An exact solution to the two-dimensional arbitrary-threshold density classification problem

Sami Torbey and Selim G. Akl

School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6
{torbey},{akl}@cs.queensu.ca

March 14, 2008

Abstract

Density classification is one of the most studied cellular automata problems. As initially presented [5], it does not have an exact solution [7]. However, [2] and [4] demonstrate that a simple change in the proposed accepting condition makes such a solution easily achievable for one-dimensional cellular automata. We show not only that a similar result is possible for two-dimensional cellular automata, but also that this result can be obtained in $O(\sqrt{n})$ expected running time (where n is the total number of cells in the automaton), compared to an $O(n)$ running time for the one-dimensional case - thus answering an open problem presented in [1]. Our proposed automaton is also capable of classifying arbitrary density thresholds (not just $\frac{1}{2}$).

Key words: cellular automata, arbitrary density classification, majority, two dimensions, rule 184, gravity

1 Background

The density classification (also known as majority) problem is arguably the most studied problem in cellular automata theory. It can be described as follows: given a two-state (black and white) cellular automaton, does it contain more black or more white cells? The accepting condition can vary as long as it provides a clear-cut answer without excessive computational work. However, the most frequently used (and first proposed [5]) accepting condition requires that the whole automaton converges to the denser colour, i.e. that all cells become black if black cells initially outnumber white cells, and vice-versa.

Land and Belew proved that a binary-state automaton with local neighbourhoods can never converge 100% accurately to such an accepting condition [7]. However, the search is still on for the rule with the highest accuracy. At the time of writing, such a rule is believed

to be a soon-to-be-published rule discovered through genetic programming in 2007 by Wolz and de Oliveira [12]. This rule achieves an 89% accuracy for a large number of random initial conditions on a one-dimensional lattice of size 149 with a neighbourhood of radius 3 (which are the conditions conventionally used to compare the efficiency of such algorithms) - larger lattice sizes generally reduce the accuracy. In two dimensions, the best known rule at the time of writing achieves an 83% accuracy with a Moore neighbourhood of radius 1 (attempts to use larger neighbourhood sizes did not significantly improve performance) [12]. In this paper, we show that a reasonable change in the conventional boundary and accepting conditions enables us to achieve a 100% accurate solution for arbitrary density thresholds.

1.1 Rule 184

Although Land and Belew showed that a binary-state cellular automaton cannot perfectly perform the density classification task under the all-black or all-white accepting condition [7], their proof does not necessarily hold under other accepting conditions. For example, Capcarrere et al. proposed an elementary cellular automaton (Rule 184) of radius 1 that is perfectly capable of performing that task under a different termination condition [2, 4, 1]: if there are two consecutive black cells anywhere in the automaton after $\frac{n}{2}$ cycles - where n is the lattice size i.e. the total number of cells - then black cells have the majority. Conversely if there are two consecutive white cells then white cells have the majority. If none of these conditions is true (the automaton is a perfect chequerboard) then both colours are in equal numbers. Capcarrere et al. also showed that under Rule 184, having both consecutive black and consecutive white cells together in the automaton after n cycles is impossible. In Figure 1, we show an example of several iterations of Rule 184 leading up to its accepting condition.

The key behind Rule 184's success is that it is conservative [1]: this means that the numbers of black cells and white cells do not change throughout the computation; they are merely reordered. In fact, it makes intuitive sense that non-conservative binary-state automata are incapable of performing majority classification perfectly with a constant neighbourhood size (not a function of the lattice size). The accepting condition chosen by Capcarrere is perfectly valid; it can be easily checked by both humans and machines. Capcarrere rightly concludes from this result that computation is in the eye of the beholder [1]. Whether this simple sorting can be seen as a computation or not is only a matter of perspective. By using human intuition to change an accepting condition, a problem that was previously unsolvable becomes very easily solvable. This also shows that universality and "interestingness" are two separate ideas: Rule 184 belongs to Wolfram's Class 1 or 2 (see [11] for more information on Wolfram's classification), yet it still perfectly solves a desired problem. This result contradicts the view on "interestingness" outlined by Gutowitz and Langton in [6].

