Struct Multidisc Optim 23, 24–33 © Springer-Verlag 2001

# On the use of energy minimization for CA based analysis in elasticity

**P. Hajela and B. Kim**

**Abstract** There has been recent interest in exploring alternative computational models for structural analysis that are better suited for a design environment requiring repetitive analysis. The need for such models is brought about by significant increases in computer processing speeds, realized primarily through parallel processing. To take full advantage of such parallel machines, however, the computational approach itself must be revisited from a totally different perspective; parallelization of inherently serial paradigms is subject to limitations introduced by a requirement of information coordination. The cellular automata (CA) model of decentralized computations provides one such approach which is ideally tailored for parallel computers. The proposed paper examines the applicability of the cellular automata model in problems of 2-D elasticity. The focus of the paper is in the use of a genetic algorithm based optimization process to derive the rules for local interaction required in evolving the cellular automata.

**Key words** cellular automata, structural analysis, evolutionary methods

## 1 Introduction

Improved efficiency of numerical simulations is a core issue in the development of the next generation of computer-based design systems. Not only will such systems be used for design of very complex artifacts, but they will have to be tailored for a more effective interface with the human designer. There is some agreement that for the human designer to be effective in their role, answers to "what- if" questions typically posed during a design cycle must be available in an expeditious manner. Towards this end, there has been considerable activity in the MDO community in developing methods for function approximations (Guinta 1997) which provide surrogate models for use in design in lieu of more expensive "exact" calculations. In many instances, however, these surrogate models themselves require substantial numerical data to establish, and such data comes from either physical or numerical experiments. As design optimization moves into addressing practical scale problems, the computational costs associated with numerical experiments has gone up sharply. This bottleneck has continued to develop despite a rapid increase in computer processing speeds over the past decade. While such speeds continue to double every 18 months, there is a growing realization that the silicon technology based processor is rapidly approaching physical constraints that would preclude further dramatic increases in processing speeds. The obvious answer lies in using massive arrays of processors, acting in parallel, to perform lengthy computations.

The use of parallel processing, however, is not without its own challenges. Two distinct paradigms have received considerable attention in the literature. The first deals with optimization of existing code to make it more amenable to improving processing speeds on a parallel computer. This strategy was extensively pursued in computational fluid dynamics, and to a lesser extent in computational solid mechanics. A second approach is based on decomposition of the analysis domain into subdomains, with analysis of different subdomains being performed on different processors. The substructuring concept in finite element analysis (Zienkiewicz and Taylor 1989) is an example of such an approach, and has been reported with varying degrees of success. Almost all applications of the approach, however, report the common problem of a decreasing rate of increase in processing speeds with additional processors. With an increase in processors, the overhead associated with coordination of

*Received August 28, 2000*

P. Hajela and B. Kim

Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
e-mail: hajela@rpi.edu

information among the decomposed subdomains overwhelms any increase in processing speed. This is common to all analysis that is rooted in serial thinking. There is very clearly the need for developing new algorithms, drawn from and developed in a parallel processing framework. The cellular automata approach is an example of such a concept, and has been studied here in the context of problems in 2-D elasticity.

Cellular automata are fully discrete models of a physical system. They emulate systems in nature which exhibit sophisticated collective information-processing through local rules of interaction with their nearest neighbours. In this regard, they offer an approach wherein distributed processors interact via restricted communication pathways to perform collective information-processing. Since information processing in such systems is intrinsically parallel, they are naturally amenable to implementation on massively parallel computers with minimal loss in efficiency for coordination requirements. Examples of such systems include the efficient foraging and construction activities in ant colonies (Pasteels and Deneubourg 1987), production of multicellular organisms from single amoeba cells (Devreotes 1989), and the PDP model of the biological brain (Rumelhart *et al.* 1986). This model of computation has been adapted in a number of combinatorial mathematics and computer science applications, and in computation of fluid flow properties (Hardy *et al.* 1976; Orszag and Yakhot 1986). In the context of the latter, a form of cellular automata referred to as the lattice gas model has received considerable attention. A lattice gas structure has been shown to be a realistic model of the equations of hydrodynamics, and to provide a simple framework for the development of a kinetic theory (Rothman and Zaleski 1997).

The rules for evolving such cellular automata, however, are generally not known a priori. They are generally established on the basis of physics governing a particular problem, and the model is sensitive to this choice. The rules can involve well known concepts such as continuity in flow, force equilibrium, or momentum conservation, or may be specific equations relating values of state variables at various points in the discrete system to those at neighourhood sites. When using the latter approach, there could be problems where such relations may not be readily apparent, and here the user would be expected to experiment with different laws of interaction to identify one that is useful in a particular problem. This form of experimentation would also be required in those cases where the physics of the problem is not properly understood. The present paper describes the cellular automata approach in the analysis of 2-D elastic structures. The focus is on describing how physical principles or known problem heuristics can be effectively combined to develop rules of cellular automata evolution that are pertinent to a given problem. Central to this approach is the rule discovery or learning process, and for which a genetic algorithm (GA) based search process is used. Subsequent sections of this paper describe the cellular automata approach and its adaptations in elasticity in greater detail.

