Parallel Processing Letters, Vol. 10, No. 4 (2000) 383–393

CELLULAR GRAVITY

FRÉDÉRIC GRUAU
Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier, 161 rue Ada, 34392 Montpellier, France
E-mail: gruau@lirmm.fr

JOHN TROMP
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
E-mail: tromp@cwi.nl

Received November 1999
Revised December 2000
Accepted by Y. Robert

ABSTRACT

We consider the problem of establishing gravity in cellular automata. In particular, when cellular automata states can be partitioned into empty, particle, and wall types, with the latter enclosing rectangular areas, we desire rules that will make the particles fall down and pile up on the bottom of each such area. We desire the rules to be both simple and time-efficient. We propose a block rule, and prove that it piles up particles on a grid of height h in time at most $3 * h$.

Keywords: Cellular automata, gravity, piling, algorithm, time complexity.

1 Motivation

Our overall direction is to perform general purpose computation on a fine grained, massively parallel computer. The interconnection topology is a 2D grid (so as to be highly scalable) and the behavior of each Processing Element is simple enough to be modeled as a cell of a 2D Cellular Automaton (CA). Each cell can be empty, contain a particle, or represent a wall. The wall cells serve to divide the CA space into rectangular regions.

The most basic operation in this model is region division. When a region divides, its rectangle is divided in two sub-rectangles, generally of approximately equal area. Particles in the original rectangle may choose to migrate to either sub-rectangle, or duplicate, one copy migrating to each sub-rectangle.

Because division is basic and happens all the time, it must be implemented as fast as possible. For implementation, we can consider only one sub-rectangle, say the lower one if the division is vertical, and consider only the particles migrating

*see 2nd author for address

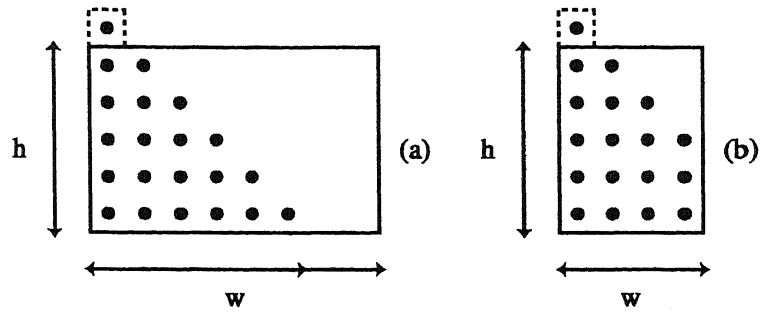


Fig. 1: piling up beyond the height of a rectangle

to it. We are left to solve the problem of quickly migrating particles contained in a rectangle of size $h \times w$ towards a smaller sub-rectangle of size $h' \times w$. If we label empty by 0 and full by 1, we can apply an algorithm that sorts the 0's and 1's on a 2D grid, in a snakelike order; once sorted, all the 1's (and therefore all the particles) are at the bottom. Leighton [1], in his seminal book, describes such an algorithm that needs $3h + O(h)$ steps (assuming $h = w$). However, the algorithm is quite complex: it involves 8 phases and needs to compute $\sqrt[8]{h}$. We want to find simple CA rules so as to minimize the silicon area needed in each cell to process migration.

If the number n of particles is exactly equal to the area $h' \times w$ of the sub-rectangle, then indeed, solving the migration problem is equivalent to sorting. However, if

1. $h' \geq w/\sqrt{2}$
2. $n \leq h' \times w/2\sqrt{2}$

then piling up the particles is sufficient to make them fit in the lower sub-rectangle. Recall that a piled particle needs support from the cell below it and the left and right neighbours of the latter, meaning none of these three cells can be empty. Figure 1 illustrates the exact maximum number of particles that can be piled depending on whether $w < h'$ or not.

