Mathematically Modeling the Spread of Smallpox

New Mexico Adventures in Supercomputing Challenge
Final Report
April 6th, 2005

Team 055
Sandia Prep

Team Members
Alex Clement
Greg Fenchel
Jayson Lynch
Beryl Wootton

Teacher
Neil McBeth

Mentor
Ara Kooser

Executive Summary

The objective of these simulations is to mathematically model the spread of smallpox. This would create a more in-depth understanding of how smallpox is transmitted as well as its vectors. This could potentially lead to the development of new vaccines, advances in anti bio-terrorism (Legal Consumer Guide. http://www.legalconsumerguide.com/world_trade_center/anthrax/small_pox.html 12-8-04), and the improvement in the health of the general populace. The models could also be altered slightly to deal with other disease that spread by contact such as the flu.

The first model was written in the programming language Starlogo. It is an agent based simulation. Agents are individuals in a program that follow specific instructions. This simulation was based upon the SIRS mathematical model, using differential equations. In it all the agents wandered around in a “city” composed of outer impassable boundaries and semi-permeable boundaries within the city. One agent would start out infected and whenever an infected agent comes into contact with a susceptible agent there is a percent chance that it will become infected.

The second model was a percolation model written in the programming language Python. Percolation models were originally used to model water flowing through pores in rock. The percolation model is used to mathematically model the movement of agents through a given system. Our model traced the spread of smallpox through a given population.

In the Starlogo model the whole city and its surroundings became infected with a high population density. In smaller populations the virus would die out before infecting large numbers of agents. The behavior of the star logo model recreates real world data. In the Python model showed how the disease could be transmitted through interacting communities. Certain communities with high population densities allowed the smallpox to spread rapidly and jump from one grouping to another

These models provide valuable information. The star logo model accurately represented the smallpox epidemic and could be used to help determine how the disease will spread and where vaccinations and quarantines would be most effective. The Python model provides information about how smallpox spreads through socially connected communities and what the critical population density for a world wide epidemic.

Table of Contents

Figures

Introduction

This report was written by Alex Clement, Gregory Fenchel, and Beryl Wootton, from Sandia Prep. We worked on making a mathematical model of the spread of smallpox over a population. A mathematical model is defined as the general characterization of a process, object, or concept, in terms of mathematics, which enables the relatively simple manipulation of variables to be accomplished in order to determine how the process, object, or concept would behave in different situations (OALJ Law Library, Dictionary of Occupation of Occupational Titles, Glossary. http://www.oalj.dol.gov/public/dot/refrnc/glossary.htm)

We chose smallpox to model because of the importance of information on the subject. If smallpox were used as a bio-terrorism weapon and nobody could predict how it would behave, the results would be devastating. Although Smallpox is currently well contained (Saffer, Barbara. Smallpox. Farmington Hills: Lucent Books, 2003), there is always the threat of an outbreak, either by accidental release, or release in a bio-terrorist attack. At the moment, the world is ill-equipped to deal with any sort of out-break at all. Vaccinations were stopped in the USA in 1980, when smallpox was eradicated

from natural existence (Frampton, david. When Plague Strikes. New York: Harper Collins, 1995).

We first used a starlogo, agent-based simulation of the spread of smallpox throughout a city and the outlaying areas using real-world data about the disease. We collected real-world data on smallpox, and figured that there is an 80% transmission rate, and about a 2 to 3 week incubation period (Oldstone, Michael. Viruses, Plagues, and History. Questia Media America. http://www.questia.com/PM.qst?a=o&d=83255564. 12-8-04). We then used a percolation model written in the Python programming language to trace the movements of smallpox through a population.

Results:

Starlogo

In the Starlogo model the entire population became infected in the models with populations greater than about 400 agents. With smaller populations (~100 agents) full percolation did not occur. Meaning all the agents did not become infected. We know that the whole population became infected because the lines that represented total population and the total immune population met. Even though the infect rate is only eighty percent everybody became infected. This is because the population density is very high even with small numbers of people because the area is so small. This causes an agent to be exposed to the disease many times. As expected the disease spread the fastest in the urban area and typically arrive at the most isolated suburb last.

