Applications of Cellular Automata

Ada YUEN and Robin KAY

1 Introduction

The term cellular automata refers to a rather broad class of computational system. There are a number of common features, but the only universal property is that they are comprised of a number of discrete elements called cells. Each cell encapsulates some portion of the state of the system. Typically, the cell population is homogeneous, each one encapsulating an equal portion of the state, and arranged spatially in a regular fashion to form an n-dimensional lattice as shown in figure 1.

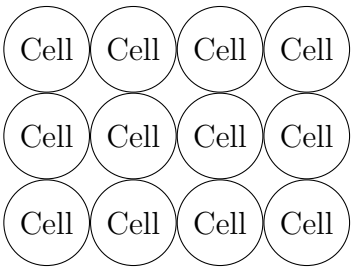

Figure 1: A 2-dimensional lattice of cells

As in any stateful system, useful work is accomplished by encoding problems into the state of a lattice. The solving method is expressed as a series of state transformations and the solution is extracted from the final lattice state. The strength of cellular automata lies in the lattice's uniformity as it facilitates the application of a particular type of state transformation, the local update rule.

A local update rule specifies how to calculate new values for a single cell's variables based on the current state of the cell and its local neighbours on the lattice. Uniformity means that such a rule is applicable to every cell. The local structure of lattice is identical at every point with the exception of the boundary, where a condition is needed to handle the case where a

cell's neighbourhood lies partially off the edge. This can be handled by, for example, wrapping the lattice at the edges.

Updates are typically organised into strict generations such that a complete set of new values are calculated using the current state before that state is overwritten with the new values. This results in a number of beneficial properties. The update of each cell is independent of the other updates in its generation and hence the processing can be partitioned into tasks which run concurrently. Updates only depend on the neighbouring cells and so the representation can also be partitioned. Partitions only need to communicate the updated boundary states to the neighbouring partitions. This reduces the communication bandwidth required to a constant with respect to the size of the lattice and makes simulating the automata embarrassingly parallel.

Cellular automata can serve as good models for system which share their homogeneity and locality, such as physical systems. Indeed, Zuse [1969] proposed that the underlying model of the universe is a cellular automata populated by digital particles. Quantum mechanics aside, physical phenomenon are fundamentally local in scope and require time in order to propagate through space. This is paralleled in automata by typical update rules which only consider the state of a cell's immediate neighbours. An effect can only reach one cell further from its source with every generation. This is in analogue to the speed of wave propagation (e.g. light) or a macroscopic equivalent in coarser problem domains.

While cellular automata can be applied to any problem, not all fit the mould. Constructing a Turing machine equivalent out of cells will not make a serial algorithm any more parallel. Overwhelmingly, automata have found application as simulations where the spatial relationships between cells correspond to real-world spatial relationships. This report two different applications, traffic modelling and structural design, which belong to this family. A third, music composition, is contrasted as one of the few applications which doesn't. These applications occupy different points on the intersecting spheres of simulation and design as shown in figure 2.

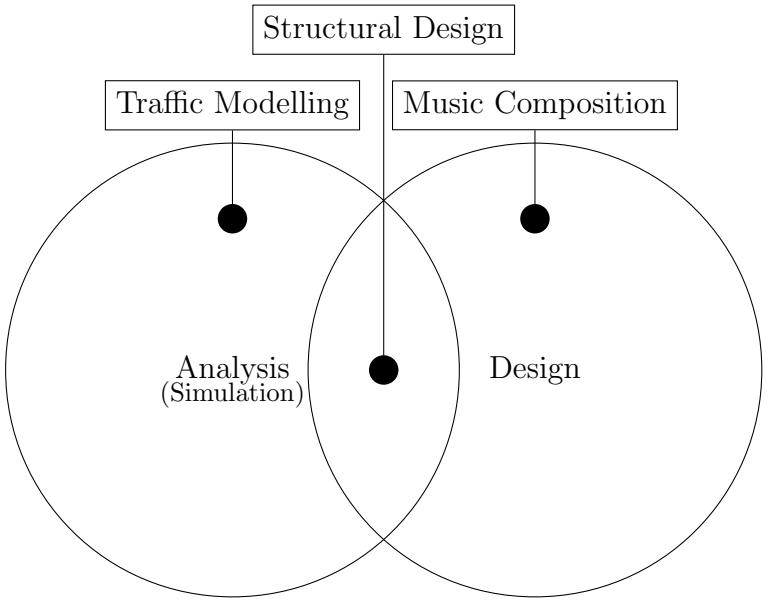

Figure 2: Classifying Applications of Cellular Automata

2 Traffic Modelling

Traffic congestion on major UK routes is an enduring problem and it is getting worse year on year due to the unrestricted trends in traffic growth. The volume of the traffic is too close to the maximum capacity of the roads.