If $w > h$ the number of particle needed to exceed the rectangle is $\frac{1}{2}(h+1)(h+2)$ (b) if $w \leq h$ it is $w(h+1) - \frac{1}{2}w(w-1) = w(h - \frac{w}{2} + \frac{3}{2})$ The first condition says that the rectangle cannot be too narrow. In our computing model it's desirable to keep the rectangles as square-like as possible, and we choose to maintain the relation $w \times \sqrt{2} \geq h \geq w/\sqrt{2}$.

The second condition says that the density of particles must not exceed $1/2\sqrt{2}$. This we also assume to be enforced. Thus the problem of migration is reduced to that of piling.

2 A simple piling up cellular automaton

In this section, we present a simple CA that solves the piling-up problem. We use a *partitioned cellular automaton* as introduced by Toffoli and Margolus [2]. This is merely syntactic sugar to describe an automaton which uses part of its state to

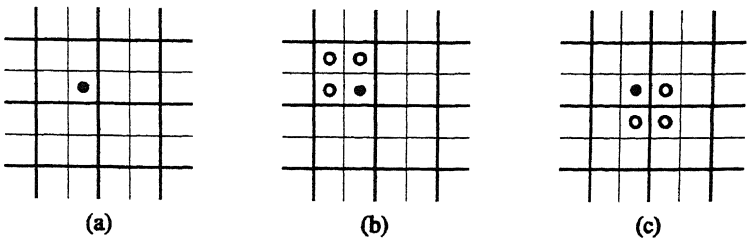


Fig. 2: The 2×2 block of the Margolos neighborhood. Consecutive steps alternate between the even grid (thick lines) and the odd grid (thin lines). Depending on the grid in use, the cell marked in (a) will have as neighborhood either an even-aligned block (b) or an odd-aligned block (c)

keep track of block structure, in a more convenient form, by abstracting away from that state. Expanding our block rules to explicit transition rules for the individual automata is a straightforward but tedious exercise, which we omit.

- The array of cells is partitioned into disjoint blocks of size 2×2
- A block rule is given, that locally updates each block. Our piling-up rule is listed in Table 1.
- The partition alternates between the even grid and the odd grid as shown in Figure 2.

2.1 The rule

Cells can be in one of three possible states: a cell can be empty, (' '), contain a particle ('•') or represent a wall ('*W*'). The wall is rectangular.

DEFINITION 1. The rule DIAG_DOWN is such that in every 2×2 block of the current partition, we try diagonally pushing down particles in the upper half along the diagonals, or, failing that, pushing them down vertically.

More precisely, if the block is $\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$ and `push(x,y)` pushes a particle from x to y if y is empty and x is not; then the DIAG_DOWN algorithm is `(push(a,d) || push(b,c)); (push(a,c) || push(b,d))`.

The block rule is given explicitly below, by listing for each possible state of a block the corresponding new state. The full table consists of 16 entries for particles moving inside the rectangle, plus 24 entries for particles touching the wall.

