Evolving Cellular Automata to Grow Microstructures

David Basanta[1], Peter J. Bentley[2], Mark A. Miodownik[1] and Elizabeth A. Holm[3]

[1]Department of Mechanical Engineering, Kings College London, The Strand, London
{david.basanta, mark.miodownik}@kcl.ac.uk
[2]Department of Computer Science, University College London,,Gower St. London
p.bentley@cs.ucl.ac.uk
[3]Sandia National Laboratories, Albuquerque, New Mexico, USA
eaholm@sandia.gov

Abstract. The properties of engineering structures such as cars, cell phones or bridges rely on materials and on the properties of these materials. The study of these properties, which are determined by the internal architecture of the material or microstructure, has significant importance for material scientists. One of the things needed for this study is a tool that can create microstructural patterns. In this paper we explore the use of a genetic algorithm to evolve the rules of an effector automata to recreate these microstructural patterns.

Evolving Cellular Automata to Grow Microstructures

Abstract. The properties of engineering structures such as cars, cell phones or bridges rely on materials and on the properties of these materials. The study of these properties, which are determined by the internal architecture of the material or microstructure, has significant importance for material scientists. One of the things needed for this study is a tool that can create microstructural patterns. In this paper we explore the use of a genetic algorithm to evolve the rules of an effector automata to recreate these microstructural patterns.

1 Introduction

Materials science is the study of materials and their properties. Most materials are crystalline and so have a regular periodic crystal structure at the atomic scale. But a brick or a spanner are not made of one single perfect crystal. They are made from billions of small crystals. Each crystal is not the same shape or composition, and they often exist as a nested structure in which one crystal will contain many types of smaller crystal. This complicated multiscale pattern is called the microstructure. The microstructure has long been recognised as the origin for the wealth of different types of materials that we see around us. The term is not limited to describe the structure of crystalline materials, it is used to describe the internal patterns of all materials.

What makes a wine glass brittle and an optical fibre strong is not explained in terms of the strength of atomic bonds, which are chemically identical. The extraordinary difference in properties is entirely attributable to the different types of microstructure in the two products [1]. Changing the microstructure changes the properties. This is the origin of 'heat treatment' in metallurgy. A sword can be made brittle and weak, or strong and tough by simply putting it in a fire. The heat treatment produces a different microstructure. The materials for jet engines, silicon chips, batteries, and even concrete building foundations are engineered to have specific microstructures to give specific properties. The study of these microstructures, these patterns, is therefore a major area of research. Figure 3 (left) illustrates the microstructure of a jet engine alloy, showing small spherical alumina crystals embedded in a bigger nickel-aluminium crystal. The size and dispersion of the alumina crystals is one of the major controlling factors of the high temperature strength of the material.

There is a major effort to design new materials with special properties [2]. This involves investigating new types of microstructure and using computer simulation to

test the properties. One of the first steps along this road is to develop a tool that can create microstructural pattern [3]. This paper deals with this issue and introduces a new method to create microstructural patterns using a genetic algorithm (GA) to evolve a type of cellular automaton (CA).

2 Background

2.1 Introduction to Microstructures

All materials have internal architectures that determine their properties, known as *microstructures.* Details of specific microstructures are normally obtained using one of the different microscopy techniques available. Among the features that can be found in a microstructure are grains, grain boundaries and phases.

Most materials are composed of various crystals or grains and are separated by grain boundaries. Inside these grains, there can be more crystals or particles of different types and orientations. Each different type of particle constitutes a phase in the grain [4].

2.2 Cellular Automata

Cellular automata (CA) are mathematical tools that can be used to model physical systems. A CA consists of cells usually arranged in a square grid and communicating with each other [5]. There are two features that differentiate CAs: the initial configuration (IC) and the rule set. The IC determines the dimensions of the lattice and the state of each automaton at the initial stage. The rule set is the set of rules that will be applied to the lattice each iteration, starting from the IC. The initial state of the lattice will change following the dictate of these rules.

CA are being used in a number of different fields and applications ranging from materials science [6] to the evolution of “artificial brains” [7]. In most of these applications, the rules of the CA are designed by humans but there is a growing interest in using evolutionary computing techniques to automate rule generation. Some examples of this trend are the use of GAs [8] and GP [9] to evolve CAs for the density classification problem or the use of a “Selfish Gene” algorithm to evolve a CA that can test digital circuits [10]. Closer to the task of evolving shapes, work performed by Kumar and Bentley [11] used GAs to evolve and compare different developmental processes, including CAs, by evolving specific bitmaps. Also relevant to this research is the evolution of CA and pheromonal agent systems to explore pattern formation described in [12].

