# Evolving Simulation Modeling: Calibrating SLEUTH Using a Genetic Algorithm

M. D. Clarke-Lauer[1] and Keith. C. Clarke[2]

[1]California State University, Sacramento, 625 Woodside Sierra #2, Sacramento, CA, 95825, USA
Telephone: 1-626-437-1552
Email: clarkelm@ecs.csus.edu

[2]Department of Geography, University of California, Santa Barbara, Santa Barbara CA 93106-4060, USA,
Telephone: 1-805-456-2827
Fax: 1-805-893-2578
Email:kclarke@geog.ucsb.edu

## 1. Introduction

SLEUTH is a simulation model for urban growth and land use changes at geographic scales. The model couples two cellular automata, and uses input data to capture past behavior as parameters during calibration. Calibration uses brute force methods, requiring either long execution times, or parallel computing. We describe the implementation of a genetic algorithm (GA) that reduced calibration time and enhanced model accuracy. While the model has been successfully applied worldwide (Clarke et al. 2007), computation time remains an obstacle to effective calibration. By designing a GA to work in conjunction with SLEUTH, the computation time was reduced by 80%, while the accuracy was improved.

## 2. SLEUTH

SLEUTH uses two complex cellular automata operating on a geographic region represented by a two-dimensional cellular grid. Every cell can perform a transition to another state, directed by a transition function and the values in adjacent cells (Clarke, et al. 1997). Cellular automata models have revolutionized urban modeling (Torrens and O'Sullivan, 2001), and are used to simulate various natural and man-made phenomena.

SLEUTH simulates urban growth and land use dynamics when calibrated with a set of mapped data reflecting past patterns. A sequence of growth rules is applied to the cells, each controlled by a set of coefficients that encapsulate the dynamics of a region (diffusion, breed, spread, road gravity, and slope). These values are not initially known and require extensive calibration to determine. SLEUTH's code automates the calibration process, nevertheless the user is still required to guide the calibration phases (Silva and Clarke, 2005). The model provides thirteen metrics describing the fit of the calibration coefficients, with the best set being selected for forecasting. The Optimal Sleuth Metric, a product of eight of the metrics, is best for optimizing calibration (Dietzel and Clarke 2007).

Implementation requires calibration (determining the best coefficients) and predicting (modeling into the future). The calibration phase simulates historical change and compares it to known data to determine how accurately the model simulates growth (Jantz et al., 2010). SLEUTH repeatedly applies sets of the five coefficients to determine which yields the highest OSM. Coefficients consist of numbers between 0 and 100, the

entire search space constituting $101^5$ coefficient sets. The brute force approach performs three passes through the search space, with each run the search granularity gets smaller. Two potential problems emerge, the first being the sheer computation time required. Each of the three calibration phases requires at least 2,000-10,000 iterations. Monte Carlo methods minimize within-run variability but further increase computation time. A recent application required over 6 months of CPU time.

The phased and stepped brute force approach may become unable to break free from a local optimum, since large areas of the search space are eliminated from the solution domain. Using a GA in the calibration addresses both problems. By allowing the values to be randomly, but evenly, distributed throughout the search space and by encouraging the best solutions to survive, both speed and accuracy can be improved. We applied the GA at the code level by replacing the source code that implements the brute force calibration. SLEUTH's modularity means that only the driver level function needed alteration, all of the model behavior modules remained unaffected.

## 3. GA Design

GAs simulate biological evolution and natural selection among a set of possible solutions, and can produce an optimal or near optimal solution. SLEUTH uses a bounded five dimensional search where the model metrics can direct the search. The five dimensions are the integer values of the five model behavior parameters, and the metrics reduce to the OSM.

