Cellular automata urban growth model calibration with genetic algorithms

SHARAF AL-KHEDER, JUN WANG, JIE SHAN
Geomatics Engineering
School of Civil Engineering
Purdue University, 550 Stadium Mall Drive, West Lafayette, IN 47907, USA
Phone: +1-765-494-2168, Fax: +1-765-496-1105
jshan@ecn.purdue.edu

Abstract— **Last few decades witness a dramatic increase in city population worldwide associated with excessive urbanization rates. This raises the necessity to understand the dynamics of urban growth process for sustainable distribution of available resources. Cellular automata, an artificial intelligence technique composed of pixels, states, neighborhood and transition rules, is being widely implemented to model the urban growth process due to its ability to fit such complex spatial nature using simple and effective rules. The main objective of our work is to use genetic algorithms to effectively calibrate, i.e., identify transition rule values, a cellular automata urban growth model that is designed as a function of multitemporal satellite imagery and population density. Transition rules in our model identify the required neighborhood urbanization level for a test pixel to develop. Calibration is performed spatially to find best rule values per township. Genetic algorithms calibration model, through proper design of their parameters, including objective function, initial population, selection, crossover and mutation, is prepared to fit the cellular automata model. Genetic algorithms start processing the initial solution space, through sequential implementation of the parameters, to identify the best rule values using a predefined criterion over the maximum number of iterations. Minimum objective function, representing the total modeling errors, is used to identify the optimal rule values. Each rule set is evaluated in term of urban level and pattern match with reality. Calibration with genetic algorithms proves to be effective in producing the optimal rule values in a time effective manner at an early generation. Proposed calibration algorithm is implemented to model the historical urban growth of Indianapolis-IN, USA. Urban growth results show a close match for both urban count and pattern with reality.**

I. INTRODUCTION

Much research efforts can be seen in the literature towards developing effective cellular automata based urban growth models. A number of these models' designs succeed to a certain extent in modeling the urban process. However, there remain unsolved issues to make the developed models more reliable. Calibration of cellular automata urban growth models is among these issues that is still a challenge. Calibration in cellular automata urban modeling is meant to find the best transition rule values to reproduce the same urban level and pattern with reference to historical data [1]. Urban cellular automata models are very sensitive to transition rules and their parameter values [2]. The difficulty in calibrating cellular automata rules is due to the complexity of the urban development process [3]. Calibration styles in literature can be classified into three categories: visual, statistical, and artificial intelligence based. SLEUTH model calibration [4;5] is an example of urban growth model calibration that makes use of visual and statistical tests to identify best urban growth parameter values. Multi-criteria evaluation (MCE) method [6] and neural networks [2] also have been used in previous research for calibration purposes. Common calibration problems of cellular automata are related to the design of model itself. Most models require large input variables and composed of a big set of rules which makes the calibration a time consuming process. Genetic algorithms represent a new calibration direction that appears recently in cellular automata urban growth modeling. The early formal start can be seen in the attempt of formalizing genetic algorithms as a calibration tool for the SLEUTH model [7]. Further improvements are still required to make genetic algorithms a robust technique for urban cellular automata models calibration. Proper design of genetic algorithms, effective setup of their parameters, and selection of the objective function are issues of further interest.

This paper focuses on improving the calibration of a previously designed cellular automata urban model [8] through adapting genetic algorithms. The developed cellular automata model is designed as a function of multitemporal satellite imagery and population density so that the transition rules identify the required neighborhood urbanization level for a test pixel to urbanize. The main objective of the work in this paper is to use genetic algorithms to automate the search method for best transition rule values of the designed cellular automata for reliable modeling. Calibration is performed spatially on a township level to take into account the spatial variation in urban dynamics where the same transition rules are applied to every township, however with different values. Temporal calibration through using historical images to recalibrate the model rules is also performed. The genetic algorithms calibration model is designed to fit the developed cellular automata urban growth model. All the design phases of genetic algorithms including: objective function, initial population preparation and encoding, selection process, crossover and mutation are setup carefully to best reflect the modeling process. Initial population of solution strings are binary encoded with objective function being designed to represent the total modeling errors associated with each string according to the evaluation results identified with reference to reality. Elitism and rank selection procedures are used to start the production of next generation for genetic algorithms. After selection, crossover and mutation operations are implemented on the selected strings to finalize the new population of solutions. Finally, the cellular automata is run for the new population to evaluate their new objective function values. The procedure of running the genetics algorithms with the cellular automata model is repeated till a convergence criterion is met. The rule set that produces the minimum objective function value over all of the iterations is selected as the optimal transition rule for urban modeling. Detailed analysis of the output modeling results is performed as compared to reality. The change in rule values as function of genetic algorithms generation is identified and the pattern of convergence is also tested.

