Applied Soft Computing 55 (2017) 503–522

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Multilevel thresholding based on Chaotic Darwinian Particle Swarm Optimization for segmentation of satellite images

Shilpa Suresh, Shyam Lal*

Department of Electronics & Communication Engineering, National Institute of Technology Karnataka, Surathkal, Mangaluru 575025, India

ARTICLE INFO

Article history:
Received 24 June 2016
Received in revised form
29 December 2016
Accepted 2 February 2017
Available online 10 February 2017

Keywords:
Chaotic sequence
Minimum cross entropy
Tsallis entropy
Metaheuristic algorithms
Convergence rate

ABSTRACT

This paper proposes an improved variant of Darwinian Particle Swarm Optimization algorithm based on chaotic functions. Most of the evolutionary algorithms faces the problem of getting trapped in local optima in its search for global optimum solutions. This is highly influenced by the use of random sequences by different operators in these algorithms along their run. The proposed algorithm replaces random sequences by chaotic sequences mitigating the problem of premature convergence. Experiments were conducted to investigate the efficiency of 10 defined chaotic maps and the best one was chosen. Performance of the proposed Chaotic Darwinian Particle Swarm Optimization (CDPSO) algorithm is compared with chaotic variants of optimization algorithms like Cuckoo Search, Harmony Search, Differential Evolution and Particle Swarm Optimization exploiting the chosen optimal chaotic map. Various histogram thresholding measures like minimum cross entropy and Tsallis entropy were used as objective functions and implemented for satellite image segmentation scenario. The experimental results are validated qualitatively and quantitatively by evaluating the mean, standard deviation of the fitness values, PSNR, MSE, SSIM and the total time required for the execution of each optimization algorithm.

1. Introduction

Nature inspired metaheuristic algorithms have found its place in solving many global optimization problems efficiently which are applicable to different scenarios [1]. The supreme problem faced by many of them is its premature convergence by which it will get trapped in some local optima. Chaotic functions diversify the solution space making space to explore and exploit more.

Most of the works so far in this area have exploited the possibility of improving the convergence rate by replacing the initial random sequences by chaotic functions since all evolutionary algorithms start its initial iteration from a random set of solution space. In our proposed Chaotic DPSO algorithm, we have explored the chaotic characteristics with some prior knowledge of the existing scenario to enhance the local searching capability for segmenting satellite images.

Chaotic behavior is commonly exhibited by dynamic non-linear systems defined as the randomness shown by the system. Particularly, in swarm based dynamic systems, the intelligence emerges from a chaotic balance between individuality and sociality. Chaotic dynamic systems eventually reach the global optimum or it is highly probable to reach its good approximation tested against a given cost function by tracing these chaotic ergodic orbits [2]. Chaotic systems can be mathematically represented in the form $x_{k+1} = f(x_k)$. They are characterized by intrinsic stochastic property and ergodic nature which is highly influenced by the initial value of the function denoted as x_0 [3]. Chaotic dynamics is characterized by a deterministic framework with non-normal, turbulent conduct [4]. They are both delicate to initial conditions and computational uncertainty. The Lyapunov exponent and correlation dimension are the most frequently used numerical calculations to qualitatively distinguish the dynamics of a system [5]. Lyapunov exponents give the rate at which the neighboring trajectories converge or diverge from each other in orthogonal directions. The correlation dimension of a chaotic system gives the measure of its complexity or geometric scaling property which is considered as one among its most basic properties. Lui et al. have done an extensive study highlighting the efficiency in introducing chaos in swarm intelligence techniques [6].

It is now an active area of metaheuristic algorithms to make use of the non-repetitious and ergodic nature shown by chaotic maps to tackle the problem of premature convergence shown by many optimization problems. Literature reveals its effectiveness in enhancing the performance of many evolutionary algorithms which includes

* Corresponding author.
E-mail addresses: shilparagesh89@gmail.com (S. Suresh), shyam.mtec@gmail.com (S. Lal).

http://dx.doi.org/10.1016/j.asoc.2017.02.005
1568-4946/© 2017 Elsevier B.V. All rights reserved.

Genetic algorithm [7,8], Simulated Annealing algorithm [9,10], Differential Evolution algorithm [11,12], Particle Swarm Optimization algorithm [13,14], Firefly algorithm [15,16], Harmony Search algorithm [17,18], Ant Colony Optimization algorithm [19,20], Cuckoo Search algorithm [21,22], Bat algorithm [23,24] and Artificial Bee Colony Optimization algorithm [17,25].

This paper also ensures addressing the following questions.

- To what extend the chaotic strategies mitigate premature convergence problem?
- What is its impact on different quality metric values used to quantify segmented images?
- Among the set of 10 different chaotic operators discussed in this paper, which one proves to be best suited for satellite image segmentation?
- Does chaotic DPSO outperform other similar algorithms in comparison?

In 2003, Caponetto et al. examined the effect of introducing chaotic sequences instead of random numbers in different phases of evolutionary processes [26]. It revealed the improvement in performance indices for sensing the convergence rate of those algorithms. In 2005, Liu et al. came up with the possibility of using chaotic maps for reducing the processing time of PSO algorithm by avoiding its premature convergence [27]. Chandramouli et al. extended the use of chaotic maps in modeling PSO for image classification and quantitatively presented its improvement in efficiency over classical approach [28]. In 2007, Yuan et al. put forward a novel approach to enhance global searching capability of PSO for optimum points by embedding the favorable aspects of chaotic strategy which proved to be very efficient in its application for traveling salesman problem [29]. Later, in 2010, Hefny et al. also presented his work on significantly improving the solution search efficiency of PSO algorithm by employing chaotic agents to explore the promising areas [13]. In 2015, Zhang et al. presented a detailed comprehensive survey on PSO algorithm and its applications in various domains [30].

Differential evolution (DE) evolved as a greedy stochastic metaheuristic algorithm for solving intricate optimization problems. A new approach combining DE algorithm with chaotic sequences and sequential quadratic programming (SQP) technique was propounded by Coelho et al. in 2006, to optimize the performance of economic dispatch problems [31]. Experiments were carried out combining chaotic sequences with Differential Evolution (DE) focusing on the potential of logistic chaotic maps in avoiding local optimum traps for contrast enhancement scenarios [11].

In 2010 Chaotic Harmony Search (CHS) algorithm was proposed, in which authors employed chaotic maps for initializing the Harmony Memory (HM) and for updating various parameters involved in the algorithm. This helped in improving the convergence rate thereby the solution quality and were tested against different benchmark functions [17]. Firefly algorithm was also adapted with chaotic characteristics for reliability and redundancy optimization [15].

Cuckoo Search (CS) algorithm is a popular metaheuristic approach commonly employed in global optimization scenarios. In 2014, Ouyang et al. furnished chaotic maps based scheme for improving the efficiency of CS algorithm to optimize elitist candidates in the population space, in terms of calculation precision. It also helped in reducing the vulnerability of the algorithm to get clogged in local optimum points in its search for optimizing high-dimensional functions [32]. The algorithm proceeds iteratively modeling Lévy flight in search of optimum solutions with a scaling factor and selective random walk with a fractional probability. Unfortunately the presence of constant terms in both these phases biased the algorithm to a greater extend which got reflected in the calculation of stability and convergence speed. Wang et al. illustrated the use of chaotic maps in all these phases which improved the solution space considerably [22]. In the same year, Wang & Deb published their work in combining chaotic theory with Cuckoo Search algorithm with an intention to enhance its performance [21]. Step size variation using 12 different chaotic operators were tested and evaluated against 27 benchmark functions.

Recently, in the year 2016, Adarsh et al. put forward a chaotic variant of Bat algorithm for solving non-linear discontinuous economic dispatch problem. The authors also demonstrated the applicability of the proposed algorithm for high dimensional problems [33]. Dhal et al. proposed two modifications for Firefly Algorithm (FA) based on chaotic sequences for enhancing the image contrast. As the first modification, the authors experimented the use of Lévy flights in FA, wherein, the step sizes where drawn from chaotic sequences. The second modification included the replacement of the local search strategy completely by chaotic search. The second strategy yielded comparatively better performance among the other state-of-the-art algorithms compared [34]. Gokhale et al. proposed a tent chaotic map initiated FA for optimal over current relay coordination problem [35]. Similarly Yi et al. proposed engineering design optimization by chaotic local search enhanced HS algorithm [36]. Lastly, Kaveh published a revised book chapter based on the detail study of chaos embedded metaheuristic algorithms [37].

The forthcoming part of the paper is organized as follows: Section 2 embodies the different multi-level segmentation approaches investigated in our work. Section 3 gives an overview about different chaotic maps and the selection strategy deployed for satellite image segmentation scenario. Section 4 briefly discusses about various chaotic bio-inspired algorithms used in our study. Section 5 presents the proposed chaotic DPSO algorithm highlighting its unique features in our application. Experimental results are investigated and discussed in Section 6. Discussions are concluded in Section 7 emphasizing the efficiency of proposed CDPSO algorithm for satellite image segmentation.

2. Multi-level segmentation techniques

Satellite images accounts for having peculiar characteristics such as ambiguous regions, randomness in intensity distribution and weak local pixel correlation. Spatial autocorrelation gives the relationship among the values of a variable entirely and its fairly close locational positions on a two-dimensional surface. Spatial autocorrelation exists in real-world phenomena typically characterized by its orderliness and systematic concentration, rather than randomness. Whereas, most of the geographic distributions projected on a 2-D surface exhibits a negative or weak spatial correlation between local pixels, making them difficult to segment [38]. Segmentation of such images by bi-level histogram thresholding proves to be inefficient since the foreground and background of the image cannot be distinguished clearly [39]. Hence segmentation of such images are made possible by a technique termed as multi-level thresholding in which it finds multiple threshold values to distinguish our region of interest from its background [40–43] mathematically formulated as:

$$\begin{aligned}
T_0 &= \left\{ f(x,y) \in F | 0 \le f(x,y) \le t_{h1} - 1 \right\}; \\
T_1 &= \left\{ f(x,y) \in F | t_{h1} \le f(x,y) \le t_{h2} - 1 \right\}; \\
T_j &= \left\{ f(x,y) \in F | t_{hj} \le f(x,y) \le t_{h(j+1)} - 1 \right\}; \\
T_n &= \left\{ f(x,y) \in F | t_{hn} \le f(x,y) \le L_{\max} - 1 \right\}
\end{aligned} \tag{1}$$

where F represents the image to segment, $f(x, y)$ represents the pixel intensity specified by the x and y coordinates, L_{max} denotes the total number of gray levels in the image and $t_{hj} = 1, 2, \ldots, n$ where n gives the total count of distinct threshold values.

2.1. Minimum cross entropy

The idea of cross entropy measure was put forward by Kullback [44]. Let $\vec{X} = \{x_1, x_2, \ldots, x_N\}$ and $\vec{Y} = \{y_1, y_2, \ldots, y_N\}$ represent the probability distribution defined on the same set of vales. Then the cross entropy between $\vec{X}$ and $\vec{Y}$ is defined as given in Eq. (2)

$$C_E(\vec{X}, \vec{Y}) = \sum_{i=1}^{N} x_i \log \frac{x_i}{y_i} \tag{2}$$

Let F be the input RGB image to be processed. Let $h(i)$ represent the histogram of corresponding image preserving the color information. We define t_h as the pixel intensity value chosen for bi-level segmentation of the image [45]. It can be mathematically formulated as:

$$F_{t_h}(x, y) = \begin{cases} \mu(1, t_h) & F(x, y) < t_h \\ \mu(t_h, N+1) & F(x, y) \geq t_h \end{cases} \tag{3}$$

where $\mu(k, l) = \left(\sum_{i=k}^{l-1} ih(i)\right) / \left(\sum_{i=k}^{l-1} h(i)\right)$.

Then the cross entropy between these two classes is defined as [45]:

$$C_E(t_h) = \sum_{i=1}^{t_h-1} ih(i) \log\left(\frac{i}{\mu(1, t_h)}\right) + \sum_{i=t_h}^{N} ih(i) \log\left(\frac{i}{\mu(t_h, N+1)}\right) \tag{4}$$

$$\text{i.e.} C_E(t_h) = \sum_{i=1}^{N} ih(i) \log(i) - \sum_{i=1}^{t_h-1} ih(i) \log(\mu(1, t_h)) - \sum_{i=t_h}^{N} ih(i) \log(\mu(t_h, N+1)) \tag{5}$$

The optimum threshold is thus determined by minimizing the cross entropy which is given by:

$$\vec{t_h}^{*} = arg \ \min \left\{C_E(t_h)\right\} \tag{6}$$

First term in Eq. (5) is a constant term. So the objective function can be rewritten as in Eq. (7) and the optimum threshold for segmentation will be the one which minimizes $\nu(t_h)$

$$\nu(t_h) = -\sum_{i=1}^{t_h-1} ih(i) \log(\mu(1, t_h)) - \sum_{i=t_h}^{N} ih(i) \log(\mu(t_h, N+1))$$
$$\text{or} \quad \nu(t_h) = -m^1(1, t_h) \log\left(\frac{m^1(1, t_h)}{m^0(1, t_h)}\right) - m^1(t_h, N+1) \log\left(\frac{m^1(t_h, N+1)}{m^0(t_h, N+1)}\right) \tag{7}$$

where $m^0(k, l) = \sum_{i=k}^{l-1} h(i)$ and $m^1(k, l) = \sum_{i=k}^{l-1} ih(i)$. The same concept can be extended to multilevel thresholding by minimizing (8).

$$\nu(t_{h1}, t_{h2}, \ldots, t_{hn}) = -\sum_{j=1}^{n+1} m^1(t_{h(j-1)}, t_{hj}) \log\left(\frac{m^1(t_{h(j-1)}, t_{hj})}{m^0(t_{h(j-1)}, t_{hj})}\right) \tag{8}$$

Instead of minimizing this function we tried to maximize the negative of this function for our set of experiments.

