Engineering Applications of Artificial Intelligence 23 (2010) 676–688

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem

Kamal Hammouche [a], Moussa Diaf [a], Patrick Siarry [b,*]

[a] Université Mouloud Mammeri, Département Automatique, Tizi-Ouzou, Algeria

[b] Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, EA 3956), Université Paris XII Val de Marne, 61 avenue du Général de Gaulle, 94010 Créteil, France

ARTICLE INFO

Article history:
Received 15 April 2009
Received in revised form
15 September 2009
Accepted 18 September 2009
Available online 5 November 2009

Keywords:
Multilevel thresholding
Image segmentation
Genetic algorithm
Particle swarm optimization
Differential evolution
Ant colony optimization
Simulated annealing
Tabu search

ABSTRACT

The multilevel thresholding problem is often treated as a problem of optimization of an objective function. This paper presents both adaptation and comparison of six meta-heuristic techniques to solve the multilevel thresholding problem: a genetic algorithm, particle swarm optimization, differential evolution, ant colony, simulated annealing and tabu search. Experiments results show that the genetic algorithm, the particle swarm optimization and the differential evolution are much better in terms of precision, robustness and time convergence than the ant colony, simulated annealing and tabu search. Among the first three algorithms, the differential evolution is the most efficient with respect to the quality of the solution and the particle swarm optimization converges the most quickly.

1. Introduction

Image thresholding is widely used as a popular tool in image segmentation. It is useful in separating objects from background, or discriminating objects from objects that have distinct gray levels. Thresholding involves bi-level thresholding and multilevel thresholding. Bi-level thresholding classifies the pixels into two groups, one including those pixels with gray levels above a certain threshold, the other including the rest. Multilevel thresholding divides the pixels into several classes. The pixels belonging to the same class have gray levels within a specific range defined by several thresholds. Both bi-level and multilevel thresholding methods can be classified into parametric and nonparametric approaches. The nonparametric approach is based on a search of the thresholds optimizing an objective function such as the between-class variance (Otsu's function) (Otsu, 1979), entropy (Kapur's function) (Kapur et al., 1985).

In parametric approach, the gray-level distribution of each class has a probability density function that is assumed to obey a given distribution. An attempt to find an estimate of the parameters of the distribution that best fits the given histogram data is made by using the least-squares method. It typically leads to a nonlinear optimization problem, of which the solution is computationally expensive and time-consuming. A great number of thresholding methods belonging to parametric and nonparametric approaches have been proposed in order to perform bi-level thresholding (Sezgin and Sankur, 2004; Lievers and Pilkey, 2004; Chu et al., 2004; Gonzales-Baron and Butler, 2006). They are extendable for multilevel thresholding as well. However, the amount of thresholding computation significantly increases with this extension. To overcome this problem, several techniques have been proposed. Some of them are designed especially for computation acceleration of a specific objective function, such as the Otsu's function (Lin, 2001; Dong et al., 2008; Riddi et al., 1987; Ng, 2004; Liao et al., 2001), while other techniques are designed to be used with a general purpose. Among the last category, we can find those that involve a sequential dichotomization technique (Yen et al., 1995; Sezgin and Tasaltin, 2000; Wu et al., 2004) and those that use an iterative scheme (Yin and Chen, 1997). In the first strategy, the histogram is dichotomized into two distributions by using a bi-level thresholding and the distribution with the largest variance is further dichotomized in two more distributions by applying the same bi-level thresholding. This dichotomization process is repeated until a stopping criterion is satisfied. The dichotomization techniques are faster algorithms. Unfortunately, they are sub-optimal techniques and they do not provide the optimal threshold values. The second approach starts with initial thresholds. Then, these thresholds are adjusted

* Corresponding author. Tel.: +33 145171567; fax: +33 145171492.
E-mail address: siarry@univ-paris12.fr (P. Siarry).

0952-1976/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2009.09.011

iteratively to improve the value of the objective function. This improvement process stops when the value of the objective function does not increase between two consecutive iterations. The implementation of this method is similar to the one presented by Luo and Tian (2000), where the Kapur's function is maximised by using the iterated conditional modes (ICM) algorithm. However, the iterative schemes are sensitive to initial values of thresholds and can converge to the local optimum. Other strategies can also be applied for fast multilevel thresholding, as the one proposed in Kim et al. (2003), where the resolution of the histogram is reduced using the wavelet transform. From the reduced histogram, the optimal thresholds are determined faster by optimizing the objective function based on an exhaustive search. The selected threshold values are then expanded to the original scale.

Another alternative to fast multilevel thresholding uses a new class of algorithms, called meta-heuristics. A meta-heuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of various problems (Blum and Roli, 2001; Glover and Kochenberger, 2003). In other words, a meta-heuristic can be seen as a general-purpose heuristic method, designed to guide an underlying problem specific heuristic toward promising regions of the search space, containing high-quality solutions. A meta-heuristic therefore is a general algorithmic framework, which can be applied to different optimization problems at the price of relatively few modifications. The meta-heuristic techniques are able to escape from local optima and their use has significantly increased the ability of finding very high-quality solutions to hard, practically relevant combinatorial optimization problems in a reasonable time. This is particularly true for large and poorly understood problems. Several different meta-heuristics have been proposed and new ones are under constant development. Some of the most famous ones are Genetic Algorithms (GA) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), Differential Evolution (DE) (Storn and Price, 1995), Ant Colony Optimization (ACO) (Dorigo and Stützle, 2000), Simulated Annealing (SA) (Kirkpatrick et al., 1983), Tabu Search (TS) (Glover and Laguna, 1997) and explorative local search methods, including Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), Guided Local Search (GLS) and Iterated Local Search (ILS) (Blum and Roli, 2001; Glover and Kochenberger, 2003).

Taking into account the advantages of the meta-heuristics to escape from local optima with a reasonable time, some meta-heuristic techniques have been extensively employed to search more fastly the optimal thresholds. These techniques are designed to be used with a general purpose, with either the parametric or the non-parametric approaches, for the multilevel thresholding problem. This problem, which consists in searching the various thresholds, can be considered as a nonlinear optimization problem and the objective functions can have several local optima. For solving such NP-hard problems, enumerative search methods are practically inappropriate because of the presence of several local minima. Their performance deteriorates with the complexity of the problem related to the number of thresholds. For example, the complexity of an exhaustive search grows exponentially with the number k of thresholds as $O(L^{(k-1)})$, L representing the number of gray levels in the image.

Indeed, the optimization techniques based on GA have been widely applied to solve the multilevel thresholding (Yin, 1999; Jinsong et al., 1999; Yang et al., 2003; Chang and Yan, 2003; Cao et al., 2008; Hammouche et al., 2008; Lai and Tseng, 2004; Tao et al., 2003; Bazi et al., 2007). In Yin (1999), Jinsong et al. (1999), Yang et al. (2003), Chang and Yan (2003), Cao et al. (2008), Hammouche et al. (2008), the GA uses binary encoding while in Lai and Tseng (2004), Tao et al. (2003) and Bazi et al. (2007), the floating encoding is used. In addition to the way of coding the vectors solutions, these techniques differ by their fitness function. In Yin (1999) and Cao et al. (2008) the objective function is similar to Otsu's or Kapur's functions. In Jinsong et al. (1999) the objective function is the Kapur's function and in Yang et al. (2003) it is assimilated to the relative entropy (Chang et al., 1994). Chang and Yan (2003) have employed a GA to maximize the conditional probability entropy (CPE) based on Bayesian theory, in order to determine the optimal thresholds. CPE considers that the pixels with the same gray level in an image may belong to different classes with different probabilities. An optimal classification method for these pixels is to classify them in the class with higher probability. More recently, the GA presented in Hammouche et al. (2008) determines the threshold number as well as the optimal threshold values, by using an objective function derived from a correlation function (Yen et al., 1995). In Lai and Tseng (2004), the intensity distributions of objects and background in an image are assumed to be Gaussian distributions with distinct means and standard deviations. The histogram of a given image is fitted to a mixture Gaussian probability density function. The GA is used to estimate the parameters in the mixture density function, so that the square error between the density function and the actual histogram is minimal. Tao et al. (2003) use a GA in order to find the optimal combination of all the fuzzy parameters by maximizing the fuzzy entropy. The fuzzy parameters describe the membership functions of three parts of the image, namely dark, gray and white parts. The optimal parameters are then used to define two threshold values. Bazi et al. (2007) use a genetic algorithm to provide the initial parameters to the expectation-maximization (EM) algorithm. The parameters of the objects and background classes are assumed to follow generalized Gaussian distributions. Each chromosome is viewed as a vector representing statistical parameters defining the mixture of the class distributions.