## 2
## CA evolution and rules of interaction

One can think of cellular automata as a stylized universe where space is represented by a uniform grid. The cells formed by such a grid contain a few bits of data, and collectively, this data describes the state of the system. The state of all cells are then advanced in discrete time steps following some "laws of the universe" which are generally in the form of a simple recipe through which each cells updates its state from that of its closest neighbours. To better understand how cellular automata function, consider a one-dimensional grid of cells to define a binary state system as shown in Fig. 1, where each cell can take on a value of 0 or 1.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t=0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| t=1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| t=2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

**Fig. 1** One-dimensional CA model

Assume further that the neighbourhood of any cell is defined as itself and the two adjacent cells. A simple local rule that can be used to evolve this binary state system from one discrete time-step to the next is that each cell state assumes a value that is a "majority vote" of the 3 cells that define its neighbourhood; boundary cells are flipped from a 0 to 1 or vice versa. This updating procedure is shown in Fig. 2. Application of this rule to the 1-D cellular automata for two time steps is shown in Fig. 1. Similar rules can be developed for cellular automata models in higher dimensions.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 | - | neighborhood |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | - | output |

**Fig. 2** Majority vote rule

Such a model is naturally amenable to implementation on massively parallel machines – the temporal updating of the state of different cell groups can be assigned to different processors. Since all such operations are performed simultaneously, proper load distribution among processors is not an issue of serious concern. Furthermore, minimal amount of information must be exchanged among processors at the end of each time step. The key to the success of this parallel distributed approach is that the proper rule for updating the states be available. A rule different from the "majority vote rule" of Fig. 2 would produce a different cellular automata evolution pattern. The pertinent question then is whether appropriate rules of interaction can be derived for a given problem of interest. The present work explores the use of soft computing based optimization techniques in this problem.

To illustrate this concept, we revert to the one-dimensional binary state cellular automata described above. Assume that while the converged state of the system is known, the "majority vote" rule for evolution is not known, and a derivation of this rule is the object of the optimization process. Since the rule is a string of binary numbers, a genetic algorithm based search process is most appropriate for this optimization. A stepwise process can be outlined as follows.

Step 1. A population of 8-digit binary bit strings (rules) is generated at random. Each bit of this string corresponds to the output of one of the eight unique combinations of a 3-digit binary bit string representing the 3 cell neighbourhoods.

Step 2. Each rule in this population of rules would be applied to the same initial state (one at a time), and the state evolved to some converged final configuration. The similarity between the converged and known final state of the system would be the fitness assigned to the selected rule. This process is repeated for each rule in the population.

Step 3. Using the fitness of rules in Step 2, the traditional selection, crossover, and mutation operations of a GA are performed to maximize the rule fitness.

Step 4. The process is repeated from Step 2 to convergence of the GA evolution process.

The converged rule obtained from this simulation should be the "majority vote rule" which was responsible for generating the desired state. This rule learning process can be extended to a problem with higher dimensions. The principal issue in this case would be a new definition of the neighbourhood of a particular cell site. In 2-D problems, examples of a neighbourhood definition are shown in Fig. 3. If the majority vote rule were to be implemented for the N-S-E-W definition of the neighbourhood, the rule length would change from 8 bits for the 1-D CA model to 32 bits, given that there would be 32 uniquecombinations of the 5 cell states (the site and its four neighbours) that would be involved in the voting process.

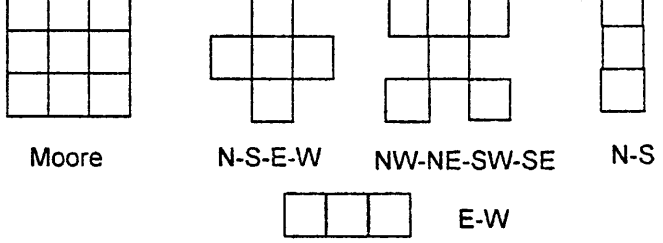

**Fig. 3** Set of possible neighbourhoods of interaction

It is clear from the foregoing example that the two most important factors in evolving a cellular automata are the definition of a cell state and of its neighbourhood of interaction. In problems of structural analysis, for example, the definition of the cell states may involve quantities such as the local stiffness, strain or stress in place of a simple binary state discussed in the previous example. The neighbourhood would also be subject to selection. While such neighbourhoods can extend beyond the immediate neighbours, the motivation for parallel implementation behind the cellular automata approach requires that they be kept as compact as possible. The rule for evolving such states could be based on known results, as in the example described above. Hajela and Kim (1999) describe a process for structural analysis problems, where known numerical solutions from finite elements are used to derive rules for interactions. When applied to a discrete representation of the structure, these rules produce results similar to those that would be obtained from a finite element solution. It was possible to add some generalization capability to the process. In the paper by Hajela and Kim (1999), this would developed by identifying rule sets for many different load cases, and by using a neural network to build the functional relationship between the load set and the corresponding rule of interaction. The motivation in this case was that if a general set of rules for evolving cellular automata for a given structural system could be derived, then such a model would be extremely efficient for structural reanalysis required during design optimization, particularly when implemented on a massively parallel computing architecture.

The approach used in the present work does not require a known solution to derive the rules of interaction for the cellular automata problem. Instead, it is based on using well-known principles from solid mechanics, more specifically, the principle of minimum strain energy to derive these rules. In this context, two different plane strain problems have been studied, the analytical solutions for which are known. Both displacement and strain components are selected as possible cell states in the evolution process. The neighbourhood of interactions are actually selected as part of the optimization process. Both binary coded and real-valued GA's are used in optimization. The numerical results clearly indicate the need to determine the globally optimal solution, as sub-optimal rules tend to generate relatively poor solutions.