II. Cellular Automata Urban Growth Model

A. Study Area and Input Data

In a previous research work, we developed a cellular automata based urban growth model [8]. The model was tested through simulating and predicting the historical urban growth of city Indianapolis, IN, USA (Fig. 1a). The model uses two types of input data: Historical classified satellite images and population density grids. A set of classified satellite images [9] over Indianapolis (1982, 1987, 1992 and 2003-TM) in NAD1983 UTM projection was prepared. Seven classes were 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. The population density grids were produced using an exponential function (1) of the distance between the census tract centroid and the overall city centroid.

$$POPULATION\ DENSITY = A \cdot e^{-(B \cdot DISTANCE)} \tag{1}$$

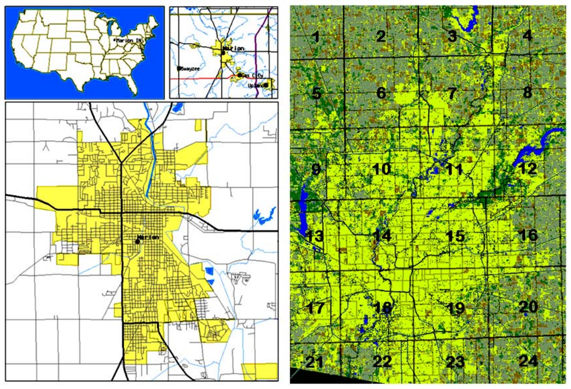
Using the census tract maps at 1990 and 2000, based on which the yearly change in model parameters (A and B) was identified, the model was used to calculate the population density for each pixel throughout the years from 1982 to 2003 to produce the input grids.

B. Transition Rules and Calibraition

Cellular automata transition rules (Ø) of the developed model were physically built over the input imagery. The rules used a 3x3 neighborhood, $A^t_{i,j}$ in (2) to identify the test pixel future state, $a^{t+1}_{i,j}$ in (3).

$$A^t_{i,j} = \begin{bmatrix} a^{(t)}_{i-1,j-1} & a^{(t)}_{i-1,j} & a^{(t)}_{i-1,j+1} \\ a^{(t)}_{i,j-1} & a^{(t)}_{i,j} & a^{(t)}_{i,j+1} \\ a^{(t)}_{i+1,j-1} & a^{(t)}_{i+1,j} & a^{(t)}_{i+1,j+1} \end{bmatrix}_{3x3} \tag{2}$$

$$a^{t+1}_{i,j} = Ø(A^t_{i,j}) \tag{3}$$



a. Indianapolis (US Census Bureau) b. Township map

Figure 1. City of Indianapolis and township map, Indiana, USA

Transition rules (Ø) were designed to identify the required neighborhood urban level for a test pixel to urbanize. The following is a summary of such rules:

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

where $(R,C)_i$ are integer numbers range from 0 to 8 (3x3 neighborhood) and P_i is a real number ranges from 0 to 3 (0.1 increment). The Calibration (i.e., identifying best $(R,C,P)_i$ parameter values) of such rules was performed spatially on a township level, T_s (Fig. 1b) to fit the local urban dynamic features and over time to consider the temporal urban changes at each township, T_t in (4).