The DE, an improved version of GA is used for image thresholding. The proposed method in Rahnamayan et al. (2006) splits the input image in n sub-images and assigns a threshold level to each sub-image. Then the DE algorithm is applied to find the optimal threshold values by minimizing the dissimilarity between the input image and its binary version. Finally, the sub-images, thresholded by the corresponding threshold optimal values, are assembled to form the binary image. Unfortunately, this method which is specifically designed in the case of local and bilevel thresholding, cannot be applied to the global multilevel thresholding.

Besides GAs, PSO is another latest evolutionary optimization technique, which is used for the multilevel thresholding (Zahara et al., 2005; Yin, 2007; Feng et al., 2005; Maitra and Chatterjee, 2008; Fan and Lin, 2007). In Zahara et al. (2005), the PSO is used in conjunction with the simplex method for the Gaussian curve fitting and for the Otsu's function optimization. It is used in Yin (2007), for optimizing the cross entropy (Li and Lee, 1993) and for determining the single threshold value maximizing the 2-D entropy (Feng et al., 2005). The strategy developed in Maitra and Chatterjee (2008) uses the Kapur's function as criterion; it employs a cooperative learning that consists in decomposing a high-dimensional swarm into several one-dimensional swarms. The comprehensive learning allows discouraging convergence in each one-dimensional swarm. In Fan and Lin (2007), the histogram is approximated by a mixture Gaussian model. The Gaussian's parameter estimates are iteratively computed by combining the PSO with the EM algorithm.

Like the GA and the PSO, SA (Cheng et al., 1998; Hou et al., 2006; Nakib et al., 2007) and ACO (Tao et al., 2007) have been, also, exploited in image thresholding. In Nakib et al. (2007), the SA is used as a multiobjective optimization technique to find the optimal threshold values of three criteria, namely the within-class criterion, the entropy and the overall probability of error criterion. The same paper proposes a variant of SA in order to solve the Gaussian curve-fitting problem. As in Tao et al. (2003), the SA (Cheng et al., 1998) and the ACO (Tao et al., 2007) are also used to find the optimal combination of all the fuzzy parameters by maximizing the fuzzy entropy. The fuzzy parameters describe also the membership functions of two parts of the image, namely dark and bright parts.

Despite the intensive use of the meta-heuristics in thresholding, no paper related to the ACO applied specifically to global multilevel thresholding was found in our bibliography search. In addition, no work would be published concerning Tabu-based multilevel thresholding and DE algorithm allowing the direct computing of the threshold values. So, in this paper, three new multilevel thresholding techniques based on the DE, ACO and TS are developed in order to determine directly the values of thresholds. A comparative study between six meta-heuristic techniques, namely a GA, PSO, DE, ACO, SA and TS, is then conducted in the multilevel thresholding framework. The choice of these six meta-heuristics, contrary to the other methods, as explorative local search methods, is motivated by their use of nature inspired concepts. For each of the six meta-heuristics quoted previously, more sophisticated versions have been developed. However, to carry out an equitable comparison between these meta-heuristics, only the standard versions will be used in this paper.

The remainder of this paper is organized as follows. In Section 2, the problem of the multilevel thresholding is formulated as an optimization problem and the objective function treated are briefly presented. The Section 3 deals with a review of the meta-heuristic optimization techniques and their application for the resolution of the multilevel thresholding problem. Section 4 gives comparative results of the six implemented meta-heuristic techniques. Concluding remarks are given in Section 5.

2. Multilevel thresholding problem formulation

The optimal thresholding methods search for the thresholds such that the segment classes on the histogram satisfy the desired property. This is performed by minimizing or maximizing an objective function which uses the selected thresholds as parameters.

For thresholding purpose, the pixels of an image I having N pixels with L gray levels $L=\{0,1,\ldots,L-1\}$, are classified into K classes $(C_1,C_2,\ldots,C_k,\ldots,C_K)$ using a set of $(K-1)$ thresholds $T=(t_1,t_2,\ldots,t_k,\ldots,t_{K-1})$ such that $t_1<t_2<\ldots<t_{K-1}$. For convenience, we assume two extreme thresholds $t_0=g_{min}$ and $t_K=g_{max}$ where g_{min} and g_{max} are the lower and higher gray level in the image, respectively (typically $g_{min}=0$ and $g_{max}=L-1$).

A pixel with gray level g is assigned to class C_k if $t_{k-1}<g<t_k$, $k=1,2,\ldots,K$.

The thresholding problem consists in selecting the set of thresholds T^* which optimizes an objective function $F(T)$, such that:

$$T^*=\underset{0\le T\le L-1}{\operatorname{argmax}} F(T) \tag{1}$$

Several objective functions have been proposed in the literature devoted to the thresholding (Sezgin and Sankur, 2004). These functions are determined generally from the histogram of the image, denoted by $h(i)$, $i=0,\ldots,L-1$, where $h(i)$ represents the number of pixels having the gray level i. The normalized probability at level j is defined by the ratio $p_i=(h(i)/N)$.

Among these functions, the objective function defined by Otsu is the most popular (Otsu, 1979). It defines the weighted sum of within-class variances of the classes:

$$F(T)=\sigma_B^2=\sum_{k=1}^{K}\omega_k\left(\mu_k-\mu\right)^2 \tag{2}$$

where ω_k and μ_k are the probability and the mean, the gray level of the class C_k, respectively.

$$\omega_k=\sum_{i=t_{k-1}}^{t_k-1}p_i,\quad \mu_k=\frac{1}{\omega_k}\sum_{j=t_{k-1}}^{t_k-1}ip_i\quad \text{and} \tag{3}$$

$\mu=\sum_{i=0}^{L-1}ip_i$ is the total mean gray level of the image.

3. Review of meta-heuristic optimization techniques

Meta-heuristic techniques are optimization algorithms which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. The meta-heuristic feature permits the low level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbor solutions to consider in local search as is the case of SA or TS, or by combining elements taken by different solutions as is the case of GA, DE, PSO or ACO.

In the following sub-sections, these six above-mentioned meta-heuristic techniques used for multilevel thresholding are described in detail.

3.1. The genetic algorithm

GA is a search technique developed by Holland which mimics the principle of natural evolution (Goldberg, 1989). In the "simple GA" technique, the decision variables are first decoded into binary numbers (0 and 1) and hence create a population pool. Each of these vectors or chromosomes is then mapped into its real value using specified lower and upper bounds. A model of the process then computes an objective function for each chromosome and gives the fitness of the chromosome. The optimization search proceeds by using three operators: reproduction, crossover and mutation. The reproduction (selection) operator selects good strings in a population and forms the mating pool. The chromosomes are copied based on their fitness value. No new strings are produced in this operation. The crossover allows for a new string formation by exchanging some portions of a string (chosen randomly) with a portion of a string of another chromosome, generating child chromosome in the mating pool. If the child chromosomes are less fitted than the parent chromosomes, they will slowly die in the subsequent generation. The effect of crossover can be detrimental or good. Hence, not all the strings are used for crossover. A crossover probability P_c is used, where only 100 P_c percent of the strings in the mating pool are involved in crossover operation, while the rest of them are kept unchanged in the next generation. Mutation is the last operation. It is used to further perturb the child vector using mutation probability P_m. The mutation alters the string locally to create a better string. Mutation is needed to create a point in the neighborhood of the current point, thereby achieving a local search and maintaining the diversity in the population. The entire process is repeated till some termination criterion is met.

Table 1
Pseudo-code of the genetic algorithm: $rand_{INT}(a, b)$: uniform pseudo random integer in the interval $[a, b]$. $rand(a, b)$: uniform pseudo-random real number in the interval $[a, b]$.

```
Procedure Genetic algorithm

1- Initialize population pop "Create population from randomly chosen threshold values"
2- Evaluate population pop "Evaluate all candidate solutions"
3- Store the best solution T* (elite) with its fitness in a separate location.

4- for a fixed number of iterations

    "Apply selection: tournament selection "
      oldpop = pop
      for all individuals j in pop
        if individual j , i.e., pop(j), is not in the elite then
           select another individual m randomly, i.e., m = rand_INT(0,N)
           if individual m in oldpop is better than individual j in pop then
              pop(j) = oldpop(m)

    "Apply crossover: arithmetic crossover"
        oldpop = pop
        for all individuals j in pop
           if individual j is not in the elite and rand(0,1)<Pc then
             select another individual m randomly, i.e., m = rand_INT(0,N)
             for all candidate solution parameters k in T
                 cw = rand (0 , 1)
                 pop(j)_t_k = cw * oldpop(j )_t_k + (1 − cw) * oldpop(m)_t_k

    "Apply mutation and check bounds: Gaussian mutation with fixed mutation rate"
        for all individuals j in pop
           if individual j is not in the elite and rand(0,1)<Pm then
             for all vector solution parameters k in T
                 pop(j)_t_k = pop(j)_t_k + N(0, 1) * σ* (g_max − g_min)
                 if pop(j)_t_k >g_max then pop(j)_t_k = g_max
                 if pop(j)_t_k <g_min then pop(j)_t_k = g_min

     Evaluate population pop "Evaluate all candidate solutions"

    Compare the best individual T of the pop with T*. If T has a fitness value better than
     T*, then replace T* with T.

5-Output best recorded solution T*
```

For the multilevel thresholding, we have chosen a numerical optimization version of the GA with floating point encoding, Gaussian mutation, arithmetic crossover and tournament selection. In our GA implementation (see Table 1), first a population of N individuals containing the vector solutions encoded in floating point numbers is created (initialization) and the fitness of each individual is evaluated by the fitness function (evaluation). As mentioned above, for the initialization of the population, the GA uses randomly chosen threshold values from the range $[g_{min}\, g_{max}]$. Each individual j of the population *pop* contains K threshold values noted $pop(j)\_t_k$ (k=1,2, …, K), such that $pop(j)\_t_k$ $[g_{min}\, g_{max}]$, where g_{min} and g_{max} are the minimum and the maximum gray values in the image, respectively.

