Soft Computing (2019) 23:6921–6935
https://doi.org/10.1007/s00500-018-3331-6

Solving reverse emergence with quantum PSO application to image processing

S. Djemame[1] · M. Batouche[2] · H. Oulhadj[3] · P. Siarry[3]

Published online: 28 June 2018

Abstract
A quantum-inspired PSO (QPSO) algorithm for solving reverse emergence is proposed that is a hybridization of the particle swarm optimization (PSO) algorithm and quantum computing principles. For potential applications, we review specific image processing problems including image denoising and edge detection. Taking cellular automata as a modeling tool, an evolutionary process carried out by the QPSO algorithm attempts to extract the rules resulting in satisfactory image denoising and edge detection. Experimental results demonstrate the feasibility, the convergence and robustness of the QPSO algorithm for solving reverse emergence in the specific application of image processing.

Keywords Metaheuristics · Quantum computing · Quantum PSO · Reverse emergence · Complexity · Cellular automata · Optimization · Image processing

1 Introduction and motivations

Emergent computing is a growing field of research and is based on three principles: simplicity, large parallelism and locality. The main principle of an emergent system is that a large group of basic individuals following only simple rules can together build something more complex without the need for a global controller or centralized chain of command. The challenge with emergent systems is discovering the rules that give rise to a desired complex behavior. This is called *reverse emergence* which has become more topical in the last few years and has itself now been categorized as a research field. A disadvantage with the emergent systems cited above is that the rules had to be carefully and laboriously generated by hand. However, this is a slow and tedious process and it does not scale well to large problems. Furthermore, this is not a convenient way to build an image processing system, so a more automated approach is required (Fig. 1).

Traditionally, rigorous approaches (also called deterministic) are based on hypotheses, characterizations, deductions and experiments. They are used for solving many optimization problems and allow the finding of optimal solutions, but they are often time-consuming when solving real-world problems (i.e., problems with large dimensions, hard constrained problems, multimodal and/or time-varying problems). Besides conventional algorithms, stochastic techniques, as metaheuristics, are widely used as resolution methods for a large range of optimization problems. They are powerful and flexible search methodologies that have successfully tackled practical difficult problems. Metaheuristic algorithms seek to produce good quality solutions in reasonable computation times and are good enough for practical purposes.

Therefore, and more recently, there has been a start to automating rule generation using optimization techniques such as evolutionary algorithms (EAs), neural networks and metaheuristics.

Communicated by V. Loia.

✉ P. Siarry
siarry@upec.fr

S. Djemame
djemame@univ-setif.dz

M. Batouche
mohamed.batouche@univ-constantine2.dz

H. Oulhadj
oulhadj@upec.fr

[1] Computer Science Department, Faculty of Sciences, University of Ferhat Abbas - Setif 1, Sétif , Algeria

[2] Computer Science Department, College of NTIC, Constantine 2 University, Constantine, Algeria

[3] LISSI Laboratory, Paris-Est Creteil University, Créteil, France

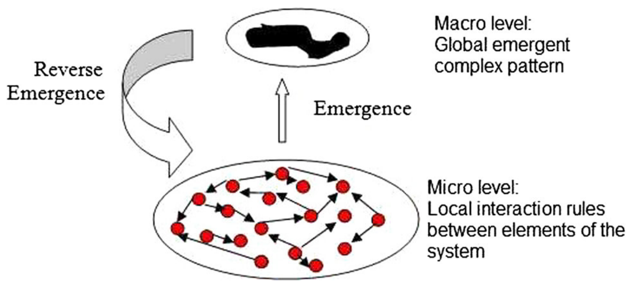
Macro level:
Global emergent complex pattern
Reverse Emergence
Emergence
Micro level:
Local interaction rules between elements of the system

Fig. 1 Complex system and the concept of reverse emergence

This introduced the term of metaheuristics which refers to general techniques that are not specific to a particular problem. Metaheuristics are approximate algorithms, and each of them has its own historical background. A metaheuristic is a set of algorithmic concepts used for defining heuristic methods that can be applied to a variety of optimization problems, without requiring substantive modifications to adapt them to particular optimization problems. Metaheuristics have successfully found high-quality solutions for a wide spectrum of "NP-hard" optimization problems.

From the literature, we have recensed a few works about searching techniques for solving reverse emergence and automating rule generation, particularly in the image processing domain. (Mitchell et al. 1996) used a standard genetic algorithm (GA) to solve the density classification task. (Sipper 1997) used evolving rules to perform thinning and gap filling in isothetic rectangles. Although the tasks were fairly simple, and the results were only mediocre, his work demonstrates that the approach is feasible. (Adorni et al. 1998) generated CAs to perform pattern classification. (Rosin 2006) used a deterministic feature selection method called the sequential floating forward search (SFFS) for training cellular automata to perform image processing tasks with a high level of performance. In Batouche et al. (2006), the authors present a solution to the problem of reverse emergence by applying a genetic algorithm as an optimization method, where the resulting evolutionary cellular automata are trained to extract contours from simple images. In Chavoya and Duthen (2006), the authors presented an algorithm for generating 2D and 3D simple binary forms from cellular automata. The transition function and the number of generations giving rise to the desired form are determined using a genetic algorithm.

Most of these approaches use genetic algorithms to search for a transition function and thereby solve reverse emergence.

However, GA presents some limitations. The main disadvantages are:

- Computationally expensive: some problems require many days or weeks to be solved.
- The learning procedure is very time-consuming.
- Blind, undirected search: it is difficult to direct a GA toward an optimal solution area if known a priori.
- GA is sensitive to initial parameters, mutation and crossover can significantly influence the search.
- It is a stochastic process, there is no guarantee that the optimal solution will be found, but there is only a high probability of finding it.

Our interest focused on another population-based metaheuristic which is particle swarm optimization.

PSO and GA share many similarities. Both techniques begin with a randomly generated population and utilize a fitness value to evaluate the population individuals. They update the population and search for the optimum with stochastic techniques.

The main difference between PSO and GA is that PSO has no evolution operators such as crossover and mutation in GA, potential solutions fly through the problem space by following the current optimum particles. Another difference resides in autonomy. PSO comes from an agent-oriented paradigm, but genes are not agent-like. Particles are semiautonomous agents which see the status of each other and decide to change their status toward the best-observed particle in their locality. On the contrary, genes do not have any abilities to sense their surrounding environment. Although there are discrete variants of PSO, it emerged as a continuous optimization approach, while GA has mainly been used for discrete optimization.

In many papers, authors claim that PSO outperforms GA when applied to various problems and PSO yields better quality and faster performance compared to GA. PSO has many advantages in comparison with GA, which can be summarized as follows:

(1) PSO is easiest in implementation.
(2) PSO has few parameters to adjust.
(3) PSO does not need evolution operators. The potential solutions emerge very simply from the search space of the problem, moving iteratively in the direction of the current optimal solutions.
(4) PSO is faster in convergence and mostly provides better solution.

In this scope, we explored the possibilities of PSO to evolve CA rules in a prior work (Djemame and Batouche 2012). The algorithm has proven effective and got satisfactory results. The experiments were encouraging and demonstrated the feasibility, the convergence and the robustness of PSO for evolving CA rules and solving reverse emergence.

Despite the success of metaheuristics (PSO, GA, ACO, etc), it became evident that the focus on pure metaheuristics is restrictive when tackling particular optimization problems, such as real-world and large-scale optimization problems. A skilled combination of a metaheuristic with components from other metaheuristics or with other optimization algorithms, such as operations research techniques (mathematical programming), artificial intelligence techniques (constraint programming) or other research areas (quantum computing), can lead to getting much better solutions for these optimization problems.

The main motivation behind these combinations is to get better performing system that exploits and includes advantages of the combined algorithms and techniques. These advantages should be complementary to each other so that the resulting hybrid metaheuristic can benefit from them.

