On Solving Edge Detection by Emergence

M. Batouche, S. Meshoul, and A. Abbassene

PRAI Group, LIRE Laboratory,
Mentouri University of Constantine
{batouche, meshoul, abbassene}@wissal.dz

Abstract. Emergence is the process of deriving some new and coherent structures, patterns and properties in a complex system. Emergent phenomena occur due to interactions (non-linear and distributed) between the elements of a system over time. An important aspect concerning the emergent phenomena is that they are observable on a macroscopic level, whereas they are produced by the interaction of the elements of the system on a microscopic level. In this paper, we attempt to grab some emergence and complexity principles in order to apply them for problem solving. As an application, we consider the edge detection problem a key task in image analysis. Problem solving by emergence consists in discovering the local interaction rules, which will be able to produce a global solution to the problem that the system faces. More clearly, it consists in finding the local rules which will have some awaited and adequate global behavior, to solve a given problem. This approach relies on evolving cellular automata using a genetic algorithm. The aim is to find automatically the rules that allow solving the edge detection problem by emergence. For the sake of simplicity and convenience, the proposed method was tested on a set of binary images,. Very promising results have been obtained.

1 Introduction

Many systems in nature produce complex patterns, which emerge from the local interactions of relatively simple individual components. Notably this type of emergent pattern formation often occurs without the existence of a central control [4]. Such systems consist of many components, with local interactions only and no central control. Examples of emergent pattern formation in such systems include the foraging and nest-building behavior of social insects, spiral waves in cultures of amoebae, synchronized oscillations in the brain, etc.

To simulate the behavior of complex systems, cellular automata (CA) have been used as a powerful mathematical [5]. CA have the advantage of being easy to understand and implement. To exploit all the power of CA (and consequently all the power of complex systems), one must take into account some paramount characteristics, especially the concept of emergence [4].

Emergence is the direct result of the complexity of interactions inside the system. The elements (components) of the system interact locally, the interactions between them are simple, but the significant number of elements and the feedback loop phenomenon (duplex interaction between the system and its environment) produces a complex and interesting behavior (see figure 1).

M. Ali and R. Dapoigny (Eds.): IEA/AIE 2006, LNAI 4031, pp. 800 – 808, 2006.

These local interactions occur on a micro level. But the observed phenomena occurring at the macro level (emergent) seem to not have any relationship with local interactions (surprising emergent phenomenon). Taking CA as modeling tools, we associate the system local interactions to the CA transition rules. These rules use local information to produce the future state of each cell [3]. CA can have very elaborate behaviors and even carry out calculations. It was proven than CA is a universal machine equivalent to a Turing machine [5].

If one wants to exploit all the power of this tool, he will have to understand the emergence phenomenon. The question is: What are the rules that provide the right global system behavior? This question is known as the "Inverse Problem" (see figure 2). A possible solution of the "Inverse Problem" is to use an optimization strategy, in order to find the appropriate CA that solves a given task. The search space would be the space of local interaction rules. Search is guided by a quality measure calculated at the global level, which would reflect the adequacy of the system to the problem [1]. A possible manner to deal with this problem is to use Genetic Algorithms (GAs) search scheme. The mixture of Cellular Automata and Genetic Algorithms known as Evolving Cellular Automata (EvCA) [6] provides a powerful and multi-purpose tool.

In this work, we try to solve the edge detection problem [9] in an emergent manner. We want to find a CA that will have a global behavior corresponding to edge detection task. Edge detection is a key problem in image analysis [9]. It is typically the primary task of any automatic image understanding process. The aim is to find border pixels between dissimilar regions using an Evolutionary Cellular Automaton. The local rule of this Cellular Automaton emerges (versus handmade) from the evolution of a population of candidate cellular automata by means of a Genetic Algorithm optimization strategy. A simple and efficient CA rule for border detection is revealed by the GA. This rule is run over a cellular automaton initialized by the pixel intensities of the image to be segmented. In this way, a simple, intrinsic parallel algorithm for border detection emerges.

