and more than the 3-level tree variant

Table 3 Results for palette size 128 (it: iteration of the algorithm; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC+K-means			
image	it.	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	1	47.95	48.72	0.8	2657	48.02	49.54	0.8	4576	49.54	51.09	0.9	35704
	5	46.70	47.10	0.2	12806	46.76	47.01	0.3	24545	47.12	47.58	0.4	177920
	10	46.40	46.80	0.2	25551	46.42	46.70	0.2	50616	46.88	47.13	0.2	365177
	15	46.32	46.67	0.2	38220	46.42	46.67	0.2	74702	46.79	47.04	0.2	544470
	20	46.32	46.67	0.2	51234	46.42	46.67	0.2	98442	46.79	47.03	0.1	727037
Girl	1	33.52	35.07	1.4	2668	32.63	33.90	0.8	5085	33.78	35.91	1.3	37366
	5	30.88	31.73	0.4	13621	29.92	31.49	0.6	24077	31.09	32.30	0.7	187627
	10	30.34	31.33	0.4	26927	29.78	31.06	0.5	49932	30.76	31.92	0.7	376608
	15	29.94	31.08	0.6	40132	29.66	30.89	0.6	75551	30.76	31.88	0.6	551590
	20	29.93	30.97	0.6	53619	29.66	30.75	0.7	99991	30.76	31.82	0.6	741261
Plane	1	36.15	38.57	1.4	2453	37.37	38.64	1.3	4574	37.00	39.74	2.2	37105
	5	31.61	33.45	1.1	12735	32.25	33.23	0.7	23816	32.84	35.07	1.7	183861
	10	30.78	32.19	0.9	26152	30.44	31.37	0.7	49053	31.22	34.01	1.6	367263
	15	29.90	30.86	0.6	39556	29.57	30.38	0.6	74893	31.22	33.49	1.1	544366
	20	29.33	30.07	0.5	52802	28.99	29.73	0.4	99025	30.67	32.99	1.3	721747
Peppers	1	87.86	90.53	1.9	2498	86.51	88.95	1.8	5117	89.24	91.87	1.6	37875
	5	84.32	84.81	0.4	13043	84.16	84.91	0.5	24428	84.17	85.30	0.5	183663
	10	83.89	84.30	0.3	26508	83.76	84.20	0.3	48915	83.81	84.48	0.4	363755
	15	83.89	84.18	0.2	40043	83.75	84.06	0.3	72879	83.81	84.33	0.3	543813
	20	83.89	84.16	0.2	53871	83.75	84.01	0.3	97338	83.81	84.15	0.2	715734
Lake	1	91.29	94.10	1.9	2653	91.05	93.27	1.7	4605	92.02	95.87	2.5	38099
	5	85.93	87.89	0.9	13316	86.72	87.61	0.6	25139	87.17	88.75	0.9	183314
	10	84.99	86.37	0.6	26600	85.48	85.99	0.4	50986	85.85	86.84	0.6	362281
	15	84.99	85.87	0.4	40067	85.18	85.65	0.3	76267	85.85	86.62	0.6	547728
	20	84.99	85.73	0.4	53867	85.18	85.52	0.3	100078	85.53	86.50	0.6	734342
Mandrill	1	155.63	158.27	2.0	2725	155.18	156.93	1.2	5521	157.32	160.49	2.8	35980
	5	152.19	153.76	0.6	13572	152.39	153.54	0.6	24996	152.15	153.01	0.9	184863
	10	151.97	152.80	0.5	27183	152.39	153.09	0.4	48899	151.70	152.16	0.3	369146
	15	151.97	152.73	0.5	40257	152.09	152.80	0.4	73136	151.29	151.85	0.3	556198
	20	151.97	152.68	0.4	52868	152.09	152.70	0.4	98692	151.17	151.81	0.4	739146
EPSZ	1	38.51	39.45	0.6	2489	38.23	39.12	0.6	5529	39.46	40.54	0.8	35987
	5	36.90	37.28	0.3	12982	36.83	37.31	0.3	25704	37.82	38.10	0.2	181030
	10	36.60	37.00	0.3	25903	36.64	36.97	0.3	50095	37.40	37.73	0.2	367568
	15	36.60	36.91	0.2	38499	36.64	36.92	0.2	73602	37.40	37.61	0.2	562110
	20	36.59	36.87	0.3	51664	36.62	36.90	0.2	99000	37.38	37.56	0.1	738989
Machine	1	27.31	28.08	0.5	2559	26.38	27.79	0.9	5019	27.75	28.96	0.7	36436
	5	25.40	26.13	0.4	12950	25.51	25.93	0.3	24617	26.53	27.19	0.5	182222
	10	25.01	25.74	0.4	26729	25.00	25.44	0.3	49305	26.26	26.82	0.4	364815
	15	25.01	25.59	0.3	40111	24.85	25.31	0.3	73596	26.26	26.64	0.4	539282
	20	25.01	25.56	0.3	52732	24.85	25.28	0.3	98026	26.26	26.56	0.3	719775

Table 3 (continued)

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*
Text	1	13.39	14.42	0.5	2884	13.77	14.46	0.4	5118	15.56	16.23	0.5	34699
	5	13.06	13.52	0.3	13580	13.02	13.57	0.2	28227	14.30	14.77	0.3	178572
	10	13.03	13.36	0.2	26675	12.84	13.28	0.2	55432	14.15	14.44	0.2	353702
	15	13.03	13.26	0.2	39789	12.79	13.14	0.2	82713	13.89	14.28	0.2	541081
	20	12.98	13.17	0.1	52991	12.75	13.07	0.2	110250	13.72	14.19	0.2	726843
Hydrant	1	66.46	68.67	1.8	2690	65.65	69.20	2.2	4919	66.45	70.02	2.4	39675
	5	61.55	62.38	0.5	13137	62.06	62.60	0.5	24415	62.95	64.22	0.8	176083
	10	61.01	61.35	0.2	26560	60.89	61.22	0.3	48399	62.28	62.93	0.5	350440
	15	60.89	61.07	0.1	39693	60.57	60.93	0.3	72605	62.00	62.70	0.5	532270
	20	60.81	61.02	0.2	52819	60.41	60.83	0.3	97023	61.79	62.47	0.4	713466
Sign	1	19.92	21.86	1.0	2628	20.45	21.93	0.6	5290	22.31	24.03	1.0	40701
	5	18.79	19.47	0.5	12701	19.04	19.51	0.3	24982	20.36	21.18	0.5	177811
	10	18.47	19.00	0.3	25905	18.61	18.96	0.2	49854	19.92	20.57	0.4	361048
	15	18.38	18.83	0.3	38902	18.61	18.82	0.2	73937	19.92	20.28	0.3	539153
	20	18.38	18.80	0.2	51642	18.55	18.77	0.2	97350	19.82	20.23	0.3	709602
Bikes	1	37.12	39.66	1.7	2582	36.33	38.96	1.8	4694	38.89	41.63	1.6	37194
	5	33.55	34.13	0.6	12851	32.57	33.57	0.6	25261	34.71	35.52	0.5	187158
	10	32.07	32.68	0.4	27032	31.99	32.43	0.3	50738	33.92	34.37	0.2	383173
	15	31.90	32.19	0.3	40341	31.55	32.04	0.3	75571	33.72	34.24	0.3	568262
	20	31.84	32.07	0.2	53748	31.55	31.92	0.2	100309	33.61	34.08	0.4	755229

of ABC + ATCQ. ABC only consumes less time than the multilevel tree variant of ABC + ATCQ for 32 and 64 colors.

When the basic ABC algorithm is applied, the fitness of each food source is computed by Algorithm 3. Nevertheless, ABC + ATCQ only applies said algorithm for the candidate food sources. Therefore, this shows that the use of ATCQ to compute the fitness of the sources exploited by the bees decreases the total execution time. The analysis of Figs. 7 and 8 indicates that each iteration of ABC + ATCQ is faster than each iteration of ABC + K-means or ABC. The reason is that ABC + ATCQ takes advantage of the tree structure to compute the fitness of a food source exploited by a bee. In this way, the fitness of the sources exploited by the bees is computed faster than that of the candidate sources.

To compare the previous results with those generated by ATCQ and K-means applied independently from ABC, Tables 5 and 6 include results for both algorithms. Since ATCQ is not an iterative method, Table 5 only presents one result for each palette size. For K-means, Table 6 shows results for the same iterations as Tables 1 to 4.

Obviously, when ATCQ or K-means are applied independently, the time consumed to reach a solution is smaller than when they are combined with ABC algorithm, since the combined algorithm requires the application of ATCQ or K-means for each employed bee at each iteration. Although ABC + ATCQ consumes more time than ATCQ, it also generates better images. Nevertheless, this improvement is not as clear when ABC + K-means and K-means are compared; the combined method is much more time-consuming than K-means is for the same number of iterations, but K-means can generate better images in the same or less amount of time.

When K-means and ABC + ATCQ are compared (for the 3-level tree variant), it is observed that the second method can generate images with less error (minimum and average) in less time (iteration 20 of K-means and iteration 5 or 10 of ABC + ATCQ are compared).