## 3 Energy minimization and CA evolution

For an elastic domain $D$ subjected to given static load field $F$, the equilibrium strain state would be one that yields a minimum strain energy. For a 2-D problem in elasticity, the expression for strain energy can be written as follows:

$$U = \frac{E}{2(1+v)} \int\int\int_{\text{vol}} \left[ \frac{1}{1-v} \left( \varepsilon_x^2 + \varepsilon_y^2 + 2v\varepsilon_x\varepsilon_y \right) + \frac{1}{2}\gamma_{xy}^2 \right] \mathrm{d}x\, g4\mathrm{d}y\, \mathrm{d}z \qquad (1)$$

where $\varepsilon_x$, $\varepsilon_y$ are the normal strains, and $\gamma_{xy}$ is the shear strain component. This is a quadratic function in terms of the strain components and will, therefore, have a unique solution that defines the minimum of $U$. These strain components are natural candidates for defining the state of the cells in a cellular automata model of a structure. An even more fundamental definition of the cell state would be in terms of displacement components at each cell site, and from which strain components can be derived through numerical gradients. In either case, we see that it is possible to have multiple state variables at any given cell site in the domain. While it may be possible to evolve a fundamental set of state variables through one rule, the evolution of states such as stresses or strains generally require as many rules as there are state variables.

When working with arbitrary state variables, it is sometimes difficult to determine the precise format of a rule of interaction. In the absence of physical guidelines for this selection, one can propose an arbitrary algebraic rule with some unknown parameters to be determined during the optimization process. A linear or nonlinear weighted averaging scheme is one such option, and can be stated as follows:

$$\mathrm{CS}(i) = \sum_{j \in J} w_j \mathrm{CS}(j) \quad \text{linear}\,, \qquad (2)$$

$$\mathrm{CS}(i) = \sum_{j \in J} w_j [\mathrm{CS}(j)]^{\eta_j} \quad \text{nonlinear}\,. \qquad (3)$$

In the above, $\mathrm{CS}(j)$ is the state of the $j$-th cell, $J$ defines the cells in the neighbourhood, and $w_j$ and $h_j$ are the unknown weights and coefficients to be determined in the optimization process. Note that nonlinear averaging indeed introduces additional degrees of freedom $h_j$ to be determined in the optimization process. For the example problems described in this paper, a linear weighted average was found to be satisfactory for both applications.

In addition to determining unknown parameters in some assumed model of interaction, it is also possible to include a search for the optimal neighbourhood of interaction during the optimization process. Figure 3 shows a limited set of possible neighbourhoods of interaction; each of these is symmetric and involves no more than cells from the immediate neighbourhood (adjacent cell sites only). If each of these neighbourhoods is defined as $N^k \in N$, then the rule discovery problem may be stated in the following mathematical form:

$$\text{minimize} \quad E\left(w_i^k, N^k\right)\,,$$

$$\left(w_i^k\right) L \le w_i^k \le \left(w_i^k\right) U\,, \quad N^k \in N\,,$$

$$g_j \le 0\,, \quad j \in M\,. \qquad (4)$$

Here a linear averaging process of (1) is assumed; the weights are bounded by lower and upper limits to $(w_i^k)L$ and $(w_i^k)U$, respectively; $E$ is the strain energy which is to be minimized, and $g_j$ are other general constraints that may be required in the problem. Examples of such constraints could include a requirement that the minimal strain energy be equal to the external work, or, a physical constraint to ensure single-valued displacements at a given point in the structural domain. In this search process, the neighbourhood selection parameter is discrete and the number of weights associated with each set is also different. A GA based search is naturally amenable to this class of optimization problems, and both binary coded and real valued GA's were used in the problem solution. A stepwise description of the optimization problem is as follows.

Step 1. Initialize a population of possible rules at random, where each rule represents a particular neighbourhood of interaction and numerical values of the weights associated with that particular neighbourhood. Prescribe an initial value of the state variable in each cell and the boundary conditions which the converged solution must satisfy.

Step 2. Apply each rule in the population to evolve the initial state to a converged solution. Compute the strain energy corresponding to each converged solution and the constraint values associated with that solution (if additional constraints are imposed).

Step 3. Using minimum strain energy and constraint satisfaction to define a fitness function, compute fitness of each rule.

Step 4 Evolve the population of rules using the GA over one generation of evolution.

Step 5. Repeat from Step 2 to convergence.

The converged rule from this GA evolution, when applied to the initial state would then yield the required state variable distribution in the structural domain.

## 4 Binary and real-coded GA's

Traditional binary-coded genetic algorithms have been extensively studied and proven to be robust and powerful when used in the solution of optimization problems involving nonconvex or disjointed design spaces (Hajela 1990), and with a mix of continuous, integer, and discrete design variables. Evolution strategies (ES) have also become the focus of research over the last decade and proven to be efficient and effective in seeking a global optimum in problems where multiple relative optima exist (Cai and Thierauf 1997). The major difference between genetic algorithms and evolution strategies resides in the design variable coding and in population size. While, in genetic algorithms, a large number of co-existing designs comprise a population that is evolved over many generations via gene exchange mechanisms and reproduction strategies, evolution strategies are constrained to the use of a few designs. The operation of gene exchanges in genetic algorithms is mainly performed on the coded design space (binary is most common) rather than the real design space. Evolution strategies, on the other hand, perform design variations directly on design variables.

One shortcoming of the binary coded algorithm is the increased numerical cost and loss in efficiency in problems where the continuous variables need to be located with a high degree of precision. Evolution strategies circumvent this problem by operating on the variables directly; however, since they work with fewer design alternatives, they do not concurrently process information from all over the design domain (as in the genetic algorithm), and are therefore less likely to locate the global optimum. This brings about the need for combining the strengths of genetic algorithms and evolution strategies by transforming the traditional binary-coded genetic algorithms into real-coded genetic algorithms (Wright 1991). A variant of this approach was implemented in the rule discovery process in the present work, and was found to be extremely useful in those cases where the weights defining the neighbourhood interaction had to be determined with increased precision. More specifically, the traditional crossover and mutation operators of binary coded GA's were replaced by the following operations applicable to using real-valued representations of the designs.

