Genetic Algorithms for the Calibration of Cellular Automata Urban Growth Modeling

Jie Shan, Sharaf Alkheder, Jun Wang

Abstract

This paper discusses the use of genetic algorithms to enhance the efficiency of transition rule calibration in cellular automata urban growth modeling. The cellular automata model is designed as a function of multitemporal satellite imagery and population density. Transition rules in the model identify the required neighborhood urbanization level for a test pixel to develop to urban. Calibration of the model is initially performed by exhaustive search, where the entire solution space is examined to find the best set of rule values. This method is computationally extensive and needs to consider all possible combinations for the transition rules. The rise in the number of variables will exponentially increase the time required for running and calibrating the model. This study introduces genetic algorithms as an effective solution to the calibration problem. It is shown that the genetic algorithms are able to produce modeling results close to the ones obtained from the exhaustive search in a time effective manner. Optimal rule values can be reached within the early generations of genetic algorithms. It is expected that genetic algorithms will significantly benefit urban modeling problems with larger set of input data and bigger solution spaces.

Introduction

Calibration in cellular automata urban growth modeling is to find the best combination of transition rule values such that the modeled urban phenomena can match the real one. Only through calibration can the cellular automata model produce an urban level and urban pattern close enough to reality. Calibration is critical in validating the performance of a designed cellular automata model (Batty and Xie, 1994a and 1994b; Batty *et al*., 1999; Landis and Zhang, 1998) and remains to be a challenge. It has been neglected until recent efforts to develop cellular automata as a reliable process for urban development simulation (Wu, 2002). It is shown (Li and Yeh, 2002; Wu, 2002) that urban cellular automata models are sensitive to transition rules and their parameter values. The practical difficulty in calibrating cellular automata rules is due to the large search space and its exponential rise when more variables and larger variable ranges are involved in the transition rules.

A number of cellular automata calibration methods have been developed for urban growth modeling. They achieved various levels of success and efficiency. The structure of a calibration algorithm is mainly dependent on the design of the cellular automata model. Reviewing the existing calibration schemes shows various calibration styles. Statistical, visual, and artificial intelligence tools (e.g., neural networks) have been used for calibration. Clarke *et al*. (1997 and 1998) calibrated the SLEUTH model by using visual and statistical tests to find the best values for the five growth parameters. Such tests were repeated for each parameter set in coarse, fine, and final phases. This calibration process took extended CPU time to reach the most appropriate parameter set in the search space (Yang and Lo, 2003). Wu and Webster (1998) defined the cellular automata transition rules using the multi-criteria evaluation (MCE) method. The objective of calibration in their model was to find the best weight factors for the input variables to match the real urban growth. Neural networks were used by Li and Yeh (2002) to calibrate the cellular automata model. A training set representing different combinations of parameter values and their corresponding modeling output were used to train the network to reproduce the desired urban pattern. The calibration in Wu (2002) estimated the probability of a particular state transition occurring at a location through a function of the development factors to balance the development probability, threshold, and allowed land consumption to reproduce the urban growth pattern.

One of the recent studies in cellular automata model calibration is based on genetic algorithms. Genetic algorithms use the biological principles to direct the search towards regions (sub-space) of the solution space with likely improvement (Goldberg, 1989). Initial efforts tried to attach genetic algorithms to cellular automata urban model design for performance improvement. Colonna *et al*. (1998) modeled the changes in land-use for Rome, Italy through using genetic algorithms to produce a new set of rules for the cellular automata model. Genetic algorithms were used to find the optimal set of possibilities of land-use planning for Provo, Utah (Balling *et al*., 1999). Wong *et al*. (2001) used genetic algorithms for the primordial Lowry model in an attempt to choose the parameters of household and employment distribution for Hong Kong. A recent study tried to formalize genetic algorithms as a calibration tool for the SLEUTH model (Goldstein, 2003). The calibration results of the genetic algorithms were compared with the traditional, exhaustive search method through designing a comparison measure (metric of fit) as the product of three spatial metrics: the number of urban pixels, the number of urban clusters, and the Lee-Sallee index (Lee and Salle, 1970). This work was a good start towards formalizing the calibration process in cellular automata modeling; however, further improvements are

Geomatics Engineering, School of Civil Engineering, Purdue University, 550 Stadium Mall Drive, West Lafayette, IN 47907 (jshan@ecn.purdue.edu).

Photogrammetric Engineering & Remote Sensing
Vol. 74, No. 10, October 2008, pp. 1267–1277.

099-1112/08/7410-1267/$3.00/0

needed to make it more robust. The objective function definition is one of the points that need enhancement since the metric fit did not show how accurately the genetic algorithms can reproduce the same real urban pattern. Furthermore, the selection step in genetic algorithm did not give higher priority to strings with better objective values. A rank selection step may produce better results if applied since this will give strings with a higher fit a better chance for selection.

Our work is focused on improving cellular automata calibration efficiency through genetic algorithms. The cellular automata model is designed as a function of multitemporal satellite imagery and population density. The transition rules identify the required neighborhood urbanization level for a test pixel to develop while satisfying the imposed constraints. To consider the spatial variation in urban dynamics, the entire study area is divided into townships of a semi-grid (see Figure 1). The same transition rules are applied to all townships, however, each of them is calibrated separately to determine its optimal rule values. The objective of this study is to use genetic algorithms to optimize the search method for best transition rule values. The proposed calibration approach consists of the following steps. First, an initial population of solutions (rule values) are randomly generated and encoded to binary (solution) strings. The cellular automata model is then run for all those initial solutions. In the next step, ground truth, if available, is used to rank the initial solutions based on a predefined objective function. The third step is producing the next generation of solutions by using genetic operations, such as elitism, rank selection, crossover, and mutation. Finally, the cellular automata are run again for the newly generated solutions. The above procedure is repeated for a maximum number of iterations (called generation) or until a required minimum objective function value is achieved. The paper will start with a brief introduction to cellular automata and the objective function used for quality evaluation. Genetic algorithms are then described under the scope of transition rule calibration. Modeling results and properties of the genetic algorithms are evaluated based on the study over Indianapolis city, Indiana.

Cellular Automata for Urban Growth Modeling

Cellular automata was originally introduced by Ulam and von Neumann in the 1940s to study the behavior of complex systems (von Neumann, 1966). It can be regarded as a dynamic discrete system in space and time that operates on a uniform grid (such as an image) being controlled by a predefined set of transition rules. It consists of cells or pixels, their states (such as land-cover classes), neighborhoods (e.g., square), and transition rules. The future state of a pixel in cellular automata is dependent on its current state, neighborhood states, and transition rules. Iterative local interaction between pixels within the neighborhood will finally produce the global urban pattern.