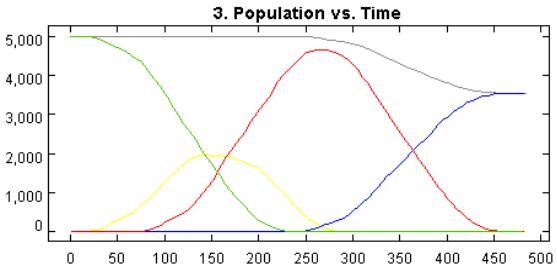

Fig. 1A

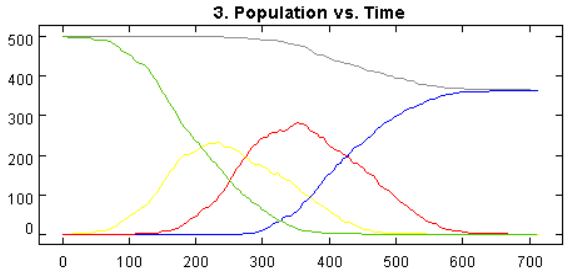

Fig. 1B

Python

Our Python model is a percolation/Game of Life model. The base units are communities. The model is programmed to connect social communities (ones with neighbors). Many times, except for when the population density is very high, there are groups of communities that are not social. The individual communities do not become infected because the communities do not move around. Therefore, unless the communities are close together and social it is highly unlikely to get full percolation of small pox. The only way to consistently get full percolation is to increase the population density which also holds true to a certain degree in starlogo.

This is a world with a small population density. There are many communities that are not connected at all.

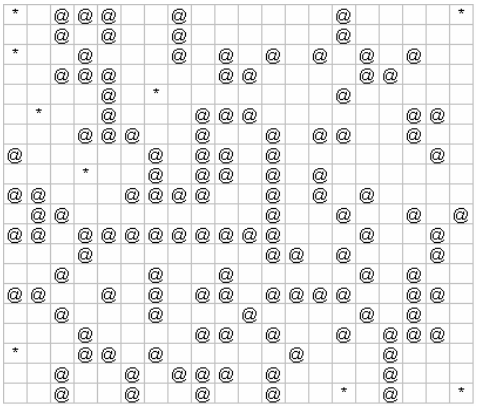

Fig. 2A

This is an example of our Python model with a high population model. Most of the communities are interconnected as shown by the @'s.

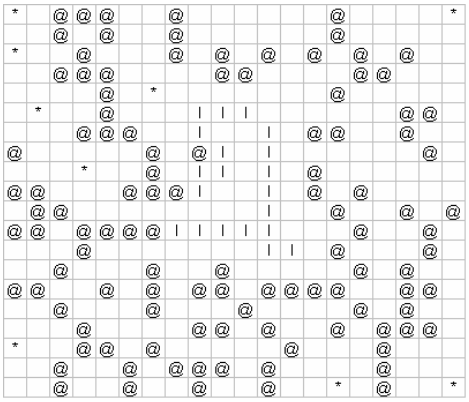

Fig. 2B

Here is the same density model that has been infected and run for about 14 repetitions. It infects a small area because it is not connected to very many communities.

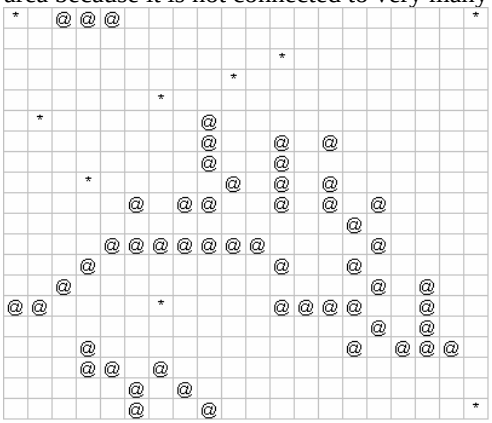

Fig. 2C

This is a world with a small population density. There are many communities that are not connected at all.