One approach to reducing congestion would be to construct new or widen existing roads to provide additional lanes and hence increase the capacity of the road infrastructure. However, this approach can be very costly and delays may worsen while road works are in progress. Sometimes it is difficult to improve the existing road system due to environmental or social objectives. Another approach may be to control the traffic in such a way that congestion would be solved with the use of traffic lights or making adjustments to road marking. However, it is not a simple task to decide which approach would be most effective for a particular road network in order to limit traffic congestion. The use of a traffic model is needed to predict the behaviour of vehicles and the interactions between them on the roads.

2.1 Approaches

Approaches to traffic modelling can be divided into three groups: traffic flow theory, car-following theory, and the particle hopping traffic model [Nagel,

1996].

Traffic flow theory Fluid-dynamics and the kinetic theory of vehicular traffic belong to the group of traffic flow theories [Sone, 2002]. These theories describe traffic as a continuous traffic flow with varying density and an average velocity. They are a type of macroscopic theory as traffic is treated on a large scale and the movements of vehicles are not modelled individually.

Car-following theory The optimal velocity model and intelligent driver model were based on the car-following theory. These models are defined by ordinary differential equations and are able to simulate each vehicle individually [Nagel, 1996]. Therefore, these models are a type of microscopic model that describes the one by one following process of vehicles on the same lane. Traffic is represented using the spacing and speed differential between each pair of adjacent cars.

Particle-hopping model Particle hopping models are known to be numerically robust with complex geometries and the interconnections of realistic road networks citepnagel1995phv. The Nagel-Schreckenberg model (Henceforth, the NaSch model) [Nagel and Schreckenberg, 1992] and Rule 184 belong to the group of particle-hopping models.

The one-dimensional rule 184 is fundamental to traffic simulation by cellular automata. The NaSch model can be seen as an elaboration of this rule with additional features such as a discrete velocity for each vehicle, acceleration, deceleration and a random tendency to slow down the vehicle [Gershenson and Rosenblueth, 2009]. Depending on the global density and probability, the slowdown function in the NaSch model attempts to model the human tendency to overreact when braking.

These models use integer state variables within each cell to describe the dynamic properties of the system. Cellular automata based traffic models generate velocity distribution as a function of position on the road network. These models are also a type of the microscopic model like the car-following theories.

Traffic modelling is a vast complex subject and the choice of the model to be used can be difficult. The focus is on cellular automata as it has the ability to produce a wide range of traffic phenomena using simple rules. Each cell of the automaton can reflect individual object characteristics and small changes in the rules or the cells state can produce dramatic consequences [Benjamin et al., 1996]. Although automata based traffic models lack the accuracy compared to the car-following model, owing to the simplicity of the

model and the use of parallel update they enable quick run times and can be used to simulate large road networks.

2.2 One-Dimensional Cellular Automata

In the one dimensional cellular automata traffic, the car motions are simple and elegant. The road is divided into an array of cells. Each cell may be either occupied with at most one vehicle or it may be empty. Every vehicle has a velocity between 0 and the maximum of velocity, typically set as the speed limit of a particular road. The positions of the cars are updated in every time step. Vehicles with non-zero velocities move as far towards their destination as they are able without being obstructed by another vehicle.

The NaSch[1] model is based on the cellular automaton model for free-way traffic proposed by Nagel and Schreckenberg [1992] and which has been studied extensively in several papers since [Sasvari and Kertesz, 1997] [Schadschneider, 1999] [Esser and Schreckenberg, 1997]. Since the introduction of the NaSch model, cellular automata have become a well-established method of traffic flow modelling [Daoudia and Moussa, 2003].

The NaSch model is based on the one dimensional cellular automata for a realistic description of single lane traffic on highways. The model uses a one dimensional array with an open or periodic boundary conditions. Therefore, the total number of vehicles on the road is constant. The single lane road is divided into cells representing a length of 7.5 metres. This corresponds to the space for the vehicles length and some space in front and behind this vehicle. Each cell in the array can either be empty or contain exactly one vehicle with an integer velocity of between 0 and V_{max}. Vehicles move in one direction from one array cell to another by at most the velocity. Hence, the cell neighbourhood size is equal to the maximum velocity. Each time step that measures 1.2 seconds such that velocity of 5 cells per step in the automata corresponds to 70 miles per hour in real traffic.

The rule set has four sub-steps to calculate the new velocity and position, which are performed in parallel for all vehicles for each time step. Therefore from these values, another set of velocity and position for all vehicles have calculated during the next time step and stored in a temporary array during update. The use of a temporary array is to ensure values from the previous time step do not change until new velocity and position have been allocated to all vehicles.

The rule set of the NaSch model is presented as follows and omitting one of the four steps will no longer lead to realistic traffic behaviour.

[1]Nagel-Schreckenberg

The first step is acceleration. Vehicles that have not reached the maximum velocity will accelerate by one unit. The rule is $v_k = \min\{v_k + 1, V_{max}\}$, where v_k is the current velocity and V_{max} is the speed limit. This step represents the driving behaviour where the drivers tend to drive as fast as possible.

The second step is adjustment for safety distance. If there are only d empty cells in front of this vehicle, but its velocity v_k after the step 1 is greater than d, then the velocity will be reduced to d. The rule is $v_k = \min\{d, v_k\}$, where v_k is the current velocity and d is the safety distance. This step performs the interaction between vehicles.

