Image Segmentation with Metaheuristic algorithms

Jhair Gallardo
Rochester Institute of Technology
gg4099@rit.edu

Nick Gardner
Rochester Institute of Technology
nag6650@rit.edu

Abstract

Partitioning an image into regions based on a defined criteria is the task known as image segmentation. This can be done by clustering similar pixels into groups. A strong clustering-based method to segment images is k-means, which iteratively updates the centroids for each cluster starting from a random initialization. However, k-means final centroids highly depend on this random initialization and can easily fall into local optima solutions. On the other hand, metaheuristic methods are known for their robustness against random initialization when finding the global optima (or close to it) of a cost function, overcoming the k-means issue. While there are works on image segmentation using metaheuristic algorithms, a review of these algorithms in the same conditions and same datasets is missing. Here, we test several metaheuristic algorithms on the image segmentation task and compare them against k-means. We measure their performance using PSNR, inter-cluster, and intra-cluster distance on classic images used by the image processing community. We found that metaheuristic algorithms are able to perform image segmentation and get comparable results to k-means.

1 Introduction

Image segmentation is a well-known task in image processing and computer vision, which consists of partitioning an image into regions based on a defined criteria. Segmenting an image is an important pre-processing phase of many image-based applications such as content-based image retrieval, machine vision, medical imaging, object recognition, and video surveillance [14]. Several techniques have been developed to solve image segmentation [14, 9], where the majority of them can be categorized in 2 groups: region-based, and edge- or boundary-based methods. This work focuses on the region-based methods which divide the image into sub-regions or clusters. This type of segmentation can be done using clustering methods and treating each pixel as a datapoint. [4].

One popular clustering method is k-means, which finds the centroids of clusters after an iterative process. It starts with randomly initialized centroids whose positions are then updated based on the mean of the datapoints that are closer to them. While k-means is a strong baseline on image segmentation, it is highly dependent on the location of the randomly initialized centroids, needing extra steps to overcome this issue [3]. On the other hand, there are works that perform clustering using metaheuristic algorithms [10, 7, 1, 16, 12]. Metaheuristic methods rely on the exploration-exploitation trade off, where some components of the algorithm explore the solution space based on randomness, and other components exploit well-performing solutions by searching for other solutions nearby. The exploration-exploitation nature of metaheuristic algorithms makes them robust against random initialization, with the ability to find good cluster solutions and not get stuck in local optima.

More related to our paper, the work in [13] applies Particle Swarm Optimization for image segmentation, showing that metaheuristic algorithms can be easily adapted to solve the image segmentation task. Other works [11, 2, 15, 6] combine k-means clustering with metaheuristic methods, reducing the dependence on random initialization for k-means. However, these methods lead to higher running times since they need to run k-means and a metaheuristic algorithm. In this work, similar to [13], we

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

focus on using metaheuristic algorithms directly on the image segmentation task. We found that a fair comparison between different metaheuristic algorithms applied on the image segmentation task is missing, so we do that here.

This paper shows a comparison between metaheuristic methods on the image segmentation task. We also compare these methods against the popular clustering method k-means. The contributions of this work are:

- We compare the performance of different metaheuristic algorithms (Simulated Annealing, Differential Evolution, Particle Swarm Optimization, Firefly Algorithm, and Bat Algorithm) on image segmentation against k-means.
- We show an easy scheme to transform any metaheuristic algorithm into solving image segmentation by doing clustering.

2 Related Work

The use of metaheuristic algorithms for clustering was first introduced by [10], where simulated annealing was used. Since then, there have been two broad classes of metaheuristic development: evolutionary-based (genetic) and swarm. The first genetic clustering algorithm was published in [7]. Subsequent iterations and improvements in this area include the use of better representations for clustering, such as the grouping genetic algorithm used in [1], and improvements on the genetic algorithm, such as the memetic clustering used in [5]. One of the early forerunners for swarm-based methods was [8], which used k-means to optimize the initial population and PSO to do final optimization. Other publications in this area include the use of varied swarm optimization algorithms, such as [16] - which compares the Cooperative Bee Algorithm to PSO and ABC for clustering. There have even been some hybrid attempts, like [12], that combined PSO and a genetic algorithm to improve accuracy and speed of power line image segmentation.

Specifically on image segmentation, a set of works [11, 2, 15, 6] have tried combining metaheuristic algorithms with k-means, overcoming the caveats of k-means. While these algorithms show good performance, their running times are longer than just running either k-means or metaheuristics. On this work we focus on using only metaheuristic algorithms to solve the task, and compare their performance against popular methods like k-means.

Most related to this paper, there are some publications that apply metaheuristics directly for image segmentation. In [13], a similar exploration was performed utilizing only particle swarm optimization. The conclusion of [13] was that a metaheuristic-based image segmentation (utilizing PSO), can outperform k-means image segmentation. However, its test metrics - quantization error, intra-cluster distance, and inter-cluster distance - were all included in the fitness function of its PSO implementation. While the implementation outperformed k-means on those metrics, there is no information as to the performance with respect to test metrics not included in the fitness function. In this paper, more metaheuristic algorithms are tested, and a more fair comparison baseline is implemented. Additionally, optimization algorithms in this paper only utilize quantization error in their fitness functions, leaving other metrics such as PSNR, intra-cluster, and inter-cluster, to assess the quality of image segmentation as compared to other algorithms as well as k-means.