Table 7 includes the results obtained from some color quantization methods described in Section 2: Wu's method [64], Octree [20], Variance-based method [58], Neuquant [16] and Particle Swarm Optimization (PSO) [37]. This table also includes results for the LBG method [35], which is similar to K-means but applies a splitting method to select the initial centroids; this technique can obtain better images

Table 4 Results for palette size 256 (it: iteration of the algorithm; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	1	31.24	32.05	0.5	4786	31.40	32.02	0.4	7918	33.06	33.18	0.2	75331
	5	30.57	30.67	0.1	23817	30.46	30.61	0.1	41375	31.18	31.41	0.2	364165
	10	30.30	30.42	0.1	47765	30.33	30.40	0.1	80231	30.94	31.08	0.2	714200
	15	30.27	30.38	0.1	71341	30.15	30.34	0.1	117859	30.94	31.08	0.2	1068643
	20	30.22	30.33	0.1	94998	30.15	30.31	0.1	157511	30.94	31.01	0.1	1426797
Girl	1	20.40	21.83	0.9	4671	21.62	22.28	0.5	8209	22.10	23.21	0.7	71366
	5	18.78	19.67	0.7	24440	18.96	19.98	0.7	42710	19.99	20.48	0.6	352751
	10	18.60	19.36	0.7	49576	18.55	19.44	0.6	85296	19.59	20.03	0.6	716660
	15	18.44	19.11	0.7	74137	18.38	19.32	0.7	127083	19.56	20.03	0.6	1055603
	20	18.44	18.89	0.6	98859	18.38	19.24	0.8	168659	19.56	19.82	0.3	1398263
Plane	1	23.41	23.92	0.5	5026	22.37	24.23	1.0	7782	23.25	25.19	1.6	73752
	5	20.38	21.32	0.6	24676	20.45	21.46	0.5	39961	20.60	22.36	0.9	382237
	10	19.83	20.70	0.5	48904	19.68	20.78	0.5	81606	20.50	21.97	0.7	745177
	15	19.83	20.54	0.5	72765	19.51	20.40	0.4	122450	20.50	21.70	0.5	1105916
	20	19.79	20.29	0.4	96996	19.49	20.28	0.4	163926	20.50	21.50	0.5	1461037
Peppers	1	56.61	58.06	1.1	5260	56.28	57.88	1.2	7435	57.95	59.64	1.0	73494
	5	54.50	54.89	0.2	24757	54.36	54.77	0.3	40830	55.20	55.72	0.4	361332
	10	54.20	54.37	0.1	48608	53.96	54.28	0.2	80003	54.61	55.11	0.4	713849
	15	54.08	54.24	0.1	73396	53.88	54.11	0.2	120005	54.55	54.98	0.4	1074316
	20	54.08	54.17	0.1	96891	53.81	54.07	0.1	159365	54.37	54.92	0.4	1421057
Lake	1	58.75	61.12	1.3	4865	59.65	61.27	1.2	8523	59.92	62.62	1.4	76465
	5	56.29	57.06	0.6	24710	56.62	57.10	0.3	40648	57.45	58.31	0.6	360733
	10	55.33	55.69	0.2	49808	55.45	55.89	0.2	81422	56.43	57.06	0.4	717604
	15	55.09	55.32	0.2	74975	54.82	55.34	0.3	122617	56.43	56.92	0.3	1080701
	20	54.89	55.11	0.1	100783	54.67	55.16	0.2	163000	56.43	56.81	0.2	1432764
Mandrill	1	99.78	101.97	1.2	4627	99.51	101.27	1.2	8180	101.39	103.70	1.7	72940
	5	98.12	98.46	0.3	24477	98.01	98.44	0.3	41115	97.70	98.17	0.4	360779
	10	97.86	98.06	0.2	48373	97.70	97.99	0.2	82069	97.26	97.58	0.2	731841
	15	97.56	97.96	0.2	72308	97.67	97.88	0.1	122947	97.26	97.58	0.2	1110974
	20	97.56	97.85	0.2	97142	97.53	97.83	0.2	164035	97.26	97.55	0.2	1466731
EPSZ	1	24.20	24.59	0.3	4924	23.85	24.71	0.5	8060	25.18	26.17	0.7	66849
	5	23.46	23.65	0.1	25193	23.37	23.53	0.1	40472	24.05	24.46	0.3	360905
	10	23.28	23.45	0.1	49279	23.19	23.33	0.1	81318	23.93	24.16	0.1	733171
	15	23.16	23.38	0.1	73282	23.14	23.28	0.1	121669	23.93	24.11	0.1	1108603
	20	23.16	23.34	0.1	98420	23.14	23.25	0.1	162512	23.93	24.06	0.1	1474454
Machine	1	16.30	17.08	0.5	4756	16.26	16.99	0.5	8207	17.90	18.46	0.4	69952
	5	15.58	15.99	0.3	23943	15.68	16.02	0.2	40296	16.82	17.01	0.2	352291
	10	15.43	15.75	0.2	47435	15.46	15.79	0.1	80673	16.69	16.77	0.1	698549
	15	15.43	15.65	0.2	71176	15.46	15.70	0.1	119447	16.52	16.71	0.1	1059331
	20	15.43	15.62	0.1	95222	15.46	15.69	0.1	159554	16.43	16.67	0.1	1415441

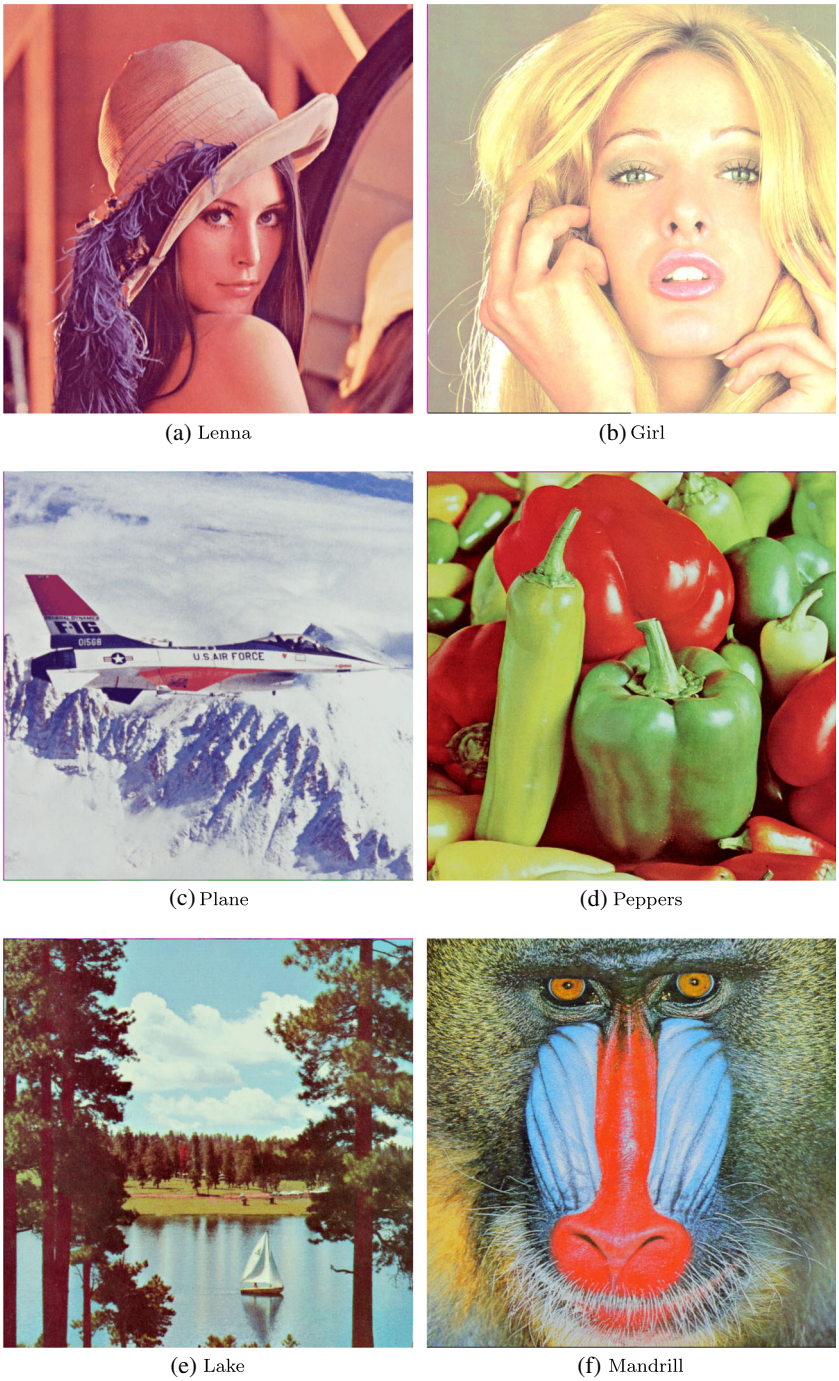
Table 4 (continued)

		ABC + ATCQ (3 levels)				ABC + ATCQ (multilevel)				ABC + K-means			
image	it.	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*
Text	1	9.03	9.34	0.2	5042	9.05	9.32	0.2	7756	10.22	10.60	0.3	74134
	5	8.61	8.73	0.1	25360	8.59	8.78	0.2	39756	9.70	9.86	0.1	353520
	10	8.50	8.60	0.1	49662	8.49	8.62	0.1	82243	9.47	9.70	0.2	707829
	15	8.50	8.56	0.1	74786	8.41	8.55	0.1	122018	9.45	9.63	0.2	1095978
	20	8.47	8.54	0.1	98744	8.38	8.52	0.1	164106	9.45	9.58	0.1	1473712
Hydrant	1	42.08	43.53	0.9	4448	40.95	42.92	1.1	8218	42.82	44.93	1.3	74140
	5	38.50	38.86	0.3	24541	37.93	38.64	0.4	41035	39.36	40.00	0.4	362485
	10	37.56	37.78	0.2	49426	37.36	37.72	0.3	83277	38.47	39.02	0.3	714028
	15	37.24	37.51	0.2	74776	37.18	37.53	0.1	123842	38.23	38.75	0.3	1057170
	20	37.21	37.45	0.2	99304	37.14	37.37	0.1	163584	38.23	38.68	0.3	1398705
Sign	1	13.28	13.70	0.4	4792	12.73	13.68	0.5	7759	14.27	15.11	0.5	74312
	5	11.77	11.98	0.2	24114	11.62	11.90	0.1	39425	12.85	13.26	0.3	343956
	10	11.44	11.62	0.1	48077	11.37	11.52	0.1	80318	12.61	12.95	0.2	693622
	15	11.31	11.49	0.1	71944	11.23	11.41	0.1	121364	12.54	12.88	0.2	1031279
	20	11.31	11.45	0.1	95787	11.22	11.38	0.1	163498	12.41	12.78	0.2	1378357
Bikes	1	22.87	23.74	0.7	4984	22.92	23.68	0.6	7692	22.12	24.33	1.2	81584
	5	19.70	20.28	0.3	25934	19.92	20.29	0.2	41514	20.25	21.43	0.5	381501
	10	18.83	19.47	0.4	51495	19.01	19.32	0.2	82996	19.92	20.73	0.4	724139
	15	18.76	19.02	0.2	77352	18.72	19.00	0.1	125057	19.92	20.63	0.4	1091732
	20	18.64	18.86	0.2	102155	18.47	18.75	0.2	165879	19.92	20.47	0.3	1452455

than K-means with less time consumption. The source code of those methods is available at [47]. To perform the tests, the sampling factor of Neuquant was set to 1, to obtain the best quantized images that this method can generate. Moreover, the parameters of PSO were established as follows: the inertia was set to 0.72; the social and the cognitive parameters were set to 1.49; the velocity was limited to the range [-5, 5] and the probability of application of K-means was set to 0.1 (these values were proposed in [37] to solve the color quantization problem by PSO); the number of particles was set to 5 (equal the number of food sources used by ABC + ATCQ and ABC + K-means) and the number of iterations of K-means was set to 5 (equal to the value used by ABC + K-means).

