Self-recovery of a distributed system after a large disruption

New Mexico Supercomputing Challenge

Final Report

April 1, 2012

Team 15
4th Grade

Aspen Elementary School, Los Alamos, NM

Team Members:
Victor Popa-Simil
Henry Poston

Sponsoring Teachers:
Mrs. Thomas

Project Mentors:
Liviu Popa-Simil

Table of contents:

Executive Summary 3
Statement Of The Problem That Have Been Investigated 4
Method Used 4
  Model 4
  Stages 4
    Stage 1 4
    Stage 2 5
    Stage 3 5
    Stage 4 5
    Stage 5 5
Results 6
Conclusions 6
The Software, ... 7
  Logical Diagram 7
  General structure of the Net-Logo code 8
    Routine “Create patches” 8
    Routine “Create Animal “ 8
    Routine “detect rabbits to move on patch 8
    Routine “move animal” (Ai(xi,yi)) 8
    Routine “disruption” 9
    Code Exemplification 9
  The Code 10
    Result 11
Acknowledgments 11
References, ... 12
Annexes: 13
  1 - Tables, and Excel simulation of a possible result 13
  The most significant and original achievements on the project 14

Executive Summary

Our goal is to identify and understand those interaction conditions and their dependence of the initial state that are driving to recovery of the systems hit by calamities, the self repairing of materials and tissues hit by radiation and how the structures have to be designed in order to increase their resilience.

To solve the problem on computer we reduced the problem to a group of animals living together in the same land. We created a group of animals such as rabbits and squirrels. Soon the rabbits and squirrels take control over territory by grouping into clusters. Then a large disruption scares all the animals and they start to scatter in every direction possible out of disruption area. Once the disruption [example: lightning] ends, the animals will try to regroup on their territory. We will apply properties to rabbits, as rabbit likes rabbit but no more than 4, and squirrels like squirrels but no more than 8 and the same properties for patches.

We used Net logo code to simulate what might happen if a population such as rabbits and squirrels is disturbed and trends to recover. In order to better understand the process we used a chessboard simulation with checker pieces, and also Excel based simulations. On Excel we have tried various configurations obtained with random number generators in order to better understand how to interpret the final results and how to make the Net logo programming. Our task is not finished remaining that next year to learn more programming and mathematics in order to better analyze the recovery dependence on agent features in order to better understand the problem.

Statement Of The Problem That Have Been Investigated

Many natural phenomena like calamities wars or other damaging events produce dislocation of populations of various elements as animals, bacteria from their habitat or may take place even inside materials exposed to damage that all trend to recover.

From the many recovery paths possible, the main interest is toward those processes that drive to the similar structure as before the incidents, and towards those conditions of interaction between them and environment that have to be met.
The process is called self-recovering, self repairing or healing.

Explicative Note

Disruption and recovery happen frequently in nature.
Recovery is a process done to fix 1 or many individuals such as humans, animals or bacteria. When there are many, they form a group that performs many activities.
We think that a large disruption will affect more than 50% of the group but less than 90%.

Method Used

Model

To solve the problem on computer we reduced the problem to a group of animals living together in the same land.
We created a group of animals such as rabbits and squirrels. Soon the rabbits and squirrels take control over territory by grouping into clusters.
Then a large disruption scares all the animals and they start to scatter in every direction possible out of disruption area. Once the disruption [example: lightning] ends, the animals will try to regroup on their territory. We will apply properties to rabbits, as rabbit likes rabbit but no more than 4, and squirrels like squirrels but no more than 8 and the same properties for patches.

Stages

Stage 1

Create patches (input number of patches and color)

Note:

We may create several colors of patches with the properties:

- Green supports more rabbits than squirrels (10 rabbits one squirrel) because green could be grass
- Brown supports more squirrels than rabbits (1 rabbit 7 squirrels) because brown might be oak forest or nuts
- Yellow supports less rabbits and squirrels (2 rabbit 2 squirrels) because it might represent a farmers land and, and so on.