3 Problem Description

The task in hand is image segmentation, where an image is partitioned into regions based on a defined criteria. This can be done by clustering similar pixels into the same group. More specifically, from an Image I containing N pixels z_p, we want to find a set of clusters $C = \{C_1, C_2, ..., C_k\}$, where k is the total number of clusters in the set. Each cluster C_j is represented with a centroid m_j. C_j would contain pixels that are similar to its corresponding centroid, meaning a pixel z_p will belong to cluster C_j if centroid m_j is the closest to the pixel in terms of Euclidean distance. All clusters must be mutually exclusive.

The k-means algorithm can solve the presented task by randomly initializing centroids and then updating their positions based on the mean of the datapoints that are closer to them. In this case, each datapoint would be a pixel from the image. The standard k-means algorithm for image segmentation is summarized below:

1. Randomly initialize k centroids $M = \{m_1, m_2, ..., m_k\}$.
2. For each pixel z_p in image I, assign it to cluster C_j if the centroid m_j is the closest one to the pixel.
3. Replace each centroid m_j with the mean value of the pixels in its cluster C_j.
4. Repeat 2 and 3 until the centroids no longer change.

In this work we explore an alternative to k-means by solving the image segmentation task using metaheuristic algorithms. These types of algorithms evaluate evaluate proposed solutions using a fitness function. One main problem to tackle while doing image segmentation with metaheuristic algorithms is the selection of a good fitness function.

4 Proposed Solution

Image segmentation can be optimized using metaheuristic algorithms directly, with a set of solution vectors representing cluster centroids. When the solution vectors are represented in this way, the optimization problem is no different from any other optimization function that these algorithms are designed to solve. The challenging aspect of the design phase is then the selection of the fitness function used by the algorithms to optimize the solution. After reviewing related work, specifically [13], it is the intention of this paper to present a more fair and holistic view of the promise of metaheuristic algorithms for this purpose. With this in mind, only one metric, quantization error, is used as the fitness function for each optimization algorithm. This metric is selected because it is quite broad - it is the average distance from each pixel to its corresponding cluster centroid. Other metrics are held in reserve, to use solely for testing, to ensure a fair comparison between the algorithms and specifically against the common standard, k-means. Finally, a varied selection of metaheuristic algorithms are chosen for testing, with the intention of exploring the dynamics of their different styles.

4.1 Fitness Function

The beauty of the metaheuristic algorithms is that they are polymorphic. Each of the algorithms used in this paper are as they are specified in literature - no modification was necessary. A single fitness function was implemented and utilized by all algorithms. The quantization error is described as follows:

$$Q_e = \frac{\sum_{j=1}^{N_c}[\sum_{\forall z_p \in C_j} d(z_p, m_j)]/|C_j|}{N_c} \tag{1}$$

where N_c is the number of clusters, z_p is the p^{th} pixel, C_j is the set of pixels that belong to cluster j, m_j is the centroid of cluster j, $|C_j|$ is the number of pixels in cluster j, and $d(z_p, m_j)$ is the Euclidean distance between pixel z_p and centroid m_j. Following the distance-based cluster assignment from k-means, each pixel z_p is assigned to the cluster with the closest centroid m_j.

4.2 Population

Solution vectors were generated as populations of the same size and shape (except for SA which is a single-agent algorithm):

$$X = [x_1, x_2, ..., x_k] \text{ where } x_i = [x_{i1}, x_{i2}, ..., x_{id}] \tag{2}$$

where k is the size of the population and d is [number of clusters $\times$ the number of channels]. Each solution vector, x_i, is constructed as a flat array to make mathematical operations easier to carry out. For an RGB image with five clusters, and a population size of 20, the population would consist of 20 solution row vectors, each of size 15 ([five clusters $\times$ three channels]).

4.3 Metaheuristic Algorithms

We briefly describe each selected metaheuristic algorithm in this section. With the objective to test different kinds of metaheuristic algorithms, we picked single-agent-based, evolution-based, and swarm-based algorithms.

4.3.1 Simulated Annealing (SA)

Simulated Annealing is a trajectory-based (single-agent) optimization algorithm that was inspired by the process of cooling metals. At its essence, SA utilizes a cooling a schedule to allow its agent to make sub-optimal decisions in early iterations, allowing for global exploration, but to choose strictly better neighbors in later iterations, to ensure exploitation.

1. Initialize an initial and final temperature. Establish a cooling schedule:

$$T(t) = T_0 \beta^t \tag{3}$$

2. Perturb agent location using a noise vector drawn from a Gaussian distribution:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \varepsilon \tag{4}$$

3. Determine change in fitness from current solution to new potential solution

$$f_\Delta = f_{t+1}(\mathbf{x}_{t+1}) - f_t(\mathbf{x}_t) \tag{5}$$

4. If the solution is better, accept it. Otherwise, draw a random number from a uniform distribution between 0 and 1. If the transition probability, p, is greater than this r, accept the new solution.

$$p = e^{\frac{f_\Delta}{T}} \tag{6}$$

5. Keep track of the best solution and fitness
6. Iterate temperature index (and hence current temperature)
7. Repeat steps 2-6 until final temperature is reached.

4.3.2 Differential Evolution (DE)

Differential evolution is a population-based, derivative-free metaheuristic algorithm that directly operates on its solution vectors (unlike Genetic Algorithms).