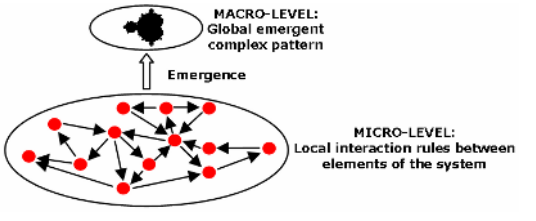

Fig. 1. Emergence of a global property from local interactions

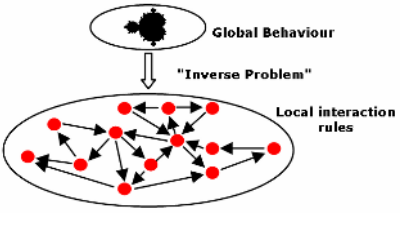

Fig. 2. The "Inverse Problem", Inversing the "Emergence Arrow"

2 Problem Formulation

A border is a frontier between two homogeneous areas of the image. A homogeneous zone is a part of the image containing pixels which exhibit similar characteristics (intensity, color, texture…). Inter-Zones contrast must be relatively strong. The border between two homogeneous areas constitutes what we call an "Edge" [2, 9]. In the "image processing" literature [9], we find many "edge detection" methods based on various algorithms. The mainstream of these methods are based on directional derivative calculations [9]. Other methods abolishing derivative calculations use preliminary knowledge concerning the nature and characteristics of the treated image, but these requirements limit the applicability of the method [2].

We propose here to construct an alternative method based on the evolution of CA population that uses simple local rules [5]. The aim is to obtain after an evolving process guided by a fitness function (edge detection quality value), a cellular automata able to segment an image and to detect significant edges by emergence.

3 Cellular Automata Principles

The Cellular Automata were invented by Stanislaw Ulam (1909-1984) and John von Neumann (1903-1957) at the end of the Forties in Los Alamos National Laboratory (United States) [5]. A CA is a D dimensional grid (lattice) composed of agents called cells. These cells interact and evolve in time, changing their states in a discontinuous way. The transition rules of each automaton (cell) are local and simple. They take into account only the state of the neighbor cells [3]. A Cellular Automaton is a discrete and dynamical system made up of a regular lattice of cells. Formally, it is a quintuplet $A = (L, D, S, NR, R)$. L is the D dimensional cellular lattice. At each discrete moment t, each cell can have only one state of the finite state set S. At each time step, all the cells update their states simultaneously. The transition process is driven by a local update rule R, which is the same for all cells (1). This rule takes the local neighborhood N of the cell (states of neighbor cells), and produce the future state of the cell [3].

$$S_{i,j}^{t+1} = R\left(N\left(S_{i,j}^{t}\right)\right). \tag{1}$$

4 Genetic Algorithms Principles

Genetic Algorithms GA [6] are stochastic search methods inspired by the "evolution species theory". The GA maintains a population of potential solutions, represented by chromosomes (genotype). Each chromosome constitutes the genetic code of an individual (phenotype) of that population [1]. The survival of an individual is conditioned by its adaptation to the environment (survival of the fittest – according to its adequacy with the role given to him). A fitness value is affected to each individual, which quantifies its capacity of adaptation. This fitness is proportional to the percentage of accomplishing a task (partially solving a problem). The fitness function

```
P0 ← Initial Population;
Calculate_individual_fitness(P0);
WHILE not_end DO
P_tmp ← P0 ∪ mutate(P0) ∪ crossover(P0);
Calculate_individual_fitness(P_tmp);
P0    ← select_best_fitness_from(P_tmp);
END
```

Fig. 3. Simplified Genetic Algorithm

is the individual selection criterion. The theory of the evolution suggests that some ratio of present generation (ancestor) can be selected to build the next generation (offspring). The new population is generated by the application of the genetic operators (crossover and mutation) [6]. An evolution process is applied trying to attain the optimal solution (or an approximate one). This solution will appear by combination and selection of individuals of the generations of candidate solutions. This process of optimization is frequently used to find approximate satisfactory solutions. Figure 3 shows the general scheme of a genetic algorithm.