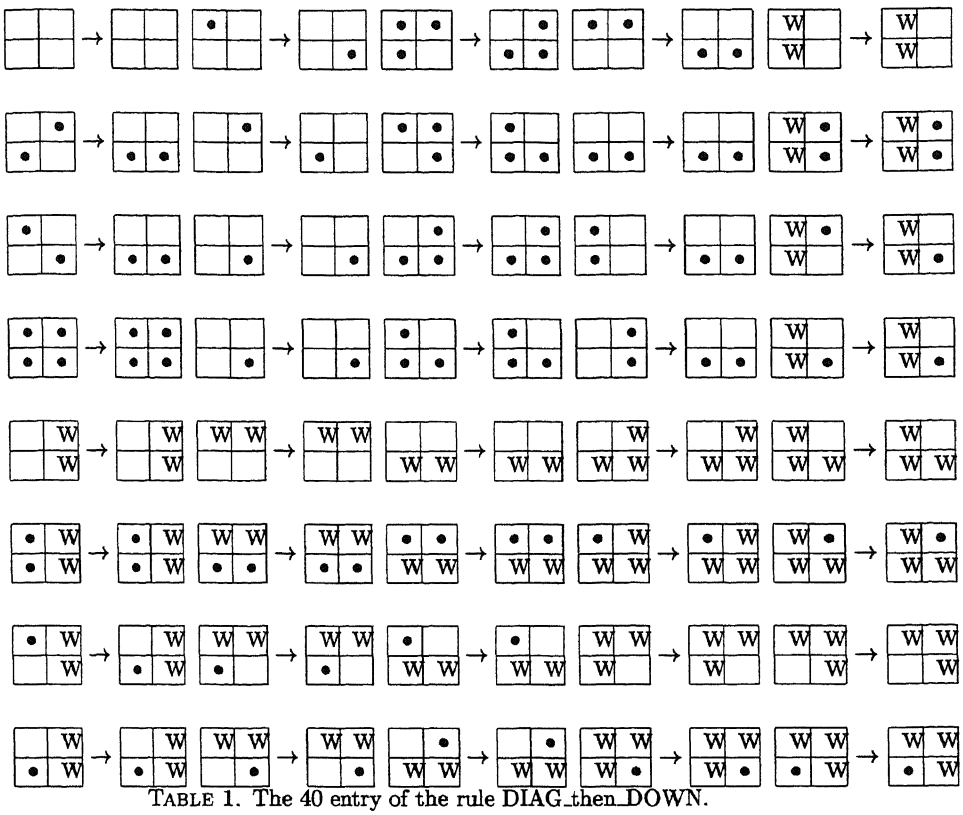
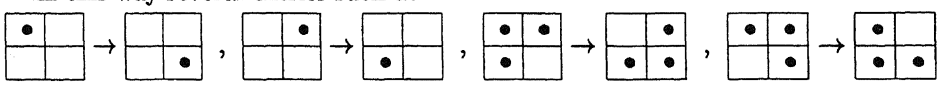


TABLE 1. The 40 entry of the rule DIAG_then_DOWN.

The present rule has both horizontal symmetry and vertical empty↔particle swapping symmetry, which reflects the equivalent alternative view of "piling down " holes from the top.

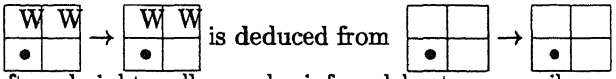
In this way several entries such as



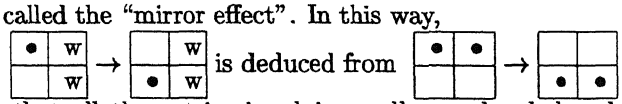
coincide up to one of the these symmetries transformation.

Similarly, entries with an upper (resp. lower) wall are equivalent to corresponding entries with empty upper half (resp. particles in lower half).

For example



Finally, the left and right walls can be inferred by temporarily replacing the wall by a copy of the cell on opposite side, applying the rule, and finally restoring the wall. This is called the "mirror effect". In this way,



It turns out, that all the entries involving walls can be deduced from the other entries.

For clarity and brevity, it will be convenient to use a compressed form of table in which each equivalence class is represented by a single entry. Table 2 show the compressed form (7 entries) of the rule DIAG_DOWN.

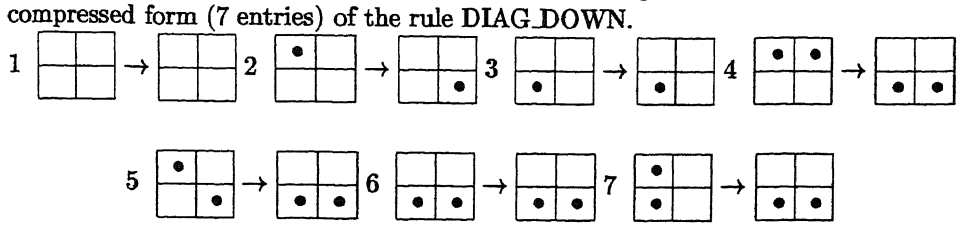


TABLE 2. Summarized form of the rule DIAG then DOWN. We show one entry per equivalence class

3 Analysis of the rule

In this section we analyse the behavior of the rule, and show that for the proof coming in the next section, we can simplify the problem by ignoring the side walls.