1. An initial population is formed
2. For each vector $\mathbf{x}_i^t$ in the population, a donor vector is constructed as a combination of three distinct vectors:

$$\mathbf{v}_i^{t+1} = \mathbf{x}_p^t + F(\mathbf{x}_q^t - \mathbf{x}_r^t) \tag{7}$$

   where F is a differential weight parameter. This step implements the mutation component of an evolutionary algorithm, introducing an element of randomization that allows the algorithm to escape local optima

3. For each vector $\mathbf{x}_i^t$ in the population, create a new vector $\mathbf{u}_i^{t+1}$ using crossover with the donor vector constructed in step 2. For example, with binomial crossover:

$$\mathbf{u}_{j,i}^{t+1} = \begin{cases} \mathbf{v}_{j,i} & \text{if } r_i \leq C_r \\ \mathbf{x}_{j,i}^t & \text{otherwise} \end{cases} \tag{8}$$

   where C_r is a crossover threshold parameter and r_i is a random number generated from a uniform distribution. Essentially this equation states that for each component of the original vector, the respective component of the new vector will be the component from the donor vector if the randomly generated number is less than or equal to the parameter, otherwise it will be the component from the original vector.

4. Finally, the fitness of the new solution vectors are calculated and compared to the fitness of the previous generation of vectors. If a new solution vector has greater fitness than its counterpart in the previous generation, it replaces that vector. Otherwise, the previous solution vector continues through.

4.3.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization is one of the more basic varieties of swarm intelligence algorithms, and is inspired by the flock/swarm behavior of birds. Essentially, each agent is aware of a current best global solution, $g*$, and a current best local solution, x_i^{*t}. The agent tends to move randomly but is influenced by current best solutions. The idea here is that having a global best solution helps the algorithm to converge and local best solutions help the algorithm to explore.

1. Initialize locations of particles and velocity of particles
2. Calculate current global best and local best (their current location) for the population
3. For all particles:
   (a) Determine new particle velocity

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha\varepsilon_1 \odot (g^* - \mathbf{x}_i^t) + \beta\varepsilon_2 \odot (\mathbf{x}_i^{*t} - \mathbf{x}_i^t) \tag{9}$$

   where ε_1. and ε_2 are noise vectors randomly distributed according to a uniform distribution between 0 and 1, and α and β are scaling factors. The first term (led by α) is the social intelligence component, where the global best solution is utilized to guide the path of this agent. The next term (led by β) is the cognitive intelligence component that utilizes the local best solution to encourage exploration.

   (b) Use particle velocity to update particle position

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{10}$$

   (c) Update local best based on fitness of new solution
4. Update global best based on local fitness bests
5. Increase iteration counter
6. Repeat steps 3-5 until maximum number of iterations is reached

4.3.4 Firefly Algorithm (FA)

The Firefly Algorithm is inspired by the signalling displayed by fireflies that wish to mate. In nature, the rhythm, rate, and delay between flashes is used to communicate between fireflies. Additionally, the natural properties of air introduce a decay mechanic to the attractiveness that follows the inverse square law. For the purposes of the algorithm, some simplifications are made. All agents are unisex (they want to mate with all other attractive solutions), their attractiveness is proportional to their brightness, and brightness is the value of the objective function for an agent's solution vector.

1. Generate a population of fireflies and calculate their brightness (fitness)
2. Compare each firefly to each other firefly
   (a) If the other firefly's brightness is higher (more fit solution), move towards it

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + (\frac{\beta_0}{1 + \gamma r_{i,j}^2})(\mathbf{x}_j^t - \mathbf{x}_i^t) + \alpha\varepsilon_i^t \tag{11}$$

   where ε_i^t follows a normal distribution with $\mu = 0$ and $\sigma = 1$, $\beta_0 = 1$ (maximum intensity of attraction is at the source), and $\alpha\epsilon[0, 1]$. γ is a light absorption coefficient that affects the distance of attraction. As γ approaches 0, attraction becomes constant. As γ approaches ∞, attraction becomes 0 and the fireflies roam randomly.
3. Determine fitness of fireflies and rank global best
4. Repeat steps 2 and 3 until maximum number of generation / iterations is reached.

4.3.5 Bat Algorithm (BA)

The motivation behind the Bat Algorithm lies in the hunting behavior of bats. Bats use echolocation to sense distance and can differentiate between food/prey and other bats. As they hunt, they move randomly and can change the frequency/wavelength of their echolocation pulses. The loudness of the emitted pulses also can vary, and this is where the true power of this algorithm comes into play. BA is highly effective at quickly converging.