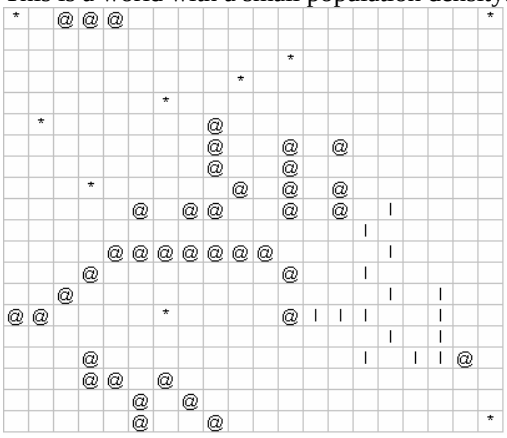

Fig. 2D

When this world is infected the infection does not spread throughout the population because many of the community groups are not connected.

Conclusion

In the Starlogo model we experienced successful transmission of smallpox in almost every test. We believe that this model is a moderately accurate model of how a smallpox epidemic would spread through a small population and is surrounding suburbs. It spreads in a similar way to the past epidemics that move through large populations (Smallpox. Center for Disease Control. http://www.bt.cdc.gov/agent/smallpox/overview/disease-facts.asp 12-7-04). The Starlogo model uses the SIRS differential equations to model the transmission of smallpox through a population.

SIRS Model equations

$ds/dt = -\beta\, is,\ s(0) = so \geq 0,$
$di/dt = \beta\, is - \gamma\, i,\ i(0) = io \geq 0,$

These equations were programmed into the Starlogo program. We were unable to model large populations with Starlogo, but we built a model using the Python language using an abstraction of population.

The model coded in Python demonstrates how smallpox moves through communities that are socially connected (have nearest neighbors). The population is handled in an abstracted way. The population is represented as immobile, interacting communities. Communities that are not connected do not contract small pox. This is better in many ways though because the models interpretation is not just limited to specific populations, it can be interpreted to

represent people or cites or even countries.

Acknowledgements

Many thanks to Ara Kooser, our mentor. Without his help our project would not have advanced as far as it did. He provided the moral support as well as the mental support we needed to finish and continue. If we encountered a problem that didn’t seem to have a solution, he helped us find one. Love for Ara!

Thanks to Anita Gallagher for lending her work office to us for meeting times whenever they were needed, as well as snacks, beverages, and the necessary hardware. Also, thanks for the time. =)

Thanks to Neil McBeth for the assurance of organized schedules and sponsorship. The program at our school wouldn’t exist, and we wouldn’t know about Adventures in Supercomputing without him.

Finally, thanks to our interim presentation evaluators, who provided much useful feedback and information, and came all the way down to Socorro to offer it to us!

Sources Consulted

1. Legal Consumer Guide. Legal Consumer Guide. http://www.legalconsumerguide.com/world_trade_center/anthrax/small_pox.html 12-8-04

2. Oldstone, Micheal. Viruses, Plagues, And History. Questia Media America. http://www.questia.com/PM.qst?a=o&d=83255564. 12-8-04

3. Smallpox. Center For Disease Control. http://www.bt.cdc.gov/agent/smallpox/overview/disease-facts.asp 12-7-04

4. Saffer, Barbara. Smallpox. Farmington Hills: Lucent Books, 2003.

5. Frampton, David. When Plague Strikes. New York: Harper Collins, 1995.

Appendix

Starlogo Program Listing

Turtle Code

```
turtles-own [age infecttime fertrate kids gender [male female] deathrate health [healthy latent sick
recovered] lastx lasty lasth]

to go
  infect
  recover
  check-health
  if (health = sick) or (health = latent)
    [setinfecttime infecttime + 1]
end

to infect
  rt random 60
  lt random 60
  cpatches
  fd 1
  if health = healthy
    [if  (((health-of one-of-turtles-here) = sick) or    ;if turtle you meet is sick or latent
       ((health-of one-of-turtles-here) = latent))
      and ((random 100) <= 80)                ;then with some chance
     [sethealth latent setinfecttime 0]]              ;become latent and note time of infection
end

to recover
  if health = sick
  [if infecttime > 49
    [ifelse (random 100) <= 70         ;recover with some chance
      [sethealth recovered] [die]]]
end

to check-health
  if health = healthy [setc green]
  if health = latent [ifelse infecttime > latent-time
                    [sethealth sick]
                    [setc yellow]]
  if health = sick [setc red]
  if health = recovered [setc blue]
end

to city
            setc 5
            fd 10
            pd
            rt 90
            fd 10
            repeat 4[rt 90 fd 21]
            fd 2
```

```
        rt 90
        fd 21
        rt 90
        fd 2
        rt 90
        fd 21
        rt 90
        setc red
                fd 23
                rt 90
                fd 21
                rt 90
                fd 23
                lt 90
        repeat 2 [
                fd 20
                rt 90
                fd 21
                rt 90
                fd 20
                lt 90]
        fd 20
        rt 90
        fd 21
        rt 90
        fd 20
        die
end

to cpatches
        if pc-ahead = red [rt 90 rt random 180 cpatches]
  if pc-ahead = 5 [if (random 100) > g [rt 90 rt random 180 cpatches]]
end
```