Create animals

We create rabbits (input number of rabbits and properties), and we create squirrels (input number of squirrels)

Note: net logo created everything in the central patch coordinate (0,0)

We have 2 options:

1. - Leave them as they are and start the program
2. - Move them randomly and start the program

Stage 2

Move animals

We went patch by patch and see how many are on each patch vs. patch properties.
If there are too many we will decide witch one will move first in first out or last in first out. We had to decide in what direction and how long it will go until it stops and how much time it will have until it has to leave. If they are not enough animals on a patch to trigger mandatory move we introduce timing for each animal and then move.
This will bring an equilibrium distribution of the animals on patches.

Stage 3

Disturbance

We created a disturbance in order to simulate a calamity like a large noise or a natural disaster (need to select the patch the calamity starts on and the dimension of the effected zone) we ignore any damages brought to patches and their recovery time or the eventuality of transforming patches into black patches where rabbit nor squirrel is able to go.
In the calamity area all the animals will evacuate immediately each animal will go on a random direction until is out of the area

Stage 4

Move animals

The calamity is over and the animals will move back to the original land and start to regroup after the same pattern as in stage 2.
From time to time we will see the differences between the distribution before the calamity and present distribution and present like maps or dislocation number

Stage 5

Final analyzes

There are 2 options:

1. - Stop and analysis data and draw conclusions.
2. - Apply a new calamity and collect more data and then stop.

Note: this scenario resembles Los Alamos mandatory evacuation during Los Conches fire.

Results

In this year we learned Net Logo and how to use it

We defined our problem and using models and simulations we got to understand it a little bit better.

The project is not yet ready because we need to see how we can make in Net logo the analysis of the results after the recovery and how we can interpret the recovery based on initial conditions and the properties imposed to the agents and patches.