After initialization, we evaluate the individuals stored in variable *pop* according to the fitness function and determine the best individual T^* (elite). The population is iteratively refined by selection of individuals, application of mutation and crossover operators, re-evaluation of the new population according to the fitness function and updating of the best solution T^*. We use tournament selection. We first save the current population *pop* as *oldpop* and for each individual j we choose another individual m randomly from *oldpop*, compare the fitnesses, and substitute j by m in population *pop* if m's fitness is better. Before applying crossover, we save again the current population *pop* as *oldpop*, and then apply arithmetic crossover, as follows:

$$pop(j)\_t_k = cw*oldpop(j)\_t_k + (1-cw)*oldpop(m)\_t_k \tag{4}$$

where $pop(j)\_t_k$ is the kth solution parameter (threshold value) of individual j, $pop(j)$ is the offspring of the parents $oldpop(j)$ and $oldpop(m)$, cw is a uniform random weight $cw \in [0 \quad 1]$, which is generated for each problem parameter (threshold) k. Gaussian mutation was chosen, such that $pop(j)\_t_k = pop(j)\_t_k + N(0,\ 1)*\sigma*(g_{max}-g_{min})$, where $N(0,\ 1)$ is the Gaussian normal distribution with mean 0 and variance 1, and σ is the variance parameter of the mutation operator which is fixed to 0.05 in our experiments.

The crossover and mutation operators are applied to each individual in the population, with a probability P_m for mutation and P_c for crossover, respectively. The algorithm terminates after a fixed number of iterations. The optimization result is given by the best individual T^* in the last generation.

3.2. Particle swarm optimization

PSO, inspired by the swarming behavior of animals as bird flocking, was introduced by Kennedy and Eberhart (1995). A particle swarm is a population of particles, where each particle is a moving object that 'flies' through the search space and is attracted to previously visited locations with high fitness. In contrast to the individuals in evolutionary computation, particles neither reproduce nor get replaced by other particles.

Each particle consists of a position vector T, which represents the candidate solution to the optimization problem, the fitness $F(T)$ of solution T, a velocity vector V and a memory vector *Tbest* of the best vector solution encountered by the particle with its recorded fitness.

The position of a particle is updated by

$$T(t+1) = T(t) + V(t) \tag{5}$$

and its velocity according to

$$V(t+1) = V(t) + \varphi_1 \Big(Tbest - T(t) \Big) + \varphi_2 \Big(T^* - T(t) \Big) \tag{6}$$

where φ_1, φ_2 are uniform distributed random numbers within $[\varphi_{min},\ \varphi_{max}]$ (typically φ_{min}=0 and φ_{max}=2) that determine the weight between the attraction to position *Tbest*, which is the best position found by the particle so far and T^* the overall best position found by all particles. Note that φ_1 and φ_2 are generated for each component of the velocity vector. Moreover, the so-called inertia weight w controls how much the particles tend to follow their current direction compared to the memorized positions *Tbest* and T^*. Finally, the velocity of the particles is limited by a maximum velocity v_{max}.

As it has been mentioned in the introduction, several multilevel thresholding methods based on the PSO have been proposed with different strategies. In our study, we use a simple PSO algorithm for a fair comparison with other meta-heuristic techniques. The algorithm works as outlined in the pseudo-code of Table 2. Denoting the number of particles by N and the swarm particles by the population *pop*, the position vector T of each swarm particle j, which contains the threshold values t_k (k=1,2, …, K), is denoted by $pop(j)\_t_k$ and the velocity vector $V=(v_1,v_2, \ldots, v_K)$ of each swarm particle j is noted $pop(j)\_v_k$. (j=1,2, …, N). The best positions of the particles are memorized in another population denoted by *bestpop*.

The initialization of the algorithm is similar to that used for the GA above (using randomly chosen threshold values), but additionally requires the initialization of the velocity vectors, which are uniformly distributed random numbers in the interval $[0,\ v_{max}]$. After initialization, the memory of each particle is updated and the velocity and position update rules are applied. The velocity of each swarm particle is updated using the following equation:

$$pop\_v_k = w*pop\_v_k + \phi_1 (bestpop\_tb_k - pop\_t_k) + \phi_2 (t_k^* - pop\_t_k) \tag{7}$$

If a component k of the velocity v of a particle exceeds v_{max} it is truncated to this value, i.e., $pop(j)\_v_k = v_{max}$. The position of each swarm particle j is also updated as follows:

$$pop(j)\_t_k = pop(j)\_t_k + pop(j)\_v_k \tag{8}$$

Table 2
Pseudo-code of the PSO algorithm.

```
Procedure Particle swarm optimization

1- Initialize population pop "create population from randomly chosen threshold values"
2- Evaluate population pop "evaluate all candidate solutions"
3- Store the best solution $T^*$ with its fitness in a separate location.

4- Update particle memories
   For a fixed number of iterations
     For all individuals j in population pop
       For all candidate solution parameters k in T

         "Update velocity"
           $\varphi_1 = rand(\varphi_{\min}, \varphi_{\max})$ and $\varphi_2 = rand(\varphi_{\min}, \varphi_{\max})$
           $pop\_v_k = w * pop\_v_k + \varphi_1(bestpop\_t_k - pop\_t_k) + \varphi_2(t_k^* - pop\_t_k)$

         "Constrain particle velocity to [$v_{min}$, $v_{max}$]"
           if $pop(j)\_v_k > v_{max}$ then $pop(j)\_v_k = v_{max}$
           if $pop(j)\_v_k < v_{min}$ then $pop(j)\_v_k = v_{min}$

         "Update position"
           $pop(j)\_t_k = pop(j)\_t_k + pop(j)\_v_k$

         "Constrain particle position to [$g_{min}$, $g_{max}$]"
           if $pop(j)\_t_k < g_{min}$ then $pop(j)\_t_k = g_{min}$
           if $pop(j)\_t_k > g_{max}$ then $pop(j)\_t_k = g_{max}$

       endfor //k
     endfor //j

     Evaluate population pop "evaluate all candidate solutions"
     "Update Particle Memories"
       For each particle j
         If its fitness is better than that of the same particle in the bestpop then
           $bestpop(j)\_t_k = pop(j)\_t_k$
       Compare the best individual T of the pop with $T^*$. If T has a fitness value better than
       $T^*$, then replace $T^*$ with T.
   endfor //iteration

5-Output best recorded solution $T^*$
```

If a component k of the new position vector is outside the domain, it is truncated to the limit values g_{min} and g_{max}. Then, the population of the best position encountered by each particle and by all particles is updated. This process is applied to all particles and repeated for a fixed number of iterations. The best recorded vector solution T^* in the last iteration and its fitness constitute the optimization result.

3.3. *Differential evolution*

DE is another evolutionary optimization technique developed by Storn and Price (1995). It is simple to implement, requires little or no parameter tuning, and is known for remarkable performance. A number of recent studies comparing DE with other heuristics, such as GA and PSO, indicate superiority of DE (Vesterstrom and Thomsen, 2004; Paterlini and Krink, 2006). The outlines of the proposed algorithm for thresholding are presented in Table 3.

