Evolving the Game of Life

Dimitar Kazakov, Matthew Sweet

5 January 2004

Abstract

Cellular Automata are used in a number of areas: fluid dynamics, ecosystem modelling, etc. In this paper we attempt to provide a method of evolving rule sets that support "interesting" life. Rather than identifying individual creatures, the entropy of the cellular automaton is used to calculate the fitness (or interesting-ness) of a rule set. Genetic algorithms are employed for the search of good solutions, including a novel technique for fitness scaling, the advantages of which are experimentally demonstrated.

1 Introduction

The aim of this work is to evolve a set of rules for a cellular automaton (CA) with a good potential for producing "interesting" life forms, e.g., such that grow, move, have a long life span, consist of differentiated types of tissue, are compact and/or preserve their shape or produce a copy of themselves as a by-product of their growth. Rather than focussing on each such property in isolation, and trying to find the combination of rules that promotes it, an attempt is made to define a number of entropy-based criteria that are used as an indicator of the likelihood of such desirable life-forms appearing in a given CA. Each of these criteria is used as a fitness function of a genetic algorithm (GA) searching through the range of possible CAs. Also, to improve the GA performance on a task in which, intuitively, good solutions are few and far apart, we employ two techniques to maintain the balance between population quality and diversity — crowding [3] and extended fitness [8] — and show the substantial advantages that the latter brings in.

Cellular automata are dynamic systems consisting of a lattice of cells (in any number of dimensions) each of which has a number of associated discrete states. The state of these cells is updated at discrete time steps, the resultant state dependent upon local state rules. Here the setup is based on John Conway's well-known Game of Life. The range of possible rules is extended by allowing for up to three different types of non-empty cell; each type of cell releases a different type of chemical into all neighbouring squares (both occupied and empty), each of which will receive a unit of this chemical. At the moment, the quantities of chemical are only dependent on the current state of the automation, but this could change in the future to include gradual depletion. Each CA is defined by the rules it provides for the preservation, creation and destruction of each type of cell. Compare the classical rules expressed in this way:

```
New Cell if Square contains 3 units of chemical
Preserve Cell if Square contains 2-3 units of chemical
Destroy Cell if Square contains less then 2 or more than 3 units of chemical
```

with some of the possible rules in the range of games allowed:

```
New Cell of Type 1 if Square contains 3 units of Chemical 1 (same rule as above)
New Cell of Type 1 if Square contains 2 units of Chemical 1 and 1 unit of Chemical 2
New Cell of Type 1 if Square contains 3 units of Chemical 1 or 2 (in any proportion)
New Cell of Type 1 if Square contains 2 units of Chemical 1 and 1 unit of Chemical 2
New Cell of Type 1 if the difference in concentration of Chemical 2 and Chemical 1 in
the square is 2 units.
```

2 Genetic Algorithms and Extended Fitness

Genetic algorithms are search algorithms based on the mechanics of Darwinian evolution and genetics. GAs differ from more standard search procedures in four main ways: they use probabilistic transition rules rather than deterministic rules, they work with a population of search points rather than a single one, they use a coding of the parameter set rather than using the actual parameters and they use a problem dependent

fitness function, but no other auxiliary knowledge about the search space [5]. Using genetic algorithms for the search of CAs with desired properties is a trend with a relatively short history [9, 10, 1]. The fitness landscape for this problem is likely to be highly multi-modal and therefore one must consider techniques for improving the effectiveness of the genetic algorithm under these conditions. Holland [7] has observed that for fitness functions with a rugged landscape two high fitness parents often generate 'lethals' (very low fitness offspring). De Jong [3] suggests that this effect can be combated using an algorithm with crowding factoring such that a new individual will replace an individual from the previous generation with a similar genetic make up.

An alternative to De Jong's crowding factor model is the extended fitness approach [8]. As in population genetics, the components of this fitness are defined to measure the relative advantage that a gene gives to its carrier with respect to all other genes that can appear in the same locus. The fitness of the whole genome then can be computed as the averaged contribution of all its loci [4]. Extended fitness favours individuals with good genetic material (building blocks) and relies on the assumption that these could potentially be useful in the evolutionary search.

3 Evolving Cellular Automata

There are four classes of cellular automata as defined by Wolfram [6]. Class 1 automata evolve after a finite number of time steps from almost all initial states to a single unique homogenous state in which all cells in the automaton have the same value. Class 2 automata generate separated simple structures dependent upon their initial configuration. The evolution of class 3 automata produces chaotic patterns from almost all initial states; the statistical properties, after sufficient time steps, of the patterns produced by almost all initial states is the same. All other automata fall into a fourth class where for most initial configurations the automaton cells will become almost entirely unpopulated, however some stable live structures will develop which will persist indefinitely.

