Chapter 1

Introduction

1.1 Motivation

The ultimate purpose of most scientific investigations is to determine how physical or other systems will behave in particular circumstances [1]. However, the adaptive capabilities of the brain and many other biological systems go far beyond those of any artificial systems thus far constructed by conventional engineering means [2]. Hence many phenomena can still not be studied using existing modeling capabilities.

In our never-ending quest to predict the reactions of physical systems, more and more methods are developed to achieve this goal. Much has been discovered about the nature of the components in physical and biological systems, but little is known about the mechanisms by which these components act together to give the overall complexity observed [3]. Nature provides many examples of systems whose basic components are simple, but whose overall behavior is extremely complex.

Mathematical models, such as cellular automata (CA) capture many essential features of such systems, and provide some understanding of the basic mechanisms by which complexity is produced [2]. It may be speculated that they are widespread in natural systems, perhaps occurring wherever nonlinearity is present [3]. To illustrate the above Stephan Wolfram makes use of the laws that govern the freezing of water and the conduction of heat. These laws have long been known, but analyzing their consequences for the intricate patterns of snowflake growth has not yet been possible.

While many complex systems may be broken down into identical components, each obeying simple laws, the huge number of components that make up the whole system act together to generate very complex behavior [4]. There is, however, extensive evidence that at a functional level, the basic components of such complex natural systems are quite simple, and could, for example, be emulated with a variety of technologies [2]. The complexity is generated by the cooperative effect of many simple identical components.

To discover and analyze the mathematical basis for the generation of complexity, simple mathematical systems that capture the essence of the process have to be identified [4]. However, as indicated above, it is not yet known how a large number of these components

act together to perform complex tasks. There seem to be general principles which govern such complex behavior, and allow this behavior to be molded to achieve particular goals. If these principles could be found and applied, they would initiate new forms of engineering solutions.

As an analysis tool, CA are sufficiently simple to allow detailed mathematical analysis, yet sufficiently complex to exhibit a wide variety of complicated phenomena [5]. While many natural systems do tend toward disorder, a large class of systems, biological ones being prime examples, show a reverse trend. In the latter case a disordered or structure-less initial state may spontaneously change to a structured state, in due course.

The development of digital computers during the last few decades has caused numerical calculations to become a discipline of increasing importance [6]. Today computers no longer are relatively expensive tools available to a very limited group of scientists, but have evolved into a tool available to almost every scientist and engineer. Consequently the interest in numerical calculations is now greater than ever. Conventional engineering or computer programming systems are built to achieve goals by following strict patterns, which specify the detailed behavior of each of their component parts [2]. The overall behavior of these systems must therefore be simple enough to ensure complete predictability.

As an example: Motion in many conventional mechanical engineering devices is simply constrained to be periodic. When simulating motion in conventional computer programming, each step consists of a single operation performed on small amounts of data elements. More complex behavior could be obtained from the basic components, whether mechanical or logical, but the principles necessary to make use of such behavior are not always known yet [2].

In structural analysis, researchers have suggested the "lattice analogy" since 1906 to solve continuum problems. The continuum is approximated by a regular mesh of elastic bars, and the method is based on well-known methods for the analysis of framed structures. More recently in structural mechanics and stress analysis, a very powerful numerical method known as the finite element method (FEM) appeared as an alternative to the finite difference method (FDM), which was the first widespread numerical method able to solve differential equations by dividing the domain of interest into elements [6]. One of the main advantages of FEM over FDM is its ability to handle elements of different sizes. FEM was developed at the same time as powerful computers. This fact combined with the universal "use-ability" of FEM contributed to its enormous success.

As a result FEM was for many years practically the only numerical method used in structural mechanics and today FEM is still the dominating numerical method in that area. FEM's popularity is a consequence of the use of "piece-wise" interpolated polynomials as basic functions. Such functions are not infinitely smooth. Even their first derivatives are usually discontinuous. Piece-wise polynomials are effective for a number of reasons: they fulfill essential boundary conditions[1], they provide a sparse resultant matrix and they can handle the complicated geometry of structures. Therefore, most commercial analysis systems are based on FEM. FEM has its drawbacks however, and the sources of inaccuracies are widely

[1]The physical constraints to movement without which the body will experience rigid body motion and be in a stress-free state.

discussed in the literature.

As opposed to FEM, the basic components of CA are discrete, but at least in some cases the aggregate behavior of large numbers of these components can be effectively continuous [7]. As a result, it is possible to use CA as models of continuum systems. This understanding may now be used to design systems whose complex behavior can be controlled and directed to particular tasks.

From complex scientific systems one must develop complex engineering systems. The laws governing the deformation of a structure are based on absorption of energy applied to the structure by the material, and one can now attempt to model this phenomenon using CA. The application of CA to structural mechanics will open an entire new direction of thought and with it new possibilities. A prerequisite for the application of a mathematical idealization like CA to continuous systems is a basic understanding of the manner in which the system reacts. Where no simple formula for the behavior of many natural systems can be given, the consequences of their evolution can only be found by direct simulation or observation.

Over the last decade, the role of computational simulations in all aspects of study in design in science and engineering has steadily increased. Algorithms and systems software are only two of the four main components required for the simulation and modeling technology for computational science and engineering (CSE) [8]. The advent of parallel computers has offered scalable high-performance engines to computational scientists and engineers. These may find applications in solving problems in shorter time periods and facing problems that were impossible to master by the use of sequential architectures. In order to make applications compatible for parallel machines, practitioners in this field are required to develop algorithms that are different from the ones previously used on sequential computers and to express their algorithms using new parallel languages and tools or parallel extensions to existing paradigms. In recent years there have been several attempts to provide more practical programing tools, environments for parallel programming and execution of CSE applications on parallel computer [9].

CA define a machine-independent parallel model of computation that can be used by various fields of computation. Despite the large amount of fields explored since the von Neumann definition of CA [10], researchers working in the field of CA have devoted their attention to the CA theory and abstract modeling of CA. By investigating the interaction of CA with parallel methods of computing it is possible to implement more complicated theories and models. This introduces a completely new fields of research exploration for the application of CA.

1.2 Objectives

The problem at hand is to approximate the mechanics of materials by CA.

To be able to effectively apply an idealization like CA to the field of structural mechanics, it is necessary to clearly understand the principles involved. It is thus required to fully understand the definition of CA and the inherent properties of CA.

The behavior of metallic materials on a microscopic level has been studied in great detail.

The behavior of such materials is believed to be understood to a reasonable extent and their properties are clearly described by various mathematical formulations. The definition of CA makes it very difficult to implement a purely mathematical description into a CA model. Integration of two fields with totally different approaches and description requires careful attention to detail, but also allows for new fields of exploration.

The inherent properties of CA such as the implicit parallelism make CA ideal for the use in current computing environments. CA are also capable of demonstrating systems with non-linearity and irreversibility. Hence it is desirable that these aspects are addressed in this study.

To summarize: This thesis aims to explore some of the possible methods of implementation of CA for obtaining approximations in structural analysis. Carefull attention is to be given to the problems that are involved when using CA in structural mechanics. An exploration into the inherent parallelism in CA is desirable to obtain approximations with low, realistic computational effort.

1.3 Approach

Firstly, a perspective of all aspects involved is obtained by carefully investigating the details of CA construction and operation. Subsequently an overview is given of the mathematics describing structural mechanics and current methods that approximate the structural solution with discrete representations. These methods allow for better understanding of the governing aspects of the problem and how an approximate method like CA could be successfully applied in structural analysis.

Secondly, machine learning using a genetic algorithm is applied to the determination of optimum rules used in the CA, using finite element, boundary element and analytical approximations as the basis for machine learning. Cell rules are obtained for various problems and investigated for global validity and transferability.

Finally, the problems and benefits associated with parallel implementation of CA are investigated. To this extent, a parallel computer is constructed using inexpensive, available resources, and by combining them in a single computing cluster, called a Linux Beowulf 'Souper' computer. Using this infrastructure, the effects of symmetric versus asymmetric rules in the CA are studied.

Chapter 2

Introduction to cellular automata

2.1 Informal definition

Cellular automata (CA) are mathematical idealizations of physical systems in which physical quantities like space and time take on a finite set of discrete values [5].

Each point in a regular spatial lattice, called a cell, can have any one of a finite number of states. The states of the cells in the lattice are updated according to a local rule. That is, the state of a cell at any given time depends only on its own state in the preeceding time step, and the states of its nearby neighbors at the previous time step. All cells on the lattice are updated synchronously. Thus the state of the entire lattice advances in discrete time steps [11]. CA provide a framework for a large class of discrete models with homogeneous interactions [12]. These are characterized by the following fundamental properties:

- They consist of a regular uniform discrete lattice (or "array") of cells.
- The evolution takes place in discrete time steps.
- Each cell is characterized by a state taken from a finite set of states.
- Each cell evolves according to the same rule, which depends only on the state of the cell and a finite number of neighboring cells.
- The neighborhood relation is local and uniform.
- The variables at each site are updated simultaneously (synchronously), based on the values of the variables in their neighborhood at the preceding time step, and according to a definite set of "local rules".

CA were originally introduced by von Neumann and Ulam (termed "cellular spaces") as a possible idealization of biological systems with the particular purpose of modeling biological self-reproduction [5].

They have been applied and reintroduced for a wide variety of purposes, and referred to by a variety of names, which include:

- Tessellation automata
- Homogeneous structures
- Cellular structures
- Tessellation structures
- Iterative arrays

One of the most well-known CA rules, namely the "game of life" was conceived by Conway in the late 1960s [13, 14] and was shown to be computation-universal[1] [16].

CA have been used to study problems in number theory. They have also been applied to tapestry design [17, 18, 19, 20, 21].

2.1.1 Simple example

As an introduction to understanding CA, a simple one-dimensional CA with only two possible cell states (e.g., see [22]) is given.

Rule Table:

Neighborhood:	111	110	101	100	011	010	001	000
Output bit:	1	1	1	0	1	0	0	0

Grid:

$t = 0$	0	1	1	0	1	0	1	1	0	1	1	0	0	1	1
$t = 1$	1	1	1	1	0	1	1	1	1	1	1	0	0	1	1

Table 2.1: A simple one-dimensional, two state, cellular automata

Each cell has a possibility of two states, namely 0 and 1. The connectivity radius is $r = 1$, meaning that each cell has two neighbors, one to its left and one to its right. The grid size is 15, and spatially periodic boundary conditions are applied, viz. connectivity between cell 1 and cell 15 is assumed. The pseudo-time t is incremented in discrete steps of 1. Considering the first neighborhood and output bit, the cell in consideration has neighborhood 101 at $t = 0$, and hence, a state 1 at time $t = 1$. (For time $t = 2$, the state remains unchanged.)

2.2 Formal definition

We can now continue to give a mathematical description of CA with conventions:

- d = dimension
- k = states per cell

[1] They can generate arbitrarily complex patterns of symbols and process them [15].

- r = radius

Assume a d-dimensional cellular automaton with $d = 1$ (for simplicity) takes as its underlying space the lattice S^Z (Z=integers, infinite in both positive and negative directions) where S is a finite set of k elements. The dynamics are determined by a global function.

$$F : S^Z \rightarrow S^Z \tag{2.1}$$

whose dynamics are determined "locally" as defined below. A "local (or neighborhood) function f" is defined on a finite region.

$$f : S^{2r+1} \rightarrow S \tag{2.2}$$

The all-important property of CA is that this function is determined by a finite look-up table [11]. Both the domain and range of f are finite. The global function F arises from f by defining:

$$F(c)_i = f(c_{i-r}, \ldots, c_{i+r}) \tag{2.3}$$

A concrete example with $k = 2, r = 1$ would take a doubly infinite string of zeroes and ones to a new string at which each site is replaced by the logical and of its two neighbors [11]. Some relevant facts from a topological standpoint are:

- The base space S^Z is compact and the global function F is continuous. This makes CA an ideal meeting point between continuous dynamics and complexity theory, since they are discreetly defined but exhibit continuous dynamics.

- The map F commutes with the shift operator which takes c_i to c_{i+1}.

Cellular automata are in fact characterized completely by these properties [11]. The transition to more dimensions is straightforward. The only difference is that the global function F is defined over S^Z (functions from a d-dimensional lattice to S) and the local function f is determined by enumerating the image of all patches of size $2^{(r+1)^d}$ [11].

2.3 Neighborhood definition and size in 2-D

Various definitions of neighborhoods are possible. Considering a two-dimensional lattice, the following definitions are common and are shown in Figure 2.1 [23]

- **von Neumann neighborhood** (Figure 2.1.a):
  The von Neumann neighborhood of a cell is defined as the cells directly above and below, and to the right and to the left of the cell in discussion. The radius $r = 1$, as only the next layer is considered.

- **Moore neighborhood** (Figure 2.1.b):
  The Moore neighborhood is an enlargement of the von Neumann neighborhood, and includes the diagonal cells. As for the von Neumann neighborhood, the radius $r = 1$.

- **Extended von Neumann neighborhood** (Figure 2.1.c):
  The extended von Neumann neighborhood is equivalent to the Moore neighborhood with the cells in the second layer above and below and to the left and to the right included. Thus the radius $r = 2$.

- **Extended Moore neighborhood** (Figure 2.1.d):
  The extended Moore neighborhood is equivalent to the description of the Moore neighborhood above, but the neighborhood reaches over the distance of the next adjacent cells. Hence the radius $r = 2$ (or larger).

- **Arbitrary neighborhood** (Figure 2.1.e):
  Any other neighborhood can be defined as long as it is uniform[2] and finite.

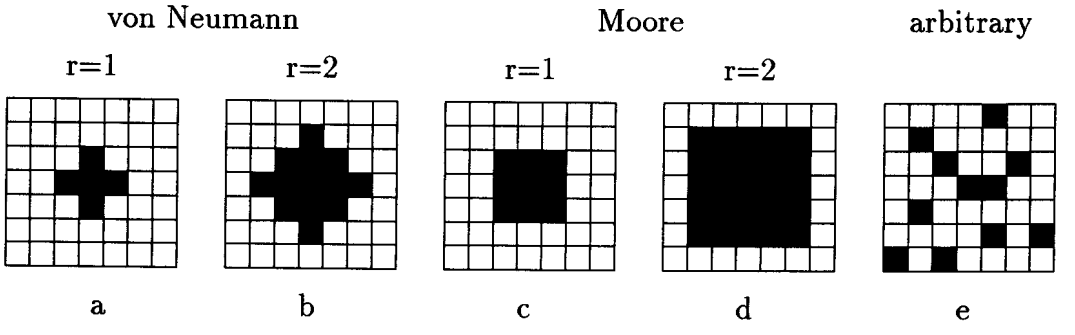

Figure 2.1: Various CA neighborhood definitions: (a) von Neuman neighborhood $r = 1$, (b) von Neuman neighborhood $r = 2$, (c) Moore neighborhood $r = 1$, (d) Moore neighborhood $r = 2$, (e) Arbitrary neighborhood

2.4 Boundary conditions

In the formal definition of CA, it is usually required that the lattice is infinite in all dimensions. For considerations of computability and complexity, this is reasonable and necessary. Since it is impossible to simulate a truly infinite lattice on a computer (unless the active region always remains finite), some boundary conditions have to be prescribed. Another reason for using certain boundary conditions is the natural boundary conditions of the problem under consideration.