2.2. Tsallis entropy

According to the concept of thermodynamics, Boltzmann–Gibbs entropy refers to the disorder present in a physical system [46]. Shannon redefined this measure to mathematically model an expression to quantitatively measure the information content associated with a process [46] as given in Eq. (9):

$$\mathbf{S} = -\sum_{i=1}^{m} p_i \ln(p_i) \tag{9}$$

where m is the total number of distinct states and p_i is probability of a pixel value to be i. This expression was valid in a restricted environment predefined by Boltzmann–Gibbs–Shannon (BGS) statistics [47]. As a part of extending this concept to a system which involves processes which take in long term memory, interactions and fractal structures, Tsallis proposed a new statistical approach generalizing BGS statistics [47]. Tsallis defined the entropy associated with a system as given in Eq. (10):

$$S_q = k \frac{1 - \sum_{i=1}^{m} p_i^q}{q - 1} \tag{10}$$

where the possibility of being in state i is denoted as $p_i \in [0, 1]$ and q is Tsallis entropy index which represents the non-extensivity of the system. When $q \to 1$, system approaches BGS statistics as given in Eq. (11) using replica-trick type expansion [48,49].

$$S_1 = \lim_{q \to 1} S_q = k \lim_{q \to 1} \frac{1 - \sum_{i=1}^{m} p_i \ \exp[(q-1) \ln p_i]}{q - 1} = -k \sum_{i=1}^{m} p_i \ln(p_i) \tag{11}$$

For a system which is statistically independent, Tsallis entropy is non-extensive and the entropy associated with the system can be defined as in Eq. (12) following pseudo-additive property.

$$S_q(F + B) = S_q(F) + S_q(B) + (1 - q) S_q(F) S_q(B) \tag{12}$$

where F and B represents two independent subsystems/subclasses.

Entropy associated with a system can be broadly classified into three depending upon the value of q considering $S_q > 0$ as:

- Extensive ($q = 1$): $S_q(F+B) = S_q(F) + S_q(B)$.
- Super-extensive ($q > 1$): $S_q(F+B) < S_q(F) + S_q(B)$.
- Sub-extensive ($q < 1$): $S_q(F+B) > S_q(F) + S_q(B)$.

The main drawback of Tsallis entropy is the selection of the 'q' index. Since it is not an intuitive idea, several applications sort to choose its value randomly. Till date, to the best of our knowledge, there is no automatic method or theory proposed for automating its computation specific to any application [50]. Hence as in our case, each image or region may demand for a different q value in order to achieve information maximization. Hence as a part of prior empirical study, we have computed entropy value as given in Eq. (10) for each image with several ranges of q and found $q = 0.5$ to be a proper selection for the set of images tested.

This method can be easily incorporated in image segmentation scenarios, either gray scale or color images. Multilevel image thresholding by Tsallis entropy method [51–53] thus can be formulated as

$$\begin{aligned}
S_q^{c_0}(t_h) &= \frac{1-\sum_{i=0}^{t_{h1}-1}\left(\frac{p_i^c}{\sum_{i=0}^{t_{h1}-1}p_i^c}\right)}{q-1};\\
S_q^{c_1}(t_h) &= \frac{1-\sum_{i=t_{h1}}^{t_{h2}-1}\left(\frac{p_i^c}{\sum_{i=t_{h1}}^{t_{h2}-1}p_i^c}\right)}{q-1}\\
S_q^{c_j}(t_h) &= \frac{1-\sum_{i=t_{hj}}^{t_{h(j+1)}-1}\left(\frac{p_i^c}{\sum_{i=t_{hj}}^{t_{h(j+1)}-1}p_i^c}\right)}{q-1};\\
S_q^{c_n}(t_h) &= \frac{1-\sum_{i=t_{hn}}^{N-1}\left(\frac{p_i^c}{\sum_{i=t_{hn}}^{L-1}p_i^c}\right)}{q-1}
\end{aligned} \tag{13}$$

where $c=1, 2, 3$ for color images and 1 for gray scale images, to yield the optimum threshold values as depicted in Eq. (14).

$$\begin{aligned}
\left[\vec{t_{h1}}^*, \ldots, \vec{t_{hn}}^*\right] &= arg\ \max[S_q^{c_0}(t_h)+S_q^{c_1}(t_h)+\cdots+S_q^{c_n}(t_h)\\
&+(S_q^{c_0}(t_h).S_q^{c_1}(t_h)\cdots S_q^{c_n}(t_h)).(1-q)]
\end{aligned}$$

subject to

$$\begin{aligned}
&|P^{c_0}+P^{c_1}|-1 < S^{c_0} < 1-|P^{c_0}+P^{c_1}|;\\
&|P^{c_1}+P^{c_2}|-1 < S^{c_1} < 1-|P^{c_1}+P^{c_2}|;\\
&|P^{c_{(n-1)}}+P^{c_n}|-1 < S^{c_{(n-1)}} < 1-|P^{c_{(n-1)}}+P^{c_n}|
\end{aligned} \tag{14}$$

$P^{c_0}, P^{c_1}, \ldots, P^{c_n}$ can be formed from the probability distribution of pixel values corresponding to the threshold levels $\vec{t_{h1}}^*, \ldots, \vec{t_{hn}}^*$ given by:

$$P^{c_0}=\sum_{i=0}^{t_{h1}-1}p_i^c;\quad P^{c_1}=\sum_{i=t_{h1}}^{t_{h2}-1}p_i^c;\quad \ldots P^{c_n}=\sum_{i=t_{hn}}^{L_{\max}-1}p_i^c \tag{15}$$

3. Chaotic maps for metaheuristics

Many recent studies discusses on the possibilities of combining chaotic sequences and metaheuristic optimization algorithms for different applications. Some of them indented to depict the chaotic characteristics of metaheuristic algorithms, whereas, some others explored the use of chaos in overcoming the limitations of many metaheuristic algorithms. Hence embedding chaos into such metaheuristic optimization algorithms can be broadly divided into two classes.

As we know, almost all metaheuristic algorithms commonly includes a population initialization phase using a random number generator which draws values from a uniform or Gaussian probability density function to achieve its random nature. The first class incorporates chaotic sequence generator instead of a random number generator which efficiently controls the convergence characteristics of the metaheuristic algorithm [54,55]. Whereas, in the second class, chaotic search is integrated into the metaheuristics, which helps in enriching the searching capability and in avoiding the premature local optimum convergence problems of the algorithm [3].

The choice of chaotic sequences is theoretically justified by their unpredictable nature, i.e., their ergodic properties and spread-spectrum characteristic [56]. A chaotic motion traverses every possible states in a particular search region, ensuring its visit to each stage only once. It is highly sensitive to its initial conditions, means small changes in the bringing a large change in chaotic moves for a small change in initial conditions, termed as butterfly effect. Chaotic search has a very special ability to avoid being trapped in local optima [57].

Ten different one-dimensional chaotic maps were investigated to choose the optimum one among them [26]. Table 1 explains the mathematical formulation and general behavior of these operators for 50 consecutive iterations obeying certain constraints. Formulation of all these 10 chaotic maps ensures the generated sequence to fall within the range [0,1] which is best suited for our application.

3.1. Selection of chaotic maps

We experimentally evaluated those 10 different chaotic maps, comparing parameters such as standard deviation, mean and CPU running time which reflects the potential of each of them in converging to the global optimum threshold values for satellite image segmentation scenario.

Studies revealed that the performance of these chaotic maps proved to be consistent for all optimization algorithms used in this paper tested for three different threshold levels ($n=3$, 4, 5). Table 2 furnishes the test results of comparing the performance of the above cited 10 different chaotic maps; incorporating them into PSO and DPSO algorithm for 5-level satellite image segmentation using minimum cross entropy as objective function.

Based on the investigated results from Table 2, we have incorporated logistic chaotic map for enriching the local searching capability of all five optimization algorithms in the study to reduce the possibility of getting tapped in local optima. The logistic map is mathematically defined as $x_{i+1}=ax_i(1-x_i)$ where x_i denotes the ith sample and $0<a\leq 4$ is the control parameter which determines the behavioral characteristics of the map [11]. Thus depending on the value of a the sequence x can have fixed length, variable length limited by some constraints or even behaves in an unpredictable manner. The initial value $x_0 \notin \{0.0, 0.25, 0.75, 0.5, 1.0\}$ also brings about substantial variation in the long term behavior of the chaotic sequence. So in our set of experiments we have defined the value of $a=4$ and $x_0=0.7$.

From the above comparison, optimization algorithms using logistic chaotic map emerged to be most stable and efficient in finding optimum threshold values for satellite image segmentation converging to its global optimum with a much higher rate without getting clogged in local optima.

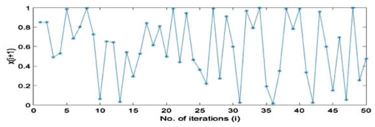
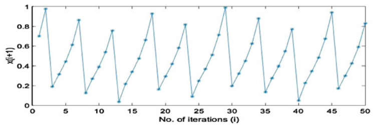
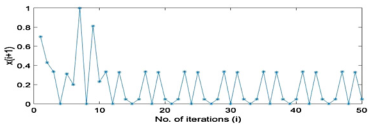
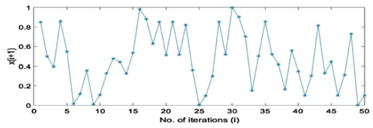
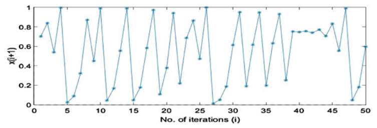
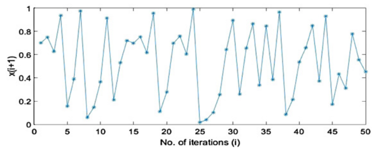
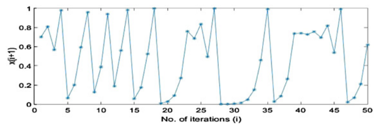
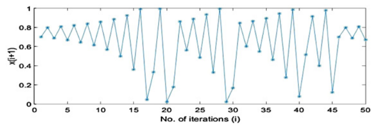
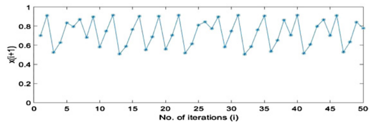
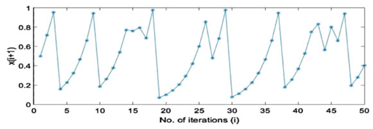
4. Chaotic bio-inspired algorithms

4.1. Chaotic Cuckoo Search (CCS) algorithm

Entomologists were attracted by the peculiar reproduction strategy shown by certain species of cuckoos by laying eggs in the nest of other birds. Based on such observation, Yang et al. [58] put forward a metaheuristic approach named as Cuckoo Search (CS) algorithm for global optimization problems. Chaotic CS adopts chaotic sequences for building the initial solution space [21,22]. This lowers the paramount problem of the algorithm being getting trapped in local optima.

CCS can be visualized as a two-stage procedure in which the Cuckoo Search algorithm works as the backbone carrying out the global search whereas chaotic operator extends the possibility of local search for optimum points. Chaotic sequences are exploited for generating the initial set of population.

Table 1
Different chaotic maps for metaheuristics.

Equation	Parameters/constraints	Plot
	I.1. Chebyshev	
$x_{i+1} = cos(icos^{-1}(x_i))$	$x \in [0, 1]$	
	I.2. Circle	
$x_{i+1} = x_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_i)\mathrm{mod}(1)$	$a = 0.5$, $b = 0.2$ then $x \in [0, 1]$	
	I.3.Gauss/mouse	
$x_{i+1} = \begin{cases} 0 & x_i = 0 \\ \dfrac{1}{x_i \mathrm{mod}(1)} & \text{otherwise} \end{cases}$	$x \in [0, 1]$	
	I.4. Iterative	
$x_{i+1} = \sin\left(\frac{a\pi}{x_i}\right)$	$a \in [0, 1], x \in [0, 1]$	
	I.5.Logistic	
$x_{i+1} = ax_i(1 - x_i)$	$x \in [0, 1]; x_0 \notin \left\{0.0, 0.25, 0.75, 0.5, 1.0\right\}; 0 < a \leq 4$	
	I.6. Piecewise linear	
$x_{i+1} = \begin{cases} \dfrac{x_i}{P} & 0 \leq x_i < P \\ \dfrac{X_i - P}{0.5 - P} & P \leq x_i < \dfrac{1}{2} \\ \dfrac{1 - P - x_i}{0.5 - P} & \dfrac{1}{2} \leq x_i < 1 - P \\ \dfrac{1 - x_i}{P} & 1 - P \leq x_i < 1 \end{cases}$	$P \in [0, 0.5], P \neq 0, x \in [0, 1]$	
	I.7.Sine	
$x_{i+1} = \frac{a}{4}\sin(\pi x_i)$	$0 < a \leq 4, x \in [0, 1]$	
	I.8.Singer	
$x_{i+1} = \mu\left(7.86x_i + -23.31x_i^2 + 28.75x_i^3 - 13.3x_i^2\right)$	$0.9 \leq \mu \leq 1.08, x \in [0, 1]$	
	I.9.Sinusoidal	
$x_{i+1} = ax_i^2\sin(\pi x_i)$	$a = 2.3, x \in [0, 1]$	
	I.10.Tent	
$x_{i+1} = \begin{cases} \dfrac{x_i}{a} & x_i < 0.7 \\ b(1 - x_i) & x_i \geq 0.7 \end{cases}$	$a = 0.7, b = 10/3, x \in [0, 1]$	

Algorithm 1. Chaotic Cuckoo Search algorithm.

```
1  Initialize population $x_{i,j}$; $i \in \{1,2,....N\}$, $j \in \{1,2,....n\}$
     using Choatic maps and set up the parameters via eqn.
     (16);
2  Compute the fitness value of each nest using the defined
     objective function $f(x)$; $x = [x_1, x_2....x_n]^T$;
3  while (iter < Maxiter) or (stopping criterion) do
4      Generate new population retaining the current best;
5      Evaluate the fitness value and record best nest;
6      if $k < p_a$ then
7          Replace worst nests by Lévy flight modeled by
             Chaotic sequence via eqn. (17);
8          Evaluate the fitness value and record best nest;
9          Update the Counter;
10         Find best fitness value so far;
11     else
12         Retain those nests;
13     end
14 end
15 Find the optimum solutions;
```

The population is initialized incorporating chaotic sequences as given in Eq. (16)

$$x_{i,j} = x_i^l + chaos(0,1)(x_i^h - x_i^l) \tag{16}$$

where $i \in \{1, 2, \ldots, N\}$; ($N$ = *number ofnests*), $j \in \{1, 2, \ldots, n\}$; ($n$ = number of birds in each nest) and $[x_i^l, x_i^h]$ represents the lower and upper bounds for the candidate solutions. '*chaos*(0, 1)' indicate elements in the chaotic sequence generated using the defined chaotic operator. In each iteration p_a denotes the probability of cuckoo's eggs being discovered by the host birds and the solution space is altered by modeling *Lévy* flight using Eq. (17)

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \alpha L\hat{e}vy(\beta) \tag{17}$$

where α denotes the step size which is generated using the defined chaotic function and $L\acute{e}vy(\beta) = t^{1-\beta}; 1 < \beta \le 2$ following *Lévy* distribution. Pseudo code given above explains the flow of CCS algorithm for finding global optima [1].