5 Evolving Cellular Automata (EvCA)

The Evolving Cellular Automata (EvCA) is a methodology that takes advantage from the notion of evolutionary strategies. We apply this method as an optimization scheme, in the process of searching the best CA rule that is able to perform some computational task. An initial CA population, composed of chromosomes coding the local transition rule of each CA of the population, is randomly generated. After which, the GA evolve this initial population (Generation Zero), by applying the genetic operators: mutation and crossover. The selection of the individuals of the next generation is conditioned by the value of the fitness function [6]. This cyclic process makes the emergence of a set of interesting individuals. These individuals have a high fitness, and are thus most suited. They constitute the elite. One of them will be the chosen final solution [6]. It is important to choose well the chromosome's coding method, which corresponds to the local rule of the CA. Another serious point is the choice of the mutation and crossover operators. But the keystone remains the choice of the fitness function, which completely conditions the behaviour of the GA and the convergence of the evolution process towards an acceptable solution.

6 The Proposed Method

The proposed approach takes advantage of the capacity of Cellular Automata (CA) for generating and transforming a wide variety of patterns to implement the computational task of edge detection. Performing a computation with a CA means

that the input to the computation (image to segment) is encoded as the initial configuration; the output (segmented image) is decoded from the configuration reached by the dynamics some later time step. The intermediate dynamical steps, that transform the input to the output, are taken as the steps in the computation. The computation emerges from the CA rule being obeyed locally by each cell [3]. Since it is very difficult to specify by hand a priori the particular dynamical rule suitable for a desired computation, an evolutionary process is applied in their search [4, 7, 8]. This calculation (edge detection task) must be an emerging phenomenon, produced by the execution of simple local rules by each cell of CA [3]. The proposed method can be summarized as follows. An Evolutionary Algorithm (Genetic Algorithm) is applied in the search for adequate CA dynamical rules that perform edge detection over a set of reference images and their segmented counterparts. The fitness function is a measure of quality of the edge detection process that results from the application of the CA on the set of reference images. The evolutionary process starts from a random initial population of CA rules. The result is a set of evolved local transition rules defining a CA edge detector.

6.1 The Rule Format

We represent the CA rule performing the edge detection task, as the concatenation of immediate neighborhood cells states of the cell to be updated (including this). And we add the next state of the central cell after update [7].

This rule can be run when its neighborhood part matches with a pixel patch (of the same size) in the image. Then we update the central pixel by the "next state" part of the rule. The matching between the neighborhood and the image patch is done modulo rotational operators (Rotate (0°, 90°, 180° and 270°); Flip-Flop (Horizontal, Vertical)). The rules are said to be rotational symmetric:

Simple rules cannot represent all possible pixel patch configurations in the image. Therefore, we assemble them into rules packets. In this manner, each individual in the GA population is represented by a chromosome. The chromosome is simply the rules packet.

We can introduce a weak matching criterion, by using a similarity threshold ε:

$$\textbf{difference(rule, pixel patch)} \leq \varepsilon \tag{2}$$

To make the CA determinist we introduce the following constraint: Each rule of the rules packet must be different from other rules, modulo the rotational operators and the similarity threshold.

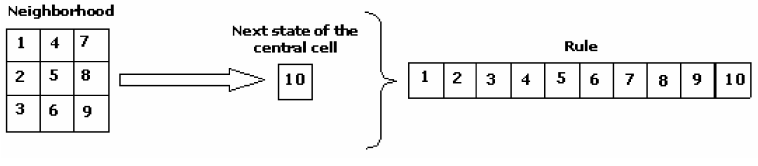

Fig. 4. CA rule formed by the neighborhood and the next state

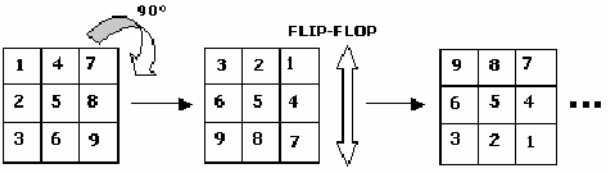

Fig. 5. Rotational equivalence

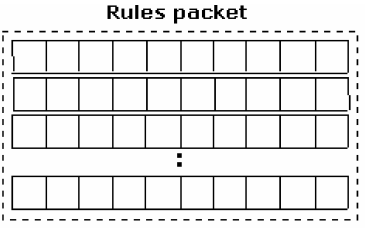

Fig. 6. Rules packet that forms an individual of the population

6.2 The Crossover Operator

Due to the chromosome (rules packet) form, we introduce two different crossover operators, a horizontal one and vertical one. They are applied in an equiprobable way.