In this context, the QPSO algorithm was developed from research in the field of the hybridization of metaheuristics and quantum computing, which began near the end of the 1990s. The goal of this combination was to enhance the benefit from each approach by mutually inspiring one another. QPSO is combining the concepts of classical PSO and quantum mechanics to improve performance of PSO. QPSO inherits the advantages of the PSO algorithm along with further improvements. Besides, unlike PSO, QPSO needs no velocity vectors for particles and also has fewer parameters to adjust, making it easier to implement. The new algorithm (QPSO) outperforms most of the time the classic one (PSO) in convergence speed and achieves better levels for the fitness function. It also demonstrates superior search performance compared to PSO and has a strong global search ability and high efficiency. Other versions of QPSO have been developed to overcome its shortcomings, such as the cooperative QPSO algorithm (Li et al. 2012).

Despite the fact that the quantum computer does not exist yet, there has been a resurgence in interest in the properties of quantum computing without focusing on massively parallel hardware implementations, i.e., they are simulated on standard serial computers.

QPSO has been applied in many areas such as image color segmentation (Zhang et al. 2009), training network traffic prediction based on a back-propagation neural network (Wang and Liu 2009) and solving mixed integer nonlinear programming (Zhang and Xing 2010). In addition, QPSO has been successfully coupled with other metaheuristics, such as the firefly algorithm, for solving discrete optimization problems (Zouache et al. 2016).

In Batouche et al. (2009), the authors proposed the use of the quantum-inspired evolutionary (QEA) algorithm for training CAs to perform image denoising. In Laboudi and Chikhi (2009), the authors used a quantum genetic algorithm to solve the density classification problem.

Within this area, we are interested in solving the inverse problem by using the QPSO algorithm and applying it to image segmentation and image denoising. The key idea is to use cellular automata (CA) as a complex system for modeling an image and then apply QPSO as a search strategy to discover the rules or subset of rules which perform the image processing tasks most efficiently. To the best of our knowledge, no one has used QPSO for solving reverse emergence, particularly in the image processing domain.

Therefore, we provide the following contributions:

- Solving an inverse problem, which consists in specifying the transition function that allows CA to obtain a specific structure in a certain generation when initial and final configurations are known. This type of problem has been described as “extremely difficult” (Ganguly et al. 2003).
- For this purpose, we use the QPSO algorithm, which is based on the hybridization of quantum computing principles and the PSO metaheuristic. This approach has never been previously used to solve reverse emergence.
- Our method allows the efficient extraction of a subset of rules, which perform the desired effect. The use of QPSO for solving reverse emergence was found to be suitable for selecting these rules as it is simple to use and results are obtained quickly.
- The extracted rules are then applied on images in a simple way to solve a range of image processing tasks, such as denoising and edge detection. Our results were satisfactory and confirmed the robustness and the effectiveness of our approach.
- Once an appropriate rule was extracted by the QPSO algorithm, we then apply it directly to images and obtain a result with good quality in minimal time.
- The rules produced by the QPSO algorithm for edge detection and noise filtering provided reasonable results compared to other algorithms previously presented in the literature.
- The further application of this work may have interesting uses in the augmented reality domain and unsupervised segmentation.

The remaining of the paper is organized as follows: in Sect. 2, we introduce the basic concepts used in this research such as PSO and QPSO algorithms. The proposed approach is described in Sect. 3, and the algorithm development is detailed in Sect. 3.2.2. Experimental results and comparisons with related work are shown in Sect. 3.2.4 with concluding remarks in Sect. 4.

2 Basic concepts

2.1 Particle swarm optimization (PSO)

Particle swarm optimization is a population-based evolutionary optimization technique originally introduced by Kennedy and Eberhart (1995) and was inspired by the metaphor of social interaction and communication such as bird flocking and fish schooling. A PSO system simulates the knowledge evolution of a social organism in which individuals (labeled as particles) representing the candidate solutions are processed through a multi-dimensional search space where they exchange information to find an optimal solution. This method has been used to solve a range of optimization problems, including network training and function minimization (Van den Bergh and Engelbrecht 2000; Shi and Eberhart 1999). Furthermore, PSO has proven its efficiency for the resolution of multi-objective optimization problems (Wang et al. 2017). PSO has also been successfully hybridized with other optimization algorithms, like the cuckoo search algorithm for parameter optimization (Li and Yin 2016).

2.2 Quantum-behaved PSO

Inspired by quantum mechanics and the trajectory analysis of PSO (Clerc and Kennedy 2002), Sun et al. recently used a strategy based on a quantum δ potential well model to sample around the previous best points (Sun et al. 2004a). They later introduced into the algorithm the mean best position and proposed a new version of PSO, called quantum-behaved particle swarm optimization (QPSO) (Sun et al. 2004b). The iterative equation of QPSO is very different from that of PSO. Specifically, QPSO does not require velocity vectors for particles, as does PSO, and is also easier to implement because it uses fewer adjustable parameters. The QPSO algorithm has proven its ability to solve a wide range of continuous optimization problems. In addition, QPSO outperforms PSO on several aspects, such as simple evolution equations, fewer control parameters, fast convergence and simple operation among other features (Sun et al. 2004b, 2011, 2012).

In the quantum model of PSO, each particle exhibits quantum behavior, so we can only measure the probability of the particles appearing in position x from probability density function, $|\psi(x,t)|^2$, which depends on the potential field the particle lies in. The state of a particle is depicted by a wave function $\psi(x,t)$, instead of position and velocity. The dynamic behavior of the particle is significantly different from that of the particle in traditional PSO systems in that the exact values of position and velocity cannot be determined simultaneously. Solving the Schrödinger equation, we obtain the normalized probability density function Q,

$$Q(y) = |\varphi(y)|^2 = \frac{1}{L}\mathrm{e}^{-2|y|/L} \tag{1}$$

where L is the most important variable as it determines the search scope of each particle. Employing a Monte Carlo method, for $y = x - p$, we can obtain the position of the particle,

$$x = p \pm (L/2) * \ln(1/u), \quad u = rand(0,1) \tag{2}$$

where u is a random number distributed uniformly on [0, 1] and p is a stochastic point between *pbest* (the position giving the best fitness value) and *gbest* (the best among all the *pbest* in the swarm). p is the local attractor of the particle, and L is evaluated with

$$L(t) = 2 * \beta * |mbest - x(t)| \tag{3}$$

$$mbest = \frac{1}{M}\sum_{i=1}^{M} P_i = \left(\frac{1}{M}\sum_{i=1}^{M} P_{i1}, \frac{1}{M}\sum_{i=1}^{M} P_{i2}, \ldots, \frac{1}{M}\sum_{i=1}^{M} P_{iD}\right) \tag{4}$$

where *mbest* (mean best position or mainstream through point) is defined as the mean value of all the particles' best positions and M is the population size. L may be interpreted as the strength of "creativity" or "imagination" because it characterizes the knowledge-seeking scope of the particle. Therefore,with a larger value of L, it is more likely the particle finds new knowledge. The parameter β, called the contraction–expansion coefficient, is the only parameter in the QPSO algorithm. From the results of stochastic simulations, QPSO has relatively better performance by varying the value of β from 1.0 at the beginning of the search to 0.5 at the end of the search to balance the exploration and exploitation (Sun et al. 2005a, b). Therefore, Eq. (2) may be written as:

$$x(t+1) = p \pm \beta * |mbest - x(t)| * \ln(1/u) \tag{5}$$

The QPSO algorithm operates as follows:

Firstly, the QPSO process is initialized: population size M and dimension D of particles. The particles P_i are randomly initialized in the search space. For each iteration t, the mean best position *mbest* is computed, using Eq. (4). Coefficient beta is linearly decreased from 1.0 to 0.5. For each particle of the swarm, the fitness function $f(X_i)$ is calculated, and the local best position is saved in P_i. The global best position is saved in G. ϕ and u denote random numbers generated uniformly and distributed on [0, 1]. The new position X_{ij} is computed according to Eq. (5). The process is repeated until reaching stopping criterion: T, the predefined number of iterations. The procedure for implementing the QPSO is given by "Algorithm 1" (Sun et al. 2004a).