3.1 Block Support

We actually cannot achieve full support for each particle, since a cell shares a block with only one of the two cells diagonally below it. Instead we'll aim to achieve 'block support', where particles end up supported by the cell below and the unique cell diagonally below it that it shares a block with.

3.2 Isolated particles

Let us first look at an isolated particle, far away from the wall. For the first step choose the even grid (solid line Figure 2). Whatever its initial position in the block, the rule will force the particle to be in the lower half of the block. Recall that the block partitioning alternates in time between the even grid and the odd grid. So at the next step we use the odd grid. The particle will find itself in the upper half, say the upper left corner of a block and the rule will move it to the lower right corner. At the next step, we are back to the even grid. The particle will again find itself in the upper left corner of a block, and the rule will again shift it in a diagonal-downward motion. Continuing this, alternating the two partitions, each isolated particle will move down on a diagonal, at a uniform rate. Which of the two possible directions it will follow is determined by its initial position. A cell (x, y) has direction $+1$ (resp. -1) if $x - y \equiv 0 \pmod 2$ (resp. $1 \pmod 2$). We assume a block partition in which particles in $+1$ directed cells move towards the lower right, while those in -1 directed cells move towards the lower left.

3.3 Ignoring the vertical walls

What happens when an isolated particle hits the left or the right wall? The rule cannot move it further along the diagonal, therefore it moves the particle down vertically. Subsequently, the particle will resume its diagonal downward movement, but with a direction perpendicular to the direction it had before meeting the wall.

Everything happens as if the particle was bouncing elastically on the wall.

To simplify the analysis, we are going to use a trick which allows us to ignore the left and the right wall. Consider our rectangular grid $\mathcal{G}$ of bounded height h and width w, with an initial configuration $\mathcal{C}_0$ of particles. We denote by $\mathcal{C}'_0$ a copy of $\mathcal{C}_0$. Put $\mathcal{C}_0$ and $\mathcal{C}'_0$ back to back to form a cylinder. When we partition the cylinder in 2×2 blocks, the border blocks with cells containing elements of both $\mathcal{C}_0$ and $\mathcal{C}'_0$, will have horizontal symmetry. Because the DIAG_DOWN rule has horizontal symmetry, the $\mathcal{C}_0$ and $\mathcal{C}'_0$ regions will remain identical, and the border block will retain horizontal symmetry. Because of the way the rule entry involving the left and right wall are defined ("the mirror effect"), it is straightforward to see that the particles in the cylinder move exactly like the particle in $\mathcal{C}_0$. Thus we can study $\mathcal{G}$ as if it had no walls.

3.4 Traffic Jams; extent and support

Consider our isolated particle going towards the lower right corner. Before the rule application, the particle is always in the upper left corner of a block. As long as the lower right corner is empty the particle will be able to go downwards. Such a particle is "*synchronized*" with the alternative *CA* partitions. Traffic-jams start to occur when the lower half of a block is occupied. If, at the initial step, the lower halves of all the blocks are empty, all the particles will start moving downwards. The traffic-jam will appear only at the bottom of the grid, and will accumulate upwards.

Definition 1 *We start the evolution of our CA at time 0, and let iteration i happen between times $i-1$ and i. A particle in row y at time t is said to be* in sync *if $y \equiv t \pmod 2$, and* out of sync *otherwise.*

Definition 2 *A particle is* jammed *in iteration t iff it is in sync at time $t-1$ and out of sync at time t.*

Note that a particle that moves in iteration t is necessarily in sync at time $t-1$ and remains so at time t because of the downward move. It follows that a particle that is out of sync at time t is necessarily in sync at times $t-1$ and $t+1$. It turns out a jam implies the occurance of particles at particular points in space-time. We next define the notion of an extent as a subset of space-time and show that it is induced by a jam.