According to the early work (Alkheder and Shan, 2006) a cellular automata urban growth model is designed as a function of two types of data: historical satellite imagery and population density. A set of satellite images over Indianapolis, Indiana from the period of 1982 to 2003 are classified according to Anderson *et al.* (1976) system to produce a set of historical land-cover images. Seven classes are identified in the images, namely: water, road, residential, commercial, forest, pasture, and row crops, with commercial and residential classes representing the urban class of interest. Besides these images, the cellular automata model uses population density as another input. An exponential function of the distance between a census tract centroid and the overall city centroid is used to represent the population density. Based on the census tract maps of year 1990 and 2000, the model is used to calculate the population density

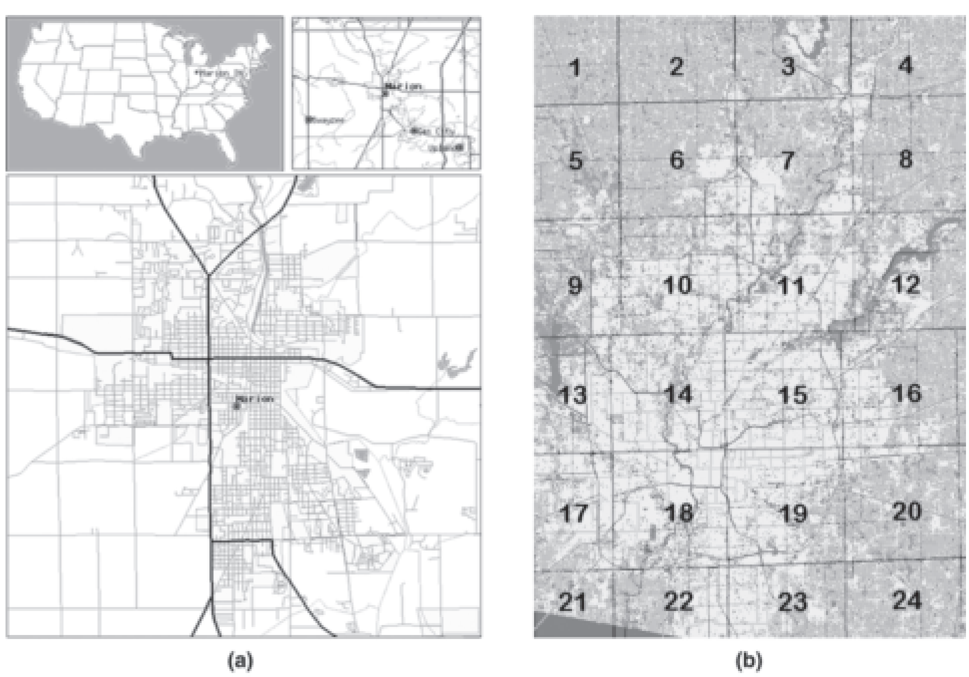

Figure 1. City of Indianapolis and township map, Indiana: (a) Indianapolis (US Census Bureau), and (b) Township map. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

for each pixel throughout the years from 1982 to 2003. Figure 1 shows the map and image of Indianapolis and its townships of a semi grid system overlaid atop.

The transition rules for the cellular automata urban growth model are designed using the above input to identify the urbanization conditions needed in the 3×3 neighborhood for a test pixel to become urban. Growth constraints for certain land-cover classes are considered as well. These transition rules are summarized as follows:

1. IF the test pixel is water, road OR urban (residential or commercial) THEN no change.
2. IF the test pixel is non-urban (forest, pasture or row crops) THEN it becomes urban if:
   - Its population density is equal or greater than P_i AND its neighboring residential pixel count is equal or greater than R_i; or,
   - Its population density is equal or greater than P_i AND its neighboring commercial pixel count is equal or greater than C_i.

where P_i is the population density threshold (0 to 3 range with 0.1 increment), and R_i and C_i are thresholds to residential and commercial rule (0 to 8 range with integer increment of 1 for a 3×3 neighborhood), respectively. There are a total of 2,511 ($9 \times 9 \times 31$) combinations of possible rule values. To consider the spatial variability of urban development, the study area is divided into 24 township grids, with each being approximately 9.6 km $\times$ 9.6 km. Although the same set of rules is applied to all townships, each township is calibrated separately and had its own rule values (R,C,P). The objective of calibration is to determine the values for the above rule thresholds such that the transition rules can best model the urbanization behavior.

To select the best rule values, certain criteria must be introduced. In this study, we use fitness and total error as evaluation measures. Fitness is the ratio of the modeled urban pixel count to the ground truth urban count. It measures how the modeled urbanization level fits the reality.

$$F\,\% = Fitness\,\% = \frac{Modeled\ urban\ count}{Ground\ truth\ urban\ count} \times 100. \tag{1}$$

The total error represents the overall number of erroneous (mismatched) pixels with respect to the total number of pixels (total count):

$$\Delta E\,\% = Total\ error\,\% = \frac{Total\ error\ count}{Total\ count} \times 100. \tag{2}$$

The combination of the above two measures is used as the objective function Φ in the genetic algorithms for calibration:

$$\Phi = Abs\,(Fitness - 100\%) + Total\ Error. \tag{3}$$

The rule values that lead to the minimum objective function will be selected as the best calibration outcome. Minimum objective function represents the rule values that produce the modeling results with fitness close to 100 (urban level close to reality) and small total modeling errors (urban pattern close to reality). Calibration based on exhaustive search examines the entire solution space (2,511 possible combinations for each township), whereas genetic algorithms only search part of the entire solution space, both with the objective to find the best rule combination (R,C,P).

Genetic Algorithms for Calibration

Genetic algorithms were first introduced by Holland (1975) as a method of mimicking meiosis in cellular reproduction to solve complex problems. The strength of genetic algorithms over other traditional search methods is their ability to effectively find global optima in the solution space. The performance of the genetic algorithms depends on the design of genetic operations, including string encoding, selection, crossover, and mutation. Genetic algorithms start with a set of initial solutions, which are usually generated randomly. Each solution set is called a population. A series of genetic operations are then applied to the solution population to create the next solution generation. Such a process continues for a number of times (generation), among which the one with the minimum objective function will be chosen as the final solution. This section will first describe the genetic operations in detail as pertinent to the cellular automata, and then present their implementation in urban modeling.

