An improved cellular automata based algorithm for the 45-convex hull problem

Adam Clarridge and Kai Salomaa

Technical Report No. 2008-547
Queen's University at Kingston
Ontario, Canada

{adam,ksalomaa}@cs.queensu.ca

July 2008

Abstract

We give a cellular automaton algorithm for solving a version of the convex hull problem. The algorithm is based on the one presented by Torbey, which requires a global transition rule change in order to complete its operation. By introducing several new states and giving a simpler set of transition rules, we lift the requirement for a global rule change in between the previous algorithm's shrinking and expanding stages. The algorithm uses several communication states to explicitly detect when the first (shrinking) stage has ended, and relying only on local state information the cellular automaton is able to begin the next (expanding) stage of the computation in such a way that correctness is ensured.

1 Introduction

Given a set of planar points, the two-dimensional convex hull problem is to find the convex polygon with the smallest possible area which completely contains all of the points. This problem has been solved efficiently using standard methods, and an $O(n \log h)$ algorithm exists [2], where n is the number of points and h is the number of vertices of the convex hull. We present a cellular automata algorithm to solve a version of the convex hull problem which can be represented exactly on a finite grid like a two-dimensional cellular automaton. It is known as the 45-convex hull problem: the polygon forming the convex hull must be composed of only horizontal lines, vertical lines, or 45 degree perfect diagonals.

Our algorithm is based on the cellular automaton algorithm described in the paper by Torbey [5]. They solve the 45-convex hull problem using a 3-state, Moore neighbourhood cellular automaton with semi-totalistic transition rules. The algorithm requires a global transition rule change after a certain number of generations, which depends on the size of the grid. The process is thereby separated into two distinct stages, and so the algorithm technically does not solve the problem with a standard two-dimensional cellular automaton. We define several additional states and transition rules that enable us to be certain that the first stage has finished, and hence we do not require a global rule change to transition to the next stage of the algorithm. We also define simpler transition rules for the first stage of the algorithm that correct errors that occurred with certain special types of input in the prototype algorithm [5].

We give a very brief introduction to cellular automata and provide some references on the subject before describing our algorithm in detail.

2 Cellular Automata

A cellular automaton can be described as a lattice of individual cells, each cell having a set of 'neighbour' cells which are usually in close proximity to it. The two standard radius 1 ($r = 1$) neighbourhoods in a two-dimensional cellular grid are shown in Figure 1.

The cellular automaton starts at time $t = 0$ with each of the cells in one of their n possible states, where n is fixed (not dependent on time or grid size). This starting configuration is known as the *initial condition.* All cells

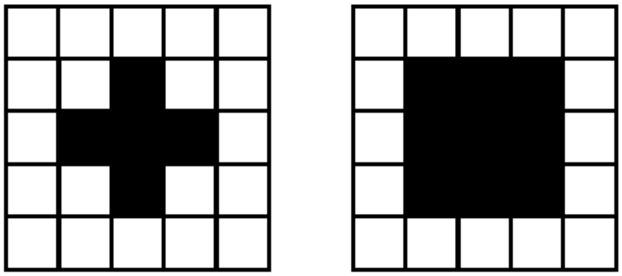


Figure 1: Radius-1, two-dimensional Von Neumann neighbourhood (left) and Moore neighbourhood (right)

have their own state transition function, whose input is their state and the states of each of their neighbours, and output is one of the possible states. When the CA advances from time step t to $t+1$, all of the cells consult their state transition functions using input from time step t, and the output of the function is their new state at time $t+1$. The entire CA updates synchronously in this fashion.

The *boundary conditions* are the CA rules which apply at the edges of a finite cellular array, which may be different from the rest of the transition rules. A CA is said to be *uniform* if all of its cells evolve using the same transition rule, and *totalistic* if the input to each cell's state transition function is only dependent on the sum of the states of its neighbours.

For more information on cellular automata in general, see [3, 4, 6–8].

3 The Algorithm

In this section we will discuss the specifics of the proposed algorithm. The algorithm has three main facets or stages:

- Using 4 states and starting with a CA that has black cells marking the input points and grey cells everywhere else with a white cell border, the grey area shrinks so that it is completely contained within the 45-convex hull of the black cells.

- Using 18 additional communication states, the algorithm identifies a point in time where it can be sure that the grey area has finished shrinking.

- Using a slightly modified version of the technique proposed by Adamatzky [1], the shrunken grey area expands to the 45-convex hull of the input points.

The initial configuration is all grey cells except for a white cell outer boundary and any number of black cells, which are the input points. Note that the white cells at the boundary could be simulated by standard CA rules that are applied at the edges of the (finite) cellular array. Having white states at the boundary spares us the trouble of defining separate boundary conditions for the simulations.

The first and third stages of the algorithm actually work toward finding the 45-convex hull: the first stage contracts the grey area so that it is completely contained within the 45-convex hull of the black cells in such a way that the third stage expands the shrunken grey region to the exact 45-convex hull. The only function of the second stage is to detect that the first stage is in fact complete. It runs in parallel with the first, and uses a set of communication states (which do not interfere with the operation of the first stage) in order to verify that the first stage is complete.