3 The proposed approach

We propose the use of the quantum-behaved particle swarm optimization (QPSO) algorithm for training cellular automata

Algorithm 1 Pseudo-code of QPSO Algorithm

```
Initialize the population size M;
Initialize $P_i$ with the corresponding initial position $X_i$;
Initialize the dimensions D of the particles;
for $t = 1$ to Maximum Iteration T do
    Compute the mean best position mbest by Eq.(4);
    Beta linearly decreases from 1.0 to 0.5;
    for i = 1 to population size M do
        if $f(X_i) < f(P_i)$ then
            $P_i = X_i$;
        end if
        $G = argmin(f(P_i))$;
        for j = 1 to D do
            $\phi = rand(0, 1)$;
            $u = rand(0, 1)$;
            $p_{ij} = \phi * P_{ij} + (1 - \phi) * G_j$;
            if $rand(0, 1) > 0.5$ then
                $X_{ij} = p_{ij} + beta * abs(mbest_j - X_{ij}) * log(1/u)$;
            else
                $X_{ij} = p_{ij} - beta * abs(mbest_j - X_{ij}) * log(1/u)$;
            end if
        end for
    end for
end for
```

(CA) to perform several standard image processing tasks, including noise filtering and image segmentation. CA is used as a modeling tool of the image, and QPSO is the search strategy which extracts the appropriate rules through reverse emergence.

CA is comprised of a grid of cells which interact locally by simple rules and evolve toward a global complex behavior (Wolfram 1984). Interactions between cells are defined only with local rules, and the set of all these rules forms the transition function of the CA.

3.1 Problem statement

In image processing, CA is used efficiently to model an image. A state of a cell is the color of the pixel. The transition rule is defined by the actual state of the cell along with the state of the neighborhood. The initial configuration of CA is the input image to be processed, and the final configuration is the output image (segmented and filtered). The CA evolves from its known initial configuration to a final global state through several iterations using a limited subset of rules. Extracting this subset from the micro-level is defined as the *reverse emergence problem.* The number of transition functions (TF) increases with the number of cell states and the size of the neighborhood. This is represented by the equation: TF $= |\sigma^{Nc}|$, where Nc $= |\sigma^{n}|$ is the number of combinations of the states of neighborhood cells and $|\sigma|$ is the number of states. For example, using a Moore neighborhood and 2-cell states, the size of the transition function is 512 and the number of possible transition functions is 2^{512}.

In the case of image processing, when dealing with binary images, using a Moore neighborhood and 2 cell states (black and white), the number of possible transition functions is 2^{512}. For grayscale images, a cell having 256 states and Moore neighborhood, the number of possible transition rules is 256^{256^9}.

This is a huge space to explore. So, for the sake of simplicity, the QPSO approach for evolving cellular automata is tested on binary images which are derived from grayscale and color images.

In general, simulating a system by cellular automata consists of creating the closest environment to that of the phenomenon studied, including initial conditions, parameters and evolution rules. This kind of problems is said to be "*direct*" because we want to find the future state of the CA when its initial configuration and the rules which govern the evolution are known. Inversely, in certain cases, it is important to determine the transition function which allows the CA to obtain a particular structure in a given generation. This kind of problem is called *an inverse problem*. Ganguly et al. (2003) distinguished different kinds of inverse problems, including

1. Specifying the transition function of the CA, which allows finding particular dynamic structures of the CA.
2. Specifying the transition function of the CA, knowing its initial and final configurations.
3. Finding the initial configuration of the CA, knowing its transition function and its final configuration.

In this study, we are interested in the second inverse problem listed above. (Ganguly et al. 2003) qualified this specific problem as "*extremely difficult*."

To address this question, we propose to solve the reverse emergence problem by the use of a hybrid quantum algorithm with a population-based metaheuristic. The goal is to extract the CA rules which perform image processing tasks, such as contour detection and noise filtering, with good quality.

3.1.1 Transition function rules

The class of CAs used here is called *totalistic CAs*. This is defined where the state of each cell is represented by a number (usually an integer value drawn from a finite set) and the value of a cell at time t depends only on the sum of the values of the cells in its neighborhood (possibly including the cell itself) at time $t - 1$ (Wolfram 2002). Totalistic CA does not take into account the position of the cells.

Cellular automata and its transition function are defined such that each cell has two states (0 or 1) and a cell is said to be "alive" if its value is equal to one or "dead" if its value is zero. The neighborhood considered is the extended Moore

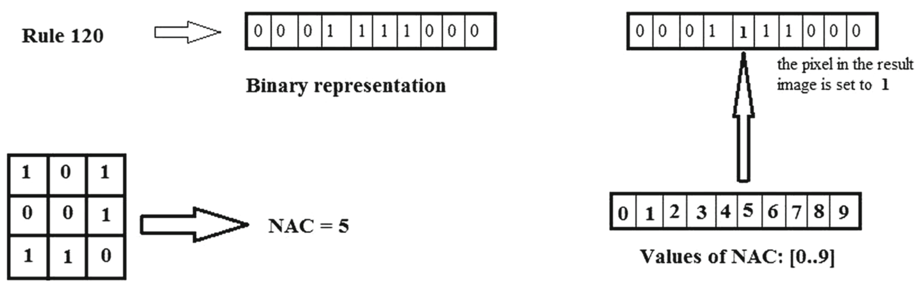

Fig. 2 Transition function rule

neighborhood (8 neighboring cells + the central cell). The total number of decision rules determines the number of possible states (1 and 0) and the number of "living" neighbors can vary between 0 and 9. This is a rather large number of possible rules to be tested, so it is important to recognize that not all the rules are considered interesting. The process will select only the rules that present more efficiency for the desired task. Let I be the original image to be processed by the evolved CA. For each pixel of the image I, the following process is executed. Let us take the central pixel P and its 8-neighbors (Moore's neighborhood). The number of alive cells (NAC) is counted. The interval of all possible values of NAC is [0.9].

On the other hand, a rule is randomly drawn and converted into the binary representation. For example, if we select rule 120, a matching is established between the binary representation of the rule and all the possible values of NAC (Fig. 2). The transition rule is expressed as follows:

if (NAC = 5), then the central pixel in the result image is set to 1.

3.1.2 Fitness functions

Whichever optimization method is used, an objective function is required, and its quality obviously has a crucial effect on the final results. The fitness function used to drive the rule selection process has an important effect on the final results. We considered three fitness functions: the structural similarity index (SSIM), root-mean-square error (RMSE) and Hamming distance (HD). The role of these functions is to measure the quality difference between the resulting image and the reference image.

For binary images, the simplest objective function is the Hamming distance. For images with more intensity values, the root-mean- square error (RMSE) between the input and target image is a straightforward measure. However, it is well known that RMSE values have limitations. In particular, given that they do not involve inter-pixel relationships, they often do not capture perceptual similarity. SSIM measures the image similarity taking into account three independent channels including luminance, contrast and structure (Wang et al. 2004). It is the well-suited measure for gray-level images.

The SSIM metric between two images x and y is defined as:

$$\mathrm{SSIM}(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{\left(\mu_x^2+\mu_y^2+C_1\right)\left(\sigma_x^2+\sigma_y^2+C_2\right)} \tag{6}$$

where $\mu_x, \mu_y, \sigma_x^2, \sigma_y^2, \sigma_{xy}$ are the mean of x, the mean of y, the variance of x, the variance of y and the covariance of x and y, respectively. Following (Wang et al. 2004), C_1 is set to $(0.01 * 255)^2$ and $C_2 = (0.03 * 255)^2$.

The Hamming distance calculates the number of different pixels between two images, where the RMSE is calculated according to Eq. (7)

$$\mathrm{RMSE}=\sqrt{\frac{1}{MN}\sum_{r=0}^{M-1}\sum_{c=0}^{N-1}[E(r,c)-O(r,c)]^2} \tag{7}$$

where $O(r, c)$ is the original image (in our case, the ground-truth image) and $E(r, c)$ is the reconstructed image (in our case, the QPSO result).

The three objective functions are used simultaneously to strengthen the quality of the results. Each fitness function reinforces the others, as far as a fitness incorporates spatial information and provides significant benefits over the others.

3.2 Quantum PSO algorithm for edge detection

3.2.1 Edge detection in image processing

Edge detection is a fundamental tool used in most image processing applications to obtain information from the frames as a precursor step to feature extraction and object segmentation. The edge detection has been used by object recognition, target tracking, segmentation, data compression and also helps for well matching, such as image reconstruction.