Definition 3 *Let (x,y) be a cell with direction d. The* extent *of a jam at (x,y) in iteration t is the set of points*

$$\{(x+di, y-i, t-1-i) | 1 \le i \le y\} \cup \{(x+di, y-i, t-i) | 1 \le i \le y\} \cup$$
$$\{(x+di, y-i-1-2j, t-i) | 1 \le i \le y, 1 \le j < y/2\} \cup$$
$$\{(x-di, y-i-1-2j, t+i) | 0 \le i \le y, 1 \le j < y/2\}$$

The top set of points is called the border *of the extent. The extent has direction d.*

Figure 3.4 shows the projection on space of an extent $(6,4,t)$, where 'J' is the jam causing the extent.

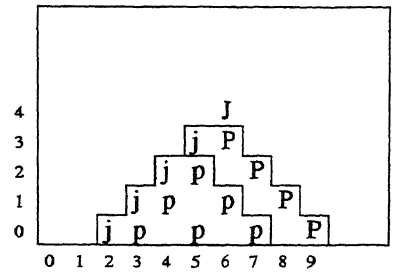


Fig. 3: extent $(6,4,t)$

Lemma 1 *A particle p jammed at (x,y) in iteration $t \geq y$ implies the extent (x,y,t).*

Proof. (By induction on height.) Let d be the direction of (x,y). If $y = 0$ then the extent is empty and thus trivially implied. Suppose $y > 0$. Then by assumption of the lemma also $t > 0$. For p to be jammed in iteration t, its downward motion must be blocked by particles at both $(x+d, y-1)$ and $(x, y-1)$ at time t (the 'j' and 'p' on row 3 in Figure 3.4). The particle at $(x+d, y-1)$ must have been present at time $t-1$ as well, and was out-of-sync at that time, hence jammed in iteration $t-1$. By induction there is an extent $(x+d, y-1, t-1)$ (whose projection on space is shown as the sub-triangle in Figure3.4). The particle at $(x, y-1)$ is in sync at time t and, having the opposite direction of p, will either move to $(x-d, y-2)$ itself, or else be blocked by another particle. In either case, there must be a particle at $(x-d, y-2)$ at time $t+1$. The exact same reasoning can be repeated to establish existence of particles at $(x-id, y-1-i, t+i)$ (The capital'P's in Figure3.4). The union of these with the particle $(x+d, y-1, t-1)$ and its established extent exactly constitute the extent claimed in the lemma. □

The notion of extent helps us prove that particles stabilize in due time. At each step, a particle will either continue moving down, or get jammed, in which case we can show an accumulation of particles underneath it. In both cases, the particle gets closer to stabilization.

3.5 Extents sediment

We need some more definitions to be able to work with the 2×2 blocks that partition the grid.

Definition 4 *A block is a* child *of either of the two blocks immediately and diagonally above it, which are it's* parents. Ancestors *and* descendants *are defined accordingly.*

Definition 5 *Two extents are said to be congruent if they have the same direction and have a non-empty intersection containing non-border particles.*

Definition 6 *The support of a 2x2 block with lower-left cell (x,y) is the set of cells*

$$\{(x+1+i, j) | 0 \leq i \leq y, 0 \leq j \leq y-i\} \cup$$

$$\{(x-i,j)|0 \leq i \leq y, 0 \leq j \leq y-i\},$$

and has height $y+1$. *An extent is said to arrive at a block* B *at time* t *if* t *is the earliest time at which either cell in the bottom half of* B *occurs in the non-border part of the extent.*

The following lemma shows that the layers will indeed "sediment".

