Modeling Population Generation Behaviour   
using Multi-Variable CD++

**ABSTRACT**

This paper displays a model for creating population conduct utilizing cellular automaton .Multiple state variables for every cell are utilized to represent change from youngster to a grown-up and afterward the passing away(death). The conduct of these state variables is controlled through utilization of arbitrary distributions to guarantee the system always advances.

The model was actualized utilizing an adaptation of CD++, a DEVS simulator, that backings the meaning of state variables for a cellular automaton. Taking after the usage of the single variable CD++ demonstrate in the multi-variable form the model was stretched out to incorporate inputs that could influence the likelihood distributions that control the birth and passing of human and in addition the simulation of impact of relocation to move people around inside the cell space.

# INTRODUCTION

The introduction of effects on birth of child and mortality rates due to density of people in an area in visual frameworks for real-time simulators is testing. The present approach is to utilize static board pictures that don't advance after some time. Past work has been done to model human conduct utilizing cellular automaton. This work presented a fundamental arrangement of state variables and transition rules to produce human conduct in view of the condition of the cell neighborhood.

To exploit the multi-variable capacities of CD++ and to develop the fundamental cellular automaton model proposed two new capacities were added to the model: the capacity to utilize input occasions to control the advancement of the system and the addition of impact of relocation to move people around through the cell neighborhood.

The proposed population generation model is a constantly developing system that is controlled utilizing likelihood distributions that decide when some state variable change state to influence the framework. This conduct is encoded in the transition rules inside the model and CD++ does not give a basic strategy to control these probability distributions. In past work these distributions were controlled by making different transition rules to characterize distinctive behaviors. This was a confounded strategy that did not accommodate a considerable measure of adaptability.

A technique for controlling the distribution utilizing inputs characterized as a part of the models CD++ event file was added to the model. Extra state variables that control the qualities of the likelihood distribution for that cell were included. The input changes the distribution state variables and is additionally sent to the cell neighbors so that a diffusion of the new distribution qualities can happen.

To additionally confound the multi-variable model and to enhance the authenticity of the human conduct the idea of relocation was added to the model. Two consistent state variables were added to the cell for movement, one for the north course and one for the east heading. The transition rules were then changed with the end goal that when an individual is available in the cell and these variables don't equivalent zero the individual in that cell is blown into a contiguous cell.

The conduct of these population generation models were tried exclusively and afterward assessed utilizing subject techniques. The execution of the multi-variable against the single variable CD++ was additionally assessed utilizing the fundamental population model.

# BACKGROUND

For the models created in this paper the essential behaviour of people is controlled by four state variables: chd, adt, act and ext. These variables represent to kid, grown-up, stage change from kid to an adult and the way toward passing away i.e demise. It is characterized by an a three dimensional neighborhood and gives four basic functions to computing the condition of every variable in the cell, in light of the variables of cells inside its neighborhood.

)

In this manner for population generation a cell should first have child present, represented by chd. On the off chance that the child is available it enters the transformation, act, if other kid in its neighborhood is as of now during the time spent developing into a grown-up. A grown-up is the present, adt, after the change procedure and stays exhibit until it dies. To enter the passing stage, ext, the cell should first have a human present and another cell in its neighborhood is as of now in the dying stage. Once the passing away process is finished, ext is valid; person is expelled from the cell.

In light of these transition rules the development of the population ext state variables. It is suggested that a procedure of utilizing random distribution to assign out genuine qualities to the variables. The strategy for controlling these distributions and the area inside the neighborhood they are utilized to simulate complicated human behavior.

## CD++ Implementation

The CD++ programming gives a programming language to actualize cell transition rules. In any case, the first CD++ version utilized just gives a single value for every cell. To display multi-variable frameworks, as in the human development model shown above, a few distinctive methodologies can be utilized. To execute the human development system a 2D plane was characterized for every state variable. This approach required the cell neighborhood to be decreased to two dimensions.

To handle the random generation of state variables an additional term was added to each transition rule that contained a probability distribution. If the other conditions failed but the normal probability distribution returned a value greater than 0.5 the variable became true.

Three methods were investigated for controlling the human growth system using these distributions. The first was a normal distribution that had the same average and deviation values across all cells for each state variable. The second was to create zones within the cell space that had transitions functions with different average and deviation values. The final method was an expansion on the zone method that modified the average and deviation values over time.

These methods were not easy to implement or modify and required extensive modifications to the transition functions to obtain different behavior.

An updated CD++ software was obtained that allows the definition of multiple variables for each cell. The addition of multiple variables will allow the existing model to be expanded to provide additional functionality. The transition functions have been expanded to allow for both the assigning of new values to the internal state variables and for sending the state variables to input ports on the neighboring cells. The transition function is broken into four components:

{port\_assignations} [ {assignations} ] {delay} {condition}

Output ports are assigned using in port\_assignations:

~variable\_port ≔ value;

The default output port can still be assigned using either ~out or (0,0). Assigning state variables is optional and is done in assignations:

$variable ≔ value;

In both of these cases the value can be the result of any function that returns a value. It should be noted that with the internal value of state variables and the variables sent on the output port can now be different.