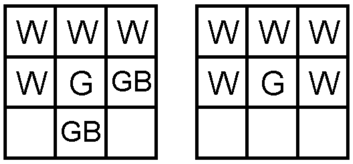
Our implementation has longer running time on average and uses many more states than the one proposed by Torbey, but does not require a global rule change at any time. Also, our algorithm addresses the problem of a set of special cases which Torbey's algorithm does not solve correctly.

3.1 Stage 1

There are initially only white, grey and black cells on the grid for this stage. An example input is depicted in Figure 3. The black cells represent the points for which we are trying to find the 45-convex hull. In this stage we want to 'shrink' the grey area (make grey cells transition to white cells) in such a way that the grey area is completely contained within the 45-convex hull of the input points. Later on, in the third stage of the algorithm, the grey area expands to cover exactly the 45-convex hull.

The challenge is to come up with a set of simple local rules for the CA so that the grey area shrinks smaller than the 45-convex hull of the set of input points, but stays connected. In the algorithm by Torbey, a complex set of pseudo-totalistic (meaning that the input to the state transition function depends on the sum of certain subsets of the neighbour set, not the entire neighbour set) transition rules almost achieved correct behaviour in all cases.

We found that a much more simply represented set of rules achieves the same result: all rotations and reflections (mirror images) of the two rules shown in Figure 2 cause the first stage to be completed correctly in almost all cases. We must note that this stage of the algorithm treats the states introduced in the second stage of the algorithm (Finder state, Communication states) as if they were the white state, since they do not affect the grey area's shrinking. In Figure 3, one can see these rules in action.



W	W	W
W	G	GB
	GB	

W	W	W
W	G	W

Figure 2: Rules for transitioning from the grey state (G) to the white state (W). All reflections and 90 degree rotations of these rules apply. GB means the cell can be either grey or black. Blank cells mean that the state of that cell is irrelevant to the rule.

However, there are some extreme special cases usually involving a very small number of input points where using just these rules can cause a disconnected grey area. A local cell configuration depicting this problem is given in Figure 4.

The algorithm proposed by Torbey has the same problem. We resolve this issue by introducing a fourth state, which we will refer to as the yellow state. This state is best understood as an intermediate state between grey and white. Yellow states exist in order to signify a temporary lack of local knowledge as to whether a particular grey cell should transition to white or stay grey, so a cell in the yellow state will either transition to the white state or the grey state on the next generation. If a grey cell has neighbouring states corresponding to any rotation/reflection of the rule in Figure 5, then it will transition to the yellow state. Note the white cell in the bottom right corner; this is the only way that the situation in Figure 4 can occur.

The transition rules for yellow cells are shown in Figure 6. These rules contain a tie-breaking protocol so that if two yellow cells are adjacent to each other, one will turn white (topmost or leftmost) and the other grey (bottommost or rightmost). It should be clear that in cases where two yellow cells are adjacent to each other, it does not matter which one turns white and