The results of Table 7 are compared with those of the ABC + ATCQ algorithm that generates a 3-level tree, since it consumes less time. It can be observed that the MSE values of this table are improved by those of iteration 1 of ABC + ATCQ in most cases (214 cases out of 288); most of the unimproved cases correspond to the same image (Plane). In addition, after a maximum of 10 iterations another 53 cases are improved. As expected, splitting methods are faster than the proposed method. As far as the clustering-based methods is concerned, Neuquant and LGB can obtain a quantized image in less time, but the quality of such image is worse in most cases. On the other hand, PSO consumes much more time to generate images with similar quality to those obtained by ABC + ATCQ.

The first iteration of ABC + ATCQ always obtains better images than Octree and Variance-based methods. Moreover, the images generated after a maximum of 10 iterations are better than those obtained by Wu's method for all problems and palette sizes, and for all cases, except one, when Neuquant is considered (this case corresponds to the Plane image with 128 colors). Regarding the LBG method, ABC + ATCQ obtains better images at iteration 1 for 30 out of 48 cases, and only 3 cases remain unimproved after iteration 10. Figure 9 shows the percentage of error reduction obtained by ABC + ATCQ at iteration 10 with respect to Wu's method, Octree, Variance-based



(a) Lenna (b) Girl

(c) Plane (d) Peppers

(e) Lake (f) Mandrill

Fig. 5 Best quantized image generated by ABC + ATCQ when a 3-level tree is used – 256 colors

method, Neuquant and LBG. This figure clearly shows that ABC + ATCQ obtains better images than the other methods in most cases. The best results are obtained when the proposed method is compared to Octree, since the percentage of error reduction is between 49.9% and 82.7%. On the contrary, the worst results are obtained when LBG is compared, since the improvement is less than 24.5% and three cases are not improved. It is also observed that all the unimproved cases correspond to the Plane image.

Although PSO can obtain better images than ABC + ATCQ in some cases (11 out of 48 cases), the difference in the execution time of both methods should be taken into account. It is clearly observed that PSO consumes much more time than ABC + ATCQ, and this can make the

(a) EPSZ (b) Machine

(c) Text (d) Hydrant

(e) Sign (f) Bikes

Fig. 6 Best quantized image generated by ABC + ATCQ when a 3-level tree is used – 256 colors

results of both methods not comparable. Figure 10 shows the percentage of error reduction obtained by ABC + ATCQ with respect to PSO. This figure compares the results of PSO at iteration 20 with those of ABC + ATCQ at iterations 1, 5, 10, 15 and 20. The figure clearly shows that the percentage of error reduction is better as the iterations of ABC + ATCQ progress. It is observed that the images generated at iteration 20 of PSO are always better than those obtained at iteration 1 of ABC + ATCQ (except for Text image with 256 colors), but the results corresponding to more iterations of ABC + ATCQ generate better images than PSO in some cases (21 cases at iteration 5, 31 cases at iteration 10, 35 cases at iteration 15 and 37 cases at iteration 20), and the differences in the error for the 11 cases that

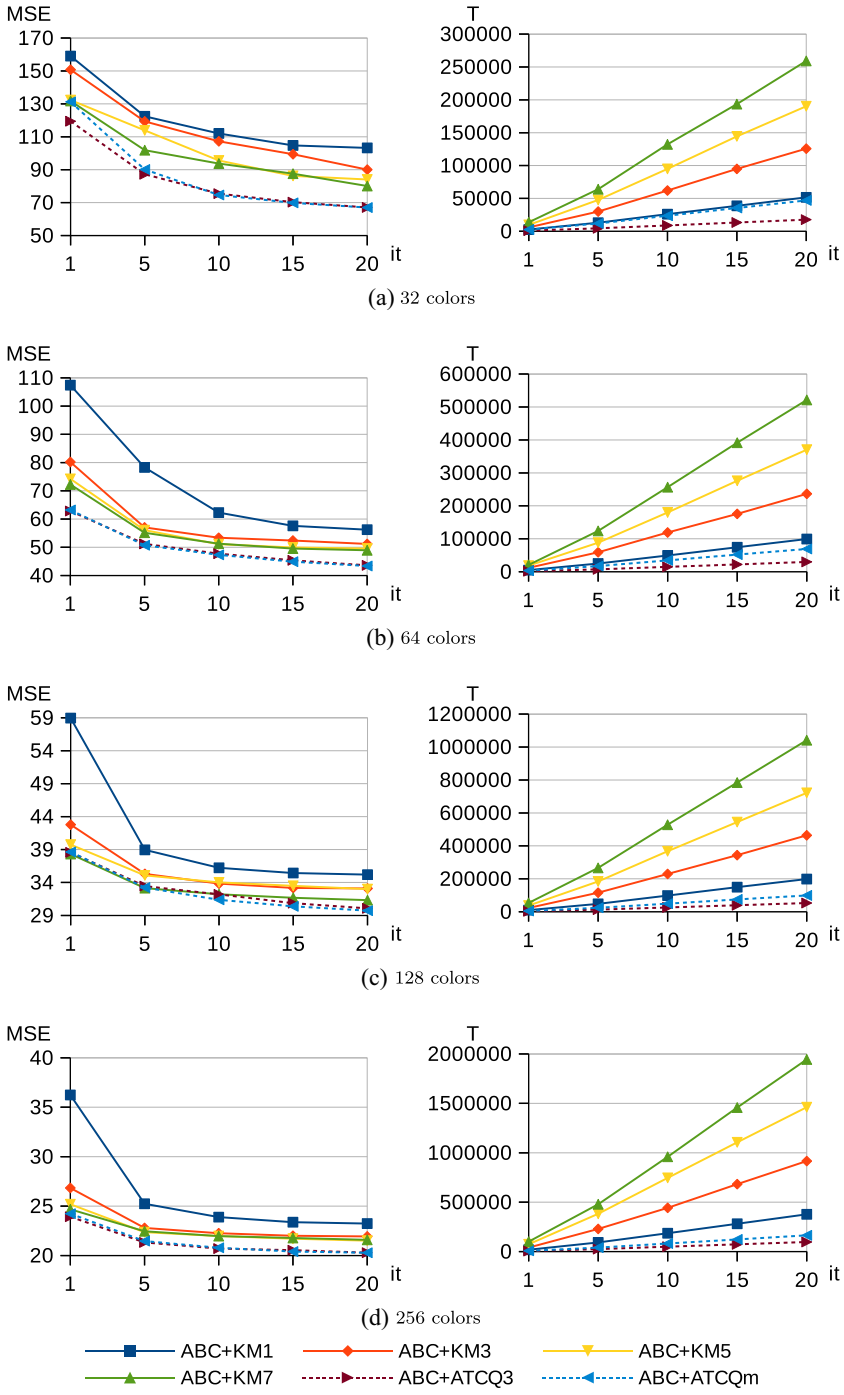


Fig. 7 Average MSE and execution time (milliseconds) for ABC + K-means applied to Plane image with different number of K-means iterations (ABC+KM1: 1 iteration; ABC + KM3: 3 iterations; ABC + KM5: 5 iterations; ABC + KM7: 7 iterations) compared to both variants of ABC + ATCQ (ABC + ATCQ3: variant that considers a 3-level tree; ABC + ATCQm: variant that considers a multilevel tree) (it: iteration of the algorithm)

remain unimproved after iteration 20 are small (between 0.2% and 2.9%). In addition, PSO consumes more time than ABC + ATCQ to perform the same number of iterations. Therefore, ABC + ATCQ can generate the quantized images faster than PSO and the quality of these images is very similar, which makes ABC + ATCQ better than PSO.

To analyze the statistical significance of the improvement obtained by ABC + ATCQ with respect to the other methods, the Wilcoxon test was applied [15]. This test compares two methods to determine that there is no significant difference between their results. In this case, the test was applied to compare the average MSE results of the 3-level variant of ABC + ATCQ and those of each of

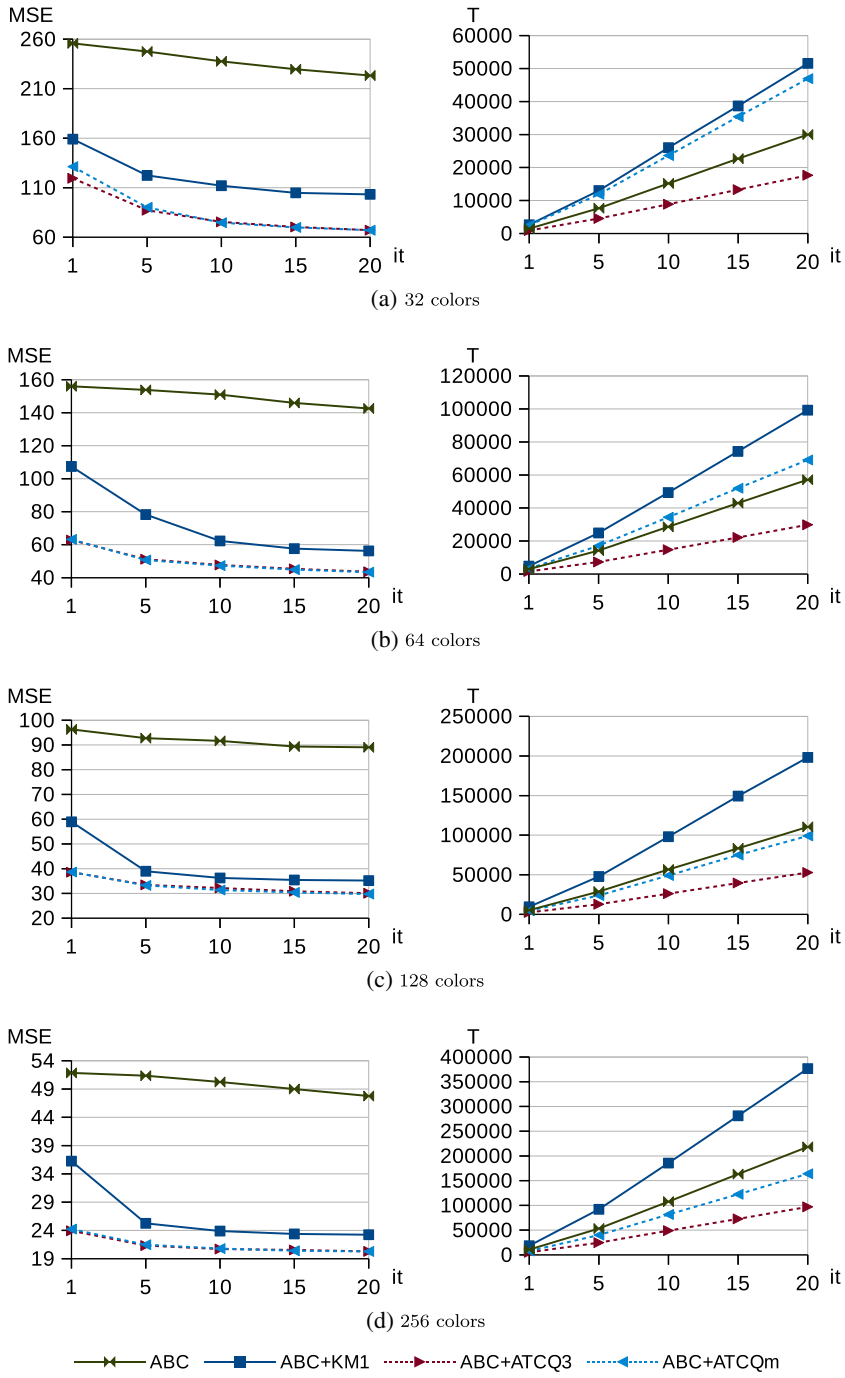


Fig. 8 Average MSE and execution time (milliseconds) for the basic ABC algorithm (ABC), ABC + K-means with 1 iteration of K-means (ABC+KM1) and both variants of ABC + ATCQ (ABC + ATCQ3: variant that considers a 3-level tree; ABC + ATCQm: variant that considers a multilevel tree) (it: iteration of the algorithm) – Plane image

the other methods. For the iterative methods, the results of iteration 20 were considered. Table 8 shows the results with a significance level equal to 0.05. The p-value obtained in all cases indicates that the differences between each pair of methods compared are significant. In addition, the sums of ranks show that ABC + ATCQ is significantly better than the other methods. Therefore, the results of the Wilcoxon test confirm the conclusions presented in this section.