2.3 Effector Automata

Effector automata (EfA) are a type of CA introduced in [13] in which the cells of the lattice represent only locations in the space and automata are entities that can occupy

those cells. The rules of an EfA are different from those in a standard CA (see Fig. 1). In a standard CA the rules specify the state of the automaton in the next time step whereas in an EfA the rules specify the location of a moving automaton in the next time step. Regardless of the differences, the EfA retains most of the properties of CA like strictly local interactions among automata, and a high degree of parallelism.

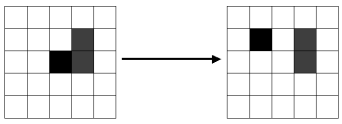

Fig. 1. The first 2D lattice represents the EfA at time step t, the second lattice represents how the darker cell (automaton) would move at time step t+1 if the rule "`if (neighbours<=3) then move`" was applied. The direction of movement is not specified in the rule; the EfA will choose randomly a direction in which the automaton will not collide with any of its neighbours.

3 Using GAs to Grow Microstructures with Effector Automata

3.1 Effector Automata Rules

The rules of the EfA used in this work can fall in one of two types:

```
If (number_of_neighbours ≥ threshold) then move, else stay.
If (number_of_neighbours < threshold) then move, else stay.
```

The neighbours of an automaton are the automata that occupy contiguous cells. In a 2D EfA, an automaton may have a maximum of eight neighbours so there are sixteen possible rules, eight of the first type and eight of the second. An EfA was chosen instead of the standard CA for several reasons. First, the number of automata to deal with is reduced from all the cells in the lattice to all the cells that actually contain automata. Second, the GA doesn't have to spend time finding the right value for the number of active automata in the lattice. These two things make EfA more computationally efficient than standard CA. The effects of using different rules with two ICs can be seen in Fig. 2.

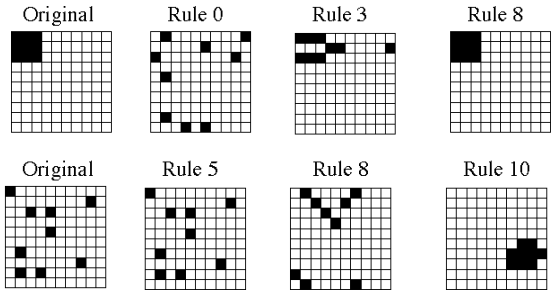

Fig. 2. The two lattices in the first column represent the ICs of the CA. The rest of the lattices represent the IC to their left after being iterating it for a number of times with different rules. Some rules, e.g., rule 1 and rule 10, can produce large changes to the IC.

3.2 Genetic Algorithm

In this work a genetic algorithm grows two-phase single-crystal microstructures. This kind of microstructure can be represented with lattices with cells in one of two different states, Fig. 3.

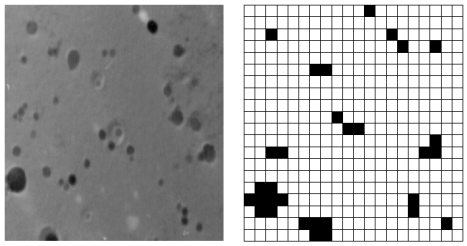

Fig. 3. Part the microstructure of an ODS ferritic superalloy (left). The discretisation into a 20x20 lattice that will be used as a target by the GA (right).

The GA reconstructs microstructures in 2D but the method described in this paper works in exactly the same fashion regardless of the dimensionality of the microstructure being reconstructed. It uses the same information needed to reconstruct microstructures in 3D. The only difference between 2D and 3D is the size of the search space. To enable easier experimentation, this work focuses on 2D reconstruction.

A fairly standard generational GA with elitism is used. Selection is performed through tournaments with two contestants per tournament; selected candidates are combined using a two point crossover operator and each chromosome has a probability of 0.1 of being mutated. There is one slot for elitism so the best candidate of each generation passes directly to the next generation without modifications.

The GA evolves a population of EfA rule sets. Every gene in the genome represents one of the 16 possible rules and is coded as an integer in the range 0-15. The length of the genome is fixed by the user, the bigger this number, the more rules in the rule set.

Each of these rule sets, together with an IC, is used to create the EfA. The IC is common to all EfA created during the run and is randomly generated. To construct an EfA using the rule set, every automaton (filled cell in the IC) is assigned one rule randomly chosen from the rule set. Once an automaton is assigned a rule, it follows that rule alone for the duration of the execution of the EfA. After iterating each EfA for a fixed number of times, the resulting lattice is input to the fitness function.