With the introduction of multiple variables into CD++ the τ(N) function used in the Cell-DEVS formal specification to define the output value of the transition function must be expanded to reflect both the internal variable transition and the value sent to the output port.

# MODELS

Three models in view of the human growth system portrayed above were actualized in the multi-variable CD++ programming. The primary model was a fundamental usage of the humans or population density system. This model was utilized to approve the execution of the multi-variable CD++ against the single variable form. The second model actualized the ability to control the distribution parameters for the system from the CD++ input event file. The last model added the idea of relocation to the system and empowered people to move around the cell space.

## Basic Multi-Variable

A basic version of the human development, rules was executed utilizing the multi-variable CD++. Act, chd, and ext where represented utilizing state variables with an estimation of 0 or 1. The nearness of a person, adt, was demonstrated by the estimation of the cell, 0 represented to no human while 10 showed the nearness of people.

Three transition rules were all that was required to define the behavior of the system:

[human-gen]

rule : { ~out := 10; ~ext := $ext; ~act := $act; }

{ $hum := #macro(chd-change);

$ext := #macro(ext-change);

$act := #macro(act-change);}

100 {$ext = 0 AND ((0,0) = 10 OR $act = 1)}

rule : { ~out := 0; ~ext := $ext; ~act := $act; }

{ $chd := #macro(chd-change);

$ext := #macro(ext-change);

$act := #macro(act-change);}

500 {$ext = 1 AND (0,0) = 10}

rule : { ~out := 0; ~ext := $ext; ~act := $act; }

{ $chd := #macro(chd-change);

$ext := #macro(ext-change);

$act := #macro(act-change);}

100 {t}

Figure 1 - Basic Transition Rules

The macros *chd-change*, *act-change*, *ext-change* implement the transition rules for each state variable combined with the distribution used to evolve the system. An example macro is provided below:

if( ( $act = 0 AND $chd = 1 AND

( (1,0)~act = 1 OR (-1,0)~act = 1 OR

(0,1)~act = 1 OR (0,-1)~act = 1 OR

(2,0)~act = 1 OR (-2,0)~act = 1 OR

(0,2)~act = 1 OR (0,-2)~act = 1 ) )

OR normal(0.5,0.1) > 0.5, 1, 0)

Figure 2 - Example Basic Macro

The basic model uses a constant normal distribution that is applied uniformly to all cells in the neighborhood. To adjust the behavior of the cellular automaton the average and standard deviation values are different for each state variable.

### Formal DEVS Specification

The formal atomic cellular automaton model is defined as:

δ , δ , λ, and ta are deined using Cell-DEVS specications.

For the basic human growth model τ(N) reflects the internal value of the state variables.

Table 1 - Basic Transition Functions

|  |  |  |
| --- | --- | --- |
| **State** | **τ(N)** | **N** |
| Ext | 1 | $ext = 0 AND (0,0) = 10 AND  ( (1,0)~ext = 1 OR (-1,0)~ext = 1 OR  (0,1)~ext = 1 OR (0,-1)~ext = 1 OR  (2,0)~ext = 1 OR (-2,0)~ext = 1 OR  (0,2)~ext = 1 OR (0,-2)~ext = 1 )  OR normal(0.5,0.1) > 0.5 |
| Chd | 1 | ($hum = 1 AND $act = 0) OR normal(0.5,0.1) |
| Act | 1 | $act = 0 AND $chd = 1 AND  ( (1,0)~act = 1 OR (-1,0)~act = 1 OR  (0,1)~act = 1 OR (0,-1)~act = 1 OR  (2,0)~act = 1 OR (-2,0)~act = 1 OR   (0,2)~act = 1 OR (0,-2)~act = 1 )  OR normal(0.5,0.1) > 0.5 |
| Adt | 10 | $ext = 0 AND ((0,0) = 10 OR $act = 1) |
| Adt | 0 | $ext = 1 AND (0,0) = 10 |
| All | 0 | True |

The coupled model used for this basic mode has no inputs or ouputs and is defined as:

C = is the cell space set, defined above.

Z is defined as with a Pi,j for each state variables:

Pi,jY1 🡪 Pi,j-1X1 Pi,j+1Y1 🡪 Pi,jX1

Pi,jY2 🡪 Pi,j-2X2 Pi,j+1Y2 🡪 Pi,jX2

Pi,jY3 🡪 Pi,j+1X3 Pi,j-1Y3🡪 Pi,jX3

Pi,jY4 🡪 Pi,j+2X4 Pi,j-2Y4 🡪 Pi,jX4

Pi,jY5 🡪 Pi-1,jX5 Pi+1,jY5🡪 Pi,jX5

Pi,jY6 🡪 Pi-2,jX6 Pi+2,jY6 🡪 Pi,jX6

Pi,jY7 🡪 Pi+1,jX7 Pi-1,jY7 🡪 Pi,jX7

Pi,jY8 🡪 Pi+2,jX8 Pi-2,jY8 🡪 Pi,jX8

select = { (0,0), (-2,0), (-1,0), (1,0), (2,0),

(0,-2), (0,-1), (0,1), (0,2) };

The final simulation model consisted of only one coupled model, defined above, with no inputs or outputs to the system.