$$Ø_{calibrated} = f(T_s, T_t, \Phi) \tag{4}$$

Φ in the calibration formula represents the criteria selected to find the best rule set for certain township spatial location T_s at given time epoch T_t. This criterion in our model represents the total modeling errors/mismatch between modeled output and reality that need to be minimized for best match. Φ in (5) was defined as a function of fitness F in (6) and total errors ΔE in (7) evaluation measures. Fitness and total errors measure the compatibility in terms of urban count and pattern within each township with respect to reality, respectively.

$$\Phi = Abs\ (F\text{-}100\%) + \Delta E \tag{5}$$

$$F = \frac{Modeled\_urban\_count}{Ground\_truth\_urban\_count} \times 100\% \tag{6}$$

$$\Delta E = \frac{Total\_error\_count}{Total\_count} \times 100\% \tag{7}$$

III. Calibration with Genetic Algorithms

Calibrating transition rules (Ø) to find the optimal set $(R,C,P)_i$ for each township among the large number of possible combinations (2511=9*x*9*x*31) is a time consuming process. Genetic algorithms are introduced to effectively identify such parameters in a time effective manner. This section covers first the design phases of the genetic algorithms calibration module as implemented to cellular automata urban growth model. Then a comprehensive analysis is performed to evaluate the modeling results.

A. Design of Genetic Algorithms Calibration Module

Genetic algorithms early starts come through an effort [10] to mimic the natural process in biology of cellular reproduction to solve problems with complex nature. Their ability as an automatic and effective search method for the global optimal solution from a limited and discontinues solution space favors them over other search methods. The development phases of genetic algorithms include objective function design, initial population preparation and encoding, selection, crossover and mutation.

Φ that was defined earlier as the total modeling errors is used as objective function to evaluate the performance for each rule set $(R,C,P)_i$. Rule set with minimum Φ value will be selected as the optimal set. Thirty sets of $(R,C,P)_i$ are randomly generated for each township to represent the total initial genetic algorithms solution population. Each $(R,C,P)_i$ set, that is associated with an objective function value, is binary encoded to represent one string in the solution pool. R_i and C_i are in the possible range of (0-8) integer values and have a binary coding range of (0000 to 1000). P_i continuously ranges between 0 and 3, and is scaled to 0-30 as integer for encoding purposes, which corresponds to the binary coding range of (00000 to 11110). An example of a rule string (7, 7, 1.0) is encoded as a binary string (**0111011100001**), 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. At the end of this phase, a total population of thirty binary strings associated with their objective values for each township is ready to be processed to produce the next genetic algorithms generation.

In the second phase of genetic algorithms calibration, rank and elitism selection methods are used to select the new solution population of thirty strings starting from the initial population. According to rank selection method, all strings are ordered based on their genetic algorithms objective function values in ascending order from minimum to maximum. The string with the lowest objective function value (lowest modeling error) 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 calculated as a ratio between its rank (r_i) and the ranks sum ($\sum r$):

$$p_i = \frac{r_i}{\sum r} \tag{8}$$

This probability when multiplied by the population size (30 strings) will identify how many copies each string is expected to contribute in the next generation. For example, a string with a selection probability of 0.03988 is expected to contribute 1.1964 strings (30x0.03988) in the next population. This means that this string will reproduce one string of its type in the next generation. Through elitism selection, the first six strings with highest ranks, or lowest objective function values, are copied directly to the new solution population. The rest of strings (i.e., 24) are selected from the old population according to their selection probability (rank selection method). By this step a new population of 30 strings is produced as a result of the selection process.

The selected strings are processed further through the crossover operation. Crossover in genetic algorithms simulates the same process in biology whereby genes from two parents meet to produce new offspring that is a mix of the parents' genes. Crossover is important in introducing new possible solutions to explore new areas of search space. The assumption is always that good parent strings, when crossed over, tend to produce offspring with the same or better qualities. Strings produced as a result of the selection process are mated in pairs at random [11]. Each pair of strings in the crossover population (as an output of the selection process) is selected randomly to be crossed over. Using a single-point crossover, an integer location k (4 in this work) between the first and one less than the string length $(l-1)$ is identified as the crossover pivot point. Two new strings are produced by exchanging all the bits between locations k+1 and l inclusively for the two old strings selected for crossover. As an example of single-point crossover, assume that two strings C_1,C_2 (13 bits each as defined earlier) are selected for crossover, where k is set to be 4, the crossover will result in two new strings as follows:

$$C_1 = (x_1, x_2, x_3, x_4 \mid x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13})$$
$$C_2 = (e_1, e_2, e_3, e_4 \mid e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13})$$

$$C_1' = (e_1, e_2, e_3, e_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13})$$
$$C_2' = (x_1, x_2, x_3, x_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13})$$