After generating and evaluating an initial population, exactly in the same way as described for the GA and PSO above, three operators named mutation; crossover and selection are successively applied in each generation. The mechanics of mutation and crossover differ from those used in GA. Basically, the mutation creates a mutant vector Y by adding the weighted difference between two population vectors to a third vector population. Hence, for each vector solution j, choose three other vector solutions l, m and n randomly from the population (with $j \neq l \neq m \neq n$), calculate the difference of l and m, scale it by multiplication with a parameter f and add the result to n in order to create a candidate solution $Y=(y_1, y_2, \ldots, y_k, \ldots, y_K)$.

$$y_k = pop(n)\_t_k + f\left(pop(m)\_t_k - pop(l)\_t_k\right) \tag{9}$$

Table 3
Pseudo-code for the DE Rand/1/Exp operator.

```
Procedure Differential evolution

1- Initialize population pop "create population from randomly chosen threshold values"
2- Evaluate population pop "evaluate all candidate solutions"
3- For a fixed number of iterations
     For all individuals j in population pop

       "Create difference vector"
         Select l ,m,n uniform randomly from [1..N] with $j \neq l \neq m \neq n$
         For all candidate solution parameters k
           $y_k = pop(m)\_t_k + f * (pop(n)\_t_k - pop(l)\_t_k)$

       "Apply DE-crossover"
         For all candidate solution parameters k
           if rand(0 1) < Cr then $c_k = y_k$ else $c_k = pop(j)\_t_k$

       "Evaluate the new candidate solution"
         Evaluate vector solution C

       "Apply selection"
         if fitness F(C) of the solution C is better than the fitness of pop(j) then
           $pop(j)\_t_k = c_k$
           $pop(j)\_fitness = F(C)$

     endfor //j
   endfor //iteration

4- Output best recorded solution $T^*$ of the population pop
```

Afterwards create another vector solution C by crossover of Y and vector solution j, as follows:

$$c_k = \begin{cases} y_k & \text{if } r \text{ and} \geq Cr \\ pop(j)\_t_k & \text{otherwise} \end{cases} \tag{10}$$

where Cr is predefined crossover constant and rand is uniform pseudo-random real number in the interval [0, 1].

The crossover process used in DE is somewhat similar to uniform crossover in GAs, such that the result of the crossover contains a certain proportion of consecutive solution parameters of each parent. Finally, evaluate the new candidate solution with the fitness function and apply tournament selection by substituting vector solution j with the new vector solution C and its fitness, in case the fitness of C is better than the fitness of vector solution j. This description which refers to the so-called *Rand/1/Exp* DE operator, which is the most successful, is adopted in our implementation. Different other strategies have been suggested. These strategies can vary based on the vector to be perturbed in the mutation process, the number of difference vectors considered for perturbation and the type of crossover used (Storn and Price, 1997). The process is repeated for a fixed number of iterations and the optimization result is the best recorded vector solution and fitness at the end of the run.

In DE algorithm, just three factors control evolution: the population size, N; the weight applied to the random differential, f; and the crossover constant, Cr. Larger values for f result in higher diversity in the generated population and lower values in faster convergence. The scaling factor f is a user supplied constant in the range $(0 < f \leq 1.2)$. The constant Cr belongs to the range [0, 1].

3.4. *Ant colony optimization*

ACO meta-heuristic, a population-based approach was proposed by Dorigo et al. to solve several discrete optimization problems (Dorigo and Stützle, 2000). The ACO mimics the way of the real ants to find the shortest route between a food source and their nest. The ants communicate with one another by means of pheromone trails and exchange information about which path should be followed. The more the number of ants traces a given path, the more attractive this path (trail) becomes and is followed by other ants by depositing their own pheromone. This auto

Table 4
Pseudo-code for the ant colony optimization.

```
Procedure Ant colony optimization

1- Create the pheromone matrix $\tau$
2- Initialize the populations pop
3- Store the best solution T* of the population pop with its fitness in a separate location.
4- For a fixed number of iterations

    "Determine the population pop from the pheromone matrix $\tau$"
      For all individuals j in population pop
        If rand >$q_0$ then
          $pop(j)\_t_k = \underset{i\in[g_{\min}\quad g_{\max}]}{\arg\max}\,\tau_{ik}$
        Else $pop(j)\_t_k = rand_{INT}[g_{\min}\quad g_{\max}]$

    Evaluate population pop "evaluate all candidate solutions"

    "Update the best solution T*"
      Compare the best individual T of the pop with T*. If T has a fitness value better than
      T*, then replace T* with T. $F_{max}=F(T^*)$

    "Update the pheromone trails"
        $\tau_{ik} = \rho\tau_{ik} + (1\quad\rho)F_{\max}$
  endfor //iteration

5- Output best recorded solution T*
```

catalytic and collective behavior results in the establishment of the shortest route. This real-life search behavior was the key motivation factor leading to the formulation of artificial ant algorithms to solve several large-scale combinatorial and function optimization problems. In all these algorithms, a set of ant-like agents or software ants solve the problem under consideration through a cooperative effort. This effort is mediated by exchanging information on the problem structure the agents concurrently collect while stochastically building solutions. The first ACO algorithm proposed in the literature was called Ant System (AS). Its computational results were promising but not competitive with other more established approaches. Therefore, improved versions, such as Ant Colony System (ACS) (Dorigo and Gambardella, 1997) and Max–Min Ant System (Stützle and Hoos, 1998), has been proposed. These algorithms differ mainly with the AS by the way of pheromone global updating. In the ACS, the management of the trails is subdivided into two levels: a local update and a global update. Hence, we have chosen in this paper to use, as basic version of ACO, the ACS without the local updating rule, like in the ant system. In the proposed ACO algorithm for multilevel thresholding, a set of concurrent distributed agents collectively discover the optimal threshold values. The summary of the proposed ant algorithm for multilevel thresholding is depicted in Table 4.

In the proposed ACO, a colony of N simple agents or artificial ants search for good solutions at every generation. The ants evolve a set of solutions and form a population of threshold values like for the GA, PSO and DE algorithms. The threshold values produced by the ant j is also noted $pop(j)\_t_k$ (k=1,2, …, K). Every artificial ant j of a generation builds up a solution $T=(t_1,t_2, \ldots, t_k, \ldots, t_K)$ by using the information provided by the pheromone matrix denoted τ, where each element τ_{ik} is pheromone trail that lies the gray level i to the kth threshold, i=0,2; …, $L-1$ and k=1,2, …, K. Initially, the trail pheromones τ_{ik} are randomly generated in the range [0, 1].

To generate a solution T, the agent selects the threshold value for each element of string T according to the following rule: using probability q_0, the gray value having the maximum pheromone concentration is chosen as threshold value, i.e. $pop(j)\_t_k = \underset{i\in[g_{min}\quad g_{max}]}{\text{argmax}\tau_{ik}}$. Otherwise the kth threshold value is determined using a stochastic distribution with a probability $(1-q_0)$, such that: $pop(j)\_t_k = rand[g_{min}\quad g_{max}]$. q_0 is a priori defined number, $0<q_0<1$, q_0 is fixed to 0.98 in our simulations. In ACS, this rule is called pseudo-random-proportional rule.

Once all ants have their solutions built up, these solutions are evaluated according to the objective function. The value of best solution in memory (T^*) is updated with the value of the solution obtained as "current iteration best solution", if it has a best objective function value than that of the best solution in memory. The pheromone trails are then updated. The trail updating process applied in this algorithm is performed as follows:

$$\tau_{ik} = \rho\tau_{ik} + (1-\rho)\Delta\tau \tag{11}$$

where ρ is the persistence of trail that lies within [0, 1] and $(1-\rho)$ the evaporation rate. Higher value of ρ suggests that the information gathered in the past iterations is forgotten faster. $\Delta\tau$ denotes the amount of pheromone trail added to τ_{ik} by the best ant corresponding to the best solution found so far: $\Delta\tau = F_{max}$, where is the fitness of the best solution T^*.

Such a pheromone updating process reflects the usefulness of dynamic information provided by the artificial ants. Thus, the pheromone matrix is a kind of adaptive memory that contains information provided by the previously found superior solutions, and is updated at the end of the iterations. At any iteration level, the algorithm essentially executes two steps, viz., (1) generation of new N solutions by artificial ants using the modified pheromone trail information available from previous iteration and (2) updating pheromone trail matrix. The algorithm repeatedly carries out these two steps for a maximum number of given iterations, and solution having the best function value represents the optimal threshold values.

3.5. Simulated annealing

SA is commonly considered to be the oldest among the metaheuristics and surely one of the first algorithms that has an explicit strategy to avoid local minima. The term simulated annealing derives from the roughly analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. It was first presented as a search algorithm for optimization problems in Kirkpatrick et al. (1983). The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (uphill moves) in order to escape from local minima. The probability of doing such a move is decreased during the search.