A linear weighted crossover operation was used in which combinations of two parents $X_1 = \{x_1^1, x_1^2, \dots, x_1^n\}$ and $X_2 = \{x_2^1, x_2^2, \dots, x_2^n\}$ were selected at random from the population and combined to yield three progenies $h_k^i$, $i = 1, n$ and $k = 1, 3$ as follows:

$$h_1^i = 0.5x_1^i + 0.5x_2^i\,, \quad h_k^2 = 0.5x_1^i - 0.5x_2^i\,,$$
$$h_3^i = -0.5x_1^i + 1.5x_2^i\,. \tag{5}$$

Two of these offspring with better fitness values replace the parents in the next generation. A simplistic mutation operator was also implemented for real variables wherein a new design variable was perturbed from its original value by a fraction of the standard deviation for that particular variable. As in binary coded GA's, the crossover and mutation were applied with a specified probability. The use of the real valued GA had a significant influence on the quality of numerical results, as is borne out by the numerical examples in a later section.

## 5 Parallel implementation

The process of extracting rules for CA simulation is computationally intense; in each generation of the GA evolution, each rule in the population must be applied to evolve a CA, and its fitness evaluated on the basis of the converged state of strains. To reduce the wall-clock time for this procedure, the entire process can be implemented on a parallel computing architecture. The optimal parallel strategy is based on the number of processors to be employed (NPROC), the memory available to each processor (NMEMP), and the amount of memory needed to store and update the CA algorithm (NMEMC). The last item is proportional to the number of cells used to describe the complete structure. Two distinct possibilities must be considered. First, if the NMEMC < NMEMP (i.e. we can store the entire CA data structure on the memory allocated to each processor) and NPROC < NPOP, then the most efficient way to proceed is as follows.

1. Initialize the entire population of chromosomes on the master processor.
2. The GA algorithm begins by sending a list of chromosomes to each processor, where the list size is NPOP/NPROC; in general this will not be an integer and may cause some imbalance in processor loading. The overall impact on the performance of such a mismatch, however, is minimal.
3. Each processor then works in parallel. for each chromosome (rule set) the CA is evolved until the cells reach a steady-state value after completing the evolution, the strain energy corresponding to the CA solution is computed, compared with the the external work, and a fitness assigned to the rule, this chromosome fitness is then stored in an array of the same shape as the chromosome itself.
4. After completing the fitness evaluation for each member of its portion of the chromosome pool, each processor sends the fitness values back to the master process to be evolved.
5. With the entire populations fitness measures available, the master then evolves the population using a standard GA approach.
6. If the population is not determined to be converged, the algorithm repeats by returning to step 3. If a converged best rule is obtained, the algorithm stops and reports that best rule to the user.

This algorithm requires communication that scales only with the number of rules that exist in the population, and is independent of the size of the CA. It can therefore be expected to scale very well so long as the amount of

**Table 1** Parallel implementation results

| No. of processors | Grid size | No. of CA iterations | No. of GA iterations | population size | work $\times(10^{-9})$ | work/ processor | wall clock time (sec) |
|---|---|---|---|---|---|---|---|
| 1 | $10\times10$ | 400 | 100 | 316 | 1.26 | 1.26 | 1097.6 |
| 4 | $20\times20$ | 800 | 100 | 320 | 10.24 | 2.56 | 2240.6 |
| 8 | $20\times20$ | 800 | 100 | 320 | 10.24 | 1.28 | 1193.6 |
| 16 | $40\times40$ | 1600 | 100 | 3201 | 81.92 | 5.12 | 5149.3 |
| 32 | $40\times40$ | 1600 | 100 | 320 | 81.92 | 2.56 | 2719.3 |

work required to evaluate the CA is large relative to the amount of work required to perform a single generation of GA evolution and to communicate the rules down to the processors and fitness back to the master node. A series of tests were performed to assess the scaling on an IBM SP2 machine, using as a test article, the plate with a circular hole described in a later section. Results from this study are summarized in Table 1.

A second form of parallelization is required when the problem size gets very large and/or the fidelity of the CA is pushed to very high levels. Here, it may not be possible to store the entire CA data structure on the memory allocated to a each processor. A modified strategy is being implemented for this scenario, and will be reported elsewhere.

## 6 Numerical implementation

The approach described in preceding sections was implemented for two linear elasticity problems. The first is a cantilever beam structure with a tip load as shown in Fig. 4. Analytical solutions to the $u-v$ displacement field for this problem are obtained as follows:

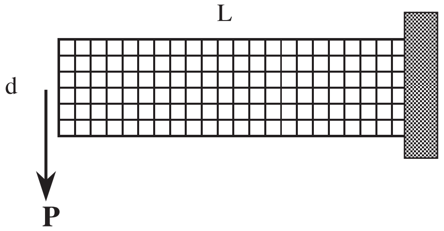

**Fig. 4** A tip loaded cantilever beam structure

$$u = -\frac{P}{2EI}x^2y + \frac{P}{3EI}\left(1+\frac{v}{2}\right)y^3 +$$

$$\frac{P}{2EI}\left[L^2-(1+v)\frac{d^2}{d}\right]y\,,$$

$$v = \frac{vP}{2EI}xy^2 + \frac{P}{6EI}x^3 - \frac{PL^2}{2EI}x + \frac{PL^3}{3EI}\,. \tag{6}$$

Here $P$ is the applied load, $E$ is the Youngs modulus, and $I$ is the bending moment of inertia; $L$ and $d$ are geometric parameters identified in the figure. The second problem deals with a plate subjected to inplane loads along an edge, and with a circular hole at its centre. This structure is shown in Fig. 5 and was selected as a problem with somewhat greater challenge, given the strain concentration effects near the hole boundary.