This model was verified to be functionally correct and evaluated against the original single variable CD++ model for both behavior and speed of execution.

## Improved Distribution Model

The fundamental population density model did not give any strategy to visualise the change of the system. The parameters of the distribution used to control generation of state variables were fixed and did not develop after some time. To actualize a strategy for controlling the distribution two state variables were included: avggen and avgext. These variables were utilized as the normal distribution: avggen was utilized for act-change and chd-change to control the development of people while avgext was utilized for ext-change to control the last phase of people i.e demise.

Several inputs where then added to the atomic model, these inputs are used to specify the *avggen* and *avgext* value for the cells they are connected to. Because CD++ does not allow a single input to be linked to a group of cells the concept of diffusing the new average state variables was implemented. A change in the average state variables within a cell is sent to the neighboring cells and they update their average state variables based on the distance from the input cell.

The transition functions defined previously were expanded to accommodate this new behavior. Two new macros were defined to update the average values, *avggen-change* and *avgext-change*. An example macro is provided in figure 3 where the first layer of cells around the input cell will have an *avgext* value of 100% the input value, the second layer of cells will have 90% and the third layer will have 80% the value.



Figure 3 - Input Average Diffusion

Figure 3 shows the resulting internal average values following an input at (4,4) of 1.0. The details of the internal calculations are provided in table 2.

The model also defines two additional transition functions that are used when an input is provided to the specified cell: *set-average-gen* and *set-average-ext*. These port-in transition functions, combined with manipulating the output *avggen* and *avgext* variables ensure the average state variable information is diffused correctly.

[set-average-gen]

~avggen := portValue(thisPort);

$avggen := $avggen + portValue(thisPort); $avgext := #macro(avgext-change);

[population-gen]

~avggen := 0; ~avgext := 0;

$chd := #macro(chd-change);

$ext := #macro(ext-change);

Figure 4 - Distribution Transition Functions

Figure 4 provides the components of the new transition functions that update the various average distribution variables. The process by which these variables are updated when an input occurs is as follows:

1. The set-average-gen or set-average-ext port-in transition function is called when an input occurs.
2. The cells internal state variable is added to the new port variable. The input can be considered as an addition to the existing distribution average.
3. The cell puts the value of the input (not its state variable) on the appropriate average output port. At the appropriate time the neighboring cells update their average state variable by calling *avgext-change* or *avggen-change*. Where they add the scaled input value obtained from the original cell output port.
4. Each cell then sets the average output ports to 0 so that the system does not have positive feedback.

The definition of the associated macros are provided in table 2.

### Formal DEVS Specification

The formal atomic cellular automaton model is based on the density of population model. With the addition of two state variables and the update the cell neighborhood the following changes are made to the formal DEVS specification.

*(-1, -1), (1, 1), (-1, 1), (1, -1)*

*(-2, -2), (2, 2), (-2, 2), (2, -2)*

*(-3, -3), (3, 3), (-3, 3), (3, -3)*

*(-2, 0), (-1, 0), (1, 0), (2, 0)*

*(0, -2), (0, -1), (0, 1), (0, 2)*

*(0, -3), (-3, 0), (3, 0), (0, 3)*

*(1, 3), (1, 2), (1, -3), (1, -2)*

*(-1, 3), (-1, 2), (-1, -3), (-1, -2)*

*(3, 1), (2, 1), (3, -1), (2, -1)*

*(-3, 1), (-2, 1), (-3, -1), (-2, -1)*

*(2, 3), (2, -3), (-2, 3), (-2, -3)*

*(3, 2), (3, -2), (-3, 2), (-3, -2) }*

This model expands the previous basic model by adding the following τ(N) functions.