4.2. Chaotic Harmony Search (CHS) algorithm

In Harmony Search (HS) algorithm each solution is termed as harmony which represents a n-dimensional vector [59]. Chaotic harmony search generates its initial population of size $H_{size}Xn$ using chaotic maps and are stored in a harmony memory (HM) [17,18] mathematically formulated as in Eq. (18)

$$x(i) = Lb(i) + (Ub(i) - Lb(i)).chaos(0,1); \tag{18}$$

where $Lb(i)$ and $Ub(i)$ represent the bounds for $x(i)$.

Algorithm 2. Chaotic Harmony Search algorithm.

```
1  Initialize HM via eqn. (18) and set other parameters;
2  Evaluate the fitness value of each harmony vectors;
3  while (iter < Improvisation Count (IC)) do
4      do
5          $r_1$= $chaos[0,1]$;
6          if $r_1$ < HMCR (memory consideration) then
7              $x_{new}(i)$ will be randomly chosen from HM
                 via eqn.(19);
8              $r_2$= $chaos[0,1]$ ;
9              if $r_2$ < PAR(pitch adjustment) then
10                 $r_3$= $chaos[0,1]$;
11                 $x[i,j] = x[i,j] \pm r_3 * BW$ via eqn.
                     (20);
12             end
13             if $x_{new}(i) < Lb(i)$ then
14                 $x_{new}(i) = Lb(i)$;
15             end
16             if $x_{new}(i) > Ub(i)$ then
17                 $x_{new}(i) = Ub(i)$;
18             end
19         else
20             $x_{new}(i) = Lb(i)+(Ub(i)-Lb(i)).chaos(0,1)$
                 via eqn.(19) ;
21         end
22     while j=1 to no. of decision variables(n);
23     Update HM via eqn. (21);
24 end
25 Find optimum solutions;
```

The parameters used includes the Harmony Memory (HM), Harmony Memory Consideration Rate (HMCR), Pitch Adjusting Rate (PAR), Distance Bandwidth (BW), and Improvisations Count (IC). IC represents the total number of iterations which are highly dependent on the application domain [60].

Improvisation phase builds a new harmony from the solution space by applying *Memory Consideration (MC) or Random Re-initialization (RR)* strategy mathematically modeled as:

$$x_{new}(i) = \begin{cases} x_i(j) \in \{x_1(i), x_2(i), \ldots, x_{Hsize}(i)\} & \text{if } r_1 \le HMCR \\ Lb(i) + (Ub(i) - Lb(i)).chaos(0,1) & \text{if } r_1 > HMCR \end{cases} \tag{19}$$

Each new solution thus formed are reanalyzed to check the requirement of pitch adjustment. Pitch Adjusting Rate (PAR) defines the pitch adjustment frequency and Distance Bandwidth

(BW) controls the local search around the selected HM components [60]. Updated components after this phase can be formulated as

$$x_{new}(j) = \begin{cases} x_{new}(j) \pm chaos(0,1).BW & \text{with} \quad prob \quad PAR \\ x_{new}(j) & \text{with} \quad prob \quad (1-PAR) \end{cases} \tag{20}$$

The harmony memory (HM) is then updated by replacing the worst candidates by new by evaluating the fitness criteria given in Eq. (21):

$$x_{old} = x_{new} \quad \text{if} \quad f(x_{new}) > f(x_{old}) \tag{21}$$

where *f* denotes the fitness or objective function. The pseudo code of CHMS is shown above.

4.3. Chaotic Particle Swarm Optimization (CPSO) algorithm

Kennedy et al. developed Particle Swarm Optimization algorithm for optimization non-linear continuous functions [61] mimicking flocking of certain bird species. In Chaotic PSO the initial population is randomized by exploiting the ergodic behavior of chaotic sequences. The algorithm proceeds to global optima minimizing its chance for getting stuck in local optimum points by global exploration and local exploitation strategy [62]. Each agent is initialized randomly in the solution space following Eq. (22).

$$x_{j,k} = x_k^{\min} + chaos(0,1)\left(x_k^{\max} - x_k^{\min}\right) \tag{22}$$

where $x_{j,k}$ points to the current location of the particle in which *j* varies from 1 to N_p where N_p represents the total count of particles in a swarm and *k* spans from 1 to maximum dimensionality of the defined problem denoted as *dim*. Chaotic sequences are exploited for velocity updation as given in Eq. (23).

$$\vec{v}(t+1) = \gamma\vec{v}(t) + \Gamma_1 chaos_1(0,1)(\vec{p_b}(t) - \vec{x}(t)) + \Gamma_2 chaos_2(0,1)(\vec{g_b}(t) - \vec{x}(t)) \tag{23}$$

where $\vec{p_b}(t)$ and $\vec{g_b}(t)$ denotes the particle best and global best respectively, γ, Γ_1 and Γ_2 serves as weighing factors.

Algorithm 3. Chaotic particle swarm optimization algorithm.

```
1 Initialize population $x_{i,j}$ using Chaotic maps and set up
    the parameters via eqn. (22) ;
2 repeat
3     Calculate fitness value of each particle;
4     Update particle bests $p_b(t)$;
5     Calculate global best $g_b(t)$;
6     Calculate the updated velocity of each particle
        $\vec{v}(t+1)$ via eqn. (23) ;
7     Update particle postion $\vec{x}(t+1)$ via eqn. (24) ;
8 until iter<Maxiter;
9 Find the optimum solutions;
```

Table 2
Performance comparison of chaotic maps.

Algorithm	PSO				DPSO			
Index	Time	Std	Iter	Mean	Time	Std	Iter	Mean
I.1	6.497	0.1982	25	368.9954	9.7721	0.2069	17	368.0099
		0.2549	28	381.7623		0.4874	18	382.7744
		0.7973	27	345.541		0.3011	21	346.5443
I.2	5.5198	0.861	29	369.0093	9.7645	0.1489	19	368.0098
		0.5187	27	381.7742		0.0593	16	382.7743
		0.6972	25	345.5438		0.1316	21	345.5442
I.3	5.614	0.408	28	367.9618	8.7885	0.7436	19	367.9879
		0.4676	29	381.7295		0.2963	17	381.7606
		0.2012	27	344.1245		0.4597	21	345.5442
I.4	5.5036	0.9002	25	367.0082	9.0856	0.1913	15	369.0099
		0.9543	28	380.7743		0.0576	14	382.7744
		0.2413	29	345.5443		0.0456	18	345.5443
I.5	**5.2005**	**0.5999**	**24**	**369.7421**	**8.6513**	**0.0273**	**13**	**369.8382**
		0.2831	**21**	**382.9743**		**0.0463**	**14**	**383.1151**
		0.4946	**20**	**346.9443**		**0.0512**	**18**	**346.9824**
I.6	5.6513	0.1653	26	368.0082	8.9292	0.1381	19	369.0099
		0.9403	23	381.7723		0.169	17	382.7744
		0.4455	22	345.5398		0.282	19	345.5443
I.7	5.8412	0.8108	24	368.004	9.6987	0.1985	14	369.0098
		0.7082	23	381.7653		0.1762	16	382.7744
		0.0576	24	345.4969		0.7412	18	345.5443
I.8	5.7688	0.6517	25	368.0098	9.4054	0.0447	19	369.0096
		0.3584	26	381.7739		0.1092	20	382.7744
		0.6671	22	346.5441		0.1384	19	346.544
I.9	5.6778	0.7704	26	369.0098	9.5777	0.0856	18	369.0099
		0.2598	23	382.0743		0.0648	16	382.7744
		0.4761	21	346.5443		0.0384	17	346.5443
I.10	5.8599	0.2971	27	368.0099	9.3315	0.0475	17	369.0099
		0.1148	23	381.7743		0.0399	18	382.7743
		0.7242	20	345.5443		0.0675	22	346.5443

The location of each particle is then updated using Eq. (24).

$$\vec{x}(t+1)=\vec{x}(t)+\vec{v}(t+1) \tag{24}$$

where $\vec{v}(t+1)$ denotes the updated velocity vector and $\vec{x}(t+1)$ is the new location of the agent particle. Each iteration brings about an improvement in fitness value and converges to the global optimum on fulfilling the termination condition. The pseudo code given above briefly depicts the flow of CPSO algorithm [1,14].

4.4. Chaotic Differential Evolution (CDE) algorithm

It is a global optimization algorithm put forward by Storn et al. [12]. In chaotic DE, the population with size N_p is initialized using chaotic functions wherein each member of the population is a n-dimensional parameter vectors in the form $\vec{X}_i(t)=[x_{i,1},x_{i,2},\ldots,x_{i,n}]$, $i=1,2,\ldots,N_p$. The algorithm proceeds evolving different generations of individuals depending on the principle of survival of fittest [63]. A donor vector $\vec{Z}_i(t)$ is created in each generation to alter the current candidates in the population space. Different variants of DE evolved depending on the method used for generating this donor vector [64,65]. In DE/rand/1 strategy, donor vector is framed by combining three vectors chosen from the current population; say v_1, v_2 and v_3 as given in Eq. (25)

$$\vec{Z}_{i,j}(t)=\vec{X}_{v_1,j}(t)+F\cdot(\vec{X}_{v_2,j}(t)-\vec{X}_{v_3,j}(t)) \tag{25}$$

Exploitation potential of the population is increased by binomial cross over on each of the n variables by choosing a random number within the range 0 and 1 by means of normalized chaotic operators limited by the c_r value [66]. This creates a trail vector $\vec{T}_i(t)$ for each $\vec{X}_i(t)$ as defined in Eq. (26).

$$\vec{T}_{i,j}(t)=\begin{cases}\vec{Z}_{i,j}(t) & \text{if } \ chaos_j(0,1)\le c_r \lfloor j=rn(i)\\ \vec{X}_{i,j}(t) & \text{otherwise}\end{cases} \tag{26}$$

where $j=1,2,\ldots,n$, $chaos_j(0,1)$ is the jth iterated value of the chaotic operator and $rn(i)\in[1,2,\ldots,n]$ is chosen to ensure that $\vec{T}_i(t)$ has atleast one of the element is from $\vec{X}_i(t)$. Last phase of the algorithm performs a selection between the trail vector [67] and the target vector and ensures the survival of the fittest member and proceeds to the next generation [68,69] governed by Eq. (27).

$$\vec{X}_i(t+1)=\begin{cases}\vec{T}_i(t) iff(\vec{T}_i(t))>f(\vec{X}_i(t))\\ \vec{X}_i(t) iff(\vec{T}_i(t))\le f(\vec{X}_i(t))\end{cases} \tag{27}$$

Pseudo code of CDE algorithm is depicted below.

Algorithm 4. Chaotic Differential Evolution algorithm.

```
1  Initialize population using chaotic maps;
2  Initialize weighting factor $F$, crossover probability $c_r$
     and maximum number of generations required;
3  Evaluate the fitness value for all elements in the
     population;
4  for (iter < termination criteria or Maxgen) do
5      for all i & j do
6          Generate random vectors $v_1, v_2 \& v_3$ ;
7          Mutation: Create donor vectors using
             $v_1, v_2$ & $v_3$ via eqn. (25) ;
8          Crossover: Form trail vectors via eqn. (26);
9          Selection : Update the candidate solutions via
             eqn. (27);
10     end
11 end
12 Find the optimum solutions;
```

5. Proposed Chaotic Darwinian Particle Swarm Optimization (CDPSO) algorithm

Tillett et al. modified PSO algorithm incorporating Darwinian principle of natural selection for finding the fittest candidate which improved its efficiency in different optimization platforms [70]. The use of multiple number of swarms in Darwinian Particle Swarm Optimization (DPSO) algorithm helped in improving its search capacity to a better extend [71,72]. Still the algorithm faces the drawback of getting trapped in local optimum points reducing its potential to trace the global best. In order to tackle this premature convergence problem, a new variant of DPSO algorithm is proposed. The proposed CDPSO algorithm ensures to be an effective strategy for improving the convergence rate and image segmentation efficiency, combining the favorable traits of chaotic maps and Darwinian principle.