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convolving the image with an operator (a 2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions. There are numerous edge detection operators available, each designed to be sensitive to certain types of edges. Variables involved in the selection of an edge detection operator include edge orientation, noise environment and edge structure. The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical or diagonal edges.

There are many ways to perform edge detection. However, the majority of methods may be grouped into two categories: *Gradient-based edge detection* The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.
Laplacian-based edge detection The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp, and calculating the derivative of the image can highlight its location.

The main techniques used in the literature for edge detection are Canny, Sobel, Deriche, Prewitt, Roberts edge detectors and Laplacian of Gaussian (LoG).

3.2.2 Application of QPSO for edge detection

In this work, the validation of the proposed solution to the reverse emergence problem is done through the problem of edge detection on images. We take advantage of the calculating faculties of the CA, to transform the initial configurations defined by a numerical image lattice as discrete input data in order to find its edges. The search space is defined by all the transition rules of the CA. The evolutionary process trained by QPSO has the effect of extracting the subset of rules which leads to an edge detection with good quality. In this context, a rule is a particle of the swarm, and the best rule which gives rise to a good segmentation corresponds to the particle with the best fitness. In this algorithm, the input image and the ground-truth image are uploaded. The QPSO process is initialized by setting the number of iterations and the swarm size. At the beginning of the process, the value of parameter beta is set to 1.0; then, it is linearly decreased during the execution of the algorithm. Beta is the only parameter tuned automatically in the QPSO process. QPSO has relatively better performance by varying linearly the value of beta from 1.0 to 0.5 in order to balance between exploration and exploitation. The particles are randomly initialized in the search space. Each particle of the swarm (a rule) is converted in binary representation and applied on the input image pixel by pixel, according to the transition function defined in (Sect. 5). For each particle, an output image is obtained. The quality of edge detection is measured by evaluating three fitness functions: SSIM, Hamming distance and RMSE. The best position is identified. The mean of the best positions *mbest* is computed, using Eq. (4). For each particle, the new position X_{ij} is computed according to Eq. (5). The fitness of the new particle is evaluated, and the new rule is applied on the image. The process is repeated until reaching a predefined maximum number of iterations. At the end of the algorithm, the best rule, the best segmentation and the best fitness are displayed. “Algorithm 2” outlines how QPSO operates for edge detection.

Algorithm 2 QPSO Algorithm for edge detection

```
· Read the input image and the ground-truth image;
· Initialize swarm size and iteration number;
· Set Beta=1.0;
· Randomly initialize the particles in the search space;
· Initialize all particles fitnesses to zero;
for p=1 to swarm-size do
    · Convert the particle (rule number) to a binary representation;
    · Apply the rule on the input image pixel by pixel according to the
    transition function defined in (3.1);
    · Compute the fitness;
    · Identify the best position $P_g$;
    · Compute the mean of the best positions $mbest$, using equation
    (4);
end for·/*end step swarm initialization */
/*application of the QPSO algorithm*/
for t= 1 to Maximum iterations number T do
    · Compute the mean best position $mbest$ by eq.(4);
    · Decrease Beta linearly from 1.0 to 0.5;
    for i = 1 to population size M do
        if $f(X_i) < f(P_i)$ then
            $P_i = X_i$;
        end if
        $G = argmin(f(P_i))$;
        $\phi = rand(0, 1)$; $u = rand(0, 1)$;
        $p_{ij} = \phi * P_{ij} + (1 - \phi) * G_j$;
        if $rand(0, 1) > 0.5$ then
            $X_{ij} = int16(p_{ij} + Beta * abs(mbest(j) - X_{ij}) * log(1/u))$;
        else
            $X_{ij} = int16(p_{ij} - Beta * abs(mbest(j) - X_{ij}) * log(1/u))$;
        end if
        · Apply the new rule $X_i$ to the image;
        · Update the fitness of the new particles;
    end for
end for
· Display the best rule, best edge detection and best fitness.
END.
```

3.2.3 Computational complexity

The most efficient way to compare algorithms is knowing their time and space requirements through computational complexity theory. However, this is very difficult with stochastic algorithms, such as metaheuristics, and they tackle a class of problems called “NP-hard,” where it is not possible to obtain an exact solution in reasonable time; thus, we use methods which give an optimal solution in a rather acceptable time. For this kind of algorithms, computational complexity is of “polynomial” order.

Regarding the QPSO algorithm, the computational complexity is $O(T * M * D)$, T is the maximum number of iterations, M is the swarm size, and D is the dimension of the particles. For the developed QPSO for edge detection, the computational complexity is $O(T * M * S)$, T is the maximum number of iterations, M is the swarm size, and S is the size of the image. The dimension D of particles is taken equal to 1.

3.2.4 Experimental results

This section presents results of the application of the QPSO algorithm on several images. QPSO has proven its effectiveness in efficiently extracting the best rules providing good edge detection.

- *Image datasets* Our evaluation of QPSO compares its results for edge detection with that of other well-known edge detectors (Canny, Sobel and Prewitt) and the handmade reference contours, from the Berkeley database.
- *Parameter settings* The QPSO approach is inspired by the PSO algorithm and only has one adjustment parameter, the beta coefficient (β), which is automatically tuned in the algorithm. The coefficient beta (β) decreases linearly from 1.0 to 0.5 during the execution. The choice of this range of values is justified by recommendations throughout the QPSO literature which suggest linearly decreasing β from 1.0 to 0.5 leads to good performance in the QPSO.

The particle's position X_i represents the CA transition rule and is an integer value varying from 0 to 1023. At the start of the algorithm, it is randomly initialized. When computing the values of $mbest$, P_i, G_i and X_i, the particle position X_i is considered in its integer value, then when applying the rule on the image for extracting edges, it is transformed to its binary form. In the following steps, the values of position X_i are calculated by Eq. (5).

At the beginning of the execution, the user is asked to determine the iteration number and population size. Several experiments were conducted, with varying population sizes in the range [150, 300] and iteration numbers in the range [200, 500]. From these trials, we collected the best rules and the best fitness.

In a second set of experiments, we reduced the values with population sizes in the range [10, 30] and iteration numbers in the range [30, 50]. We concluded that the QPSO algorithm performs well even with a smaller size in population and with only a few iterations.

The best fitness values were achieved after approximately 40 generations. Beyond this threshold, no improvement in the results was noticed. So, it was not required to further increase the iterations number and population size, which only increased the execution time even when no amelioration in fitness value was observed.

- *Best packet of rules* The evolutionary process guided by the QPSO algorithm on a set of cellular automata rules demonstrated efficient extraction of the subset of rules that produced the desired final result, which suggests an optimal solution to the reverse emergence problem.

Experiments were carried on several test images from Mathworks and the database of Berkeley University. In this paper, we illustrate some examples: moon, coins from Mathworks and bird, swan, woman from Berkeley database. After dozens of experiments, we observed that four rules appeared most frequently than others and allowed extracting good edges after only one application on the input image. The rules identified were rule 56, rule 120, rule 112 and rule 116.

It is important to note that once the best rules emerge, they may be directly applied to an image quickly leading to the desired result.

- *Visual results* The results in Fig. 3(Moon) clearly demonstrate that rule 112, extracted by the QPSO algorithm, produces a satisfactory result when compared to Canny, Sobel and Prewitt. Edges are continuous, clean and fine.

In Fig. 4(Coins), the edge detection by rule 112 also shows acceptable results when compared to Canny, Sobel and Prewitt. The external contour of the coins is accurate, continuous and without noise. However, the internal details in the coins do not appear, except for one case.

In Figs. 5(Bird), 6(Swan) and 7(Woman), where the ground-truth edge is a handmade edge obtained from the Berkeley database, we can clearly see that rule 112 as is extracted by the QPSO algorithm provides good edges with a fine level of details, particularly in the body of the bird and the shadow of the swan. In the image of the Woman, rule 112 gives more accuracy and better contour outlines of the edge (nose and hands of the pullover).

Table 1 shows the best fitness values obtained for the Coins, Bird, Swan and Woman input images. The best values are highlighted in bold. For each, Canny and Sobel edge detectors are tested using the fitness functions Hamming distance, SSIM and RMSE. For Canny and Sobel, only one iteration is sufficient to collect the fitness values. The QPSO algorithm is tested with over 25 runs and 40 generations, and the best fitness values are collected in the table. The numerical results clearly show that QPSO provides performances equal or superior to Canny and Sobel.