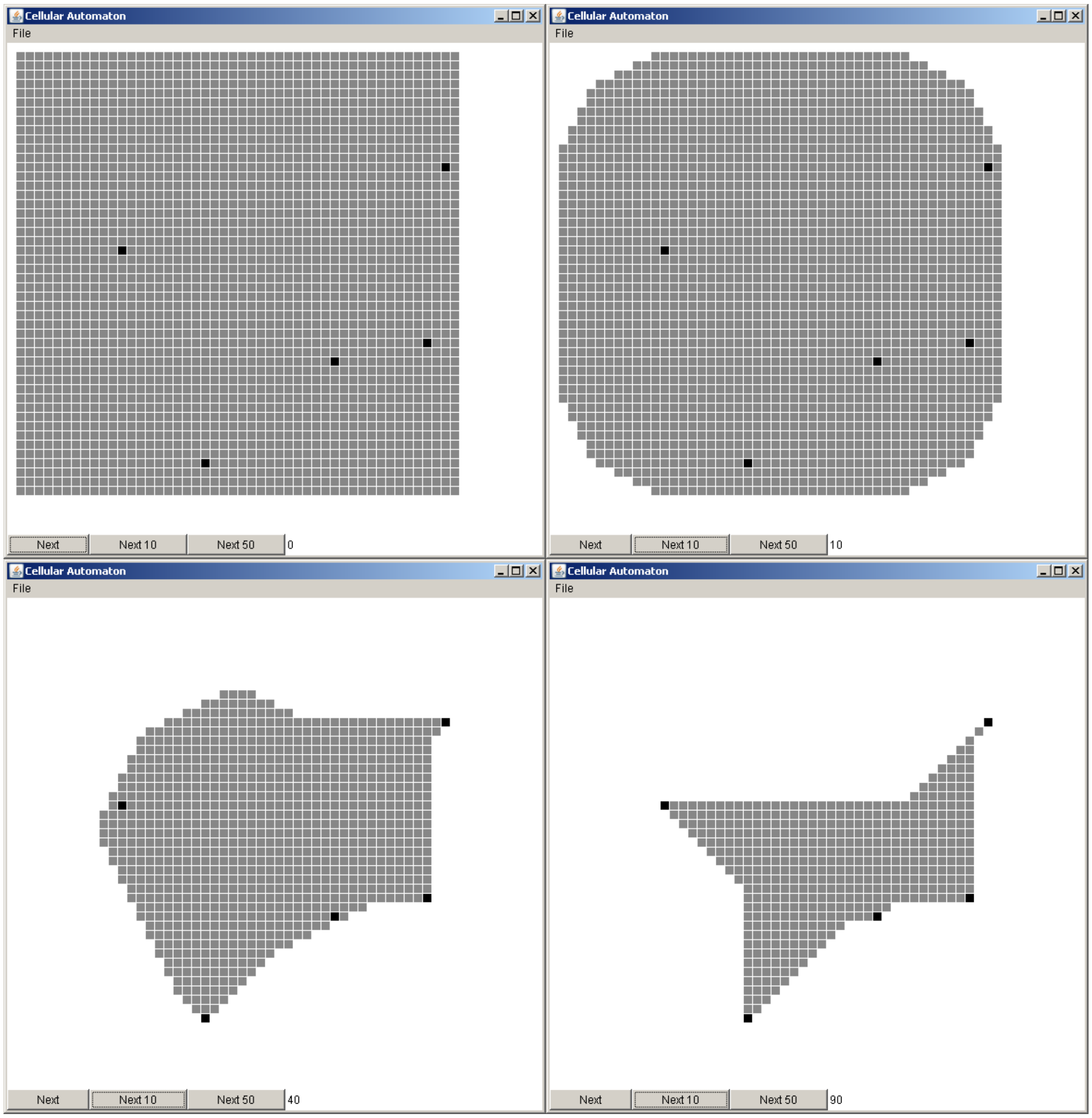


Figure 3: An example of the first stage of the algorithm in action - the grey area is contracted around the black cells.

which one turns grey since the grey area will definitely stay connected, and both cells in question must be inside the 45-convex hull of the input points.

The problem shown in Figure 4 is resolved by the yellow states, as shown in Figure 7.

W	W	W	W	W	B
W	W	W	W	G	W
W	W	G	G	W	W
B	G	G	W	W	W

→

W	W	W	W	W	B
W	W	W	W	G	W
W	W	W	G	W	W
B	G	W	W	W	W

Figure 4: A problem with using only the rules in Figure 2. Two of the grey cells will both turn white in the next generation, disconnecting the grey area. Note that there is a white cell boundary around the CA which is not shown.

W	W	W
W	G	GB
	G	W

Figure 5: Rule for going from the grey state to the yellow state. This rule applies in the situation where we are not sure whether or not a grey cell should turn white or stay grey. Again, all reflections and 90 degree rotations of this rule apply, and blank cells mean that the state of that cell is irrelevant to the rule.

3.2 Stage 2

We want to make sure that there are no grey areas still shrinking before the third stage of the algorithm begins. If we can construct 'Communicator' states which travel in one direction (counterclockwise, say) along the edge of a grey area only if it is not shrinking, and can somehow detect when they have traveled once around a stationary grey area, then this detection is the signal to begin the third stage of the algorithm. We should note that we require some room for the communication cells to propagate along; therefore we impose the restriction that no black cells (input points) be placed along the outer edge of the grey cells.

In order to facilitate this communication and detection, we want to start communicating counterclockwise from some cell that is guaranteed to be on the grey cell boundary when it has finished shrinking. Any of the outer black cells meet this requirement. However, we do not want to impose the restriction that one must designate one of the outer black cells as this com-

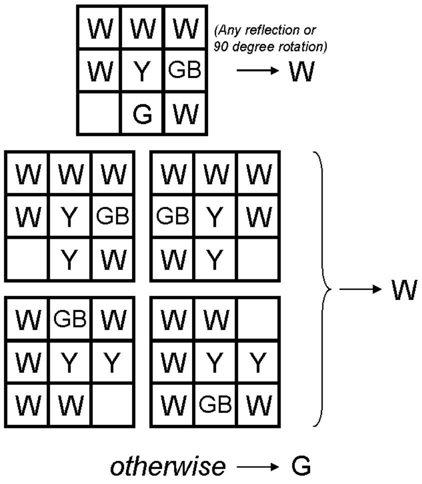


Figure 6: Rules for transitioning from the yellow state to the white or grey state. Y means the yellow state. Blank cells mean that the state of that cell is irrelevant to the rule.

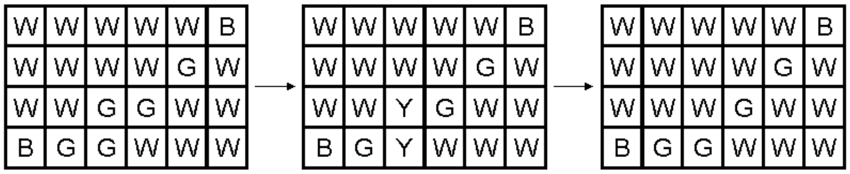
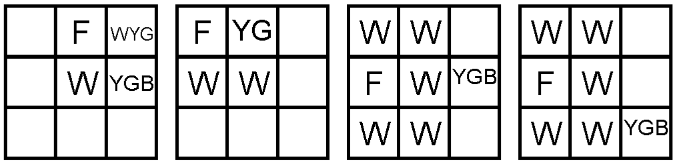


Figure 7: The yellow states in action, allowing the grey area to stay connected in the same situation as was shown in Figure 4.

munication starter cell (let us refer to it as the Anchor cell). We construct the algorithm to automatically 'find' one of the black cells that is on the edge of the grey area and designate it as the Anchor cell.