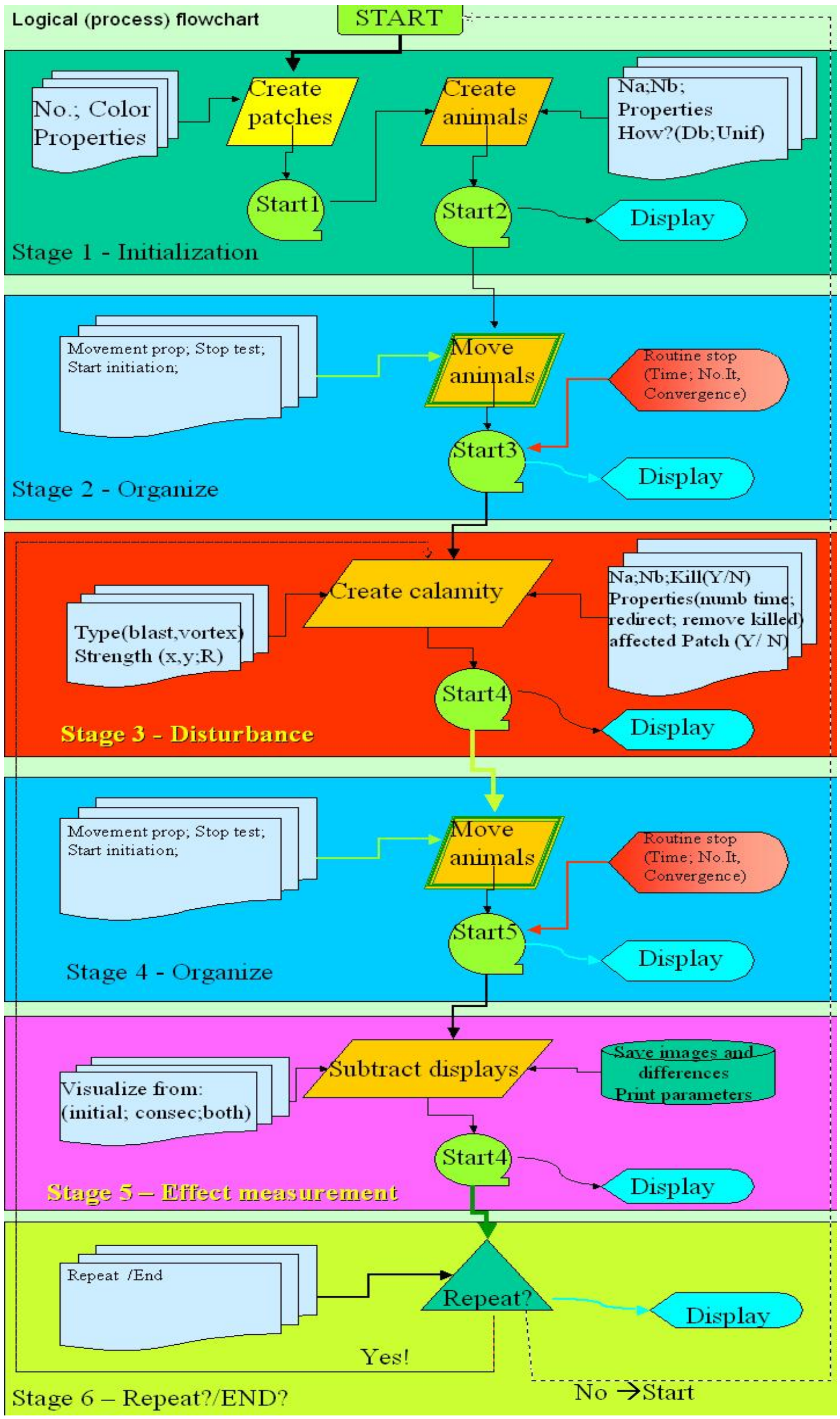
Conclusions

Net Logo is a programming environment and easy to use and we still need to learn more about it in order to master it and completely solve our problem.

The disruption and recovery of systems is a very complex process that depends of many conditions and properties of the actors involved in the process.

The Software,

Logical Diagram



Is presenting the same model as a map or chart of actions used for representation and to carry out certain types of reasoning?
The logical diagram has 6 stages each designed to be a sub-routine coded and tested separately. There are marked with different colors and input and output data are highlighted.

General structure of the Net-Logo code

<u>Routine "Create patches"</u>

Introduce no of patches on direction x = Npx; on direction y = Npy, where Np are max numbers of patches on each direction. We may easily generate various patterns of patches.

Make stripe patches
Along x direction IF (y=odd; pcolor=green; else = brown)
Along y direction IF (x=odd; pcolor=green; else = brown)

See the example form simulation

Make chess board patches diagonal strips
Test: IF (x+y=odd; pcolor=green; else = brown)

<u>Routine "Create Animal "</u>

We entered the no of rabbits No.r we enter the number of squirrels No.s.
This numbers of animals and number of patches is a strong initial condition that determines the behavior of the system. In simulations we observed that for less animals than patches the recovery is difficult to evaluate. It is an example of reduced population. For animals numbers higher than the number of patches continues movement is observed and limits have to be introduced in order to declare the moment when the recovery is achieved and what follows is the system movement.

See below "Code exemplification" and "the code".

<u>Routine "detect rabbits to move on patch</u>

We established (patch limit; time limit) in order to bring supplementary conditions like in real life.
If Np(xi,yi) > Limit one rabbit have to go
Rabbit data (x,y, time on patch)
Decide what rabbit to go:
That with the longest time on patch (first arrived)
The last arrived
Random (Rabbit to go index =INT(Np(x,y)*Rand)

<u>Routine "move animal" (Ai(xi,yi))</u>

We took each animal and moved in a random distance (fd random Value) because we don not know what movement would be appropriate for this animal in case of disruption or in normal case when they are moving patch by patch searching for food.