It would be unlikely that interesting creatures could exist in class 1 automata since after a finite time period all cells in the automaton would have the same value; therefore no interesting creatures could persist past this point. Class 2 automata merely generate simple separated structures (either periodic or static), which means that no reproducing or dynamic creatures could be created. Class 3 automata cannot support interesting creatures since the patterns produced are chaotic. Therefore the focus of this paper must be the fourth class of automata since they are most likely to be able to satisfy the criteria of supporting interesting creatures.

When the fitness of a set of rules is to be determined then the cellular automata that is represented by that set of rules needs to be run on some initial configuration of cells in the lattice. There are two techniques to consider that have been used in previous research to generate initial states — random generation and specific pattern generation. In random generation some portion of the board is populated at random with live cells and the density of live cells is dependent upon the probability of each cell being live. In specific pattern generation a user defined pattern is created usually at the centre of an otherwise unpopulated lattice.

In their work [2] Basanta *et al.* used a static initial state — only the central cell is live and all others are initially unoccupied. This reduces the amount of computation necessary for each fitness calculation. For an algorithm using a randomised initial state multiple different initial states must have their fitness calculated to attain an accurate fitness for the rule set, however with a single static initial state the fitness must only be calculated once.

The approach adopted here is based on two GAs. Each individual of the GA encodes the rules of one CA. For each of these CAs, another GA is used to test how well the so defined automaton supports life. The individuals of this second GA encode an initial configuration of cells each, and the CA is run for a (fixed) number of steps, after which the success of the initial configuration is evaluated with respect to the predefined fitness criterion. The best ever performing individual of the second GA passes on its fitness to the individual encoding the CA in the first GA. By running these two nested loops of natural selection, one aims to select and evolve a range of CAs with the desired properties.

4 Entropy Based Fitness of Cellular Automata

In this section, we introduce the fitness criterion used by the inner of the two above mentioned GAs.

The entropy of a system is defined to be the level of orderliness or chaos in that system – the higher the level of chaos, the higher the entropy. Therefore the more predictable the next state of the automaton is,

the lower the entropy of the automaton. Wolfram [6] defines the entropy of a cellular automaton to be:

$$S = \sum_i p_i \ \log_2 p_i \tag{1}$$

where S is the entropy and p_i is the probability of state i. For two-dimensional automata an equation was developed by Wolfram and Packard [11] to express the *set entropy* of an automaton.

$$S = \lim_{X,Y \to \infty} \frac{1}{XY} \log_k N(X,Y) \tag{2}$$

where S is the entropy, X and Y are dimensions of the area for which the entropy is being calculated and k is the number of different states. When used to calculate the entropy of a particular state N(X,Y) is the number of possible different states by which the current configuration can be represented. E.g. a 3×3 area containing one live cell could be represented by 9 different states so N(X, Y) = 9. The division by XY normalises the entropy values so that entropies over different tile sizes can be compared.

To calculate an approximation of this set entropy is far cheaper than to calculate the entropy using the first equation, since this first option would require us to enumerate all possible states. Also we are not interested in the overall automaton entropy but rather in the entropy of *tiles* [10], a lattice of cells which is a component of the overall lattice of cells making up the automaton. If we based the fitness function on the overall entropy then selecting for high entropy would result in a chaotic class 3 automaton which could not support interesting life. Conversely, selecting for a high degree of order (low entropy) would favour class 1 and class 2 automata which reach a stable state and never leave it. Interesting life is most likely to develop on the boundary between order and chaos; we need dynamic behaviour inherent in chaotic systems but we also need a degree of order to keep any life which develops coherent. Therefore we wish to promote rule sets which contain both order and chaos - this can be achieved by assigning a *fitness based on the spread (standard deviation) of the entropies of (a sample of) the tiles making up the automaton.*

5 Results and Evaluation

Several transitions of the fittest rule set generated when the genetic algorithm was run for 200 generations are shown in Figure 1. This rule set started from almost any initial configuration will generate a pattern resembling that of animal spots or organs made of layers of tissue. The automaton is resilient to damage and will regrow any killed cells so long as the core green cell is not removed.