Table 2 - Improved Distribution Transition Functions

|  |  |  |
| --- | --- | --- |
| **State** | **τ(N)** | **N** |
| Avgext  (internal) | max(0, min ( $avgext +  1.0\*( (0,-1)~avgext + (0,1)~avgext +  (-1,0)~avgext + (1,0)~avgext +  (-1,1)~avgext + (1,1)~avgext +  (-1,-1)~avgext + (1,-1)~avgext ) +  0.9\*( (0,-2)~avgext + (0,2)~avgext +  (-2,0)~avgext + (2,0)~avgext +  (2,2)~avgext + (-2,-2)~avgext +  (2,-2)~avgext + (-2,2)~avgext +   (1,2)~avgext + (-1,2)~avgext +  (2,1)~avgext + (-2,-1)~avgext +  (2,-1)~avgext + (-1,-2)~avgext +  (1,-2)~avgext + (-2,1)~avgext) +  0.8\*( (0,-3)~avgext + (0,3)~avgext +   (-3,0)~avgext + (3,0)~avgext +  (3,3)~avgext + (-3,-3)~avgext +   (3,-3)~avgext + (-3,3)~avgext +   (1,3)~avgext + (1,-3)~avgext +   (-1,3)~avgext + (-1,-3)~avgext +   (3,1)~avgext + (-3,1)~avgext +  (3,-1)~avgext + (-3,-1)~avgext +  (2,3)~avgext + (2,-3)~avgext +  (-2,3)~avgext + (-2,-3)~avgext +  (3,2)~avgext + (-3,2)~avgext +   (3,-2)~avgext + (-3,-2)~avgext ), 1.0 | No input |
| Avgext  (external) | 0 | No input |
| Avgext  (internal) | $avgext + X input | X input |
| Avgext  (external) | X input | X input |
| Avggen  (internal) | max(0, min ( $avggen +  1.0\*( (0,-1)~avggen + (0,1)~avggen +  (-1,0)~avggen + (1,0)~avggen +  (-1,1)~avggen + (1,1)~avggen +  (-1,-1)~avggen + (1,-1)~avggen ) +  0.9\*( (0,-2)~avggen + (0,2)~avggen +  (-2,0)~avggen + (2,0)~avggen +  (2,2)~avggen + (-2,-2)~avggen +  (2,-2)~avggen + (-2,2)~avggen +   (1,2)~avggen + (-1,2)~avggen +  (2,1)~avggen + (-2,-1)~avggen +  (2,-1)~avggen + (-1,-2)~avggen +  (1,-2)~avggen + (-2,1)~avggen) +  0.8\*( (0,-3)~avggen + (0,3)~avggen +   (-3,0)~avggen + (3,0)~avggen +  (3,3)~avggen + (-3,-3)~avggen +   (3,-3)~avggen + (-3,3)~avggen +   (1,3)~avggen + (1,-3)~avggen +   (-1,3)~avggen + (-1,-3)~avggen +   (3,1)~avggen + (-3,1)~avggen +  (3,-1)~avggen + (-3,-1)~avggen +  (2,3)~avggen + (2,-3)~avggen +  (-2,3)~avggen + (-2,-3)~avggen +  (3,2)~avggen + (-3,2)~avggen +   (3,-2)~avggen + (-3,-2)~avggen ), 1.0 | No input |
| Avggen  (external) | 0 | No input |
| Avggen  (internal) | $avggen + X input | X input |
| Avggen  (external) | X input | X input |

For the examples used in this paper a 20 x 20 cell space was used with both birth of a child and death of a human inputs at (4,4), (15,5), and (10,15). The coupled model specification is modified from the basic model in the following manner:

*Xlist = { inputDistAverageGenOne (4,4),*

*inputDistAverageExtOne (4, 4),*

*inputDistAverageGenTwo (15,5),*

*inputDistAverageExtTwo (15, 5),*

*inputDistAverageGenThree (10,15),*

*inputDistAverageExtThree(10,15) }*

*(0,0), (-1, -1), (1, 1), (-1, 1), (1, -1)*

*(-2, -2), (2, 2), (-2, 2), (2, -2)*

*(-3, -3), (3, 3), (-3, 3), (3, -3)*

*(-2, 0), (-1, 0), (1, 0), (2, 0)*

*(0, -2), (0, -1), (0, 1), (0, 2)*

*(0, -3), (-3, 0), (3, 0), (0, 3)*

*(1, 3), (1, 2), (1, -3), (1, -2)*

*(-1, 3), (-1, 2), (-1, -3), (-1, -2)*

*(3, 1), (2, 1), (3, -1), (2, -1)*

*(-3, 1), (-2, 1), (-3, -1), (-2, -1)*

*(2, 3), (2, -3), (-2, 3), (-2, -3)*

*(3, 2), (3, -2), (-3, 2), (-3, -2) }*

With the expansion of the cell neighborhood the Z function and associated select function increases significantly.

Z is defined as, with a Pi,j for each state variable:

|  |  |  |  |
| --- | --- | --- | --- |
| Pi,jY1 🡪 Pi,j-1X1 | Pi,j+1Y1 🡪 Pi,jX1 | Pi+1,j-3Y19 🡪 Pi,jX19 | Pi-1,j+3Y19 🡪 Pi,jX19 |
| Pi,jY2 🡪 Pi,j-2X2 | Pi,j+1Y2 🡪 Pi,jX2 | Pi-1,j+3Y20 🡪 Pi,jX20 | Pi+1,j-3Y20 🡪 Pi,jX20 |
| Pi,jY3 🡪 Pi,j+1X3 | Pi,j-1Y3🡪 Pi,jX3 | Pi+3,j+1Y21 🡪 Pi,jX21 | Pi-3,j-1Y21 🡪 Pi,jX21 |
| Pi,jY4 🡪 Pi,j+2X4 | Pi,j-2Y4 🡪 Pi,jX4 | Pi+2,j+1Y22 🡪 Pi,jX22 | Pi-2,j-1Y22 🡪 Pi,jX22 |
| Pi,jY5 🡪 Pi-1,jX5 | Pi+1,jY5🡪 Pi,jX5 | Pi+3,j-1Y23 🡪 Pi,jX23 | Pi+3,j+1Y23 🡪 Pi,jX23 |
| Pi,jY6 🡪 Pi-2,jX6 | Pi+2,jY6 🡪 Pi,jX6 | Pi+2,j-1Y24 🡪 Pi,jX24 | Pi-2,j+1Y24 🡪 Pi,jX24 |
| Pi,jY7 🡪 Pi+1,jX7 | Pi-1,jY7 🡪 Pi,jX7 | Pi-3,j+1Y25 🡪 Pi,jX25 | Pi+3,j-1Y25 🡪 Pi,jX25 |
| Pi,jY8 🡪 Pi+2,jX8 | Pi-2,jY8 🡪 Pi,jX8 | Pi-2,j+1Y26 🡪 Pi,jX26 | Pi+2,j-1Y26 🡪 Pi,jX26 |
| Pi,jY9 🡪 Pi-3,jX9 | Pi+3,jY9🡪 Pi,jX9 | Pi-3,j-1Y27 🡪 Pi,jX27 | Pi+3,j+1Y27 🡪 Pi,jX27 |
| Pi,jY10 🡪 Pi+3,jX10 | Pi-3,jY10 🡪 Pi,jX10 | Pi-2,j-1Y28 🡪 Pi,jX28 | Pi+2,j+1Y28 🡪 Pi,jX28 |
| Pi,jY11 🡪 Pi,j+3X11 | Pi,j-3Y11 🡪 Pi,jX11 | Pi+2,j+3Y29 🡪 Pi,jX29 | Pi-2,j-3Y29 🡪 Pi,jX29 |
| Pi,jY12 🡪 Pi,j-3X12 | Pi,j+3Y12 🡪 Pi,jX12 | Pi+2,j-3Y30 🡪 Pi,jX30 | Pi-2,j+3Y30 🡪 Pi,jX30 |
| Pi+1,j+3Y13 🡪 Pi,jX13 | Pi-1,j-3Y13 🡪 Pi,jX13 | Pi-2,j+3Y31 🡪 Pi,jX31 | Pi+2,j-3Y31 🡪 Pi,jX31 |
| Pi+1,j+2Y14 🡪 Pi,jX14 | Pi-1,j-2Y14 🡪 Pi,jX14 | Pi-2,j-3Y32 🡪 Pi,jX32 | Pi+2,j+3Y32 🡪 Pi,jX32 |
| Pi-1,j-3Y15 🡪 Pi,jX15 | Pi+1,j+3Y15 🡪 Pi,jX15 | Pi+3,j+2Y33 🡪 Pi,jX33 | Pi-3,j-2Y33 🡪 Pi,jX33 |
| Pi-1,j-2Y16 🡪 Pi,jX16 | Pi+1,j+2Y16 🡪 Pi,jX16 | Pi+3,j-2Y34 🡪 Pi,jX34 | Pi-3,j+2Y34 🡪 Pi,jX34 |
| Pi+1,j-2Y17 🡪 Pi,jX17 | Pi-1,j+2Y17 🡪 Pi,jX17 | Pi-3,j+2Y35 🡪 Pi,jX35 | Pi+3,j-2Y35 🡪 Pi,jX35 |
| Pi-1,j+2Y18 🡪 Pi,jX18 | Pi+1,j-2Y18 🡪 Pi,jX18 | Pi-3,j-2Y36 🡪 Pi,jX36 | Pi+3,j+2Y36 🡪 Pi,jX36 |

*(0,0), (-1, -1), (1, 1), (-1, 1), (1, -1)*

*(-2, -2), (2, 2), (-2, 2), (2, -2)*

*(-3, -3), (3, 3), (-3, 3), (3, -3)*

*(-2, 0), (-1, 0), (1, 0), (2, 0)*

*(0, -2), (0, -1), (0, 1), (0, 2)*

*(0, -3), (-3, 0), (3, 0), (0, 3)*

*(1, 3), (1, 2), (1, -3), (1, -2)*

*(-1, 3), (-1, 2), (-1, -3), (-1, -2)*

*(3, 1), (2, 1), (3, -1), (2, -1)*

*(-3, 1), (-2, 1), (-3, -1), (-2, -1)*

*(2, 3), (2, -3), (-2, 3), (-2, -3)*

*(3, 2), (3, -2), (-3, 2), (-3, -2) }*

The final simulation model consisted of only one coupled model, defined above, with six inputs into the system.

*X = { dist-average-gen-one*

*dist-average-ext-one,*

*dist-average-gen-two,*

*dist-average-ext-two,*

*dist-average-gen-three,*

*dist-average-ext-three }*

The desired functionality of the improved distribution functionality was validated and the behavior of the system was evaluated using a subjective methodology.

## Migration Model

To further upgrade the models effectively built up the idea of migration was added to the cellular automaton. On the off chance that idea of relocation(migration) was present in a cell, with people, then the movement would move the people into a neighboring cell. The turnaround is likewise valid, if the relocation in a neighboring cell is bringing on development in the correct direction and there is human present in the cell then the human will move into the cell.

Two state variables were added to represent the migration, *migrationup* and migration*right*. Positive values for these variables indicate the migration is occuring from the bottom to the top of the cell space or from the left to the right of the cell space. Negative values indicate the migration is occuring in the opposite direction. If both migration state variables are non-zero then human will move in the appropriate diagonal.