We introduce a 'Finder' state which travels counterclockwise along the edge of a (possibly shrinking) grey cell boundary. Finder states always transition to white cells. White cells which are adjacent to Finder states change to Finder states in such a way that there can be at most one Finder state in the automaton at any given generation. The Finder state's propagation along the grey boundary stops when it is adjacent to any black cell, since when this happens it means that a suitable Anchor cell has been found. The transition rules for white cells changing to the Finder state are shown in Figure 8.



	F	WYG
	W	YGB

F	YG	
W	W	

W	W	
F	W	YGB
W	W	

W	W	
F	W	
W	W	YGB

Figure 8: Rules for transitioning from the white state to the Finder state. All 90 degree rotations of these rules apply, but their reflections do not, since we only want the Finder state to move in a counterclockwise direction. F means the cell must be in the Finder state. Blank cells mean that the state of that cell is irrelevant to the rule.

We must note that there are two 'dummy' states required to facilitate the initialization of the Finder state; the first dummy Finder state starts in the top left corner of the grid on the first generation, and on the next two generations it transitions to the second dummy state and then to the Finder state. This was just a simple way to have the Finder state start on the third generation in the top left corner, where it will still be able to follow the shrinking grey area while not affecting the shrinking process in any way. The first dummy state (part of the initial configuration) appears as a grey cell to its neighbours and the second dummy state appears as a white cell to its neighbours, so the behaviour of the first stage of the algorithm does not change. The first dummy state could be obtained from the CA boundary conditions applied at the upper left corner and hence this behaviour can be achieved on a standard two-dimensional CA.

When the Finder state is directly adjacent to a black cell, the black cell changes to the Anchor cell state. The conditions for this occurrence are shown in Figure 9. Note the tie-breaking rule; a black cell may see a Finder state directly above it, yet it will not claim itself the Anchor cell if the Finder state has another black cell to its immediate left or right. So in tie-breaking situations, the left and right cells 'win' the Anchor state. The white cells around this interaction also notice that a Finder is adjacent to a black cell and do not continue the Finder state's propagation (this behaviour is implied by the previously mentioned rules of Figure 8). Note that it is impossible for a Finder state to ever be in between two black cells (i.e. to have a black cell on its left and right, or top and bottom) since a grey cell must be in such a location, and grey cells never transition to the Finder state. A small example of the Finder state successfully 'finding' an Anchor cell is given in Figure 10.

We should also note at this point that it is impossible for the grey area to shrink faster than the Finder state can follow it. This is because the Finder state always moves to be directly adjacent to grey cells, and any grey cell that turns white can turn to the Finder state on the next generation since it will have both the Finder cell and the grey cell as neighbours. One of the latter two rules of Figure 8 will apply.

	B	
!B	F	!B

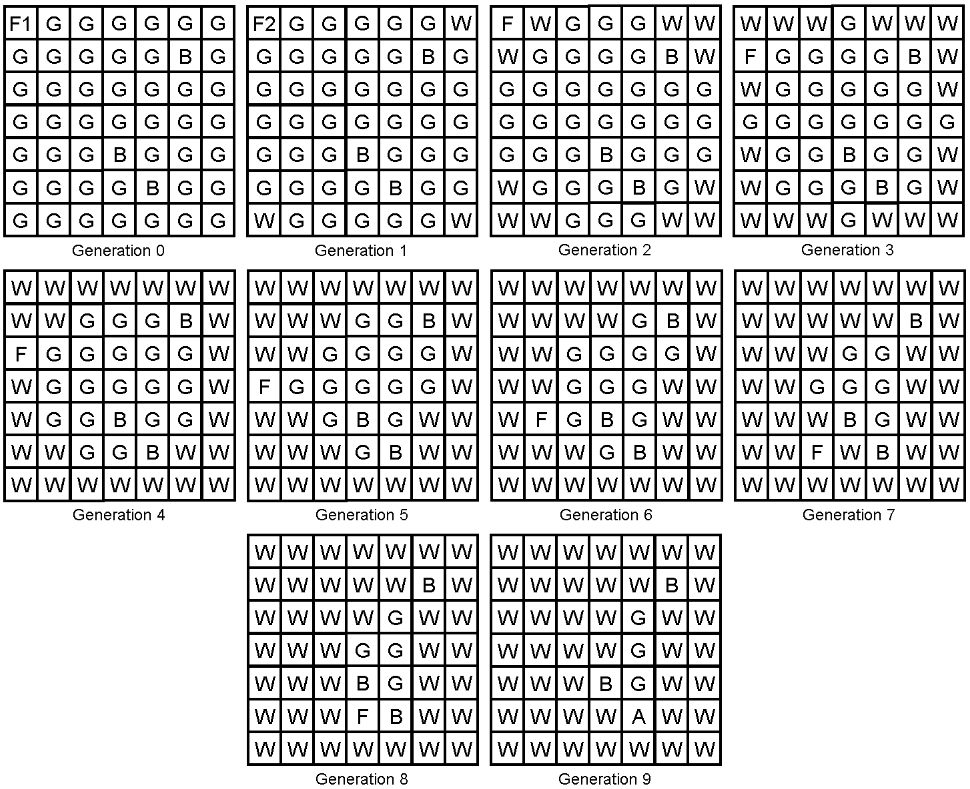
!B	F	!B
	B	

F	B	

	B	F

Figure 9: Rules for transitioning from the black state to the Anchor state. '!B' means that the cell can be in any state except for the black state, and F denotes the Finder state. Blank cells mean that the state of that cell is irrelevant to the rule.