Encoding and Initial Population

Genetic algorithms usually do not directly handle the parameter values, but their encoding. This is the first step towards applying the genetic algorithms. This study adopts the common binary encoding technique. Each combination of $(R,C,P)_i$ represents one solution string. R_i and C_i are in the possible range of [0 to 8] integer values and have a binary coding range of [0000 to 1000]. P_i continuously ranges between 0 and 3, and is scaled to 0 to 30 as an integer for encoding purpose, which corresponds to the binary encoding range of [00000 to 11110]. An example of a rule string (5, 7, 1.0) is encoded as a binary string [0101<u>0111</u>01010], in which the first four digits are for R_i, the second four digits for C_i, and the last five digits for P_i. The underlined part in the string is for C_i and meant to separate the three rule values for better legibility. Each string, either in its original form or encoded binary form corresponds to one rule combination. For initial population, a number of such solution strings are created randomly and encoded to binary strings. In this study, the size of each population is 30, i.e., there are 30 solutions in each population.

Rank Selection and Elitism

Once the initial population is prepared, the cellular automata model is run for each string in the population until a year with ground truth (calibration year). The modeling results are then evaluated in terms of the objective function defined in the previous section. The rank selection and elitism operations are first used to select the strings for the next generation. For rank selection, all strings are ordered based on their objective functions in ascending order (from minimum to maximum). The string with the lowest objective function is given a rank of 30, the second 29, etc., until the last string, which will receive a rank of 1. The selection probability (p_i) for each string is then calculated based on the ratio between the string rank r_i and the sum of all of the ranks:

$$p_i = \frac{r_i}{\sum r}. \tag{4}$$

Multiplying the selection probability with the population size (30 strings) will identify how many copies each string is expected to contribute to the next generation. For example, a string with a selection probability of 0.06 will contribute 1.8 (30×0.06), rounded up to two strings in the next population.

According to elitism selection, the best six strings out of 30 in terms of their objective functions are copied directly to the next generation. This step is performed to retain good strings of the current generation. The remaining 24 strings in the next generation are selected by using rank selection from the current 30 strings based on their selection probability.

As is stated above, during this process some strings may be repeated more than once, and some will simply not be selected due to their low selection probability.

Crossover and Mutation

Crossover and mutation are applied to the selected 30 strings from the above step. Crossover mimics the natural process in biology where good strings from parents meet to produce offspring that are expected to be the same or better quality than their parents. Crossover occurs with two strings and yields two new strings. As an example, Table 1 shows a list of six strings which randomly form three crossover pairs. A single crossover point is used as the crossover pivot for each string pair. Under the crossover operation, bits after the fourth ($k = 4$, counting from 1) bit in the two strings are interchanged, and a new set of six strings are formed as the result of this operation.

The last stage in finalizing the new population (after crossover) is the mutation operation. Mutation produces more diverse structure types and prevents the solution from local minima. In this study, mutation is performed on $(R,C)_i$ parameters over the best six strings in the population. This process is done through random addition of $+1$ or -1 to these parameters as shown in Table 2 as an example.

Table 3 uses a population of six strings as an example to demonstrate the operation of genetic algorithms for transition rule calibration. After the initial population is prepared (column 1), the strings are encoded to binary (column 2). The cellular automata runs for all six strings to evaluate their objective functions (column 3). Strings are ranked (column 4) based on their objective functions or the selection probability (column 5). The expected count (column 6) contributing to the next generation is determined. Elitism (the best two strings in this example) and rank selection (strings with higher rank are selected to fill the remaining four spots in the population) are used to produce an intermediate new generation (column 7). At the end, crossover between the selected three pairs with $k = 4$ single point crossover (column 8) and mutation (through addition of $+1$ and -1 to the first and second best strings, respectively, column 9) are introduced to produce the final new generation (column 10).

Calibration

Calibration of transition rules is carried out through repeating the above genetic operations. Once a new population is obtained, the next step is to run the cellular automata model with the newly generated population and evaluate their objective functions. The genetic operations are applied to the newly created 30 strings based on their objective functions to produce a new generation of 30 strings, among which the rule producing the minimum objective function is selected. The above process repeats for 20 times or yields 20 generations. This will result in selecting the rule combination with the minimum objective function out of all the generations as the optimal solution. It should be noted that such process is carried out for each township, and every township gets its own best rule values.

This approach is used to model the historical urban growth of Indianapolis as an effective alternative to the exhaustive search. Starting from a set of initial solution population for each township, the process runs for 20 generations to find the best set of rules for each township, which are then used to model the urban growth between specific time epochs. Through the designed model, urban growth modeling is simulated from 1982 to 1987, where calibration is performed using genetic algorithms to find the best rule values. The best rules in 1987 are used to predict 1992 for a short term prediction of five years. Another calibration is carried out in 1992 to predict the urban growth in 2003 for an interval of 11 years. Finally, after calibration in 2003 the best rule values are used for future prediction in year 2010.

Evaluation of Calibration Results

Table 4 compares the computation time taken by the exhaustive search and genetic algorithms. The CPU time for genetic algorithms is far less than that for the exhaustive search. On a computer of Pentium-4 CPU with 3.4 GHZ and 2.00 GB RAM, genetic algorithms takes an average of about 6.5 hours, while about 27 hours is needed for the exhaustive search. Such computation efficiency is due to the reduction in the search space in genetic algorithms. In our implementation, only 600 (20 generations $\times$ 30 solutions per generation) out of all the 2,511 possible solutions needs to be searched, which yields about one-fourth the computation time of the exhaustive search.

Figure 2 shows the best objective functions at different generations for some selected townships that are spatially close to or away from the city center in order to cover different urban growth behaviors. As is shown, the objective function reaches its minimum at early stage (within the first ten generations) of the computation for some townships. However, some townships behave differently. The variability

Table 1. Crossover of Strings (Crossover Point $k = 4$)

String #	Input Strings	Crossover Pairs	After Crossover
1	0111<u>011</u>100001	1, 3	0101<u>011</u>100001
2	0111<u>000</u>01100		0111<u>000</u>101111
3	0101<u>000</u>101111	2, 4	0111<u>100</u>001100
4	0111<u>100</u>000110		0111<u>100</u>000110
5	1000<u>100</u>010011	5, 6	0111<u>100</u>010011
6	0111<u>011</u>101010		1000<u>011</u>101010

Table 2. Mutation of Strings

String #	Input String	Rule Value	Mutation Value	Rule Value After Mutation	String After Mutation
1	0111<u>011</u>100001	(7,7,1)	+1,+1	(8,8,1)	1000<u>100</u>000001
2	0111<u>100</u>001100	(7,8,12)	+1,−1	(8,7,12)	1000<u>011</u>101100
3	0101<u>000</u>101111	(5,1,15)	+1,+1	(6,2,15)	0110<u>001</u>001111