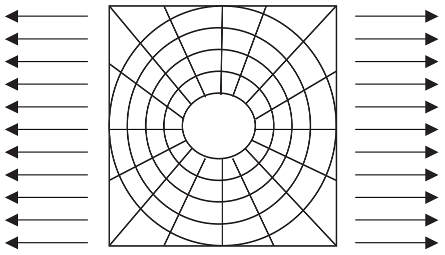

**Fig. 5** A flat plate with a hole and inplane loading

### 6.1 Problem 1

The analytical strain distributions obtained upon differentiation of 6 are shown in Fig. 6. For a cellular automata based solution of the 2-D state of strain in the beam, the structure was discretized into a $n\times m$ uniform grid, where $n$ and $m$ denote the number of cells in

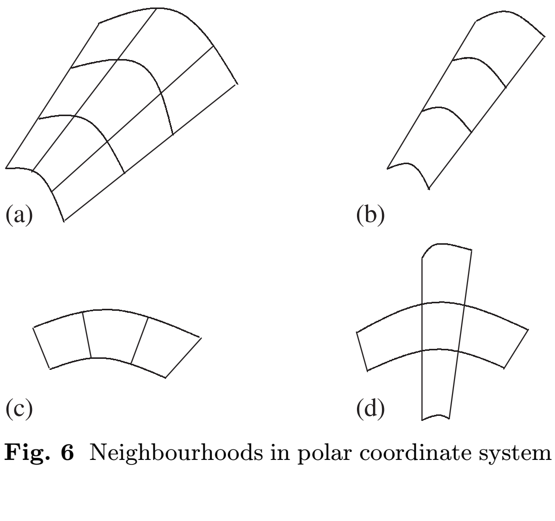

**Fig. 6** Neighbourhoods in polar coordinate system

the $x$ and $y$ directions, respectively. The state of each cell was defined in terms of the normal and shear strain components, respectively. Known values of these strains were specified along the supported boundary and along the mid plane of the beam; all other cells were set at zero strain values. A linear weighted averaging procedure was used as the base model to discover the rules for interaction. To use the GA based energy minimization, a population of weights was initialized at random. Each of these weights were then used to evolve the given initial state along with the boundary conditions to a converged state. The strain energy corresponding to the converged state for each rule was computed and used to establish the fitness function for the GA. Evolving the population over many generations of evolution minimized the fitness function. A constraint requiring that the strain energy be equal to the external work was also imposed, although such a constraint is not strictly required. In this optimization, a binary coded GA was used to determine three different sets of optimal weights, one for each of the strain components. Furthermore, separate rules were derived for cells on the boundary of the domain as they do not have neighbours on one side. It is important to bear in mind that the binary coded GA searches through a discrete subset of the continuous variable space. A search with greater resolution requires longer binary strings, which makes the search process less efficient. The string length were selected such that the weight components were searched in increments of $\Delta w = 0.012$. The optimal sets of weights obtained in this optimization yield a CA evolution of strains as shown in Fig. 7. While the normal strain components show reasonable agreement with the analytical results of Fig. 6, there is considerable discrepancy in the shear strain components.

In another numerical experiment, in addition to values of the optimal weight sets, the optimal neighbourhood of interaction was a variable to be determined from the optimization process. For this exercise, 5 different neighbourhoods were candidates for selection – the Moore neighbourhood, N-S-E-W, N-S, E-W, and NE-NW-SW-SE. Each of these has a different number of weights to be determined. Using the binary coded GA, this optimization problem in a mix of discrete and continuous variables was solved as before to minimize the strain energy. The optimal rule sets for this problem were interesting. While the Moore neighbourhood was picked as the optimal neighbourhood for interaction for the normal strain components, a N-S neighbourhood was established for the shear strains. These neighbourhoods along with the optimal values of the weights yields a strain distribution pattern as shown in Fig. 8.