Once the Anchor cell is found, the communication begins. The goal of the communication is to send a signal counterclockwise around the edge of the grey area, and this signal may only propagate if the grey edge it is travelling along is not in the process of shrinking. The Anchor cell changes to one of 8 states based on the white cells around it (the Finder state is considered a white cell as well, since it will be turning white on the next generation anyway). Each of these states defines a different transmitter and receptor, as shown in Figure 11. The transmitter and receptor labels are not states - they are simply there to clarify the mechanism for starting and stopping communication. The T and R labels just signify that white cells who notice one of T1-T8 in a certain location in their neighbourhood will recognize themselves as the transmitter or receptor and will therefore abide by different state transition rules. That is, when a cell D is in the white state and the cell to the lower right of D is in state T1, the cell D transitions to the communication state. The same happens if the state to the left of D is in state T2, and so on. Figure 11 attempts to explain the intuitive idea behind the construction, since if written out in full, the rules are slightly more complicated. The behaviour of our cellular automaton guarantees that



Generation 0

F1	G	G	G	G	G	G
G	G	G	G	G	B	G
G	G	G	G	G	G	G
G	G	G	G	G	G	G
G	G	G	B	G	G	G
G	G	G	G	B	G	G
G	G	G	G	G	G	G

Generation 1

F2	G	G	G	G	G	W
G	G	G	G	G	B	G
G	G	G	G	G	G	G
G	G	G	G	G	G	G
G	G	G	B	G	G	G
G	G	G	G	B	G	G
W	G	G	G	G	G	W

Generation 2

F	W	G	G	G	W	W
W	G	G	G	G	B	W
G	G	G	G	G	G	G
G	G	G	G	G	G	G
G	G	G	B	G	G	G
W	G	G	G	B	G	W
W	W	G	G	G	W	W

Generation 3

W	W	W	G	W	W	W
F	G	G	G	G	B	W
W	G	G	G	G	G	W
G	G	G	G	G	G	G
W	G	G	B	G	G	W
W	G	G	G	B	G	W
W	W	W	G	W	W	W

Generation 4

W	W	W	W	W	W	W
W	W	G	G	G	B	W
F	G	G	G	G	G	W
W	G	G	G	G	G	W
W	G	G	B	G	G	W
W	W	G	G	B	W	W
W	W	W	W	W	W	W

Generation 5

W	W	W	W	W	W	W
W	W	W	G	G	B	W
W	W	G	G	G	G	W
F	G	G	G	G	G	W
W	W	G	B	G	W	W
W	W	W	G	B	W	W
W	W	W	W	W	W	W

Generation 6

W	W	W	W	W	W	W
W	W	W	W	G	B	W
W	W	G	G	G	G	W
W	W	G	G	G	W	W
W	F	G	B	G	W	W
W	W	W	G	B	W	W
W	W	W	W	W	W	W

Generation 7

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	G	G	W	W
W	W	G	G	G	W	W
W	W	W	B	G	W	W
W	W	F	W	B	W	W
W	W	W	W	W	W	W

Generation 8

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	G	G	W	W
W	W	W	B	G	W	W
W	W	W	F	B	W	W
W	W	W	W	W	W	W

Generation 9

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	A	W	W
W	W	W	W	W	W	W

Figure 10: The first few generations of a small test case showing the Finder state identifying a particular black cell as the Anchor state. Note that the outer boundary of white cells (part of the initial configuration) is not shown. F1 and F2 refer to the two 'dummy' finder states.

any configuration can have only one occurrence of the states T1-T8, however, formally the rules for the white cell D need to define what happens in cases where more than one of the neighbours of D is in states T1-T8.

It does not matter which of the 8 states the Anchor cell chooses to transition to, as long as both the transmitter and receptor cells are white. So as soon as the Anchor state is found, it transitions to one of the T1-T8 states. We should note at this point that the algorithm prototype includes an explicit rule for transitioning to each of the T1-T8 states: the Anchor cell will transition to T1 if the top-left and top cells of its neighbourhood are in the white state, but if this is not the case then it will transition to T2 if the

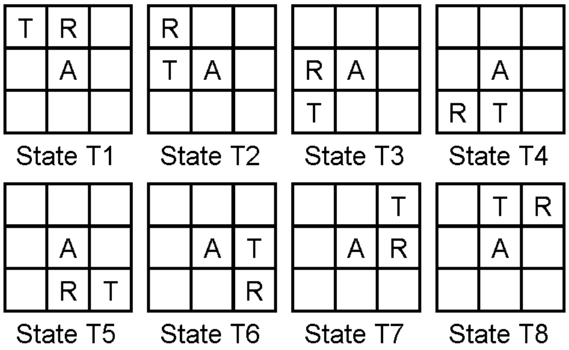
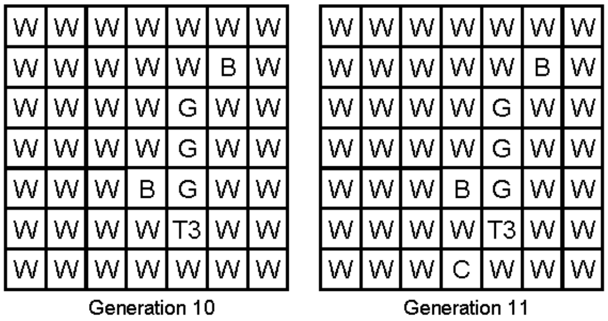