Three kinds of boundaries will be considered. These are periodic, reflective and fixed value boundaries.

[2]The same neighborhood is used for each cell in consideration.

- Periodic boundaries are obtained by periodically extending the lattice. The one-dimensional version can be seen in Table 2.1. It is only necessary to store only the original array and a boundary of width r, where r is the radius of the neighborhood of the CA. These periodic boundary conditions come closest to simulating an infinite lattice, and are therefore often used [12].

  The two-dimensional version of periodic boundaries, where the top and bottom are connected, and the left and right edges are connected, leads to the topology of a torus (it is not possible to obtain the topology of a sphere with a regular lattice of arbitrary size).

- Reflective boundary conditions are obtained by reflecting the lattice at the boundary, which means that in one dimension the end cell sees a copy of itself as the boundary. This type of boundary is best suited for boundaries where the system to be simulated also has a boundary, and where the value of the physical variables is not fixed (e.g. von Neumann, zero-flux boundary conditions in a diffusive system).

- Fixed value boundary conditions are obtained by simply prescribing a fixed value for the cells on the boundary. This value must be determined from application.

All three boundary conditions can be combined, so that different boundaries can have different conditions. If one edge has periodic boundary conditions, the opposite edge must also have periodic conditions. Often these boundary conditions can be usefully combined with one another. To simulate a long channel, periodic boundaries in the horizontal, and reflective boundaries in the vertical direction have to be used.

2.5 Initial conditions

Not only do the boundary conditions need to be specified, but the initial condition must also be specified. In most cases the initial condition greatly influences the subsequent evolution. Initial conditions can either be constructed, or they can be generated randomly.

An important consideration in the generation of initial conditions is that many CA rules conserve some quantities. This can be e.g., the total number of particles, the total momentum, or energy. It can also be some conserved quantities in the system to be modeled, e.g. the number of particles in one row or one column. In generating initial conditions, care must be taken that the intended value for the conserved quantities is reached (especially with random initial conditions), and that the conserved quantities do not generate undesired effects.

2.6 Geometry

An important choice is the selection of a specific lattice geometry. The definition of CA requires the lattice to be regular [12]. The different possibilities considered are for one, two, and three dimensions. Different geometries are constructed to establish the influence they may have on the specific problem.

2.6.1 One dimension

For one-dimensional automata, there is only one possibility: a linear array of cells as shown in the example in Section 2.1.1. The only variations that are possible are the number of neighbors to the left and right that are considered.

2.6.2 Two dimensions

In two dimensions there are three common regular lattices, namely triangular, square, and hexagonal lattices.

The main characteristics of the three lattices are [12]:

- **Triangular lattice:**
  - *Advantage:* Small number of nearest neighbors (three).
  - *Disadvantage:* Difficult to represent and visualize, since it must be mapped to square arrays and display pixels. The mapping is described in the following subsection.
- **Square lattice:**
  - *Advantage:* Simple representation using square arrays and simple visualization.
  - *Disadvantage:* In some cases the square lattice has insufficient isotropy[3].
- **Hexagonal lattice:**
  - *Advantage:* The hexagonal lattice has the lowest anisotropy of all regular two-dimensional lattices. Often this lower anisotropy makes simulations appear more natural, and in some cases it is absolutely necessary to model the phenomena correctly.
  - *Disadvantage:* More difficult to represent and visualize, since it must be mapped to a square lattice. The mapping is described in the following section.

Mapping of hexagonal and triangular lattice to a square lattice

There are two possible mappings of a hexagonal lattice to a square lattice. The first is simple shear, as illustrated in Figure 2.2.

The cells are filled with random colors to show the correspondence. Mathematically the transformation can be described as follows. The cells of the hexagonal lattice have their centers at locations as described in (2.4)

$$\left(i + \frac{(j - 2.floor(\frac{j}{2}))}{2}, j\sqrt{\frac{3}{2}}\right) \text{ with i, j in } Z \tag{2.4}$$

[3]The same properties in all directions.

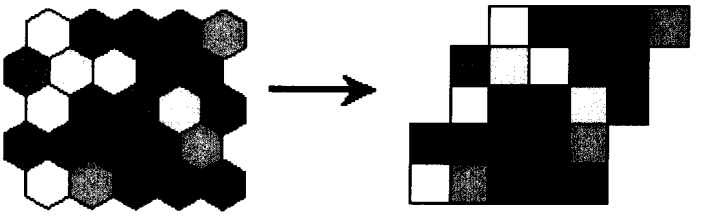

Figure 2.2: Shear mapping of the hexagonal lattice to the square lattice

where *floor(x)* denotes the largest integer smaller or equal to x and where the distance between cells is set to unity. In the shear mapping, the cell number *(i,j)* is mapped to position

$$shear(i,j) = \left(i + floor\frac{j}{2}, j\right) \tag{2.5}$$

The nearest neighbor relations are transformed as shown in Figure 2.3.

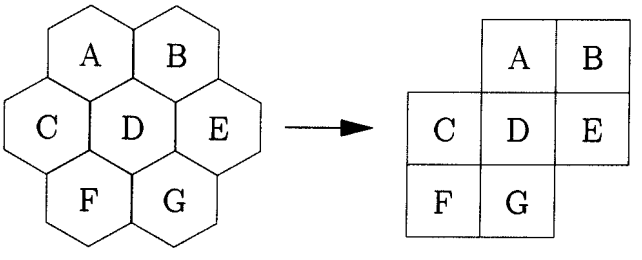

Figure 2.3: Mapping of the nearest neighbor in the shear mapping

The second possible mapping is to shift alternate rows in opposite directions, as shown in Figure 2.4. This mapping is represented by the formula

$$shift(i,j) = (i,j) \tag{2.6}$$

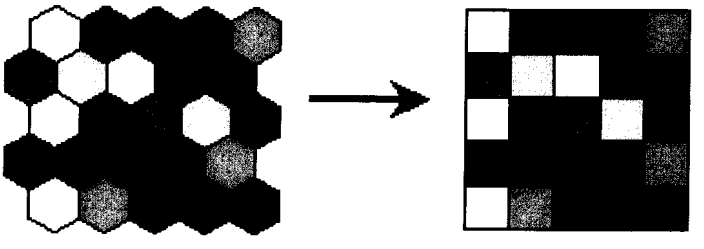

Figure 2.4: Shift mapping of the hexagonal lattice to the square lattice (the cells are filled with random colors to indicate the correspondence)

In this second (shift) mapping, the neighborhood relation is transformed differently depending on whether the row index j is even or odd, as shown in Figure 2.5.

The two mappings each have advantages and disadvantages.

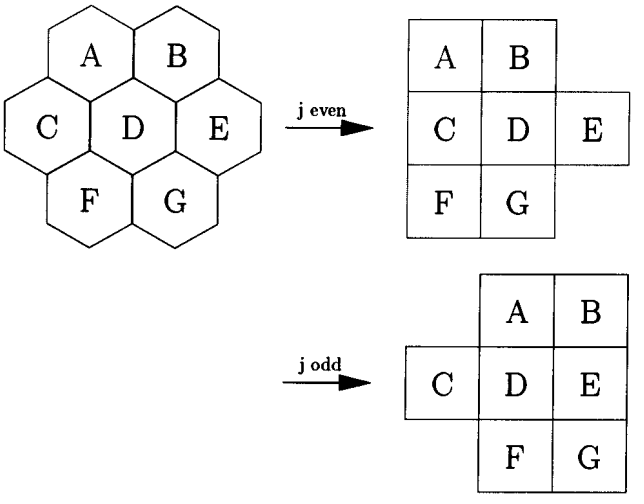

Figure 2.5: Mapping of the nearest neighbors in the shift mapping (j is the row index)

- Shear mapping
  - *Advantage:* Shear mapping has the advantage that the local neighborhood remains uniform in the square representation.
  - *Disadvantage:* It has the disadvantage that boundary conditions are more difficult to implement, and that the transformation has to be reversed for visualization to avoid presenting a sheared image.
- Shift mapping
  - *Advantage:* The shift mapping has the advantage that boundary conditions are as simple to implement as for the square lattice (in fact, the same boundary conditions can be used), and that visualization can also be very simple.
  - *Disadvantage:* The shift mapping has the disadvantage that the neighborhood now depends on the parity of j. This contradicts the definition of CA, where the neighborhood is homogeneous and the rules are homogeneous, too.

We can nevertheless turn the mapped automaton into a real CA by extending the neighborhood to eight cells, which covers the two possible neighborhoods for even and odd rows. One bit is added to the state of the CA to indicate whether the cell lies on an even or an odd row. The rules are modified accordingly and initial conditions are specified so that the indicator bit is correctly set. The rules do not modify this part of the state, it is only used to select the six relevant neighbors out of the total eight neighbors. Thus the mapped CA has uniform rules and neighborhoods again.

The mapping of the triangular automaton to the square lattice is similar to the shear mapping for the hexagonal lattice. In the triangular case, every second cell has a different orientation. The mapping is shown in Figure 2.6.

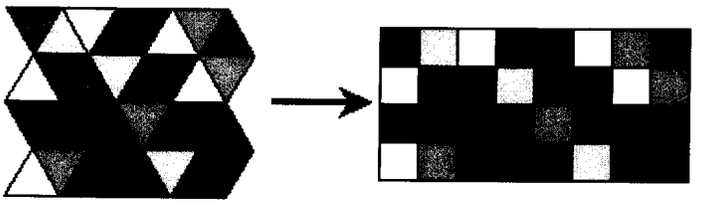

Figure 2.6: Mapping of the triangular lattice to the square lattice (the cells are filled with random colors to show the correspondence)

Each row of triangles is simply mapped to one row of squares. This leads to a non-uniform neighborhood depending, on the parity of the cell in a checkerboard coloring (parity of i+j). The neighborhood is shown in Figure 2.7.

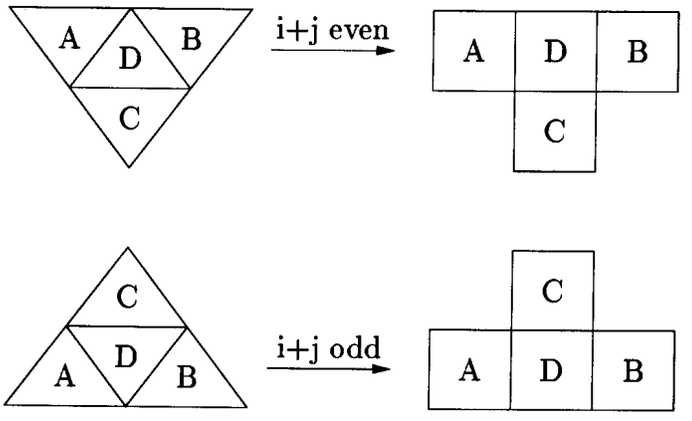

Figure 2.7: Mapping of the nearest neighbors in the triangular mapping (i is the column index and j is the row index)

A second, and somewhat surprising method is to map two triangular cells into one square cell (See Figure 2.8).

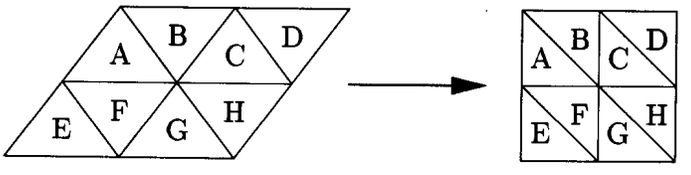

Figure 2.8: Mapping of two triangular cells into one square cell

It is clear that the state of the square cell has to be extended to $S \times S$ to store the states of both triangular cells. The nearest neighbors of the two triangular cells are just the nearest neighbors of one square cell. This method has the advantage of a uniform transition rule and neighborhood, and the disadvantage of a larger state set. Consequently the transition table is larger for the compound cells [12].

2.6.3 Three dimensions

In three dimensions there are many possible regular lattices. One difficulty with three-dimensional CA is the graphical representation (on two-dimensional paper or screen) of the state [12]. In this study is restricted to one- and two-dimensional cases.

2.7 Cell state and precision

According to the definition of CA, each cell is a finite automaton, and therefore the set of states has a finite size [12]. In addition, the set is usually fairly small. There are several reasons for this. Previous researchers have investigated all possible combinations of CA with a given number of states and a given neighborhood size. This is only possible when the set of states is very small, since the number of possible CA combinations with s states and n cells in the neighborhood (including the cell to be updated), is $(s^s)^n$. This number rapidly becomes too big to explore all possibilities as n and s grow:

s	n	s^n	$(s^s)^n$
2	3	8	64
2	5	32	1024
5	3	125	15625

Table 2.2: Number of possible combinations

The second reason for the preference of small state sets, is that it is possible to specify the CA rules explicitly when there are few states and store them in a table (the table size is s^n).

A reason for using large numbers of states is that CA can better approximate a continuous system with more states. This is clear when considering the representation of n particles in CA. When using CA with one bit per cell, $O(n)$ cells are required. When particles are placed randomly in these cells, the fluctuations in particle number in a subset of cells is $O(\sqrt{n})$. Therefore the precision is $O(\frac{\sqrt{n}}{n})$. Conversely, with k bits per cell, $2^k - 1$ particles can be represented in each cell. Therefore only $O(\frac{n}{2^k-1})$ cells are needed, and the fluctuations are $O(\sqrt{\frac{n}{2^k-1}})$. Thus the precision is $O(\frac{\sqrt{\frac{n}{2^k-1}}}{n})$ for different choices of k and n [12].

It is clear from the above that low precision is obtained when using few bits. But modeling with high precision, when the equations are exactly known, is just the domain of numerical analysis and finite element or finite volume methods [12]. CA's are stronger for lower precision, and in situations where the fluctuations are important, or where the interactions in the system are not exactly known. They are more easily described using simple CA rules, rather than partial differential equations.

Even though a simulation using floating-point variables still uses a finite set of states (about 2^{32} states for a single-precision floating point variable), such models are strictly speaking not a CA model. Such programs are usually a computer approximation of models using real, i.e., continuous numbers, and the discretization is poorly controlled. In models with

Number of particles	Bits per cell	Number of cells	Total number of bits	Precision O
n	k	$\frac{n}{2^k-1}$	$k\frac{n}{2^k-1}$	$\frac{\sqrt{\frac{n}{2^k-1}}}{n}$
10^{12}	1	10^{12}	10^{12}	10^{-6}
10^6	1	1000000	1000000	1×10^{-3}
10^6	2	333334	666668	5.8×10^{-3}
10^6	3	142858	428574	3.8×10^{-3}
10^6	5	32259	161295	1.8×10^{-3}
10^6	10	978	9780	3.1×10^{-4}
10^6	15	31	465	5.6×10^{-6}
10^6	20	1	20	10^{-6}

Table 2.3: Possible precision

real variables the names *coupled map lattice* [24, 25] or *finite difference method* might be appropriate for partial differential equation (PDE's).