1. Initialize population of bats with locations and velocities
2. For each agent in the population:
   (a) Update frequency
   $$\mathbf{f}_i = f_{min} + (f_{max} - f_{min})\beta_i \tag{12}$$
   (b) Update velocity
   $$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + (\mathbf{x}_i^t - g*) \odot \mathbf{f}_i \tag{13}$$
   (c) Update position
   $$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{14}$$
   (d) Generate a random number from a uniform distribution. If this number is larger than the individual pulse rate, replace the current solution with a solution near the global best.
   $$\mathbf{x}_i^{t+1} = \begin{cases} \mathbf{g}^* + 0.01 * \varepsilon & \text{if rand} \geq r_i \\ \mathbf{x}_i^{t+1} & \text{otherwise} \end{cases} \tag{15}$$
   where ε is drawn from a normal distribution with unit variance.
   (e) Calculate solution fitness
   (f) Generate another random number and decide whether to replace solution in population
   $$\mathbf{x}_i = \begin{cases} \mathbf{x}_i^{t+1} & \text{if } f(\mathbf{x}_i^{t+1}) > f(\mathbf{x}_i) \text{ and if rand} < \text{A} \\ \mathbf{x}_i & \text{otherwise} \end{cases} \tag{16}$$
   (g) Update loudness and pulse rate based on decided schedule. For example:
   $$\begin{aligned} A_i^{t+1} &= \alpha A_i^t && \text{update loudness} \\ r_i^{t+1} &= r_i^f[1 - e^{-\gamma t}] && \text{update pulse rate} \end{aligned} \tag{17}$$
   where α and γ are scale factors between 0 and 1. A good starting location for these are $\alpha = \gamma = 0.9$.

5 Experimental Setup

We perform image segmentation on 4 classic images used by the image processing community: "lena", "tiger", "cameraman", and "coins". 1 shows the original images. Two images are in color while the other two are in gray scale. We set the number of cluster fixed for each image, having 6 clusters for lena, 8 clusters for tiger, 4 clusters for cameraman, and 2 clusters for coins.

We performed a fine-tuning stage for each algorithm before getting the final results. We describe the hyperparameters for each metaheuristic algorithm after fine-tuning below:

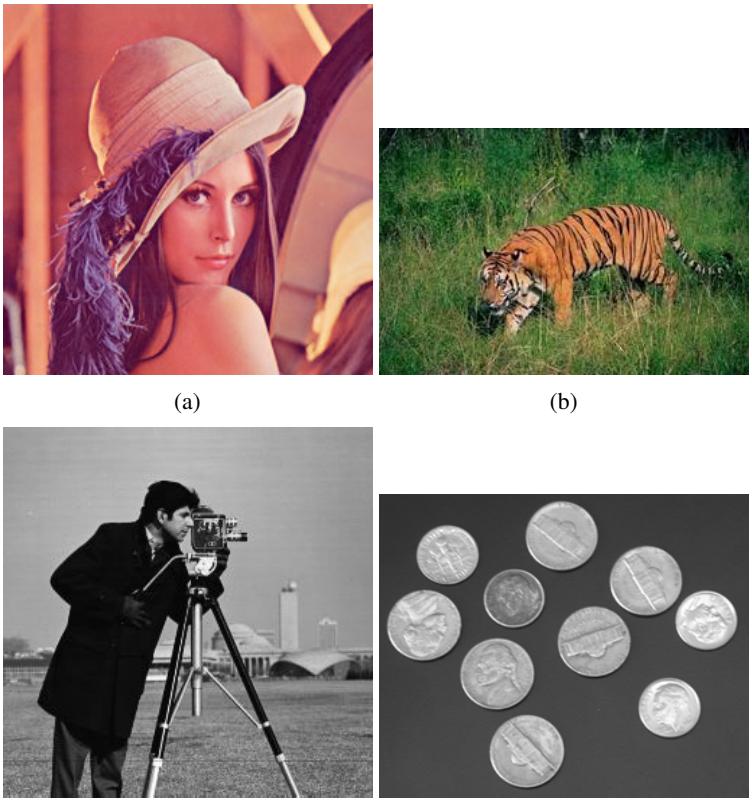
- For SA we use $T_0 = 10$, $\beta = 0.8$, and 1000 iterations.
- For DE we use $F = 1$, $C_r = 0.5$, population size of 20, and 50 iterations.
- For PSO we use $\alpha = 1.5$, $\beta = 0.5$, population size of 20, and 50 iterations.
- For FA we use $\gamma = 0.0001$, $\alpha = 0.1$, $\beta = 1$, population size of 20, and 50 iterations.
- For BA we use $\alpha = 0.8$, $\gamma = 0.9$, population size of 20, and 50 iterations.

It is worth to mention that we used 1000 iterations on DE to mimic the effective number of iterations of the population based algorithms which is [population size $times$ iterations]. For k-means, we run the algorithm until the centroids have zero movement.

To measure the quality of segmentation of each algorithm, we use the following metrics:

Peak Signal-to-Noise Ratio (PSNR): It measures the ratio between the maximum possible power of a signal and the power of corrupting noise. It is a popular metric used for image reconstruction quality on image compression. On image segmentation, a higher PSNR means that the segmentation has a good quality and the original image can be detected. PSNR is calculated as follows (assuming we normalize images to have pixels with values from 0 to 1):

$$PSNR = 10 \log_{10} \frac{1}{\sqrt{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I_{i,j} - S_{i,j})^2}} \tag{18}$$



(a) (b)

(c) (d)

Figure 1: Original images to be segmented. (a) lena. (b) tiger. (c) cameraman. (d) coins.

where I is the original image, S is the segmented image, m is the total number of rows, and n is the total number of columns.