Fuks later extended Capcarrere's idea to show that even under the all-black or all-white

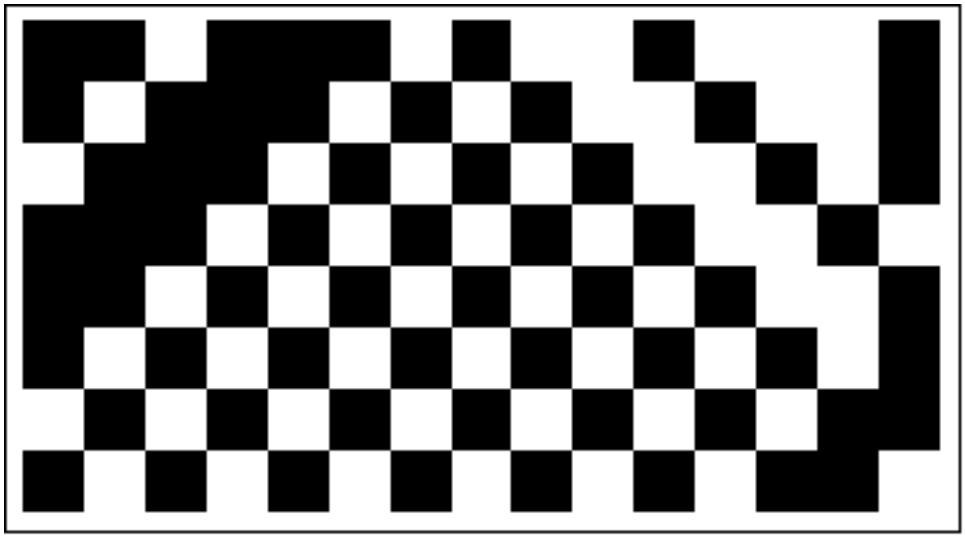

Figure 1: 7 iterations of Rule 184 highlighting an initial majority of black cells

accepting condition, a perfect classification is possible using a binary-state automaton [4]. The trick is to allow a rule change from Rule 184 to Rule 232 after $\frac{n}{2}$ cycles; the automaton then converges to the desired accepting condition after no more than n cycles. Rule 232 is a simple totalistic local majority automaton; it only becomes effective for global majority after Rule 184 ensures that no consecutive black or white cells exist unless they constitute the global majority.

1.2 Static boundary conditions

We propose a slight modification of Capcarrere's idea in order to make it even more powerful: replacing the periodic boundary conditions with static boundary conditions - a black cell to the right and a white cell to the left. Under these conditions, Rule 184 stacks all black cells to the right side of the automaton while leaving all white cells on the left side. The system can then be seen as exerting a force (such as gravity) attracting all black cells to the right; we expand on this concept in the next section to provide the first automaton capable of performing density classification perfectly in two dimensions. For now however, we notice that by changing the boundary conditions, only the middle cell (or the middle two cells if the lattice size is even) needs to be checked for one cycle to provide an answer to the majority question (under Capcarrere's approach one needs to check either all cells for one cycle or one cell for n cycles). The change to static boundary conditions simplifies spotting the accepting condition for both humans and machines. It also allows generalization of the problem to arbitrary density thresholds simply by checking the colour of the cell corresponding to the desired density threshold (the middle cell for a $\frac{1}{2}$ threshold), while this cannot be done

under periodic boundary conditions. We recommend consulting [3] for more information on arbitrary density thresholds in one-dimensional cellular automata.

2 Two-dimensional exact density classification

After explaining his solution to the one-dimensional majority classification problem, Capcarrere presents an open problem [1]: how can this solution be extended to two dimensions, and can the two-dimensional majority classification problem be completed in $O(\sqrt{n})$ cycles (assuming a square shape where n is the total number of cells in the automaton)?

We devised a solution based on Rule 184 that performs in expected $O(\sqrt{n})$ time. This solution, which we call the "gravity automaton" has static vertical boundary conditions and periodic horizontal boundary conditions: the top row is all-white, the bottom row is all-black, and the left and right columns are connected to each other. The gravity automaton can therefore be seen as an upright cylinder. All cells have a uniform neighbourhood shown in Figure 2: it is a Moore neighbourhood of radius 1 with one additional cell immediately to the right of the middle row (this can also be seen as a von Neumann neighbourhood of radius 2 where three cells are not used).