In our calibration algorithm, the best six strings are copied directly (elitism) and crossed over to produce new 12 strings. To complete the population, the best 18 strings are also crossed over resulting in a new total population of 30 strings. Crossover is performed using the above designed criteria.

The last genetic algorithms operation to produce the new generation of solution space based on the post crossover population is the mutation process. Mutation in genetic algorithms is defined as a random deformation of the strings with a certain probability [12]. Mutation is introduced in genetic algorithms to produce new formations of strings in order to preserve genetic diversity and to avoid local optimum [12]. It simply represents the altering of selected variable bits in the string to enrich the search process with new possible solution combinations representing various sections of the solution space. In the calibration algorithm development, the best six strings within the crossover population are mutated through random addition of +1 or -1 to the $(R,C)_i$ parameter values. The following is an example of string mutation:

0111011100001 (7,7,1) +1,+1 (8,8,1) 1000100000001

By mutation, a new generation of genetic algorithms representing a solution space of thirty strings is generated. Cellular automata model runs to evaluate the new objective function values for the new solution pool.

B. Modeling and Evaluation

The genetic algorithms operations discussed above are repeated recursively for a total of twenty generations. The rule set that produces the minimum objective function value over the course of the total number of iterations (20 iterations) is selected as the optimal for each township.

City of Indianapolis urban growth represented by the set of classified historical images discussed earlier is used as a test bed to test the proposed calibration algorithm. Through designing an initial random solution space (30 strings), the calibration-modeling process is implemented to model (simulation and prediction) the historical urban growth of the city. Urban growth modeling is simulated from 1982 to 1987, where calibration is performed using genetic algorithms to find the best township rule values. The best rules at 1987 are used to predict 1992 for a short term prediction of five years. Another calibration is carried out at 1992 to predict the urban growth at 2003 for a long term prediction of 11 years. Table I refers to part of the numerical evaluation results for simulation year 1987 and predicted year1992 associated with their images at Fig. 2.

TABLE I. NUMERICAL EVALUATION RESULTS

Town-ship #	1987 Simulation		1992 Prediction	
	Fitness%	*Total Error, ΔE %*	*Fitness %*	*Total Error, ΔE %*
1	148.8	22.93	101.34	19.80
2	110.0	23.21	110.27	23.45
3	96.7	26.36	85.69	32.36
4	99.2	25.42	97.64	29.48
5	113.4	23.84	92.26	25.89
6	99.3	25.10	82.33	28.89
7	99.9	30.78	87.45	34.66
8	102.2	26.81	90.82	30.40
9	100.1	28.00	97.59	31.58
10	100.3	27.59	113.25	24.20
11	101.9	26.15	111.01	21.81
12	100.1	31.38	89.59	36.65
13	85.5	22.85	87.30	26.57
14	86.6	18.00	97.33	16.51
15	101.1	11.78	113.75	7.10
16	100.2	25.93	103.42	27.17
17	90.9	24.86	83.56	28.60
18	98.7	10.89	109.63	8.11
19	100.1	16.78	99.86	16.15
20	99.4	29.42	110.35	29.47
21	118.8	26.12	75.66	27.62
22	103.6	28.00	114.58	30.57
23	105.7	24.00	76.05	28.58
24	127.2	31.15	103.02	29.73
Avg.	**103.74**	**24.47**	**97.24**	**25.64**