The third step is randomisation. The velocity v_k will reduce by one unit with a chosen probability p between 0.01 and 0.50. This step represents many complex things that happen in real traffic such that a vehicle does not travel at a constant speed.

The forth step is motion. A new velocity v_k for this time step has obtained from step 1 to 3. The rule is $x_k = x_k + v_k$, where x_k is the current position of the vehicle and v_k is the new velocity. This step performs the movement of vehicles for this time step.

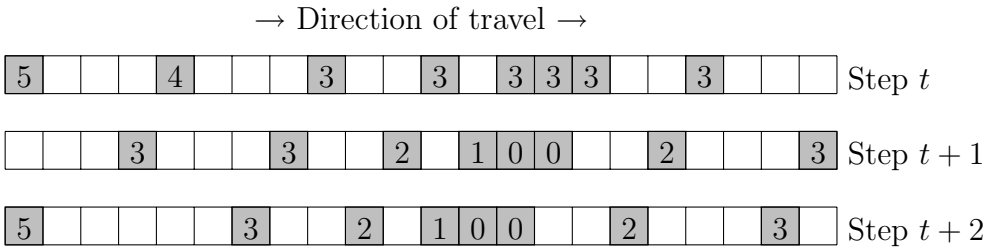

Figure 3: Example of the Nagel-Schreckenberg model

Since the NaSch model is a probabilistic traffic model, if randomisation probability p is zero and in the case of the maximum speed limit is one, then it is equivalent to the CA rule 184 in the Wolfram notation [Wolfram, 1983]. Three time steps of an example NaSch model are shown in figure 3. A shaded box represents the presence of vehicle in that cell and the number on the cell is the speed of the vehicle.

2.3 Two-dimension cellular automaton traffic model

Most of the major roads have at least two lanes with passing allowed, however the one dimensional NaSch model can only simulate a single lane road; therefore several researchers have constructed multi-lane models for major

road traffic. Moreover the NaSch model does not allow passing and the faster vehicle is always forced to slow down and follow the slower vehicle. Therefore the average velocity is therefore reduced to the free flow velocity of the slowest vehicle.

Rickert et al. [1996] extended NaSch model into a two lane cellular automaton traffic model. The two lane traffic model is constructed from two parallel single lane models with periodic boundary conditions and four additional rules are introduced in order to model the interaction of vehicles between the two lanes. The four additional rules are presented as follows with explanation:

The first additional rule is $gap(i) < V_k + 1$, where $gap(i)$ indicates the number of empty cells in front of the target vehicle. This rule represents the driver looking ahead to see if any vehicles are currently in front of his vehicle within the maximum distance he can move in the next time step.

The second additional rule is $gap(i)_o < V_k + 1$, where $gap(i)_o$ represents the forward gap on the other lane. This rule represents the driver looking at the other lane if a better speed can be obtained by switching lanes.

The third additional rule is $gap(i)_{o,back} < V_{max}$, where $gap(i)_{o,back}$, back represent the backward gap on the other lane. This rule represents the driver checking for a safety gap behind the target space in case the change of the lane will force another vehicle to slow down. This rule is particular important if one wants to maintain the realistic behaviour of the dynamics consisting laminar to the start-stop traffic flow.

The forth additional rule is $rand < 1$, where $rand$ is a random value, the use of the random value in the lane changing decision is to prevent the ping pong effect from occuring in the simulation of the model. When only using the first three additional rules, high traffic density on one lane will cause the ping pong effect as follows: All the vehicles see another vehicle in front of them, however there is no vehicle on the fast lane. This will led to all vehicles move to the fast lane in the hope to obtain a better speed, however in the next time step they will return to the slow lane since it is empty.

If the target vehicle can satisfy the conditions from all of the four additional rules, then the target vehicle would able to move to the other lane to get better performance.

These four additional rules are performed before the original rules from the NaSch model, therefore the update step is divided into two sub-steps. In the first sub-step, the exchange of vehicles between lanes (sideways movement) is performed using the extended rule set. The update process in the first sub-step is parallel with each vehicle making its decision based on the configuration at the beginning of the time step. In the second sub-step, the rules from the NaSch model are used on each lane. The configuration re-

sulting from the first sub-step is used to perform the update process in the second sub-step.

Nagel et al. [1998] extended the two lane traffic model to model some attributes of German and American traffic law. In Germany, the lane usage is governed by two laws where the right (slow) lane has to be used by default and passing has to be on the left (fast) lane. The United States is similar to Germany except that passing on the right is not explicitly forbidden. The rules for the legal constraints are as follows:

For the German road network, the legal criterion is $v_r \leq v \vee v_l \leq v$ and therefore the change back rule is $v_r > v \wedge v_l > v$. Since passing is not allowed on the right, if there is a slow vehicle on the left lane then the vehicle on the right lane has to change to the left lane behind the slow vehicle. Therefore a vehicle changes lane if there is a slow car in front or on the left, then changes back to the slow lane when its desired velocity can also be maintained on the slow lane.