Table 3. Genetic Operations for Transition Rule Calibration

String #	(1) Rule $(R,C,P)_i$	(2) String	(3) Objective function	(4) Rank r_i	(5) Selection probability $p_i = r_i/\Sigma r$
1	(7, 7, 10)	0111011101010	0.4465	5	0.238
2	(4, 3, 18)	0100001110010	0.6616	4	0.190
3	(6, 7, 1)	0110011100001	0.6661	3	0.143
4	(6, 1, 11)	0110000101011	1.2553	1	0.048
5	(5, 6, 3)	0101011000011	0.7585	2	0.095
6	(5, 1, 15)	0101000101111	0.4455	6	0.286
New String#	(6) Expect Count $6 \times p_i$	(7) After Elitism and Selection	(8) After Crossover (2,6), (1,3), (4,5), $k = 4$	(9) New String (After Mutation)	(10) New Rule $(R,C,P)_i$
1	1	0101000101111	0111001110010	1000010010010	(8,4,18)
2	1	0111011101010	0100011101010	0011011001010	(3,6,10)
3	1	0101000101111	0101000101111	0101000101111	(5,1,15)
4	0	0101000101111	0101000101111	0101000101111	(5,1,15)
5	1	0111011101010	0101011101010	0101011101010	(5,7,10)
6	2	0100001110010	0111000101111	0111000101111	(7,1,15)

Table 4. Computation Time of Genetic Algorithms and Exhaustive Search

Calibration Interval	Genetic algorithms	Exhaustive search
1973–1982	7hr 53m 10s	32hr 04m 46s
1982–1987	4hr 33m 30s	17hr 55m 34s
1987–1992	4hr 37m 59s	19hr 08m 40s
1992–2003	9hr 6m 22s	39hr 18m 34s
Average	6hr 32m 45s	27hr 6m 54s

in the objective functions is related to the genetic operators, such as crossover and mutation, which update the string structures for better solutions. For some townships (e.g., township 1, 14, 19) the change pattern stabilizes after a certain generation; while for other townships, it continues to fluctuate over time throughout the searching process (e.g., township 7, 16, 24). This fact is clearer if we let the model run for more generations as shown in Figure 3, where the best objective functions for the selected townships (7, 9, 20) fluctuate considerably over the first 50 generations. The results demonstrate that the variation of the objective function occurs within and beyond the maximum iteration threshold we used (i.e., 20). This property can be mainly attributed to the mutation operation. The best six strings that are copied directly through elitism operation are exposed to considerable structure redesign through mutation, which produces new strings with rather different objective functions. This is important and essential to allow the calibration process to explore new possible solutions and reach the best solution at an early generation. The above behavior suggests that the objective function usually does not change smoothly over the generations, and the best solution can be achieved multiple times at different generations. This warrants the optimal results within a certain number of generations (e.g., 20 in this study) without searching the entire solution space.

Figure 4 further demonstrates the change of the two components (fitness and total error) in the objective function. The best rule corresponding to the minimum objective function is achieved that balances the fitness (presented in the left y-axis) and total error (presented in the right y-axis). Illustrated are two examples for townships 11 and 21 at two calibration years, where the minimum objective function corresponds to a fitness close to 100 and a small total error.

This section examines the difference between the modeling results obtained from the genetic algorithms and exhaustive search. The quality, including fitness and total error are compared for the two calibration methods. The numerical results for the calibration years (1987 and 1992) are shown in Table 5, and for the prediction years (1992 and 2003) in Table 6. Figure 5 and 6 present the corresponding graphics. The dF and d ΔE columns in Table 5 and 6 are respectively the fitness difference and total error difference between the two calibration methods (genetic algorithms minus exhaustive search). As is shown in Table 5 and 6, the average of the fitness differences between the two calibration methods are respectively −0.9 percent (1987) and −1.7 percent (1992) for calibration, and −4.4 percent (1992) and −8.2 percent (2003) for prediction; while the standard deviations are respectively 4.8 percent (1987) and 6.3 percent (1992) for calibration, and 7.2 percent (1992) and 13 percent (2003) for prediction. The same quantities for the total error differences are: average −1.6 percent (1987) and 0.7 percent (1992) for calibration, and 0.8 percent and 0.9 percent for prediction; standard deviation 5.6 percent (1987) and 3.7 percent (1992) for calibration, and 3.8 percent (1992) and 3.1 percent (2003) for prediction. These small numbers suggest that genetic algorithms can essentially produce the same level of total errors (maximum magnitude 1.6 percent difference) as the exhaustive search, whereas the fitness is more sensitive (up to 13 percent difference) to the calibration methods. This is because the fitness measure depends on the level of urban development: the same amount of modeling error will cause a smaller fitness offset in a well developed township than in an underdeveloped township (see Equation 1). Moreover, the fitness difference between the two calibration methods is accumulated in proportion to the prediction period. As is shown in Table 6, for a prediction over 11 years (1992 to 2003) its fitness difference is about 2 times the one over five years (1987 to 1992).