2.8 Cell rule

Up to this point cells arranged in a lattice represented a static state. Rules have to be integrated into these systems in order to add a dynamical dimension. The function rules is to define the state of the cells for *the next discrete time step.*

In CA a rule defines the state of a cell, which depends on the state of the neighborhood of the cell. The most important aspect of a cellular automaton is the transition rule or transition function. The transition rule depends on the lattice geometry, the neighborhood of cells, and the set cells states. Even though the transition rule determines the evolution, it is in many cases not possible to predict this evolution by other means than explicit simulation (one can prove that this is the case for those automata that can be shown to be computationally universal, such as simple automata like the "game of life"). The most direct specification is to establish the outcome of the transition rule for each possible state configuration in the neighborhood. An important property of CA is that a small change in the transition rule can have dramatic consequences.

Many CA rules merely require information about the number of neighbors in a certain state. This can be used to obtain a simpler specification and also a smaller transition table for the implementation. In classical CA theory an automation rule is called *totalistic* if it depends only on the sum of the states of all cells in the neighborhood. The rule is called *outer totalistic* if it also depends on the state of the cell to be updated [12].

In many cases it is convenient, for the specification and the implementation to split the automation rule into several sub-steps. In theory however, the sub-steps can be combined into a table or rule for the whole step.

An important class of transition rules are *probabilistic* rules. Functions of the transition rule do not have exactly one result for each neighborhood configuration. One or more possible

outcomes are possible with associated probabilities. The sum of probabilities of all outcomes must be one for each input configuration. The probabilistic choices of all cells are independent of one another (uncorrelated) [12].

The introduction of probabilistic rules is very important in many modeling situations, since many natural systems are noisy. In some cases it is sufficient to introduce random initial conditions, which also lead to a noisy or random behavior. In many cases probabilistic rules have advantages [12].

The possibility of such self-organization is therefore a consequence of the irreversibility of the cellular automaton evolution, and the structures obtained through self-organization are determined by the characteristics of the attractors [4].

2.9 Cellular automata classification

According to Wolfram [4], the final results obtained by CA simulations can be divided into four classes. These basic classes of behavior may be characterized empirically as follows:

- **Class 1** evolution leads to a homogeneous state in which, for example, all sites have value 0.
- **Class 2** evolution leads to a set of stable or periodic structures that are separated and simple.
- **Class 3** evolution leads to a chaotic pattern.
- **Class 4** evolution leads to complex structures, sometimes long-lived [4].

The different classes of cellular automaton behavior allow different levels of prediction of the outcome of cellular automaton evolution from particular initial states [26].

The existence of only four qualitative classes implies considerable universality in the behavior of CA. Many features of CA depend only on the class in which they lie and not on the precise details of their evolution [4]. Such universality is analogous, though probably not mathematically related, to the universality found in the equilibrium statistical mechanics of critical phenomena. In this case many systems with quite different detailed construction are found to lie in classes with critical exponents that depend only on general, primarily geometrical features of the systems and not on their detailed construction.

To proceed in analyzing universality in CA, a more quantitative definitions of the classes identified above must first be given. One approach to such definitions is to consider the degree of predictability of the outcome of cellular automaton evolution in conjunction with given knowledge of the initial state.

For class 1 CA complete prediction is trivial regardless of the initial state, since the system always evolves to a unique homogeneous state [4]. Such CA may be considered to evolve to simple "limit points" in phase space. Their evolution completely destroys any information on the initial state. Some exceptional configurations in finite class 1 CA may not evolve to a homogeneous state, but may in fact enter non-trivial cycles [26].

Class 2 CA have the feature that the effects of particular site values propagate only a finite distance, that is, only to a finite number of neighboring sites. Thus a change in the value of a single initial site affects only a finite region of sites around it, even after an infinite number of time steps [4]. The simple structures generated by class 2 CA are either stable, or are periodic, typically with small periods. The set of persistent structures generated by a given class 2 cellular automaton is typically quite simple [26].

Class 2 CA may be considered as "filters" that select particular features of the initial state. A class 2 CA may for example be constructed in which initial sequences 111 survive, but sites not in such sequences eventually attain value 0. Such CA are of practical importance for digital image processing: they may be used to select and enhance particular patterns of pixels. After a sufficiently long time any class 2 cellular automaton evolves to a state consisting of blocks containing nonzero sites separated by regions of zero sites. The blocks have a simple form, typically consisting of repetitions of particular site values or sequences of site values (such as 101010). The blocks either do not change with time or cycle between a few states.

In class 3 CA, the prediction of the value of a site at infinite time would require knowledge of an infinite number of initial site values. Class 3 CA exhibit chaotic *aperiodic behavior*[4]. Although chaotic, the patterns generated by class 3 CA are not completely random. In fact, they may exhibit important self-organizing behavior. In contrast to class 2 CA, the statistical properties of the states generated by many time steps of class 3 cellular automaton evolution are the same for almost all possible initial states. The large-time behavior of a class 3 cellular automaton is therefore determined by these common statistical properties [4].

Class 4 CA are distinguished by an even greater degree of unpredictability than class 3 [4].

For class 1, 2 and 3 CA, fluctuations in statistical quantities are typically found to become progressively smaller as larger numbers of sites are considered. Such systems therefore exhibit definite properties in the "infinite volume" limit. For class 4 CA, it seems likely that fluctuations do not decrease as larger number of sites are considered, and no simple smooth infinite volume limit exists. Important qualitative effects can arise from special sequences appearing with arbitrarily low probabilities in the initial state [26].

The complexity apparent in the behavior of class 4 CA suggests the conjecture that these systems may be capable of universal computation. The initial sequence may be considered as a program and data stored in computer memory, and part of the final sequence may be considered as the result of the computation. CA may be considered as computers, their initial configurations represent programs and initial data, and their configurations after a long time contain the results of computations [4]. A system is a universal computer if, given a suitable initial program, its time evolution can implement any finite algorithm [27].

However, if CA in the class are indeed capable of universal computation, then this dependence may be arbitrarily complex, and the behavior of the system can be found by no procedure significantly simpler than direct simulation. No meaningful prediction is therefore possible for such systems [26].

[4]Periodically returns to the same state.

The results from a single cell rule may indeed also hold promise for encryption of secure data. The evolving nature and invisibility of CA would make it impossible to encrypt the data without all the necessary CA variables.

The four classes of CA may be distinguished by the level of predictability of their "final" large time behavior given their initial state [26].

2.10 CA internal representation possibilities

CA as an idealization can be used for an approximation for the governing equations in structural analysis in a similar way as FDM. For structural analysis, the physical geometry has to be converted into a discrete mesh. For complex geometries it will not be possible to create a mesh with equally sized cells. In section 2.6 we discuss methods by which triangular and hexagonal physical meshes can be represented on a square CA computational map. A problem with this method is keeping track of the neighbors for each cell (indexing complications). As for the FDM mesh, different mesh sizes will complicate the calculation. The cells on the edges have fewer neighbors to keep track of than the cells in the middle. Table 2.4 shows how these neighbors can vary

Mesh type	Maximum neighbors	Minimum neighbors
Triangular	3	2
Square	8	3
Hexagonal	6	2

Table 2.4: Neighborhood size difference for difference mesh types

Thus as far as computational methods are concerned, it is only necessary to deal with a square lattice. As mentioned previously, the problem arises of keeping track of the neighbors in the physical geometry in the CA map. Since CA do not require the solution of systems of equations, an internal representation of the system under consideration may be used. Table 2.5 illustrates an example of data storage. The first columns store the physical coordinates of the cell center. The next set of columns represent the cell states. Each cell state corresponds to a physical quantity, (in structural analysis for example, both displacements and stresses may be stored), even though they evolve according to different cell rules, and could possibly be a function of different cell states. The following column contains the neighborhood flag, with each flag number corresponding to a specific number of neighbors and configuration. The last columns of Table 2.4 give the indices of the neighbors in accordance to the neighborhood flag.

Each column does not have to be in the same array. There must merely be corresponding indices. For instance, the first few columns that contain the coordinates require real values to be stored. For the remainder of the columns, an integer value is sufficient and would save memory.

The methodology mentioned above allows for the complicated expansion of the cell rule definition. Most CA simulations in existence have no need for a complicated representation

Physical coordinates		Cell States		Neighbor Flag	Neighbor Indexes	
x_1	y_1 ...	State 1_1 ...	State m_1	Flag 1	Index 1_1 ...	Index n_1
x_2	y_2 ...	State 1_2 ...	State m_2	Flag 2	Index 1_2 ...	Index n_2
x_3	y_3 ...	State 1_3 ...	State m_3	Flag 3	Index 1_3 ...	Index n_3

Table 2.5: Internal Computer Representation for a complicated CA mapped to a square computational lattice

since they are idealized with simple geometries. This implies simple mapping and "neighbor look-up". Now it is possible to not only represent for instance the displacements for each cell but also the stresses. The cell rules can be formed by using a combination of the cell states. One of the cell states can be represented by two possible states, one for existence and one for dead, where dead implies that the material of the cell is physically not present. If a cell dies, the program call easily changes the neighborhood flag and indices of all its neighbors to ensure that the dead cell is not used for further computations. If the cell is not used for further computations it is not removed from the array the indices of each cell whether dead or alive remains unchanged. It is also possible to add cells to the bottom of the index in a similar fashion. Table 2.5 offers a method in which the CA can be handled such that the only significant problem lies with the initial setup of the cell map. The computational part of the problem is simplified and flexible and more attention can be given to the computational parameters.

2.11 Partial differential equations, continuous systems and cellular automata

The mathematical formulation of most problems in science involving rates of change with respect to two or more independent variables leads to Partial differential equations (PDEs) or a set of such equations [28]. Equation 2.7 shows a typical two dimensional second-order equation.

$$a\frac{\partial^2\phi}{\partial x^2} + b\frac{\partial^2\phi}{\partial x\partial y} + c\frac{\partial^2\phi}{\partial y^2} + d\frac{\partial\phi}{\partial x} + e\frac{\partial\phi}{\partial y} + f\phi + g = 0 \tag{2.7}$$

Where a, b, c, d, e, f and g are a function of independent variables x and y and the dependent variable ϕ. By definition (2.7) is considered to be *elliptic* when $b^2 - 4ac < 0$, *parabolic* when $b^2 - 4ac = 0$, and *hyperbolic* when $b^2 - 4ac > 0$.

Several conditions are necessary for the overall behavior of a system with discrete elements to appear continuous. First, continuum behavior must be associated with some kind of extensive quantity. Such a quantity must be additive, and must be conserved in the dynamical evolution of the system. In a gas, one example of such a quantity is particle number. Other examples are energy and momentum [7].

CA rules may be compared with traditional approaches to emulating these continuum sys-

tems on digital computers. In the conventional approach, one starts from partial differential equations, then makes discrete numerical approximations to them. These approximations involve considering a discrete lattice of points. But unlike in CA, each of these lattice points carries a continuous variable which represents the precise value of a continuum quantity, such as particle density, at that point. In actual computer implementations, the continuous variable is represented by a floating-point number, say 64 bits in length. The number is updated in a sequence of time steps, according to a discrete rule. The rule in general involves arithmetic operations, which cannot be carried out precisely on the finite precision number. As a result, low-order bits of the number are truncated. Numerical analysis has studied in detail the propagation of such round-off errors, and has suggested schemes which minimize their effects. Instead of systematically truncating the numbers, their low-order bits are modified according to dynamics which yields effectively random behavior. The result is similar to random round-off, but includes a precise particle conservation law. By adjusting the number of unary and digital bits, one can determine the trade-offs between cellular automaton and numerical analysis approaches [7].

One of the major issues in numerical analysis is *convergence*. This is very difficult to prove for all but the simplest equations and the simplest schemes. But in CA, the analogue of convergence is the process of coming to equilibrium. Thus the problem of convergence is related to a fundamental problem of physics [7].

CA may thus be considered as discrete idealizations of the partial differential equations often used to describe natural systems [4]. Physical systems containing many discrete elements with local interactions are often conveniently modeled as CA [5]. Any physical system satisfying differential equations may be approximated as a cellular automaton by introducing finite differences and discrete variables [29]. The time evolution given particular initial conditions is represented by a trajectory in the space of variables described by the differential equations. In the simplest cases (such as those typical for chemical concentrations described by the Boltzmann transport equations), all trajectories tend at longer times to a small number of isolated limit points, or approach simple periodic limit cycle orbits [5].

While there is every evidence that the fundamental microscopic laws of physics are intrinsically reversible (information-preserving, though not precisely time-reversal invariant), many systems behave irreversibly on a macroscopic scale and are appropriately described by irreversible laws. For example, while the microscopic molecular interactions in a fluid are entirely reversible, macroscopic descriptions of the average velocity field in the fluid, using, say, the Navier-Stokes equations, are irreversible and contain dissipative terms. CA provide mathematical models at this macroscopic level [4].

The second law of thermodynamics implies that isolated microscopically reversible physical systems tend with time to states of maximal entropy and maximal disorder [5]. However, dissipative systems involving microscopic irreversibility, or those open to interactions with their environment, may evolve from disordered to more ordered states. In almost all cases, cellular automaton evolution is irreversible [26]. Every configuration in a cellular automaton has a unique successor in time. A configuration may however have several distinct predecessors [30]. The presence of multiple predecessors implies that the time evolution mapping is not invertible but is instead contractive. The CA thus exhibits irreversible behavior in which

information on initial states is lost through time evolution [30]. The existence of configurations with multiple predecessors implies that some configurations have no predecessors [5]. These configurations occur only as initial states, and may never be generated in the time evolution of the cellular automaton. They appear on the periphery of the state transition. Their presence is an inevitable consequence of irreversibility and of the finite number of states [30].

Trajectories in the configuration space for CA therefore merge with time, and after many time steps, trajectories starting from almost all initial states become concentrated onto attractors [26]. These attractors typically contain only a very small fraction of possible states. Evolution to attractors from arbitrary initial states allows for self-organizing behavior, in which structure may evolve at large times from structure-less initial states. The nature of the attractors determines the form and extent of such structures. Statistical mechanics, and the continuum equations derived from it, provide a considerably reduced description of these systems [7].

Chapter 3

Cellular automata in structural analysis

3.1 Introduction

In this chapter the CA parameters in structural analysis and the CA implementation for structural analysis are described. The problems to be solved using CA are formulated and the method of determining the optimum cell rule is discussed. Finally example problems are analyzed and the results compared to current established methods in structural analysis.