For the United State road network, legal criterion is $v_r \leq v \wedge v_r \leq v_l$ and therefore the change back rule is $v_r > v \vee v_r > v_l$. Since passing on the right is not explicitly forbidden, the left lane is in use unless the vehicle can obtain a higher speed than in its own lane. Therefore a vehicle changes from right lane to left lane only if there is a slower vehicle in front of it and only when a higher speed can be obtained, then changes back to the slow lane when traffic on the right lane is running faster than the left lane.

Simon and Gutowitz [1998] investigated extra rules for passing in bidirectional traffic on a single lane. These rules ensure a vehicle must not decelerate while passing another and if an oncoming vehicle is seen then rapidly decelerate the vehicle in order to break the symmetry between lanes.

A richer version of the two dimension cellular automaton traffic model was created by [Esser and Schreckenberg, 1997]. They have also made a complete simulation tool for urban road networks. The model can simulate realistic traffic light intersections, arbitrary kinds of roads and crossings, priority rules, and parking capacities are considered as well as the circulation of public transport. Figures 4 and 5 illustrate their work. The vehicles can be moved according to route plans or at random depending on the available data.

An approach on modelling the traffic on a traffic circle (similar to a roundabout) was undertaken by Dupuis and Chopard [2003] based on the NaSch model. NaSch is already periodic and so naturally simulates a circular flow of traffic. Vehicle in rotaries always have priority over the other vehicles and each vehicle entering the rotaries is randomly characterised by an exit with some probability.

Most of the cellular automata traffic models use a cell to represent a sec-

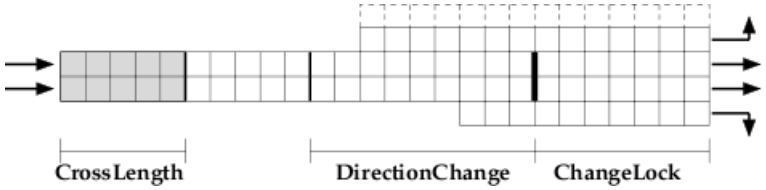

Figure 4: Structure of a two-lane edge with additional turning sections from Esser and Schreckenberg [1997, Fig. 1]

tion of road of 7.5 metres in length. There are a few attempts on modelling the traffic using cellular automaton on vary different cell size [Lawrence et al., 2003] [Mallikarjuna and Ramachandra Rao, 2007] [Xiao-Mei Zhao and Jiang, 2009]. All the experiments with a similar idea of modifying the cell size in order to model heterogeneous traffic instead of just cars, including motorbikes, buses, and lorries.

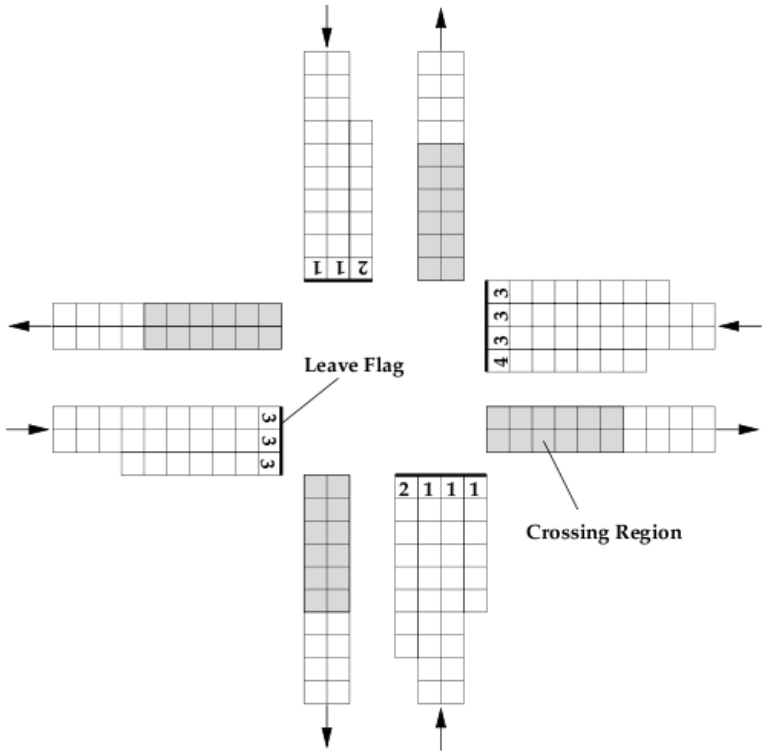

Figure 5: Schematic of road junctions from Esser and Schreckenberg [1997, Fig. 3]