- *Comparison with related works* Considering the result in Naidu et al. (2015), the authors used two approaches to find edges on the Lena image: a neural network with a

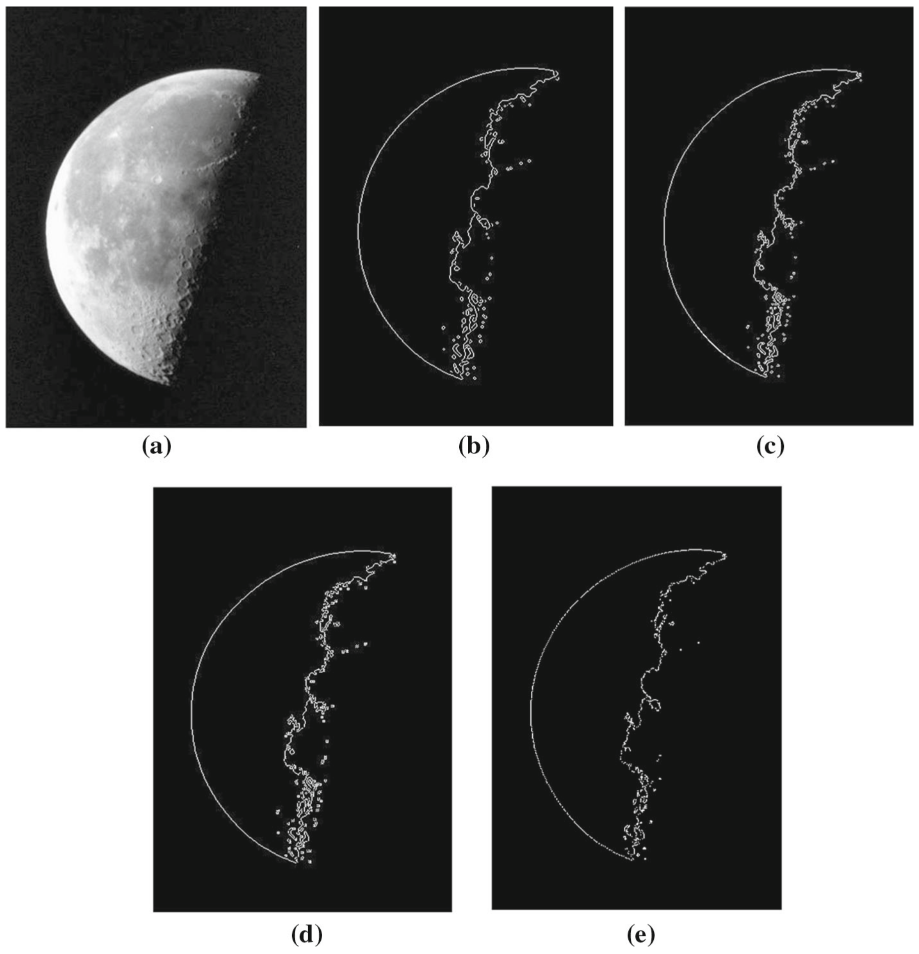

Fig. 3 Visual results of QPSO and comparison. **a** Original image; **b** Canny edge; **c** Sobel edge; **d** Prewitt edge; **e** QPSO edge rule 112

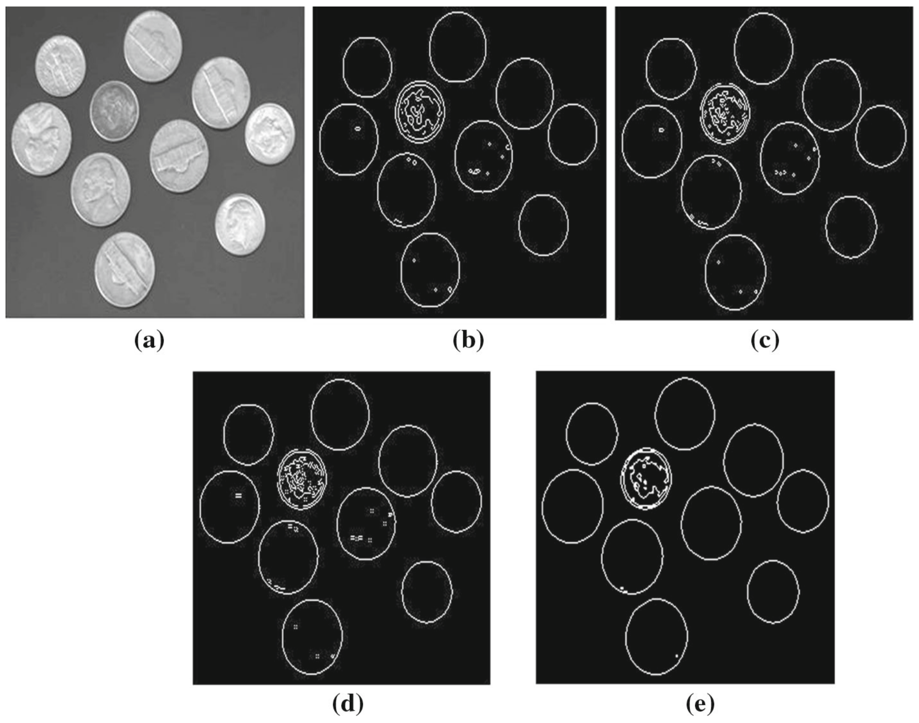

Fig. 4 Visual results of QPSO and comparison. **a** Original image; **b** Canny edge; **c** Sobel edge; **d** Prewitt edge; **e** QPSO edge rule 112

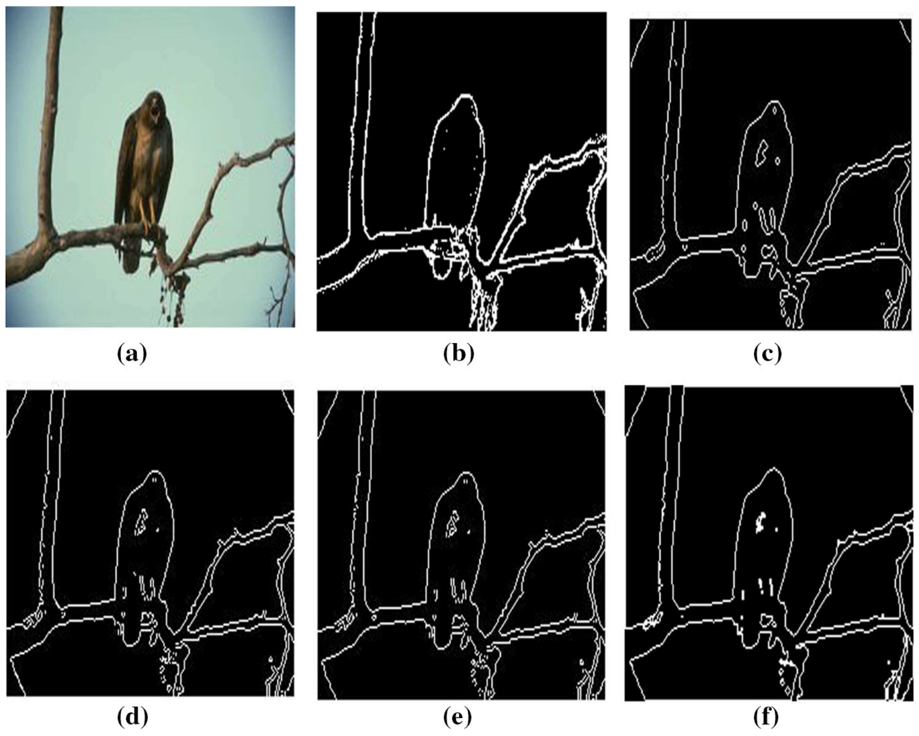

Fig. 5 Visual results of QPSO and comparison. **a** Original image; **b** Berkeley edge; **c** Canny edge; **d** Sobel edge; **e** Prewitt edge; **f** QPSO edge rule 112

general back-propagation algorithm (BP-NN) and a neural network with PSO (PSO-NN). The outputs for the Lena image are shown in Fig. 8.