The question whether CA can model not only general phenomenological aspect of our world, but also model the laws of physics themselves was raised by Toffoli [31]. A primary theme of this research is the computational models of physics that are *information-preserving*, and thus retain one of the most fundamental features of microscopic physics namely reversibility [32]. CA have been used to provide extremely simple models of common differential equations of physics, such as the heat and wave equations [33] and the Navier-Stokes equation [34, 35]. CA also provide a useful discrete model for a branch of dynamical systems theory which studies the emergence of well-characterized collective phenomena such as ordering, turbulence, chaos, etc. [36, 37]. Systematic research of CA in this context was pioneered by Wolfram and studied extensively by him, identifying four qualitative classes of CA behavior as discussed in Section 2.9.

In Appendix A a brief summary of the governing equations in structural mechanics as well as current approximation methods in structural analysis are discussed. Knowledge of these equations and the approximation methods is essential in order to determine how well the CA approximation will compare with these methods. Most of the approximations discussed in Appendix A are derived from the governing equations and their relations.

The discrete formulation of CA makes it very difficult to predict the outcome of the method, as is clearly shown in [5]. In most cases the simulation requires too many components, and a direct approach of construction fails to predict the outcome of the simulation. One must instead attempt to extract the mathematical essence of the process by observing the results obtained by various formulations.

The governing equations in structural mechanics describe the way in which the material in a structure adjusts itself to absorb energy input into the structure. These fundamental mathematical components are independent of the size of the problem, making it possible to analyze small structures to understand the mechanisms common to such problems. The possibility embedded in such an approach is to identify fundamental mathematical mechanisms that are common to many different structural systems. Such commonly used mechanisms would correspond to universal features in the behavior of very different complex natural systems [4].

3.1.1 CA parameters in structural analysis

To analyze a structural anlysis problem using cellular automata, various parameters have to be taken into consideration. Section 2.6 deals with established methods of handling the physical geometry to establish a structural mesh. In addition to the mesh that must be taken into consideration, there are other aspects to consider in CA simulations for practical problems in structural analysis:

1. How many cell states are sufficient to simulate the deformation of a structure?
2. How would one determine the cell rule to be used?
3. How would the boundary conditions be obtained and implemented?
4. What would the influence of the initial condition be and how would the initial conditions be obtained?
5. Which neighborhood size would be easy to handle yet sufficient to simulate the problem at hand?

We have already discussed the accuracy dependency of the CA solution in Section 2.7 and showed how this depends on the number of cell states. We now need to attempt a prediction on the influence the number of cell states has on the simulation time of the problem at hand.

Example problem: Influence of the number of cells and number of cell states

A very simple model is now set up, using a simple one-dimensional model in which we change the number of cell states and the number of cells in the system: Consider a bar which is clamped at one end and with an axial force applied, as depicted in Figure 3.1.

We assume that the point where the force is applied has the biggest displacement (the maximum or highest cell state used) and that the clamped side has zero displacement (lowest cell state used). Obviously the boundary conditions prescribe the smallest cell state as 0.

The discrete nature of CA is one of the major advantages when used as a computational mechanics method, since it allows for easy interaction on digital computers. It is logical to use cell states that can be stored effectively in a conventional computer's memory. Conventional computers use binary logic, thus we will use cell states that are of the form 2^n, where n

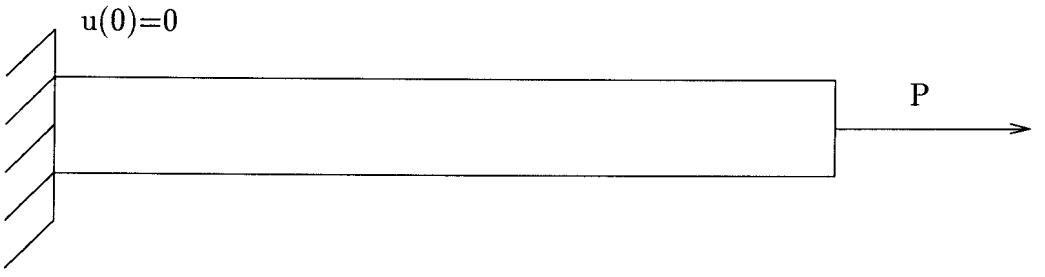

Figure 3.1: One-dimensional bar problem

indicates the number of bits used in memory. In computer programming languages, integers are usually limited to $2^{(16-1)}$, which is equal to 32768. One bit is subtracted as it is used internally to indicate the sign of the number. Since this number is easily handled by current programming languages, the maximum number of cell states is selected as 32768. This also results in a sufficiently accurate approximation in accordance with Section 2.7.

The cell rule used is the basic two-dimensional FDM equation as set out in Table A.1. By not using the information of the cell under consideration Laplace's equation (A.1) is satisfied and the cell rule is *totalistic*.

For this static problem, the solution will converge to a fixed state, which will place our CA in *class 2*. Thus the CA will reach the deformed state independently of its initial conditions. The number of cell states used will not influence the final state achieved but the evolution will indeed be different. For comparison, the initial cell state was set to 0 except at the natural boundary. This is performed for a different numbers of cells, and the result is shown in Figure 3.2.

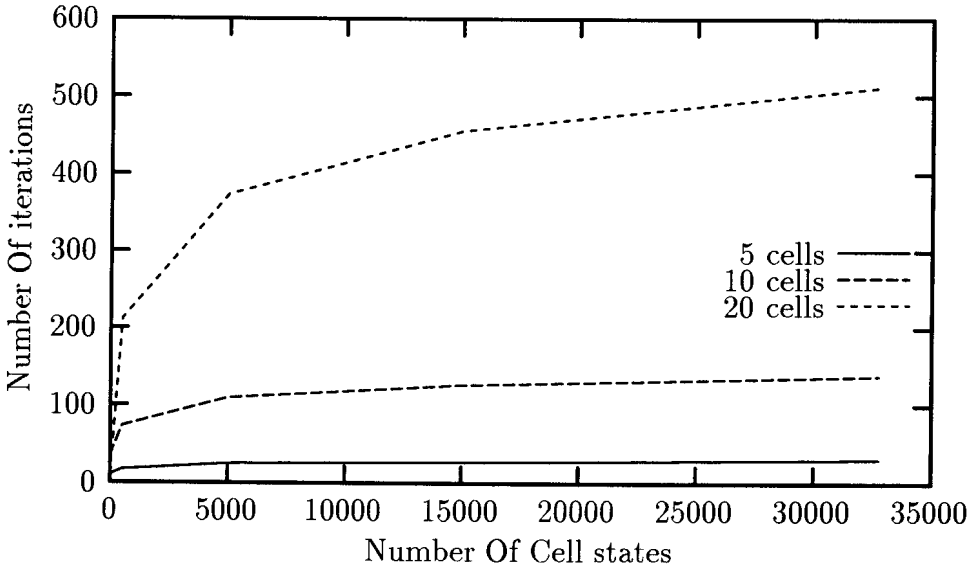

Figure 3.2: One-dimensional computational cost comparison

From Figure 3.2 it is clear that the cost of evaluation dramatically increases with the number of cells in the simulation. The number of cell states initially has a large influence on the

number of iterations (for few cell states), but as the number of cell states increase, the influence becomes small. This implies that the influence in processing time between a medium number of cell states and a full integer is not significant. The evolution of a CA with different numbers of cell states would, however, be different, although the model will reach the same final state.

From the comparisons performed above it is apparent that the combinations of variables that can influence the system are endless. One must instead attempt to logically identify combinations that are both practical and capable of delivering an extensive understanding of how a structural approximation would react on such parameters.

3.1.2 CA implementation

The values of interest in structural analysis are not directly prescribed by the boundary conditions (since the boundary conditions are typically prescribed in terms of displacement constraints, while the values of interest are displacements and stresses or strains). For CA to be applicable to general structural analysis, a method has to be found to "translate" the prescribed boundary conditions into a form that can be used by the CA.

Comparatively the BEM calculates values for displacements at the boundary of the structure and then calculates displacements at internal points in a separate step. FEM calculates all the displacements simultaneously and then calculates the stress at the integration points of each element in a separate step.

Since the stresses in a structure are directly related to the strains, it seems natural to only calculate only the displacements, then the stresses can be calculated from the strains. In BEM, only boundary conditions are approximated [38]. BEM can be used to calculate the displacements at the structures boundary and then this data could be transferred to the boundary of the CA simulation. BEM thus only calculates the data required for the CA simulation and the second step in the BEM is replaced by the CA.

3.2 Problem formulation

3.2.1 Problem descriptions

For a CA approximation to be universally acceptable, the approximation to the structural problem at hand must be able to handle various displacement and stress fields. This implies that more than one type of problem has to be considered. Figures 3.3 through 3.5 depict the problems considered in the remainder of this chapter.

The first problem depicted in Figure 3.3 is well suited to showing how the force applied at a point dissipates through a structure. Since the problem is square, it presents no problems for meshing or neighborhood definition.

For the second problem, we consider a similar problem to problem 1, obtained by simply modifying the boundary condition (the constraints, or *natural boundary conditions* as described in Appendix A). This problem is discussed by Hajela [39] and it is a classical problem

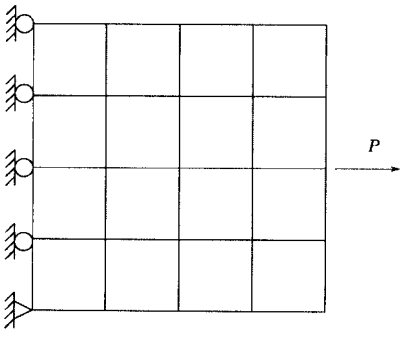

Figure 3.3: CA problem 1

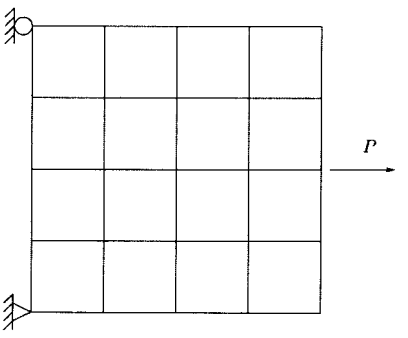

Figure 3.4: CA problem 2

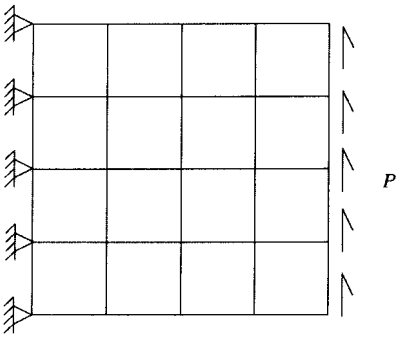

Figure 3.5: CA problem 3

used in topology optimization where the geometry of the structure is optimized. When a point load is applied in the numerical problem, the stresses at the point where the load is applied, tend towards infinity. It is physically impossible to apply a point load to a real structure.

The third problem is therefore subjected to a distributed load, which yields a complex stress field. However, the problem is still sufficiently simple for easy implementation. This problem is shown in Figure 3.5.

3.2.2 Neighborhood descriptions

For two-dimensional models, there are two computational FDM molecules frequently used as shown in Table A.1. These models are derived directly from the governing differential equations and are symmetrical. In CA terminology these computational molecules represent the cell rule, the difference being the discrete nature of the CA formulation, were the FDM method is continuous in nature.

For the first two-dimensional rule, no other combination of variables that are linearly independent from the original rule will result in convergence. This is because this rule is a simple averaging rule as implied by Laplace's equation A.1. Any other combination of variables will obviously result in a diverging rule.

The second molecule consists of a fixed neighborhood which is larger than the first one. Furthermore, if we presume that we are dealing with an isotropic[1] material, it would be logical to assume a symmetric cell rule similar to the one originally derived for FDM. This means that there are four possible variables that would form the cell rule. The four variables are depicted in Figure 3.6. No predictions can be made regarding which combination of variables would converge to a steady state, or even that if these variables have converged, that they will predict a correct solution.

Both of the neighborhoods used in FDM are capable of delivering good approximations. The definition of CA neighborhood however allows for the definition of any neighborhood to be used as long as it is uniform. To minimize the number of boundary cells required from an external source, (no more than one row of boundary cells), whilst maximizing the number of

[1]The same governing rule holds in all directions of the material.

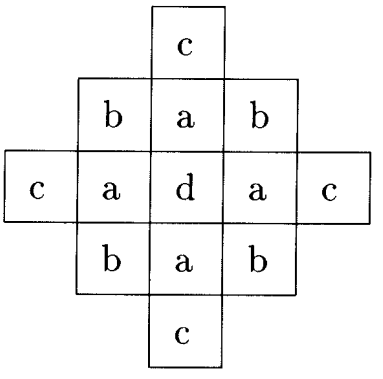

Figure 3.6: Extended neighborhood description

cells available for use, we can return to classical CA neighborhoods and conclude the need for a Moore neighborhood as defined in Section 2.3. The variables are defined in Figure 3.7.

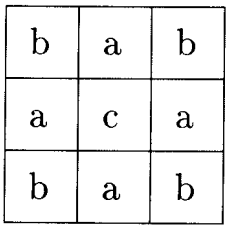

Figure 3.7: Moore neighborhood description

3.2.3 Analysis criteria

Introduction

As mentioned in structural analysis there is more than one value of interest to describe the state of a structure. These values are directly related to one another, e.g. the deformation in one direction is related to the deformation in the directions perpendicular to it by Poisson's ratio v. Deformation is caused by forces acting on the structure, this in turn induces stresses in the structure. When the stresses in the structure exceed the stress that the material is capable of handling the material fails and implicitly the structure also fails. But a structure might be considered as having failed when the deformations become large even if the stresses remain sufficiently low. (Such failures are considered as geometrically non-linear and beyond the scope of this investigation.)

To use CA as a computational mechanics method applicable to structural mechanics we need to isolate the values of interest in our simulation. These values should be sufficient to describe the deformation and failure of the structure as well as being capable of being competitive in terms of the computational time required.

Displacement analysis

The values approximated in the FDM are displacements. Herein the first CA simulations are also performed using displacements. For an initial cell rule we use the first two-dimensional finite difference molecule shown in Table A.1, which makes use of only the four immediate neighbors. Because BEM calculates values all along the edge of the structure, the cell at the boundary could be given a fixed value for the CA simulation. The difference between this example and the work done by Hajela et al. [39] is that they presumed that fixed boundary cells (the boundary condition of the CA) only appeared at the points where the structural boundary conditions are applied. They assume free edges for the other cells. However, since the BEM calculates these points, in a combined method it would be senseable to use them even though the approach of Hajela is more general.

Example problem: Separated displacement components

For the first attempt the two displacements are analysed separately; viz. the x displacements are calculated independently from the y displacements after the boundary conditions are obtained. The maximum cell state is set to a full integer for high precision and this value is prescribed as the maximum displacement encountered. This means that the difference in displacement between every cell state is directly proportional to the maximum displacement encountered at the boundary.