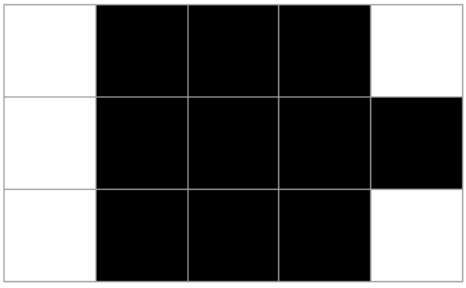

Figure 2: Representation of the neighbourhood in the gravity automaton

2.1 Transition rules

Since this a conservative cellular automaton, we explain its behaviour in terms of particle motion where black cells are considered particles and white cells are considered free space. We believe such a description to be much more intuitive than one describing every transition rule in detail. The reason why we call this system the "gravity automaton" becomes more apparent with this description; the top white row boundary condition is just additional free space above all particles, while the bottom black row is the ground, beyond which particles cannot fall. The particles then behave as follows:

1. If a particle has nothing right under it (the cell below it is white), it simply falls by one cell

2. If the particle has another particle immediately below it and no particle immediately above it (it is the top particle in a column of particles), it moves by one cell to the left (the gravity vector is not fully vertical - it leans slightly to the left)

3. If the particle has other particles both immediately below it and immediately above it (it is part of a column), it moves to the right by one cell; here, the top particle in the column (the one which satisfies rule number 2) can be seen as having with its left movement pushed all particles below it to the right

Rules 1 to 3 are in priority sequence. This means that under rule 2, the particle also checks if there is a particle diagonally to its top left. If so it does not move in order to avoid a collision with that particle, which would be falling into the same spot according to rule 1. The same check is done to the top right under rule 3 to avoid a collision with rule 1. A particle satisfying the conditions of rule 3 also avoids a collision with another cell under rule 2 by checking the cell two positions to its right; if that cell contains a particle then it is given priority and the checking particle does not move to the right (the running time caused by this prioritization can be reduced using a larger neighbourhood size). Formally, each one of the three rules can be described using two simulaneously-executing rules: one where the particle disappears (a black cell becomes white) and another one where the particle reappears (a white cell becomes black). Figure 3 shows the resulting six formal rules. Figure 4 presents a small example illustrating these rules. Figure 7 shows a larger, more realistic example.

2.2 Terminating condition

The accepting condition is reached when as many rows as needed are completely full. Figure 5 illustrates this condition for the example automaton presented in Figure 4. The behaviour of the topmost row is similar to Rule 184. Thus to solve the majority problem, if the number of rows is even, one only needs to check if there is any particle on the upper middle row after a certain number of iterations; if so, then black has the majority. Otherwise, it is either white that has the majority or the initial numbers of black and white cells are equal; if the latter case is a concern one can check the lower middle row for any gaps that, if present, would confirm that white has a strict majority (if no such gaps exist then there would be no majority). If the number of rows is odd, one only needs to check if there are any consecutive black or white cells in the middle row. If so, whichever colour has consecutive cells has the majority.

Similarly to what we did with Rule 184, we can generalize this rule for arbitrary density thresholds: for thresholds other than $\frac{1}{2}$ and up to an $O(\sqrt{n})$ resolution, we easily adapt the accepting condition by checking the row corresponding to the desired threshold instead of the middle one. We can also achieve the maximal $O(n)$ resolution by using $O(\log \sqrt{n})$ bits of memory to count the number of black or white cells in the desired row (instead of simply checking for consecutive cells of the same colour).