Inter-cluster distance: It measures the minimum distance between 2 cluster centroids. Having a high inter-cluster distance means that each cluster is well separated from another. It is calculated as follows:

$$d_{min} = min_{\forall i,j,i \neq j}\{d(m_i, m_j)\} \tag{19}$$

Intra-cluster distance: It measures the maximum average distance between a cluster centroid and its pixels. Having a lower intra-cluster distance means that each cluster is compact. It is calculated as follows:

$$\bar{d}_{max} = max_{j=1,\ldots,N_c}\left\{\frac{\sum_{\forall z_p \in C_j} d(z_p, m_j)}{|C_j|}\right\} \tag{20}$$

After performing clustering with the algorithms and getting the corresponding centroids, we assign each pixel to a cluster by finding the closest cluster centroid, resulting in a mask that specifies the cluster index for each pixel. To transform this mask into the pixel space of values, we replace the index numbers with the mean pixel value of each cluster (which is not necessarily the centroid value for the metaheuristic algorithms). This image in the pixel color space is what we use in the PSNR metric as S.

6 Results

We compare different metaheuristic algorithms against k-means on image segmentation by doing pixel clustering. We measure the performance of each algorithm on each image using $PSNR$, d_{min}, and $\bar{d}_{max}$. We average results over 5 trials for all the algorithms.

Table 1: PSNR results for different clustering algorithms and different images. We show the mean (± standard deviation) over 5 trials. A higher PSNR corresponds to better results. Bold numbers show best values, and underlined numbers show second best values.

METHODS	Lena	Tiger	Cameraman	Coins
K-means	**25.5910** (± 0.0006)	**24.9751** (± 0.0698)	**25.9339** (± 0.0468)	**23.9844** (± 0.0000)
SA	22.5184 (± 2.5776)	<u>22.8917</u> (± 1.5042)	23.9775 (± 3.4892)	<u>23.9839</u> (± 0.0009)
DE	20.9154 (± 0.5908)	20.5944 (± 1.0359)	22.5594 (± 1.7642)	23.9590 (± 0.0226)
PSO	23.0332 (± 1.2088)	22.2437 (± 1.0630)	21.8333 (± 1.3805)	23.9676 (± 0.0272)
FA	<u>24.0371</u> (± 1.1009)	21.4019 (± 1.2603)	<u>25.1963</u> (± 0.9350)	23.9834 (± 0.0011)
BA	22.0791 (± 1.1043)	20.5917 (± 0.7558)	22.8699 (± 3.2174)	23.9561 (± 0.0092)

Table 2: Inter-cluster distance (d_{min}) results for different clustering algorithms and different images. We show the mean (± standard deviation) over 5 trials. A higher Inter-cluster distance corresponds to better results. Bold numbers show best values, and underlined numbers show second best values.

METHODS	Lena	Tiger	Cameraman	Coins
K-means	**0.1761** (± 0.0000)	0.1212 (± 0.0026)	**0.2236** (± 0.0396)	0.7913 (± 0.0002)
SA	0.1209 (± 0.0191)	<u>0.1327</u> (± 0.0209)	0.1653 (± 0.0091)	**0.7917** (± 0.0003)
DE	0.1115 (± 0.0265)	0.0971 (± 0.0270)	<u>0.2134</u> (± 0.0721)	0.7906 (± 0.0022)
PSO	0.1158 (± 0.0556)	**0.1450** (± 0.0271)	0.1530 (± 0.0413)	0.7903 (± 0.0017)
FA	0.1451 (± 0.0214)	0.1221 (± 0.0230)	0.2089 (± 0.0544)	<u>0.7914</u> (± 0.0005)
BA	<u>0.1514</u> (± 0.0389)	0.1281 (± 0.0404)	0.1938 (± 0.0556)	0.7910 (± 0.0024)

Table 1 summarizes the results for the $PSNR$ metric. We can see that k-means is the best method across all the four images for this metric. We notice that, for coins image, the performance of all the algorithms are close to each other, with a maximum difference of 0.0254. This means that all the algorithms are able to find a segmentation with similar quality for this specific coins image. We believe this happens because we segment the coins image only using 2 clusters, and the image itself is simple (only coins and a grey background), making it easy to find clusters. On the other hand, we see larger differences for the other images between different algorithms. For the lena image, FA is the second best method performing 1.5539 below k-means. The worst method is DE, performing 4.6756 below k-means. For the tiger image, SA is the second best method performing 2.0834 below k-means. The worst method is BA, performing 4.3834 below k-means. For the cameraman image, FA is the second best method performing 0.7376 below k-means. The worst method is PSO performing 4.1057 below k-means.

Table 2 summarizes the results for the inter-cluster d_{min} distance metric. In contrast to $PSNR$ results, k-means does not perform the best across all images for d_{min}. We again notice that the performance of all the algorithms on the coins image are close to each other. For the lena image, the best method is k-means, the second best method is BA performing 0.0247 below k-means, and the worst method is DE performing 0.0646 below k-means. For the tiger image, the best method is PSO, the second best method is SA performing 0.0123 below PSO, and the worst method is DE performing 0.0479 below PSO. For the cameraman image, the best method is k-means, the second best method is DE performing 0.0102 below k-means, and the worst method is PSO performing 0.0706 below k-means.

Table 3 summarizes the results for the intra-cluster $\bar{d}_{max}$ distance metric. We see again that the performance of all the algorithms on the coins image are close to each other. For the lena image, the best method is FA, the second best method is k-means performing 0.0015 above FA, and the worst method is DE performing 0.0441 above FA. For the tiger image, the best method is k-means, the second best method is SA performing 0.0105 above k-means, and the worst method is DE performing 0.05 above k-means. For the cameraman image, the best method is k-means, the second best method is FA performing 0.0041 above k-means, and the worst method is BA performing 0.0516 above k-means.