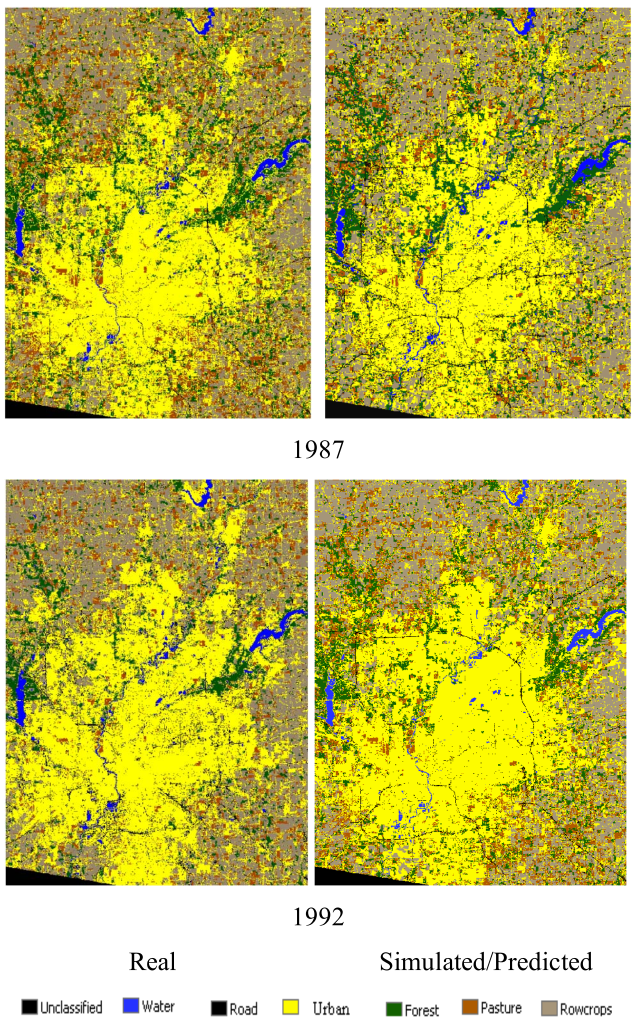
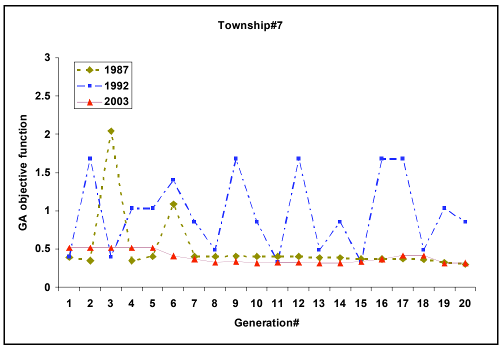


Figure 2. Simulation (1987) and prediction (1992) image results

C. Analysis

Simulation and prediction urban modeling results, as shown in Table I and Fig. 2, show a general close match to reality in term of urban count and pattern. Table I fitness results for both prediction and simulation show close match in urban count (close values to 100%) between the modeled and real data with average fitness of 103.74 (little overestimate) and 97.24 (little underestimate), respectively. The urban pattern match is also clear in the table where an average total error between 24-26% is achieved. This indicates an approximate match level of 75% on a pixel by pixel basis between modeling and reality. This is a high accuracy level compared to the results shown in literature for the urban land spatial fit area that was only 28.15 to 44.6% [13]. The close urban pattern match is also clear in Fig. 2 where both predicted and simulated images have urban distribution similar to those shown in their corresponding real images.

On the side of computation time for transition rule calibration, genetic algorithms show higher efficiency as compared to traditional exhaustive search calibration method. On average, it took genetic algorithms six and half hours of continuous CPU time to run for the twenty generations to reach the optimal rule set per township. This is about ¼ of time needed by the exhaustive search in this study. Looking at the fine scale (township level) regarding the change in objective function shows this fact as well (Fig. 3). As shown in this Figure for selected townships 7 and 14 at different calibration years (1987, 1992 and 2003), the minimum objective function values are achieved at early generations (within the first 10 generations for most townships). This indicates the ability of the proposed calibration algorithm in reaching an optimal solution in a time effective manner while preserving the modeling quality as referred to reality.