Routine "disruption"

We thought at 2 methods but we picked method 2 being much easier to use.
Method 1 (draw 4 random numbers)
1 Generate a center coordinates (x,y) center
2 Generate a dimension on x and y (Dx, Dy)
Issue: for an even Dx, Dy, rounding up or down defines the affected zone…
Method 2
1 Generate a corner coordinates (Cx,Cy) ;Cx=Npx*rand; Cy=Npy*rand
2 Generate a another corner (Dx,Dy): Dx=Npx*rand; Dy=Npy*rand
Method 2 is preferred because it clearly works with integer numbers.

Everything inside the rectangle has to immediately move out
IF (Min (Cx,Dx)< xAi <Max (Cx,Dx) and Min(Cy,Dy) < yAi <
<Max (Cy,Dy)) Move Ai else do nothing.
Animal has the position Ai (xi,yi) The patches 0-npx on x and on y

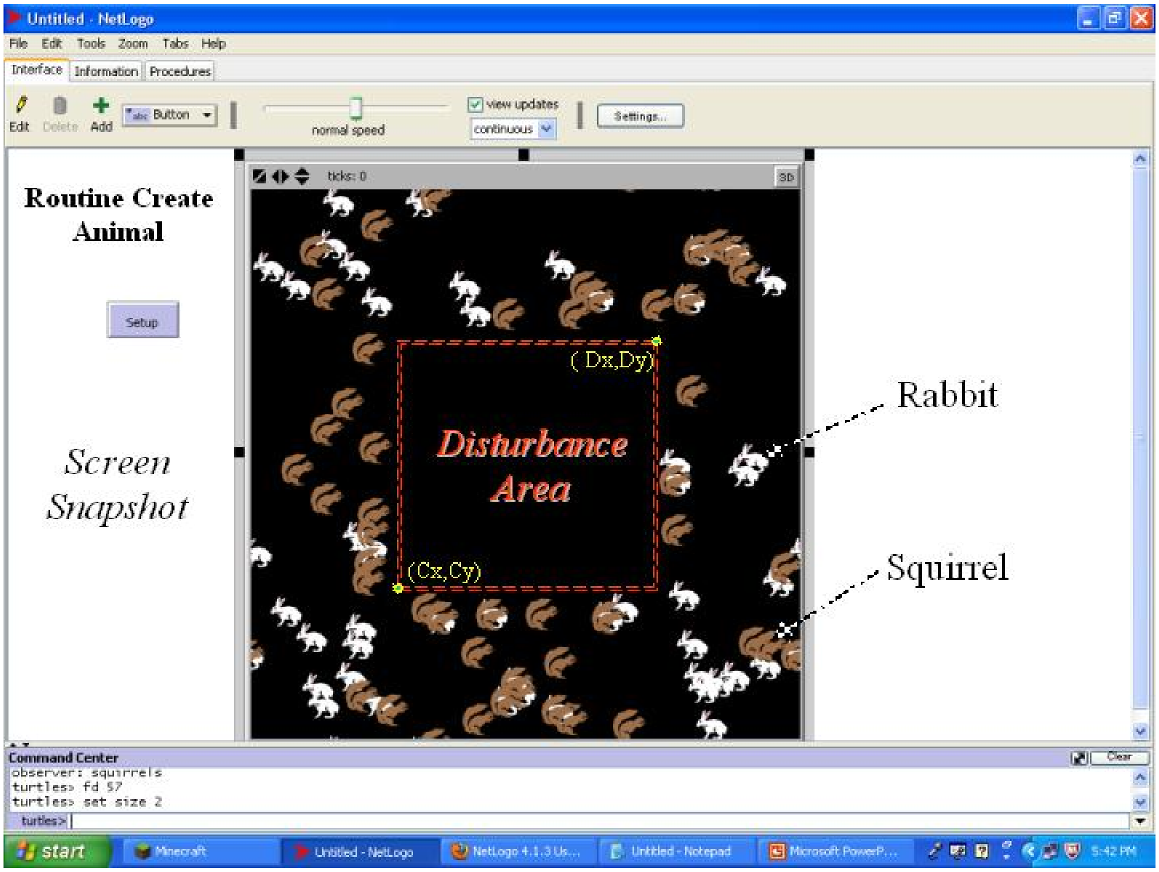
Code Exemplification

As a intuitive example the we used the chess board simulating patches and the checkers as rabbits or squirrels. We used the dices to generate the corners of the disruption area and we made an immediate movement convention for the checkers inside that area. This example helped us understanding the application's coding.