3.3 Fitness function

The fitness function examines the distribution of automata on the lattice and compares it to the distribution of the target provided to the GA. Fitter individuals have distributions of automata that match the target more closely.

To obtain the distribution of automata along the lattice, a distance is obtained for every pair of automata in the lattice. Using this information a distribution of automata

and distances is built for every automaton. The following two-point correlation function was used:

$$f(d) = \frac{1}{N^2} \sum_{i=1}^{i=N} n_d \quad (1)$$

where d is the correlation distance, N is the total number of automata in the system and n_d is the number of automata at distance d from automaton i. This distribution relates a given automaton with any other automaton in the lattice and the Euclidean distance that separates them. The distribution obtained as a result of averaging the distributions of all automata in the lattice is used to compare different lattices with the target lattice provided to the GA.

4 Experiments

The objective of these experiments is to see if it is possible to reconstruct shapes of different levels of complexity using EfA whose rules have been evolved by a GA.

In addition, the impact of two system parameters on the GA was investigated. To measure this impact, two different experiments, one for each parameter, were devised. For each experiment, a number of different variations of the parameters were tried. Every variation of the parameters was tested with three different target lattices (see Fig. 4) and each of these tests was run ten times.

Experiment 1: *Size of rule set.* Tests evolving rule sets of sizes: 1, 2, 3, 4, 5 and 10.
Experiment 2: *Maximum number of iterations.* Tests iterating the EfA for 100, 1000 and 100000 time steps.

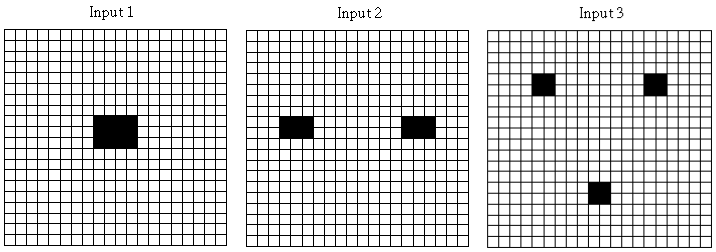

Fig. 4. The three input 2D lattices used as targets for the GA in the tests. Though the number of clusters increases in each target and so does the difficulty of finding a solution, the number of automata remains the same in all cases

The GA population size for all the experiments was 20 individuals. The GA terminated when it found a perfect match according to the fitness function, otherwise it evolved up to 500 generations and returned the best candidate found. In experiment 1 each EfA was iterated a maximum of 10000 times. In experiment 2 the size of the rule set was 5.

5 Results

5.1 Results for Experiment 1.

The first noticeable thing is that the differences in terms of fitness between the different values for this parameter are rather small (see Fig. 5). This result is probably due to an inadequate fitness function. As expected, for a simple target such as the first, the GA always finds the perfect candidate regardless of the size of the rule set. More surprising is the fact that bigger rule sets provide worse candidates for targets 2 and 3. The problem with bigger rule sets is probably that they mean a bigger or more difficult search space for the GA.

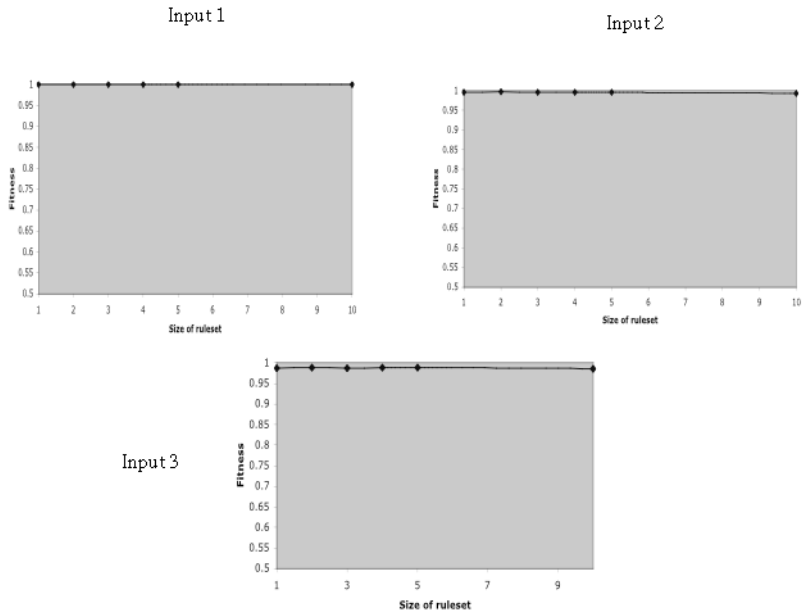

Fig. 5. Average fitness obtained by the GA with different values for the size of the rule set. When the target is simple, like in target 1, the GA always finds a perfect match regardless of the number of rules.