Observer Code

```
globals [dead time-step pop]

to startup
 plotid 3
end

to setup
  plotid 3
  clearplot
  setplot-title ""
  ca
  crt 1
  ask-turtles [city]
  setdead 0
  settime-step 0.1
  crt number
  ask-turtles [setinfecttime 0
        setfertrate 0
        if who > number / 3 [setxy 22 0]
        if who > number / 2 [setxy 0 20]
```

```
            if who > number * 2 / 3 [setxy 46 0]
            if who > number * 5 / 6 [setxy 0 42]
            ifelse who < infected
             [sethealth sick setinfecttime random (sick-time + latent-time)]
              [sethealth healthy]
            check-health
                        rt random 360
                        fd random 10]
  setup-graph
end

to setup-graph
  pp1 ppreset setppc green     ;healthy
  pp2 ppreset setppc red       ;sick
  pp3 ppreset setppc blue      ;recovered
  pp4 ppreset setppc yellow    ;latent
  pp5 ppreset setppc grey      ;pop
  setplot-yrange 0 number
  setplot-xrange 0 50
  setplot-title "Population vs. Time"
end

to graph
  pp1 ppd plot healthy#
  pp2 ppd plot sick#
  pp3 ppd plot recovered#
  pp4 ppd plot latent#
  pp5 ppd plot pop
end

to start
  startgobutton
  startgraphbutton
end

to sick#
  output (count-turtles-with [health = sick])
end

to recovered#
  output (count-turtles-with [health = recovered])
end

to latent#
  output (count-turtles-with [health = latent])
end

to healthy#
  output (count-turtles-with [health = healthy])
end

to deaths
  output (number - pop)
end

to population
```

```
  setpop count-turtles
  output pop
end

to sick-%
  output (((count-turtles-with [health = sick]) / number) * 100)
end

to recovered-%
  output (((count-turtles-with [health = recovered]) / number) * 100)
end

to latent-%
  output (((count-turtles-with [health = latent]) / number) * 100)
end

to healthy-%
  output (((count-turtles-with [health = healthy]) / number) * 100)
end

to stopit
  stopgobutton
  stopgraphbutton
end
```