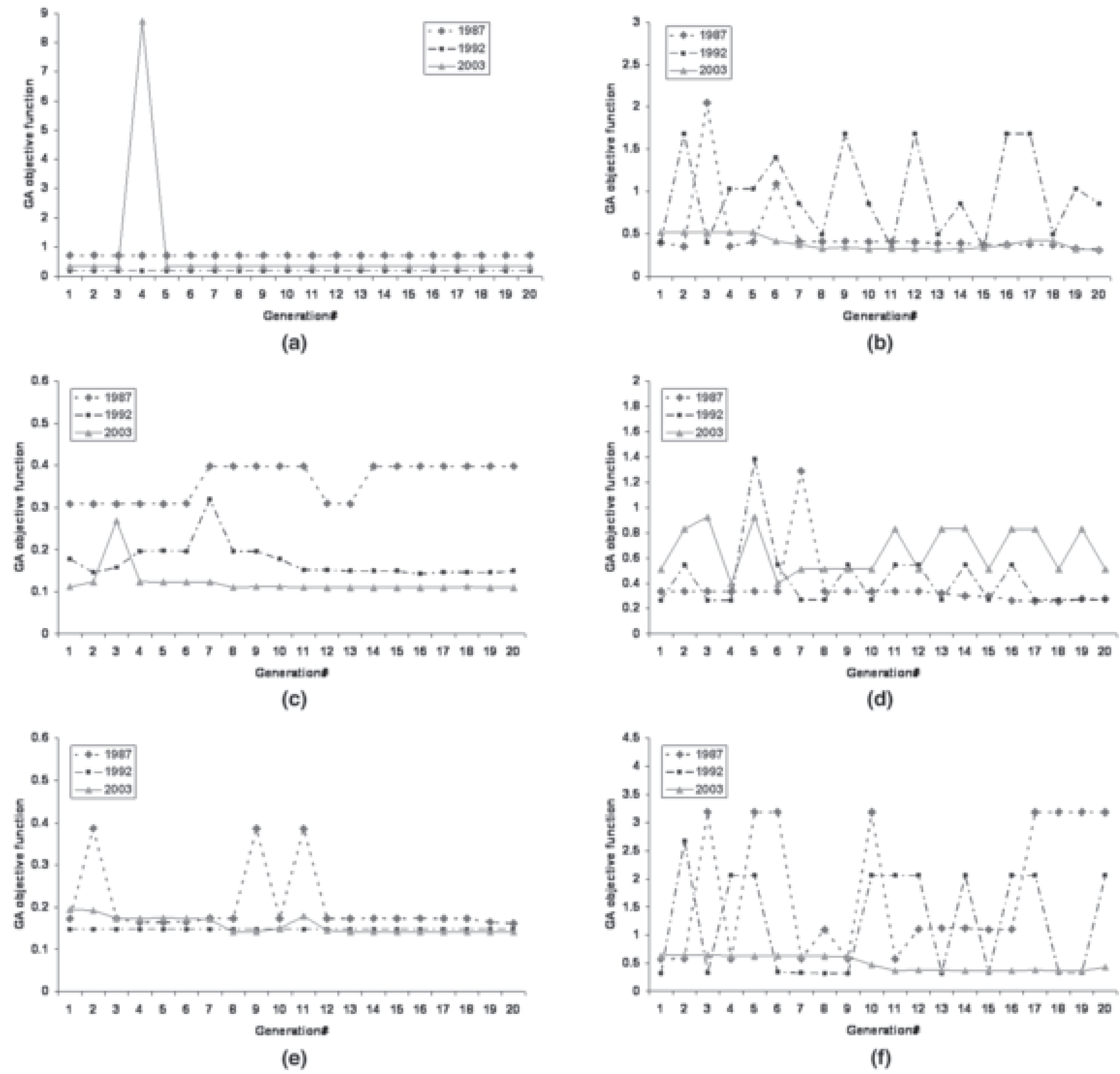

Figure 2. Best objective function at different generations for selected townships: (a) Township 1, (b) Township 7, (c) Township 14, (d) Township 16, (e) Township 19, and (f) Township 24. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

Finally, the modeling results obtained from genetic algorithms with reference to the ground truth images are quantified by the F and ΔE columns in Table 5 and 6, and the plots in Figure 5 and 6. Because of the general close match (small dF and d ΔE) between the two calibration methods, modeling outcome from genetic algorithms follows the same behavior as the one from the exhaustive search (Alkheder and Shan, 2006), both yielding an average modeling error of ~25 percent. Figure 7 presents the real cities obtained from image classification and the modeling results from cellular automata with genetic algorithms. These results show a close urban pattern match with reality for both the calibrated and predicted images. This comes as a result of minimizing the objective function that combines the fitness discrepancy and total error. It should be noted that township based spatial calibration contributes in this regard as well, since it takes into account the spatial variation in urban dynamics. The small differences between genetic algorithms and exhaustive search method show the ability of the proposed calibration algorithm in reproducing the real urban growth pattern. This encourages adapting the genetic algorithms as a replacement to the exhaustive search to save computation efforts while preserving practically the same modeling quality.

Conclusions

Utilization of the developed cellular automata urban models needs effective calibration due to the likely large number of variables involved in the modeling process. Genetic algorithms are proposed as a suitable solution in this study due to their ability to find an optimal transition rule set through searching a limited solution space and their flexibility to accommodate multiple variables in urban modeling.

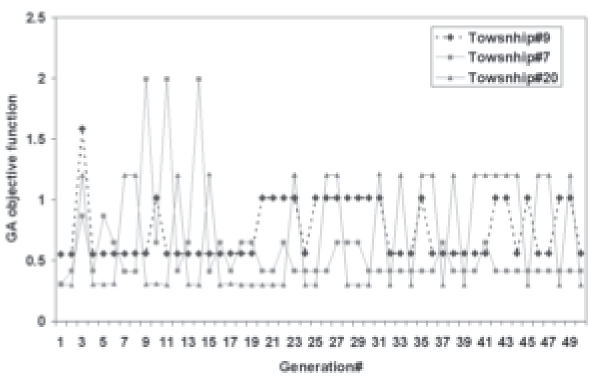

Figure 3. Objective function variation over 50 generations for year 1987. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

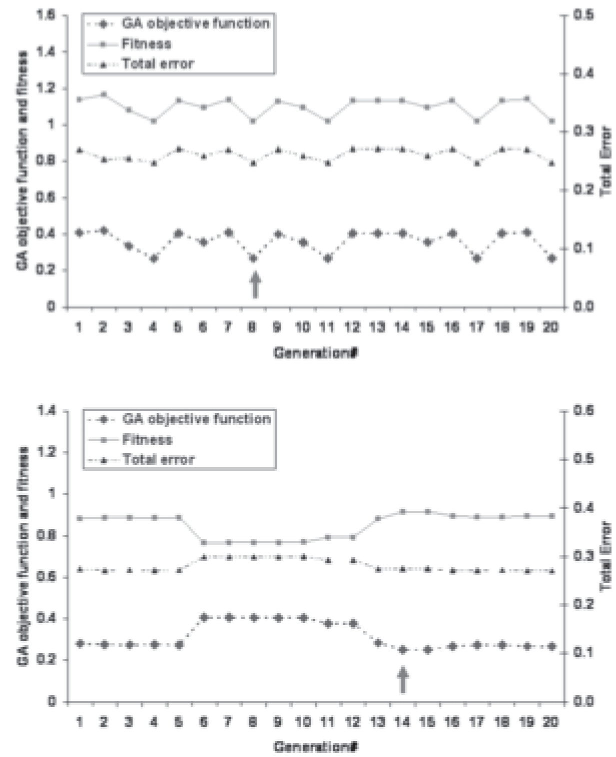

Figure 4. Fitness, total error, and objective function at different generations: (a) Township 11, 1987, and (b) Township 21, 1992. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

A number of experiences are gained in applying genetic algorithms to the cellular automata modeling. A proper selection of the initial solution population and other parameters for encoding, crossover, and mutation is necessary. It is essential that a set of transition rules are encoded as a binary string such that the genetic operations can be applied. The design of genetic algorithms objective function to represent fitness and total error in terms of urban level and pattern respectively proves to be efficient in directing the search process towards the optimal solution. Minimizing the objective function ensures that the selected rules yield satisfactory urban level and pattern close to reality. Rank selection in genetic algorithms through favoring solutions with smaller total error and better fitness succeeds in focusing the searching algorithm towards better solutions. Crossover and especially mutation operations play an essential role in approaching the optimal solutions at an early generation of the computation.