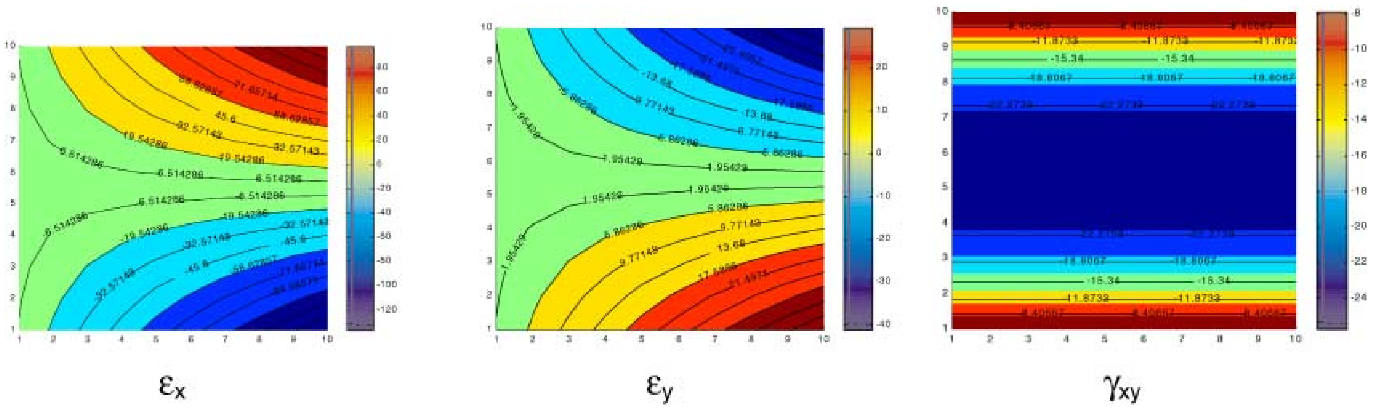

**Fig. 7** Analytical strain distributions for cantilever beam

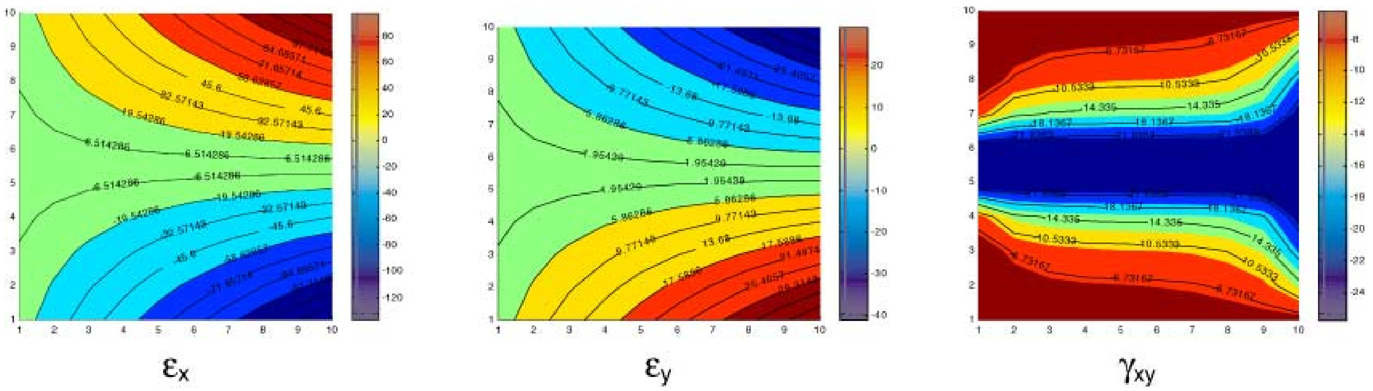

**Fig. 8** Strain distribution obtained from the CA evolution

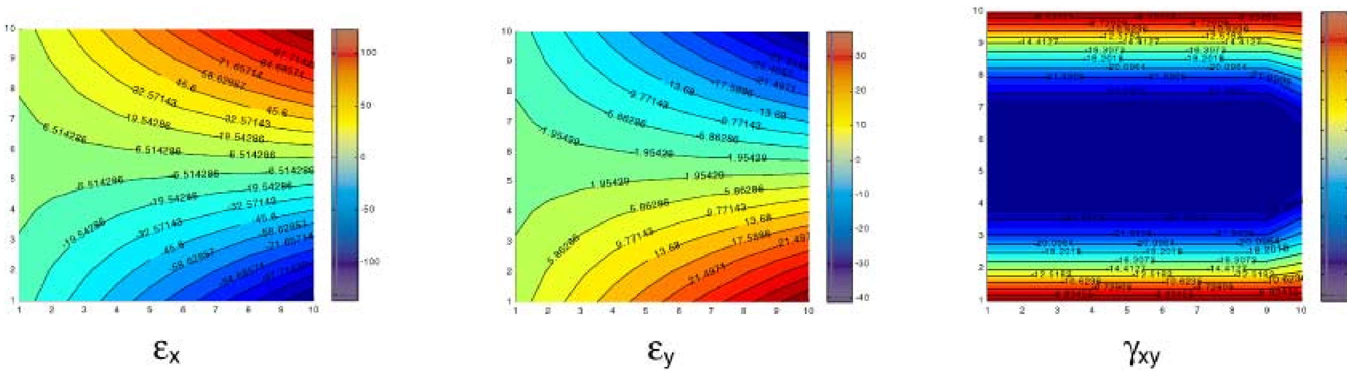

**Fig. 9** CA evolution with optimal neighbourhood interaction

**Table 2** Ca implementation results

| | Cantilever beam | | | Plate with a hole | | |
|---|---|---|---|---|---|---|
| CA evolution strategy | strain field | | displacement field | strain field | | displacement field |
| Design variable representation for GA | binary | real–valued | real–valued | binary | real–valued | real–valued |
| population size | 200 | 100 | 200 | 100 | 100 | |
| prob. of crossover | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 |
| prob. of mutation | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 |
| total external work (lb · in) | 426.7 | | | 2581.6 | | |
| converged minimum strain energy (lb · in) | 432.9 | 427.5 | 427.3 | 2590.8 | 2581.9 | 2582.2 |

It is clear from this figure that the distribution of the shear strain compares better with the known analytical solution.

Using the optimal neighbourhoods of interaction identified in the previous step, the optimal weights were also determined using a continuous valued GA. This approach removes the artificial constraint of searching from a discrete representation of a continuous space. Indeed, as shown in Table 2, the optimal weights determined from this optimization yields a minimum energy that is very close to the external work. The corresponding strain distributions are shown in Fig. 9.

## 6.2 Problem 2

A second elasticity problem used for numerical testing involved a flat plate with a circular hole and subjected to inplane loading. A Moore neighbourhood (A) of the type shown in Fig. 10 was used to define the cell and its immediate neighbourhood. To facilitate the implementation of the CA approach, the plate was divided into a grid structure as shown in Fig. 5. For this problem, it was convenient to compute the strains in the polar coordinate system. As in the previous problem, the strain

