2014

Genetic Programming For Cellular Automata Urban Inundation Modelling

Mike J. Gibson

Edward C. Keedwell

Dragan A. Savić

11th International Conference on Hydroinformatics
HIC 2014, New York City, USA

GENETIC PROGRAMMING FOR CELLULAR AUTOMATA URBAN INUNDATION MODELLING

M J GIBSON (1), E C KEEDWELL (1), D SAVIĆ (1)
(1): Centre for Water Systems, University of Exeter, Harrison Building, North Park Road, Exeter EX4 4QF, UK

Recent advances in Cellular Automata (CA) represent a new, computationally efficient method of simulating flooding in urban areas. A number of recent publications in this field have shown that CAs can be much more computationally efficient than methods that use standard shallow water equations (Saint Venant/Navier-Stokes equations). CAs operate using local state-transition rules that determine the progression of the flow from one cell in the grid to another cell, and in a number of publications the Manning's Formula is used as a simplified local state transition rule. Through the distributed interactions of the CA, computationally simplified urban flooding can be simulated, although these methods are limited by the approximation represented by the Manning's formula.

An alternative approach is to learn the state transition rule using an artificial intelligence approach. One such approach is Genetic Programming (GP) that has the potential to be used to optimise state transition rules to maximise accuracy and minimise computation time. In this paper we present some preliminary findings on the use of genetic programming (GP) for deriving these rules automatically. The experimentation compares GP-derived rules with human created solutions based on the Manning's formula and findings indicate that the GP rules can improve on these approaches.

INTRODUCTION

Flood modelling is a key method for such techniques as hazard mapping, uncertainty analysis, climate change scenarios, and catchment enhancement design such that potential damage can be averted. Unfortunately such models can be very computationally expensive, and with large (or very high resolution) catchments, the demand for computationally efficient methods is great. These standard methods are typically based on a form of grid/mesh covering the target area where each grid cell contains a terrain elevation level and a water level among other data. Typical physically-based flood models solve Shallow Water Equations (SWEs), which are computationally expensive, and recent literature shows efforts to find more computationally efficient methods often by attempting to reduce the complexity of terms within the SWEs [1]. Examples of the flood modelling approach include the JFLOW model [2], Urban inundation Model (UIM) [3], and the diffusion version of LISFLOOD-FP [4]. Other models use the full SWEs but focus on the use of multi resolution grids or irregular mesh grids and rely on highly optimised code to achieve high performance, like InfoWorks ICM [5] and Mike 21 [6] [7].

These last two models are commercial software packages, and the code applied in the optimisation techniques are not in the public domain.

An alternative to these approaches that has recently been proposed is the use of cellular automata in this domain. Cellular Automata, initially developed in the 1950s [8] represent a computationally efficient method, which consist of a lattice of cells , whereby a state-transition rule using only local neighbour states is used to calculate the state change in the automaton. From a flood modelling perspective, the state of the cell represents the water depth in each cell, and the state transition rules calculate the amount of flow between cells given the water and terrain levels of itself and its neighbouring cells from the previous time step in the simulation. The other major factors used within each given grid cell for the calculation of the next water level are the Manning's terrain roughness factor, and the cell size in metres. The objective is to use such local interactions, which are simple to process, and yet are capable of creating far more complex emergent behaviours, as has been shown in many other application areas including: John. H Conway's famous 'Game of Life' [8], simulating forest fire [9], and chemical reaction-diffusion [10], to name but a few. The main driver for using the CA approach is to improve computational efficiency whilst maintaining or perhaps slightly improving on the accuracy of the standard SWEs.

Work by Dottori & Todini [11], demonstrates how an open channel flow rate formula (Manning's formula) may be used as a good approximation of the local transition rule, within limitations of the cell size and time step. Both Dottori & Todini [11] and Hunter, et al. [1] describe a checkerboard pattern in execution when either the cell size is too small, or time step is too large when oscillations can occur. Other results have supported this finding and a number of different state-transition rules have been proposed that deliver improved accuracy and stability in a cellular automaton simulation. Clearly the state-transition rule is a key component of the cellular approach and should be the focus of experimentation to improve CA performance. Manning's formula has proved effective but its instability makes it difficult to use in all circumstances. The underlying reason for this is that the Manning's formula is based on the flow of an open channel (e.g. a river), and using it in both 2D of cells which represent small basins (the area covered by the cell), and fails to capture all the dynamics including momentum and energy conservation.