The previous discussion compares ABC + ATCQ to classical color quantization methods, used by many authors to analyze the quality of new methods [8–11, 21, 26, 39, 41, 43–45, 51, 52, 54, 57, 63, 67]. However, to conclude the analysis of the proposed method, the results presented in this article are compared to those published in recent articles related to color quantization. The comparison considers several recent articles that use some of the test images

Table 5 ATCQ results (q: colors of the quantized palette; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		ATCQ (3-levels tree)				ATCQ (multilevel tree)			
image	q	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	32	163.55	193.82	27.27	60	167.00	202.09	27.87	215
	64	96.16	123.82	31.04	95	92.04	125.15	34.85	280
	128	58.87	95.92	47.11	146	58.99	96.60	46.91	331
	256	38.78	88.61	53.21	174	38.52	84.89	52.10	380
Girl	32	132.69	163.61	24.36	62	134.16	171.84	25.37	239
	64	97.80	117.48	19.73	90	84.52	112.53	16.62	268
	128	55.38	87.40	35.72	129	50.33	84.35	29.60	324
	256	28.07	74.12	45.88	169	28.07	71.00	42.97	371
Plane	32	116.71	157.62	38.58	57	102.56	159.90	50.91	249
	64	64.15	98.13	42.61	80	62.63	88.60	19.89	283
	128	35.49	79.53	51.85	101	36.09	64.41	26.18	327
	256	24.62	75.05	55.39	115	23.16	58.53	33.44	335
Peppers	32	334.17	393.59	43.45	61	306.95	568.16	362.89	232
	64	188.96	282.20	114.48	94	173.91	290.42	102.09	269
	128	114.89	243.20	146.97	142	110.65	225.46	126.45	327
	256	75.58	232.02	157.22	176	74.19	207.06	138.85	391
Lake	32	268.93	371.64	88.15	64	268.07	483.18	264.46	214
	64	182.78	214.68	45.17	101	182.97	216.92	30.14	278
	128	122.82	171.63	69.74	151	122.85	154.99	39.42	351
	256	78.16	155.30	82.42	204	75.30	131.59	54.50	413
Mandrill	32	538.49	2319.32	3460.53	54	512.05	1226.41	1809.00	221
	64	362.35	2217.82	3513.70	76	331.39	865.39	965.44	250
	128	220.56	2151.45	3549.54	111	201.36	832.85	1447.25	299
	256	147.20	2128.12	3562.55	145	132.46	1037.90	2165.73	333
EPSZ	32	187.43	244.55	97.26	57	146.45	204.95	59.16	251
	64	99.05	172.53	136.91	91	87.04	138.41	71.69	294
	128	59.17	151.99	151.15	122	57.82	120.03	85.07	356
	256	38.68	146.14	155.80	164	36.36	108.96	83.33	404
Machine	32	128.77	187.52	82.86	52	117.43	210.31	159.58	224
	64	65.05	133.02	111.63	84	60.07	83.66	13.75	267
	128	39.75	108.07	125.08	114	40.09	60.64	24.24	325
	256	22.95	100.64	129.91	144	23.39	52.39	33.55	376
Text	32	43.32	57.11	13.06	57	42.23	62.33	18.67	243
	64	26.25	41.19	18.38	80	25.95	39.71	14.96	282
	128	17.58	38.41	21.24	93	16.46	36.01	18.32	317
	256	15.70	38.03	21.71	103	12.72	34.86	18.81	340
Hydrant	32	302.76	395.56	122.22	63	246.37	400.31	117.84	210
	64	159.43	230.44	71.81	99	140.16	238.40	83.79	263
	128	102.08	153.27	67.96	122	81.07	140.48	54.07	339
	256	57.46	133.94	83.78	165	51.15	115.16	64.71	420

Table 5 (continued)

		ATCQ (3-levels tree)				ATCQ (multilevel tree)			
image	q	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Sign	32	109.56	144.80	24.39	60	99.27	168.68	62.00	222
	64	64.94	81.55	30.96	86	58.95	86.84	27.93	273
	128	27.90	61.11	44.93	114	26.57	59.88	33.91	331
	256	16.57	54.62	49.76	147	15.78	51.99	40.46	355
Bikes	32	147.56	198.60	69.23	59	145.05	215.06	68.14	216
	64	82.18	107.99	29.07	95	82.61	113.70	27.94	282
	128	48.27	77.33	43.83	137	45.51	76.58	43.68	361
	256	28.07	70.03	50.25	169	26.21	67.07	47.08	447

Table 6 K-means results (it: iteration of the algorithm; $min.$: minimum MSE; $av.$: average MSE; $dev.$: standard deviation of MSE; T: average time (milliseconds))

		32 colors				64 colors				128 colors				256 colors			
image	it.	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T	$min.$	$av.$	$dev.$	T
Lenna	1	154.70	195.04	21.6	314	107.19	121.72	10.2	614	63.40	70.63	3.6	1210	42.51	44.27	1.1	2364
	5	126.77	140.20	12.1	1556	80.91	85.35	3.7	3036	50.18	53.14	1.4	6000	34.09	35.00	0.6	11852
	10	122.53	131.33	7.4	3120	77.70	79.78	1.8	6066	48.65	50.28	0.8	11968	32.71	33.40	0.4	23755
	15	121.25	128.02	6.2	4693	75.93	77.66	1.5	9091	47.99	49.22	0.7	17901	32.10	32.70	0.4	35566
	20	119.88	125.92	5.4	6242	75.02	76.62	1.3	12112	47.58	48.68	0.7	23864	31.85	32.32	0.4	47326
Girl	1	175.74	237.08	38.5	339	113.04	159.85	30.0	641	74.04	120.63	28.0	1310	48.14	87.31	18.3	2565
	5	120.60	139.12	24.7	1572	55.18	83.48	31.1	3017	36.30	57.37	14.8	6099	22.96	25.63	2.0	11988
	10	87.85	118.61	24.3	3159	52.71	60.45	13.3	5988	34.23	37.48	2.6	12006	21.72	23.31	0.7	23724
	15	84.84	95.52	11.2	4719	52.00	55.85	7.7	8952	33.53	35.60	1.5	17982	21.06	22.50	0.7	35394
	20	84.09	90.80	8.4	6261	51.48	52.61	1.1	11908	33.14	34.81	1.3	23938	20.69	22.06	0.8	47092
Plane	1	160.25	264.10	90.3	315	90.01	139.57	26.5	607	75.02	90.93	11.0	1200	44.63	66.43	15.6	2374
	5	125.29	162.22	27.6	1574	52.97	89.07	21.8	3023	38.70	47.95	8.5	5876	25.18	31.39	8.8	11672
	10	104.58	145.98	22.2	3137	50.68	82.94	20.2	6007	37.08	39.95	2.4	11715	23.85	26.69	1.4	23330
	15	102.94	141.59	22.1	4702	49.68	71.21	20.3	8997	35.69	38.70	2.3	17559	23.43	25.47	1.2	34972
	20	101.23	138.41	22.0	6273	49.06	67.55	17.9	11987	34.95	38.07	2.2	23411	23.20	24.99	1.1	46544
Peppers	1	323.77	399.93	54.4	311	224.58	239.09	10.5	614	129.28	153.71	14.4	1188	84.15	101.92	13.2	2363
	5	260.05	277.70	13.0	1543	142.96	158.83	13.0	3022	94.28	98.13	3.4	5907	60.07	63.26	2.9	11745
	10	237.89	255.23	13.6	3090	139.08	144.86	5.4	6037	89.29	92.68	4.0	11761	57.68	58.97	0.8	23466
	15	235.97	247.18	10.3	4708	137.18	142.41	5.1	9058	87.39	90.75	4.1	17594	56.94	57.66	0.5	35139
	20	234.58	243.83	9.9	6259	135.95	141.07	4.9	12103	86.27	89.63	4.2	23449	56.46	57.01	0.5	46823
Lake	1	278.62	310.94	33.5	325	190.48	213.04	26.2	667	122.13	139.84	8.6	1169	80.90	93.75	6.2	2342
	5	228.08	243.57	12.0	1600	149.75	159.80	7.7	3320	94.48	102.70	4.5	5875	64.47	67.50	2.5	11704
	10	214.49	228.62	13.6	3172	140.98	150.73	5.6	6456	90.59	97.16	3.6	11692	61.12	63.73	2.2	23437
	15	208.80	221.17	14.2	4749	138.26	146.66	4.9	9542	89.17	94.38	3.0	17523	59.74	62.04	1.9	35151
	20	206.06	216.65	13.5	6359	137.15	143.91	4.0	12643	88.51	92.69	2.5	23374	58.93	60.99	1.7	46845

Table 6 (continued)