The application of GA to SLEUTH was first achieved by Goldstein (2004) using both elitism and tournament selection, and combining gene competition strategies (stratified, partial random, and random). Crossover employed both uniform and self-crossover, and mutation used a 10% randomization. The approach was tested for Sioux Falls, South Dakota over 200 generations, with 18 chromosomes in each run, but for only one Monte Carlo iteration, with the calibration repeated 10 times. Results showed that 70% of the chromosomes outperformed brute force yet used one fifth as much CPU time, giving better goodness of fit measures. Nevertheless, there was evidence that the GA became stuck in local maxima, and some optimization ambiguity as the work predated the OSM, and so compared different metrics. While the GA was only simulated (separately generating the parameters, that were fed to independent runs across 10 computers), Goldstein did explore the consequences of sub-optimal calibrations for model forecasting, but not which gene selection, cross-over and mutation strategies worked best. Our approach first tested possible strategies, and then hard coded a single strategy into the SLEUTH source code driver module.

The GA for SLEUTH calibration was designed based on Goldstein's findings and prior GA research including choices on encoding, fitness evaluation, crossover, mutation, and survival selection (Eiben and Smith 2003). The model provides a natural encoding, each gene is represented as a set of five integer coefficients in the range {0,0,0,0,0} to {100, 100,100,100,100}. Each coefficient represents a separate piece of genetic material for a specific gene, with all five combined composing the entire composition of a gene. When running the model with the five coefficients, the OSM metric provided creates a natural fitness evaluation for an individual gene. Crossover, the process of combining existing genes to create new genes, takes a subset of the coefficients from one gene and

combines them with the opposite subset from the other, which was simpler than Goldstein's method (2004). This was performed by using a random number between 0 and 4 and using that value to decide how many elements from the first parent are used to create the offspring. The remaining elements were provided by the second parent. A second offspring was produced from the opposite elements that created the first offspring. Parents were selected using tournament selection, a random subset of the population is chosen and two selected using probabilities proportional to fitness. Mutation replaces coefficients within a gene with a random value at the mutation rate frequency to maintain diversity. The mutation rate was provided as an input to the genetic algorithm and can be tuned based on the performance of the model. Lastly, survival selection is the method for selecting a subset of the population and its offspring for the next generation. Each generation replaces the weakest genes in the old population with the strongest of the offspring, until at least half of the population is replaced and there are no old population genes that are weaker than any remaining offspring. Elitism prevents the fitness from regressing during the calibration.

The GA was first tested to determine population size and mutation rate. Testing used 2,000 iterations through the model per run of the GA. Using the Demo_City test data provided with SLEUTH showed that neither a low nor high mutation rate was ideal, but within the range 0.10 to 0.16 was satisfactory (Figure 1). Results showed that population sizes between 15 and 30 were good choices (Figure 2). A population size of 25 and a mutation rate of 0.16 were chosen. While 15 showed the strongest fitness, a population size must be sufficiently large to maintain genetic diversity.