The literature demonstrates a number of techniques for limiting either flows rates, or conversely time step taken, in order to avoid the limits of Manning's formula as the approximation of the local state transition rule. All these approaches view this local rule as static, and attempt to work around its flaws and limitations whereas an alternative approach is to view the state transition rule as a possible candidate for change and update. In particular this paper addresses the question of whether the rules can be learned from simulation data by using artificial intelligence techniques. In this work we investigate the possibility of using an evolutionary computing technique, called Genetic Programming (GP) Koza [12] and Banshaf, et al. [13] to create new state transition rule for CA modelling flood inundations. In the field of both computer science and hydrology a wide variety of projects use evolutionary/genetic algorithms, in order to perform optimisation. GP extends these powerful search and optimisation algorithms, from a fixed number of decision parameter to the creation and optimisation of an entire computer program (or mathematical function) of variable size and shape. This technique has been used on simple symbolic regression tasks by White, et al. [14], where often thousands of training points are required, for a spread of decision variables values, in order to establish a mean fitness. Since the state transition rule of a CA is instantiated for multiple cells, and then for multiple iterations/time steps, this represents an ideal candidate for

the training of GP. We use the output on a simple training terrain of the Urban Inundation Model [3], with a simple inflow hydrograph, as the target for the combined Genetic Programming Cellular Automaton (GPCA) system. I.e. it is the global emergent behaviour which is trained upon, in order to try and learn the underlying local state transition rule.

This paper assesses the capability of the GP to derive accurate and fast transition rules that can compete with Manning's formula in flood modelling. Although computationally complex to train, once GP-derived formula are created, these should generalise to other flood modelling examples outside of the training set. In the following experimentation we perform two main batches of tests, the first where the GPCA system has a minimum hydraulic slope explicitly preprogrammed (i.e. the GP does not need learn this); secondly we also attempt to learn the Manning's formula and the minimum hydraulic slope.

METHOD

Cellular Automata System

The change of each cell state is based upon the current cell state and its neighbouring cells states, from the previous iteration. For the purposes of this paper the Von Neumann neighbourhood is employed (Shown in Figure 1).

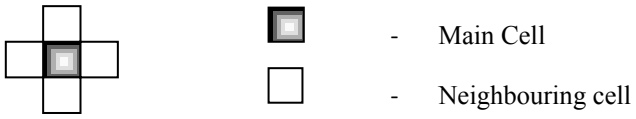

Figure 1. The Von Neuman Neighbourhood.

The state transition rule is broken down in to the four possible pairs of cells within each neighbourhood. For each pairing of cells, if the neighbouring cells water level is lower (i.e. only for possible outflows from the main cell), then the given GP individual is used to calculate the amount of water outflow. Once this is performed for each pairing of cells within each neighbourhood, then outflows will have been written for every cell pairing with a water level difference; this is done to avoid excessive processing of each pair of cells twice (i.e. in each direction) (Shown in Figure 2, stage 1). Once all inter-cell fluxes are calculated then, each cell subtracts the outflows to its neighbouring cells from its current water level, and adds any inflows from neighbouring cells (Shown in Figure 2, stage 2). Through the combination of applying the given function in all four directions for each cell, in each iteration, each cell calculates a new water level for the next iteration. Readers are directed to (Wolfram, 1984) for a detailed description of the cellular automaton approach used here.