In the proposed CDPSO algorithm, this initial solution space is formed using logistic chaotic function mathematically defined as in Eq. (28).

$$x_{(i,j)}=chaos(0,1)*(x_{\max}(j)-x_{\min}(j))+x_{\min}(j) \tag{28}$$

where $i\in 1,2,\ldots,N$ and $j\in 1,2,\ldots,N_s$. The performance of swarms are validated in each iterative steps and the best among them is allowed to spawn new to increase local solution exploitation as given in Eq. (29).

$$\left[\vec{x_{g_b}}(t),\vec{x_{p_b}}(t)\right]\longleftarrow f(\vec{x}(t)) \tag{29}$$

where $f(\cdot)$ denotes the fitness function defined to compute global and particle bests $\left[\vec{x_{g_b}}(t),\vec{x_{p_b}}(t)\right]$ among each swarm. This resembles natural selection process of favoring the fittest solution to survive and procreate. Drift velocity of each agent is calculated as given in Eq. (30) which depends on three weighing factors $\chi,\xi_1\,\&\,\xi_2$.

$$\vec{v}(t+1)=\chi\vec{v}(t)+\xi_1 chaos_1(0,1)(\vec{x_{p_b}}(t)-\vec{x}(t))+\xi_2 chaos_2(0,1)(\vec{x_{g_b}}(t)-\vec{x}(t)) \tag{30}$$

This computed drift velocity is used to update the current location of each agent particle following Eq. (31).

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1) \tag{31}$$

where $\vec{v}(t)$ and $\vec{x}(t)$ denote the velocity vector and position vector respectively. The pseudo code given below explains the flow of CDPSO algorithm for finding global optima in concise.

Rules governing spawning and deletion of particles and swarms are furnished below.

- *Spawning new swarms*: when it fulfills the condition that there are no particles deleted from a particular swarm (i.e $n_{kill} = 0$) and the upper bound of the total number of swarms that can coexist in the scenario is not met, a new swarm will be spawned.
- *Spawning new particles*: when a better fitness value is achieved by a particle, a new particle will be spawned.
- *Deleting swarm*: when the number of particles present in a swarm falls beyond its defined lower bound that particular swarm will be deleted.
- *Deleting a particle*: when search counter value exceeds the maximum counter value which is set in prior to processing, the particle will be deleted. On spawning a new swarm search counter C_s will be initialized to zero. It gets incremented by 1 whenever it proceeds without improving the particle's fitness value.

Algorithm 5. Proposed Chaotic Darwinian Particle Swarm Optimization algorithm.

```
1  Initialize particle positions $x_{i,j}$ via eqn (28) and set other
     parameters;
2  Calculate fitness values of all particles;
3  while (t < Maxiter) or (termination condition) do
4      Compute particle best and global best via eqn. (29);
5      Calculate velocity vector via eqn. (30) ;
6      Update particle positions via eqn. (31);
7      Evaluate the fitness values;
8      if $fitness_{new} > fitness_{old}$ then
9          Spawn new particle;
10     else
11         Update search counter by one;
12         if $C_s = C_s^{max}$ then
13             Delete a particle;
14             Increment $n_{kill}$ by one;
15         end
16     end
17     if $n_{kill} = 0$ and $N_s < N_s^{max}$ then
18         Spawn new swarm;
19     end
20 end
21 Find the optimum solutions;
```

Immediately after deleting a particle, the search counter value will be re-initialized to

$$C_s(n_{kill}) = C_s^{\max}\left(\frac{n_{kill}}{n_{kill}+1}\right) \tag{32}$$

where n_{kill} denotes the count of deleted particles without improving fitness over time.

6. Results and discussions

We have investigated the performance of five different chaotic optimization algorithms presented in Sections 4 and 5 on different satellite images chosen from *Pl êiades* satellite imaginary. All those multi-spectral images have a resolution of 50 cm acquired in 4-bands. Test results of two among them (Fig. 1) have been furnished in this section, validated by means of quantitative and qualitative analysis. Figs. 2–5 presents the segmented results of that satellite image using five different chaotic optimization algorithms which includes CCS [Section 4.1], CHS [Section 4.2], CPSO [Section 4.3], CDE [Section 4.4] and proposed CDPSO [Section 5]. Logistic chaotic function with initial value $x_0 = 0.7$ was experimentally found to be best suited for satellite image segmentation application [Section 3.1]. Both the images were segmented for three different levels of thresholding utilizing minimum cross entropy measure [Section 2.1] and Tsallis entropy measure [Section 2.2] as objective functions. Similar studies were carried out on two natural images to examine the robustness of the algorithm. The results are furnished in Tables 3-6 and Figs. 2–5 in the supplementary data. Quantitative performance comparison of CDPSO and DPSO algorithms are presented as Table 1 and 2 in the supplementary data. Table 3 gives the values of different parameters used by each of the chaotic optimization algorithms in the study. Values of all these parameters were determined based on practical experiments and empirical evidences. Quantitative analysis of the segmented results are presented in Tables 4–7. Test results were simulated using MATLAB R2015a on an Intel® Core^{TM} i7 PC with 2 GB RAM and 2.93 GHz CPU.

6.1. Quantitative analysis

The performance evaluation of different chaotic optimization algorithms presented in the paper is carried out by comparing quality metric values like Mean, Standard Deviation (STD), Peak Signal to Noise ratio (PSNR), Mean Square Error (MSE), Structural Similarity Index (SSIM) and CPU running time. All five algorithms were tested for three levels of thresholding optimizing two different objective functions. The corresponding optimum threshold values obtained for multilevel segmentation using different thresholding levels (n) are also presented in Tables 4–7.

PSNR and MSE value suggests the reconstruction accuracy of the original image from the segmented one whereas SSIM rates the segmented image quality based on the original image being processed. Eq. (33) formulates the mathematical description for computing PSNR and MSE value.

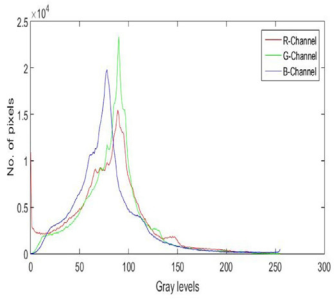
$$PSNR(dB) = 10\log_{10}\left(\frac{(L_{\max}-1)^2}{MSE}\right) \quad \text{where}$$

$$MSE = \frac{\sum_{m=1}^{M_1}\sum_{n=1}^{M_2}\left[I_{m,n} - I_{m,n}^t\right]}{M_1 \times M_2} \tag{33}$$

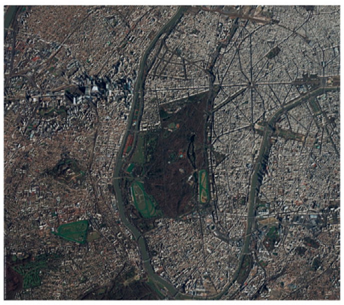
where L_{max} represents the maximum intensity value of the image, $M_1 \times M_2$ denotes the size of the original image. The original and thresholded images are represented by I and I^t respectively [73,74].

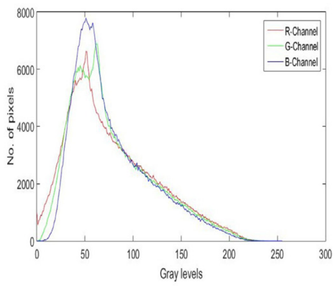
Structure Similarity Index (SSIM) is measured taking into account of the similarity in the high frequency content (edge information) of segmented and original image [75]. Higher value of

(a)



(b)


(c)



(d)

Fig. 1. Test images (a) Satimage1: City of Space Toulouse, France (900X923) (c) Satimage2: Paris, France (900X575) (http://www.satpalda.com/gallery/) and their respective histograms (b), (d).

Table 3
Parameters used in each optimization algorithm.

Algorithm	Parameters	Value
CCS [Section 4.1]	No. of nests (N), No. of iterations (*Maxiter*)	50, 150
	Alien egg discovery rate (p_a), β	0.25, 1.5
	Lower &upper limits for solution vector (x_i^l, x_i^h)	0, 255
CHS [Section 4.2]	No. of solution vector (H_{size}), Improvisation Count (IC)	50, 300
	Lower and upper bounds of harmony ($Lb(i)$, $Ub(i)$)	1256
	HMCR, *PAR*, *BW*	0.75, 0.5, 0.5
CPSO [Section 4.3]	Population size (N_p), No. of iterations (*Maxiter*),	180, 150
	Inertial factor (γ), Weight factors (Γ_1, Γ_2)	0.8 [1.2, 0.8]
	Bounds of particle position (x^{min}, x^{max})	0, 255
	Velocity of a particles ($v_{\min}$, $v_{\max}$)	0, 5
CDE [Section 4.4]	Population size (N_p), Max no. of generations (*Maxgen*)	150, 150, 150
	Min. &max range of candidate solutions (x_{min}, x_{max})	0, 255
	Weighting factor (F), Crossover probability (c_r)	0.5, 0.9
CDPSO [Section 5]	Initial population of a swarm (N), No. of swarms (N_s)	30, 4
	No. of iterations (*Maxiter*), Stagnancy count (C_s)	150, 10
	Min. &max. population in a swarm (S_{min}, S_{max})	10, 50
	Min. &max. no. of swarms ($N_s^{\min}$, $N_s^{\max}$)	2, 6
	Stagnancy counter limit ($C_s^{\max}$)	10
	Weighting factors (χ, ξ_1 & ξ_2)	0.8, 1.2 & 0.8
	Bounds of particle position (x_{min}, x_{max})	[0, 255]
	Bounds of particle velocity ($v_{\min}$, $v_{\max}$)	[0, 5]

SSIM index indicates higher amount of edge information in the segmented image [76,73]. Mathematically it can be represented as in Eq. (34)

$$SSIM(I, I^t) = \frac{(2\mu_I\mu_{I^t} + \psi_1)(2\sigma_{II^t} + \psi_2)}{(\mu_I^2 + \mu_{I^t}^2 + \psi_1)(\sigma_I^2 + \sigma_{I^t}^2 + \psi_2)} \tag{34}$$

where ψ_1 and ψ_2 are constants, μ, σ and σ^2 represents mean, standard deviation and variance of the defined image.

SSIM was proved to be a better quality measure metric, but the computational complexity is more since the formula involves the use of one number per pixel. The computational complexity of each algorithm is measured by calculating the total CPU time taken for processing the image. The stability of the algorithm is judged by finding the standard deviation (STD) of the fitness value.

Standard deviation (STD) measure can be mathematically formulated as:

$$\sigma = \frac{1}{iter_{\max}}\left(\sum_{i=1}^{iter_{\max}} (x_i - \mu)^2\right)^{\frac{1}{2}}$$

$$\text{where mean } \mu = \frac{\sum_{j=1}^{iter_{\max}} x_i}{iter_{\max}};$$

$$x_i \text{ denoting the fitness value at the } i^{\text{th}} \text{ iteration} \tag{35}$$

Tables 4–7 furnishes the above mentioned metric values of all algorithms used for comparison.

6.2. Qualitative analysis

Subjective analysis can be done by examining the segmented results of the given image using all five chaotic optimization algorithms incorporating either of the two objective functions presented in the study and carried out for three different thresholding levels. The segmented results are shown in Figs. 2–5.

6.3. Test 1: Minimizing cross-entropy measure between classes

Segmentation of satellite images with minimum cross entropy as objective function resulted in the segmented outputs shown in Figs. 2 and 4 and performance metrics as in Tables 4 and 6. CDPSO was proved to have the best performance among the five chaotic algorithms on comparing all the performance evaluation metrics. Use of chaotic sequences instead of random numbers for initializing the population for each of the five algorithms resulted in the fast convergence of all of them towards the optimum solution vector.