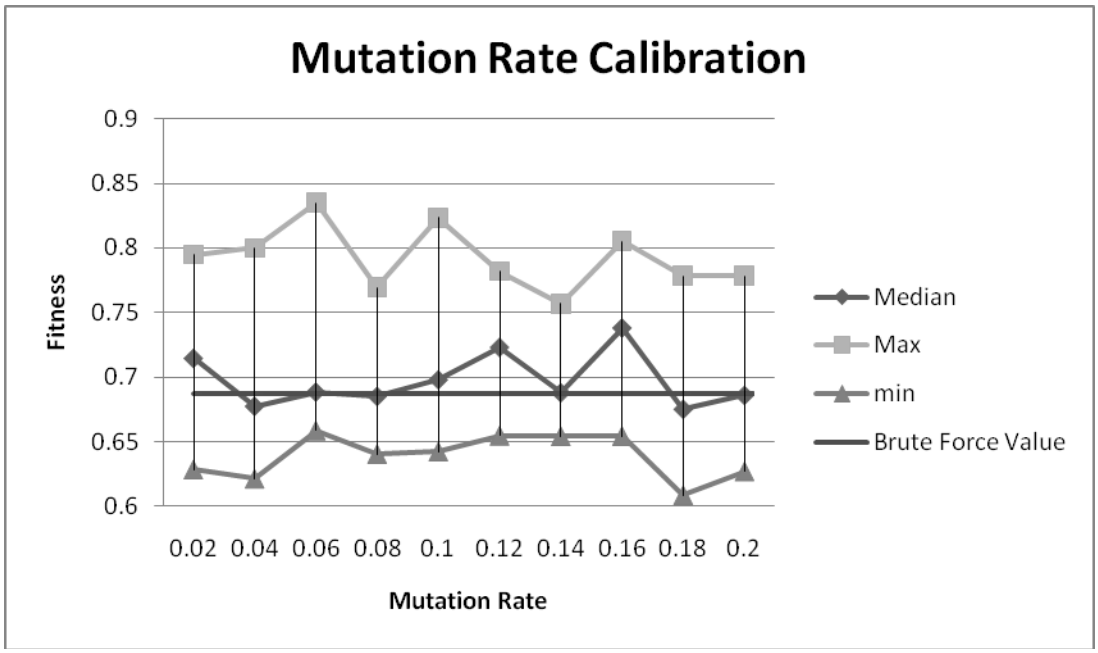


Figure 1. Results of GA test: Mutation Rate

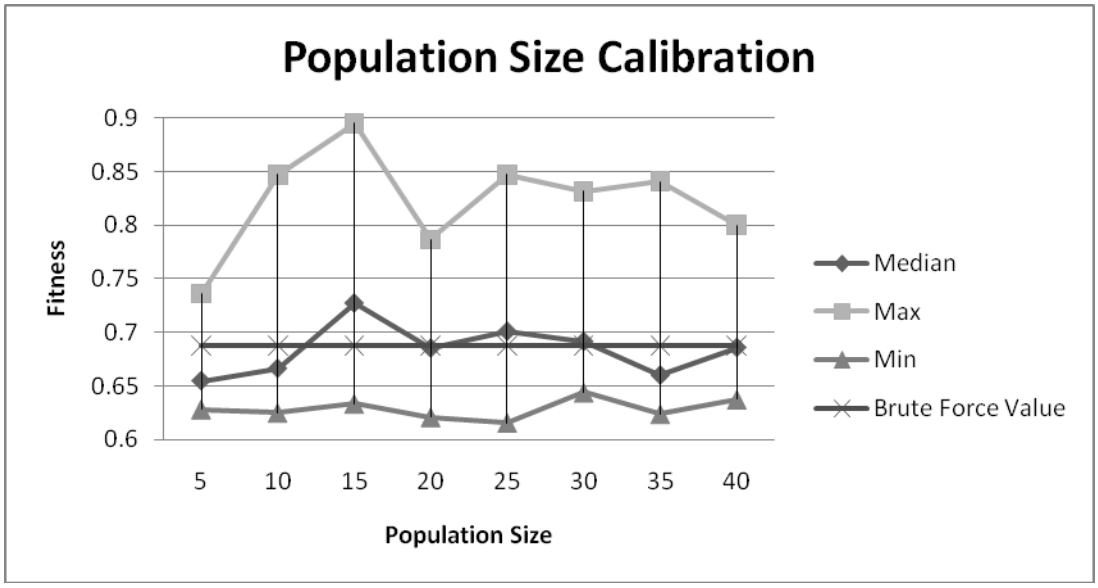


Figure 2. Results of GA test: Population Size

## 4. Results

The GA was used in SLEUTH and the results compared to the brute force method as applied to Demo_City. OSM values obtained were similar to those achieved in other SLEUTH applications (Table 1).

| *Statistic* | *Fitness (OSM)* | % Improvement |
|---|---|---|
| Mean | 0.705013 | 3% |
| Median | 0.697704 | 2% |
| Standard Deviation | 0.051764 | -- |
| Minimum | 0.620926 | -10% |
| Maximum | 0.870902 | 27% |
| Brute Force Calibration | 0.687381 | -- |

*Table 1: Genetic Algorithm Calibration Results*

The GA on average performed slightly better than brute force. Due to the stochastic nature of a GA, there were rounds where it performed up to 10% worse or 27% better than brute force. As with Goldstein's test, the model was calibrated using a single Monte Carlo iteration to reduce computation time and allow for rapid evolution in the GA.