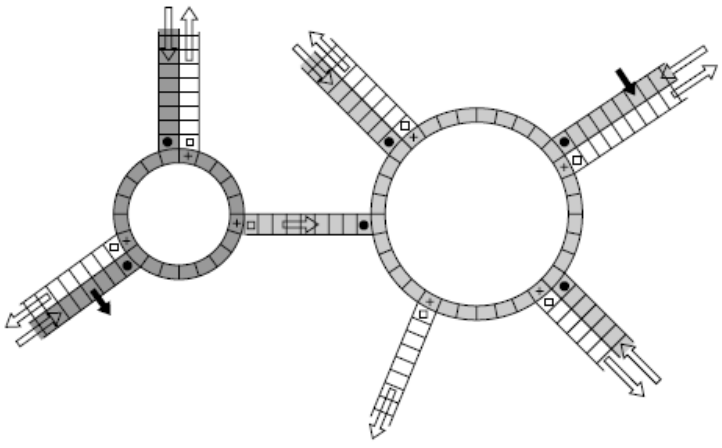

Figure 6: Schematic of road junctions from Dupuis and Chopard [2003, Fig. 1]

3 Structural Design

Given a situation which requires a structural element such as a load bearing beam or truss, it is necessary to have an appropriate design which can withstand the stresses in the environment where the element is put to use. Typically this involves striking a balance between the strength of the element's various components and the resulting weight of the design [Kita and Toyoda, 2000]. Unnecessarily heavy elements both have a greater material cost and place a greater load on anything beneath them.

Several applications of cellular automata to the design of such elements are detailed in the literature (Kita and Toyoda [2000], Gürdal and Tatting [2000], and Missoum et al. [2005]). Significantly, these differ from a purely simulation-based approach such as has been applied to, for example, the design of road systems. In those systems cellular automata are used to evaluate the performance of traffic flow, but the process of refining the design is not addressed. The approach taken here uses automata to perform the entire design process end to end.

Nevertheless, design automata still possess a simulation aspect in that their lattices are an analogue to the physical object or system being designed. The structural element is decomposed into regular parts, each one represented by a cell. The cellular neighbourhood corresponds to the physical locality of that part, the establishing a straightforward mapping between the lattice and the element. Furthermore, the state of each cell encapsulates variables which collectively specify the design of a the element. These design variables are subject to refinement by local update and it is in this manner that automata converges on a good design.

3.1 Representation

The most significant choice in implementing a cellular automata for design is how to represent a useful portion of the total design space using the state of a cell lattice. The design space cannot be unconstrained for reasons of tractability and so there must be some preconception of what a reasonable design would look like and how that type of design might be parameterised by variables. This can be drawn from the domain knowledge and conceptual model used by human designers. Representations may differ in how much preconception is built into the model and how much is left to the automata.

Kita and Toyoda [2000] examines the design of plates or beams using a two dimensional automata. This approach takes the lengthways cross section or sideways projection of a structural element and lays the lattice over it such that each cell corresponds to a regular square area of material. The thickness

of the beam within that area is expressed as a continuous design variable of the cell. Note however that this simplification precludes applying any forces out of the plane of the lattice without special casing shearing forces.

Using thickness as the design variable is akin to the simplification of using a height-map in the representation of terrain volumes. Likewise, the generalisation of this approach would be to use a three dimensional discrete automata in which cells correspond to cube volumes and their state indicates presence or absence. However, this would increase the number of cells and the size of the neighbourhood and, perhaps most importantly when scaling to larger lattices, increase the dimensionality required of the computational medium. Hence, this would increase both the per iteration cost of using the automata and likely also the number of iterations required to reach convergence.

An alternative to breaking the element into regular areas is to impose some higher level structure on the design. Gürdal and Tatting [2000] (and Tatting and Gürdal [2000]) and Missoum et al. [2005] take this approach by restricting themselves to the design of planar trusses. A truss is a type of structural element formed by a framework of member beams which are typically arranged into triangular faces for rigidity. The thicknesses of these members can also be expressed with continuous design variables on the lattice, although with some additional complexity compared to regular cells.

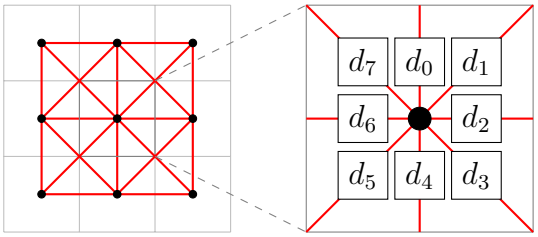

Figure 7: Representation of a Truss

Gürdal and Tatting [2000] assumes a particular design in which member beams run horizontally, vertically, and in both diagonal directions, meeting at regular grid-spaced junction points. Foreshadowing, each junction point connects to the eight surrounding points matching the layout of a Moore neighbourhood as shown in figure 7. The heterogeneous nature of the member's orientations, not to mention the overlapping diagonals, make it impossible to lay a homogeneous lattice over the truss and maintain a mapping one member and hence one design variable to each cell. Hence, rather the lattice is laid over the junction points such that each cell covers half the length of each of the eight members meeting at that point. This approaches uses re-

quires eight variables per cell and the thicknesses at each end of a member may differ.

The granularity of the lattice and hence the density of potential junctions is of particular interest when designing trusses. When the lattices are small, as in the bulk of Gürdal and Tatting [2000] which uses trusses only one member high, the majority of the members must have a non-zero thickness (i.e. be present) in any viable truss design. This means that the design of any truss is strongly constrained by the underlying configuration of the lattice. By contrast, Missoum et al. [2005] uses much finer lattices such that the majority of potential members can be eliminated from the design. Compound members stretching the breadth of the truss can be built in arbitrary positions from the component members connecting the points on the lattice.