These results clearly demonstrate that the QPSO algorithm produced a good contour outline for edge detection in the hat, face and mouth.

In Table 2, entropy is used to differentiate the efficiency of these three algorithms. This is a statistical measure of randomness that may be used to characterize the texture of the input image and is calculated by

$$\text{Entropy} = -\Sigma p_i \log(p_i)$$

where p_i is the probability of difference between two adjacent pixels, i.

The entropy value using QPSO is very low when compared to that of BP-NN and PSO-NN (as seen in Table 3) indicating that the noise is removed and edges are better detected with the additional information in the algorithm. Rule 68, extracted by the QPSO algorithm, produces a fewer number of edges than BP-NN and PSO-NN.

The authors Veni and Suresh (2015) used the ant colony optimization (ACO) algorithm to extract features in face images. In comparison, as is seen in Fig. 9, the QPSO algorithm can optimally extract illumination invariant features from face image as seen in how the eyes, nose and mouth appear more accurate in the QPSO result. QPSO has good detection effect on eyebrows, eyes, nose and mouth. It fairly detects the edges with improved quality and captures precisely the most important features in the face.

3.3 Quantum PSO algorithm for image denoising

3.3.1 Image denoising

A very large field of digital image processing includes image restoration which is a method of removal or reduction of degradations that are incurred during the image capturing. Degradation comes from blurring as well as noise due to the electronic and photometric sources. Noise is an unwanted signal that interferes with the original signal and degrades the visual quality of digital image. The main sources of noise in digital images are imperfect instruments, problem with data acquisition process, interference natural phenomena, transmission and compression. Image denoising forms the preprocessing step in the field of photography, technology and medical science, where somehow image has been degraded and needs to be restored before further processing. Image denoising is still a challenging problem for researchers as image denoising causes blurring and introduces artifacts. Different types of images inherit different types of noise and different noise models are used to represent different noise types. Denoising method tends to be problem specific and depends on the type of image and noise model (Patil and Jadhav 2013).

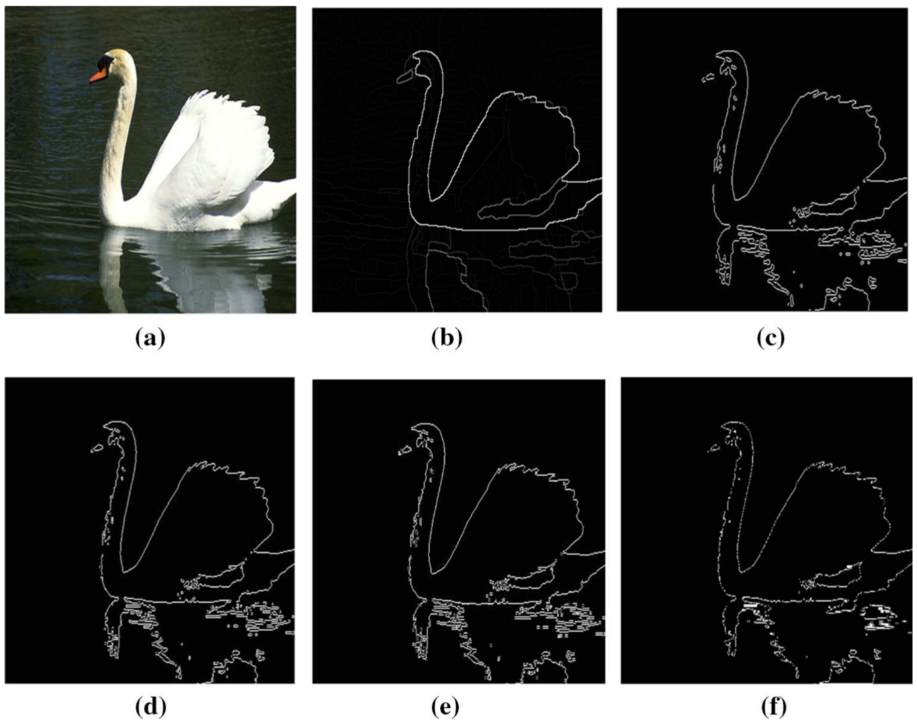
Fig. 6 Visual results of QPSO and comparison. **a** Original image; **b** Berkeley edge; **c** Canny edge; **d** Sobel edge; **e** Prewitt edge; **f** QPSO edge rule 112

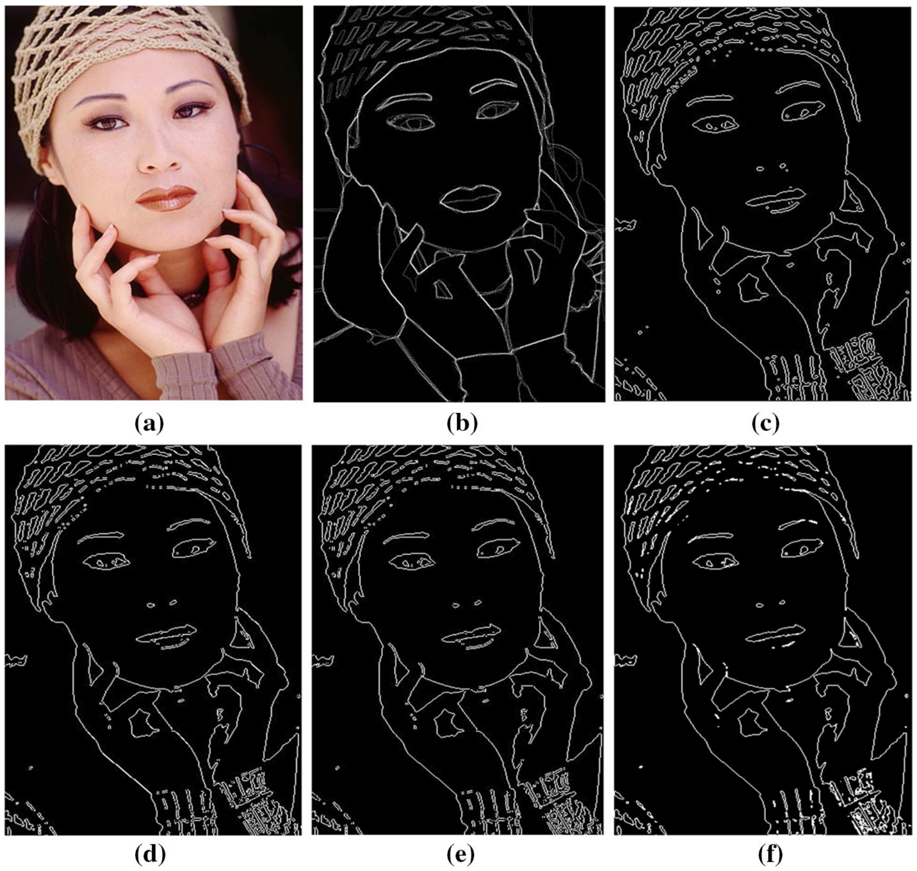
Fig. 7 Visual results of QPSO and comparison. **a** Original image; **b** Berkeley edge; **c** Canny edge; **d** Sobel edge; **e** Prewitt edge; **f** QPSO edge rule 112

Table 1 Best fitness results for 4 images

Image	QPSO			CANNY			SOBEL		
	HD	SSIM	RMSE	HD	SSIM	RMSE	HD	SSIM	RMSE
Bird	**1482**	**0.99933**	**0.194428**	1968	0.99934	0.2240	1677	0.99944	0.2068
Woman	**1322**	**0.9985**	**0.2482**	3586	0.9716	0.2751	2136	0.9992	0.2413
Swan	**3894**	**0.9994**	**0.1580**	5476	0.9993	0.1874	4043	0.9994	0.1610
Coins	**1575**	**0.9997**	**0.1450**	2037	0.9995	0.1649	2030	0.9995	0.1646