Table 3: Intra-cluster distance ($\bar{d}_{max}$) results for different clustering algorithms and different images. We show the mean ($\pm$ standard deviation) over 5 trials. A lower Intra-cluster distance corresponds to better results. Bold numbers show best values, and underlined numbers show second best values.

METHODS	Lena	Tiger	Cameraman	Coins
K-means	<u>0.1180</u> ($\pm$ 0.0000)	**0.1249** ($\pm$ 0.0052)	**0.0981** ($\pm$ 0.0071)	0.1251 ($\pm$ 0.0002)
SA	0.1491 ($\pm$ 0.0459)	<u>0.1354</u> ($\pm$ 0.0131)	0.1399 ($\pm$ 0.0950)	**0.1246** ($\pm$ 0.0003)
DE	0.1606 ($\pm$ 0.0060)	0.1749 ($\pm$ 0.0174)	0.1264 ($\pm$ 0.0112)	0.1258 ($\pm$ 0.0023)
PSO	0.1345 ($\pm$ 0.0174)	0.1404 ($\pm$ 0.0077)	0.1278 ($\pm$ 0.0158)	0.1261 ($\pm$ 0.0018)
FA	**0.1165** ($\pm$ 0.0159)	0.1421 ($\pm$ 0.0109)	<u>0.1022</u> ($\pm$ 0.0102)	<u>0.1250</u> ($\pm$ 0.0005)
BA	0.1409 ($\pm$ 0.0190)	0.1468 ($\pm$ 0.0132)	0.1497 ($\pm$ 0.0848)	0.1254 ($\pm$ 0.0026)

(a) (b) (c)

(d) (e) (f)

Figure 2: Segmented images for "lena" using 6 clusters. (a) k-means. (b) Simulated Annealing. (c) Differential Evolution. (d) Particle Swarm Optimization. (e) Firefly Algorithm. (f) Bat Algorithm

Figures 2, 3, 4 ,5 show qualitative results for lena, tiger, cameraman, and coins respectively. Since we use the mean pixel value of each cluster, a good quality segmentation should look similar to the original image (or as close as it can be). We show the results obtained by the trial with the best $PSNR$. We can see that for the coins image (Fig 5), all the methods find similar clusters where the background is separated from the coins. This correlates with the PSNR results on this specific image where all the methods performed similar. For the cameraman image (Fig 4), we that k-means, SA, FA and BA have similar clusters. This correlates with PSNR results where DE and PSO are the worst performers. For the tiger image (Fig 3), we can see that k-means, SA, and PSO show better visual results than the other algorithms. We see that DE, FA, and BA did not work well on this specific image. We include some comments on the discussion section about this finding. Finally, for the lena image (Fig 2), we see that k-means and PSO look the best visually, while DE and BA are the worst. The best method here was FA for the PSNR metric, interestingly the segmented image for FA doesn't look the best visually.

(a) (b) (c)

(d) (e) (f)

Figure 3: Segmented images for "tiger" using 8 clusters. (a) k-means. (b) Simulated Annealing. (c) Differential Evolution. (d) Particle Swarm Optimization. (e) Firefly Algorithm. (f) Bat Algorithm