Figure 11: T1-T8 are the possible states that the Anchor cell may transition to. T and R refer to transmitter and receptor, which are not states, simply cells which must be in the white state in order for the transition to occur, and whose behaviour after the transition changes.

left and top-left cells of its neighbourhood are in the white state. That is, the rule transitioning into T2 requires that the top cell is not white, and the top-left and left cells are both white. Otherwise it will try to transition to T3, etc. Because of the way the grey area shrinks, we are guaranteed to always be able to transition to at least one of T1-T8, since there will always be at least two adjacent white cells in the neighbourhood of the Anchor cell.

Depending on which state the Anchor cell changes to, the white cells around it know if they are supposed to start the transmission. The white cell who notices that it is the transmitter (say, if its top-right neighbour is in state T3) changes to the Communication state on the next generation. An example is given in Figure 12, which shows the next two generations of the computation continued from Figure 10.

The Communication state always changes to a state which corresponds to the sum of the grey and black neighbours it can count in its neighbourhood (the T1-T8 states count as a black state in this sum). From this state, if the number of grey or black (GB) neighbours has changed, it transitions to the white state. Otherwise, it transitions back to the Communication state. This ensures that Communication states cannot continue to propagate along a grey area that is still in the process of shrinking, because there will be changes in the number of GB neighbours along the edge. So the states involved in communication are constantly flipping back and forth between their 'GB sum' state and the communication state.



W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	W	W	W	W

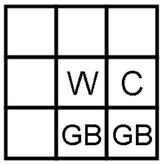
Generation 10

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	C	W	W	W

Generation 11

Figure 12: An example of a white cell starting the counterclockwise communication. It notices that the T3 state is its upper-right neighbour, and so it transitions to the Communication state.

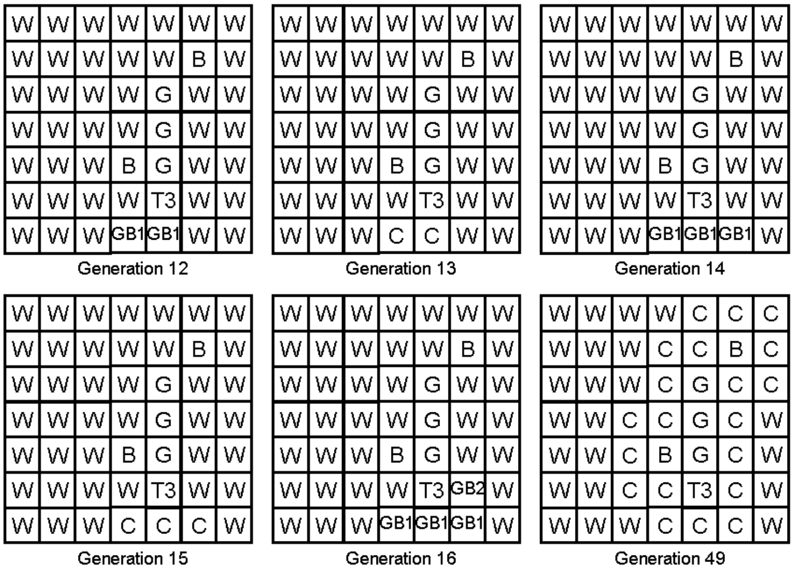
The communication goes counterclockwise from the transmitter cell. The rule shown in Figure 13 defines the conditions under which a cell in the white state will transition to one of the 'GB sum' states, a state which corresponds to the total number of a cell's GB neighbours. Note that we only need 4 of these 'GB sum' states, since if a cell has more than 4 grey or black neighbours it is clear that the first stage of the algorithm must not be complete, and therefore communication must not continue (the cell in question transitions to the white state). Also, if the cell has zero grey or black neighbours then it is clear that again, the first stage of the algorithm is not complete and the cell must transition back to the white state.



	W	C
	GB	GB

Figure 13: Rule for transitioning from the white state to a state corresponding to the total number of grey or black states in a cell's neighbourhood. All 90 degree rotations of this rule apply. C refers to the communication state. Blank cells mean that the state of that cell is irrelevant to deciding whether this rule should be applied, but if the blank cells in this figure are in the grey or black state, then clearly they will influence which 'GB sum' state that the white state will transition to.

Only four of these GB sum states are necessary, since a candidate communication cell must have at least one GB neighbour and at most 4 - any other situation signifies a transition to the white state, cutting off communication temporarily. If the GB sum stays the same on the next generation, then the cell goes back to the communication state, allowing the communication signal to propagate further while ensuring that the edge of the grey area is not changing. If the GB sum changes at all on the next generation, the cell will go from the GB sum state to the white state. Continuing the example from Figures 10 and 12, the operation of the Communication and GB sum states are shown in Figure 14.