6.2.1 Horizontal Crossover

This operator takes two parents rule packets and provides two children rule packets by interchanging some rules between them. This makes it possible to combine the best rules to obtain better individuals. This strategy keeps the cohesion and the stability of the crossover operator. The horizontal crossover operator adjusts and ameliorates the solutions obtained focussing on a local search area (exploitation).

6.2.2 Vertical Crossover

The vertical crossover operator exchanges rule parts between parent chromosomes. This allows obtaining children rules packets appreciably different from the parents. The vertical crossover operator explores a wider area of the search space. In this way, it has a comparable behaviour to a mutation operator, which diversifies the search process (exploration).

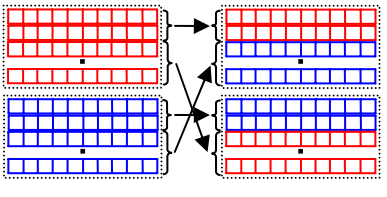

Fig. 7. Horizontal crossover operator

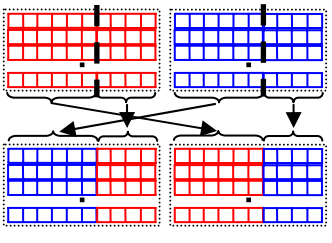

Fig. 8. Vertical crossover operator

6.3 Mutation Operator

Applying a mutation operator on a rule packet, can be done by changing the state of some randomly chosen cell in a rule of the packet according to a probability function. Of course precautions are taken to keep the integrity of the chromosome. It must always be valid and runnable, and not present a rule duplication or contradiction.

6.4 The Fitness Function and the Selection Operator

To evaluate the segmentation quality of a rules packet, we execute it on CA. Taking image pixel intensities as initial configuration. We compare the resulting pattern with the beforehand segmented image (target edge). We use the following fitness:

$$\mathbf{F = 1 / (Nb\_Bad\_Classified\_Pixels + 1)} \quad (3)$$

This formula attains 1 (best fitness) when Nb_Bad_Classified_Pixels (wrongly classified edge) approaches 0.

The selection operator randomly chooses the most suited individuals from the current population (current generation). The probability of pulling some individual is proportional to the value of its fitness. Then, we apply the crossover and mutation operators on the selected individuals in order to build the new population (next generation).

Another fitness function could be used. It quantifies more accurately the quality of the segmentation. So, let us consider the following quantities [2]:

TP : `True Positive`, correctly classified pixels;
FP : `False Positive`, incorrectly classified pixels;
FN : `False Negative`, unclassified pixels.

We calculate then the following values, **Building Detection Percentage** (**BDP**) (4), i.e. the percentage of correctly classified pixels in a particular class:

$$\mathbf{BDP = TP / (TP + FN)} \quad (4)$$

We compute the **BDP** ratio for "edge" and for "non-edge" class pixels, respectively **BDP_Edge** and **BDP_background**. The selected fitness function would be calculated as the multiplication of these two values:

$$\mathbf{F = BDP\_Edge \times BDP\_background.} \quad (5)$$

7 Experimental Results

The images of size 100x100 (figure 9) were obtained using GA on 500 generations. The size of the population was 50 packets (individuals); each packet contains 15 symmetrical rules. The individual who scored the best fitness has a value of 95,637%.

The fifteen (15) selected rules are represented as "neighborhood" ➔ "future state". The rules are symmetrical and keep the same result when applied to the rotated image.