3.2 Environment

The design process is performed in the context of a particular target environment. The environment exerts forces on the element which cause structural stress and it is the aim of the design process to produce a structural element attuned to operate in those conditions.

Firstly, the environment requires that a structural element have a particular size and shape so that it fits correctly into the larger construction for which it's intended. This is fixed into the design ahead of time by the configuration of the lattice.

There are essentially two different ways in which the environment can interact with a structural element. The first way is for the element to be placed under load by something on top of it or hanging off it. When such a force is applied this will cause the element's shape to deform slightly and such that parts of it will be displaced from their original positions. The second way is for the environment to prevent or resist deformation where the element rests on or is affixed to something else. Figure 8 depicts a lattice subjected to an environment.

Interaction with the environment naturally occurs at the physical boundary of a structural element. In many cases this is the same thing as the boundary of the lattice, but not always. As real elements are three dimensional objects of which the representations used here are two dimensional simplifications, it's possible to have an environment which would affect cells in the middle of the lattice.

Whether or not a model is capable of handling this depends on to what degree is relies on the boundary condition. When a performing local update on a boundary cell its neighbourhood will extend beyond the lattice and the rule will require the values of state variables from cells which don't exist.

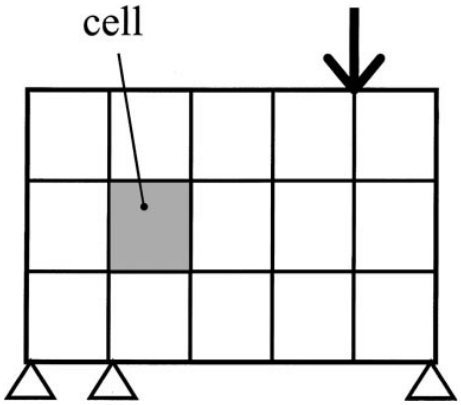

Figure 8: Lattice subjected to an environment from Kita and Toyoda [2000, Fig. 1]

Gürdal and Tatting [2000] uses the boundary condition to represent environment effects by synthesising values for these non-existent cells based on the specification of the environment. Just as physical effects such as mechanical stress are propagated between cells within the lattice, the variables of these virtual can be set up to propagate them in from outside. Where an element does not have contact with another part of the structure the thickness of these virtual cells are set to zero so that they offer no resistance to deformation. Hence, this model does not support environmental effects in the middle of the lattice.

By contrast, both Kita and Toyoda [2000] and Missoum et al. [2005] use a uniform boundary condition. None of the virtual cells surrounding the boundary apply any force nor offer any resistance. Instead, the environment manipulates arbitrary cells on the lattice. Applying a force to a particular cell to represent a load or fixing its displacement to zero to represent a non deformable part of the element.

3.3 Analysis

In order to use local update to revise the design it is necessary to know what the local conditions are, such as the stress or displacement, at each part of the structural element. There is where the simulation aspect of the automata comes into play. The effects of the environment at the element's boundary need to be propagated across the lattice to calculation the conditions in each cell. Again, there are several different approaches to achieving this.

Kita and Toyoda [2000] uses a technique common in structural mechanics called finite element analysis. Each cell has a number of variables specifying it's physical condition. These comprise the design variables, for which every cell has a definite value, and a number of simulation variables. Some of the simulation variables will be specified at various points by the environment, but the majority are free. The lattice is transformed into a system of linear equations or sparse matrix and then solved. This yields a full set of values for the free variables of each cell.

Finite element analysis is global process which operates over the entire lattice at once. It is not a form of local update and so no boundary condition is involved. While parallel matrix solvers exist, the problem is more complex than the embarrassingly parallel nature of local update rules. Hence, this both somewhat violates the spirit of cellular automata and compromises the scalability advantages offered by the low coupling of local update.

Both Gürdal and Tatting [2000] and Missoum et al. [2005] remedy this by developing a local update rule to take the place of finite element analysis. These both use the displacement of the cell as a simulation variable. Applied forces displace the cells at particular points. The local update rule then propagates this by way of displaced cells applying pressure to their neighbours and hence causing them to be displaced also.

3.4 Design

Finally, having selected a representation for both the design and the environment and method of analysing potential designs, it only remains to describe how the design update is performed. Figure 9 illustrates a problem specification and design resulting from Kita and Toyoda's approach.