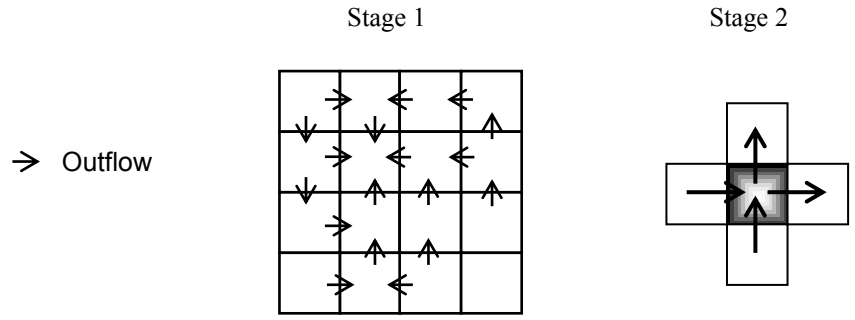

Figure 2. Two stages to each CA iteration; Stage 1 for every cell pairing with a water level difference a flow is calculate. Stage 2 each cell updates its new water depth by means of subtracting outflow and adds any inflows.

Genetic Programming System

Genetic programming is based on work by Koza [12], which evolves solutions as tree structures rather than the more common linear chromosome seen in genetic algorithms. Nodes on the tree are separated into two groups, functions and terminals. Functional node operations include the simple mathematical function plus, minus, protected division (whereby division by zero returns zero), multiplication, and a heavily protected power function. Logic operators include equality, greater than, less than, and a triple branched if-statement (first branch conditional, second if true, third if false). Each GP tree has available to it the following terminals (as these are different for each cell pairing): the water level and depth, and terrain level for both the main and neighbouring cell; also the cell size, terrain roughness factor, and time step (totaling 9 variable values). Generated GP trees also have available constant integer value from 0 to 10 inclusive.

The GP algorithm works in a similar way to other evolutionary algorithms, with the generation of a random initial population of potential individuals, which are then evaluated against a given target to obtain an error value for each. In the case of the GPCA system, a simple CA simulation is run for each GP individual; where the GP instantiated between each cell pairing, at each time step (as described in the CA method section), and the error value is calculated between target simulation of each GPCA simulation. The population of GP individuals is sorted by its fitness level (1/error), and the top proportion of individuals is copied directly into the next generation (known as elitism). In order to fill the rest of the new population, individuals are selected from the old population (given the selection schema, e.g. fitness proportion roulette, or tournament). Once an individual is selected, it has a small chance of going straight through to the new population unchanged, and a small chance of being mutated slightly before copying to the new population, and finally the greatest chance of being crossed-over with another selected individual. During both cross-over and mutation, a minimum depth is placed upon tree growth, as unbounded GP is prone to 'bloat' [13].

Mutation operators allow for the change of one node to a different operation, variable or constant term, or the insertion or removal of a given node. It is also possible to grow a new sub-tree to a given minimum depth. The most powerful operator in GP is the cross-over, which given two parent GP trees, selects at random a node and its sub-tree and transplants it with another random node and sub-tree in the other parent. Both operators are restricted by the minimum global depth of the GP trees.