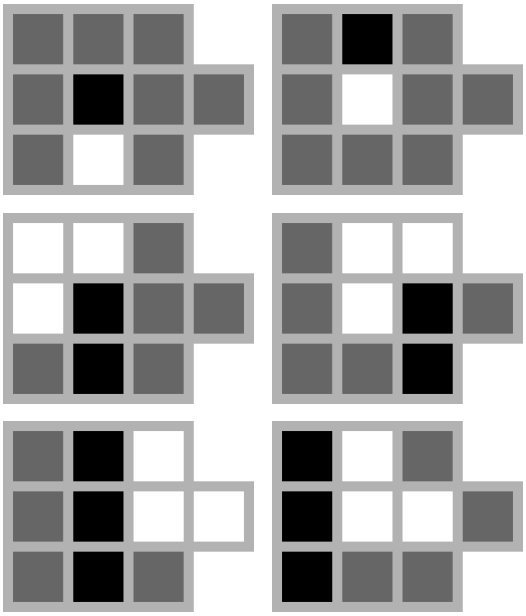

Figure 3: The six formal rules governing the gravity automaton. The gray cells are "don't cares" (each one of them can be arbitrarily either white or black - the rule would still apply). These six rules present the only instances where a transition would change a cell's state from black to white or vice-versa. The cells maintain their states under all other neighbourhood configurations. The rules on the left side are the ones where the particles disappear, and the ones on the right side are the corresponding rules where the particles reappear. The top two rules collectively constitute "rule 1" as described in the text, the middle two correspond to "rule 2" and the bottom two to "rule 3".

As with the one-dimensional case, we can reach an all-black or all-white terminating condition if desired by marking the middle row (private static information under the framework we presented in [9] and [10]) and changing the rules to propagate its majority colour across the rest of the automaton.

2.3 Analysis

By giving priority to the first rule, we guarantee that particles keep on falling while they can. If they cannot, they move left and right until they find an empty spot where they can fall. This simple strategy ensures that after a certain number of iterations, as many bottom rows as possible are guaranteed to be filled with particles. Then when the top row is partially full, the particles in it cannot follow the first and third rules and as such they always follow the second rule which mimicks Rule 184 exactly (but in the reverse direction). This means that the rules are correct and the accepting condition is guaranteed

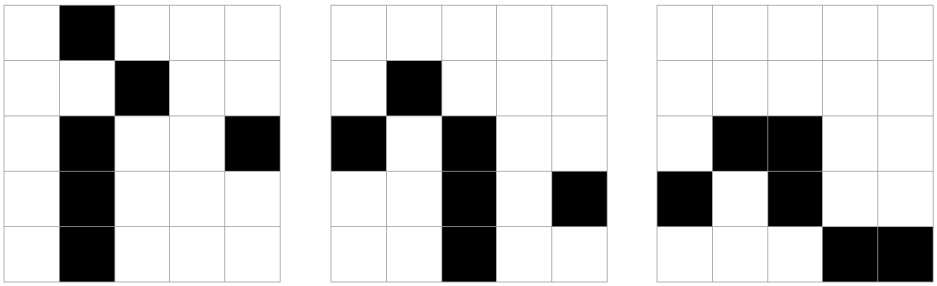

Figure 4: Three consecutive iterations of a small example gravity automaton. Notice how the two rightmost particles and the top particle simply fall (following the first rule). The top particle in the column moves to the left (rule 2), and the lower two particles move to the right (rule 3) - there are no conflicts in the first transition. In the second transition, the two uppermost particles in the column face a conflict (the first one under rule 2 and the second one under rule 3) and therefore do not move.

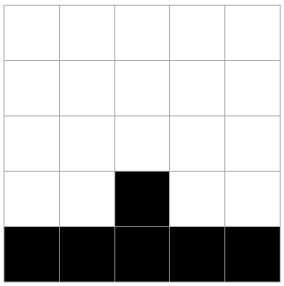

Figure 5: Terminating condition of the example automaton reached two transitions later

to be reached after some number of cycles.

The question that remains to be answered is: how many cycles are needed? By dissecting the vertical (1) and the horizontal (2 and 3) rules individually as illustrated below, we obtain a running time estimate of $O(\sqrt{n})$ under random intial conditions.

All particles in a column need no more than $\sqrt{n}$ time before they cannot fall anymore. This is illustrated by two cases: the individual particle and the column of particles. An individual particle in the worst case (topmost row) falls to the ground in $\sqrt{n}$ time. On the other hand, in a column of particles, only the lowest particle can fall at one time. The particles above it have to wait (we are assuming that there is no horizontal motion for now). k successive particles take $k-1$ cycles to be separated from each other, with their number shrinking by one particle at a time (the lowest particle separates itself from the column at each time unit). At this point, they can be seen as individual particles since they can all fall at the same time. The time for their fall to be completed is the time required for the topmost particle to reach the top of the column when the column is completely affixed

to the ground. For a column of size k, the top of the column is situated at k cells above the ground. Therefore, assuming the column is initially located at the topmost position, the particle needs to fall $\sqrt{n} - k + 1$ cells. The total running time for the fall becomes $k - 1 + \sqrt{n} - k + 1 = \sqrt{n}$.