## 5. Conclusions

Results showed the GA can maintain or improve the fit of SLEUTH. While the median solution was a small improvement, performance boost varied from -10% to 27%. Yet the real value of GA is in reducing computation time, where it outperforms brute force

calibration by a factor of 5, without subjective input. This speed-up was also achieved by Goldstein (2004), and may be further improvable by experiment. We capped the GA at 2,000 generations, while the brute force required a minimum of ~10,000 iterations. On an Intel XEON 5570 CPU one run of the GA was completed in ~30 minutes and eight runs could be performed simultaneously per CPU without taxing the server. This would allow model calibration in hours, compared to weeks with brute force. Such a saving would permit calibration sensitivity tests not feasible otherwise. The SLEUTH code used in this research was posted to the SourceForge open source site (https://sourceforge.net/projects/sleuth-ga/).

Future improvements can be made to the GA through algorithm optimization and parallelization. These would increase the efficiency of the GA further improving speed, and reducing calibration to minutes. Such times would overcome one of the last remaining obstacles to SLEUTH's application in urban planning and land management (Clarke, 2008). Furthermore, it is a good example of geocomputation, where computer science optimization methods (GA) meet simulation modelling in geography.

## 6. References

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

Clarke, K. C, Gazulis, N, Dietzel, C. K. and Goldstein, N. C. 2007. A decade of SLEUTHing: Lessons learned from applications of a cellular automaton land use change model. Chapter 16 in Fisher, P. (ed) *Classics from IJGIS. Twenty Years of the International Journal of Geographical Information Systems and Science*. Taylor and Francis, CRC. Boca Raton, FL. pp. 413-425.

Clarke, K.C. 2008. A decade of cellular urban modeling with SLEUTH: Unresolved issues and problems, Ch. 3 in *Planning Support Systems for Cities and Regions* (Ed. Brail, R. K., Lincoln Institute of Land Policy, Cambridge, MA, pp 47-60.

Dietzel, C. and Clarke, K.C. 2007, Toward Optimal Calibration of the SLEUTH Land Use Change Model. *Transactions in GIS 11*(1): 29-45.

Eiben, A. E. and Smith, J. E. 2003, *Introduction to Evolutionary Computing*, Springer-Verlag, Berlin.

Goldstein, N. C. 2004. *Brains vs. Brawn: Comparative strategies for the calibration of a cellular automata–based urban growth model*. Chapter 18 in Atkinson, P., Foody, G., Darby, S., and Wu, F. (eds) GeoDynamics. Boca Raton, FL, CRC Press.

Jantz, C. A., Goetz, S. J., Donato, D. and Claggett, P. 2010, Designing and implementing a regional urban modeling system using the SLEUTH cellular urban model. *Computers, Environment and Urban Systems*, 34, 1-16.

Silva, E. A. and Clarke, K., (2005) Complexity, emergence and cellular urban models: Lessons learned from appling SLEUTH to two Portuguese metropolitan areas. *European Planning Studies*, vol. 13, no. 1, pp. 93-115.

Torrens, P. M. & O' Sullivan, D. 2001. Cellular automata and urban simulation: where do we go from here? Environment and Planning B, 28, 163-168.

US Geological Survey. 2007, Project Gigalopolis: Urban and Land Cover Modeling. http://www.ncgia.ucsb.edu/projects/gig/index.html