Genetic algorithms can produce the modeling results virtually as good as the exhaustive search in terms of the defined objective function. This is demonstrated by cross-validation between the two methods and with reference to the ground truth. Larger difference between the two calibration methods occurs in fitness with an average of −8.2 percent and standard deviation of 13 percent for the prediction over a period of 11 years; whereas the difference in the total error is less than 5.6 percent in both calibration and prediction.

Spatial and temporal calibrations of the cellular models are also applied in this study and proved to be necessary and effective. The township based spatial calibration takes into account the spatial variation in urbanization, whereas the temporal change in urban dynamics is considered by calibration with reference to the historical images over the period of modeling. It is shown that genetic algorithms can find the optimal rule set for each township and many townships can reach their best rules within the first ten generations.

The computation time is significantly reduced from more than a day (27 hours) in the exhaustive search to a few (6.5) hours in genetic algorithms. This relation is the proportion between the search spaces of the two methods. It is expected that genetic algorithms will even more significantly benefit urban modeling problems that have larger set of input data and bigger solution spaces.

Table 5. Numerical Results of Calibration

Township #	1987					1992				
	Gen#	Fitness %		Total Error %		Gen#	Fitness %		Total Error %	
		F	dF	ΔE	d ΔE		F	dF	ΔE	d ΔE
1	1	148.8	−0.02	22.93	0.34	1	101.34	−0.06	20.13	−0.72
2	1	110.0	0.05	23.21	0.21	1	110.27	0	23.66	−9.51
3	1	96.7	−3.24	26.36	−0.02	7	87.46	−13.33	33.05	2.69
4	1	99.2	−0.76	25.42	0.12	2	100.57	0.68	29.73	0.9
5	1	113.4	−0.09	23.84	0.20	19	100.52	0.77	25.78	0.74
6	7	99.3	−0.73	25.1	0.01	1	82.21	−17.59	28.87	1.01
7	20	99.9	0.02	30.78	0.28	11	100.19	1.38	34.94	1.26
8	1	102.2	−0.04	26.81	0.01	19	96.84	−1.38	30.31	−0.68
9	3	100.1	6.79	28	0.68	2	96.22	−3.99	32.23	2.56
10	1	100.3	3.42	27.59	0.59	13	102.60	2.55	21.83	0.13
11	8	101.9	2.05	26.15	0.88	19	100.39	0.41	20.79	0.85
12	8	100.1	0.66	31.38	0.18	15	81.82	−17.95	35.14	1.35
13	11	85.5	−16.48	22.85	−2.64	11	99.97	0.84	24.34	1.01
14	2	86.6	−10.79	18	−1.87	16	100.14	0.02	15.29	0.7
15	1	101.1	1.18	11.78	1.51	9	100.05	0	10.20	0.61
16	18	100.2	1.22	25.93	0.08	1	101.60	0.03	26.78	−0.47
17	2	90.9	−8.39	24.86	−0.61	9	100.17	0.15	27.60	3.26
18	1	98.7	−1.18	10.89	−0.39	1	101.03	0	10.95	−0.22
19	20	100.1	0.92	16.78	0.32	2	100.08	0	16.22	0.74
20	6	99.4	−0.59	29.42	0.02	2	108.07	0.88	29.02	−6.68
21	1	118.8	0.04	26.12	−9.76	14	91.61	−4.16	26.30	4.56
22	18	103.6	2.42	28	2.67	4	98.89	−0.84	26.74	3.84
23	1	105.7	0.69	24	−4.75	8	110.77	11.12	33.97	10.55
24	1	127.2	1.75	31.15	−25.17	1	103.15	0.62	30.04	−1.8
Avg		103.74	−0.88	24.47	−1.55		99.00	−1.66	25.58	0.70
Sigma		13.12	4.79	5.37	5.58		7.18	6.30	7.09	3.66

Gen#: The generation number that produces the minimum objective function.
F and ΔE: Fitness and total error in percentage, see Equations 1 and 2.
dF and d ΔE: %differences with the fitness and total error from exhaustive search.

Table 6. Numerical Results of Prediction

Township #	1992				2003			
	Fitness %		Total Error %		Fitness %		Total Error %	
	F	dF	ΔE	d ΔE	F	dF	ΔE	d ΔE
1	101.34	−1.06	19.80	−0.07	117.68	1.55	19.76	0.31
2	110.27	−6.4	23.45	−0.22	82.35	0.55	24.99	0.25
3	85.69	−5.97	32.36	0.61	125.86	−3.77	36.87	−2.33
4	97.64	−0.68	29.48	0.21	81.88	−13.81	32.27	−0.38
5	92.26	−0.13	25.89	0.19	105.18	0.11	26.24	0.09
6	82.33	−0.53	28.89	0.44	62.92	−36.48	32.99	−1.37
7	87.45	−13.56	34.66	0.21	84.47	−32.8	32.23	−2.2
8	90.82	0	30.40	0.28	60.43	−19.01	36.86	0.7
9	97.59	−9.81	31.58	−0.18	85.01	−2.94	24.36	0.31
10	113.25	−6.54	24.20	3.04	93.39	4.2	18.24	−0.28
11	111.01	8.13	21.81	2.15	109.12	−0.27	13.25	−0.04
12	89.59	−13.93	36.65	1.79	70.25	−31.91	32.24	3.6
13	87.30	−23.6	26.57	4.29	91.14	−8.33	19.52	1.04
14	97.33	−13.03	16.51	3.75	98.00	−2.09	11.78	0.23
15	113.75	1.52	7.10	1.64	97.61	0	6.50	0
16	103.42	−5.09	27.17	2.83	73.17	−0.01	25.92	0
17	83.56	−13.42	28.60	2.64	98.73	−5.5	19.10	0.82
18	109.63	−4.06	8.11	1.3	99.85	0	8.20	0
19	99.86	−0.76	16.15	−0.7	94.97	−0.03	13.79	0.02
20	110.35	−1.77	29.47	0.99	69.18	0.67	29.13	−0.09
21	75.66	0	27.62	9.99	117.61	−0.48	36.39	13.63
22	114.58	5.93	30.57	9.54	87.11	−0.03	26.05	3.76
23	76.05	−0.64	28.58	0.44	84.81	−19.25	28.36	3.01
24	103.02	−0.13	29.73	0.66	67.67	0	32.69	0.22
Avg	97.24	−4.40	25.64	1.91	89.93	−8.23	24.49	0.89
Sigma	12.13	7.19	7.49	2.76	17.66	12.95	9.13	3.08

F and ΔE: Fitness and total error in percentage, see Equations 1 and 2.
dF and d ΔE: %differences with the fitness and total error from the exhaustive search.