In the horizontal case, once the column has reached the ground, the particles are separated from it at the rate of one to two particles per cycle (one to the left under rule 2 and one to the right under rule 3 if it finds a gap to fall into). There could be some conflicts slowing this movement down but generally it remains within $O(\sqrt{n})$. Combining these two movements yields an $O(\sqrt{n})$ total running time; this is especially the case when the initial conditions are generated at random, which means that there are no large discrepancies in the column sizes. Even if there were such discrepancies, this set of rules helps resolve them, both while the particles are in the air and when they touch the ground.

A large number of practical trials with random initial conditions (and varying concentrations of black and white cells) validated this result by showing that the number of cycles before the accepting condition is reached generally falls between $\sqrt{n}$ and $2\sqrt{n}$, and is never above $3\sqrt{n}$.

2.4 Special case

Although our analysis and every practical test we did on the gravity automaton resulted in an $O(\sqrt{n})$ running time, we were able to artificially generate a special case where more time is required for the automaton to reach its accepting condition. This is the case of a pyramid-like structure, as depicted in Figure 6, of size (number of cells) greater than $O(\sqrt{n})$. The running time in such a case is asymptotically equal to the size of the pyramid, given that particles constituting the pyramid fall one by one (at each cycle) to the left starting with the topmost particle. Obviously, the worst case would require an $O(n)$ running time. The pyramid itself does not need to be in the initial condition; any structure (such as an inverted pyramid) that can lead to a pyramid structure over time is sufficient. A superset of these structures is the set of structures where, in a rectangle, the middle column contains the largest number of black cells, followed by one column to its left and one column to its right containing one less black cell, etc; this continues until there are no black cells left so that if all black cells in a column are stacked on top of each other in the bottom of the rectangle, a shape similar to the pyramid depicted in Figure 6 is obtained.

Note that structures which do not exactly match Figure 6 but are similar enough also impose a relative performance toll (depending on their degree of similarity) on the algorithm. We believe these to be the only special cases, without excluding the possibility of other potential special cases. Practically, the chance of generating such a structure with a size greater than $O(\sqrt{n})$ from random initial conditions is very small.