Ranking these five algorithms according to the PSNR and SSIM values resulted in ordering them in the sequence *CDPSO* > *CHS* >

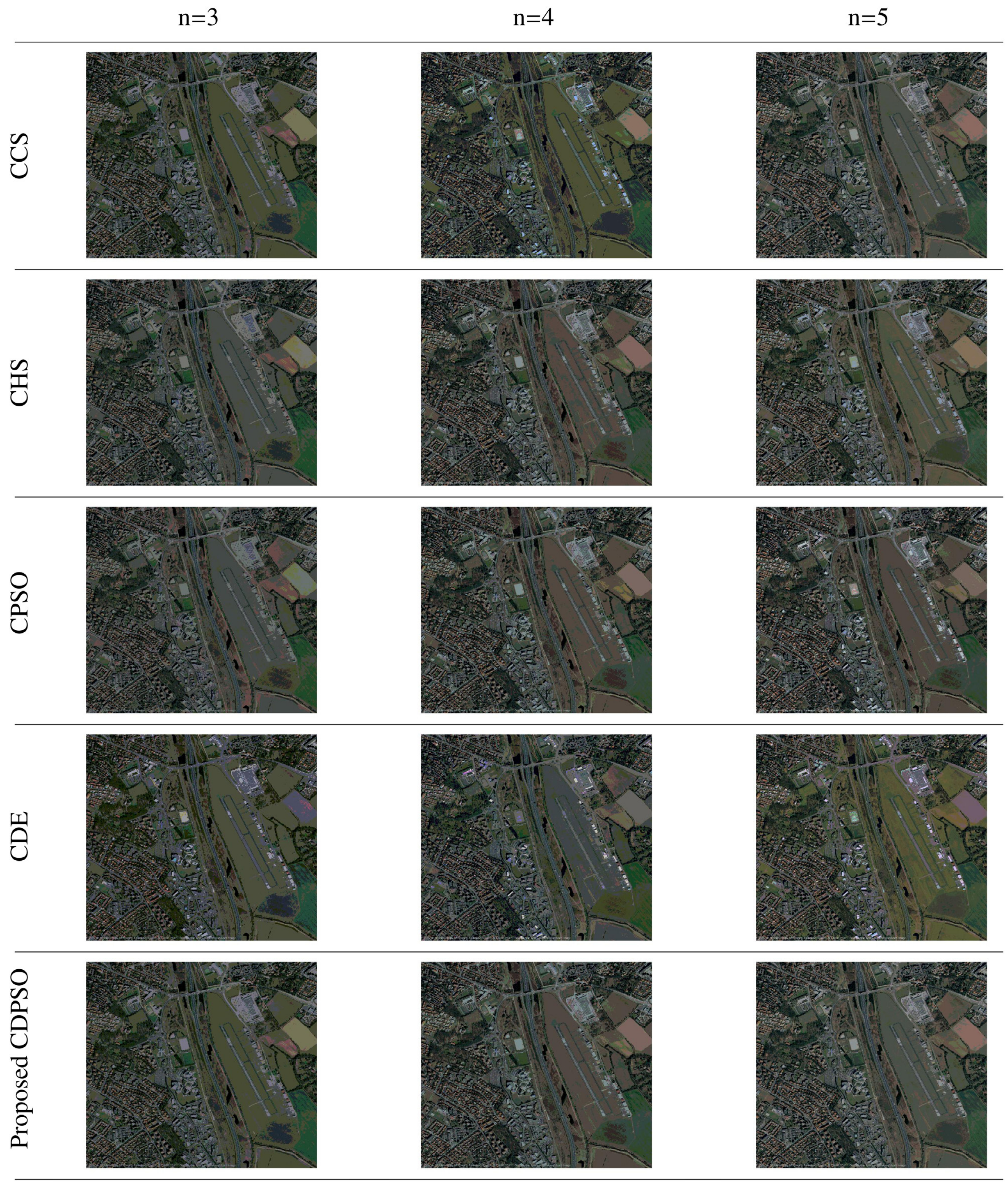


Fig. 2. Results of segmenting Satimage1 using five different chaotic optimization algorithms for three different levels of thresholding minimizing cross entropy measure.

$CCS > CPSO > CDE$. Taking into account of the computational time, we can line up them as $CPSO < CHS < CDPSO < CCS < CDE$ according the increasing order of execution time. On examining the stability capability of those algorithms which reflect the number of iterations required to attain its optimum fitness value, the ranking will be as $CDE \sim CCS > CDPSO > CPSO > CHS$. Although CDE ranks to be the first among them, it suffers the problem of converging to a local optimum point. Even the comparison of all other quantifying parameters rates CDE to be inferior among others.

6.4. *Test 2: Maximizing Tsallis entropy measure between classes*

Tables 5 and 7 and Figs. 3–5 highlights the performance of CDPSO algorithm maximizing Tsallis entropy for finding threshold values to segment satellite images. Comparison of PSNR and MSE values grade the five algorithms in the order $CDPSO > CPSO > CCS > CHS > CDE$. Mean of fitness value seems to be high on comparison with other five algorithms, which clearly suggests the efficiency of CDPSO to converge to local maxima. Stability

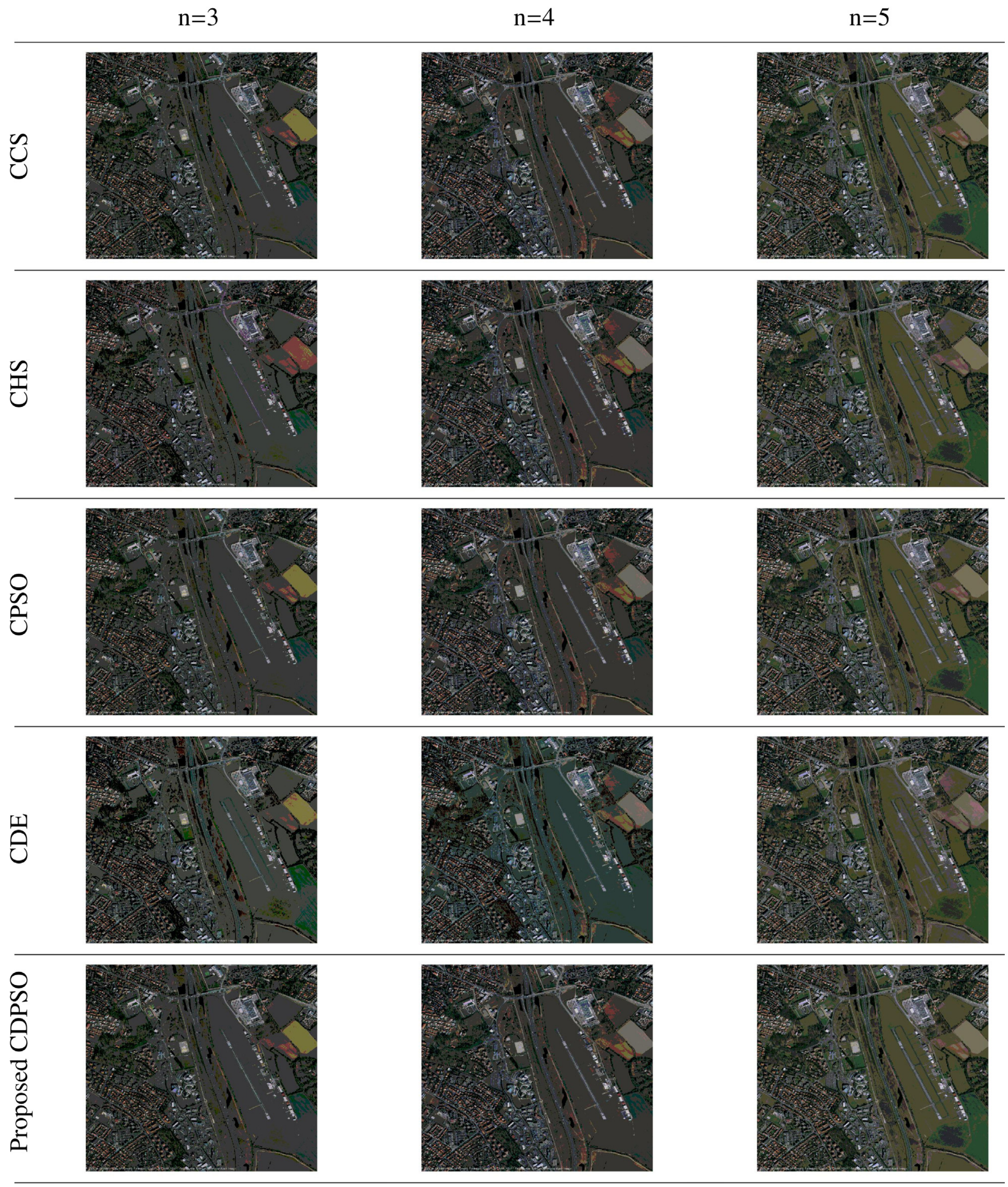


Fig. 3. Results of segmenting Satimage1 using five different chaotic optimization algorithms for three different levels of thresholding maximizing Tsallis entropy measure.

analysis can be justified by the measurement of standard deviation of the final fitness value from its mean which ranks the five algorithms in the order *CDPSO > CDE > CCS > CPSO > CHS*. Sequencing these algorithms in the increasing order of computational time gives *CHS < CCS < CPSO < CDPSO < CDE*.

By setting a decent trade-off between all the performance evaluation parameters, we can conclude that the proposed CDPSO algorithm clearly emerges as the best algorithm for satellite image segmentation application.

6.5. *Convergence rate analysis*

Figs. 6–9 depicts the convergence characteristics obtained by 3-level thresholding of the given satellite images using two different objective functions for various optimization algorithms. The characteristic curves are plotted by mapping the fitness values against the number of iterations in each case.

Figs. 6 and 8 show the fitness plot using minimum cross entropy as objective function in PSO, CPSO, DPSO and CDPSO algorithms.

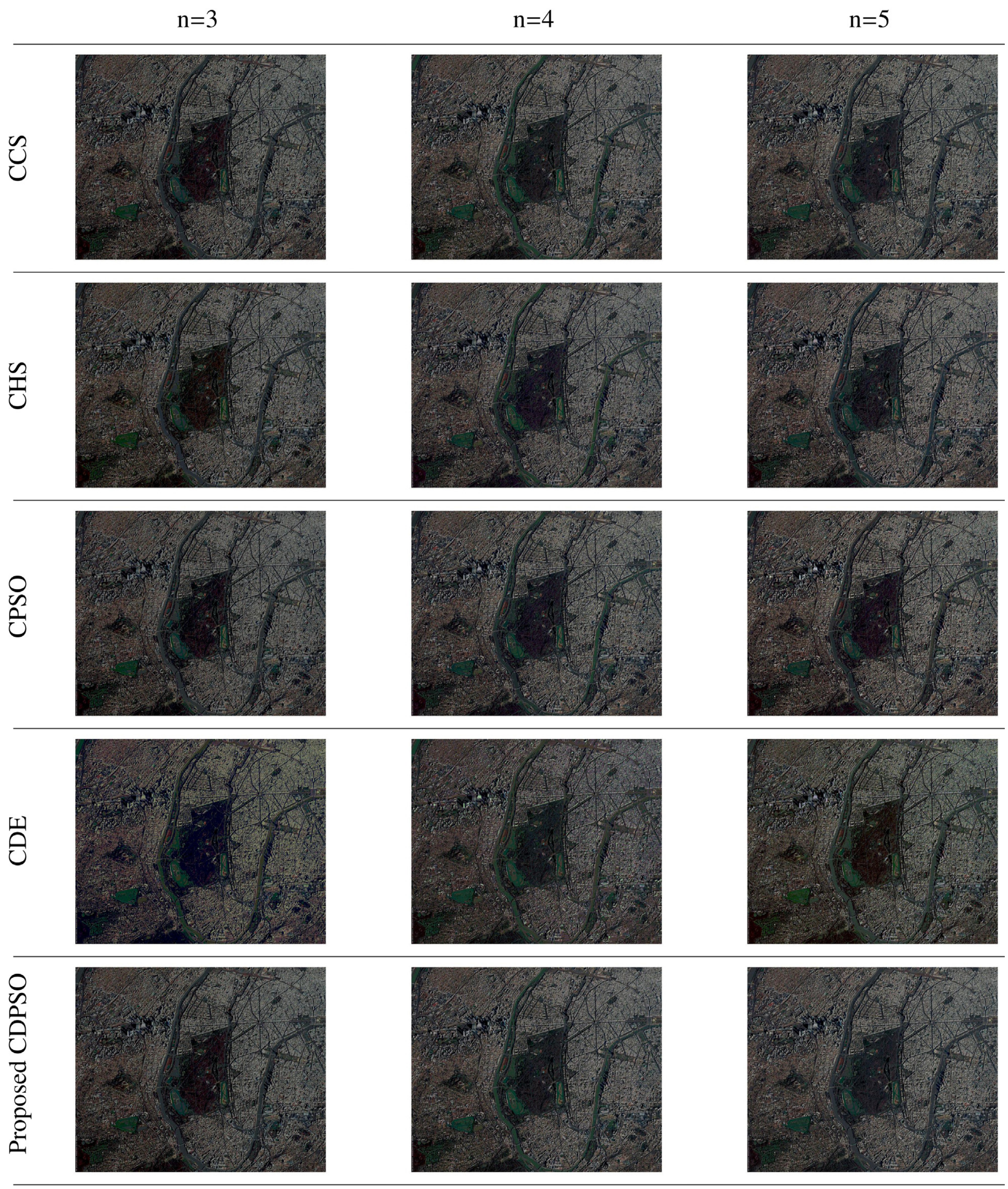


Fig. 4. Results of segmenting Satimage2 using five different chaotic optimization algorithms for three different levels of thresholding minimizing cross entropy measure.

Proposed CDPSO algorithm emerges to be the most stable by attaining the maximum fitness value in less number of iterations. The plot clearly reveals that the proposed algorithm avoids the chance of premature convergence by escaping from local clogs on its way to attains global optima. Instead of minimizing the cross entropy value we have performed our experiments by maximizing the negative of the objective function given in Eq. (8). Fitness plots using Tsallis entropy as objective function furnished in Figs. 7 and 9 which also shows the similar trends in performance. Fitness plots reveals that in comparison with Tsallis entropy method, the scheme using minimum cross entropy as objective function converges to the global optima in less number of iterations uniformly for R, G and B channels. Hence the convergence rate analysis of these satellite images using the four different algorithms given above, establishes the efficiency of the proposed CDPSO algorithm in converging to the global optimum fitness values in less number of iterations.