Genetically Programed Cellular Automata System

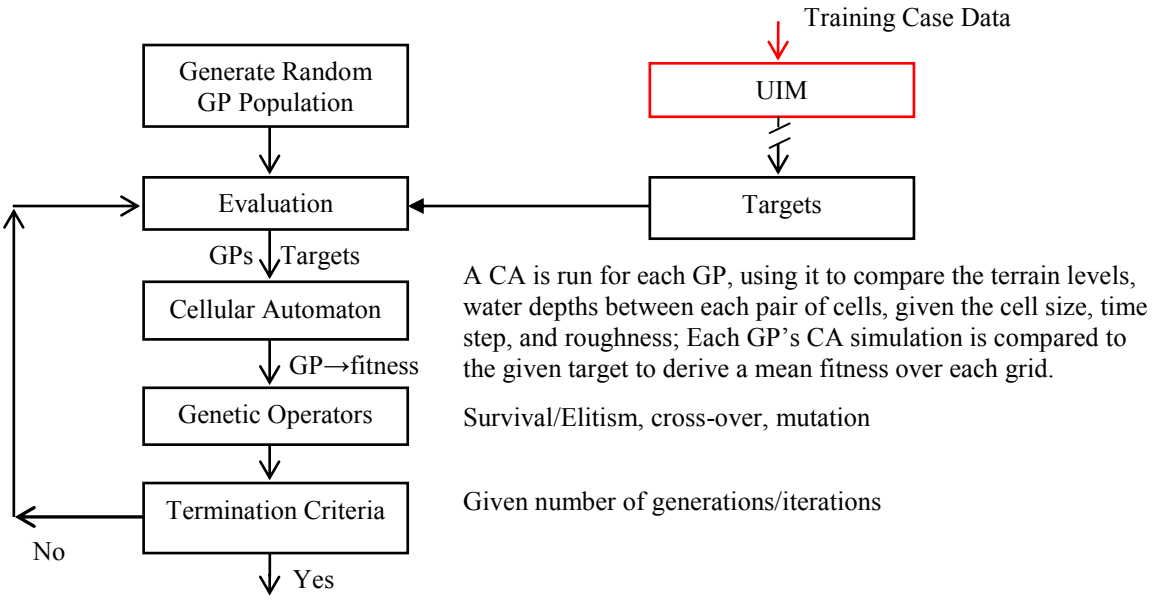

Figure 3. Genetically Programmed Cellular Automata system overview.

EXPERIMENTAL SET-UP

Training is conducted upon a small hypothetical terrain case (20x30 cells), which has a slight slope in both dimensions, as well as a pond and crest area two-thirds along the longest dimension. A uniform hydrograph is used as inflow for all cells in the first hour of the simulation, of 20mm/hr. A uniform roughness factor of 0.01 is applied to all cells, which start dry, and the simulation is run for 4 hours to allow water level to rise and fall at the key points in the terrain. The border condition reflects inwards for each border cell, apart from a single out flow cell at the lowest corner of the slopes, from which 10mm is subtracted from the external border cell. This simulation has been run in UIM [3] to produce a target water depth for the grid second of the simulation, which is recorded to disk. Cell sizes of 50m, 25m and 2m are processed as potential targets.

The GPCA system generates a random initial population, with the following parameters (show in table 1); which were discovered to be reasonably effective from limited testing. A static 1 second time step is used for each CA iteration, and processed for 240 minutes of simulation time at cell sizes of 50m and 25m. The RMSE of each cell in each time step is compared against the target simulation.

If the minimum water depth of 0.001 is not met then no out flow is processed; also if outflows exceed the main cells current water depth, they are normalised by this amount; finally a cell may not accept more inflow than water depth is present of all of its neighbouring cells.

Table 1. Genetic Programming Parameters

Initial GP tree depth	3
Growth type	Full
Maximum depth	10
Mutation chance	0.1
Cross-over chance	0.9
Population size	100
Generations	100 -1,000
Selection Type	Tournament (size 10)

The Manning's Formula

In order to verify the system and to provide a human benchmark for the GPCA system, a GP tree has been manually configured, using the Manning's formula to calculate the flow rate; which is then simplified with the discharge and water depth change formula (Shown in Figure 4). Finally a minimum slope is used to smooth out the simulation, which also minimises the small error compared to the UIM simulation.

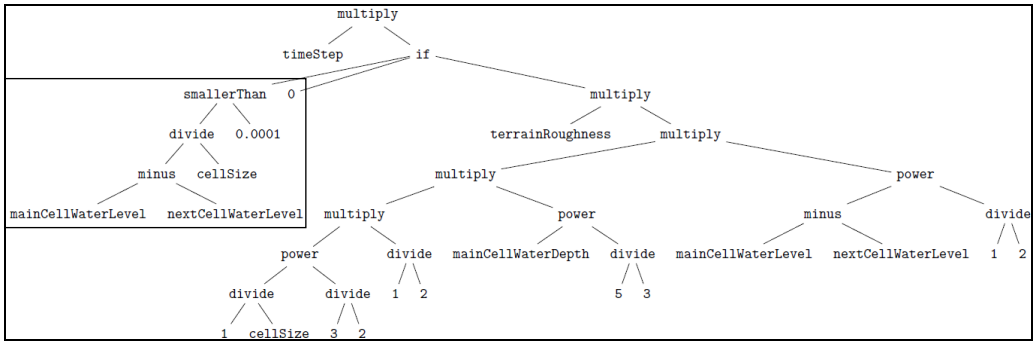

Figure 4. Human programmed GP tree, derived from the combing the Manning's, discharge and water depth change formula, and finally a minimum hydraulic slope which is highlighted.