The algorithm starts by generating an initial solution (either randomly or heuristically constructed) and by initializing the so-called temperature parameter $temp$. Then the following process is repeated until the termination condition is satisfied: A solution T' from the neighborhood $V(T)$ of the solution T is randomly sampled and it is accepted as new current solution depending on $F(T)$, $F(T')$ and T. T' replaces T if $F(T')<F(T)$ or, in case $F(T')> =F(T)$, with a probability which is a function of $temp$ and $F(T')$–$F(T)$. The probability is generally computed following the Boltzmann distribution

$$\exp(-(F(T')-T(T))/temp).$$

The temperature $temp$ is decreased during the search process; thus at the beginning of the search the probability of accepting uphill moves is high and it gradually decreases, converging to a simple iterative improvement algorithm. The temperature is reduced using a geometric rule $temp(t+1)=temp(t)*\alpha$.

α is a factor chosen in the range [0.5, 0.99].

The application of SA for multilevel thresholding is summarized in Table 5.

Table 5
Pseudo-code for the simulated annealing.

```
Procedure Simulated annealing

1- Generate randomly one initial vector solution $T=(t_1,t_2,\ldots, t_k,\ldots,t_{K-1})$
2- Evaluate the vector solution and store it as the best solution $T^*$ $(T^*=T)$
3- $Temp=Temp_{init}$
   While $temp<temp_{final}$
   For a fixed number of iterations
     Create a vector solution $T'$ in neighborhood of the vector solution $T$
       $t'_k = t_k \pm rand_{INT}(0 \quad \frac{L \quad 1}{32})$
       if $t'_k < g_{min}$ then $t'_k = g_{min}$
       if $t'_k > g_{max}$ then $t'_k = g_{max}$

     $\Delta E=F(T')-F(T)$
     if $\Delta E<0$ then $T=T'$
     else if $rand(0 \quad 1)<e^{-\Delta E/temp}$ then $T=T'$

    "Update the best solution"
      if $F(T^*) < F(T)$ then $T^*=T$
   endfor // iteration
   $temp = temp * \alpha$
   endwhile

4- Output best recorded solution $T^*$.
```

Table 6
Pseudo-code for the tabu search.

```
Procedure Tabu search

1- Generate randomly one initial vector solution $T=(t_1,t_2,\ldots, t_k,\ldots,t_{K-1})$
2- Evaluate the vector solution and store it as the best solution $T^*$ $(T^*=T)$
3- Initialize the tabu memory $pop=\Phi$
4- For a fixed number of iterations
     "Create a best vector solution $v_{best}$"
     Build a set of neighborhoods $V(T)$ for the vector solution $T$
     For each $v \in V(T)$
       If $v \notin pop$ and $F(v) > F(v_{best})$ then $v_{best} = v$
     endfor
     $T = v_{best}$
     "Update of the tabu memory"
       $pop = pop \cup v_{best}$
     "Update the best solution"
       if $F(T^*) < F(T)$ then $T^*=T$
   endfor //iteration

5- Output best recorded solution $T^*$.
```

3.6. Tabu search

The TS proposed by Glover and Laguna (1997) is a well-known meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. Its major components are a set of moves for generating a set of trial solutions, a set of tabu restrictions (recorded in a tabu memory) for forbidding some moves, and a set of aspiration criteria for releasing some forbidden moves. For a given solution, a given tabu memory recording the given solution, and a given set of aspiration criteria, the TS regards the given solution as the current solution and the best solution visited. It then iteratively operates according to the following steps, until the termination condition is satisfied. First, the TS performs a set of moves on the current solution to generate a set of trial solutions. Based on the tabu memory, the TS checks whether each newly generated trial solution is tabu. If the newly generated trial solution is identical to one of the solutions in the tabu memory, it is tabu; otherwise, it is an allowable solution. If the tabu solution satisfies one of the aspiration criteria, the TS then restores the tabu solution to an allowable solution.

The TS selects the best among all allowable trial solutions as the next solution, and then, inserts the next solution into the tabu memory. If the tabu memory is full, the oldest solution in the tabu memory is removed. Finally, the next solution replaces the best solution visited, if the next solution is better.

The role of the tabu restrictions is to prevent solution cycling (repeatedly visiting the same points in the search space), which forces the search in different directions instead of trapping it into local optima. The aspiration criteria enable the generation of better solutions from the tabu solutions because the tabu solutions may contain good attributes. Two important components of TS are intensification and diversification strategies (Blum and Roli, 2001). Intensification strategies encourage the exploration of the search space around good solutions already found, while the diversification strategies seek to spread the search towards previously unexplored regions of the search space. Some advanced mechanisms exploiting information collected during the whole search process are commonly introduced in tabu search to deal with the intensification and the diversification of the search (Gendreau, 2002).

TS has exhibited great success in many applications; however, in our knowledge, no paper has until now treated the problem of thresholding through the TS. Hence, we propose an application of this algorithm to optimize an objective function related to the multilevel threshold purpose. Although there are more or less sophistical variants of the TS algorithm, we have considered the standard version, in order to keep the requirement of an equitable comparison. The proposed algorithm is described in Table 6.

In this proposed algorithm, the tabu memory of size N is noted *pop*. The set of trial solutions are denoted by $V(T)=(v_1, v_2, \ldots, v_i, \ldots v_{Nv})$, they correspond to the neighborhoods of the vector solution T. A trial solution v_{ik} can be created, as in SA, by modifying locally and randomly the threshold values t_k, as follows:

$$v_{ik} = t_k \pm rand_{INT}[0, \quad (L-1)/32] \tag{12}$$

The number N_v of neighbors is fixed, after several experimentations, to 20. The interval $[0, \quad (L-1)/32]$ constitutes a small part of number L of gray values present in the image.

4. Experimental results and comparison study

In this section, we will evaluate the performance on the multilevel thresholding methods based on the six meta-heuristic techniques presented in the previous section. Some experiments with real images are presented to illustrate the key features of each optimization technique in the efficiency of the thresholding computation. Six well-known images named *Lena, peppers, house, airplane, lake* and *boats* with 256 gray levels are used. These images of size (256×256) are shown in Fig. 1. Fig. 2 shows their respective histograms.

In order to compare the quality of the solutions provided by the six meta-heuristic techniques for the multilevel thresholding, the value of the best fitness $F(T^*)$ corresponding to the best threshold solution T^* is used as comparative criterion. Of course, the smaller the objective function value, the better the algorithm. Additional results are presented in order to investigate the influence of the time computation of each optimization technique.

The meta-heuristic techniques presented in the Section 3 have several parameters of which values considerably affect the performance of the algorithms. We have done preliminary testing on these algorithms for the purpose of getting good combinations of parameters and results are listed in Table 7. Broadly speaking, these preliminary tests consisted in tuning, one by one, each parameter of each algorithm. For each algorithm, we varied the values of the parameter to be tuned while fixing the values of the other parameters (initially these values were selected in accordance with the recommendations mentioned in the literature). The optimal value of the parameter to be tuned is that which gives the smallest value of the objective function. The size N of the population and the number of iterations have a great influence on the computing time and on the convergence. As these

Fig. 1. Test images: (a) Lena, (b) Peppers, (c) House, (d) Boats, (e) Lake, and (f) Airplane.

two parameters are linked, the population size N was kept unchanged, while the number of iterations was taken as a variable for all algorithms, in order to further facilitate the comparison between them for time convergence. The size N of population is also linked to the number of thresholds (i.e. the number of variables) to be determined, as it is explained later in this section. A common value of N equal to 100 for all algorithms was a good choice for all tests carried out in this paper.

Since the heuristic algorithms are of stochastic type, the six algorithms are run 50 times. Table 9 reports the mean and the standard error of the mean fitness over 50 runs, for each image with a threshold number varying from 1 to 4. These mean values of the fitness can be compared to the optimal values of the objective functions found by an exhaustive search (Table 8).

We can see that all the algorithms give good results both in terms of accuracy (mean fitness) and robustness (small variance, i.e. similar results of repeated runs), when the number of thresholds is small. The TS give mean fitness values equal to the optimal values when only one threshold is searched, while the ACO and SA are efficient only when the number of thresholds do not exceed 2. For a great threshold number, the results obtained by GA, PSO and DE are much better than the others. These three algorithms converged consistently to the same solution with the same fitness value and very lower variance. However, in few situations, these techniques become less robust since the variance values are relatively high. It is the case when the number of thresholds is equal to 4 for the GA with House, Lake and Airplane images; for PSO with the Peppers, House and Airplane images and the DE with the Boats image. DE maintains a robust convergence for all images (except for Boats image). Thus, the DE seems the most efficient. However, in a general way, we can say that the results of the GA, PSO and DE are comparable and are close to those provided by the exhaustive search.