The boundary conditions of the structure are then imported from BEM. In order to benchmark how well the CA simulation performs, the problem is compared to the same problem solved using FEM as discussed in Section A.3.3. The FEM model is set up to be more refined by placing a FEM node at the center of each cell (where the value for the cell is calculated). In this manner, four 4-noded isoparametric FEM elements are used to model one cell. Using four FEM elements produced a more accurate solution than BEM, and the FEM solution is considered an ideal solution. With BEM as an input, CA could not achieve the same accuracy and for this reason, the BEM code was also used to calculate internal values. Both were then compared to the FEM analysis by the error function, as given below.

$$\epsilon = \sum |f_1 - f_2| \tag{3.1}$$

The calculations are performed separately for the x and the y components and the results are shown in Table 3.1.

Number of	BEM		CA	
cells	x	y	x	y
5×5	4.7041E-2	1.3452E-1	1.3167E-1	1.4656E-1
10×10	9.3842	1.5656	9.5645	1.77099

Table 3.1: Displacement error comparison of BEM and CA in comparison to FEM

A number of features are apparent from Table 3.1. Firstly, the CA simulation delivers similar results to BEM. This is a good indication since CA cannot be expected to deliver

better performance based on the same boundary calculations as BEM. In order to determine what difference the cell rule makes to the results, the size of the neighborhood is increased. There are no other possible cell rules to be used with this neighborhood that are able to converge. The fact that the two displacements are calculated separately means that the analysis is run twice. Since it is clear from Figure 3.2 that this would result in a dramatic increase in the processing time required as the number of cells increases, it is obviously not a viable option.

All attempts to combine the displacements into one cell by making use of the Poisson's relation failed, since none of the combinations constructed were able to correctly keep track of the direction of each component of the combined displacement, because an extension in one direction meant a narrowing in the other direction. This is too much information to be contained in one cell state while the physics of the problem at hand are also being captured.

Stress analysis

A single criterion is obviously desirable that describes the behavior of the structure by one cell rule. Such a criterion (von Mises stress) was previously used by Hajela [39].

The von Mises stress is used as a criterion in determining the onset of failure in ductile materials [40]. The failure criterion states that the von Mises stress σ_{VM} should be less than the yield stress σ_Y of the material [40]. In the inequality form, the criterion may be stated as

$$\sigma_{VM} \leq \sigma_Y \tag{3.2}$$

where the von Mises stress σ_{VM} is given by

$$\sigma_{VM} = \sqrt{I_1^2 - 3I_2} \tag{3.3}$$

In (3.3), I_1 and I_2 are the first two invariants of the stress tensor [40]. I_1 and I_2 are given by

$$\begin{aligned} I_1 &= \sigma_x + \sigma_y + \sigma_z \\ I_2 &= \sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x - \tau_{yz}^2 - \tau_{xz}^2 - \tau_{xy}^2 \end{aligned} \tag{3.4}$$

In terms of the principal stresses, σ_1, σ_2 and σ_3, the invariants can be written as

$$\begin{aligned} I_1 &= \sigma_1 + \sigma_1 + \sigma_3 \\ I_2 &= \sigma_1\sigma_2 + \sigma_2\sigma_3 + \sigma_3\sigma_1 \end{aligned} \tag{3.5}$$

Hence, (3.3) can be written as

$$\sigma_{VM} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} \tag{3.6}$$

The equation is simplified in the case of plane stress and plane strain. For plane stress we obtain

$$\begin{aligned} I_1 &= \sigma_x + \sigma_y \\ I_2 &= \sigma_x\sigma_y - \tau_{xy}^2 \end{aligned} \tag{3.7}$$

and for plane strain we obtain

$$\begin{aligned} I_1 &= \sigma_x + \sigma_y + \sigma_z \\ I_2 &= \sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x - \tau_{xy}^2 \end{aligned} \tag{3.8}$$

where $\sigma_z = v(\sigma_x + \sigma_y)$.

The von Mises criteria will always have a positive value. This value is an indication of how far the structure is from failure at any point within the structure. It is thus a sensible criterion to ascertain the overall performance of the structure.

3.3 Determination of the optimum cell rule

3.3.1 Motivation

Given generic macroscopic behavior, it is important for both theoretical and practical purposes to try and find the simplest microscopic dynamics which can reproduce the macroscopic behavior. One may, for example, seek the simplest cellular automaton rule which reproduces a particular form of continuum behavior[2].

In the simplest CA, one considers rules which specify the new value of a single site in terms of the values of a neighborhood of sites around it at the previous time step. In most such rules, no additive quantities can be conserved. In addition, such rules are usually highly irreversible, so that they evolve towards attractors which contain only a subset of the possible states [7]. One needs to identify these rules that govern the physics of the problem. These rules would create macroscopic behavior common in all structural problems.

In problems where such rules are difficult to identify, or situations where the physics of the problem is not properly understood, alternative strategies must be implemented [39]. Hence

[2] "Simplest" can be defined for example, as requiring minimum storage space and minimum number of logical operations to implement [7].

one can consider finding minimal cellular automaton rules by various iterative and adaptive procedures [7]. This process of discovering "the laws of the universe" for a given problem is central to an ability to extend the approach to diverse analysis and design problems [39]. Before a rule can be extracted for a certain problem or set of problems one needs to define a criterion by which the performance of a certain rule can be evaluated. Once such a criterion is in place, an attempt can be made to identify the governing rule by an artificially intelligent selection process.

3.3.2 Cell rule comparison definition

From the description of the genetic algorithm (GA) in Appendix B.2 we define the best cell rule as the "fittest individual". In order to determine the performance of our CA model, we need to define an *error* or *residual* with which all the points in the system can be combined to ascertain the "fitness" of the rule. For each cell in our discrete mesh we can define an error given by

$$\epsilon = f(x_k) - y_k \quad \text{for} \quad 1 \leq k \leq N \tag{3.9}$$

Here, N is the total number of cells in the system, y_k is the real function value at point k and $f(x_k)$ is the approximate value, in this case the value obtained by our CA simulation. We now have to combine all these small errors into a function value that can be used to describe the "fitness" of the rule. In classical curve fitting there are three criteria by which this can be done, as shown in (3.10) [41]. The CA simulation actually performs a three-dimensional curve fit, where the stress is seen as the function value dependent on the x and y coordinates.

$$\begin{aligned} \text{Maximum error:} \quad & E_1 = \max|\epsilon| \\ \text{Average error:} \quad & E_2 = \tfrac{1}{N}\textstyle\sum_{k=1}^{N}|\epsilon| \\ \text{Root-Mean-Square(RMS) error:} \quad & E_3 = \left[\tfrac{1}{N}\textstyle\sum_{k=1}^{N}|\epsilon^2|\right]^{\frac{1}{2}} \end{aligned} \tag{3.10}$$

The RMS error is considered as the best criterion, since it penalizes the solution heavily for a single point with low accuracy, but uses all the values in the system to calculate the total error. We have deduced that the von Mises stress criterion is a good value to be approximated by the CA simulation, since it is described by one physical positive quantity, but it is sufficient to describe failure in linear problems.

With a discrete optimization problem, it is clear that in finding the best cell rule, traditional gradient-based approaches to numerical optimization would be unsuccessful. With all the numbers being discrete values, the optimization is more difficult than for continuous problems. If we consider that, for the extended von Neumann neighborhood each of the four optimization variables has 64 possible discrete values, this means that the total number of combinations are $64^4 = 16777216$. If we presume that one second were required per function evaluation (as a result of the rule, the problem either converged to a single solution or diverged until the maximum number of iterations is exceeded), evaluating all possibilities

would take 194.18 days. A smart searching (optimization) method therefore has to be used to extract the optimum cell rule. GAs make use of the survival of the fittest strategy found in nature to search the solution space of a function as described in Appendix B.2.

3.4 Problem analysis and results

3.4.1 Extended neighborhood

Consider Problem 1, shown in Figure 3.3. In accordance with Figure 3.6 (the extended Moore neighborhood), there are four variables to be used in the optimization methodology. We presume that the cell rule will involve some type of averaging to be able to reach a steady state prescribed by Poisson's equation. Presuming that these variables will have discrete integer values, similar to the FDM rule, we define a maximum and minimum value for each number. For the sake of computational efficiency we presume that each cell can have an influence of $2^6 = 64$. This means that the variables a, b and c can have values between -31 and 32 and the variable d can vary between 1 and 64. The rule for a cell with coordinates (i, j) is shown in

$$\begin{aligned} Cell_{ij}^{\text{New}} = (a(&Cell_{i+1,j} + Cell_{i-1,j} + Cell_{i,j+1} + Cell_{i,j-1}) \\ -b(&Cell_{i+1,j+1} + Cell_{i-1,j+1} + Cell_{i-1,j-1} + Cell_{i+1,j-1}) \\ -c(&Cell_{i+2,j} + Cell_{i-2,j} + Cell_{i,j+2} + Cell_{i,j-2}))/d \end{aligned} \tag{3.11}$$

This rule uses the value of the cell under consideration and is consequently considered *outer totalistic*. It is clear that not all possible combinations will converge. Rules that diverge are monitored, stopped and then penalized with a large error, to prevent propagation into future generations in the optimization process.

The GA search is performed on a small mesh size which makes the simulation run reasonably fast. The FEM analysis is now used to prescribe boundary conditions for the CA simulation. The rules obtained for each specific number of cell states are shown in Table 3.2.

Table 3.2 shows the computations performed with different variables. The first column describes the maximum amount of cell states used in the form 2^n, with $3 \leq n \leq 15$, which uses the optimal amount of computer resources. Finally this is done with a high precision continuous 64 bit floating point variable for comparison with the traditional FDM rule. The second column shows the total RMS error for all the points in the system. The fact that the total error is divided by the amount of points used enables us to compare different mesh sizes. The cost is the amount of iterations taken for convergence of the whole system and the variables show the rules extracted in accordance with (3.11).

The RMS error decreases as more cell states are used and then starts increasing again. The lowest RMS error is obtained at 512 cell states. A very interesting result is seen with the rules obtained:

- The higher the number of cell states, the more often the same cell rule is obtained.

Cell states	Error RMS	Cost	a	b	c	d
8	0.43291552E-01	5	2	-15	0	56
16	0.27817578E-01	5	-2	-16	-4	62
32	0.15375927E-01	6	-1	-20	2	62
64	0.79718766E-02	8	0	-14	0	52
128	0.56412691E-02	8	2	-15	0	64
256	0.57616191E-02	10	8	-6	0	54
512	0.26902889E-02	17	16	-1	2	59
1024	0.31755326E-02	17	13	-2	1	55
2048	0.35064670E-02	21	16	-1	1	63
4096	0.40553210E-02	20	13	-2	1	55
8192	0.34612693E-02	23	14	-2	1	59
16384	0.35566437E-02	24	14	-2	1	59
32768	0.35830220E-02	25	14	-2	1	59
Cont.	0.36324311E-02	46	14	-2	1	59

Table 3.2: Optimized rules for 8×8 mesh on Extended neighborhood FEM problem 1

- A problem with the continuous values used for accuracy, is that almost twice the number of iterations are required to obtain convergence on the same cell rule with closely the same error.

None of these rules correspond with the FDM equation derived and shown in Table A.1. Smaller numbers of cell states display higher errors than the larger numbers. The RMS errors obtained are compared in Figure 3.8.

The errors achieved with the optimized rules are lower than those achieved with the FDM rule. The FDM rule formulation uses continuous variables and it would be expected that the FDM rule would perform far better with more cell states than with lower numbers of cell states. However, the error remained almost constant and independent of the number of cell states used for large numbers of cell states.

Just as important as the error achieved, is the number of iterations required for the solution to converge. Bigger mesh sizes take longer to update their cell states with each iteration and since the influence of the boundary values propagates through the structure, require more iterations to converge. The cost in terms of the number of iterations required is shown in Figure 3.10.

The rules obtained converge in fewer iterations than the FDM rule, and are less dependent on the number of cell states used. This could mean a large reduction in computational cost for the rules used in large problems since each iteration requires a CA adjustment of each cell.

The RMS error indicates how close the approximation is to the real solution in terms of the curve fit. But, as for common structural analysis, we need to display the stresses in a form which is universally understandable in engineering. The von Mises stress plots for the

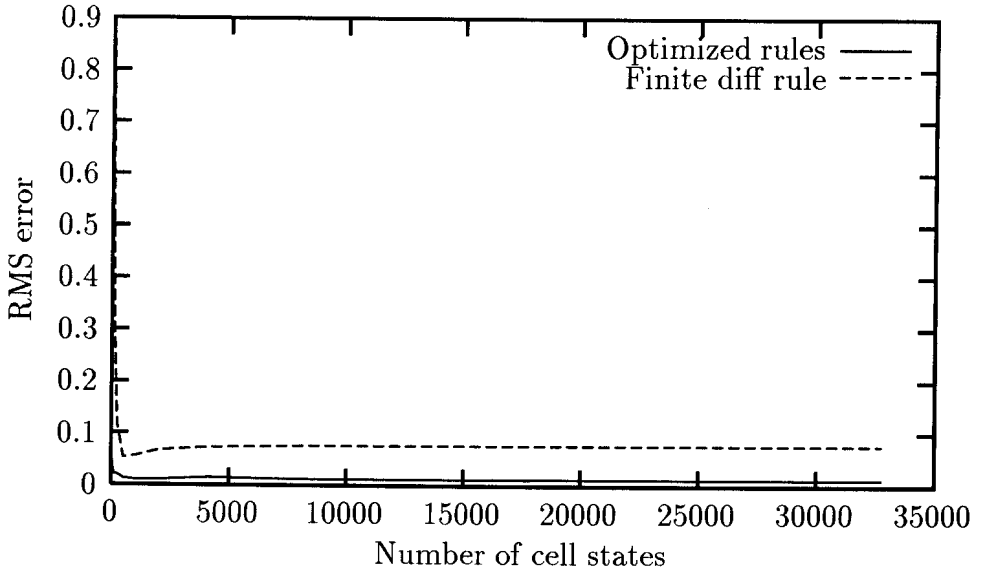

Figure 3.8: Extended neighborhood RMS error comparison

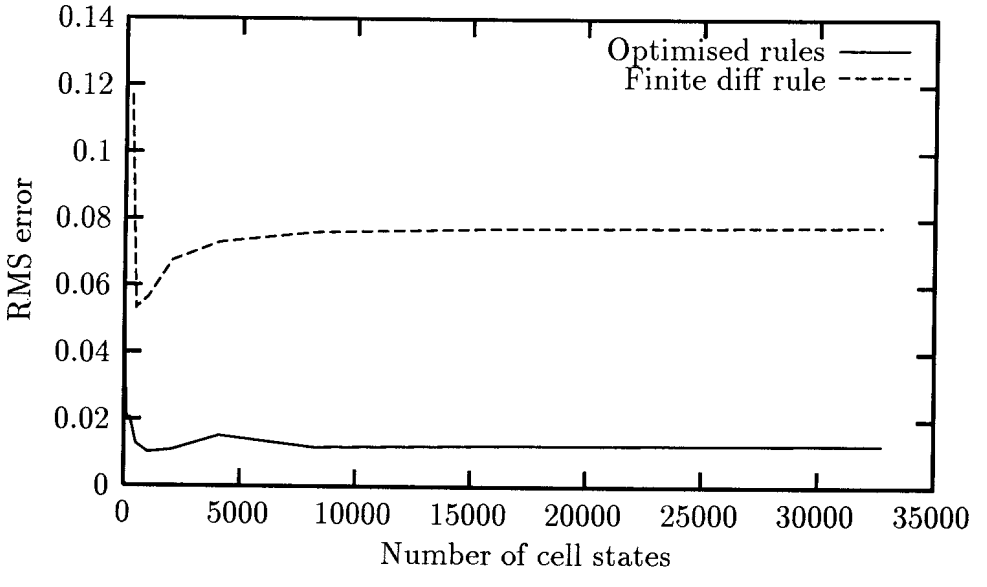

Figure 3.9: Extended neighborhood RMS error closer comparison

various stresses are given in Figures 3.11 to 3.16.