RESULTS

The minimum hydraulic slope has proved difficult to discover by the GP during initial training runs (shown in figure 5). It is thought that this is due to fact that this occurs infrequently during the simulation, and because the outflow so abruptly drops off; thus not following the general pattern of the rest of the simulated cells, from which the GPCA system is learning. Experimentation has been conducted with the highlighted minimum slope sub tree, both explicitly pre-programmed, and without; which demonstrates this (shown in figure 5). None of the GP individuals managed to exceed the fitness of the Manning's formula (Figure 4), when learning both the Manning's formula section and the minimum hydraulic slope (implicit). However a number of GP individuals did manage to produce smooth simulations beyond that of the Manning's formula with only a small disagreement with the target. It is noted that the target in this paper [3], is known to use the Manning's formula, a minimum slope, and other formula such as a critical flow rate. It may be possible with larger amounts of training data, or even real-world data, to train CA rules more which follow the pattern of multiple formulas with only minor disagreement with the global pattern.

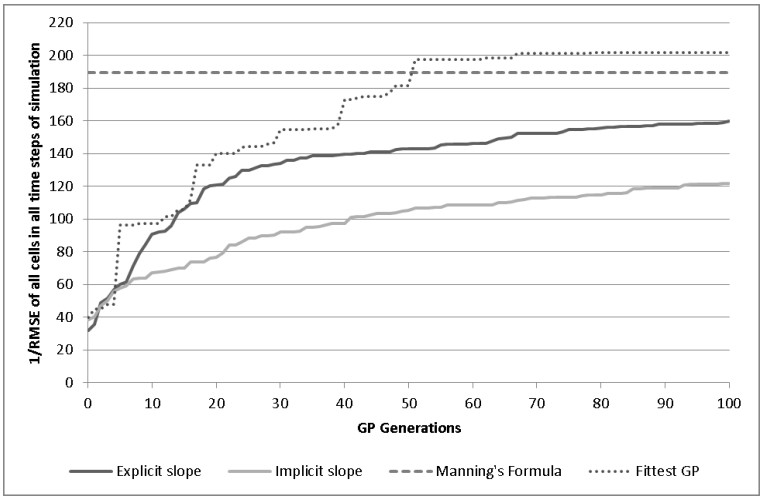

Figure 5. Mean fittest individuals of 10 GP populations, with the minimum slope both explicitly pre-programmed, and without; also shown is the fitness of the Manning's formula (shown in figure 4).

Figure 6, demonstrates a clear difference in performance of the GP where those with the minimum hydraulic slope explicitly pre-programmed perform significantly better. Also shown is a GPCA run which has managed to exceed to fitness of the human derived GP tree (Shown in Figure 4).

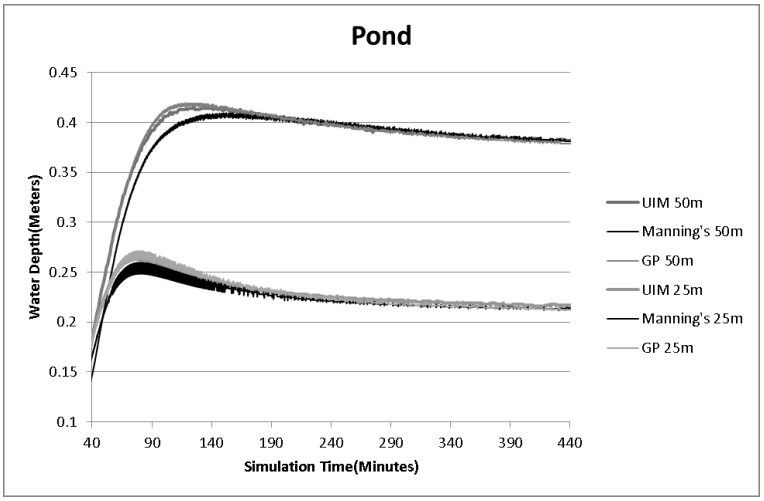

Figure 6. Ponding point in training simulation water depths, for UIM, Manning's (Figure 4), and fittest GP (Figure 5); for 50m and 25m simulations.