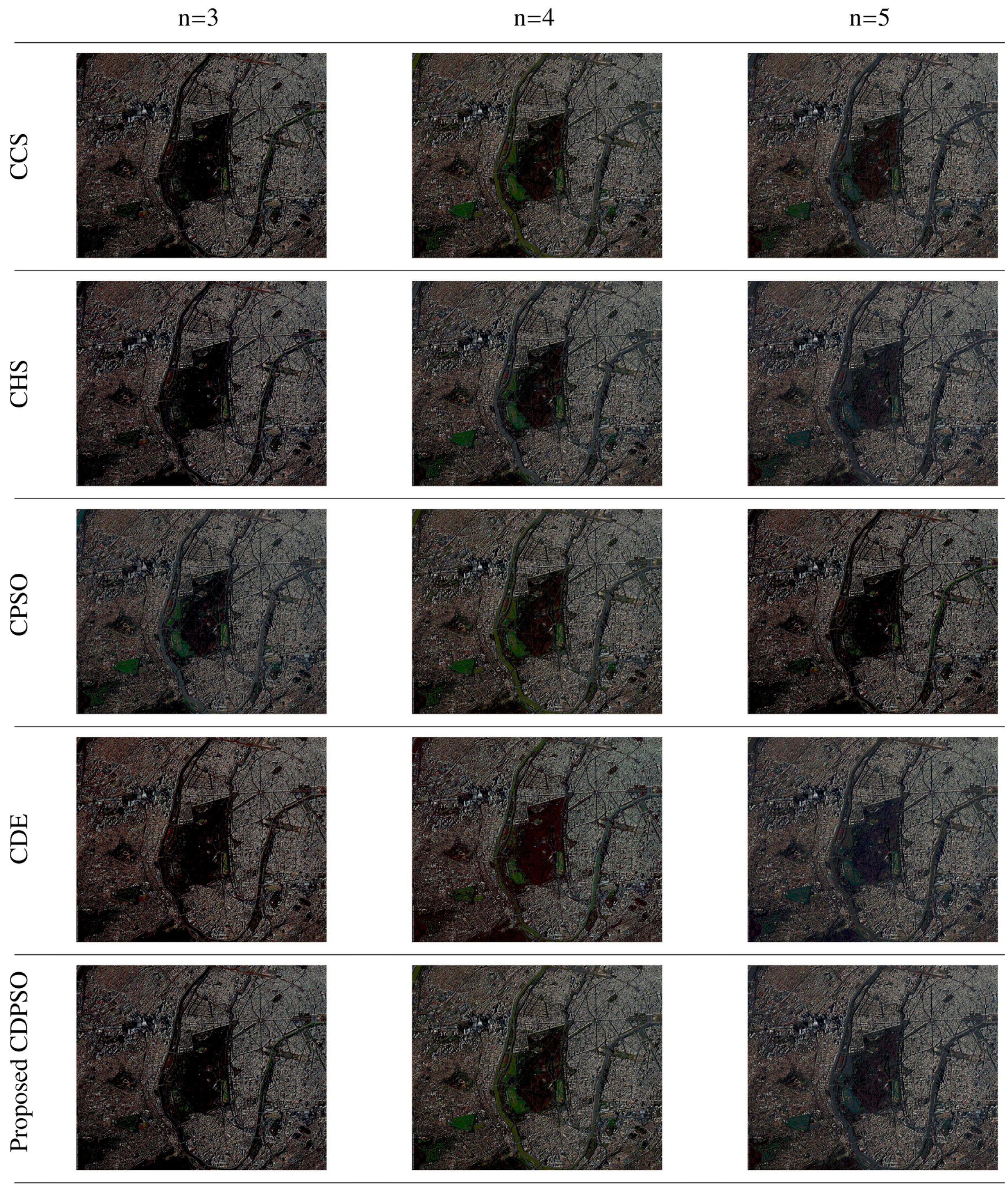


Fig. 5. Results of segmenting Satimage2 using five different chaotic optimization algorithms for three different levels of thresholding maximizing Tsallis entropy measure.

7. Conclusion

Most of the nature inspired algorithms faces the problem of getting trapped in local optimum points other than converging to its global optima. Random sequences used in its initialization phase and along its run have high impact on this premature convergence. In this paper a chaotic DPSO scheme have been proposed towards improving the convergence rate and the segmentation quality of satellite images. Among the 10 chaotic maps presented in this paper, logistic chaotic map proved to be the better choice for segmentation application. The proposed algorithm has been compared with some existing chaotic variants of optimization algorithms like Cuckoo Search (CS), Harmony Search (HS), Differential Evolution(DE) and Particle Swarm Optimization (PSO) exploiting the chosen optimal chaotic map for finding optimum threshold values for satellite image segmentation. The performance of each algo-

Table 4
Comparison of standard deviation (STD), mean, PSNR, MSE, SSIM and running time of different chaotic optimization algorithms for three different thresholding levels (*n*) using minimum cross-entropy for segmenting Satimage1. (Chl: Channel; Iter: No. of Iterations.)

Algorithm	*n*	Chl	STD	Iter	Mean	PSNR	MSE	SSIM	Time (s)	Thresholds
CCS [Section 4.1]	3	R	0.0177	6	368.1165	19.9888	651.9237	0.7355	8.3167	30 72 118
		G	0.0049	8	381.6461					42 76 116
		B	0.0059	5	346.0129					51 86 133
	4	R	0.0147	10	369.5825	21.8233	427.3151	0.8088	13.2746	19 51 82 124
		G	0.0014	7	382.913					39 72 101 142
		B	0.0099	9	346.8178					39 66 94 142
	5	R	0.0126	11	369.8385	23.449	293.8859	0.8525	18.828	17 47 75 101 140
		G	0.0149	12	383.1152					32 58 81 106 147
		B	0.0154	8	346.982					33 54 73 98 145
CHS [Section 4.2]	3	R	0.0378	39	369.096	20.2565	612.9571	0.7254	4.6814	29 71 116
		G	0.0111	42	382.6408					42 76 116
		B	0.01	46	346.4018					43 72 112
	4	R	0.0098	41	369.5131	21.9714	412.9881	0.8038	5.8188	25 60 88 130
		G	0.0029	40	382.9094					39 72 101 142
		B	0.0298	44	346.7916					40 66 94 143
	5	R	0.0396	38	369.7585	23.4083	296.6571	0.859	6.7292	22 53 77 94 132
		G	0.0455	37	383.0444					37 63 83 111 150
		B	0.0153	43	346.9358					38 65 89 120 171
CPSO [Section 4.3]	3	R	0.1705	18	369.0277	20.2406	615.2041	0.7249	3.2539	30 68 104
		G	0.1029	17	382.6422					42 74 115
		B	0.9789	19	346.4052					42 72 111
	4	R	0.254	21	369.5769	21.8093	428.6978	0.8083	4.2843	18 52 83 127
		G	0.1425	24	382.909					39 73 104 144
		B	0.0349	23	346.8107					39 66 97 147
	5	R	0.2766	24	369.741	22.7336	346.5107	0.8291	5.4428	18 50 79 115 176
		G	0.148	21	382.9811					41 73 99 138 196
		B	0.0478	20	346.9083					39 67 94 133 200
CDE [Section 4.4]	3	R	0.2316	6	369.1013	19.372	751.4101	0.6928	10.3538	34 76 151
		G	0.1664	4	382.6245					41 79 152
		B	0.1255	7	346.4					59 100 165
	4	R	0.1407	9	369.5486	20.8711	532.0676	0.7635	11.6485	18 62 101 199
		G	0.1276	4	382.8507					51 70 100 199
		B	0.0962	8	346.7509					36 73 96 183
	5	R	0.0672	8	369.6626	22.7403	345.9825	0.8186	11.977	15 57 90 123 246
		G	0.0944	8	383.0663					23 44 85 147 248
		B	0.0678	9	346.8843					27 52 97 165 241
Proposed CDPSO [Section 5]	3	R	0.0321	11	369.1165	**21.8233**	**427.3151**	**0.8088**	5.0745	30 72 118
		G	0.0035	13	382.6461					42 76 116
		B	0.0048	14	346.4129					51 86 133
	4	R	0.0172	17	369.5825	**23.3978**	**297.371**	**0.8534**	6.0311	19 51 82 124
		G	0.0009	12	382.913					39 72 101 142
		B	0.0134	10	346.8178					39 66 94 142
	5	R	0.0419	13	369.8382	**24.5788**	**226.5697**	**0.8905**	8.8942	15 45 73 99 138
		G	0.0497	14	383.1151					32 58 81 105 145
		B	0.0291	18	346.982					32 53 72 97 144

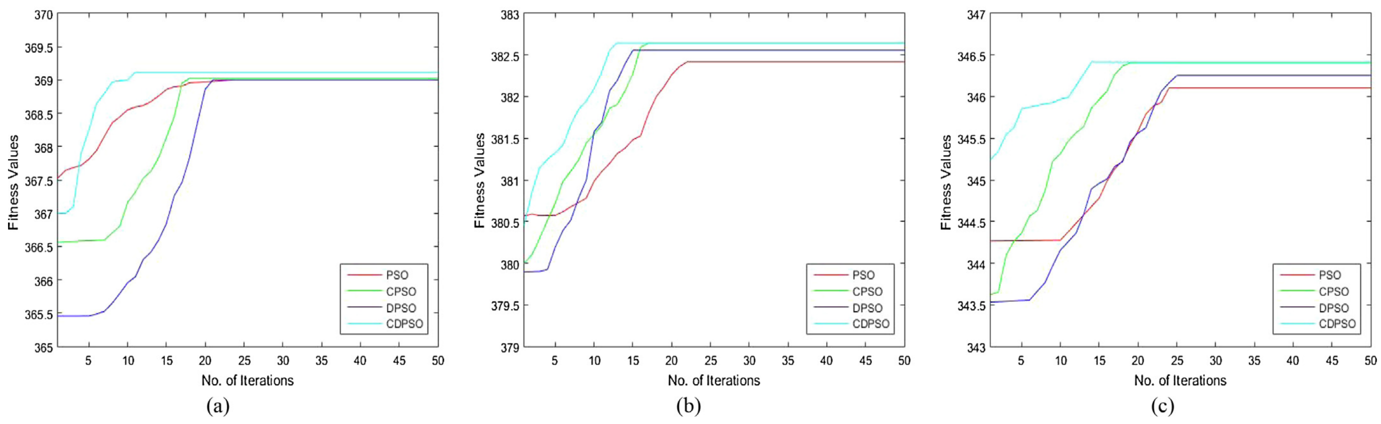


Fig. 6. Convergence characteristics of segmenting Satimage1 using Minimum Cross Entropy method with 3 thresholding levels using *PSO*, *CPSO*, *DPSO* and *CDPSO*$_{(proposed)}$ algorithms (a) R-channel, (b) G-channel, (c) B-channel.

Table 5
Comparison of standard deviation (STD), mean, PSNR, MSE, SSIM and running time of different chaotic optimization algorithms for three different thresholding levels (*n*) using Tsallis entropy measure for segmenting Satimage1. (Chl: Channel; Iter: No. of Iterations.)

Algorithm	*n*	Chl	STD	Iter	Mean	PSNR	MSE	SSIM	Time	Thresholds
CCS [Section 4.1]	3	R	0.8895	14	1.5142	17.8487	1067.1	0.5839	8.5078	58 122 184
		G	0.4576	18	1.3754					57 117 175
		B	0.6142	19	1.5348					58 124 187
	4	R	0.7854	24	1.1186	18.6908	879.0309	0.6671	9.4389	55 108 153 201
		G	0.5692	27	1.0027					54 107 153 200
		B	0.648	28	1.1392					51 99 149 201
	5	R	0.8752	22	7.0194	21.0408	511.6823	0.7634	10.1864	40 78 118 159 205
		G	0.6565	24	6.1635					40 76 114 156 200
		B	0.8696	20	7.4816					47 90 130 171 213
CHS [Section 4.2]	3	R	0.5869	46	1.5134	17.6962	1105.2	0.5875	4.1052	58 121 183
		G	0.8694	47	1.3711					62 133 186
		B	0.4578	43	1.5321					59 125 190
	4	R	1.0236	51	1.1184	18.674	882.4219	0.6581	5.1523	55 108 153 201
		G	0.8569	57	1.0009					52 105 149 197
		B	0.4562	59	1.1387					51 99 149 201
	5	R	1.0869	48	6.9897	21.061	509.3079	0.7582	7.1924	38 76 117 158 204
		G	0.8846	41	6.151					39 74 112 155 201
		B	1.0023	40	7.1369					41 84 119 159 205
CPSO [Section 4.3]	3	R	0.7542	25	1.5142	17.8508	1066.6	0.5794	9.2534	57 121 183
		G	0.8942	28	1.3754					56 116 174
		B	1.0258	27	1.5348					57 123 186
	4	R	0.4692	29	1.1186	18.685	880.2079	0.6695	10.8809	54 107 152 200
		G	0.4136	34	1.0027					53 106 152 199
		B	0.7785	35	1.1392					50 98 148 200
	5	R	1.0456	27	7.0168	21.0396	511.8246	0.7624	12.8176	39 77 118 158 204
		G	0.882	28	6.1635					39 75 113 155 199
		B	0.9456	29	7.4013					44 87 124 163 207
CDE [Section 4.4]	3	R	0.1286	17	1.5137	17.7741	1085.6	0.5971	34.514	65 127 182
		G	0.3646	19	1.3739					67 121 177
		B	0.648	14	1.5335					62 121 198
	4	R	0.9958	16	1.1013	18.3522	950.2937	0.5711	39.4196	43 108 150 208
		G	0.7584	18	0.9876					61 105 145 206
		B	0.8241	20	1.1292					57 99 142 199
	5	R	1.5365	24	6.9431	21.0792	507.1811	0.7632	43.2449	42 75 107 148 209
		G	0.8937	19	6.1338					39 72 104 143 202
		B	0.3525	14	7.3636					38 81 114 158 211
Proposed CDPSO [Section 5]	3	R	0.3618	19	1.514	**17.8721**	**1061.4**	**0.5991**	15.7829	57 119 180
		G	0.4658	17	1.3754					56 116 174
		B	0.898	18	1.5348					58 124 187
	4	R	0.1098	17	1.1186	**18.6815**	**880.8984**	**0.67**	16.198	55 108 153 201
		G	0.3245	19	1.0027					53 106 152 199
		B	0.7792	20	1.1392					50 98 148 200
	5	R	1.0187	21	7.0194	**21.4049**	**470.5342**	**0.7637**	17.9251	40 78 118 159 205
		G	0.6279	21	6.1635					40 76 114 156 200
		B	0.3661	14	7.4772					46 88 128 169 211