Figure 2 shows an interesting result related to the way GA is implemented: extended fitness produces faster improvements of the average population fitness despite the overhead present at for the calculation of each new generation. Moreover, extended fitness has a positive effect not only when the number of generations is compared, but also in terms of computational time, which is a very rewarding result. In a final observation on extended fitness, we have shown that the average population fitness can improve faster (again, in the stronger terms of time needed) with a larger population, which may seem counter-intuitive. This could be explained by the multi-modal fitness landscape in which, to obtain an accurate map of gene fitness, one may need a larger population of rule sets to cover a larger number of combinations of genes.

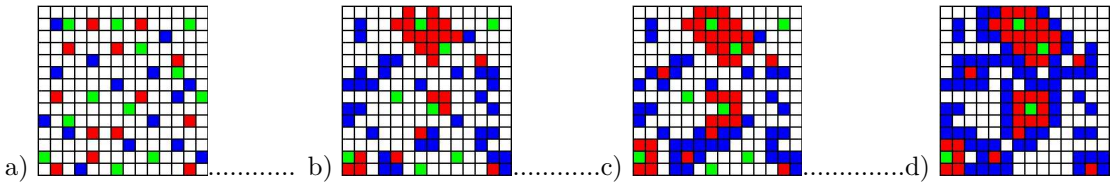

Figure 1: (a) Randomly generated initial configuration of the cellular automaton (density 0.2); (b) state after 1 transition; (c) state after 2 transitions; (d) state after a large number of transitions.

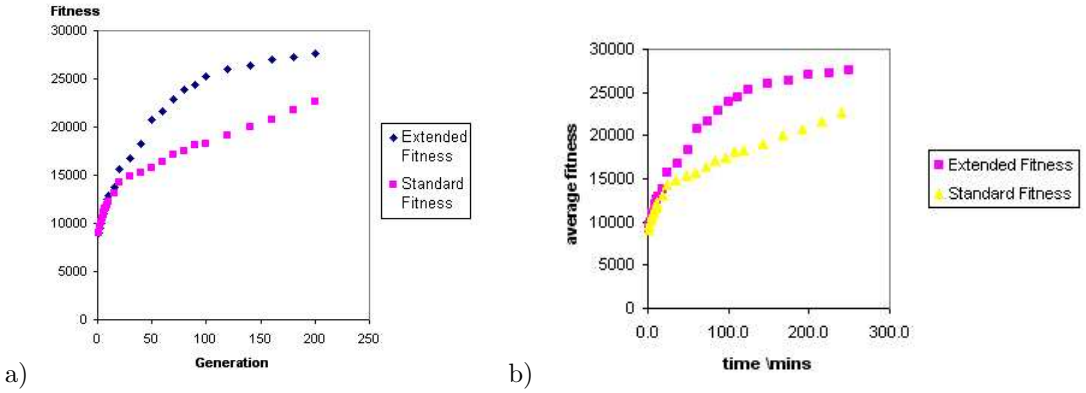

Figure 2: Comparison of the average fitness of rule sets for extended fitness and standard fitness approaches by (a) generation and (b) time (averaged over 10 runs using a population size of 100).

6 Further Work

To discover more interesting rule sets the resulting population from running the entropy based genetic algorithm can be used as the initial population for another genetic algorithm. The fitness function for this algorithm will identify creatures based on their proportional cell content and therefore should be able to identify movement and reproduction of creatures. Another fitness function to be tested uses the mean deviation from the expected number of each type of cell to produce a measure of entropy. Also further work could be done on the population of the entropy based genetic algorithm to identify the optimal population size.

References

[1] Evolving Cellular Automata (EvCA) Group papers, Santa Fe Institute, New Mexico. URL: http://www.santafe.edu/projects/evca/Papers/papers.html.

[2] D. Basanta. Evolving automata to grow patterns. 2003.

[3] K.A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. 1975.

[4] D.S. Falconer. *Introduction to Quantitative Genetics*. Longman, London, second edition, 1981.

[5] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[6] Gotts. *Statistical Mechanisms of Cellular Automata*. 1983.

[7] J.H. Holland. *Adaption in natural and artificial systems*. 1975.

[8] D. Kazakov. Evolutionary algorithms with extended fitness. Technical Report YCS 370, Department of Computer Science, University of York, 2003.

[9] Melanie Mitchell, Peter T. Hraber, and James P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

[10] E. Sapin, O. Bailleux, and J. Chabrier. Research of complex forms in the cellular automata by evolutionary algorithms. In P. Liardet et al., editor, *Proc. of the 6^{th} Intl. Conf. on Artificial Evolution*, Marseille, Oct 2003.

[11] S. Wolfram and N. Packard. Two-dimensional cellular automata. *Statistical Physics*, 38:901–946, March 1985.