Lemma 2 *If* $2y$ *incongruent extents arrive at block* B *before or at time* t, *where* y *is the height of the support of* B, *then at time* t, *the support has stabilized, i.e., is full of particles.*

Proof. By induction on y. The case $y = 1$ holds trivially because height 1 supports consist of only 2 cells and, by incongruence, one gets filled by each arriving extent. Suppose $y > 1$. Let E, E', E'', with directions d, d', d'', be the last 3 extents to arrive, at times $t \geq t' \geq t''$. Let $B_{-d'}$ at height $y-1$ be the child block of B in direction $-d'$. We will show that the support of $B_{-d'}$ has stabilized at time $t'-1$. Since at most 2 incongruent extents can arrive in a block at a time, and since all arrival times must have the same parity, there must be at least $2y-3$ extents having arrived at B before time $t'-1$, and these arrive at $B_{-d'}$ before or at time $t'-1$. If $t'' < t'$ then E'' also arrives at $B_{-d'}$ before or at time $t'-1$. Otherwise $t'' = t'$ so that $d'' \neq d'$ but then E'', which goes 'back in time' in direction $d'' = -d'$, also arrives at $B_{-d'}$ at time $t'-1$. In both cases induction implies stabilization of $B_{-d'}$'s support at time $t'-1$. The non-border particle of E' that is in B's bottom half at time t' is necessarily at the intersection with $B_{-d'}$ and thus will remain blocked there. Let p be the non-border particle of E that is in B's bottom half at time t. The exact same reasoning applied to E instead of E' (using corresponding variables with one fewer $'$) shows that p remains blocked at the intersection with B_{-d}. If $d \neq d'$ then B's support is stabilized at time t and we're done. Otherwise $d = d'$. Then, by incongruence, $t > t'$.

There must be $2y-1$ extents having arrived at B before or at time $t' \leq t-2$, and these arrive at $B_{d'}$ before or at time $t-1$. Induction implies stabilization of $B_{d'}$'s support at time $t-1$. If p is adjacent to a jammed 'j' particle in E, then the latter will necessarily be at the intersection with $B_{d'}$ at time $t-1$ and remain blocked there. If p is not adjacent to the border of E, then at time $t-1$ there is a particle in E diagonally above p that tries to move toward p in iteration t. This implies occupation of the cell below it at time t, again showing stabilization of B's support at time t. □

3.6 Blocks stabilize over time

Definition 7 *At any given time, a* top block *is a block* B *such that*

- B *has one or two particles in its top row*
- *no ancestor of* B *has any particles in its top row*

A block trajectory *is a mapping* B *from times* $t_0, t_0+1, \ldots, t_0+m$ *to top blocks, such that*

- t_0 *exceeds the height of* $B(t_0)$.
- *for all* $t_0 \leq t \leq t_0 + m$, $B(t+1)$ *is a descendant of* $B(t)$.

The instability *of a block is twice its height minus the number of extents that have arrived there.*

Lemma 3 *The instability of blocks in a block trajectory* $B(t)$ *decreases with time.*

Proof. Suppose block $B = B(t)$ is active in iteration $t+1$. In the worst case, there are 2 particles in the top row, one of which moves down while the other jams. Then only one extent arrives, while $B(t+1)$ necessarily equals $B(t)$. Since it is inactive in iteration $t+2$, its instability may not decrease. We will see how it makes up for this. In iteration $t+3$, B is active again. If the single particle in the top row still cannot move down, it is because of a double jam, which compensates for the next inactive iteration. When the single particle in the top row finally does move down in iteration $t+1+2k$, we have $B' = B(t+1+2k)$ being a proper descendant of B, and all extents that arrived at B necessarily arrive at B' as well. Altogether the instability has dropped by 1 in iteration $t+1$, and by (at least) 2 in iterations $t+3, t+5, \ldots, t+1+2k$, for a total of $2k+1$. □

Theorem 1 *A rectangle of height* h *stabilizes within time* $3h$.