The migration variables are static during the simulation and are set as initial conditions in the system file. Due to the static nature of the migration state variables their value is only sent through the output port during the initialization phase.

Several changes were made to the transition functions to accommodate the new behavior. These changes affect both the conditions and the D value used to set the time for the state to update.

{~out=10 ….} {…}

{ if( #macro(blown-adt-in), 300, 100 ) }

{ (#macro(blown-adt-in) AND NOT   
 #macro(ext-adt)) OR  
 (#macro(gen-adt) AND NOT   
 #macro(blown-adt-out)) }

Figure 5 - Movement of humans due to migration

{~out=0 ….} {…}

{ if( #macro(mov-adt-in), 300, 300 ) }

{ #macro(mov-adt-out) OR   
 #macro(ext-adt) }

Figure 6 - Effects on humans due to migration

Two additional macros were added to simplify the transition rules. *mov-adt-in* determines if any of the cells in the immediate neighborhood has population moving in the correct direction as a result of migration and it has humans present. If these conditions are met it returns true and the cell becomes crowded. *mov-adt-out* controls if the cell should lose its current population due to migration. If the migration values are non zero and cell has a human being present in it, this macro returns true and is used to move the population from the cell. The details of the macros are provided in table 3.

### Formal DEVS Specification

The DEVS specification provided for the improved distribution model in section 3.2 was expanded for the addition of the migration variables.

Table 3 - Migration Transition Functions

|  |  |  |
| --- | --- | --- |
| **State** | **τ(N)** | **N** |
| Adt | 10 | ((((0,1)~migrationright < 0 AND (0,1) = 10)   OR ((0,-1)~migrationright > 0 AND   (0,-1) = 10)) AND (-1,0)~migrationup = 0   AND (1,0)~migrationup = 0) OR  ((((-1,0)~migrationup < 0 AND (-1,0) = 10)  OR ((1,0)~migrationup > 0 AND (1,0) = 10))  AND (-1,0)~migrationup = 0  AND (1,0)~migrationup = 0) OR ((-1,-1)~migrationright > 0 AND  (-1,-1)~migrationup < 0 AND (-1,-1) = 10) OR  ((1,1)~migrationright < 0 AND   (1,1)~migrationup > 0 AND (1,1) = 10) OR  ((-1,1)~migrationright < 0 AND   (-1,1)~migrationup < 0 AND (-1,1) = 10)  OR ((1,-1)~migrationright > 0 AND   (1,-1)~migrationup > 0 AND (1,-1) = 10) OR  ((-1,-1)~migrationright > 0 AND   (-1,1)~migrationup < 0 AND (-1,-1) = 10)  AND NOT  $ext = 1 AND (0,0) = 10  OR  $ext = 0 AND ((0,0) = 10 OR $act = 1)  AND NOT  $ext = 1 AND (0,0) = 10 |
| Adt | 0 | (0,0) = 10 AND  sqrt( $migrationup \* $migrationup +   $migrationright \* $migrationright ) > 0  OR ($chd = 1 AND $act = 0) OR normal(0.5,0.1) |

*D = {500 if S = 10 and S` = 0 and not #macro(mov-adt-out)*

*else 300 if S=10 and S`=0 and #macro(mov-adt-out)*

*else 300 if S=0 and S’=10 and #macro(mov-adt-in)*

*else 100 if S=0 and S’=10 and not #macro(mov-adt-in) }*

No changes were made to the couple model, neighborhood so no additional modifications to the formal DEV specification are required.

The desired functionality of the improved distribution functionality was validated and the behavior of the system was evaluated using a subjective methodology.

# Results

Each of the models developed for this paper were examined in a subjective manner to evaluate the realistic behavior of the population generation. Some examples of the behavior of each model are presented here along with commentary on the expected, desired and actual results.

Furthermore the efficiency of developing and executing the models in the multi-variable CD++ is also evaluated.

## Basic Multi-Variable

Implementing the population density rules previously developed in the single variable CD++[3] was found to be quite easy once the syntax of the multi-variable system was understood. It was also found that depending on the type of model being developed the multi-variable CD++ was not significantly faster.

Initially the basic model was developed without the output port values of the state variables were being updated correctly. Because of this the number of messages being handled by the simulator was drastically reduced and the execution time was significantly faster, but the desired behavior was not being produced by the system.

After the output state port values were being updated correctly and the appropriate messages were being processed by the simulator the execution speed of the model was found to be on the level of the single variable CD++.

The basic model, in both single variable and correct and incorrect multi-variable versions, was run on a 20x20 cell space for 005:000 seconds with a time step of 00:100. The timing results are presented in figure 7.