W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	GB1	GB1	W	W

Generation 12

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	C	C	W	W

Generation 13

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	GB1	GB1	GB1	W

Generation 14

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	W	W
W	W	W	C	C	C	W

Generation 15

W	W	W	W	W	W	W
W	W	W	W	W	B	W
W	W	W	W	G	W	W
W	W	W	W	G	W	W
W	W	W	B	G	W	W
W	W	W	W	T3	GB2	W
W	W	W	GB1	GB1	GB1	W

Generation 16

W	W	W	W	C	C	C
W	W	W	C	C	B	C
W	W	W	C	G	C	C
W	W	C	C	G	C	W
W	W	C	B	G	C	W
W	W	C	C	T3	C	W
W	W	W	C	C	C	W

Generation 49

Figure 14: An example of the communication states in action. Note that the T3 state here counts as a black cell in the computation of the 'GB sum' state transition.

The total number of states needed for this stage is 17; 3 states to find an Anchor cell (2 dummies and the actual Finder state), 9 states for the Anchor cell (8 for the various possible positions of the transmitter and receptor, and 1 is the actual Anchor state), and 5 communication states (1 Communication state, and 4 'grey/black sum' states).

3.3 Stage 3

Once the communication makes it all the way around the static grey area, the Anchor cell (which is in one of states T1-T8) notices that its receptor cell has changed to the Communication state, and so it changes to a 'Detect' state. All cells which are either grey or black that have a detect state anywhere in their neighbourhood also turn into the Detect state. Communication type cells change to the white state if there is a Detect state in their neighbourhood. White cells in the presence of Detect states follow the rules outlined in the expanding stage of the algorithm discussed in Torbey's paper [5] - Adamatzky's method with a slight modification. The basic rule is that any white cell with 4 or more neighbours in the Detect state will transition to the Detect state. The modification to this scheme is simply that all rotations/reflections of the rule in Figure 15 also call for a change to the Detect state - this rule simply catches the special cases where the grey area is connected only by a thin 'thread' of cells on a diagonal.

	W	D
D	D	W

Figure 15: A rule for changing from the white state to the Detect state. All rotations/reflections apply.

It should be clear that it is irrelevant to the correctness of the algorithm that the Detect states propagate outwards from a certain point instead of occurring all at once, globally. An example of this expansion of the grey area to the 45-convex hull is given in Figure 16.

3.4 Time Complexity

Our algorithm has introduced many additional states to the algorithm by Torbey [5]. We briefly compare the time complexity of the two algorithms.

In the following analysis we are assuming the input is given as a square grid having edges of length m. In this case the time used by the algorithm from [5] is at most $3.5m$.