		32 colors				64 colors				128 colors				256 colors			
image	it.	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*	*min.*	*av.*	*dev.*	*T*
Mandrill	1	487.88	533.68	38.5	310	314.92	339.04	13.1	606	200.07	210.13	6.2	1178	129.71	133.57	2.4	2359
	5	396.57	412.50	10.5	1553	253.85	260.39	4.9	3071	160.92	164.79	2.9	5888	104.68	106.01	1.0	11785
	10	379.39	391.80	6.7	3118	244.94	247.44	2.6	6115	155.79	158.03	1.5	11744	100.50	101.81	0.8	23500
	15	377.14	385.73	5.4	4675	241.51	243.48	1.6	9141	154.42	155.65	0.9	17612	99.27	100.24	0.7	35261
	20	376.80	382.98	5.0	6228	239.94	241.50	1.0	12182	153.53	154.42	0.7	23450	98.65	99.39	0.6	47041
EPSZ	1	165.96	209.56	51.2	317	90.41	110.12	19.1	644	55.19	59.70	2.8	1192	34.38	36.40	2.2	2353
	5	126.48	135.70	6.5	1589	71.19	74.21	2.2	3149	41.43	43.61	1.6	6011	26.94	27.56	0.5	11757
	10	119.12	125.65	4.5	3184	66.07	68.86	1.7	6242	39.78	41.02	1.2	11917	25.53	26.13	0.4	23589
	15	117.05	121.88	3.7	4779	64.79	67.05	1.5	9439	38.98	40.09	1.1	17968	25.14	25.62	0.3	35328
	20	116.52	120.26	3.3	6368	63.90	66.02	1.3	12540	38.57	39.58	0.9	23836	24.80	25.29	0.3	47053
Machine	1	107.29	123.60	13.8	313	63.72	70.63	3.2	608	38.24	41.00	2.1	1167	23.90	25.14	0.8	2388
	5	80.47	87.37	5.4	1541	47.94	51.19	2.8	3011	30.12	31.31	0.9	5827	18.70	19.28	0.6	11809
	10	78.58	82.84	5.2	3083	45.60	48.70	2.7	6027	28.94	29.72	0.9	11658	17.50	18.15	0.5	23570
	15	76.20	80.65	5.2	4619	44.64	47.74	2.7	9034	28.13	29.06	1.0	17489	17.02	17.74	0.4	35326
	20	74.72	79.57	5.0	6162	44.25	47.13	2.6	12068	27.63	28.62	0.9	23291	16.87	17.53	0.4	47070
Text	1	57.93	78.65	23.5	308	30.69	43.92	12.1	608	21.62	25.49	3.6	1184	14.25	15.11	0.8	2394
	5	38.87	48.90	7.4	1547	24.17	28.93	4.0	3011	15.78	18.16	2.0	5867	10.79	11.34	0.4	11829
	10	37.43	44.96	5.9	3119	22.63	26.38	2.5	6004	15.15	17.03	1.7	11781	10.17	10.74	0.4	23793
	15	36.88	43.42	5.0	4687	22.15	25.33	2.0	9023	14.71	16.45	1.5	17613	9.99	10.49	0.4	35640
	20	36.72	42.44	4.5	6246	21.74	24.50	1.6	12042	14.51	16.10	1.4	23431	9.92	10.35	0.4	47431
Hydrant	1	253.98	326.58	49.8	306	148.52	183.47	22.9	610	88.09	112.62	15.1	1161	62.26	68.24	4.6	2346
	5	192.85	210.00	15.6	1525	116.89	125.18	7.1	3072	71.07	77.34	5.4	5775	46.50	48.86	2.3	11665
	10	181.05	191.73	12.8	3047	110.06	115.89	5.0	6135	67.22	70.91	3.0	11546	43.33	44.97	1.8	23274
	15	175.51	185.85	10.7	4568	107.38	112.00	4.5	9160	65.31	68.00	1.7	17328	42.00	43.38	1.5	35006
	20	174.19	183.36	9.3	6107	106.10	109.87	4.2	12191	64.10	66.58	1.4	23124	41.28	42.45	1.3	46659
Sign	1	105.93	128.31	31.5	309	56.16	65.70	6.4	616	33.62	41.17	5.6	1203	20.87	25.14	2.5	2349
	5	73.87	81.60	8.1	1540	40.59	44.15	4.0	3077	23.86	25.80	2.0	5939	14.78	16.29	0.9	11605
	10	68.71	75.11	6.8	3086	37.24	39.78	2.1	6186	22.06	23.47	1.1	11907	13.84	14.97	0.6	23166
	15	66.77	72.88	7.0	4621	36.20	38.35	1.8	9342	21.43	22.48	0.9	17863	13.51	14.38	0.5	34762
	20	65.17	71.40	6.9	6171	35.80	37.45	1.6	12394	20.94	22.00	0.8	23783	13.31	14.03	0.4	46382
Bikes	1	157.35	202.73	38.1	306	98.80	119.00	17.0	597	62.97	76.73	16.0	1183	38.57	41.37	3.7	2372
	5	117.93	134.21	10.0	1534	67.26	77.16	5.7	2982	41.89	46.41	3.9	5879	26.04	27.04	0.8	11808
	10	109.30	122.36	7.7	3056	63.94	70.08	4.2	5967	39.40	41.38	2.2	11745	23.82	24.51	0.8	23582
	15	105.87	117.35	6.5	4599	61.96	67.38	3.5	8975	37.98	39.47	1.9	17614	22.89	23.52	0.7	35412
	20	104.01	114.25	5.6	6129	60.62	65.60	2.8	11955	36.84	38.23	1.7	23482	22.40	22.92	0.5	47162

considered in this article. In this case the results published for the different methods are considered because the source code of these methods is not available on the web. Only the quality of the quantized images is compared, since each method has been tested on a different machine and therefore the execution time is not comparable. In addition, some articles do not include information about execution times. Table 9 summarizes the information of the methods selected

Table 7 MSE and average time (milliseconds) for some color quantization methods: Wu's method (WU), Octree (OC), Variance-based method (VB), Neuquant (NQ), LGB method (LBG) and Particle Swarm Optimization (PSO) (q: colors of the quantized palette)

		WU		OC		VB		NQ		LBG		PSO			
image	q	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	$min.$	$av.$	$dev.$	T
Lenna	32	158.61	105	482.03	177	203.07	333	152.13	763	128.81	1155	118.74	119.56	0.6	111870
	64	99.16	106	212.92	175	135.86	381	85.60	831	78.46	1572	73.26	73.79	0.6	218572
	128	61.79	105	140.53	188	94.68	438	53.73	964	50.42	2358	46.85	47.34	0.4	444111
	256	39.53	105	74.12	190	69.41	490	34.88	1134	35.42	3237	31.05	31.33	0.2	861795
Girl	32	107.55	102	477.64	173	232.43	209	123.89	738	95.27	1065	82.30	82.83	0.5	111038
	64	63.03	105	163.08	174	129.43	250	66.32	801	53.50	1453	49.65	50.19	0.5	226080
	128	36.84	103	89.86	182	82.22	273	38.30	907	33.93	2173	30.06	31.74	0.9	452348
	256	23.59	107	50.79	184	54.42	294	23.86	1144	21.40	3988	19.04	19.21	0.1	912204
Plane	32	85.45	102	342.23	168	123.56	305	123.70	744	86.98	1036	65.13	70.72	5.0	119450
	64	51.33	105	225.98	168	80.73	353	57.57	789	45.68	1400	41.31	42.55	0.8	226010
	128	32.60	104	133.59	175	52.60	398	29.71	883	27.26	2014	27.49	29.22	1.0	434841
	256	21.66	103	52.31	182	36.85	459	21.09	1088	18.01	3062	19.68	20.17	0.3	897188
Peppers	32	279.27	105	777.14	166	451.10	410	283.69	781	276.69	1073	229.08	230.85	1.4	115199
	64	165.37	107	495.51	172	304.21	468	166.37	851	152.75	1448	134.50	138.10	3.7	234004
	128	102.31	102	308.76	177	212.68	535	95.69	953	97.97	2269	83.84	84.39	0.4	430605
	256	66.08	111	156.24	181	147.18	605	63.08	1161	64.28	3061	54.56	55.46	0.7	865466
Lake	32	249.81	106	922.04	172	357.26	456	274.45	777	252.22	1026	202.26	203.43	0.9	116859
	64	161.34	104	466.14	170	251.70	538	164.26	831	169.43	1343	130.47	130.89	0.6	224287
	128	102.55	104	198.49	178	170.97	644	100.88	934	103.41	2272	84.70	85.34	0.6	438649
	256	66.47	106	159.21	178	120.10	770	65.70	1170	69.86	3322	54.96	55.17	0.2	888006
Mandrill	32	468.39	108	1094.11	174	531.03	472	456.13	771	425.54	1116	374.64	376.42	1.8	115720
	64	288.33	105	576.19	180	346.58	506	272.25	844	262.97	1598	235.49	236.39	1.0	223442
	128	186.33	107	357.13	177	248.02	585	168.22	942	171.28	2119	151.16	151.94	0.6	426874
	256	118.65	106	195.82	184	181.94	695	109.34	1145	112.76	3111	97.23	97.95	0.6	863318
EPSZ	32	149.84	116	539.37	187	182.44	314	141.71	944	124.29	1304	114.28	115.18	0.6	117754
	64	85.95	111	280.23	190	113.99	372	72.76	892	71.99	1627	61.82	62.66	0.7	221641
	128	52.04	112	124.29	194	78.97	477	43.26	998	42.66	2142	37.37	37.78	0.2	451469
	256	32.00	111	55.06	201	57.37	508	27.45	1397	26.67	3346	23.61	24.20	0.3	852636
Machine	32	98.40	109	371.64	183	120.22	356	96.94	812	77.06	1004	73.11	73.95	1.4	114551
	64	57.81	109	136.94	193	82.35	422	53.35	886	48.96	1370	42.59	43.31	0.6	226829
	128	35.59	110	85.40	197	60.15	543	30.63	965	28.44	1896	25.88	26.29	0.3	436022
	256	22.02	110	49.07	202	41.62	587	19.73	1263	17.56	3111	16.41	16.63	0.2	865967
Text	32	45.51	110	104.38	191	60.96	218	55.06	858	37.15	1245	34.60	35.43	0.7	115043
	64	27.26	109	72.16	196	42.87	260	28.58	1057	22.31	1446	20.68	21.34	0.6	226545
	128	18.76	119	40.46	200	33.12	297	15.95	1054	14.48	2260	13.81	14.19	0.3	459097
	256	14.05	112	27.43	207	26.48	351	10.06	1236	9.73	3183	9.22	9.45	0.2	861340
Hydrant	32	218.21	120	491.45	182	367.05	333	235.63	837	181.33	1243	169.96	171.69	2.3	110532
	64	131.49	111	280.76	188	236.15	427	127.00	920	112.23	1361	101.78	103.24	1.2	221006
	128	79.91	113	208.59	192	175.19	497	74.10	1063	69.11	1979	61.76	62.81	0.6	420866
	256	48.62	107	120.00	203	118.85	574	45.88	1368	42.00	3379	38.51	39.37	0.5	845297
Sign	32	82.07	114	308.53	187	113.41	273	98.36	766	72.67	1133	59.98	61.01	0.9	112814
	64	45.21	110	121.85	191	62.61	337	52.50	816	43.18	1440	33.08	33.62	0.5	241044
	128	26.98	110	57.33	199	38.48	433	26.16	928	24.76	2298	19.60	19.80	0.2	428347
	256	17.23	111	42.01	210	25.79	442	14.43	1117	15.40	3240	12.27	12.52	0.2	853010