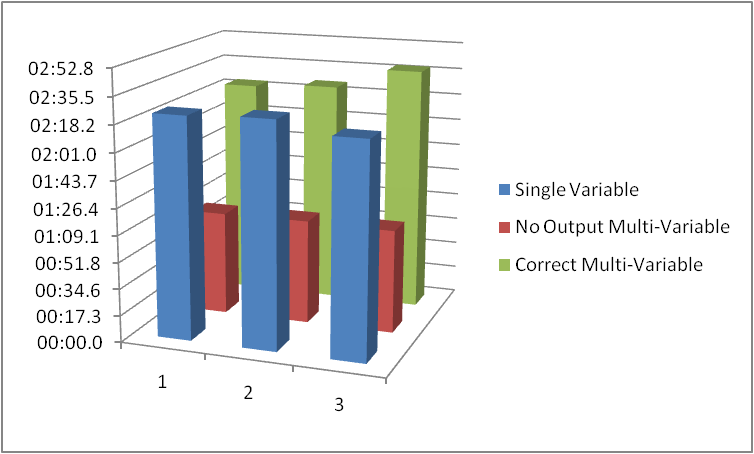


Figure 7 - Basic Model Timing Results

These results indicate that the execution speed of a multi-variable CD++ model is directly related to the number of state variable messages that must be processed. Models that require multiple state variables that are not communicated to the cell neighborhood will experience a significant increase in processing time.

Models, like the population generation defined in this paper, that do require the state variables to be communicated across the cell

neighborhood do not receive the speed bonus associated with the multi-variable CD++ simulator because the number of messages associated with an update is not drastically reduced.

The multi-variable implementation was also found to perform in the same manner as the single variable implementation. As expected the basic distribution model used for this implementation did not produce realistic human behavior.

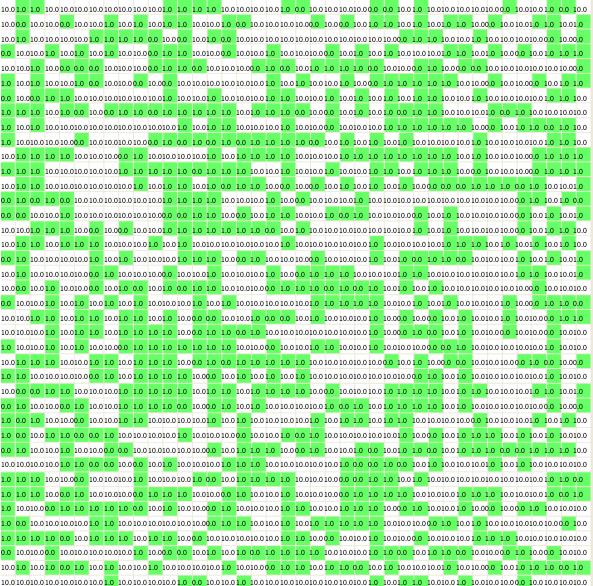


Figure 8 - Basic Multi-Variable Visualization

## Improved Distribution Model

The improved distribution model was implemented in the multi-variable CD++. It was found that the ability to define multiple state variables made it much easier to expand a model with additional behavior. The new variable was defined and the associated transition function and macros were created. Unlike the single variable CD++ additional dimensions did not need to be added and the impact of their presence on the neighbourhood and model considered.

For the results presented in this section a 20x20 cell space was run for 005:00. The distribution model was first tested to determine that diffusion of average results occurred correctly. To do this the output port used in the drawlog application was updated to reflect the appropriate average state variable. Several different average inputs were added to the event file.

00:00:00:000 dist-average-gen-one 0.5

00:00:00:100 dist-average-gen-two 0.5

00:00:00:200 dist-average-gen-three 1.0

00:00:00:300 dist-average-gen-three -0.5

Figure 10 shows the effects that multiple different average inputs will have on the average value in a cell if the input cells are close enough that the diffusion regions overlap. Figure 9 shows the effect using the input events to affect the same distribution over time. The image on the left is at 00:300 and the image at the left is at 00:400, after the effects of the new average input have been propagated.

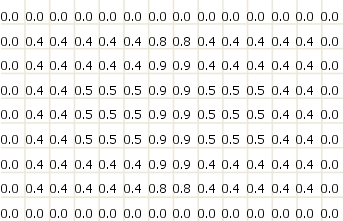


Figure 9 - Multiple Distribution Inputs

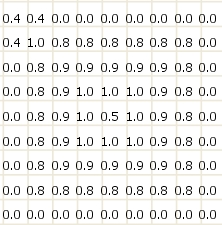
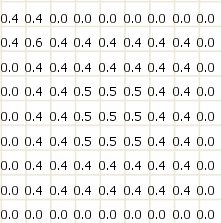
 

Figure 10 - Adding Distribution Inputs

These results show that the average state variables demonstrate the desired behavior in response to inputs. The next step was to validate the effect the average state variables were having on the human behavior. This test was done in two stages, the first to test the growth of humans and the second to test the death phase

To test the growth of humans the following input file was used on the cell space.

00:00:00:000 dist-average-gen-one 1.0

00:00:00:500 dist-average-gen-one -0.5

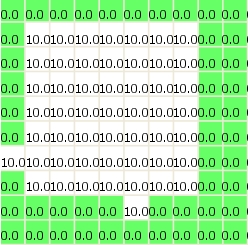
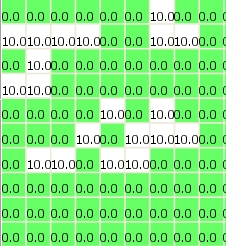
 

Figure 11 – Generation Test

On the left of figure 11 is shown the state of the population model at 00:400, after the input of an average of 1.0 has taken effect. Clearly the presence of a human in the region affected by this average is almost guaranteed as the generation average is greater than 0.5. The right side of figure 11 shows the system at 01:100, after the generation average has been reduced to 0.5 through the input. The growth of humans in the system is much more random as effect of the normal distribution is more pronounced.