Proof. By the previous lemma, it suffices to establish the existence of a block trajectory $B(t)$ that starts before time h and ends with one of the last blocks to stabilize. This is because the instability of any block starts out as $2h$, where h is its height, and therefore any trajectory from it yields a stabilized block within at most $2h$ steps. Indeed, if t_Ω is the first time at which all is stabilized, then we can choose $B(t_\Omega - 1)$ to be any top block that is not yet stabilized. Given a trajectory from times t to t_Ω, we can extend it backwards in time by mapping $t-1$ to any ancestor of B that is a top block at time $t-1$. Note that this necessarily exists (and might equal B). Repeating this until $t < h$ proves the result. □

4 Discussion of the result

We implemented a computer simulation with grid sizes ranging from 10×20 to 40×80 or 40×40. We observe that:

(i) A random distribution of particles with density $1/2$ has an average stabilization time of $h/2$

(ii) In the extreme case where the left half of the grid is full, and the right half is empty, the piling time is $< h if h > 2w$

Hence our upper bound of $3h$ does not meet the "experimental" upper bound of $2h$. Furthermore, in the average case, piling up seems to be "as quick as possible" taking the time $h/2$.

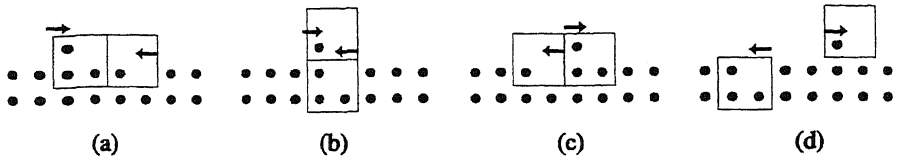
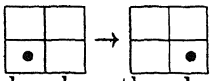


Fig. 4: With the rule DIAG_DOWN_SIDE, a hill going rightward can cross a valleys going leftwards without anhilation of the two. Thin lines represent blocks.

5 Two open problems

5.1 *Removing dirty rows*

Intuition suggests that modifying rule entry number 3 (see Table 2) of DIAG_DOWN to be



could initiate a transfer of particles along the edge of the piles and "flatten down" all the piles to become horizontal in a time $O(n)$ if n is the number of particles.

We can call the modified rule "DIAG_DOWN_SIDE". This means: we try to move down along the diagonals; if we can't, we try moving down vertically; and if that too fails we we try going sideways.

Computer simulation shows that the piling time is the same as for DIAG_DOWN, and the piles are indeed flattened after less than n more iterations of the same rule. At the end the number of dirty rows (with both full and empty cells) is at most 1 for odd width and at most 2 for even width.

Figure 4 shows how two dirty rows can remain dirty forever. Using a slightly refined rule to suppress this exceptional case, it is possible to limit the number of dirty rows to one.

In the context of our cellular machine it is quite interesting to reduce the number of dirty rows. It can ensure that we use the minimal area possible, especially, when we need to fit a small number n of particles in a rectangle whose width is greater than n. With the rule DIAG_DOWN_SIDE, the height needed to always fit the particles is $\sqrt{n}$. With DIAG_DOWN_SIDE height 1 suffices.

5.2 *Filaments*

We believe that there is a critical storage capacity for each cell sufficient to store particles for any programs. Nevertheless, this remains to be proven. For reasons of efficiency it is desirable to be able to spread one particle among several processors. In this way we can have a minimal storage size of 1 kbit per cell, and use a group of cell to store unusually big particles. The simplest way we can think of for structuring the data is to represent a big particle as a chain (called filament) of small particles which fit on a cell. We need to store the filament structure in the CA state and define a rule able to migrate filaments. Simulation can then suggest time bounds to be proven.

6 Conclusion

In this work, we have defined a new class of problems for 2D Cellular Automata called "piling up". This operation is the core mechanism of a new model of massive-fine grain parallelism, briefly outlined. It is crucial to implement it with minimal hardware and optimal time. We have provided an extremely simple CA rule for piling up particles, and proved a near-optimal time upper bound. We have proposed another rule to flatten piles (without proof) and suggested the more difficult open problem of piling up filaments.

Acknowledgements

Thanks to Minnie Middelberg for editing and typing, and Tobias Baanders for the pictures.

References

1. F. Thomson Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees & Hypercubes (Morgan Kaufmann Publishers, 1992).
2. T. Toffoli and N. Margolus, Cellular Automata Machines: A New Environment For Modeling (MIT Press 1987).