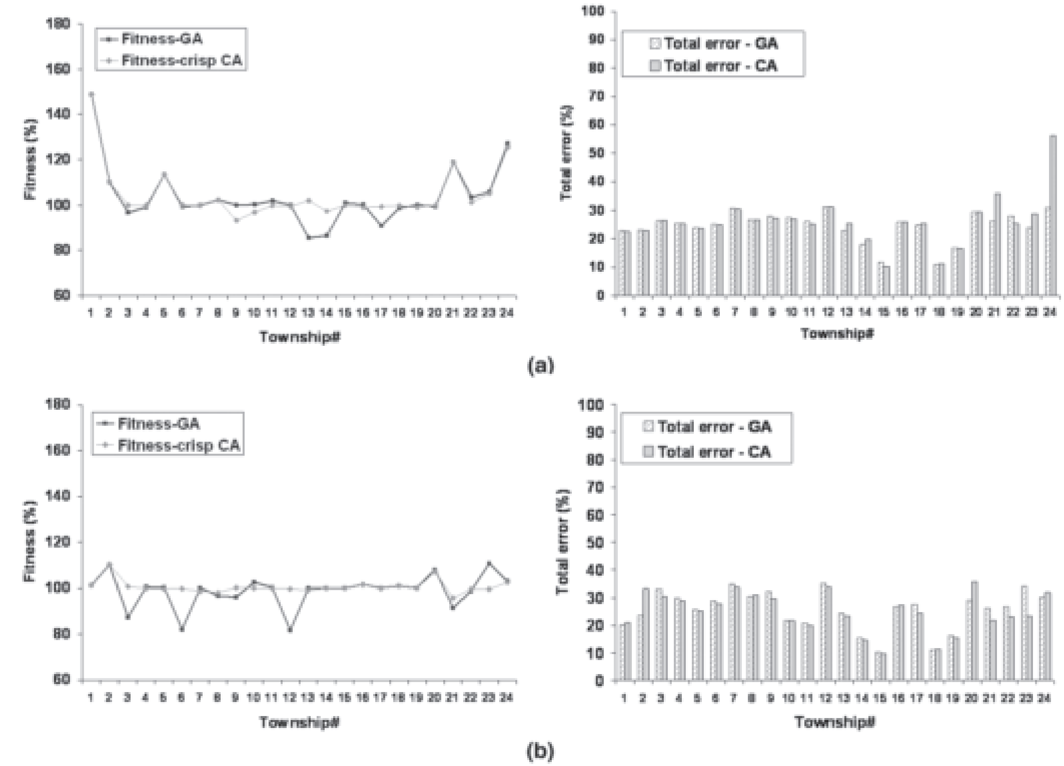

Figure 5. Fitness and total errors from genetic algorithms and exhaustive search for calibration years (a) 1987, and (b) 1992. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

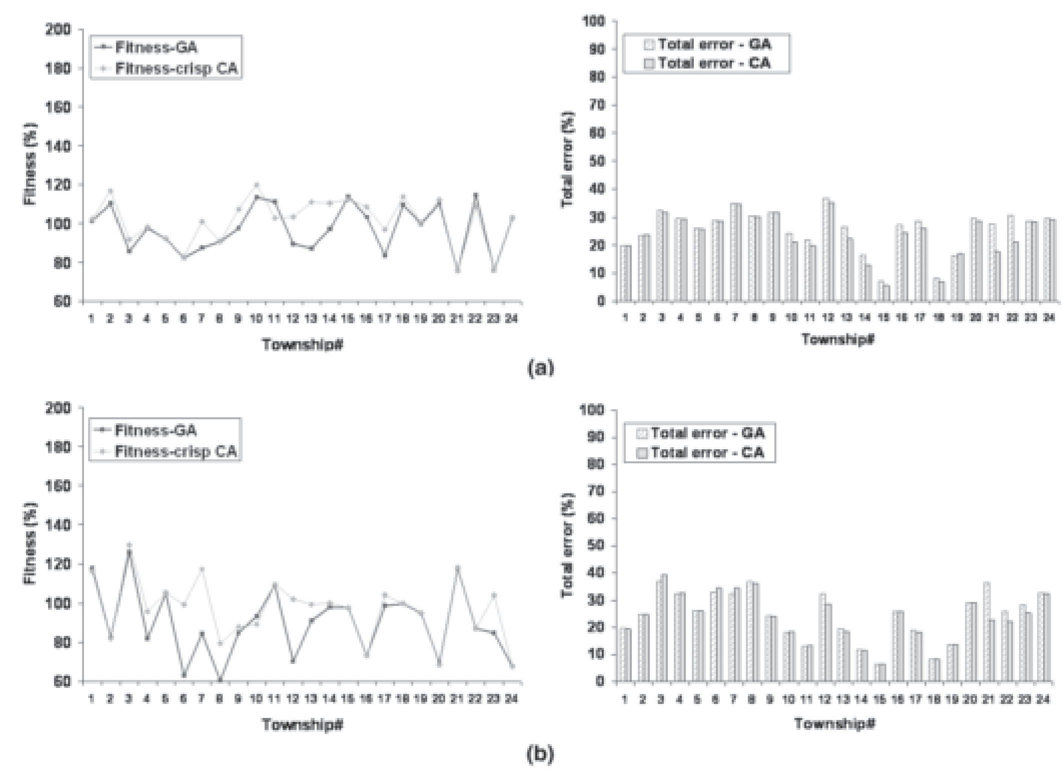

Figure 6. Fitness and total errors from genetic algorithms and exhaustive search for prediction years (a) 1992, and (b) 2003. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