The best values are highlighted in bold

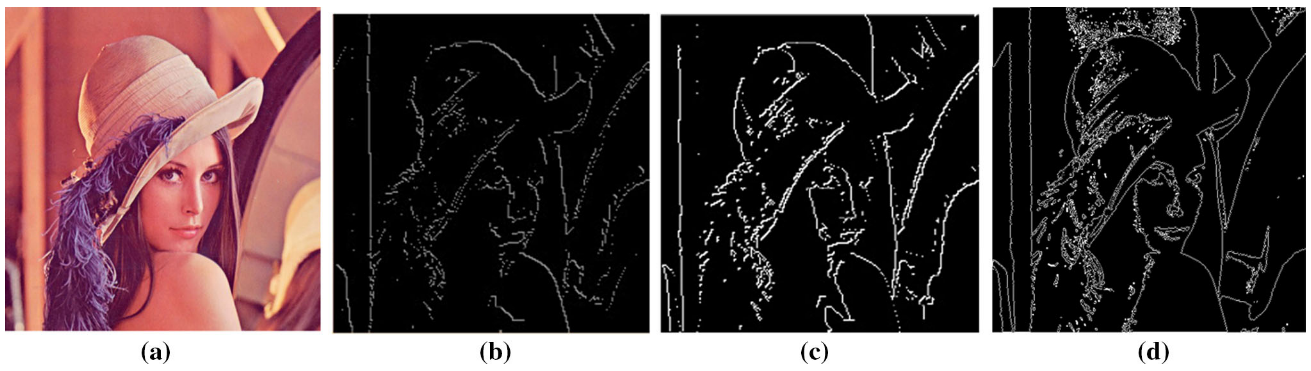

Fig. 8 Visual results of QPSO and comparison. **a** Original image; **b** edges using BP-NN; **c** edges using PSO-NN; **d** edges using QPSO rule 68

Table 2 Comparison of BP-NN, PSO-NN and QPSO edge detectors

Edge detector	Entropy	Number of edges
BP-NN	1.2335	50625
PSO-NN	0.2892	50625
QPSO	**0.2411**	**10502**

The best values are highlighted in bold

Table 3 Best fitness results for 3 images

Image	QPSO		Median filter	
	HD	SSIM	HD	SSIM
Cameraman	**367**	**0.9986**	370	**0.9986**
Pout	**300**	**0.9997**	**300**	**0.9997**
Coins	**252**	0.9997	278	**0.9999**

The best values are highlighted in bold

3.3.2 Application of QPSO for image denoising

The QPSO algorithm for edge detection was modified to deal with image denoising. Here, we present examples illustrating the effectiveness of our proposed method. Binary images are used to train the QPSO algorithm for the denoising task.

The original image is taken as the reference, and the "noisy" image is obtained by adding "Salt and Pepper 0.05" noise to this image. As before, many experiments are completed by varying the number of iterations and population size. The parameter settings are the same as for edge detection algorithm with QPSO.

Figure 10 illustrates the results of applying QPSO on the three images Cameraman, Pout and Coins, and a comparison is made with the Median Filter.

From these results, we can see that QPSO allowed for the extraction of rules which perform as good as the Median Filter. Rule 31 provides good results on the Cameraman image, while rule 63 works well on the Pout and Coins images. Noise is filtered successfully with each of these simple rules 31 and 63 that emerged from the algorithm. This suggests our proposed approach has the ability to discover the appropriate rules governing the dynamics of CA.

Table 3 shows the best fitness results for the three images Cameraman, Pout and Coins. QPSO is executed for 10 runs where the number of iterations is fixed at 50 and a population size at 30. A comparison is performed with the Median Filter, which is known to be one of the best noise removal filters available. The fitness functions used here are SSIM and Hamming distance (HD).

For the Pout image, we obtained the same values from SSIM and Hamming distance, which suggests resulting images from the Median Filter and rule 63 are very close. For the Cameraman image, the values of SSIM are the same for the Median Filter and rule 31, while the HD value shows an improvement in the rule 31 result. For the Coins image, the value of HD is better for the rule 31, while the result of SSIM is better for the Median Filter, suggesting the two results are close.

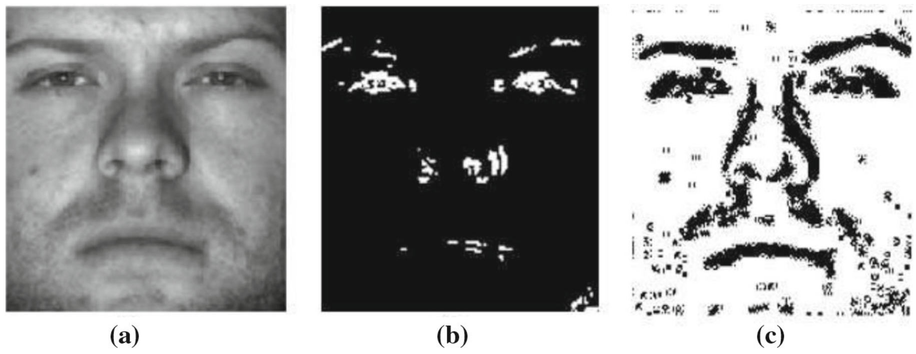
Fig. 9 Visual results of QPSO and comparison. **a** Original image; **b** Features using ACO; **c** Features using QPSO rule 21

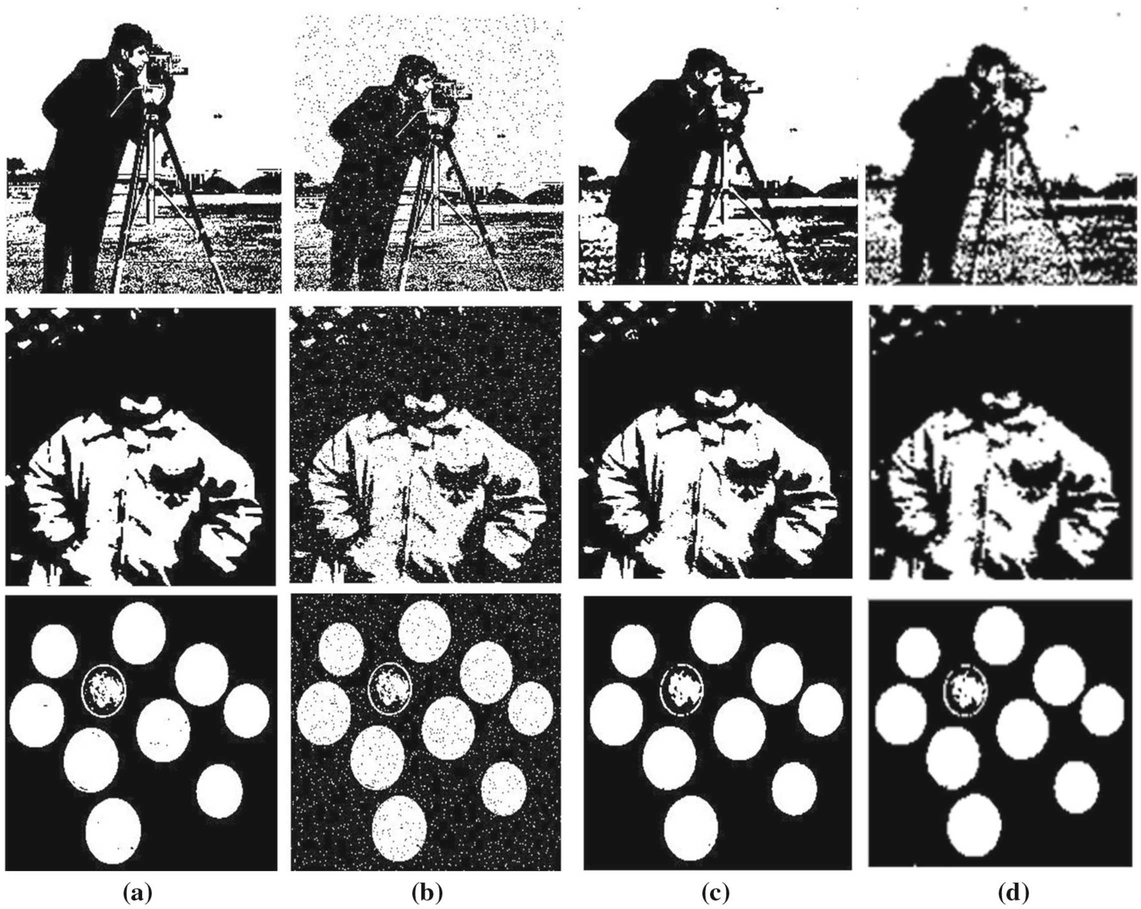
Fig. 10 Visual results of QPSO and comparison. **a** Original image; **b** Noisy image; **c** Result of median filter; **d** QPSO result (rule 63 and rule 31)

4 Conclusion

Our goal is to develop a new and efficient solution to solve the difficult problem of reverse emergence. It was cast as an optimization problem and handled using a quantum PSO algorithm for solving image processing tasks. Cellular automata is used to model an image and then evolves from an initial configuration (the original image) to final configuration (the resulting image) after applying several rules, which produce a desired processing task, either edge detection or denoising. The solution of inverse emergence is equivalent to finding or extracting the appropriate small subset of rules from a large search space that accomplish the desired task.