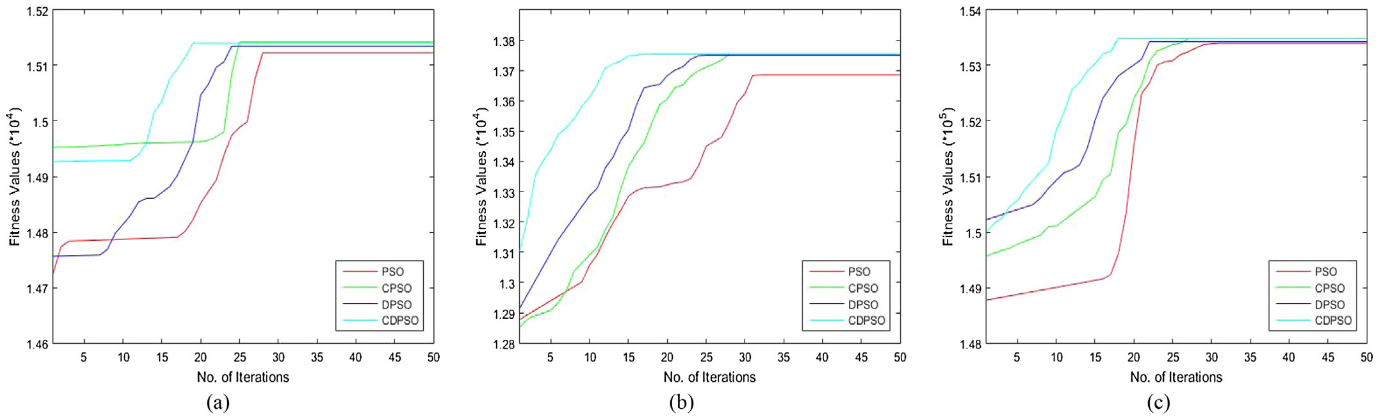


Fig. 7. Convergence characteristics of segmenting Satimage1 using Tsallis method with 3 thresholding levels using *PSO*, *CPSO*, *DPSO* and *CDPSO*$_{(proposed)}$ algorithms (a) R-channel, (b) G-channel, (c) B-channel.

Table 6
Comparison of standard deviation (STD), mean, PSNR, MSE, SSIM and running time of different chaotic optimization algorithms for three different thresholding levels (*n*) using minimum cross-entropy for segmenting Satimage2. (Chl: Channel; Iter: No. of Iterations.)

Algorithm	*n*	Chl	STD	Iter	Mean	PSNR	MSE	SSIM	Time (s)	Thresholds
CCS [Section 4.1]	3	R	0.0146	5	358.8885	19.3948	747.4838	0.8061	8.3118	37 72 122
		G	0.0086	10	357.5947					45 79 126
		B	0.0082	8	352.5414					49 79 124
	4	R	0.0105	11	359.4173	21.2895	483.1984	0.8674	13.0192	31 58 91 136
		G	0.0103	13	358.024					36 59 90 134
		B	0.0058	10	353.2263					44 67 97 138
	5	R	0.0315	9	359.6784	22.8556	336.9151	0.9077	18.96	24 45 69 101 143
		G	0.0271	7	358.2431					33 53 76 106 146
		B	0.025	7	353.3874					39 57 78 107 145
CHS [Section 4.2]	3	R	0.1291	11	358.8757	19.3219	760.1344	0.8016	2.4124	37 70 120
		G	0.1412	32	357.6397					46 79 126
		B	0.1524	45	352.8419					50 84 130
	4	R	0.1532	43	359.3089	21.1963	493.6842	0.8693	3.6937	30 57 84 132
		G	0.1397	41	357.8856					40 63 91 128
		B	0.1433	38	353.0772					40 68 96 133
	5	R	0.0431	39	359.6051	22.6562	352.747	0.8963	4.826	30 54 79 107 148
		G	0.0487	40	358.1604					37 59 81 115 162
		B	0.0306	43	353.3125					41 64 85 118 161
CPSO [Section 4.3]	3	R	0.744	12	358.9315	19.1634	788.3942	0.7954	2.178	41 72 114
		G	0.5704	14	357.6691					46 76 117
		B	0.9013	18	352.93					49 78 118
	4	R	0.814	19	359.4074	21.3216	479.6489	0.8698	3.2253	30 61 95 139
		G	0.6199	21	358.0088					40 64 98 142
		B	0.9383	22	353.2007					42 66 101 149
	5	R	0.8315	24	359.5456	21.8096	428.663	0.875	4.3503	29 55 88 126 181
		G	0.6219	19	358.0441					37 69 100 141 188
		B	0.9428	24	353.2378					44 67 98 141 206
CDE [Section 4.4]	3	R	0.2296	9	358.9851	18.8241	852.4484	0.7769	10.2652	51 85 124
		G	0.2096	8	357.6911					51 97 131
		B	0.1733	4	352.9125					35 70 96
	4	R	0.1452	7	359.3699	19.541	722.7346	0.8152	11.6124	19 51 72 143
		G	0.1478	5	357.9923					36 57 125 188
		B	0.1197	6	353.1714					43 82 134 243
	5	R	0.0397	9	359.5172	20.5495	572.9672	0.8193	12.475	23 44 85 147 248
		G	0.0616	7	358.074					45 56 88 102 166
		B	0.1083	8	353.2705					26 63 81 143 246
Proposed CDPSO [Section 5]	3	R	0.0274	12	358.9885	**19.3948**	**747.4838**	**0.8061**	5.8229	36 71 121
		G	0.0161	10	357.6947					45 79 126
		B	0.0075	10	352.9414					49 79 124
	4	R	0.0135	9	359.4173	**21.3358**	**478.0744**	**0.8702**	6.4791	31 58 91 136
		G	0.0104	15	358.024					36 59 90 134
		B	0.0109	14	353.2263					43 66 96 137
	5	R	0.0721	12	359.6784	**22.8708**	**335.7401**	**0.9101**	7.096	23 44 68 100 142
		G	0.0771	13	358.2428					32 52 75 105 145
		B	0.067	11	353.3874					38 55 76 105 143

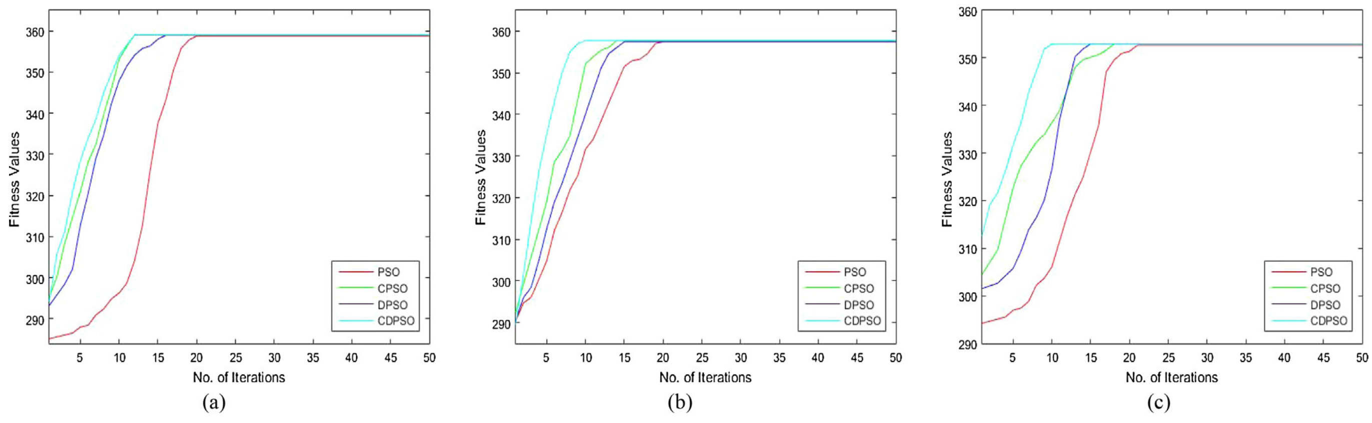


Fig. 8. Convergence characteristics of segmenting Satimage2 using Minimum Cross Entropy method with 3 thresholding levels using *PSO*, *CPSO*, *DPSO* and *CDPSO*$_{(proposed)}$ algorithms (a) R-channel, (b) G-channel, (c) B-channel.

Table 7
Comparison of standard deviation (STD), mean, PSNR, MSE, SSIM and running time of different chaotic optimization algorithms for three different thresholding levels (*n*) using Tsallis entropy measure for segmenting Satimage2. (Chl: Channel; Iter: No. of Iterations.)

Algorithm	*n*	Chl	STD	Iter	Mean	PSNR	MSE	SSIM	Time	Thresholds
CCS [Section 4.1]	3	R	0.87	18	1.2981	16.4433	1474.8	0.6414	8.7927	61 117 173
		G	0.9945	22	1.2161					69 124 179
		B	1.0543	15	1.0857					72 124 176
	4	R	0.5678	24	8.8631	18.5425	909.5627	0.7406	9.5576	46 92 138 184
		G	0.7786	22	8.1934					53 98 143 188
		B	1.0054	24	7.0746					63 105 148 190
	5	R	1.0426	18	5.4334	20.4478	586.5469	0.8256	10.1562	36 75 114 153 192
		G	0.8946	19	5.0152					45 84 124 164 203
		B	0.7896	17	4.1728					52 89 127 165 202
CHS [Section 4.2]	3	R	0.6534	24	1.2966	16.5072	1453.3	0.6457	4.1564	61 116 171
		G	0.8475	46	1.2152					69 125 179
		B	0.4596	51	1.0831					68 121 174
	4	R	0.8879	58	8.8483	18.5802	901.6959	0.7418	6.1719	45 90 137 184
		G	1.0456	59	8.1798					52 95 139 184
		B	1.0045	63	7.0456					59 99 142 185
	5	R	0.8756	54	5.4069	20.9821	518.648	0.8498	7.3581	35 71 110 150 190
		G	0.9756	48	4.9382					39 75 114 155 197
		B	0.8475	35	4.0315					43 73 112 154 194
CPSO [Section 4.3]	3	R	0.9564	22	1.2981	16.5473	1440	0.6474	9.1575	60 116 172
		G	0.8642	27	1.2161					68 123 178
		B	1.4526	29	1.0857					71 123 175
	4	R	0.8423	24	8.8606	18.6934	878.4914	0.7485	10.8511	44 91 137 183
		G	0.8845	32	8.1911					52 96 142 187
		B	1.0336	31	7.0746					62 104 147 189
	5	R	0.9541	27	5.4283	21.232	489.6476	0.8589	12.4834	35 73 112 151 190
		G	0.8452	22	4.9767					40 78 117 156 195
		B	1.0045	32	4.1521					45 82 119 157 195
CDE [Section 4.4]	3	R	0.8754	15	1.2971	16.4334	1478.2	0.6412	34.9663	58 102 151
		G	1.0045	17	1.2156					72 125 176
		B	1.8913	19	1.0846					75 126 174
	4	R	1.5705	12	8.8113	18.1648	992.2082	0.7262	39.2716	34 80 140 196
		G	1.347	17	8.1602					62 97 134 180
		B	0.9735	20	7.0491					64 95 139 185
	5	R	0.8741	18	5.3676	21.1911	494.2824	0.8632	43.877	37 72 117 163 189
		G	0.9806	14	4.9492					40 77 115 151 205
		B	1.1503	19	4.1194					38 80 108 157 203
Proposed CDPSO [Section 5]	3	R	0.073	14	1.2981	**16.5966**	**1423.7**	**0.6522**	15.1065	60 116 172
		G	0.7515	18	1.2161					69 124 179
		B	0.3959	21	1.0857					71 123 175
	4	R	0.3321	22	8.8631	**18.9578**	**826.6062**	**0.7629**	18.5802	46 92 138 184
		G	0.0991	19	8.1934					52 97 142 187
		B	0.6719	17	7.0746					63 105 148 190
	5	R	0.5822	20	5.4334	**21.473**	**463.2107**	**0.8684**	17.8261	35 74 113 152 191
		G	0.3434	22	5.0048					42 81 121 160 202
		B	0.5286	24	4.1645					47 83 121 159 196

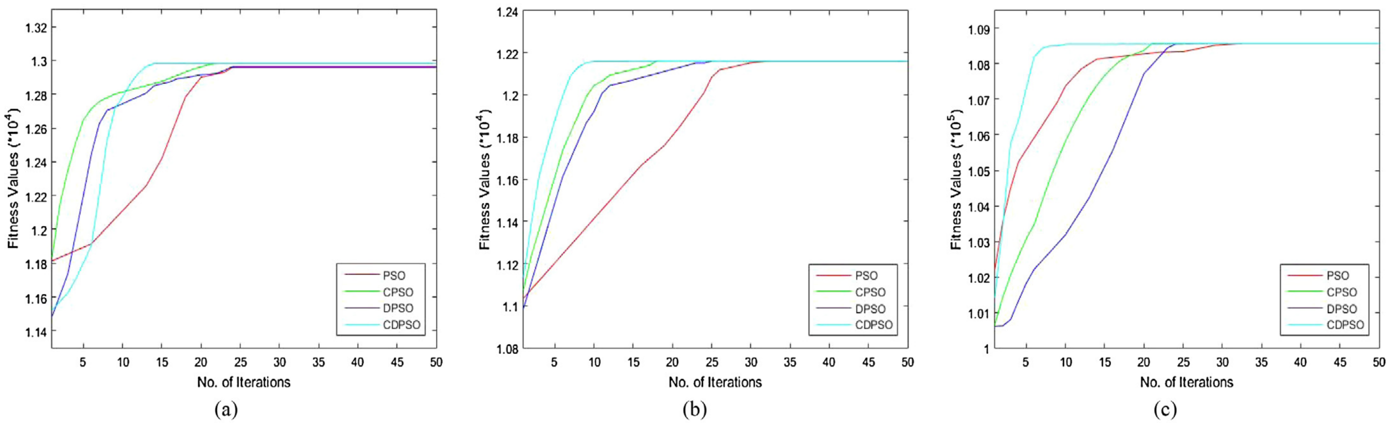


Fig. 9. Convergence characteristics of segmenting Satimage2 using Tsallis method with 3 thresholding levels using *PSO*, *CPSO*, *DPSO* and $CDPSO_{(proposed)}$ algorithms (a) R-channel, (b) G-channel, (c) B-channel.

rithm was tested by optimizing two different objective functions which includes Minimum cross entropy and Tsallis entropy measure. Qualitative and quantitative analysis presented in the paper clearly reveals that, the efficiency of the proposed CDPSO algorithm to outperform other existing chaotic algorithms to a greater extend. CDPSO algorithm minimizing cross entropy measure between the classes proved to be efficient than that using Tsallis method, on examining the quality metric values and on subjective evaluation, for satellite image segmentation scenario. The algorithm was also tested for its robustness by investigating its performance on natural images and proved to be very efficient compared with other state-of-the-art algorithms. In short chaotic DPSO ensures to be a stable, robust and fast algorithm for segmenting satellite images preserving image quality.