Table 7 (continued)

		WU		OC		VB		NQ		LBG		PSO			
image	q	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	$min.$	$av.$	$dev.$	T
Bikes	32	131.08	111	409.62	186	162.72	264	140.32	835	121.56	1303	103.06	105.18	1.9	113390
	64	77.65	110	193.87	188	103.25	313	75.52	925	63.97	1483	57.21	58.67	1.0	226147
	128	44.62	111	129.45	194	65.29	401	39.23	1020	37.28	2008	33.27	34.32	0.7	459890
	256	26.12	112	77.85	200	41.30	429	23.65	1231	20.82	3120	20.06	20.73	0.5	857927

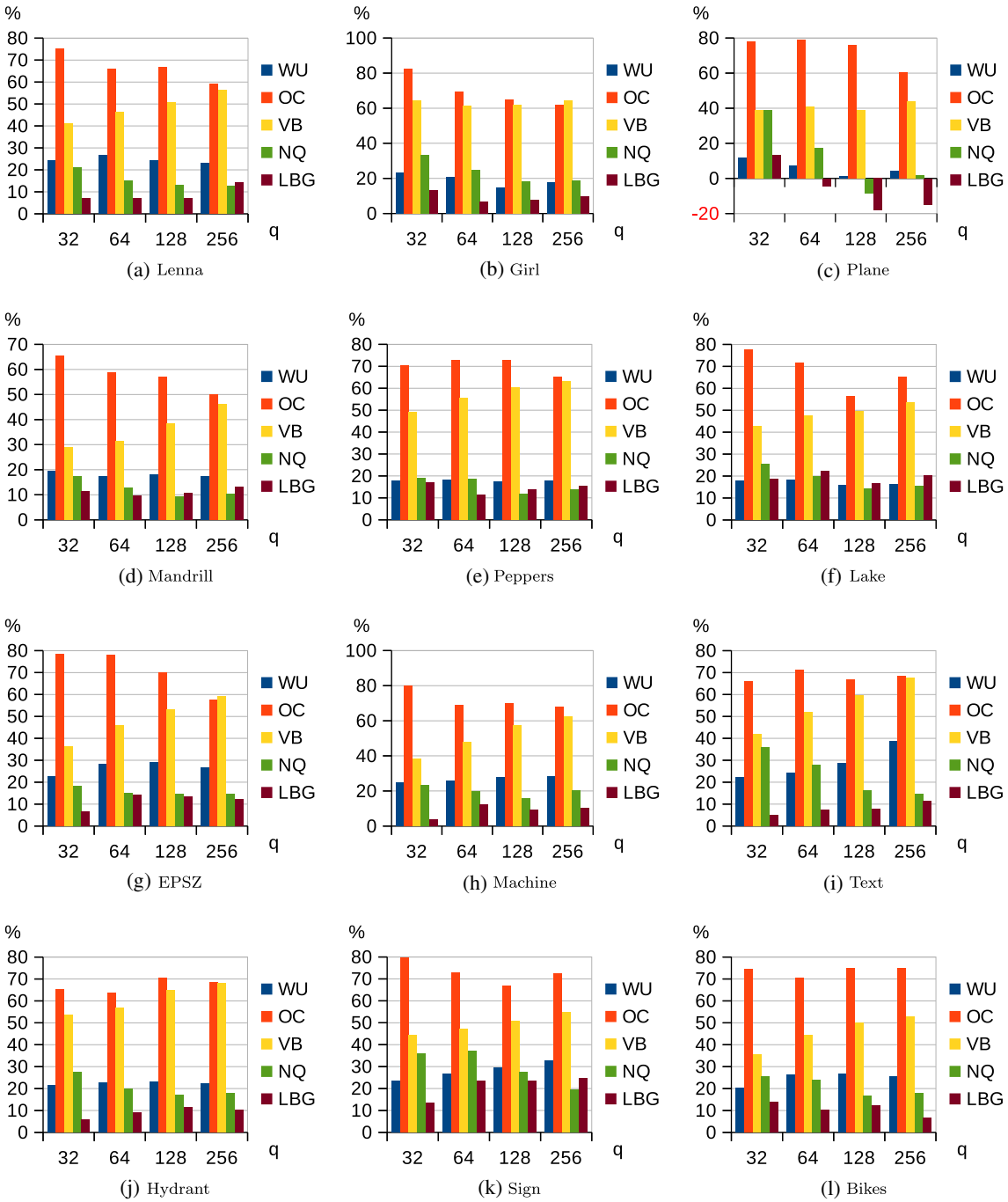


Fig. 9 Percentage of error reduction obtained by ABC + ATCQ (iteration 10) with respect to Wu's method (WU), Octree (OC), Variance-based method (VB), Neuquant (NQ), and LBG (q: colors of the quantized palette)

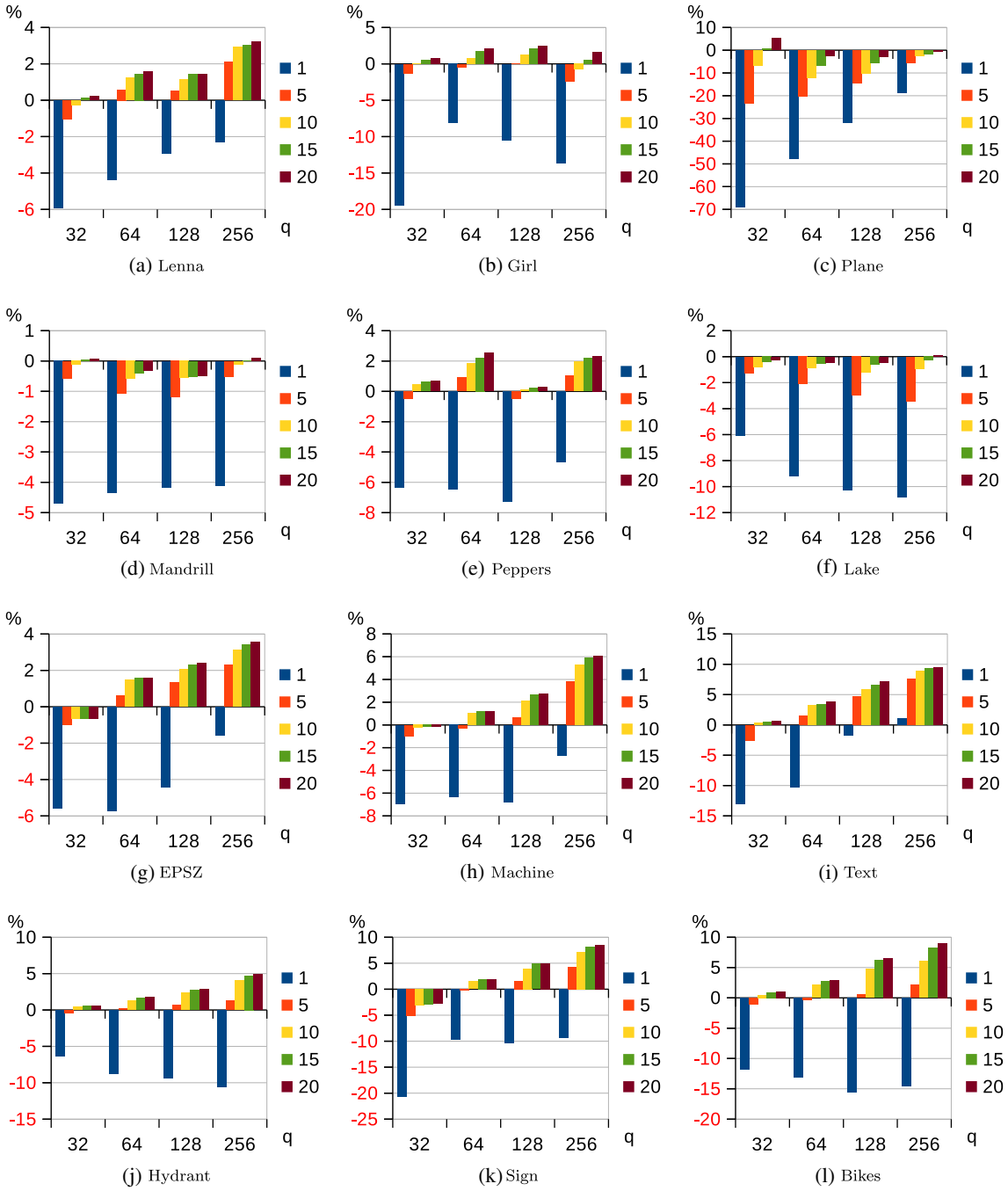


Fig. 10 Percentage of error reduction when the results of several iterations of ABC + ATCQ are compared to those of PSO at iteration 20 (q: colors of the quantized palette)

for the comparison: the color quantization method applied, the article that includes the results of the method used for the comparison and the label assigned to identify each method.

All the methods compared were described in Section 2, except the Hierarchical self-organizing feature map (M2). Said method is a compression technique that uses a tree structure where each level of the tree includes several independent SOMs [6]. The first level of the tree includes a SOM that is trained with all the items in the data set. When the training of this SOM concludes, a SOM is created in the second level of the tree for each neuron that was associated with more than one input pixel. Then, each SOM of the second level is trained with the subset of data that was associated with the parent neuron. The same process is applied to each level of the tree.

There are two results corresponding to the FS-SOM method (M4 and M5). M5 corresponds to the results presented by the author of the method, but M4 is also

Table 8 Results of the Wilcoxon test that compares ABC + ATCQ (3 levels) to other color quantization methods (Z-value: value of the test statistic; p-value: probability value corresponding to the Z-value; $sum+$: sum of positive ranks; $sum-$: sum of negative ranks)

Method	Z-value	p-value	$sum+$	$sum-$
Wu	-6.0308	0	0	1176
Octree	-6.0308	0	0	1176
Variance-based	-6.0308	0	0	1176
Neuquant	-6.0206	0	1	1175
LBG	-5.8462	0	18	1158
PSO	-4.0462	0	193.5	982.5
K-means	-6.0308	0	0	1176
ATCQ (3 levels tree)	-6.0308	0	0	1176
ATCQ (multilevel tree)	-6.0308	0	0	1176
ABC+K-means	-4.4975	0	149.5	1026.5

included because it uses the six images of the test set. The results associated with M4 correspond to a simpler and faster version of the FS-SOM proposed in [2] to apply said algorithm to image segmentation. In this version, initial random weights are used, the input pixels are randomly selected, the neighborhood function is the original of the SOM network and the weights of the non-winning neurons are replaced by random pixels of the original image.