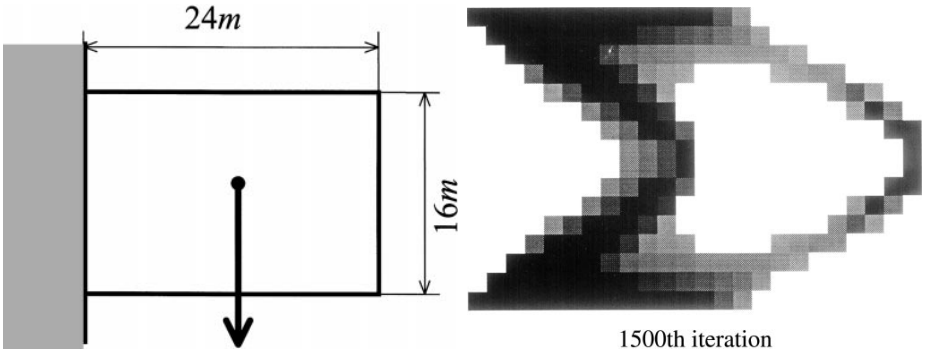

Figure 9: Example 3 from Kita and Toyoda [2000, Fig. 7 and 12]

Kita and Toyoda [2000] interleaves performing finite element analysis with applying the design update rule. The analysis does not use a local update and calculates the state of the entire lattice in one go starting from the environmental conditions. Hence, no simulation state is persisted from one generation to the next. The design process is a simple optimisation problem with the analysis pass providing the input to the objective function.

The stress sensitivity is calculated by dividing the magnitude of each cells stress vector by its thickness. Minimising the stress vector will therefore will thicken the parts of the structural element which are most under stress. This is combined together with a weight penalty and a "CA-constraint condition" which smooths the lattice [Kita and Toyoda, 2000, eq. 17] in order to produce the objective function. The derivative of the objective function is then used to descend the gradient on each design update. The multi-criteria aspect of the weight penalty prevents the maximally strong design from always being the optimal solution.

Gürdal and Tatting [2000] and Missoum et al. [2005] both use local update for the analysis. This means that simulation state persists across time steps, including design updates, and that the convergence of the simulation state will increase over multiple steps with the same design. Both found that ratio between simulation and design updates is important as the direction in which to refine the design can be better chosen with more accurate simulation data. Gürdal and Tatting [2000] reports that a ratio of six to one simulation to design updates halved to number of generations required to reach the optimal design compared to a one to one ratio.

Gürdal and Tatting [2000] evaluates whether external forces should be applied constantly during simulation or applied gradually over time and finds the best choice depends on the formulation of the analysis rule.

4 Musical Composition

Composition is the name given to the "design" of musical pieces. A range of different cellular automata based techniques have been applied to this process in the literature [Burraston et al., 2004] [Burraston and Edmonds, 2005]. However, computerisation is problematic because the criteria for a good piece of music are nebulous. Enjoyment is at least partially subjective and driven by complex processes locked up in the brain.

Conventional optimisation techniques start from a random part of the search require a heuristic in order to work towards a good design. Cellular automata offer a different approach based on the intrinsic artistic appeal patterns resulting from the emergent properties of simple cellular systems.

These patterns can be translated into musical notes or other audio while a human tunes the parameters of the system to find a pleasing result.

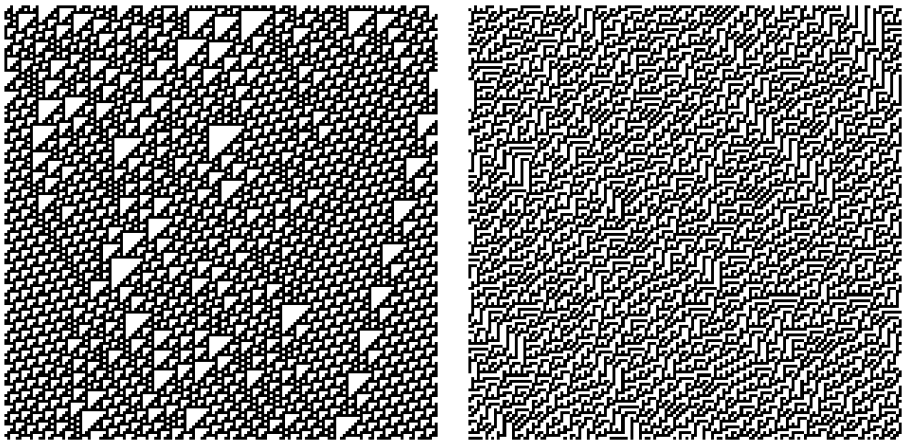

Figure 10: Complex (left) and chaotic (right) patterns produced by projecting one-dimensional automata over time [Burraston et al., 2004, Fig. 3 and 4]

Interesting results can be obtained from simple binary automata like Conway's Game of Life or even simpler one dimensional automata. Figure 10 shows complex patterns generated by a one dimensional automata over time. Each row corresponds to a generation. These patterns can be translated into music by mapping the state of the lattice in each generation onto a note.

Langton [1986] conceiving the lambda parameter to indicate for what proportion of the neighbourhood states a given local update rule will emit a live cell in the next generation. The can be used as an aid to rule creation by randomly generating rules which meet a specified lambda parameter.

5 Evaluation and Conclusion

Cellular automata excel at modelling physical systems because the properties of the physical world and cellular lattices are very similar. Namely that, when viewed at any scale, objects are independent apart from localised interactions. As a vast number of events in the physical world happen concurrently, so cellular automata are amenable to massive parallelism. It's a natural fit.