In the above experiment, the number of iterations, which is used as stopping criterion, differs from an algorithm to another one. However, this parameter considerably affects their time computation. In order to compare the convergence time of the six meta-heuristic algorithms, we have computed the iteration number and the time necessary to ensure the convergence to the optimum for each algorithm. The run of each algorithm was stopped when the fitness value $F(T^*)$ of the best solution reached the optimal value of the objective function (F_{op}) stored in Table 8, i.e. $|F(T^*) - F_{op}| \le \varepsilon = 10^{-9}$, where ε is a threshold value which fixes the accuracy of the measurement. We note then the number of iterations and time taken by each algorithm to achieve the desired accuracy. Therefore, the stopping criterion of all algorithms (Tables 1–6) is modified; it is based on the value of the fitness and not on the number of iterations. Each algorithm was run 50 times, the Table 10 gives the mean number of iterations and the average of the CPU time taken by each algorithm to meet the stopping criteria. Our experiments are performed on a HP Pentium IV-2.8 GHz PC with 256 Mb RAM.

As for the exhaustive search, for all meta-heuristic algorithms, the number of iterations and the run time increase with the threshold number but not of the same manner. The convergence times of the GA, PSO and DE are faster than those of the exhaustive search except for one threshold. This exception can be explains by the size N of the population which is fixed to 100 for all algorithms. Indeed, the size N of the population has a great influence on the run time. However, when the threshold number is small, it not necessary to use a great size of the population. Therefore, for one or two thresholds, the run times of the meta-heuristic algorithms can be significantly shorted by reducing the size of the population. The GA, PSO and DE converge in little iterations. However, by comparing these three algorithms between them, we can see that the PSO is the most efficient in terms of time execution, followed by the GA and the DE. The GA, PSO and DE converge much faster comparatively to the ACO, SA and TS. The iteration numbers necessary to ensure of the ACO, SA and TS are higher to those of the GA, PSO and DE. The convergence times of the ACO, SA and TS are also shorter than those of the exhaustive search except for one or two thresholds for the same raisons previously explain.

Finally, the tests carried out show that the performances of the GA, PSO and DE in terms of precision, robustness and time convergence are much better comparatively to the ACO, SA and TS. Among the first three algorithms, the DE is the most efficient with respect to the quality of the solution and the PSO converges the most quickly.

5. Conclusion

In this paper, three new multilevel thresholding techniques based on DE, ACO and TS were proposed in order to determine directly the values of thresholds. Furthermore, we considered three other meta-heuristic algorithms for solving the multilevel thresholding problem, namely GA, PSO and SA.

These six meta-heuristic algorithms were then compared by testing them on various images. We have found that all algorithms

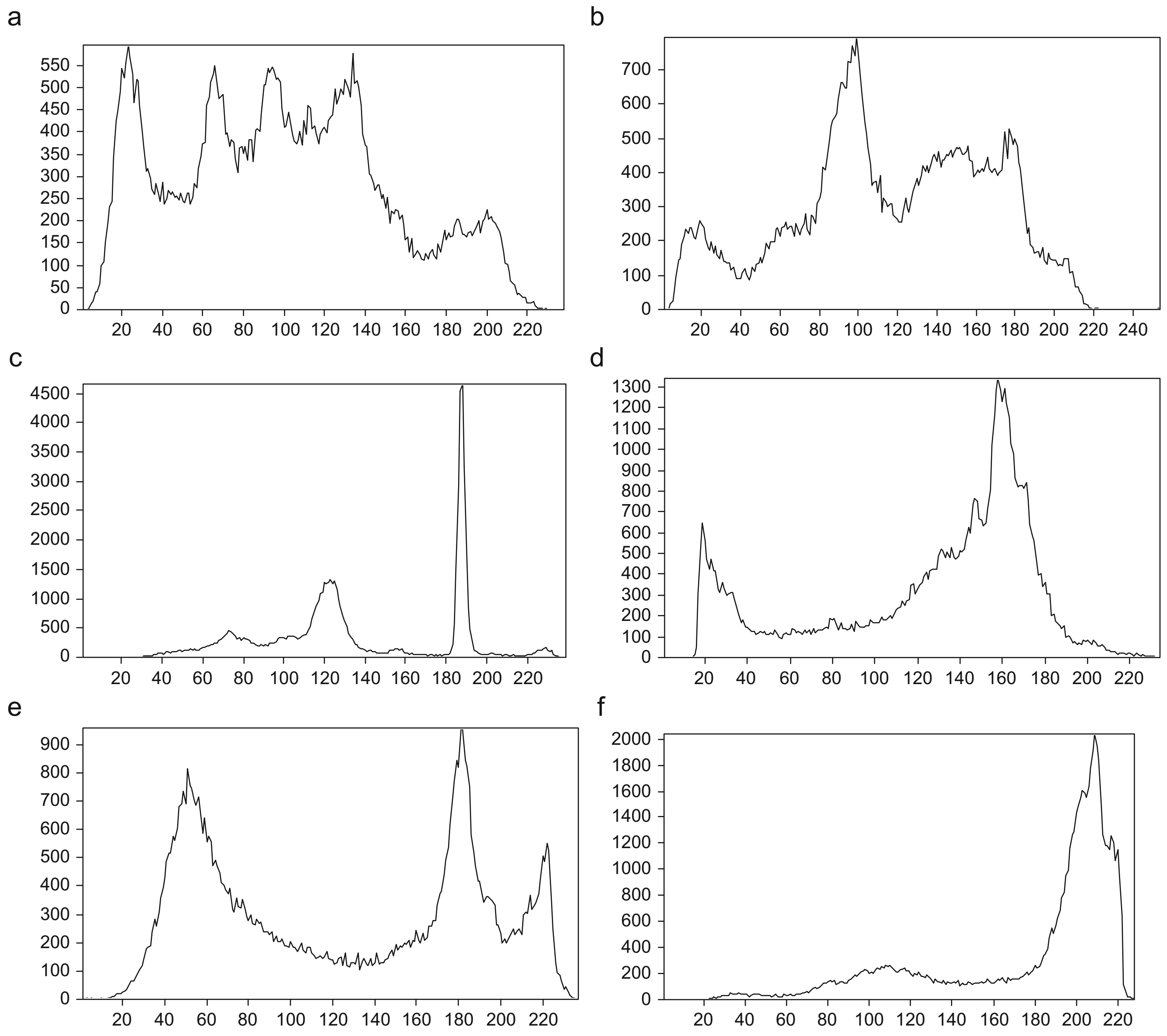

Fig. 2. Gray-level histograms of test images: (a) Lena, (b) Peppers, (c) House, (d) Boats, (e) Lake, and (f) Airplane.

Table 7
Parameters of the GA, PSO, DE, ACO, SA and TS.

AG		PSO	
Population size (N)	100	Population size (N)	100
Maximum number of iterations	1000	Maximum number of iterations	1000
Crossover probability (P_c)	0.9	Inertia weight (w)	0.5
Mutation probability (P_m)	0.1	Maximum velocity (V_{min})	-5
Mutation variance parameter (σ)	0.05	Minimum velocity (V_{min})	$+5$
		ϕ_{min}	0
		ϕ_{max}	2
DE		ACO	
Population size (N)	100	Population size (N)	100
Maximum number of iterations	5000	Maximum number of iterations	10 000
Scaling factor (f)	0.3	Probability (q_0)	0.5
Crossover constant (Cr)	0.9	Persistance of trail (ρ)	0.1
SA		TS	
Maximum number of iterations	10 000	Tabu list size (N)	100
Initial temperature ($Temp_{init}$)	100	Maximum number of iterations	10 000
Final temperature ($Temp_{final}$)	0.01	Number of neighbor solutions	20
Temperature multiplier (α)	0.9		

Table 8
Threshold values, objective function values and time processing provided by an exhaustive search for the test images.