The size of the rule set has also a noticeable impact on the rules that are used to reconstruct shapes (see Fig. 6). If the size of the rule set is small, the range of possible rules is only 2, rule 9 and rule 10. As the size of the rule set increases, so does the range of possible rules. Still, it is clear that rules in the range 8-11 have better chances of being included in a given rule set than rules outside this range.

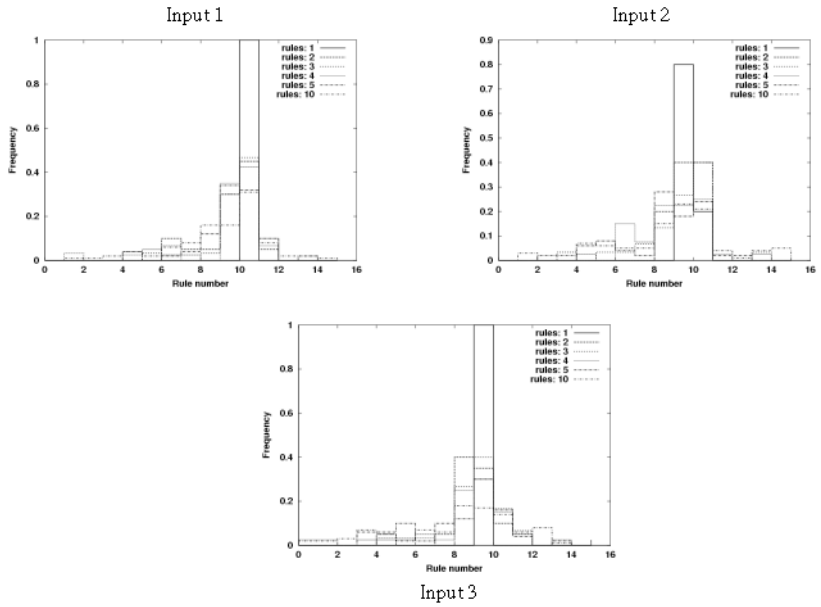

Fig. 6. Rules used to reconstruct the inputs in the first experiment

5.2 Results for Experiment 2.

In the second experiment, the performance of the GA as the EfA iteration variable increases seems to improve in target 1 and 2 but not in 3 (see Fig. 7).

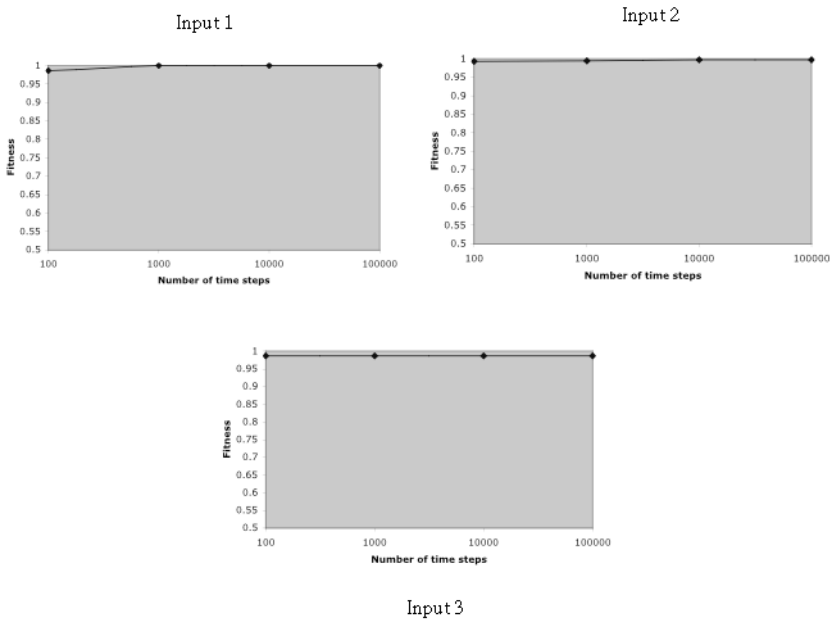

Figure 7. Fitness of the best candidates for each input in the second experiment

Another interesting result from this experiment is that only rules in the range 8-11 have a chance of appearing in the rule set of a good candidate for a solution in the later stages of the evolution of the GA.

6 Analysis

In terms of performance, the experiments suggest that there are no major differences between the different parameters used for both experiments. It is conceivable that this result may be affected by the fitness function, which may be allocating good fitness ratings for too many individuals.