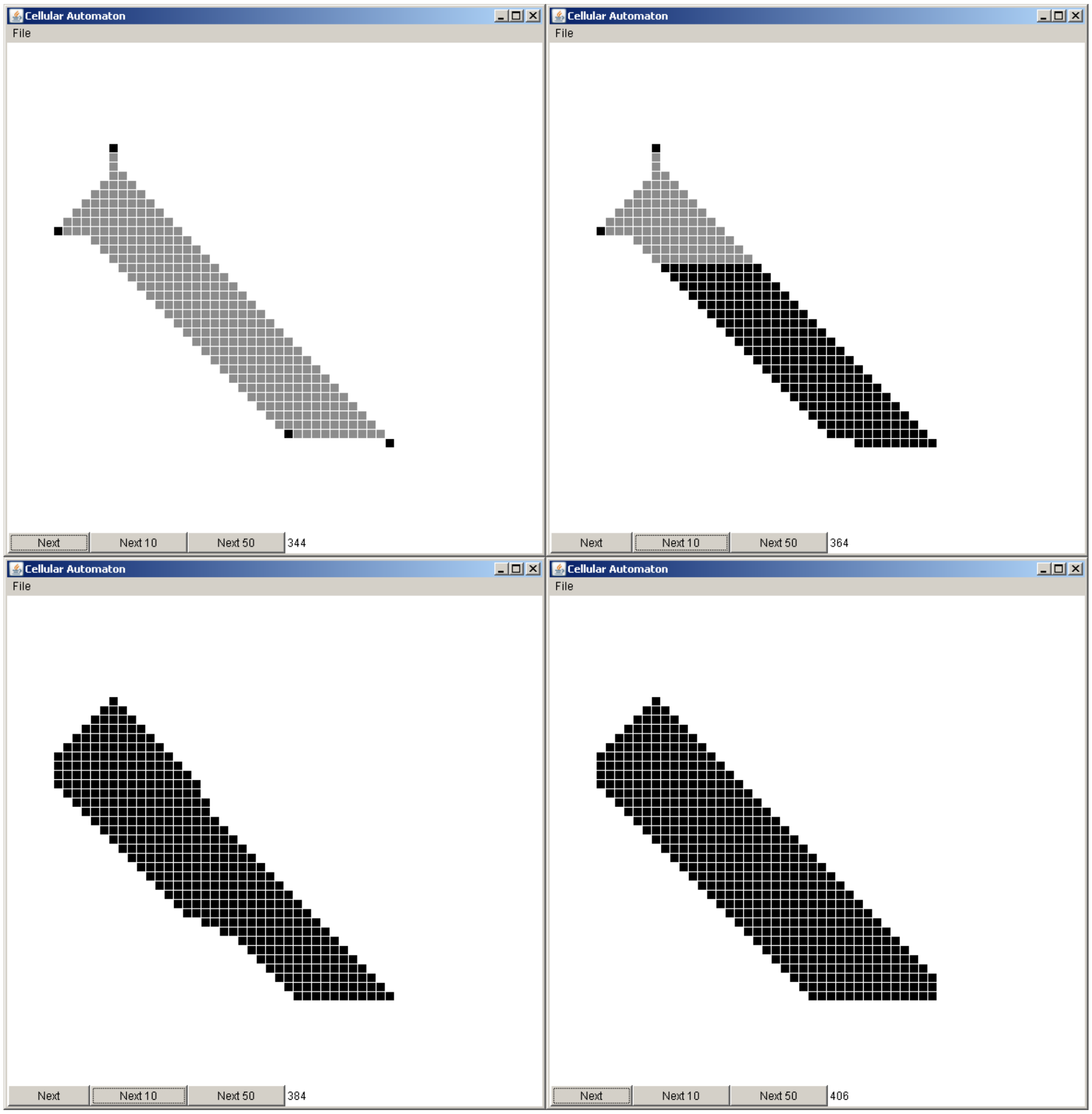


Figure 16: An example of the third stage of the algorithm working. The communication states are not shown here. Detect states are the same colour as the black state for simplicity. In this example the detection occurred at the bottom left black cell.

The shrinking phase of the algorithm from [5] uses time at most $2.5m$. In order to deal with the discussed problematic inputs, our algorithm sometimes uses yellow intermediary states when deciding whether to transition from a grey state to a white state. So using a very conservative estimate, the running time of Stage 1 is upper bounded by $5m$.

Stage 2 of the algorithm sends a communication signal around the grey area that is possibly still shrinking. Note that if the grey area continues shrinking, this disrupts the communication signal and potentially increases the time bound. But since we know that the shrinking must have stopped after $5m$ steps, when estimating the time needed for the communication phase, without loss of generality we can assume that the grey area has stopped shrinking. The length of the outer edge of the grey area is upper bounded by $4m$ which means that the communication signal takes time at most $8m$. (The signal alternates between the communication state and 'GB sum' states which means that it travels one cell in two cycles.) However, the communication signal travels slowest along diagonals (since it does not travel diagonally, only horizontally and vertically) so we must consider the case where there are two input points in opposing corners, causing the maximum amount of diagonal travel. In this case the algorithm would also take at most $8m$ cycles to communicate around the diagonal. There is one last issue to consider for this stage: the Finder state may not find an Anchor cell by the time the first stage has finished. Experimental evidence as well as intuition seem to indicate that this can only happen if the first stage of the algorithm finishes very quickly, and hence this case would not change our upper bound for Stage 2. If any counterexample were to exist though, it would take time at most m for the Finder state to find an Anchor cell. We conclude that Stage 2 of our algorithm takes time at most $9m$.

Finally, Stage 3 of our algorithm is very similar to the expanding stage in [5] which is completed in time m. The only difference is that our algorithm may take at most an additional m cycles to distribute the Detect state throughout the grid.

By combining the above estimates, we note that the total running time of our algorithm is upper bounded by $16m$. Here we have just wanted to establish that the running time remains linear in m. By using a more detailed analysis, the upper bound estimate could clearly be improved. For example, the upper bound for Stage 1 corresponds to a situation where the grey area initially shrinks to be very small, whereas the upper bound estimate for the next two stages uses a worst case example where, even after the shrinking has stopped, the grey area remains relatively large.

4 Conclusion

In summary, the two-stage algorithm proposed by Torbey [5] has been modified so that it does not require a global rule change to transition between the two stages, has simpler rules for shrinking the grey area in the first stage, and keeps the grey area connected in a not-so-obvious special case. A mechanism has been described for finding an anchor point (a point on the edge of the grey area), communicating around the grey area in such a way as to ensure that it has not changed, and detecting this successful communication to enable a smooth transition to the last stage of the algorithm.

References

[1] Andrew Adamatzky. Automatic programming of cellular automata: identification approach. *Kybernetes: The International Journal of Systems and Cybernetics*, pages 26(2):126–135, 1997.

[2] Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry*, 16:361–368, 1996.

[3] Howard Gutowitz. *Cellular Automata: Theory and Experiment.* MIT Press/Bradford Books, Cambridge Mass., 1991.

[4] Tommaso Toffoli and Norman Margolus. *Cellular automata machines: a new environment for modeling.* MIT Press, Cambridge, MA, USA, 1987.

[5] S. Torbey and S.G. Akl. An exact and optimal local solution to the two-dimensional convex hull of arbitrary points problem. To appear in the International Journal of Cellular Automata.

[6] Stephen Wolfram. *Cellular Automata and Complexity.* Addison-Wesley, 1994.

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

[8] Andrew Wuensche and Mike Lesser. *The Global Dynamics of Cellular Automata.* Addison-Wesley, 1992.