Chaotic DPSO emerges out with the highest mean value of fitness function among the five algorithms, particularly using minimum cross entropy as objective function, which reveals its efficiency in attaining global maxima. The problem of premature convergence shown by DPSO has been mitigated ensuring the algorithm to reach its optimum value in minimum number of iterations without getting trapped in local optima.

The algorithm suffers from the limitation of increased computational complexity as compared with some of the other state-of-the-art algorithms compared. Among the two, CDPSO maximizing Tsallis entropy resulted in the lowest quality metric values and highest computational time. However comparing with its chaotic counterparts, CDPSO proves to be well suited for satellite image segmentation application avoiding its chance of premature convergence.

The use of chaotic sequences in stochastic metaheuristic algorithms proves to be a good alternative to diversify its searching capability. The random and ergodic behavior of chaotic maps enriches the global search avoiding premature convergence. This encourages researches in remote sensing, computer vision and many other related domains. The scope of extending this algorithm for improving the efficiency of satellite image enhancement need to be explored. CDPSO optimizing multiobjective functions can be included as a future work. The proposed algorithm can also be exploited for segmenting other image datasets.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments which helped to improve the quality of the research paper.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.asoc.2017.02.005.

References

[1] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
[2] J.-P. Eckmann, D. Ruelle, Ergodic theory of chaos and strange attractors, Rev. Mod. Phys. 57 (3) (1985) 617.
[3] A. Kaveh, Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer, 2014.
[4] T. Mullin, The Nature of Chaos, Oxford University Press, USA, 1993.
[5] E. Mosekilde, Topics in Nonlinear Dynamics: Applications to Physics, Biology and Economic Systems, World Scientific, 1997.
[6] H. Liu, A. Abraham, M. Clerc, Chaotic dynamic characteristics in swarm intelligence, Appl. Soft Comput. 7 (3) (2007) 1019–1026.
[7] X.F. Yan, D.Z. Chen, S.X. Hu, Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model, Comput. Chem. Eng. 27 (10) (2003) 1393–1404.
[8] J.J. Yin, Chaotic Genetic Algorithms, in: Electronic Engineering, CityU Institutional Repository, 2004.
[9] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[10] J. Mingjun, T. Huanwen, Application of chaos in simulated annealing, Chaos Solitons Fractals 21 (4) (2004) 933–941.
[11] L. dos Santos Coelho, J.G. Sauer, M. Rudek, Differential evolution optimization combined with chaotic sequences for image contrast enhancement, Chaos Solitons Fractals 42 (1) (2009) 522–529.
[12] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[13] H.A. Hefny, S.S. Azab, Chaotic particle swarm optimization, in: 2010 The 7th International Conference on Informatics and Systems (INFOS), IEEE, 2010, pp. 1–8.
[14] Y. Sun, G. Qi, Z. Wang, B.J. van Wyk, Y. Hamam, Chaotic particle swarm optimization, in: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, 2009, pp. 505–510.
[15] L.D.S. Coelho, D.L. de Andrade Bernert, V.C. Mariani, A chaotic firefly algorithm applied to reliability-redundancy optimization, in: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2011, pp. 517–521.
[16] A. Gandomi, X.-S. Yang, S. Talatahari, A. Alavi, Firefly algorithm with chaos, Commun. Nonlinear Sci. Numer. Simul. 18 (1) (2013) 89–98.
[17] B. Alatas, Chaotic harmony search algorithms, Appl. Math. Comput. 216 (9) (2010) 2687–2699.
[18] M.F. El-Santawy, A. Ahmed, Z. El-Dean, A. Ramadan, Chaotic harmony search optimizer for solving numerical integration, Comput. Inf. Syst. 16 (2) (2012).
[19] J. Cai, X. Ma, L. Li, Y. Yang, H. Peng, X. Wang, Chaotic ant swarm optimization to economic dispatch, Electr. Power Syst. Res. 77 (10) (2007) 1373–1380.
[20] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.
[21] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. (2016) 1–14.
[22] L. Wang, Y. Zhong, Cuckoo search algorithm with chaotic maps, Math. Probl. Eng. (2015).
[23] A.H. Gandomi, X.-S. Yang, Chaotic bat algorithm, J. Comput. Sci. 5 (2) (2014) 224–233.
[24] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[25] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[26] R. Caponetto, L. Fortuna, S. Fazzino, M.G. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms, IEEE Trans. Evolut. Comput. 7 (3) (2003) 289–304.
[27] B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang, Improved particle swarm optimization combined with chaos, Chaos Solitons Fractals 25 (5) (2005) 1261–1271.
[28] K. Chandramouli, E. Izquierdo, Image classification using chaotic particle swarm optimization, 2006 International Conference on Image Processing (2006).
[29] Z. Yuan, L. Yang, Y. Wu, L. Liao, G. Li, Chaotic particle swarm optimization algorithm for traveling salesman problem, in: 2007 IEEE International Conference on Automation and Logistics, IEEE, 2007, pp. 1121–1124.
[30] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Probl. Eng. (2015).
[31] L.D.S. Coelho, V.C. Mariani, Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect, IEEE Trans. Power Syst. 21 (2) (2006) 989–996.
[32] A. Ouyang, G. Pan, G. Yue, J. Du, Chaotic cuckoo search algorithm for high-dimensional functions, J. Comput. 9 (5) (2014) 1282–1290.
[33] B. Adarsh, T. Raghunathan, T. Jayabarathi, X.-S. Yang, Economic dispatch using chaotic bat algorithm, Energy 96 (2016) 666–675.
[34] K.G. Dhal, M.I. Quraishi, S. Das, Development of firefly algorithm via chaotic sequence and population diversity to enhance the image contrast, Nat. Comput. 15 (2) (2016) 307–318.
[35] S. Gokhale, V. Kale, An application of a tent map initiated chaotic firefly algorithm for optimal overcurrent relay coordination, Int. J. Electr. Power Energy Syst. 78 (2016) 336–342.
[36] J. Yi, X. Li, C.-H. Chu, L. Gao, Parallel chaotic local search enhanced harmony search algorithm for engineering design optimization, J. Intell. Manuf. (2016) 1–24.
[37] A. Kaveh, Chaos embedded metaheuristic algorithms, in: Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer, 2016, pp. 369–391.
[38] D.A. Griffith, Spatial Autocorrelation, A Primer, Association of American Geographers, Washington, DC, 1987.
[39] P.L. Rosin, Unimodal thresholding, Pattern Recognit. 34 (11) (2001) 2083–2096.
[40] S. Arora, J. Acharya, A. Verma, P.K. Panigrahi, Multilevel thresholding for image segmentation through a fast statistical recursive algorithm, Pattern Recognit. Lett. 29 (2) (2008) 119–125.
[41] H. Gao, W. Xu, J. Sun, Y. Tang, Multilevel thresholding for image segmentation through an improved quantum-behaved particle swarm algorithm, IEEE Trans. Instrum. Meas. 59 (4) (2010) 934–946.
[42] S. Pare, A. Kumar, V. Bajaj, G. Singh, A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve, Appl. Soft Comput. 47 (2016) 76–102.

[43] S. Suresh, S. Lal, An efficient cuckoo search algorithm based multilevel thresholding for segmentation of satellite images using different objective functions, Expert Syst. Appl. 58 (2016) 184–209.
[44] S. Kullback, Information Theory and Statistics, Courier Corporation, 1968.
[45] C. Li, P.K.-S. Tam, An iterative algorithm for minimum cross entropy thresholding, Pattern Recognit. Lett. 19 (8) (1998) 771–776.
[46] R.M. Gray, Entropy and Information, Springer, 1990.
[47] W.-H. Tsai, Moment-preserving thresholding: a new approach, Comput. Vis. Graph. Image Process. 29 (3) (1985) 377–393.
[48] C. Tsallis, Possible generalization of Boltzmann–Gibbs statistics, J. Stat. Phys. 52 (1–2) (1988) 479–487.
[49] Y. Zhang, L. Wu, Pattern recognition via PCNN and Tsallis entropy, Sensors 8 (11) (2008) 7518–7529.
[50] P.S. Rodrigues, G.A. Giraldi, Computing the q-Index for Tsallis Nonextensive Image Segmentation, SIBGRAPI, 2009, pp. 232–237.
[51] S. Agrawal, R. Panda, S. Bhuyan, B. Panigrahi, Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm, Swarm Evolut. Comput. 11 (2013) 16–30.
[52] M.P. de Albuquerque, I. Esquef, A.G. Mello, Image thresholding using Tsallis entropy, Pattern Recognit. Lett. 25 (9) (2004) 1059–1065.
[53] Y. Zhang, L. Wu, Optimal multi-level thresholding based on maximum Tsallis entropy via an artificial bee colony approach, Entropy 13 (4) (2011) 841–859.
[54] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimisation with chaos, Neural Comput. Appl. 25 (5) (2014) 1077–1097.
[55] J.C. Sprott, J.C. Sprott, Chaos and Time-Series Analysis, vol. 69, Oxford University Press, Oxford, 2003.
[56] G. Heidari-Bateni, C.D. McGillem, A chaotic direct-sequence spread-spectrum communication system, IEEE Trans. Commun. 42 (234) (1994) 1524–1527.
[57] C.H. Skiadas, C. Skiadas, Chaotic Modelling and Simulation: Analysis of Chaotic Models, Attractors and Forms, CRC Press, 2008.
[58] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: NaBIC 2009. World Congress on Nature & Biologically Inspired Computing, 2009, IEEE, 2009, pp. 210–214.
[59] Z.W. Geem, J.H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[60] D. Oliva, E. Cuevas, G. Pajares, D. Zaldivar, M. Perez-Cisneros, Multilevel thresholding segmentation based on harmony search optimization, J. Appl. Math. (2013).
[61] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 760–766.
[62] B. Akay, A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding, Appl. Soft Comput. 13 (6) (2013) 3066–3091.
[63] S. Sarkar, S. Paul, R. Burman, S. Das, S. Chaudhuri, A fuzzy entropy based multi-level image thresholding using differential evolution, in: Swarm, Evolutionary, and Memetic Computing, Springer, 2014, pp. 386–395.
[64] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, 2006, pp. 485–492.
[65] R. Storn, On the usage of differential evolution for function optimization, in: NAFIPS. 1996 Biennial Conference of the North American Fuzzy Information Processing Society, 1996, IEEE, 1996, pp. 519–523.
[66] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evolut. Comput. 10 (6) (2006) 646–657.
[67] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evolut. Comput. 15 (1) (2011) 55–66.
[68] S. Kumar, M. Pant, A. Ray, Differential evolution embedded Otsu's method for optimized image thresholding, in: 2011 World Congress on Information and Communication Technologies (WICT), IEEE, 2011, pp. 325–329.
[69] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[70] J. Tillett, T. Rao, F. Sahin, R. Rao, Darwinian particle swarm optimization, Proceedings of the 2nd Indian International Conference on Artificial Intelligence (2005) 1474–1487.
[71] P. Ghamisi, M.S. Couceiro, N.M.F. Ferreira, L. Kumar, Use of Darwinian particle swarm optimization technique for the segmentation of remote sensing images, in: 2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE, 2012, pp. 4295–4298.
[72] P. Ghamisi, M.S. Couceiro, F.M. Martins, J. Atli Benediktsson, Multilevel image segmentation based on fractional-order Darwinian particle swarm optimization, IEEE Trans. Geosci. Remote Sens. 52 (5) (2014) 2382–2394.
[73] A. Hore, D. Ziou, Image quality metrics: PSNR *vs* SSIM, in: 2010 20th International Conference on Pattern Recognition (ICPR), IEEE, 2010, pp. 2366–2369.
[74] Q. Huynh-Thu, M. Ghanbari, Scope of validity of PSNR in image/video quality assessment, Electron. Lett. 44 (13) (2008) 800–801.
[75] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[76] D. Brunet, E.R. Vrscay, Z. Wang, On the mathematical properties of the structural similarity index, IEEE Trans. Image Process. 21 (4) (2012) 1488–1499.