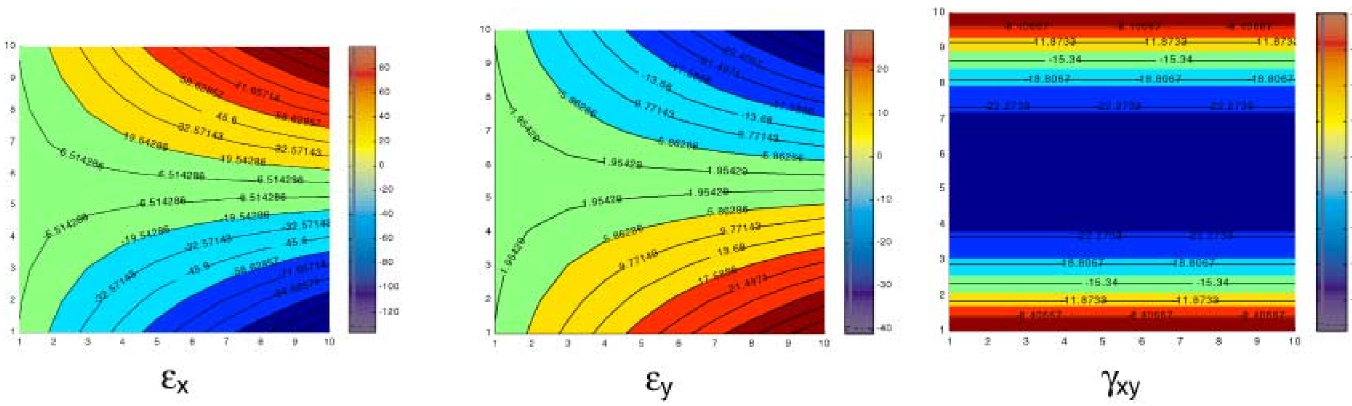

**Fig. 10** Strain distributions for CA solution with weights discovered using continuous valued CA

components comprise the state of an individual cell and collectively, the state of the structure. In searching for the optimal neighbourhoods, the four options that were included in the optimal search are also shown in Fig. 10. As in the previous example, separate rules were derived for cells on the boundary of the structure. Furthermore, separate sets of weights were determined for each of the strain components. Both real and binary coded GA's were used in the optimal search; however, result are presented here only for the real valued implementation. Figure 11 shows the distribution of analytically computed strain in this plate under the applied loads. The strain distribution obtained for a $20 \times 20$ grid is as shown in Fig. 12, and compares very favourably with the analytical solution of Fig. 11. Table 2 also shows that the minimal strain energy is very close to the value of the external work. The optimal neighbourhoods corresponding to this solution was a Moore neighbourhood for the $\varepsilon_r$ and $\varepsilon_\theta$ components, and the neighbourhood B from Fig. 10 for the $\varepsilon_{r\theta}$ component.

An alternative definition of the state variables was also considered for both of these problems. For example, in the first problem, the displacements $u$ and $v$ at each cell site were used to defined the cells states in place of the three strain components. The advantage of using these more basic variables is two fold. It not only al-