	K	Otsu		
		Threshold values	Objective function	Time (ms)
Lena	2	102	0.3128012303	0
	3	78–145	0.1438152862	141
	4	57–106–159	0.0710375037	10313
	5	47–84–119–164	0.0428575875	598016
Peppers	2	117	0.3028234202	0
	3	67–132	0.1355182600	156
	4	62–116–160	0.0754486001	13843
	5	44–83–121–162	0.0493072659	877875
House	2	147	0.2171193157	0
	3	96–155	0.0717776906	109
	4	82–112–158	0.0523300014	8297
	5	82–112–156–205	0.0355574650	463594
Airplane	2	101	0.2132116778	0
	3	77–140	0.0998682872	109
	4	59–111–151	0.0605269320	8000
	5	56–104–141–168	0.0393339123	441500
Lake	2	125	0.1452788445	0
	3	88–155	0.0776876954	125
	4	80–140–193	0.0445815164	10016
	5	69–111–158–197	0.0280979914	586407
Boats	2	154	0.1694023170	0
	3	117–175	0.0974979570	94
	4	94–146–191	0.0609037629	6188
	5	88–132–174–203	0.0392906184	319797

Table 9
Mean values of the optimal fitness values and standard deviation over 50 runs.

	k	Lena		Peppers		House	
		Mean value	Variance	Mean value	Variance	Mean value	Variance
AG	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438152862	9.32E−17	0.1355182600	6.54E−17	0.0717776906	5.10E−17
	4	0.0710375037	5.10E−17	0.0754486001	9.27E−17	0.0523300014	2.74E−17
	5	0.0428575875	1.94E−18	0.0493072659	1.20E−17	0.0366138075	8.49E−4
PSO	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438152862	9.32E−17	0.1355182600	6.54E−17	0.0717776906	5.10E−17
	4	0.0710375037	5.10E−17	0.0754486001	9.27E−17	0.0523300014	2.74E−17
	5	0.0428575875	1.94E−18	0.0493077099	1.75E−6	0.0356448038	6.11E−4
DE	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438152862	9.32E−17	0.1355182600	6.54E−17	0.0717776906	5.10E−17
	4	0.0710375037	5.10E−17	0.0754486001	9.27E−17	0.0523300014	2.74E−17
	5	0.0428575875	1.94E−18	0.0493072659	1.20E−17	0.0355574650	1.19E−17
ACO	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438152862	9.32E−17	0.1355182600	6.54E−17	0.0717776906	5.10E−17
	4	0.0711280917	7.97E−05	0.0755734681	1.00E−4	0.0523917871	4.81E−5
	5	0.0434308107	3.16E−4	0.0499121252	3.14E−4	0.0358975545	1.86E−4
SA	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438152862	9.32E−17	0.1355182600	6.54E−17	0.0717776906	5.10E−17
	4	0.0710393038	7.12E−6	0.0754487676	1.17E−6	0.0523313206	4.67E−6
	5	0.0429009037	2.87E−5	0.0493695811	2.83E−5	0.0356106533	2.74E−5
TS	2	0.3128012303	2.95E−16	0.3028234202	1.99E−17	0.2171193157	8.99E−17
	3	0.1438369655	1.55E−5	0.1355241399	5.42E−6	0.1098835500	5.81E−2
	4	0.0710653743	1.76E−5	0.0754837403	1.75E−5	0.0536064735	1.10E−3
	5	0.0428666703	8.27E−6	0.0493248294	1.23E−5	0.0370671796	2.06E−3
	k	Boats		Lake		Airplane	
		Mean value	Variance	Mean value	Variance	Mean value	Variance
AG	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998682872	0	0.0776876954	7.05E−17	0.0974979570	9.43E−18
	4	0.0605269320	0	0.0445815164	3.19E−17	0.0609037629	3.58E−17
	5	0.0393339123	3.52E−17	0.0280981361	4.89E−7	0.0392921946	2.80E−6

Table 9 (continued)

	k	Boats		Lake		Airplane	
		Mean value	Variance	Mean value	Variance	Mean value	Variance
PSO	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998682872	0	0.0776876954	7.05E−17	0.0974979570	9.43E−18
	4	0.0605269320	0	0.0445815164	3.19E−17	0.0609037629	3.58E−17
	5	0.0393339123	3.52E−17	0.0280979914	1.06E−17	0.0392910125	1.55E−6
DE	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998682872	0	0.0776876954	7.05E−17	0.0974979570	9.43E−18
	4	0.0605269320	0	0.0445815164	3.19E−17	0.0609037629	3.58E−17
	5	0.0393339731	4.24E−7	0.0280979914	1.06E−17	0.0392906184	4.38E−17
ACO	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998682872	0	0.0776876954	7.05E−17	0.0974979570	9.43E−18
	4	0.0605869078	4.18E−5	0.0446386081	3.39E−5	0.0609348623	2.77E−5
	5	0.0398126462	2.82E−4	0.0283452932	1.41E−4	0.0396366980	1.93E−4
SA	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998682872	0	0.0776876954	7.05E−17	0.0974979570	9.43E−18
	4	0.0605279198	3.83E−6	0.0445846257	6.28E−6	0.0609045158	2.17E−6
	5	0.0393911669	3.14E−5	0.0281417618	2.51E−5	0.0393471027	3.16E−5
TS	2	0.2132116778	1.03E−16	0.1452788445	8.21E−17	0.1694023170	1.50E−16
	3	0.0998723866	5.01E−6	0.0776948986	8.12E−6	0.0975005093	1.99E−6
	4	0.0605524295	1.82E−5	0.0445916701	9.70E−6	0.0609108877	8.30E−6
	5	0.0393640708	1.14E−5	0.0281081831	7.62E−6	0.0393078092	1.24E−5

Table 10
Computational time of the GA, PSO, DE, ACO, SA and TS.

	k	Lena		Peppers		House	
		Time (ms)	Iteration number	Time (ms)	Iteration number	Time (ms)	Iteration number
AG	2	1.56	1.36	1.56	1.8	1.56	1.28
	3	7.5	9.82	7.2	10.1	6.58	10.02
	4	20	27.14	22.2	35.24	30.62	54.34
	5	74.36	102.64	85.32	131.18	609.38	945.34
PSO	2	1.24	1.42	1.56	1.4	1.88	1.3
	3	4.7	7.26	5.3	8.46	5.62	7.26
	4	9.4	15.64	9.68	15.84	11.86	15.84
	5	15.94	26.48	28.44	44.9	85.3	124.8
DE	2	1.86	1.34	1.56	1.44	1.56	1.42
	3	11.56	14	9.38	12.56	9.7	13.54
	4	99.7	116.98	116.56	133	78.14	115.4
	5	976.26	1265.2	966.88	1281.2	707.5	883.46
ACO	2	4.38	5.64	4.38	5.2	3.12	3.96
	3	610.32	927.98	860.94	1208.44	674.38	1072.62
	4	7213.74	9696.6	7511.56	9348.32	6950	9705.54
	5	8233.44	9896.96	9481.56	10000	10376.5	10000
SA	2	4.06		112.8		6.26	
	3	177.5		263.42		175.64	
	4	4081.88		4243.74		5089.68	
	5	6019.68		5716.88		5312.8	
TS	2	0.94	9.52	0.94	8.76	0.94	9.84
	3	818.42	6607.18	584.06	4452.36	786.56	6625.08
	4	1072.8	7915.1	1199.4	8829.78	1190.94	9611.48
	5	1098.12	8050.76	1361.88	9314.46	1317.5	9816.74
		Boats		Lake		Airplane	
	k	Time (ms)	Iteration number	Time (ms)	Iteration number	Time (ms)	Iteration number
AG	2	1.26	1.3	1.24	1.6	0.94	1.46
	3	6.24	10.12	7.18	11.54	5.3	10.68
	4	20.62	35.36	23.74	34.06	65.28	10.44
	5	45	76.26	100	224.06	455.98	472.92
PSO	2	1.24	1.42	1.56	1.32	1.24	1.24
	3	4.06	6.88	4.68	7.7	3.44	6.86
	4	8.74	15.26	9.36	15.98	9.06	16.12
	5	13.74	23.9	148.74	259.1	187.18	358.88

Table 10 (continued)

	k	Boats		Lake		Airplane	
		Time (ms)	Iteration number	Time (ms)	Iteration number	Time (ms)	Iteration number
DE	2	1.26	1.48	1.58	1.5	1.24	1.38
	3	8.44	13.26	9.38	13.82	9.08	15.66
	4	85	135.46	89.7	129.34	73.44	129.28
	5	970	1460	1128.74	1697.24	784.06	1332.98
ACO	2	3.14	4.16	4.06	5.56	2.82	4.64
	3	553.74	890.78	717.5	1053.08	450.62	760.42
	4	6794.06	9613.04	7036.56	9229.02	5795.64	8755.64
	5	7903.74	10000	8557.5	10000	7531.88	10000
SA	2	3.76		4.38		96.86	
	3	173.12		197.82		93.74	
	4	4276.88		3886.58		3287.5	
	5	5445.94		5660		5050.92	
TS	2	0.94	9.48	1.26	10.36	0.94	9.02
	3	491.56	3835.36	462.18	3637.64	625.96	5418.48
	4	1129.06	9006.02	933.44	6947.32	760	6337.02
	5	1271.56	9505	1230.32	8507.04	1271.24	9741.52

are comparable in term of solution quality when the threshold number is small, i.e. less than or equal to 2. While this number increases, the GA, PSO and DE provide better results than ACO, SA and TS with a little advantage to the DE.

In term of execution time, the GA, PSO and DE are most efficient than other algorithms with a great speed for the PSO.

Finally, it turned out that, in the multilevel thresholding framework, the PSO is superior compared to other meta-heuristics both respect to precision, as well as robustness of the results and runtime.