In some cases, as with structural design, it may be possible to devise update rules which vary the non-simulation parameters of a physical model. This is perhaps an under-explored approach to using automata which could

be applied to other areas. Envisage a traffic simulation in which congestion forced roads to widen or the lack of sufficiently direct route to their destination caused the cars to bore new roads through the country-side. Even though an automata cannot replace a traffic engineer given the nuances of infrastructure planning, its proposed designs may help humans to better understand the problem than simply looking at simulations.

The usefulness of automata outside of the physical domain is less clear though. However, this is perhaps less of a drawback than it first appears. We live in the physical world and many of the computational problems we find interesting are physical in nature. Many computational techniques do not map easily onto an automata, but were created to solve non-local formulations of physical problems. Given a readiness to discard conventional techniques, as local update rules were developed to replace finite element analysis in the structural design application, automata still have broad applicability.

One clearly non-physical problem was music composition. The pattern generating aspects of automata could be applied to other forms of art than music. Indeed, the are lattices arguably already a form of visual art. However, abstract emergent behaviour doesn't appear to be a particularly useful way of designing much else. Having listened to some of the results, these authors question how good it is at composing music. If we return to the physical however and consider a concrete version of cellular computation, cellular automata would make an excellent model for simulating microelectronic circuitry.

References

S.C. Benjamin, N.F. Johnson, and PM Hui. Cellular automata models of traffic flow along a highway containing a junction. *Journal of Physics A: Mathematical and General*, 29:3119–3127, 1996.

D. Burraston and E. Edmonds. Cellular automata in generative electronic music and sonic art: a historical and technical review. *Digital Creativity*, 16(3):165–185, 2005.

D. Burraston, E. Edmonds, D. Livingstone, and E.R. Miranda. Cellular automata in midi based computer music. In *Proceedings of the International Computer Music Conference*, pages 71–78, 2004.

A.K. Daoudia and N. Moussa. Numerical simulations of a three-lane traffic model using cellular automata. *Chinese Journal of Physics*, 41(6), 2003.

A. Dupuis and B. Chopard. Cellular automata simulations of traffic: a model for the city of geneva. *Networks and Spatial Economics*, 3(1):9–21, 2003.

J. Esser and M. Schreckenberg. Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C: Physics and Computer*, 8(5):1025–1036, 1997.

C. Gershenson and D.A. Rosenblueth. Modeling self-organizing traffic lights with elementary cellular automata. *arXiv:0907.1925*, 2009.

Z. Gürdal and B. Tatting. Cellular automata for design of truss structures with linear and nonlinear response. In *41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference & Exhibit*, Atlanta, US-GA, April 2000.

E. Kita and T. Toyoda. Structural design using cellular automata. *Structural and Multidisciplinary Optimization*, 19(1):64–73, 2000.

C.G. Langton. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1-3):120–149, 1986.

W. Lawrence, C.W. Chang, and P.D. Student. Motorbike's moving behavior in mixed traffic: Particle-hopping model with cellular automata. *Journal of the Eastern Asia Society for Transportation Studies*, 5, 2003.

Ch. Mallikarjuna and K. Ramachandra Rao. Identification of a suitable cellular automata model for mixed traffic. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 6, pages 2454–2469, 2007.

S. Missoum, Z. Gürdal, and S. Setoodeh. Study of a new local update scheme for cellular automata in structural design. *Structural and Multidisciplinary Optimization*, 29(2):103–112, 2005.

K. Nagel. Particle hopping models and traffic flow theory. *Physical Review E*, 53(5):4655–4672, 1996.

K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221–2229, 1992.

K. Nagel, D.E. Wolf, P. Wagner, and P. Simon. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1425–1437, 1998.

M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A: Statistical Mechanics and its Applications*, 231(4):534–550, 1996.

M. Sasvari and J. Kertesz. Cellular automata models of single-lane traffic. *Physical Review E*, 56(4):4104–4110, 1997.

A. Schadschneider. The nagel-schreckenberg model revisited. *The European Physical Journal B: Condensed Matter and Complex Systems*, 10(3):573–582, 1999.

PM Simon and HA Gutowitz. Cellular automaton model for bidirectional traffic. *Physical Review E*, 57(2):2441–2444, 1998.

Y. Sone. *Kinetic theory and fluid dynamics*. Birkhauser, 2002.

B. Tatting and Z. Gürdal. Cellular automata for design of two-dimensional continuum structures. In *Proceedings of 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, US-CA, September 2000.

S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, 1983.

Zi-You Gao Xiao-Mei Zhao, Bin Jia and Rui Jiang. Traffic interactions between motorized vehicles and nonmotorized vehicles near a bus stop. *Journal of Transportation Engineering*, 135(11):894–906, 2009.

Konrad Zuse. *Rechnender Raum*. Friedrich Vieweg & Sohn, 1969. English translation: *Calculating Space*, MIT Technical Translation AZT-70-164-GEMIT, Massachusetts Institute of Technology (Proj. MAC), Cambridge, MA 02139, Feb. 1970.