The results obtained seem promising since the resultant rules out-perform the FDM rules in all cases. The stress plots seem accurate for low numbers of cell states. Before these tendencies can be considered as a phenomenon, a few remarks should however, be made:

- Is it practical to use an extended neighborhood since the neighborhood requires that two rows of imported cells be used as the boundary conditions?
- If this good performance is indeed true, is it also true for other, more practical, prob-

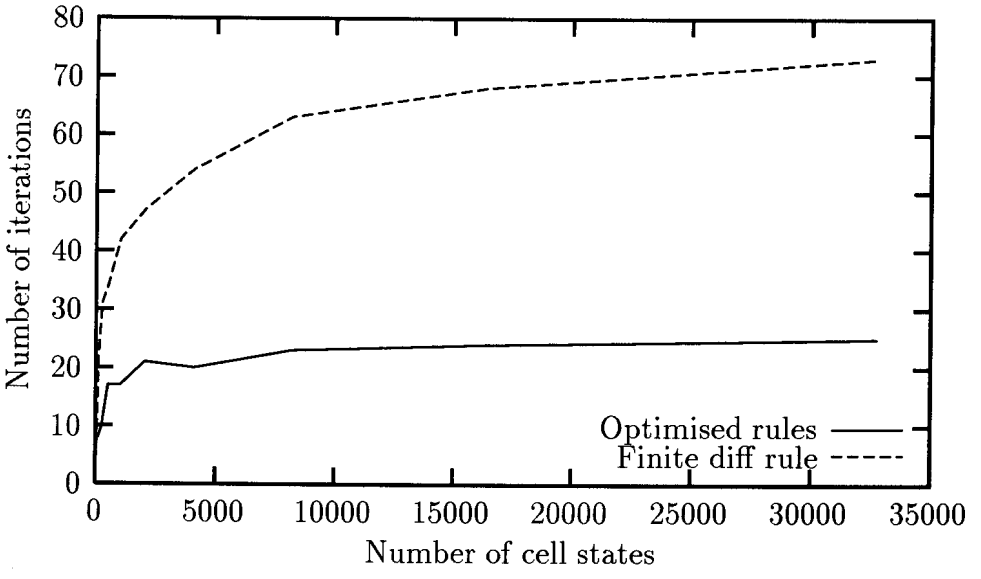

Figure 3.10: Extended neighborhood computational cost comparison

lems?

- It is clear that the extended neighborhood would involve more complications in larger mesh sizes. The number of cells used in each calculation requires many neighboring cells to be "looked up". This procedure is known to be expensive in terms of computational cost in a conventional computer, despite the differences in instruction sets used by various computer implementation.
- In the foregoing example, the number of "free" cells in the simulation is relatively low.

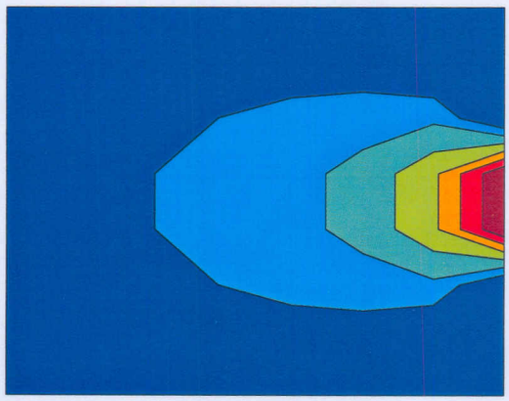

Figure 3.11: Problem 1 FEM 8×8

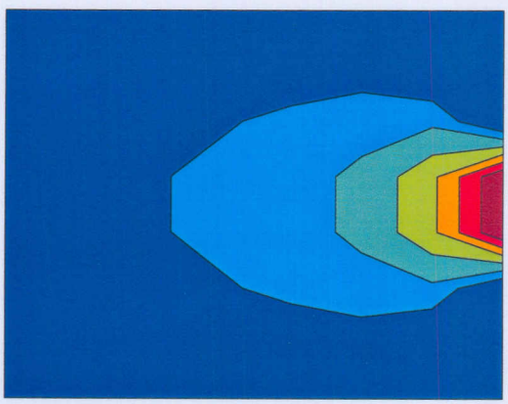

Figure 3.12: CA optimized 32 cell states

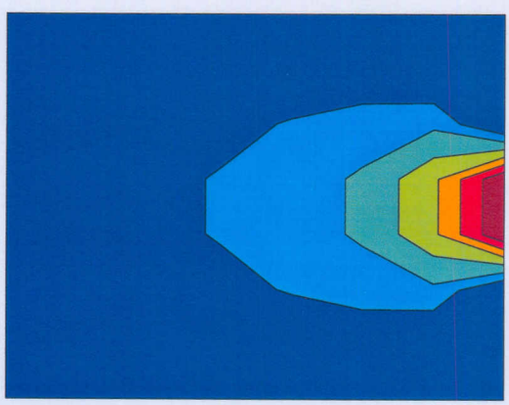

Figure 3.13: FDM 4 neighbors

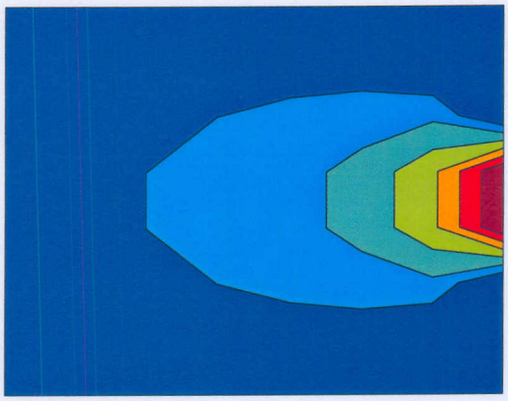

Figure 3.14: CA optimized 1024 cell states

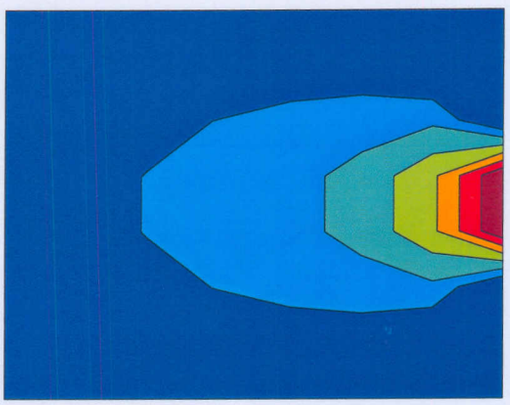

Figure 3.15: CA optimized 32768 cell states

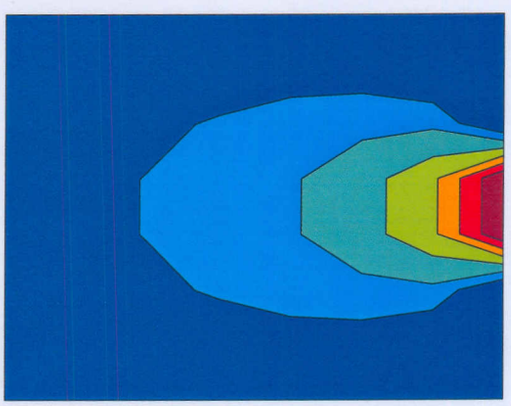

Figure 3.16: Extended FDM

3.4.2 Moore neighborhood

We shall now attempt to answer some of the questions that arose in Section 3.4.1. By reducing the neighborhood size to a Moore neighborhood, the cell rule for the cell at any point i, j can be written in terms of the variables

$$Cell_{ij}^{\text{New}} = (a(Cell_{i+1,j} + Cell_{i-1,j} + Cell_{i,j+1} + Cell_{i,j-1}) \\ +b(Cell_{i+1,j+1} + Cell_{i-1,j+1} + Cell_{i-1,j-1} + Cell_{i+1,j-1}))/c \quad (3.12)$$

These rules are also considered *outer totalistic*. The number of variables is now reduced to three. This means that the number of possible combinations is $64^3 = 262144$, which is $O(2)$ less than the extended neighborhood. Exploring all possible combinations on an AMD Athlon 800MHz requires less than 20 minutes and hence all solutions shown for this problem are the known optimum combinations for the three variables used. This time measurement is reliant on the assumption that the range of a and b may vary between -31 and 32 and variable c has a range between 1 and 64. The mesh size is also increased to $16 \times 16 = 256$ cells. Either FEM or BEM can be used for the boundary input for this problem, but FEM is chosen as it is capable of obtaining more accurate solutions. Still considering Problem 1 (Figure 3.3) the best cell rules are now extracted, and depicted in Table 3.3.

Cell states	Error RMS	Cost	a	b	c
8	0.53015190E+00	3	-1	14	35
16	0.47712490E+00	4	3	8	34
32	0.42685911E+00	10	2	7	31
64	0.34310171E+00	41	1	15	58
128	0.30804555E+00	64	0	11	42
256	0.53125077E-01	75	9	2	43
512	0.61781108E-01	58	8	8	63
1024	0.93192644E-01	54	1	15	63
2048	0.11832671E+00	60	0	16	63
4096	0.13436320E+00	69	0	16	63
8192	0.14203794E+00	75	0	16	63
16384	0.14606122E+00	86	0	16	63
32768	0.14467440E+00	106	4	1	20

Table 3.3: Optimized rules for 16×16 mesh on Moore neighborhood FEM problem 1

Table 3.3 shows the same tendencies as Table 3.2. The error decreases as the number of cell states increases and then increases again as even more cell states are used. The smallest error is obtained with 256 cell states. Again the cell rule seems to converge to a given rule for this problem. The RMS errors are greater than that obtained for the previous problem. This is to be expected, since the number of points used for the error calculation is increased

dramatically. The cost (in terms of the number of iterations required) for this problem increases dramatically when compared to the smaller mesh size. The rules obtained are totally different from those obtained in Table 3.2. This change is expected due to the change in neighborhood. Once again, most of the obtained CA rules are superior to the FDM rule.

The fact that fewer cell states result in more accurate solutions is cause for investigation. The believed reason for this tendency is simplified and explained in Figure 3.17.

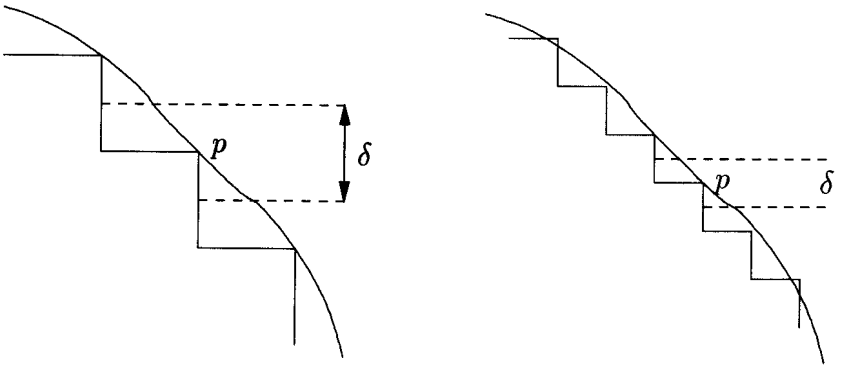

Figure 3.17: Cell state accuracy round off occurrence

As the solution is only being approximated at point p, the course grid has a bigger area which is seen as the exact value at that point. The smaller grid has a smaller area which is translated into the exact point. With a smaller number of cell states, the number of points which yields a solution close to the real values is greater in comparison to the total number of points.

The stress contours for selected results are plotted in Figures 3.18 to 3.23.