The Code

```
breed [rabbits rabbit]

breed [squirrels squirrel]

to setup

  clear-all

  create-rabbits 50

  ask rabbits [ set color white ]

  create-squirrels 50

  ask squirrels [ set color brown ]

  show [breed] of one-of rabbits    ;; prints rabbits

  show [breed] of one-of squirrels   ;; prints squirrels

  ask squirrels [set shape "squirrel"]

  ask rabbits [set shape "rabbit"]

end
```

Result



This is the result we obtained with net-Logo code.
The figure represents a screen shot after the calamity was generated were intentionally we took very few animals in order to look like the chess board simulation showed in the previous figure.

Acknowledgments

We want to thank to our mentor

Liviu Popa-Simil for :

- guiding us in the supercomputing initiative and
- teaching us valuable lessons

to

Mrs. Kathryn Thomas our sponsoring teacher, to

Mr. Jeff Fenchel our Interim Report evaluator for his valuable advices, and to

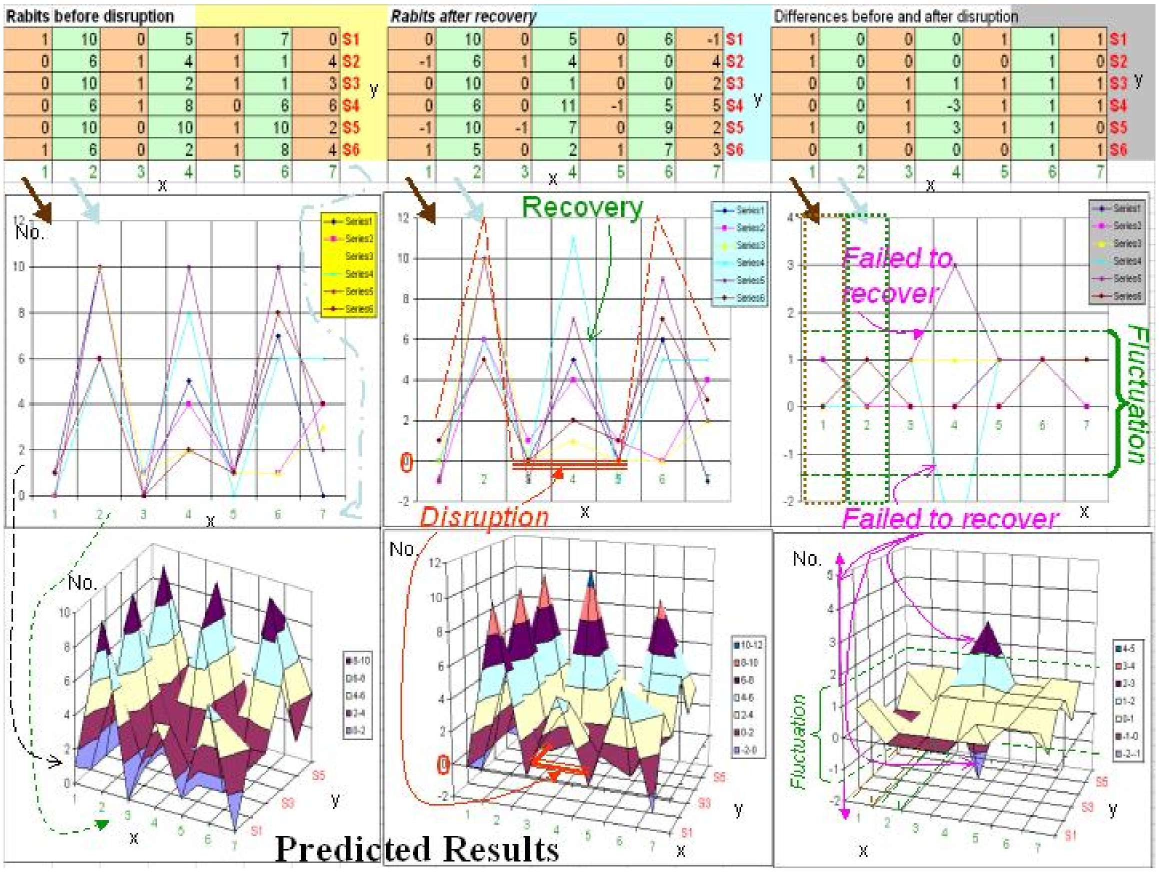
Supercomputing challenge organizers for helping us learn more on supercomputers and programming