DISCUSSION

Figure 6 shows how a simulation run with the GP can produce stable results even after the training period of the simulation of 240 minutes. With expanded training it should be possible to create even more generalist rules; where current training may take in the order of hours to days to complete, once trained, a simulation is processed in seconds. Where the above

experiments are conducted with a static time step of one second, it may be possible to both create rules which are faster than current rules, and are capable of operating where the use of the Manning's formula alone begins to break down. Experiments have been conducted with a 2m cell size, and one second time step, where the Manning's formula breaks down; whereas the GPCA system is capable of producing rules which produce smooth simulations with only minor disagreement with the target, and which takes approximately an order of magnitude longer to process.

Acknowledgments

The authors wish to Thank Dr. Michele Guidolin for his help with use of the CADDIES system, also Dr. Albert Chen for his knowledge of UIM and hydrology.

REFERENCES

[1] N. M. Hunter, P. D. Bates, M. S. Horritt and M. D. Wilson, "Simple spatially-distributed models for predicting flood inundation: A review," Geomorphology, vol. 90, no. 3-4, pp. 208-225, 2007.

[2] K. F. Bradbrook, S. N. Lane, S. G. Waller and P. D. Bates, "Two dimensional diffusion wave modelling of flood inundation using simplified channel representation," Internation Journal of River Basin Mangement, vol. 2, no. 3, pp. 211-223, 2004.

[3] A. S. Chen, S. Djordjević, J. Leandro and D. Savić, "The urban inuncation model with bidirectional flow interactions between 2D overland surface and 1D sewer networks," NOVATECH, pp. 465-472, 2007.

[4] P. D. Bates and A. P. J. De Roo, "A simple raster-based model for flood inundation simulation," Journal of Hydrology, vol. 236, no. 1-2, pp. 54-77, 2000.

[5] Innovyze, "InfoWorks ICM," 2014. [Online]. Available: http://www.innovyze.com/products/infoworks_icm/. [Accessed 13 03 2014].

[6] DHI Software, "MIKE 21," 2014. [Online]. Available: http://www.dhisoftware.com/Products/CoastAndSea/MIKE21.aspx. [Accessed 13 03 2014].

[7] J. Hénonin, H. Ma, Z.-Y. Yang, J. Hartnack, K. Havnø, P. Gourbesvile and O. Mark, "Citywide multi-grid urban flood modelling: the July 2012 flood in Beijing," Urban Water Journal, pp. 1-15, 2013.

[8] S. Wolfram, "Cellular automata as models of complexity," in Nonlinear Physics for beginners, vol. 311, 1998, p. 197.

[9] L. Karafyllidis and A. Thanailakis, "A model for predicting forest fire spreading using cellular automata," Ecological Modelling, vol. 99, no. 1, pp. 57-97, 1997.

[10] E. Roberts, J. E. Stone, L. Sepúlveda, W.-M. W. Hwu and Z. Luthey-Schulten, "Long time-scale simulations of in vivo diffusion using GPU hardware," Rome, 2009.

[11] F. Dottori and E. Todini, "A 2D Flood Inundation Model Based on Cellular Automata Approach," in Internation Conference on Water Resources (CMWR), Barcelona, 2010.

[12] J. R. Koza, Genetic Programming: on the programming of computers by means of natural selection, 1992.

[13] W. Banshaf, P. Nordin, R. E. Keller and F. D. Francone, Genetic Programming: An Introduction, 1998.

[14] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly and S. Luke, "Better GP benchmarks: community survey results and proposals," Genetic Programming and Evolvable Machines, vol. 14, pp. 3-29, 2013.