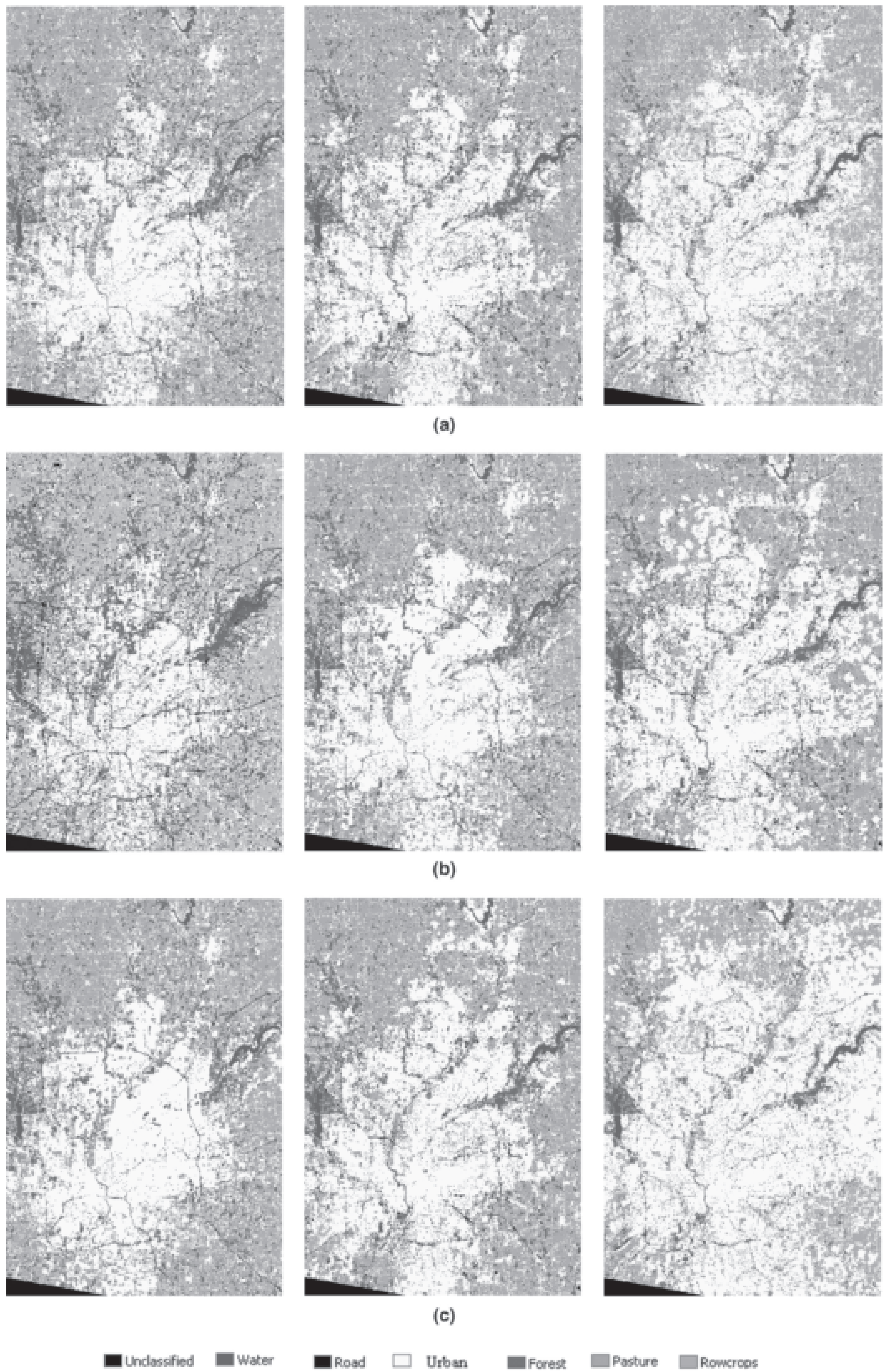

Figure 7. (a) Real city, classified images: 1987, 1992, 2003, (b) Calibrated images: 1987, 1992, 2003, and (c) Predicted images: 1992, 2003, 2010. A color version of this figure is available at the ASPRS website: **www.asprs.org**.

References

Alkheder, S., and J. Shan, 2006. Change detection-Cellular automata method for urban growth modeling, *Proceedings of International Society of Photogrammetry and Remote Sensing Mid-term Symposium*, WG VII/5, The Netherlands.

Anderson, J.R., E.E. Hardy, J.T. Roach, and R.E. Witmer, 1976. *A Land Use and Land Cover Classification System for Use with Remote Sensor Data*, USGS Professional Paper 964, Sioux Falls, South Dakota.

Balling, R.J., J.T. Taber, M.R. Brown, and K. Day, 1999. Multiobjective urban planning using genetic algorithm, *Journal of Urban Planning & Development*, ASCE, 125(2):86–99.

Batty, M., and Y. Xie, 1994a. From cells to cities, *Environment and Planning B*, 21:531–548.

Batty, M., and Y. Xie, 1994b. Modelling inside GIS: Part 2-Selecting and calibrating urban models using ARC-INFO, *International Journal of Geographical Information Systems*, 8:451–470.

Batty, M., Y. Xie, and Z. Sun, 1999. Modelling urban dynamics through GIS-based cellular automata, *Computers, Environment and Urban Systems*, 23:205–233.

Clarke, K.C., S. Hoppen, and L. Gaydos, 1997. A self-modifying cellular automaton model of historical urbanization in the San Francisco Bay area, *Environment and Planning B*, 24:247–261.

Clarke, K.C., and L. Gaydos, 1998. Loose-coupling a cellular automaton model and GIS: Long-term urban growth prediction for San Francisco and Washington/Baltimore, *International Journal of Geographical Information Science*, 12:699–714.

Colonna, A., V. Distefano, S. Lombardo, L. Papini, and G.A. Rabino, 1998. Learning urban cellular automata in a real world: The case study of Rome metropolitan area, *Proceedings of ACRI'98 Third Conference on Cellular Automata for Research and Industry* (S. Bandini, R. Serra, and F. Suggi-Liverani, Editors), Trieste, 07–09 October.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Massachusetts.

Goldstein, N.C., 2003. Brains vs. brawn-Comparative strategies for the calibration of a cellular automata-based urban growth model, *Proceedings of the 7th International Conference on GeoComputation*, 08–10 September, University of Southampton, Southampton, UK, unpagniated CD-ROM.

Holland, J.H., 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan.

Landis, J., and M. Zhang, 1998. The second generation of the California urban futures model, Part 2: Specification and calibration results of the land use change submodel, *Environment and Planning B*, 25:795–842.

Lee, D.R., and G.T. Sallee, 1970. A method of measuring shape, *The Geographical Review*, 60:555–63

Li, X., and A.G.O. Yeh, 2002. Neural network-based cellular automata for simulating multiple land use changes using GIS, *International Journal of Geographical Information Science*, 16(4):323–343.

von Neumann, J.,1966. *Theory of Self-reproducing Automata* (A.W. Burks, editor), University of Illinois Press, Illinois.

Wong, S.C., C.K. Wong, and C.O. Tong, 2001. A parallelized genetic algorithm for the calibration of Lowry model, *Parallel Computing*, 27(12):1523–1536.

Wu, F., 2002. Calibration of stochastic cellular automata: The application to rural-urban land conversions, *International Journal of Geographical Information Science*, 16:795–818.

Wu, F., and C.J. Webster, 1998. Simulation of land development through the integration of cellular automata and multi-criteria evaluation, *Environment and Planning B*, 25:103–126.

Yang, X., and C.P. Lo, 2003. Modelling urban growth and landscape changes in the Atlanta metropolitan area, *International Journal of Geographical Information Science*, 17:463–488.