One can see through this study that, except for ACO, the population based meta-heuristics like GA, PSO and DE outperform the meta-heuristics like TS and SA, which handle a single solution. Several local optima of the objective function appear when the number of thresholds increases and TS and SA found difficulties to escape from these local optima. Further works consist in applying sophistical versions of ACO and TS to solve the multilevel thresholding problem. This comparison can also be extended to other meta-heuristics, such as the iterated local search, the variable neighborhood search, or by using a hybridization between two or several meta-heuristics.

References

Bazi, Y., Bruzzone, L., Melgani, F., 2007. Image thresholding based on the EM algorithm and the generalized Gaussian distribution. Pattern Recognition 40, 619–634.

Blum, C., Roli, A., 2001. Metaheuristic in combinatorial optimization: overview and conceptual comparison. Technical Report IRIDIA-2001-13.

Cao, L., Bao, P., Shi, Z., 2008. The strongest schema learning GA and its application to multilevel thresholding. Image and Vision Computing 146 (9–10), 387–390.

Chang, C.I., Chen, K., Wang, J., Althouse, M.L.G., 1994. A relative entropy-based approach to image thresholding. Pattern Recognition 27, 1275–1289.

Chang, Y., Yan, H., 2003. An effective multilevel thresholding approach using conditional probability entropy and genetic algorithm. In: Jin, J.S., Eaqdes, P., Feng, D.D., Yan, H. (Eds.), Conferences in Research and Practice in Information Technology. ACS, pp. 21–22.

Cheng, H.D., Chen, J.-R., Li, J., 1998. Threshold selection based on fuzzy c-partition entropy approach. Pattern Recognition 31 (7), 857–870.

Chu, C.P., Lee, D.J., Tay, J.H., 2004. Bilevel thresholding of floc images. Journal of Colloid and Interface Science 273, 483–489.

Dong, L., Yu, G., Ogunbona, P., Li, W., 2008. An efficient iterative optimization algorithm for image thresholding. Pattern Recognition Letters 29, 1311–1316.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1 (1), 53–66.

Dorigo, M., Stützle, T., 2000. The ant colony optimization metaheuristic: algorithms, applications and advances. Technical Report IRIDIA-2000-32.

Fan, S.-K.S., Lin, Y., 2007. A multi-level thresholding approach using a hybrid optimal estimation algorithm. Pattern Recognition Letters 28, 662–669.

Feng, D., Wenkang, S., Liangzhou, C., Yong, D., Zhenfu, Z., 2005. Infrared image segmentation with 2-D maximum entropy method based on particle swarm optimization (PSO). Pattern Recognition Letters 26, 597–603.

Gendreau, M., 2002. Recent advances in tabu search. In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 369–377.

Glover, F., Laguna, M., 1997. In: Tabu Search. Kluwer, Boston.

Glover, F., Kochenberger, G.A., 2003. In: Handbook on Metaheuristics. Kluwer Academic Publishers.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Gonzales-Baron, U., Butler, F., 2006. A comparison of seven thresholding techniques with the k-means clustering algorithm for measurement of bread-crumb features by digital image analysis. Journal of Food Engineering 74, 268–278.

Hammouche, K., Diaf, M., Siarry, P., 2008. A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation. Computer Vision Image Understanding 109 (2), 163–175.

Hou, Z., Hu, Q., Nowinski, W.L., 2006. On minimum variance thresholding. Pattern Recognition Letters 27, 1732–11743.

Jinsong, C., Hongqi, W., Xiaokuan, Z., 1999. Entropic thresholding method using genetic algorithm. In: Proceedings of the Geoscience and Remote Sensing Symposium, vol. 2, Hamburg, Germany, pp. 1247–1249.

Kapur, E.J.N., Sahoo, P.K., Wong, A.K.C., 1985. A new method for grey-level picture thresholding using the entropy of the histogram. Computer Vision Graphics and Image Processing 29, 273–285.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN'95), vol. IV, Perth, Australia, pp. 1942–1948.

Kim, B.-G., Shim, J.-I., Park, D.-J., 2003. Fast image segmentation based on multi-resolution analysis and wavelets. Pattern Recognition Letters 24, 2995–3006.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 71–680.

Lai, C.-C., Tseng, D.C., 2004. A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding. International Journal of Hybrid Intelligent Systems 1 (3), 143–152.

Li, C.H., Lee, C.K., 1993. Minimum cross entropy thresholding. Pattern Recognition 26 (4), 617–625.

Liao, P.-S., Chen, T.-S., Chung, P.-C., 2001. A fast algorithm for multilevel thresholding. Journal of Information Science and Engineering 17, 713–727.

Lievers, W.B., Pilkey, A.K., 2004. An evaluation of global thresholding techniques for automatic image segmentation of automotive aluminum sheet alloys. Materials Science and Engineering A381, 134–142.

Lin, K.C., 2001. Fast thresholding computation by searching for zero derivates of images between-class variance. In: 27th Annual Conference on IEEE Industrial Electronics Society, pp. 393–397.

Luo, X., Tian, J., 2000. Multi-level thresholding: maximum entropy approach using ICM. In: Proceedings of the 15th International Conference on Pattern Recognition, vol. 3, pp. 778–781.

Maitra, M., Chatterjee, A., 2008. A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. Expert Systems with Applications 34, 1341–1350.

Nakib, A., Oulhadj, H., Siarry, P., 2007. Image histogram thresholding based on multiobjective optimization. Signal Processing 87, 2516–2534.

Ng, H.-F., 2004. Automatic thresholding for defect detection. In: Proceedings of the Third International Conference on Image and Graphics, pp. 532–535.

Otsu, N., 1979. A threshold selection method for grey level histograms. IEEE Transactions on System, Man and Cybernetics SMC-9 (1), 62–66.

Paterlini, S., Krink, T., 2006. Differential evolution and particle swarm optimisation in partitional clustering. Computational Statistics & Data Analysis 50, 1220–1247.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A., 2006. Image thresholding using differential evolution. In: Proceedings of International Conference on Image Processing, Computer Vision and Pattern Recognition, Las Vegas, USA, pp. 244–249.

Riddi, S.S., Rudin, S.F., Jeshavan, H.R., 1987. An optimal multiple threshold scheme for image segmentation. IEEE Transactions on System Man and Cybernetics SMC-14, 661–665.

Sezgin, M., Tasaltin, R., 2000. A new dichotomization technique to multilevel thresholding devoted to inspection applications. Pattern Recognition Letters 21, 151–161.

Sezgin, M., Sankur, B., 2004. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13 (1), 146–156.

Storn, R., Price, K., 1995. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Technical Report TR-95-012, ICSI.

Storn, R., Price, K., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.

Stützle, T., Hoos, H.H., 1998. In: Improvements on the Ant System: Introducing the MAX–MIN Ant System. Artificial Neural Networks and Genetic Algorithms. Springer Verlag, pp. 245–249.

Tao, W., Jin, H., Liu, L., 2007. Object segmentation using ant colony optimization problem and fuzzy entropy. Pattern Recognition Letters 28, 788–796.

Tao, W.-B., Tian, J.-W., Liu, J., 2003. Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm. Pattern Recognition Letters 24, 3069–3078.

Vesterstrom, J., Thomsen, R., 2004. A comparative study of differential evolution, particle swarm optimisation and evolutionary algorithms on numerical benchmark problems, Proceedings of the Sixth Congress on Evolutionary Computation. IEE Press, Piscataway, NJ, USA, pp. 1980–1987.

Wu, B.-F., Chen, Y.-L., Chiu, C.-C., 2004. Recursive algorithms for image segmentation based on a discriminant criterion. International Journal of Signal Processing 1, 55–60.

Yang, Z.-H., Pu, Z.-B., Qi, Z.-Q., 2003. Relative entropy multilevel thresholding method based on genetic algorithm. In: IEEE International Conference on Neural Networks and Signal Processing, Nanjing, China, pp. 583–586.

Yen, J.C., Chang, F.J., Chang, S., 1995. A new criterion for automatic multilevel thresholding. IEEE Transactions on Image Processing IP-4, 370–378.

Yin, P.-Y., Chen, L.-H., 1997. A fast iterative scheme for multilevel thresholding methods. Signal Processing 60, 305–313.

Yin, P.Y., 1999. A fast scheme for optimal thresholding using genetic algorithms. Signal Processing 72, 85–95.

Yin, P.-Y., 2007. Multilevel minimum cross entropy threshold selection based on particle swarm optimization. Applied Mathematics and Computation 184 (2), 503–513.

Zahara, E., Fan, S.-K.S., Tsai, D.M., 2005. Optimal multi-thresholding using a hybrid optimization approach. Pattern Recognition Letters 26, 1085–1095.