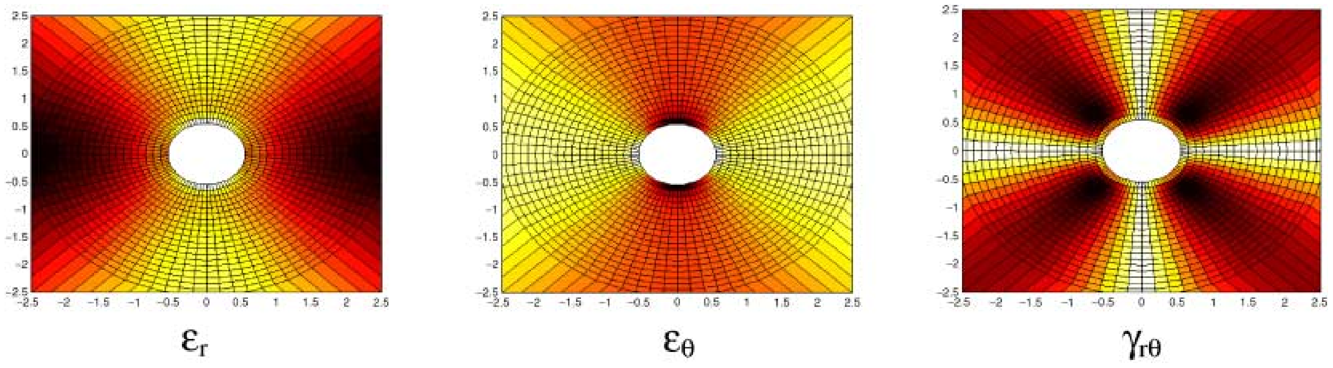

**Fig. 11** Analytical strain distribution for the flat plate

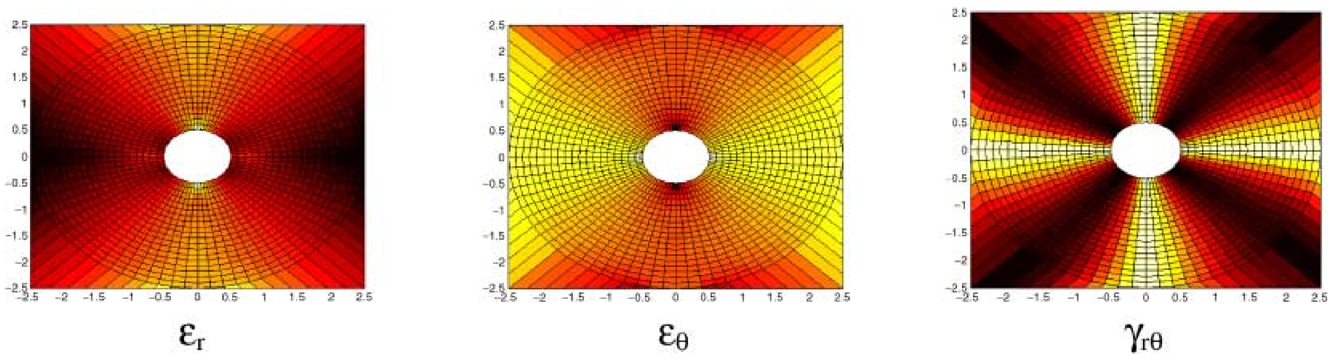

**Fig. 12** Majority vote rule

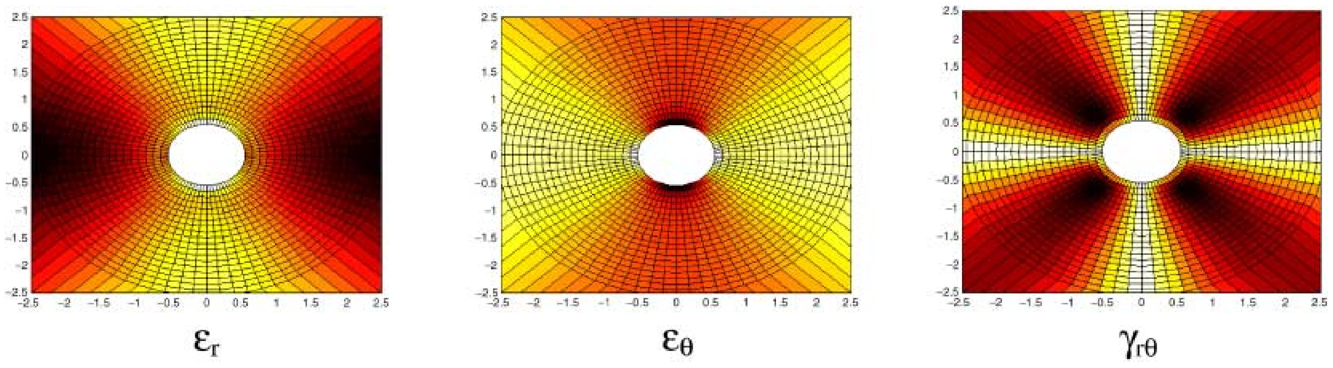

**Fig. 13** Strain distribution obtained from evolving displacements

lows for an easier definition of boundary conditions but also reduces the number of rule sets that must be independently derived during optimization. For example, after evolving $u$ and $v$, it is possible to compute each of the three strain components from this field using a finite difference approximation. This approach, when implemented for the flat plate problem, yields strain fields as shown in Fig. 13. These strains are virtually identical to those obtained using direct strain evolution, and yield a minimal strain energy very close to the theoretical value.

## 7 Closing remarks

The paper describes a cellular automata based approach for the analysis of 2-D elasticity problems. The unique aspect of this work is the derivation of the required rules for CA evolution using a genetic algorithm based search procedure. The guiding principle in rule selection is the principle of minimum energy. The approach, as presented in this work, allows for the exploration of discrete concepts (different neighbourhoods of interaction), discrete models for the rules (selection from among many different rules of interaction), and a continuous refinement of a selected model (by refining the weighting coefficients used in rule definition). The approach is implemented in the analysis of two elasticity problems. The CA approach is assessed for performance with changes in the rules for interaction, including both the interaction model as well as the neighbourhood of interaction. Rules for interaction have been derived for different choices of the state variables. Evolving the displacement field and computing strains from this field by finite difference produces results similar to those obtained through evolving the strain fields directly. Issues relating to a parallel implementation of the rule learning process are also examined.

*Acknowledgements* Support received for this work under grant NAG 1–2089 from the NASA Langley Research Center is gratefully acknowledged.

## References

Cai, J.; Thierauf, G. 1997: Evolution strategies in engineering optimization. *Eng. Opt.* **29**, 177–199

Devreotes, P. 1989: Dictyostelium: a model for cell–cell interactions in development. *Science* **245**, 1065

Giunta, A.A. 1997: *Aircraft multidisciplinary design optimization using design of experiments theory and response surface modeling*. Ph.D. Dissertation, Virginia Polytechnic Institute and State University

Hajela, P. 1990: Genetic search – an approach to the nonconvex optimization problem. *AIAA J.* **26**, 1205–1210

Hajela, P.; Kim, B. 1999: GA based learning in cellular automata models for structural analysis. *3rd World Cong. on Structural and Multidisciplinary Optimization* (held in Niagara Falls, NY, May)

Hardy, J.; De Pazzis, O.; Pomeau, Y. 1976: Molecular dynamics of a classical lattice gas: transport properties and time correlation functions. *Phys. Rev. A* **13**, 1949–1960

Orszag, S.; Yakhot, V. 1986: Reynolds numbers scaling of cellular-automaton hydrodynamics. *Phys. Rev. Lett.* **56**, 1691–1693

Pasteels, J.M.; Deneubourg, J.L. (eds.) 1987: *From individual to collective behavior in social insects* (Proc. Les Treilles Workshop), Experientia Supplementum, Vol. 54. Basel: Birkhauser

Rothman, D.H.; Zaleski, S. 1997: Lattice – gas cellular automata – simple model of complex hydrodynamics. Cambridge, UK: Cambridge University Press

Rumelhart, D.E.; Hinton, G.E.; McClelland, J.L. 1986: A general framework for parallel distributed processing. In: *Parallel distributed processing*, Vol. 1, pp. 45–76. Cambridge, MA: MIT Press

Wright, A. 1991: Genetic algorithms for real parameter optimization. In: Rawlin, G.J.E. (ed.) *Foundations of genetic algorithms 1*. San Mateo, CA: Morgan Kaufmann

Zienkiewicz, O.C.; Taylor, R.L. 1989: *The finite element method, Vol. 1, Basic formulation and linear problems*. New York: McGraw Hill