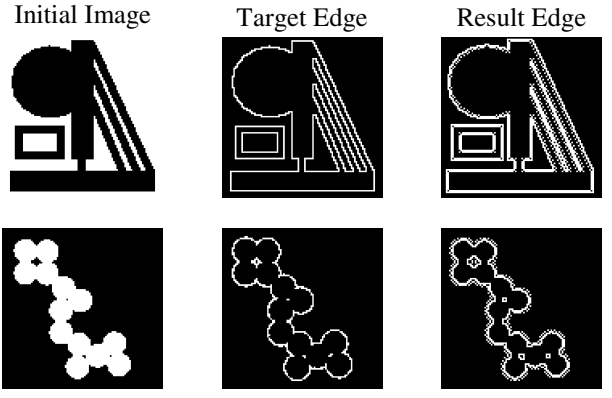

Fig. 9. Sample results from training set

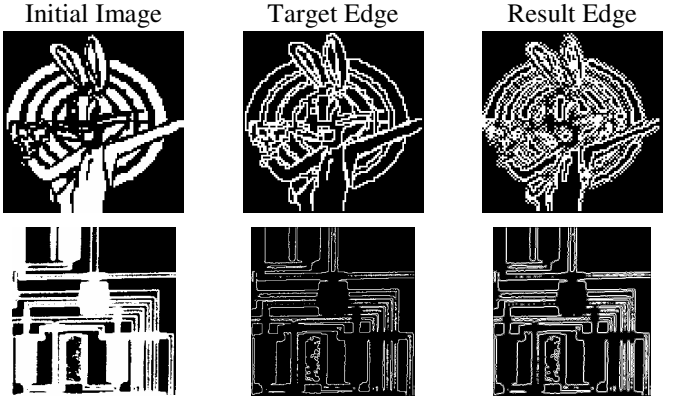

Fig. 10. Sample results from test

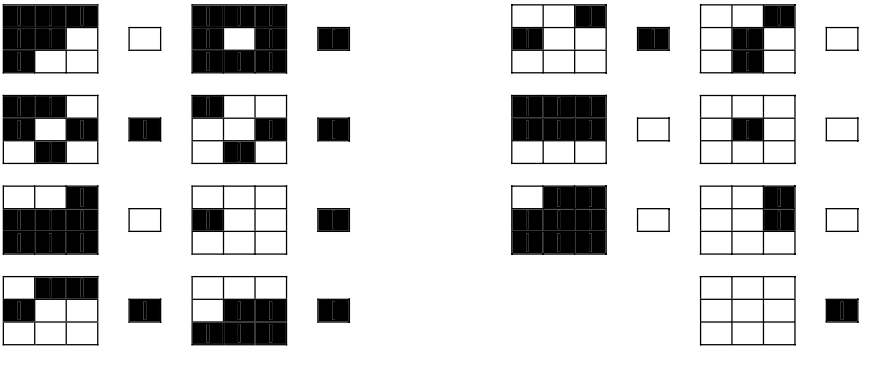

Fig. 11. GA final rules packet

8 Conclusion

The proposed methodology is a purely emergent one. It is based on the evolution and training principles. A CA was trained to solve "Edge Detection" by emergence. GAs were used to evolve cellular automata. The search process is enlightened by the optimization of a fitness function which symbolizes the segmentation quality. It is the criterion to be ameliorated during the search of a valid solution. Our future works will concern the extension of the method to more complex images such as 16 and 256 grey level images and also to color images. We could also explore the utilization of other evolutionary algorithms such as "Genetic Programming" or "Artificial Immune Systems". This approach seems promising and could be transposed to solve other general problems. Indeed, this methodology may be applied for building a powerful general "Framework", for solving broad-spectrum problems "by emergence".

Acknowledgments

This work was supported by CMEP – PROGRAMME TASSILI under Project 05 MDU 642.

References

1. Bar-Yam, Y.: Dynamics of complex systems, The Advanced Book studies in nonlinearity series, Westview Press, (2000)
2. Davis L. S.: A Survey of Edge Detection Techniques, Computer Graphics and Image Processing, 12 (1975) 248-270
3. Ganguly, N., Sikdar, B. K., Deutsch, A., Canright, G., P.P.Chaudhuri, P. P.: A Survey on Cellular Automata, Project BISON (IST-2001-38923), (2001)
4. Georgé, J. P. : Résolution de problèmes par émergence, PhD Thesis, Université Toulouse III, July 2004.
5. Langton, C. G.: Studying artificial life with cellular automata, Physica D., 22 (1986) 120-149
6. Mitchell, M., Crutchfield, J. P., Das, R.: 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, SFI) Moscow, (1996)
7. Moreno, J. A., Paletta, M. : Evolving Cellular Automata for Noise Reduction in Images, in Proceedings of CAEPIA'2001, (2001)
8. Rosin, P. L.: Training Cellular Automata for Image Processing, in proceedings of the Scandinavian Conference on Image Processing, SCIA'05, (2005) 195-204
9. Shapiro, L. G., Stockman, G. C.: Computer Vision, Prentice Hall inc. (2001)