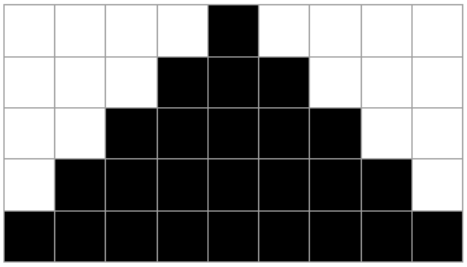

Figure 6: Worst case structure for the gravity automaton

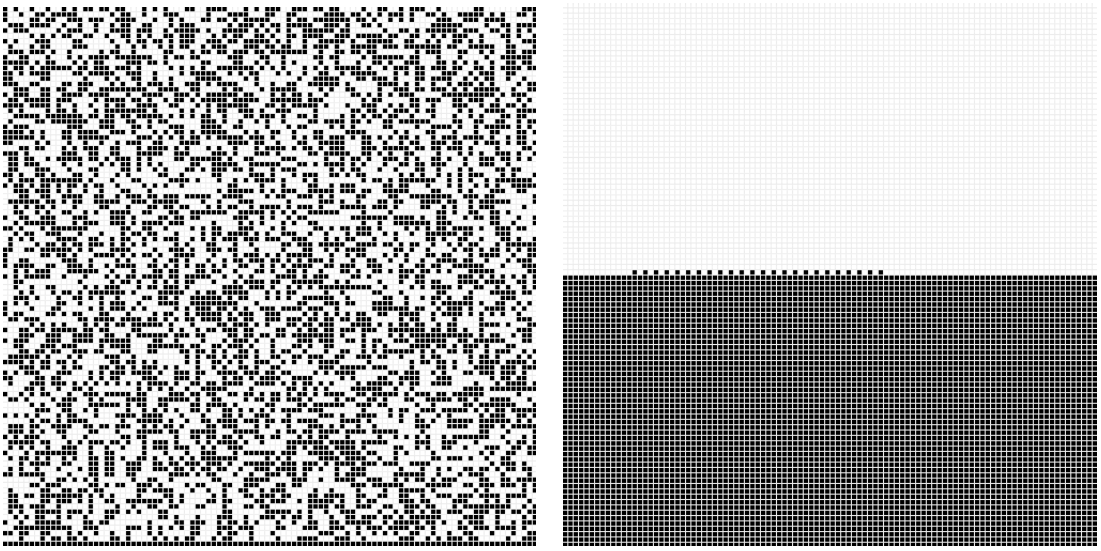

Figure 7: Randomly generated initial condition and terminating condition (reached after 160 cycles) of a 10,000-cell gravity automaton

3 Concluding thoughts

We have shown that by making simple changes to the accepting and boundary conditions, we were able to find an exact solution to the long-standing unsolvable (under traditional conditions) density classification problem in two dimensions. We believe that the changes we made are reasonable:

- The proposed accepting condition is better than the original one, in the sense that it is easy to check for both humans and sequential computers; the original all-black or all-white accepting condition is designed mainly for humans, since a sequential computer needs $O(n)$ time to check it (longer than the computation itself).

- On the other hand, the proposed boundary conditions are not difficult to implement on a cellular automaton with traditional periodic boundary conditions; they only require two consecutive rows to be set respectively to all-black and all-white, and then "de-activated" (using idle transition rules). As such, our solution uses two additional rows and can be seen as implemented using traditional periodic boundary conditions on non-uniform cellular automata (similar to the ones studied in [8]).

As a result of using such conditions, we were able to easily generalize our solution to arbitrary density thresholds. In addition, we achieved an expected $O(\sqrt{n})$ running time, which we believe to be optimal (although there could be an $O(\sqrt{n})$ worst-case solution using more complicated rules) for an exact solution since some implicit communication needs to take place and the farthest points are $O(\sqrt{n})$ cells apart.

This solution validates our opinion (expressed in [9] and [10]) that human intuition still plays a central role in the design of cellular automata, although thinking in terms of low-level rules can be difficult for humans. Instead, designers can use tools such as the ones presented in [9] and [10] to derive rules from higher-level ideas. For example, thinking in terms of particle movements and prioritization instead of transition rules was instrumental in reaching the solution outlined in this paper.

References

[1] Mathieu Capcarrere. *Cellular Automata and other Cellular Systems: Design & Evolution*. PhD dissertation, Swiss Federal Institute of Technology Lausanne, March 2002.

[2] Mathieu Capcarrere, Moshe Sipper, and Marco Tomassini. Two-state, r = 1 cellular automaton that classifies density. *Physical Review Letters*, 77(24):4969–4971, 1996.

[3] H.F. Chau, L.W. Siu, and K.K. Yan. One dimensional n-ary density classification using two cellular automaton rules. *International Journal of Modern Physics C*, 10(5):883–889, 1999.

[4] Henryk Fuks. Solution of the density classification problem with two cellular automata rules. *Physical Review E*, 55:2081R, 1997.

[5] Peter Gacs, Georgii L. Kurdyumov, and Leonid A. Levin. One-dimensional homogenuous media dissolving finite islands. *Problems of Information Transmission*, 14(3):92–96, 1978.

[6] Howard A. Gutowitz and Chris G. Langton. Methods for designing cellular automata with "interesting" behavior. Available at `www.santafe.edu/~hag/interesting/interesting.html`, 1988.

[7] Mark Land and Richard K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):5148–5150, Jun 1995.

[8] Moshe Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, 1997.

[9] Sami Torbey. Towards a framework for intuitive programming of cellular automata. Master's thesis, Queen's University, 2007.

[10] Sami Torbey and Selim G. Akl. Towards a framework for high-level manual programming of cellular automata. In *13th International Workshop on Cellular Automata*, Toronto, Canada, August 2007. The Fields Institute.

[11] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, January 2002.

[12] Dietmar Wolz and Pedro P.B. de Oliveira. Very effective evolutionary techniques for searching cellular automata rule spaces. *To appear in the Journal of Cellular Automata*, 2008.