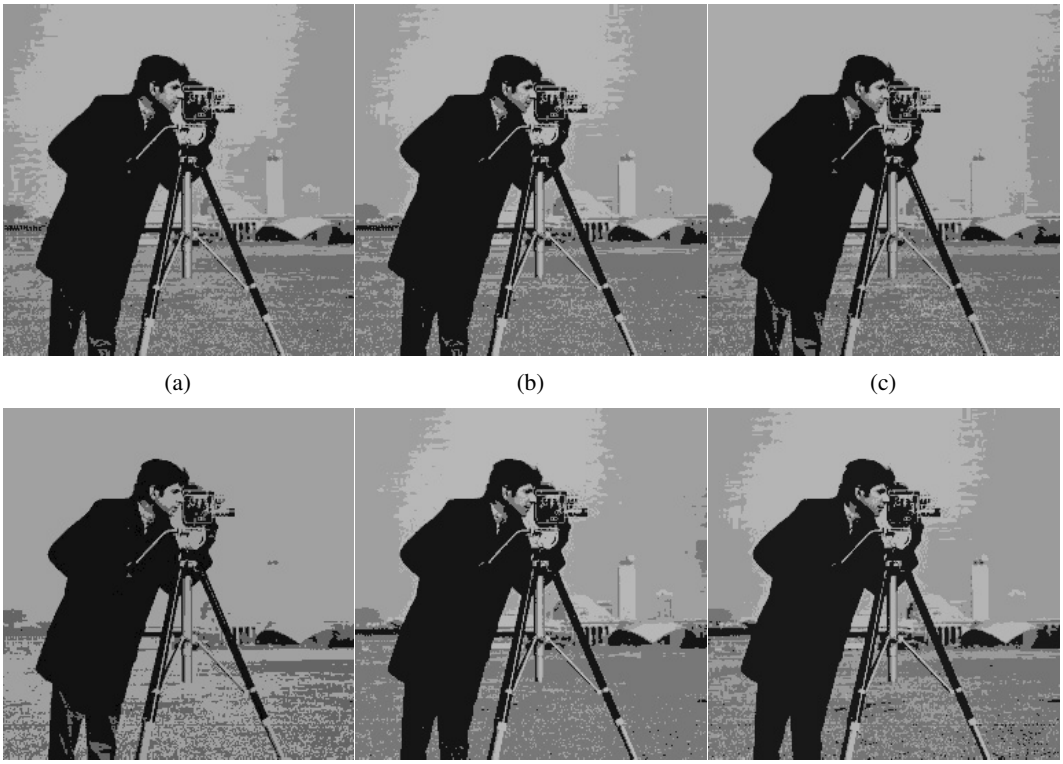
7 Discussion

We noticed that the standard deviation of the metaheuristic algorithms results is higher than k-means. This would suggest that k-means is more robust against random initialization. We believe that the metaheuristic algorithms need further tuning like higher number of iterations and a bigger population size. With these modifications, the standard deviation should decrease. Due to time constraints, we did not fine-tune the algorithms further (bigger population size and higher number of iterations increases running time). Anecdotally, more iterations did not necessarily cause algorithms to converge to better values, but it did increase the consistency with which they located optima. This would almost certainly solve the issue of higher standard deviation for metaheuristic algorithms.

Another interesting trend is the over-performance of Simulated Annealing for the tiger image. Our expectation prior to this exploration was that SA would be somewhat of a baseline performance, with evolutionary and swarm algorithms outperforming it for the more complicated images. On Lena, this expectation is realized, with either PSO, BA, or FA always outperforming it. However, for the tiger, SA is consistently the second best performer. It is our belief that this is due to the unique nature of the solution space of the tiger image, where SA is able to find local optima more effectively than algorithms such as FA or BA. If the global optima was known, and population vectors could be initialized in a Gaussian distribution around that global optima, it is likely that more complicated algorithms would be able to find a better solution. Additionally, the algorithms could be allowed to train longer, but there is no guarantee that would change the dynamic of the algorithms. Also, the goal of this exploration was to be fair to all algorithms, and running algorithms with vastly different parameters and level of tuning would be an unfair comparison. What this may reveal is an application example of the No Free Lunch theorem. While the more complicated algorithms perform better on some tasks, there will always be some tasks that they do not perform as well on. The tiger image seems to be one of those tasks, and SA was able to perform better.

After conducting the experiment, we also noticed that every algorithm performs almost identically when segmenting the coins. In hindsight, this result was to be expected, as two clusters were used and it is very easy to differentiate the background and the coins. This image can provide a decent control experiment - if an algorithm cannot properly cluster this then it certainly needs to be fixed or tuned - but it is not effective as a differentiator between the algorithms. Perhaps in future work this would solely be used as a initialization test, and we would find another reasonable image for the final results.

Another interesting takeaway is the validity of the testing metrics. As discussed in the proposed solution, quantization error was selected as a broad metric to assess segmentation accuracy as a



(a) (b) (c)

(d) (e) (f)

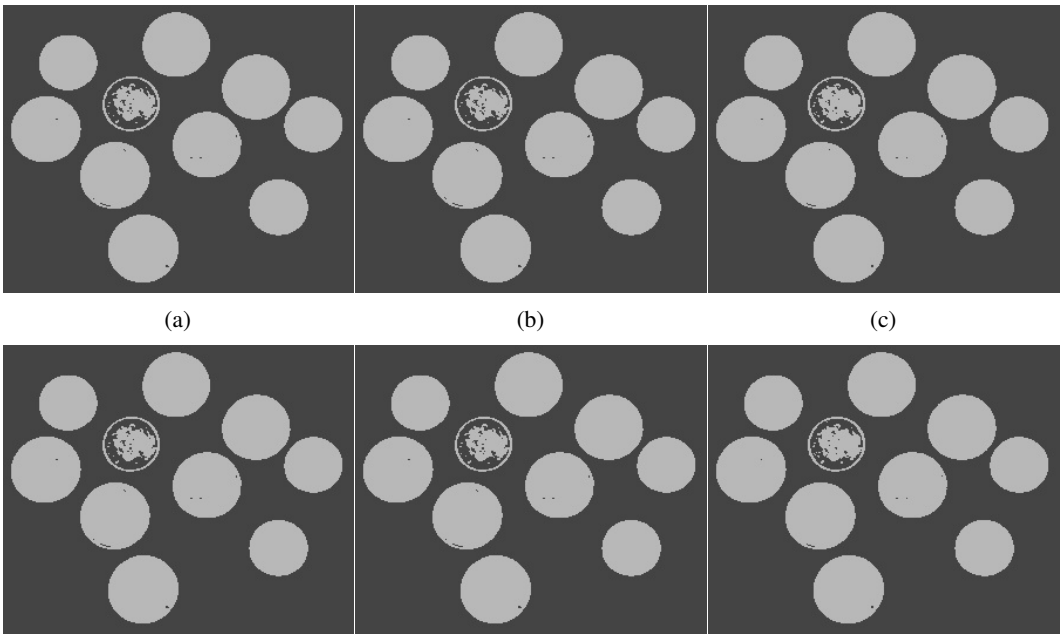
Figure 4: Segmented images for "cameraman" using 4 clusters. (a) k-means. (b) Simulated Annealing. (c) Differential Evolution. (d) Particle Swarm Optimization. (e) Firefly Algorithm. (f) Bat Algorithm