The second test performed on this model was on the effect of the death average. For this test the following input files was used:

00:00:00:000 dist-average-gen-one 1.0

00:00:00:500 dist-average-ext-one 1.0

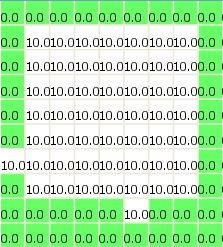
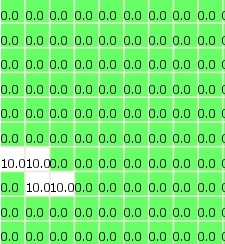
 

Figure 12 - Death Test

On the left of figure 12 is shown the state of the population model at 00:400, after the input of a generation average of 1.0 has taken effect. The right side of figure 12 shows the system at 01:100, after the input of an death average of 1.0 has taken effect. The death of humans takes priority over the birth; this explains the lack of population in the input region despite a generation value of 1.0. Also the effect of the death occurs much later than the effect of generation due to the value of 500 used for D when death occurs.

After the correct behavior of the input distribution model was verified using the individual tests the subjective behavior of humans was examined. The use of input values the changed over time for death and birth provided a much improved manner of controlling the human growth behavior than was seen in the basic population model.

The following input values were used to generate the examples provided in figures 13 – 15.

00:00:00:000 dist-average-gen-one 0.8

00:00:00:100 dist-average-gen-two 0.3

00:00:00:200 dist-average-ext-two 0.4

00:00:00:300 dist-average-gen-three 1.0

00:00:00:400 dist-average-gen-ext 0.5

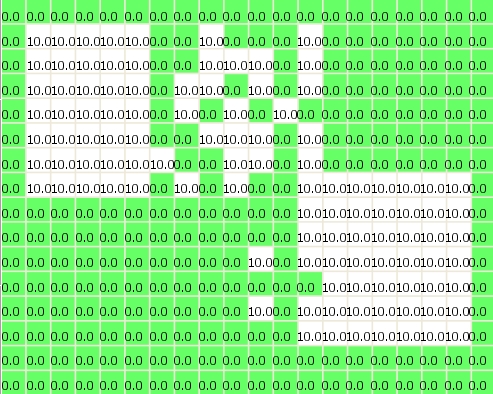


Figure 13 - Full Distribution T=01:000

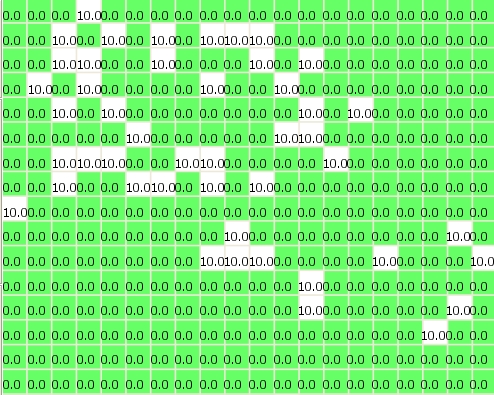


Figure 14 - Full Distribution T=01:500

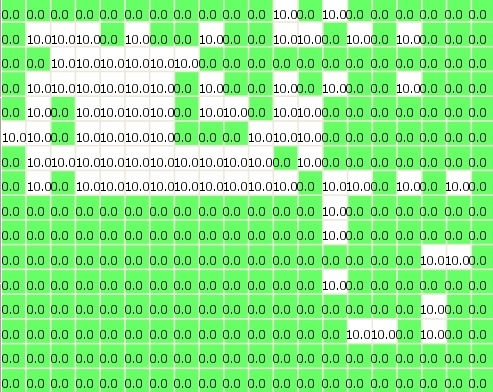


Figure 15 - Full Distribution T=05:000

The improved distribution model allowed a much greater degree of control over where humans were seen growing up. However the behavior of the humans within the simulation still left something to be desired if they were to be used as a basis for generating humans in a visual system.

# REFERENCES

1. A cellular automata model on residential migration in response to neighborhood social dynamics. (n.d.). Retrieved December 21, 2016, from http://www.sciencedirect.com/science/article/pii/S089571771000333X
2. Migration Processes Modeling with Cellular Automation Shmidt Yu. D., Ivashina N.V., Ozerova G.P., Lobodin P.N. Far Eastern Federal University, Vladivostok, Russia {syd, ivashina.nv, ozerova.gp,lobodin pn}@dvfu.ru
3. G. Wainer: "CD++: a Toolkit to Define Discrete-Event Models", Software, Practice and Experience, Wiley, Vol. 32, No 3. pp. 1261-1306. November 2002
4. Genetic Algorithms for the Calibration of Cellular Automata Urban Growth Modeling, Jie Shan, Sharaf Alkheder, Jun Wang