Nevertheless, the results show that the GA converges, on average, to an optimal solution in fewer generations when the EfA runs for more time steps. This advantage has to be offset with the fact that the GA normally evolves faster with EfA running for fewer iterations. When the GA evolves larger rule sets it can potentially construct shapes with richer complexity, but this advantage turns to be a disadvantage when the target to be reconstructed is simple. In these cases, the fact that the search-space increases as the size of the rule set gets bigger, is enough to give smaller rule sets an edge in terms of performance.

Another observation from these tests is that of all the 16 potential rules, there is a small subset (rule numbers in the range 8-11) that are much more likely to be included

in the rule set of a fit candidate. Previous analysis has shown that these particular rules tend to cluster the IC they are provided with. Since the ICs are randomly generated and on average contain no clusters, is not surprising that rules that create clusters are popular. The reason why rules with numbers greater than 11 are not usually included in the rule sets of fit candidates is that automata using these rules need a large number of automata as their neighbours so they stop moving around. Also, previous analysis has shown that these rules need more iterations before reaching a stable situation. Therefore, it is likely that automata that follow rules with numbers larger than 11 would be more successful in an EfA with a high proportion of automata over empty cells, that were iterated a huge number of time steps.

7 Conclusion

This paper has demonstrated that it is possible to reconstruct shapes of moderate complexity using a GA that evolves rules of an effector automata. The system we described takes advantage of the features of emergence and self-organisation of CA. Though none of the automata have a plan of how the lattice should look at the end, they manage to move to appropriate new positions by interacting with each other and following their evolved rules. By doing this they ensure that the resulting pattern in the lattice looks similar to the target that the system wants to recreate, regardless of the position they occupied at the beginning.

These results are interesting by themselves and could be very significant for material scientists. A system able to interpolate 3D microstructures from 2D cross-sections is important, and could enable new materials to be engineered that will be used to build smaller mobile phones, faster engines or safer cars. The system shown in this paper can easily be extended to create 3D shapes starting from 2D images as the information it uses to reconstruct them is the same.

References

[1] Gordon G. E. *Structures or Why Things Don't Fall Down*, Pelican Books, London. (1978)

[2] Raabe D. *Computational Materials Science: The simulation of materials, microstructures and properties*, Wiley, Weinheim.. (1998)

[3] Basanta D., Miodownik M. A., Holm E. A., and Bentley P. J. Designing the Internal Architecture of Metals using a Genetic Algorithm. *Computer-Based Design. Engineering Design Conference 2002. Professional Engineering Publishing Ltd, London, UK. pp. 349-355.* (2002)

[4] Brandon, D., Kaplan, W. D. Microstructural characterization of materials. Wiley, Weinheim, (1999)

[5] Ulam, S. On some mathematical properties connected with patterns of growth of figures. In Proceedings of Symposia on Applied Mathematics, volume 14, pages 215-224. American Mathematical Society. (1962.)

[6] Raabe D. Cellular automata in materials science with particular reference to recrystallization simulation. Annual review of materials research. (2002)

[7] De Garis, H. The Evolutionary Engineering of a Billion Neuron Artificial Brain by 2001 Which Grows/Evolves at Electronic Speeds Inside a Cellular Automata Machine. Published in Sanchez, E., and Tomassini, M. Towards Evolvable Hardware; The Evolutionary Engineering Approach. Springer. (1996)

[8] Mitchell M., Crutchfield J. P., Das R. Evolving Cellular Automata to perform computations. Published in Baeck T., Fogel D., and Michalewicz (Eds.), Handbook of Evolutionary Computation. The institute of physics. (1997)

[9] Andre, D., Bennett, F., Koza, J. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. Published in Koza, J.R, Goldberg, D.E., Fogel, D.B., Riolo, R.L., Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press. (1996.)

[10] Corno, F., Reorda, M. S., and Squillero, G. Exploiting the Selfish Gene Algorithm for Evolving Hardware Cellular Automata. Proceedings of the 2000 Congress on Evolutionary Computation CEC00. IEEE press. (2000).

[11] Kumar, S., Bentley, P. The ABCs of evolutionary design. Investigating the evolvability of embryogenies for morphogenesis. Genetic and Evolutionary Computation Conference (GECCO '99) RN/99/2. (1999)

[12] Bentley K. A. (2002). Exploring aesthetic pattern formation. Generative Art 2002 conference proceedings. (2002)

[13] Lohn, J. and Reggia., J. Discovery of Self-Replicating Structures using a Genetic Algorithm. In 1995 IEEE International Conference on Evolutionary Computing. (1995)