whole. This average-based fitness function appears to have been effective, based on the fact that every algorithm saw reasonable results as compared to k-means with the other test metrics. However, another test of success is a subjective one. The goal of image segmentation is to determine the best clusters to group data into so that the original data is still well-represented. It is pretty obvious in the tiger image that k-means and SA create reasonable representations of the original image, and FA and BA do not. This categorization mostly agrees with the results for tiger for PSNR, intra-cluster and inter-cluster distance. But a subjective categorization may not agree with the results for lena. k-means scores best on the test metrics for lena, and it appears to be the best representation additionally. However, the second best performer - FA - seems to be a worse representation of the original image than SA or PSO. These both had relatively average scores on the test metrics, but the eye-test would say they performed better than FA. Perhaps the human analysis is flawed and the metrics capture more meaningful information for data analysis than the human vision system, but the goal of image segmentation is often to represent the information encoded in the original image in a reduced but still meaningful way. Perhaps instead, additional metrics should be included in testing to more definitively analyze the goodness of segmentation.

8 Conclusion

This paper has shown that direct metaheuristic optimization can be used to segment images. We tested several metaheuristic algorithms under the same conditions and measure their performance with metrics not used in the fitness function. Our results show that applying metaheuristic methods on image segmentation has comparable success to k-means. Additionally, this paper has explored the dynamic between metaheuristic algorithms, showing that each has applications and sub-problems that it may be optimal for.

As future work, it would be interesting to explore more complex representations of each pixel of the image instead of just using the RGB values. For example, adding features like the row and column



(a) (b) (c)

(d) (e) (f)

Figure 5: Segmented images for "coins" using 2 clusters. (a) k-means. (b) Simulated Annealing. (c) Differential Evolution. (d) Particle Swarm Optimization. (e) Firefly Algorithm. (f) Bat Algorithm

positions would introduce spatial information to the clustering method, or using semantic labels of each pixel would introduce some semantics to the system. These extra features can potentially improve results. Additionally, the images used in this analysis are classic image segmentation problems that are used as a benchmark or proof-of-concept for a process. The next test case for comparing metaheuristic algorithms against k-means would be on challenging real-life applications, such as medical imaging or video surveillance. Finally, it would be interesting to see an analysis on the different possible fitness functions that can be used for pixel clustering with metaheuristic algorithms. Defining a good fitness function is a key step on getting a metaheuristic method to work well on pixel clustering.

References

[1] L. Agustin-Blas, S. Salcedo-Sanz, S. Jiménez-Fernández, L. Carro-Calvo, J. Del Ser, and J. Portilla-Figueras. A new grouping genetic algorithm for clustering problems. *Expert Syst. Appl.*, 39(10):9695–9703, aug 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.02.149. URL https://doi.org/10.1016/j.eswa.2012.02.149.

[2] A. Arjmand, S. Meshgini, R. Afrouzian, and A. Farzamnia. Breast tumor segmentation using k-means clustering and cuckoo search optimization. In *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 305–308, 2019. doi: 10.1109/ICCKE48569.2019.8964794.

[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[4] S. A. Burney and H. Tariq. K-means cluster analysis for image segmentation. *International Journal of Computer Applications*, 96(4), 2014.

[5] L. Jiao, M. Gong, S. Wang, B. Hou, Z. Zheng, and Q. Wu. Natural and remote sensing image segmentation using memetic computing. *IEEE Computational Intelligence Magazine*, 5(2): 78–91, 2010. doi: 10.1109/MCI.2010.936307.

[6] Y. Kao and S.-Y. Lee. Combining k-means and particle swarm optimization for dynamic data clustering problems. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1, pages 757–761. IEEE, 2009.

[7] U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(99)00137-5. URL https://www.sciencedirect.com/science/article/pii/S0031320399001375.

[8] D. Merwe and A. Engelbrecht. Data clustering using particle swarm optimization[c]. volume 1, pages 215–220, 01 2003. doi: 10.1109/CEC.2003.1299577.

[9] H. Mittal, A. C. Pandey, M. Saraswat, S. Kumar, R. Pal, and G. Modwel. A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. *Multimedia Tools and Applications*, pages 1–26, 2021.

[10] S. Z. Selim and K. Alsultan. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008, 1991. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(91)90097-O. URL https://www.sciencedirect.com/science/article/pii/003132039190097O.

[11] A. Sharma and S. Sehgal. Image segmentation using firefly algorithm. In *2016 International Conference on Information Technology (InCITe)-The Next Generation IT Summit on the Theme-Internet of Things: Connect your Worlds*, pages 99–102. IEEE, 2016.

[12] F. Sun and Y. Tian. Transmission line image segmentation based ga and pso hybrid algorithm. *2010 International Conference on Computational and Information Sciences*, pages 677–680, 2010.

[13] M. T. Wong, X. He, and W.-C. Yeh. Image clustering using particle swarm optimization. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 262–268. IEEE, 2011.

[14] N. M. Zaitoun and M. J. Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015.

[15] B. Zhao, Z. Zhu, E. Mao, and Z. Song. Image segmentation based on ant colony optimization and k-means clustering. In *2007 IEEE International Conference on Automation and Logistics*, pages 459–463, 2007. doi: 10.1109/ICAL.2007.4338607.

[16] W. Zou, Y. Zhu, H. Chen, and X. Sui. A clustering approach using cooperative artificial bee colony algorithm. *Discrete Dynamics in Nature and Society*, 2010:459796, Nov. 2010.