References,

- http://ccl.northwestern.edu/netlogo/docs/ Net Logo website
- http://en.wikipedia.org/wiki/Healing Wikipedia website/healing
- http://www.thefreedictionary.com/healing the free dictionary website
- http://jmvidal.cse.sc.edu/netlogomas/ Multiagent NetLogo Models
- http://www.challenge.nm.org/interims/interim_guidelines.shtml

Annexes:

1 - Tables, and Excel simulation of a possible result



Rabits before disruption

1	10	0	5	1	7	0	S1
0	6	1	4	1	1	4	S2
0	10	1	2	1	1	3	S3
0	6	1	8	0	6	6	S4
0	10	0	10	1	10	2	S5
1	6	0	2	1	8	4	S6
1	2	3	4	5	6	7	

Rabits after recovery

0	10	0	5	0	6	-1	S1
-1	6	1	4	1	0	4	S2
0	10	0	1	0	0	2	S3
0	6	0	11	-1	5	5	S4
-1	10	-1	7	0	9	2	S5
1	5	0	2	1	7	3	S6
1	2	3	4	5	6	7	

Differences before and after disruption

1	0	0	0	1	1	1	S1
1	0	0	0	0	1	0	S2
0	0	1	1	1	1	1	S3
0	0	1	-3	1	1	1	S4
1	0	1	3	1	1	0	S5
0	1	0	0	0	1	1	S6
1	2	3	4	5	6	7	

We encountered difficulties in understanding and interpreting the results therefore a Excel simulation was created using random number generators that simulated the chess board example with a little bit more animals but only one type and less patches in order to be more visible.
The left side represents a simulation of the rabbits only, distribution on a stripe patches brown and green, generated randomly after the routine “create animals”.
The table represents the values of rabbit number on each patch on a 6 x 7 surface, with brown and green strips.
The charts represents the same data in two types – one in line by line, also called “series”, and the bottom one in a 3D view where the colors are showing various levels of the number of rabbits.
In the center by the red-triple-continuous line is represented the effect of the “disruption”, that emptied the central space and brought the number to “zero” in center and increased the number on the borders – in the “no causality” version we plant to develop.
The arrows show this effect, not shown in the table.

After the disruption ends, recovery starts, and the central patches are again repopulated and after a time frame or number of moves, it reaches the numbers presented in the central table, and represented in the two charts.
In order to evaluate how good was the recovery we subtracted the two matrixes – element by element and obtained the right table.
We understand that in life systems the recovery may not be perfect and the difference may not be zero, but have some small values we will attribute to fluctuation.
In the points the differences will be significant, say greater than 20% of the initial value, we may attribute that to the failures in recovery, inside the timeframe selected.
That feature will help us learn the recovery time needed for each system, to recover completely and further reach our project goals.

The most significant and original achievements on the project

Our original achievement is that we succeed to model a complex problem like calamity and recovery using a chess board-checkers distribution simulating the behavior of 2 types of animals “rabbits” and “squirrels” distributed on a territory.
A significant result for us was that we started to learn net logo and Excel and to use the computers in order to solve our problem and see how well it may match the reality.