QPSO presents many advantages compared to classical PSO: parallelism in searching solutions within a large space, the model uses only one iterative equation, and there is only one parameter to tune. QPSO process reveals very suitable for training a cellular automata in an evolutionary process to identify the good rules which perform the desired tasks.

We demonstrated the validity and adequacy of the QPSO algorithm to select the appropriate rules within a large search space. Our iterative process allowed the extraction of simple and efficient rules. Applied to different kinds of images, the rules yielded acceptable results in comparison with edge based image segmentation algorithms using the best filtering operators, such as Canny, Sobel or Prewitt. Comparisons were made with other approaches based on neural networks and ant colony optimization. QPSO revealed very competitive, robust and performed better than the previous algorithms.

Once the rules emerge (revealed by the QPSO algorithm), we process them on images directly and obtain contours or filtered images in minimal time. This suggests a powerful flexibility and key advantage of our proposed method. We can explore additional interesting applications of this work in the augmented reality domain, particularly for the analysis of scenes, to restore missing data after variation of shooting conditions (bad weather, poor lighting, etc), as well as in medicine, digital angiography and scintigraphy where the contrast medium has disappeared. For all of these applications, we can first extract previously the best rules of segmentation from a set of sample images (supervised segmentation with learning) with which they may be used later to segment new images that are considered worse than the sample images.

For unsupervised segmentation (segmentation without learning, which does not involve reference images), we have to automatically find the best rule that provides the best segmentation. For future work, we plan to study the automatization of the rule choice in order to determine which rule should be used for which kind of images.

This study represents a step toward the use of quantum metaheuristics to overcome the obstacles of reverse emergence.

For example, further work will focus on improving the QPSO algorithm for solving reverse emergence by introducing a mutation operator and a Cauchy distribution to ensure a better exploration of the search space and to increase the global search ability. Furthermore, an interesting issue to this work will be the use of generalized island models for optimization. For example, combining several optimization algorithms, such as GA, PSO and QPSO and processing in parallel, may provide further improvements to the optimization process.

Another interesting issue is testing larger neighborhoods for the modeling cellular automata, for example, a $5*5$ neighborhood, and its impact on the optimization of search space.

Possible future research direction could be extended to test QPSO with more complex images, such as color images. This is still a challenging problem, since the search space will drastically increase.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

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

References

Adorni G, Bergenti F, Cagnoni S (1998) A cellular-programming approach to pattern classification. In: European conference on genetic programming. Springer, New York, pp 142–150

Batouche M, Meshoul S, Al Hussaini A (2009) Image processing using quantum computing and reverse emergence. Int J Nano Biomater 2:136–142

Batouche M, Meshoul S, Abbassene A (2006) Advances in applied artificial intelligence. In: Chapter on solving edge detection by emergence. Springer, Berlin, pp 800–808

Chavoya A, Duthen Y (2006) Evolving cellular automata for 2D form generation. In: Proceedings of the ninth international conference on computer graphics and artificial intelligence GECCO'06, Seattle, pp 129–137

Clerc M, Kennedy J (2002) The particle swarm: explosion, stability and convergence in a multi-dimensional complex space. IEEE Trans Evolut Comput 6:58–73

Djemame S, Batouche M (2012) Combining cellular automata and particle swarm optimization for edge detection. Int J Comput Appl 57(14):16–22

Ganguly N, Sikdar BK, Deutsch A, Canright G, Chaudhuri P (2003) A survey on cellular automata. In: Technical report, Centre for high performance computing, Dresden University of Technology

Kennedy J, Eberhart RC (1995) Particle Swarm Optimization. In: Proceedings of international conference on neural networks, Perth, Australia, pp 1942–1948

Laboudi Z, Chikhi S (2009) Evolving cellular automata by parallel quantum genetic algorithm. In: First international conference on networked digital technologies, 2009. NDT'09. IEEE, pp 309–314

Li X, Yin M (2016) A particle swarm inspired cuckoo search algorithm for real parameter optimization. Soft Comput 20(4):1389–1413

Li Y, Xiang R, Jiao L, Liu R (2012) An improved cooperative quantum-behaved particle swarm optimization. Soft Comput 16(6):1061–1069

Mitchell M, Crutchfield JP, Das R, et al (1996) Evolving cellular automata with genetic algorithms: a review of recent work. In: Proceedings of the first international conference on evolutionary computation and its applications (EvCA?96). Moscow

Naidu DL, Rao CS, Satapathy S (2015) A hybrid approach for image edge detection using neural network and particle swarm optimization. In: Advances in intelligent systems and computing. Springer, New York

Patil J, Jadhav S (2013) A comparative study of image denoising techniques. Int J Innov Res Sci Eng Technol 2(3):787–794

Rosin PA (2006) Training cellular automata for image processing. IEEE Trans Image Process 15(7):2076–2087

Shi Y, Eberhart RC (1999) Empirical study of Particle Swarm Optimization. In: Proceedings of congress evolutionary computation, Washington, pp 1927–1930

Sipper M (1997) The evolution of parallel cellular machines: toward evolware. Biosystems 42:29–43

Sun J, Fang W, Palade V, Wua X, Xu W (2011) Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point. Appl Math Comput 218:3763–3775

Sun J, Fang W, Wu X, Palade V, Xu W (2012) Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection. Evol Comput 20(3):349–393

Sun J, Feng B, Xu W (2004) Particle Swarm Optimization with particles having quantum behavior. In: Proceedings of IEEE congress on evolutionary computation, Portland, pp 325–331

Sun J, Wenbo X, Bin F (2005) Adaptive parameter control for Quantum-behaved Particle Swarm Optimization on individual level. In: Proceedings of IEEE conference on systems, man and cybernetics, Hawaii, pp 3049–3054

Sun J, Xu W, Feng B (2004) A global search strategy of Quantum-behaved Particle Swarm Optimization. In: Proceedings of IEEE conference on cybernetics and intelligent systems, Singapore, pp 111–116

Sun J, Xu W, Liu J (2005) Parameter selection of Quantum-behaved Particle Swarm Optimization. In: Advances in natural computation. Springer, Berlin, pp 543–552

Van den Bergh E, Engelbrecht AP (2000) Cooperative learning in neural networks using Particle Swarm Optimizers. South Afr Comput J 26:84–90

Veni SH Krishna, Suresh L Padma (2015) An analysis of various edge detection techniques on illuminant variant images. In: Advances in intelligent systems and computing, vol 325, Springer, Berlin

Wang P, Liu Y (2009) Network traffic prediction based on BP neural network trained by improved QPSO. Appl Res Comput 26(1):299–301

Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612

Wang D, Tan D, Liu L (2017) Particle Swarm Optimization algorithm: an overview. Soft Computing, pp 1–22

Wolfram S (1984) Universality and complexity in cellular automata, Physica 10D. Elsevier, New York

Wolfram S (2002) A new kind of science. Wolfram Media, Champaign

Zhang L, Xing Z (2010) Quantum-behaved Particle Swarm Optimization for mixed-integer nonlinear programming. Comput Eng Appl 9:49–50

Zhang H, Ming L, Zhang Y, Long H (2009) Image color segmentation based on Quantum-behaved Particle Swarm Optimization data clustering. Control Autom 25:304–305

Zouache D, Nouioua F, Moussaoui A (2016) Quantum-inspired firefly algorithm with Particle Swarm Optimization for discrete optimization problems. Soft Comput 20(7):2781–2799