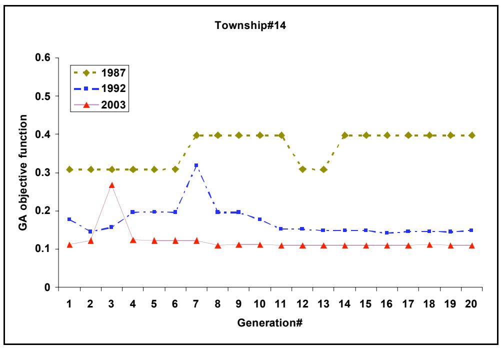


Figure 3. Objective function change with generations

D. Conclusions and Final Comments

As this study demonstrated, genetic algorithms are able to produce modeling results, both quantitatively and qualitatively, close to the reality in a time effective manner. A proper selection of the initial solution population and parameters for encoding, crossover and mutation can enhance the performance while searching for the optimal rule values. It is shown that the computation time is significantly reduced from 27 hours in the traditional exhaustive search to 6.5 hours in the case of genetic algorithms. Optimal rule values for most townships can be reached at an early (<10) stage of genetic algorithm generations. Therefore, it is expected that the genetic algorithms will more significantly benefit urban modeling problems with larger set of input data and larger solution spaces.

There is a need to carry out calibration in spatial units smaller than townships to test the effect of spatial modeling unit size on the reliability of modeling with the purpose of improving the results. For this purpose, it is suggested to use census tracts that represents the smallest spatial units based on which attributes, such as population density, are distributed in spatial calibration. This will help in capturing finer details in the modeling process while calibrating the model over smaller spatial units to reduce modeling uncertainty.

References

[1] F. Wu, “Calibration of stochastic cellular automata: the application to rural-urban land conversions,”. *International Journal of Geographical Information Science*, vol. 16, pp. 795–818, 2002.

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

[3] M. Batty, Y. Xie, and Z. Sun, “Modelling urban dynamics through GIS-based cellular automata,” *Computers, Environment and Urban Systems*, vol. 23, pp. 205–233, 1999.

[4] K. C. Clarke, S. Hoppen, and L. Gaydos, “A self-modifying cellular automaton model of historical urbanization in the San Francisco Bay area,” *Environment and planning B*, vol. 24, pp. 247–261, 1997.

[5] K. C. Clarke and J. Gaydos, “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*, vol. 12, pp. 699–714, 1998.

[6] F. Wu and C. J. Webster, “Simulation of land development through the integration of cellular automata and multi-criteria evaluation,” *Environment and Planning B*, vol. 25, pp. 103–126, 1998.

[7] N. C. Goldstein, “Brains vs. Brawn – comparative strategies for the calibration of a cellular automata – Based Urban Growth Model,” *7th International Conference on GeoComputation*, Southampton, UK, September 2003.

[8] S. Alkheder and J. Shan, “Change detection - cellular automata method for urban growth modeling” *International Society of Photogrammetry and Remote Sensing Mid-term Symposium*, WG VII/5, Netherlands, May 2006.

[9] J. R. Anderson, E. E. Hardy, J. T. Roach, and R. E. Witmer, “A land use and land cover classification system for use with remote sensor data,” USGS Professional Paper 964, Sioux Falls, SD, USA, 1976.

[10] J. H. Holland, “Adaptation in natural and artificial systems,” University of Michigan Press, Ann Arbor, MI, 1975.

[11] D. E. Goldberg, “Genetic algorithms in search, optimization, and machine learning,” Addison-Wesley publisher, MA., USA, 1989.

[12] U. Bodenhofer, “Genetic algorithms: theory and applications,” Lecture notes. Johannes Kepler University in Linz. http://www.flll.uni-linz.ac.at/teaching/ga/ga-notes.pdf. 2004.

[13] X. Yang, and C. P. Lo, “Modelling urban growth and landscape changes in the Atlanta metropolitan area,” *International Journal of Geographical Information Science*, vol. 17, pp. 463–488. 2003.