Figure 11 shows the percentage of error reduction obtained by ABC + ATCQ with respect to the eleven methods identified in Table 9. This figure compares the results of iteration 20 of the ABC + ATCQ algorithm that generates a 3-level tree. It should be noted that the architecture of the Hierarchical self-organizing feature map network does not allow to obtain results for palettes with 32 and 128 colors [42]. On the other hand, there are no results of all the methods for Girl, Lake and Plane images.

It can be observed that the images obtained by ABC + ATCQ are always better than those obtained by M1, M2, M3, M4, M5, M8 and M9. Moreover, ABC + ATCQ is worse than M7 only for the Plane image with 256 colors, but the difference is small (1.45%). M10 obtains better images than ABC + ATCQ for 4 out of 16 cases, but the difference is very small (between 0.08% and 0.66%). Similar results are obtained when considering M11, since this method improves 6 out of 16 cases with a percentage of error reduction that varies between 0.06% and 1.61%. On the contrary, M6 improves the results of ABC + ATCQ for 10 out of 16 cases; the reduction in the error for these 10 cases is between 1.08% and 10.33%.

The best improvement is obtained when ABC + ATCQ is compared to M1, M2 and M8, since the improvement ranges in between 35% and 80%. The improvement is also significant for M4 (between 15% and 75%). On the other hand, it should be noted that the results of ABC + ATCQ at iteration 1 always improve the results of M1, M2, M4, M8 and M9.

It can be concluded that the proposed method can obtain better images than most of the eleven methods analyzed. When the six SOM-based methods are considered, a significant improvement is obtained for three methods, but for one of them worse results are obtained in some cases. Although the execution time of the last group of methods has not been compared, it must be taken into account that SOM-based solutions are usually time-consuming.

Table 9 Recent color quantization methods that are compared to ABC + ATCQ (*label*: name used to refer to the method in the text; *method*: method applied; *article*: article that includes the results compared)

Label	Method	Article
M1	basic Self-organizing feature map	[42]
M2	Hierarchical self-organizing feature map	[42]
M3	Growing neural gas	[42]
M4	Frequency sensitive SOM	[42]
M5	Frequency sensitive SOM	[11]
M6	Color importance-based SOM	[44]
M7	splitting method proposed in [54]	[54]
M8	splitting method proposed in [26]	[26]
M9	Variance-cut	[10]
M10	Weighted sort-means	[10]
M11	Adaptive distributing units	[10]

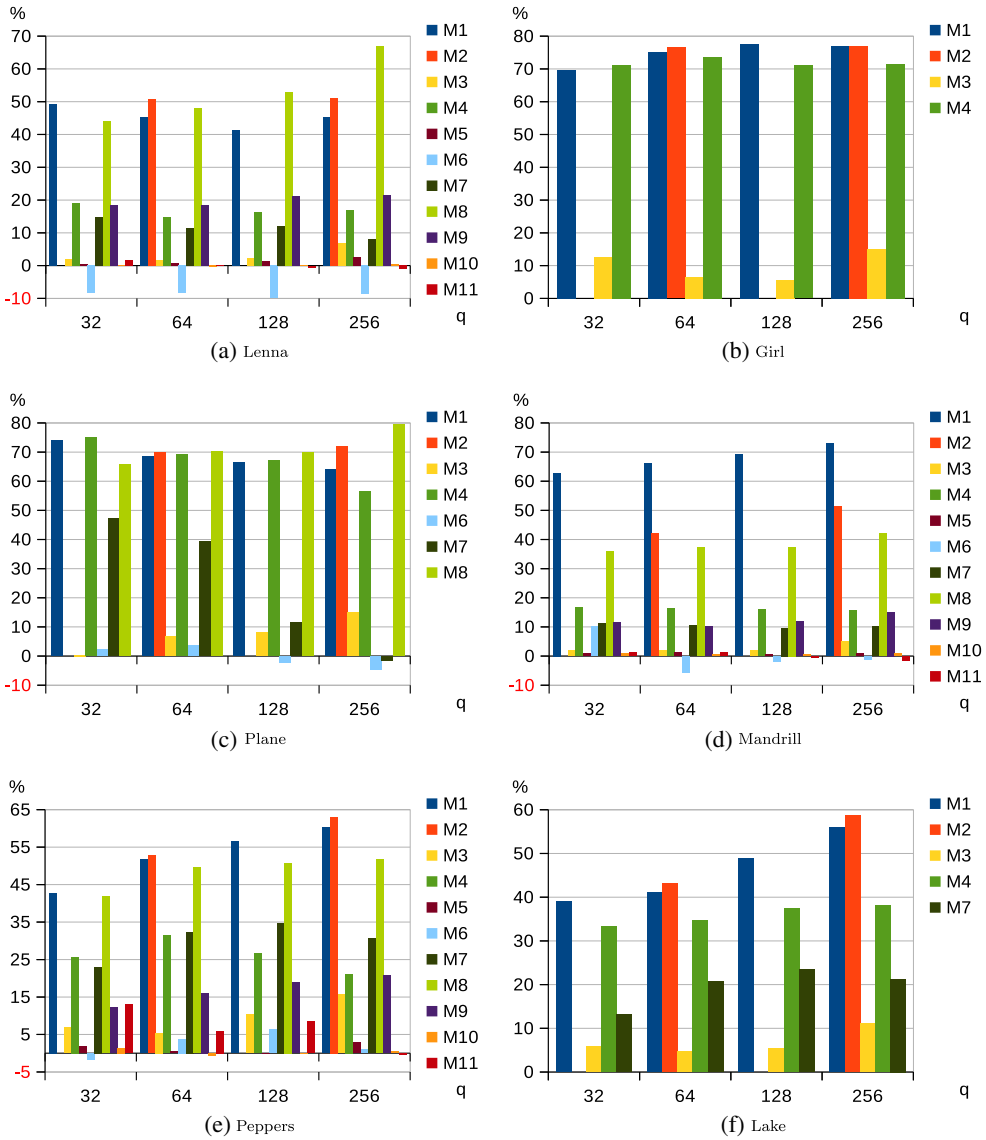


Fig. 11 Percentage of error reduction obtained by ABC + ATCQ (iteration 20) with respect to the methods described in Table 9 (q: colors of the quantized palette)

7 Conclusions

A color quantization method that combines artificial bees and artificial ants has been proposed. This method is based on the algorithm described by Ozturk et al. for solving the same problem by combining artificial bees and the K-means algorithm. Thus, the K-means algorithm is replaced by the ATCQ algorithm, as the K-means algorithm is very time-consuming.

Computational results show that the new method can generate images with a similar quality to that of the method proposed by Ozturk et al., but requires considerably less amount of time to reach the final solution. In addition, this method generates better images than other well-known color quantization methods such as Wu's method, Neuquant, K-means, Octree or the Variance-based method. Although PSO can generate better images than ABC + ATCQ in some cases, is should be noted the excessive time required by PSO to obtain such results.

The ABC algorithm has been combined with the general ATCQ algorithm, which builds a multilevel tree, and with the variant of the algorithm that generates a 3-level tree. The analysis of the results shows that the second case is faster and generates images very similar to the first case.

Acknowledgements This work was supported by the Samuel Solórzano Barruso Memorial Foundation of the University of Salamanca.

Funding Information This study was funded by the Samuel Solórzano Barruso Memorial Foundation of the University of Salamanca (grant number FS/102015).