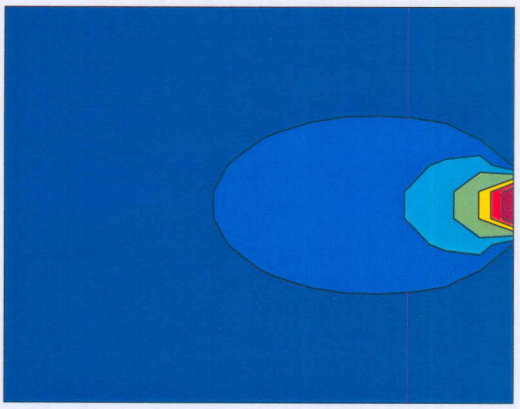

Figure 3.18: Problem 1 FEM 16×16

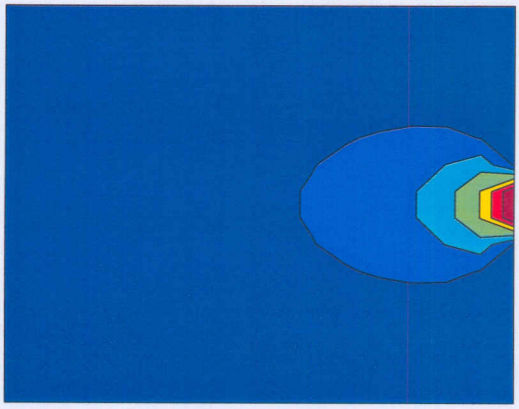

Figure 3.19: FDM 4 neighbors

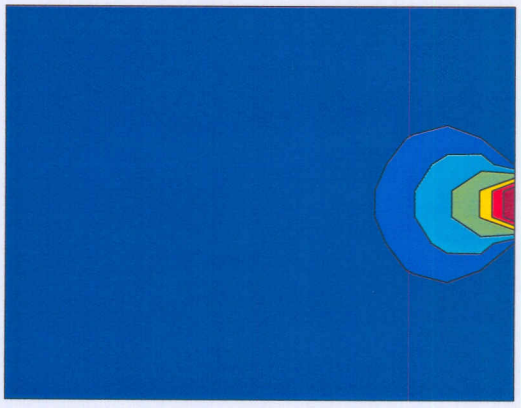

Figure 3.20: CA optimized 32 cell states

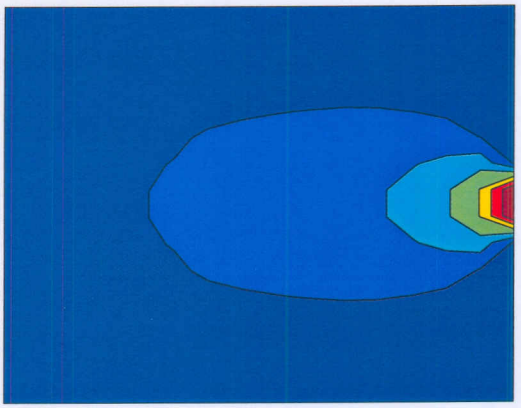

Figure 3.21: CA optimized 256 cell states

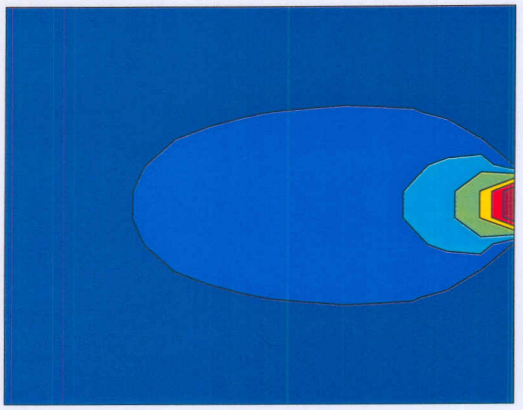

Figure 3.22: CA optimized 2048 cell states

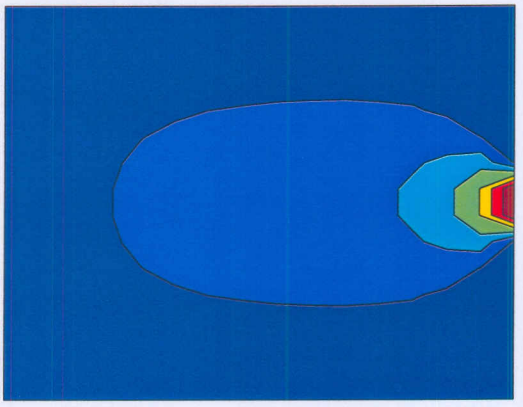

Figure 3.23: CA optimized 32768 cell states

Problem 2

To gain a better understanding of how a CA model approximates structural problems, the required rules for different stress fields are now investigated. The optimum rules for a Moore neighborhood are determined for the second problem investigated (see Figure 3.4). The cell rules shown in Table 3.4 are obtained using BEM.

Cell states	Error RMS	Cost	a	b	c
8	0.60469618E+00	2	6	1	22
16	0.55514977E+00	3	3	4	24
32	0.54489576E+00	5	14	1	57
64	0.50717726E+00	10	13	3	62
128	0.47369715E+00	12	4	1	20
256	0.38586495E+00	29	1	0	4
512	0.37508172E+00	44	8	-1	28
1024	0.36018374E+00	43	14	1	61
2048	0.35617211E+00	47	15	0	61
4096	0.35266954E+00	54	15	0	61
8192	0.35122104E+00	60	15	0	61
16384	0.35072195E+00	70	15	0	61
32768	0.35035057E+00	77	15	0	61

Table 3.4: Optimized rules for 16×16 mesh on Moore neighborhood BEM problem 2

In this problem the optimum cell rule is obtained using the greatest number of cell states. An interesting fact is that the rule obtained is indeed the 4 neighbor FDM rule, since $\frac{15}{61} = 4.066$. The rules obtained with this problem are clearly different from those obtained in our first problem, which indicates that the change in stress field influences the best rule for the problem. The stress contours in this problem are shown in Figures 3.24 to 3.29.

It is clear that the stress contours from the resultant optimum rules do not yield the correct stress contours. The BEM solution shows a band which spreads the stress from the point where the force is applied to the corner points, resulting in a core in the center where no stress occurs, as shown in Figure 3.24. The CA rules are not capable of reproducing this phenomenon. Results from the refinement of the meshes for FEM and BEM are shown in Figures 3.30 to 3.33. By using a non-linear FEM package, the problem was found to be geometrically non-linear. The CA simulation is, however, capable of using these values and obtaining reasonably good results.

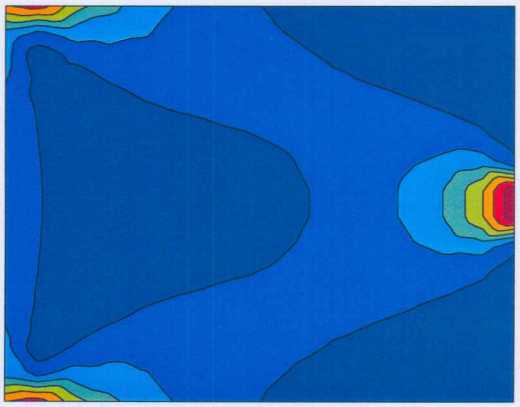

Figure 3.24: Problem 2 BEM 16 × 16

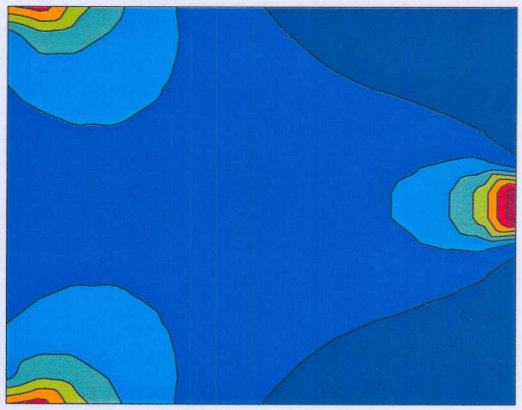

Figure 3.25: FDM 4 neighbor

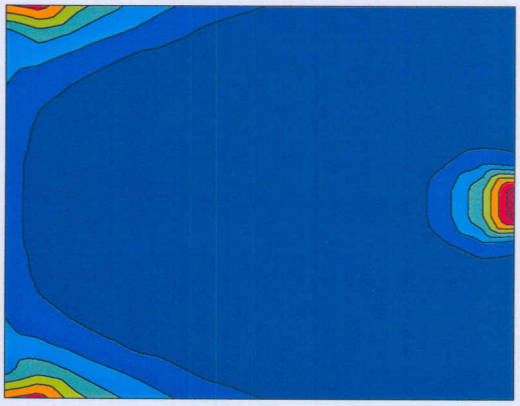

Figure 3.26: CA optimized 32 cell states

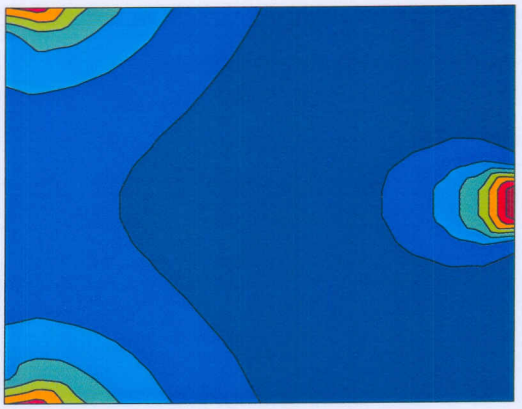

Figure 3.27: CA optimized 256 cell states

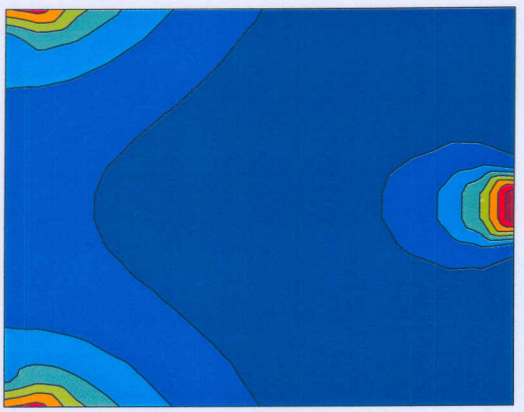

Figure 3.28: CA optimized 2048 cell states

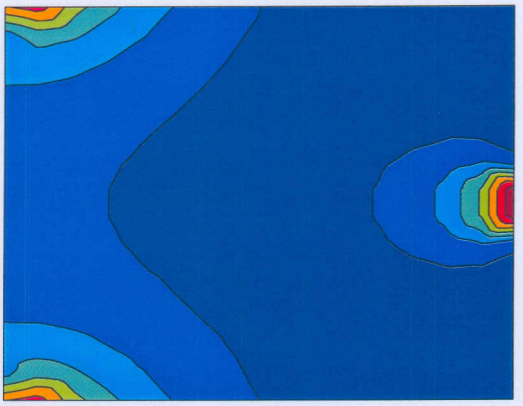

Figure 3.29: CA optimized 32768 cell states

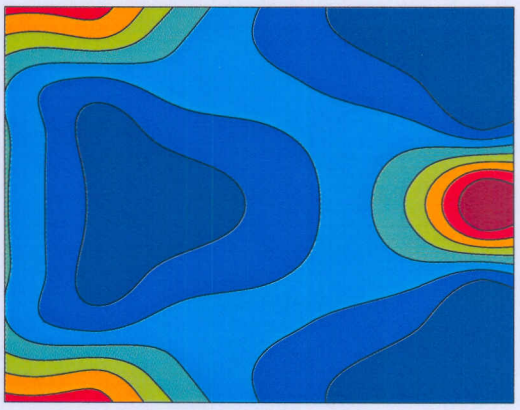

Figure 3.30: Problem 2 BEM 8×8

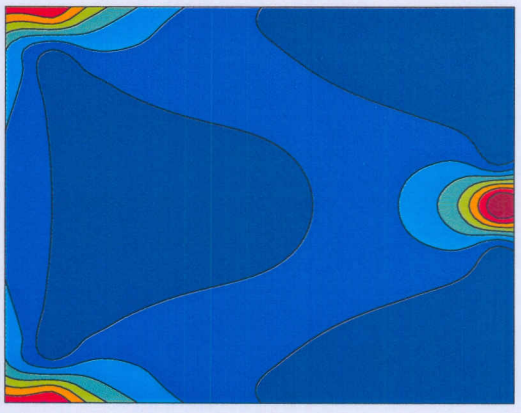

Figure 3.31: Problem 2 BEM 16×16

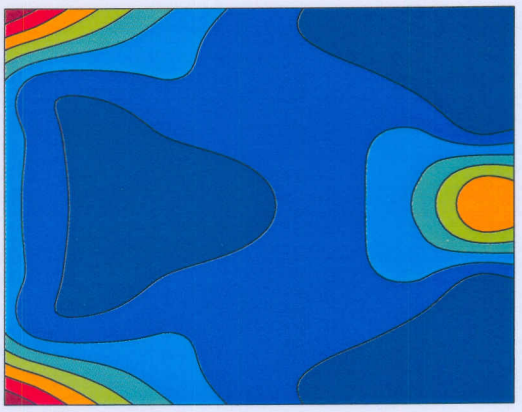

Figure 3.32: Problem 2 FEM 8×8

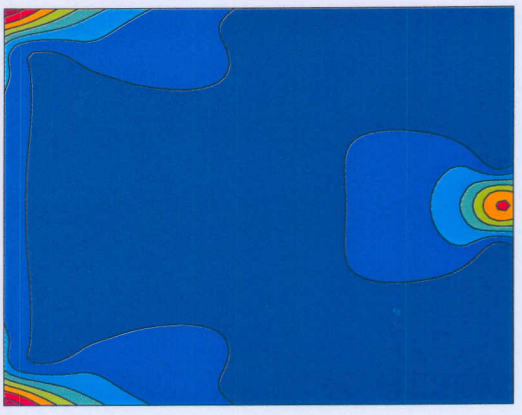

Figure 3.33: Problem 2 FEM 16×16

Bending problem

We now consider Problem 3 (Figure 3.5). It is possible to obtain an analytical solution to this problem [42]. However, the FEM and BEM codes are still used in this section. As opposed to Problem 2, this problem is linear, which allows for better comparison between different mesh sizes. Searching for the best cell values, we obtain the results given in Table 3.5.

Cell states	Error RMS	Cost	a	b	c
8	0.25105715E+00	11	1	3	14
16	0.11395269E+00	15	5	2	26
32	0.84983295E-01	17	11	2	50
64	0.78008803E-01	24	14	-3	43
128	0.84722215E-01	19	13	2	59
256	0.87620568E-01	21	12	1	52
512	0.77930804E-01	26	7	1	32
1024	0.75518361E-01	29	5	1	24
2048	0.74438666E-01	31	8	1	36
4096	0.74259859E-01	34	7	1	32
8192	0.74165211E-01	35	5	1	24
16384	0.74118742E-01	42	6	1	28
32768	0.74098354E-01	42	5	1	24

Table 3.5: Optimized rules for 8×8 mesh on Moore neighborhood FEM problem 3

For this problem, the error decreases as the number of cell states decreases but the decrease is not dramatical. At 64 cell states, the problem is already very close to the minimum error. The error then increases slightly due to the rounding as described in section 3.4.2 and then decreases slowly.

The cell rules obtained appear to approximate the FDM 4 neighbor rule. The ratio is always very close to four. The rules obtained do, however, use some of the information from the corner cells. Selected plots for the rules obtained are shown in Figures 3.34 to 3.39. The best stress contours are obtained with 64 cell states.

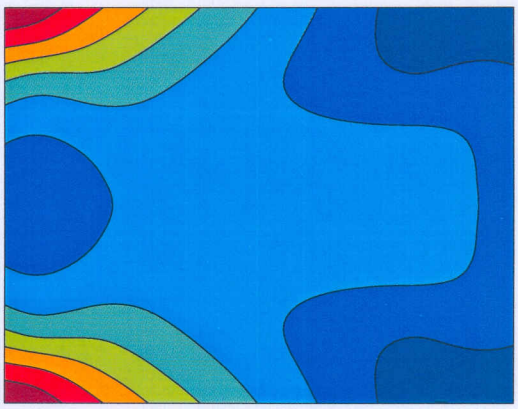

Figure 3.34: Problem 3 FEM 8×8

Figure 3.35: CA optimized 32 cell states

Figure 3.36: CA optimized 64 cell states

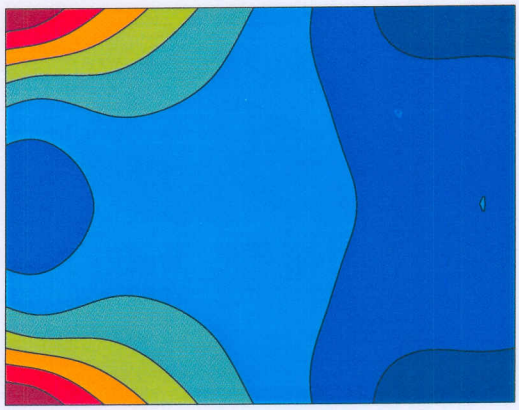

Figure 3.37: CA optimized 512 cell states

Figure 3.38: CA optimized 1024 cell states

Figure 3.39: CA optimized 32768 cell states

For continuity, we now increase the mesh size to 16 × 16 for the same bending problem (Figure 3.5). The results are shown in Table 3.6.

Cell States	Error RMS	Cost	a	b	c
8	0.97218127E+00	12	21	-5	52
16	0.90144640E+00	17	13	-4	32
32	0.83100041E+00	30	23	-7	60
64	0.75792672E+00	33	23	-8	58
128	0.58255949E+00	76	25	-9	63
256	0.61141889E+00	64	25	-9	64
512	0.31104888E+00	138	25	-9	64
1024	0.17773177E+00	166	11	-4	28
2048	0.12366561E+00	235	11	-4	28
4096	0.10983362E+00	260	11	-4	28
8192	0.10741246E+00	296	11	-4	28
16384	0.10732287E+00	323	19	-7	48
32768	0.10771372E+00	354	11	-4	28

Table 3.6: Optimized rules for 16 × 16 mesh on Moore neighborhood FEM Problem 3

Once again we find that the solution converges to one cell rule. This time the converged cell rule is not the same as the FDM rule and the error terms decrease as the number of cell states increases. The rules are completely different from those obtained in Table 3.5 although the stress fields are comparable. The stress contours for the selected rules are plotted in Figures 3.40 to 3.45. Note that, although the rules obtained are reasonable, none of them are capable of closely resembling the correct result.

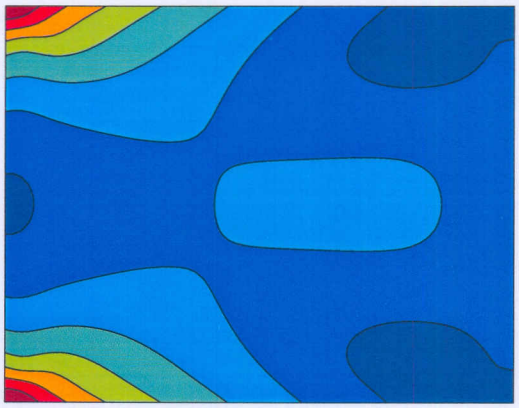

Figure 3.40: FEM 16 × 16 stress plot

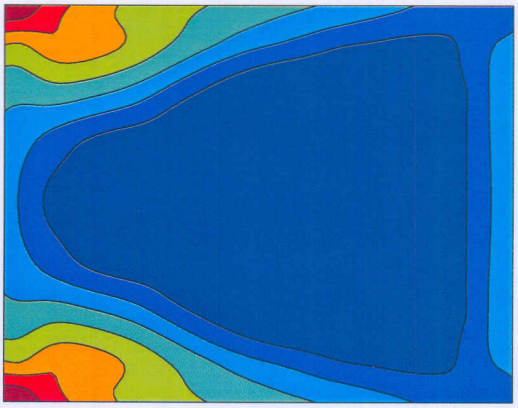

Figure 3.41: CA optimized 32 cell states

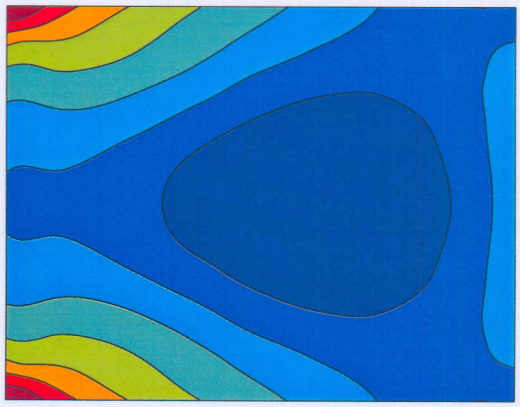

Figure 3.42: CA optimized 128 cell states

Figure 3.43: CA optimized 512 cell states

Figure 3.44: CA optimized 2048 cell states

Figure 3.45: CA optimized 32768 cell states

We shall now investigate the influence of increasing the number of variables that influence each cell. Doubling the number of cell variables per cell implies that the range of variables a and b may vary between -63 and 64 and variable c has a range between 1 and 128. This increases the number of combinations. The rules obtained can be seen in Table 3.7.

Cell states	Error RMS	Cost	a	b	c
8	0.97218127E+00	12	21	-5	52
16	0.88905889E+00	18	27	-7	71
32	0.83100041E+00	30	23	-7	60
64	0.69464500E+00	46	44	-15	112
128	0.52526133E+00	85	47	-17	118
256	0.54318169E+00	118	53	-23	119
512	0.30518049E+00	132	48	-17	124
1024	0.17581473E+00	173	36	-13	92
2048	0.12366561E+00	235	11	-4	28
4096	0.10983362E+00	260	11	-4	28
8192	0.10730810E+00	285	47	-17	120
16384	0.10724750E+00	322	47	-17	120
32768	0.10768116E+00	358	35	-13	88

Table 3.7: Optimized rules for 16×16 mesh for more cell states on a Moore neighborhood FEM Problem 3

We find that for four of the rules obtained (8, 32, 2048 and 4096 cell states) the increase in the number of cell variables had no influence in the rule obtained. For the other cell rules obtained, the reduction in the RMS error does not seem to be significant. The stress plots for selected rules can be seen in Figures 3.46 to 3.51.

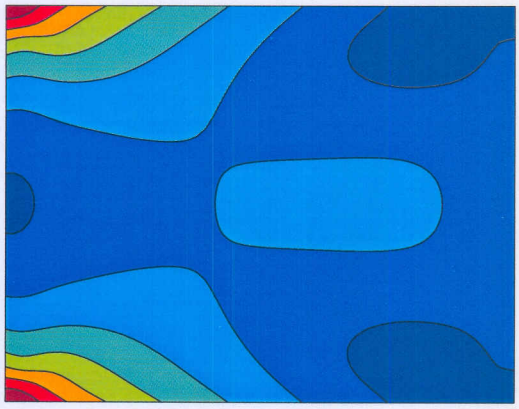

Figure 3.46: FEM 16 × 16 stress plot

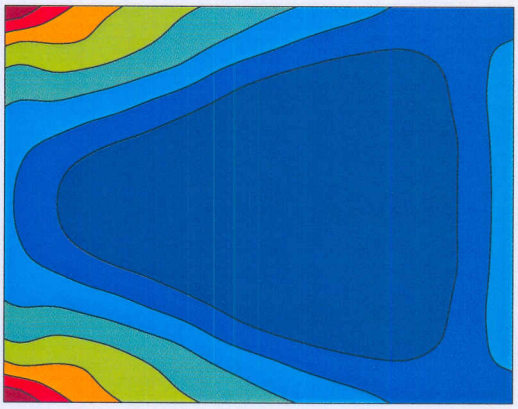

Figure 3.47: CA optimized 32 cell states

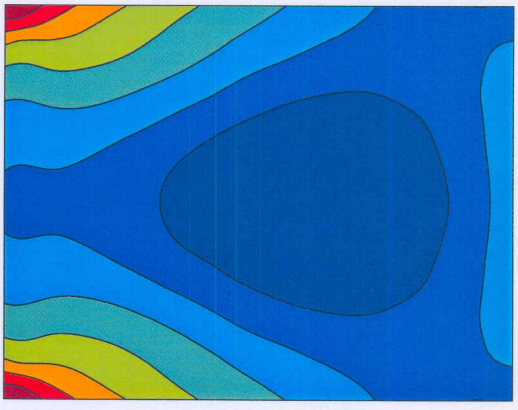

Figure 3.48: CA optimized 128 cell states

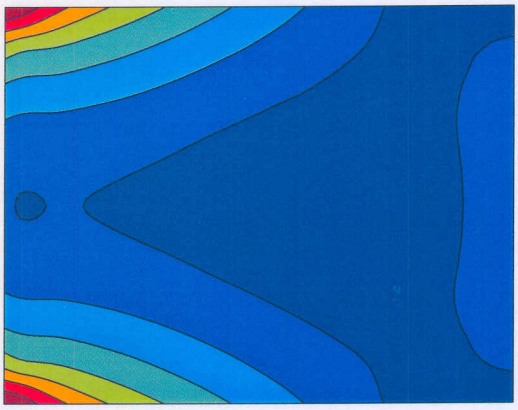

Figure 3.49: CA optimized 512 cell states

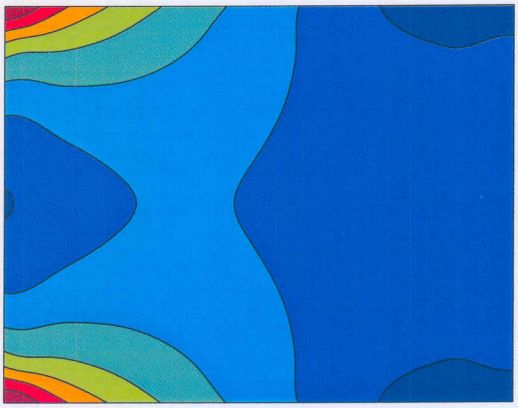

Figure 3.50: CA optimized 2048 cell states

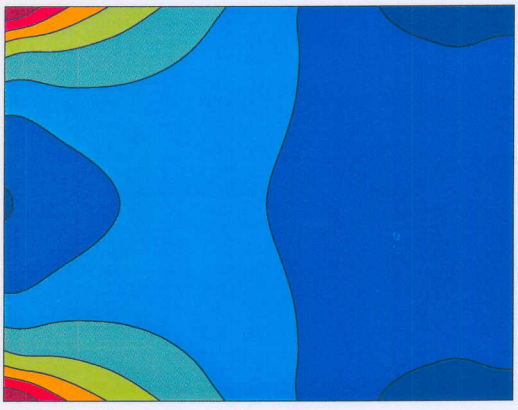

Figure 3.51: CA optimized 32768 cell states

Asymmetric rule

We shall now attempt to extract the optimum cell rule using an asymmetric rule, where symmetry is not enforced, even though the problem studied is symmetric. Instead, we use asymmetry as a qualitative measure of the fitness of the rule derived. Owing to the large number of possibilities, the cell rule extraction is attempted with only 1024 cell states with the FEM 8×8 model as learning basis. As a result the optimization problem becomes enormous and exploring all possibilities with all nine variables impossible. The GA was restarted a number of times and the best cell rules obtained were kept. The values of each cell position are shown in Figure 3.52. The best cell rules obtained for this problem are shown in Table 3.8.

f	b	h
c	p	d
e	a	g

Figure 3.52: Asymmetric neighborhood problem description

Rule number	RMS error	a	b	c	d	e	f	g	h	p
1	0.558182E-01	18	16	9	-1	-1	5	3	11	59
2	0.532882E-01	19	0	3	-14	-1	16	4	31	56
3	0.528102E-01	16	17	1	-14	0	13	7	21	59
4	0.474360E-01	30	-3	-18	5	4	32	-6	19	61
5	0.492669E-01	28	13	-1	18	2	24	-13	4	59
6	0.515053E-01	32	7	-1	20	-9	23	-17	11	63

Table 3.8: Optimized rules 8×8 FEM problem 3 (asymmetric)

The rules deliver errors that are far lower than the best solution obtained for the symmetric cell rule. It is, however, very difficult to comprehend if they deliver any significant pattern in terms of how the rule develops. For this reason, the rule development is depicted in Figure 3.53.

The rules clearly follow no particular pattern. The RMS errors obtained are much lower than those obtained with the symmetric rule. The stress fields obtained are plotted in Figures 3.54 to 3.59. Even though the rules obtained are asymmetrical, the stress fields obtained are remarkably symmetrical. The symmetry can be used as a qualitative measure of the fitness of the rules, and the rules are superior to the rules found in Section 3.4.2.

5	16	11
9	59	-1
-1	18	3

Rule 1

16	0	31
3	56	-14
-1	19	4

Rule 2

13	17	21
1	59	-14
0	16	7

Rule 3

32	-3	19
-18	61	5
4	30	-6

Rule 4

24	13	4
-15	59	18
2	28	-13

Rule 5

23	7	11
-1	63	20
-9	32	-17

Rule 6

Figure 3.53: Asymmetric rule orientation for Problem 3 FEM 8×8 1024 cell states

Figure 3.54: Asymmetric rule one

Figure 3.55: Asymmetric rule two

Figure 3.56: Asymmetric rule three

Figure 3.57: Asymmetric rule four

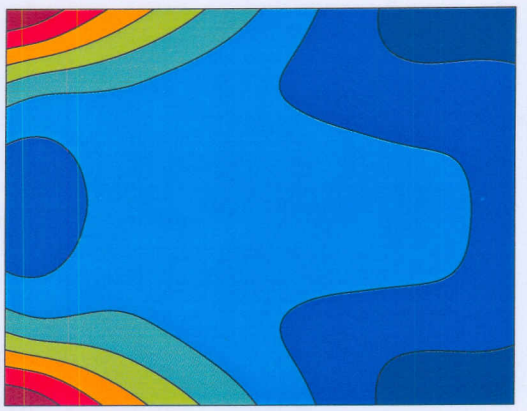

Figure 3.58: Asymmetric rule five

Figure 3.59: Asymmetric rule six

3.5 Cellular automata computational conclusions

The results obtained from using the CA simulations may be approached from different points of view. Each of the problems under investigation were capable of delivering a cell rule which is capable of approximating the state in which the structure found itself with relatively few cell states. Each of the rules obtained were more accurate than the corresponding finite difference equations. The use of a discrete number of cell states made the convergence criteria easy and allowed for a faster solution than continuous variables. By directly using the von Mises stress as the criteria to describe the state of the structure, we are able to develop a fast approximation ability which immediately provides enough information about the integrity of the structure. The ability to obtain the correct rule is of vital significance to the applicability of CA describing this natural system.

Although in each of the examples we were able to extract a cell rule in accordance with the neighborhood used that was able to surpass the performance of the FDM rules, not one of the rules were the same. Transferring the rules obtained from one problem to another yields very poor results. In fact, the rules obtained for the same problem where only the mesh was refined were not compatible at all. It is, however, clear that every problem has a cell rule which best described the stress field in that problem. This rule was usually obtained with a high number of cell states.

The lack of capability to obtain a universal cell rule indicates that the total formulation of the problem is not capable of capturing the governing aspects of the problems at hand. This failure in the CA simulation is not a failure of the CA itself but rather a failure in the formulation and approach for capturing the essence of the problem. With vast amounts of possible approaches, judging the CA capability using only one method would be unjustified. The results do indeed suggest that with more insight a cell rule construction might indeed be possible from the governing equation even if a complex criteria like the von Mises stress is used directly.

When CA rules and neighborhoods have developed far enough to allow for all classes of problems to be analyzed, the serious problem of obtaining the correct boundary values for CA simulations will require urgent attention. The current methods discussed in this thesis are more well known as numerical approximations. A number of other methods exist which can be used for this approximation. For instance, Victor Apanovitch [43] uses an external approximation which requires no mesh to obtain the solution. The approach of such methods lies with the mathematical formulation and assumptions made with such formulations. Underlying assumptions are already present in what are considered the governing equations in Appendix A. By using different assumptions in the understanding of the physics of the problem it might even be possible to formulate a mathematical model directed towards CA simulation.