Compliance with Ethical Standards

Conflict of interests The author declares that she has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. An NY, Pun CM (2014) Color image segmentation using adaptive color quantization and multiresolution texture characterization. SIViP 18(5):1–12. https://doi.org/10.1007/s11760-012-0340-2
2. Araújo AR, Costa DC (2009) Local adaptive receptive field self-organizing map for image color segmentation. Image Vis Comput 27(9):1229–1239. https://doi.org/10.1016/j.imavis.2008.11.014
3. Atsalakis A, Papamarkos N (2006) Color reduction and estimation of the number of dominant colors by using a self-growing and self-organized neural gas. Eng Appl Artif Intell 19(7):769–786. https://doi.org/10.1016/j.engappai.2006.05.004
4. Azzag H, Venturini G, Oliver A, Guinot C (2007) A hierarchical ant based clustering algorithm and its use in three real-world applications. Eur J Oper Res 179(3):906–922. https://doi.org/10.1016/j.ejor.2005.03.062
5. Balasubramani K, Marcus K (2013) A comprehensive review of artificial bee colony algorithm. Int J Comput Technol 5(1):15–28
6. Barbalho JM, Duarte A, Neto D, Costa JA, Netto ML (2001) Hierarchical SOM applied to image compression. In: Proceedings of the international joint conference on neural networks, IJCNN'01, vol 1. IEEE, pp 442–447
7. Brun L, Trémeau A (2003) Color quantization. Digital color imaging handbook, pp 589–638
8. Celebi ME (2009) An effective color quantization method based on the competitive learning paradigm. In: Proceedings of the international conference on image processing, computer vision and pattern recognition, pp 876–880
9. Celebi ME (2011) Improving the performance of k-means for color quantization. Image Vis Comput 29(4):260–271. https://doi.org/10.1016/j.imavis.2010.10.002
10. Celebi ME, Wen Q, Hwang S (2015) An effective real-time color quantization method based on divisive hierarchical clustering. J Real-Time Image Proc 10(2):329–344. https://doi.org/10.1007/s11554-012-0291-4
11. Chang CH, Xu P, Xiao R, Srikanthan T (2005) New adaptive color quantization method based on self-organizing maps. IEEE Trans Neural Netw 16(1):237–249. https://doi.org/10.1109/TNN.2004.836543
12. Chen X, Kwong S, Feng JF (2002) A new compression scheme for color-quantized images. IEEE Trans Circuits Syst Video Techn 12(10):904–908. https://doi.org/10.1109/TCSVT.2002.804896
13. Cheng G, Yang J, Wang K, Wang X (2006) Image color reduction based on self-organizing maps and growing self-organizing neural networks. In: Proceedings of the sixth international conference on hybrid intelligent systems, HIS'06. IEEE, p 24, https://doi.org/10.1109/HIS.2006.264907
14. Cheng SC, Yang CK (2001) A fast and novel technique for color quantization using reduction of color space dimensionality. Pattern Recogn Lett 22(8):845–856. https://doi.org/10.1016/S0167-8655(01)00025-3
15. Corder GW, Foreman DI (2009) Comparing two related samples: the Wilcoxon signed ranks test. Wiley Online Library, pp 38–56
16. Dekker AH (1994) Kohonen neural networks for optimal colour quantization. Netw Comput Neural Syst 5(3):351–367. https://doi.org/10.1088/0954-898X/5/3/003
17. Deng Y, Manjunath B (2001) Unsupervised segmentation of color-texture regions in images and video. IEEE Trans Pattern Anal Mach Intell 23(8):800–810. https://doi.org/10.1109/34.946985
18. Fritzke B (1995) A growing neural gas network learns topologies. In: Proceedings of the 1995 conference on advances in neural information processing systems, pp 625–632
19. Garey M, Johnson D, Witsenhausen H (1982) The complexity of the generalized Lloyd-max problem (corresp.) IEEE Trans Inf Theory 28(2):255–256. https://doi.org/10.1109/TIT.1982.1056488
20. Gervautz M, Purgathofer W (1990) A simple method for color quantization: Octree quantization. In: Glassner AS (ed) Graphics gems. USA, San Diego, pp 287–293. https://doi.org/10.1007/978-3-642-83492-9_20
21. Ghanbarian AT, Kabir E, Charkari NM (2007) Color reduction based on ant colony. Pattern Recogn Lett 28(12):1383–1390. https://doi.org/http://doi.org/10.1016/j.patrec.2007.01.019
22. Heckbert P (1982) Color image quantization for frame buffer display. In: Proceedings of the 9th annual conference on computer graphics and interactive techniques, SIGGRAPH '82. ACM, New York, pp 297–307. https://doi.org/10.1145/800064.801294
23. Hu YC, Su BH (2008) Accelerated k-means clustering algorithm for colour image quantization. The Imaging Sci J 56(1):29–40. https://doi.org/10.1179/174313107X176298
24. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recogn Lett 31(8):651–666. https://doi.org/10.1007/978-3-540-87479-9_3
25. Joy G, Xiang Z (1993) Center-cut for color-image quantization. Visual Comput 10(1):62–66. https://doi.org/10.1007/BF01905532
26. Kanjanawanishkul K, Uyyanonvara B (2005) Novel fast color reduction algorithm for time-constrained applications. J Vis Commun Image Represent 16(3):311–332. https://doi.org/10.1016/j.jvcir.2004.07.002
27. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report TR-06. Tech. rep. Erciyes University, Engineering Faculty, Computer Engineering Department
28. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132. https://doi.org/http://doi.org/10.1016/j.amc.2009.03.090
29. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471. https://doi.org/http://doi.org/10.1007/s10898-007-9149-x
30. Karaboga D, Ozturk C (2011) A novel clustering approach: Artificial bee colony (ABC) algorithm. Appl Soft Comput 11(1):652–657. https://doi.org/http://doi.org/10.1016/j.asoc.2009.12.025
31. Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: International conference on modeling decisions for artificial intelligence. Springer, pp 318–329. https://doi.org/http://doi.org/10.1007/978-3-540-73729-2_30
32. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42(1):21–57. https://doi.org/10.1007/s10462-012-9328-0
33. Kasuga H, Yamamoto H, Okamoto M (2000) Color quantization using the fast k-means algorithm. Syst Comput Japan 31(8):33–40. https://doi.org/10.1002/1520-684X(200007)31:8<33::AID-SCJ4>3.0.CO;2-C
34. Kohonen T (1998) The self-organizing map. Neurocomput 21(1-3):1–6. https://doi.org/10.1016/S0925-2312(98)00030-7
35. Linde Y, Buzo A, Gray R (1980) An algorithm for vector quantizer design. IEEE Trans Commun 28(1):84–95. https://doi.org/10.1109/TCOM.1980.1094577

36. Liu GH, Yang JY (2013) Content-based image retrieval using color difference histogram. Pattern Recogn 46(1):188–198. https://doi.org/10.1016/j.patcog.2012.06.001
37. Omran M, Engelbrecht AP, Salman A (2005a) A color image quantization algorithm based on particle swarm optimization. Inform (Slovenia) 29(3):261–270
38. Omran M, Engelbrecht AP, Salman A (2005b) Particle swarm optimization method for image clustering. Int J Pattern Recognit Artif Intell 19(03):297–321. https://doi.org/10.1142/S0218001405004083
39. Orchard MT, Bouman CA (1991) Color quantization of images. IEEE Trans Signal Proc 39(12):2677–2690. https://doi.org/10.1109/78.107417
40. Özdemir D, Akarun L (2002) A fuzzy algorithm for color quantization of images. Pattern Recogn 35(8):1785–1791. https://doi.org/10.1016/S0031-3203(01)00170-4
41. Ozturk C, Hancer E, Karaboga D (2014) Color image quantization: a short review and an application with artificial bee colony algorithm. Inform 25(3):485–503. https://doi.org/10.15388/Informatica.2014.25
42. Palomo EJ, Domínguez E (2014) Hierarchical color quantization based on self-organization. J Math Imaging Vis 49(1):1–19. https://doi.org/10.1007/s10851-013-0433-8
43. Papamarkos N, Atsalakis AE, Strouthopoulos CP (2002) Adaptive color reduction. IEEE Trans Syst Man Cybern B Cybern 32(1):44–56. https://doi.org/10.1109/3477.979959
44. Park HJ, Cha EY, Kim KB (2015) An effective color quantization method using color importance-based self-organizing maps. Neural Netw World 25(2):121–137. https://doi.org/10.14311/NNW.2015.25.006
45. Pérez-Delgado ML (2015) Colour quantization with ant-Tree. Appl Soft Comput 36:656–669. https://doi.org/10.1016/j.asoc.2015.07.048
46. Pérez-Delgado ML (2018a) Campus Viriato images. http://audax.zam.usal.es/web/mlperez/fotos_campus.html. Accessed 19.10.18
47. Pérez-Delgado ML (2018b) Source code for colour quantization. http://audax.zam.usal.es/web/mlperez/cq.html. Accessed 20.07.18
48. Phung SL, Bouzerdoum A, Chai D (2005) Skin segmentation using color pixel classification: analysis and comparison. IEEE Trans Pattern Anal Mach Intell 27(1):148–154. https://doi.org/10.1109/TPAMI.2005.17
49. Ponti M, Nazaré TS, Thumé GS (2016) Image quantization as a dimensionality reduction procedure in color and texture feature extraction. Neurocomputing 173:385–396. https://doi.org/10.1016/j.neucom.2015.04.114
50. Şaykol E, Güdükbay U, Ulusoy Ö (2005) A histogram-based approach for object-based query-by-shape-and-color in image and video databases. Image Vis Comput 23(13):1170–1180. https://doi.org/10.1016/j.imavis.2005.07.015
51. Schaefer G, Zhou H (2009) Fuzzy clustering for colour reduction in images. Telecommun Syst 40(1):17–25. https://doi.org/10.1007/s11235-008-9143-8
52. Scheunders P (1997) A genetic c-means clustering algorithm applied to color image quantization. Pattern Recogn 30(6):859–866. https://doi.org/10.1016/S0031-3203(96)00131-8
53. Shukran MAM, Chung YY, Yeh WC, Wahid N, Zaidi AMA (2011) Artificial bee colony based data mining algorithms for classification tasks. Mod Appl Sci 5(4):217–231. https://doi.org/10.5539/mas.v5n4p217
54. Sirisathitkul Y, Auwatanamongkol S, Uyyanonvara B (2004) Color image quantization using distances between adjacent colors along the color axis with highest color variance. Pattern Recogn Lett 25(9):1025–1043. https://doi.org/10.1016/j.patrec.2004.02.012
55. Tsai CF, Jhuang CA, Liu CW (2008) Gray image compression using new hierarchical self-organizing map technique. In: ICICIC'08; 3rd international conference on innovative computing information and control. IEEE, pp 544–544. https://doi.org/10.1109/ICICIC.2008.298
56. Uchiyama T, Arbib MA (1994) An algorithm for competitive learning in clustering problems. Pattern Recogn 27(10):1415–1421. https://doi.org/10.1016/0031-3203(94)90074-4
57. Verevka O, Buchanan JW (1995) The local k-means algorithm for colour image quantization. In: Proceedings of graphics interface '95, Québec. Canadian Information Processing Society, Canada, pp 128-128, https://doi.org/10.20380/GI1995.15
58. Wan S, Prusinkiewicz P, Wong S (1990) Variance-based color image quantization for frame buffer display. Color Res Appl 15(1):52–58. https://doi.org/10.1002/col.5080150109
59. Wang CH, Lee CN, Hsieh CH (2007) Sample-size adaptive self-organization map for color images quantization. Pattern Recogn Lett 28(13):1616–1629. https://doi.org/10.1016/j.patrec.2007.04.005
60. Wang J, Yang WJ, Acharya R (1997) Color clustering techniques for color-content-based image retrieval from image databases. In: Proceedings of the IEEE international conference on multimedia computing and systems' 97. IEEE, pp 442–449, https://doi.org/http://doi.org/10.1109/MMCS.1997.609755
61. Wang XY, Yu YJ, Yang HY (2011) An effective image retrieval scheme using color, texture and shape features. Comput Stand Interfaces 33(1):59–68. https://doi.org/10.1016/j.csi.2010.03.004
62. Weber A (2018) USC-SIPI image database. http://sipi.usc.edu/database/database.php/. Accessed 19.10.18
63. Wen Q, Celebi ME (2011) Hard versus fuzzy c-means clustering for color quantization. EURASIP J Adv Signal Proc 2011(1):1–12. https://doi.org/10.1186/1687-6180-2011-118
64. Wu X (1991) Efficient statistical computations for optimal color quantization. In: Graphics gems, vol II. Academic Press, pp 126–133, https://doi.org/10.1016/B978-0-08-050754-5.50035-9
65. Wu X (1992) Color quantization by dynamic programming and principal analysis. ACM Trans Graph 11(4):348–372. https://doi.org/10.1145/146443.146475
66. Yang CK, Tsai WH (1998) Color image compression using quantization, thresholding, and edge detection techniques all based on the moment-preserving principle. Pattern Recogn Lett 19(2):205–215. https://doi.org/10.1016/S0167-8655(97)00166-9
67. Yang CY, Lin JC et al (1996) RWM-Cut for color image quantization. Comput Graph 20(4):577–588. https://doi.org/10.1016/0097-8493(96)00028-3
68. Zhang C, Ouyang D, Ning J (2010) An artificial bee colony approach for clustering. Expert Syst Appl 37(7):4761–4767. https://doi.org/10.1016/j.eswa.2009.11.003