Python Program Listing

```
"""Percolation/Game of Life Model for SmallPox Transmission"""

##############################################################################
### Version 1.0 March, 20th 2005
### Original code provided by Danny Yoo (Conway's Game of Life)
###
### Sandia Prep High School Team: Jayson Lynch, Alex Clement, Greg Fenchel, Beryl Wootton
###
### Modified by Ara Kooser (mentor for Sandia Prep Team)
###
### Additonal coding by Ara Kooser, Jayson Lynch, Alex Clement
### Addtional support, code, and ideas from: Danny Yoo, Lee Haar, Max Noel, Kent Johnson
###
### Many thanks to the Python Tutor List at http://mail.python.org/mailman/listinfo/tutor
### and Alan Gauld's Tutorial at http://www.freenetpages.co.uk/hp/alan.gauld/
##############################################################################
###

print """
Please pick your option:
1) Run the percolation/game of life model
"""

option = raw_input("Press [1] to start? ")

#if option == '1':
```

```
#    print "Instructions"


elif option == '1':

    import random
    import copy

    PERSON, EMPTY, CONNECTED, INFECTED = '*', '.', '@', 'I'


    perc = raw_input("Please enter a thresold between 0-1.   ")
    perc = float(perc)


    def percolation(perc):
        """Sets up a percolation threshold value"""
        randval = random.random()
        if randval > perc:
            return EMPTY
        else:
            return PERSON


    n = int(raw_input("Please enter a n dimension.   "))
    m = int(raw_input("Please enter a m dimension.   "))


    def random_world(M, N):
        """Constructs random world of size MxN."""
        world = {}
        for j in range(N):
            for i in range(M):
                world[i, j] = percolation(perc)
        world['dimensions'] = (M, N)
        return world


    small_world = random_world(m,n)


    def print_world(world):
            """Prints out a string representation of a world."""
            M, N = world['dimensions']
            for j in range(N):
                for i in range(M):
                    print world[i, j],
                print

##############################################################################
### This section starts the percolation part of the code to show connections
### betweeen all the points on the world that are occuipied.
##############################################################################

    def get_state(world, i, j):
        """Returns the state of the cell at position (i, j)in the world"""
```

```
    return world.get((i, j), EMPTY)


def count_live_neighbors(world, i, j):
    """Returns the number of live neighbors to this one."""
    live_count = 0
    for i_delta in [-1, 0, 1]:
        for j_delta in [-1, 0, 1]:
            if (i_delta, j_delta) == (0, 0):
                continue
            if get_state(world, i+i_delta, j+j_delta) == PERSON:
                live_count += 1
    return live_count


def is_connect(world, i, j):
    """Returns True if the cell at (i, j) is connected by 1-8 other"""
    return (get_state(world, i, j) == PERSON and
            (1 <= count_live_neighbors(world, i ,j) <= 8))


def next_world(world):
    """Shows which points are connected by denoting them with a @"""
    M, N = world['dimensions']
    new_world = copy.copy(world)
    for j in range(N):
        for i in range(M):
            if is_connect(world, i, j):
                new_world[i, j] = CONNECTED
    for j in range(N):
        for i in range(M):
            world[i, j] = new_world[i, j]


print_world(small_world)

raw_input("Press return to percolate the world")
print ""
print ""
next_world(small_world)
print_world(small_world)
print "The @ indicates points that are interconnected."

###############################################################################
###The infection portion of the code starts here
###############################################################################

raw_input("Press return to infect the world")

print "The upper left hand corner of the world is point (0,0)"


def infect_world(world):
    """Rebuilds the world with a single infected point"""
    M, N = world['dimensions']
    new_world = copy.copy(world)
```

```
        i = int(raw_input("Please enter a m value to infect   "))
        j = int(raw_input("Please enter a n value to infect   "))

        new_world[i, j] = INFECTED
        for j in range(N):
            for i in range(M):
                world[i, j] = new_world[i, j]

    infect_world(small_world)
    print_world(small_world)

##############################################################
## This portion spreads the pox around
##############################################################

    def count_infected_neighbors(world, i, j):
        """Returns the number of live neighbors to this one."""
        infected_count = 0
        for i_delta in [-1, 0, 1]:
            for j_delta in [-1, 0, 1]:
                if (i_delta, j_delta) == (0, 0):
                    continue
                if get_state(world, i+i_delta, j+j_delta) == INFECTED:
                    infected_count += 1
        return infected_count


    def infect_connect(world, i, j):
        """Returns True if the cell at (i, j) is connected by 1-8 other"""
        return (get_state(world, i, j) == CONNECTED and
                (1 <= count_infected_neighbors(world, i ,j) <= 8))


    def next_infected_world(world):
        """Shows which points are connected by denoting them with a @"""
        M, N = world['dimensions']
        new_world = copy.copy(world)
        for j in range(N):
            for i in range(M):
                if infect_connect(world, i, j):
                    if random.random() < .81:
                        new_world[i, j] = INFECTED
        for j in range(N):
            for i in range(M):
                world[i, j] = new_world[i, j]
##############################################################################
# This section contains the loop that evolves the infection in the world
##############################################################################

    def death() :
        for j in range(N):
            for i in range(M):
                if world[i,j] == INFECTED :
                    if random.random() < .31 :
                        new_world[i, j] = EMPTY
```

```
    def infect_repeat():
        L = int(raw_input("Number of repetations (in whole numbers)   "))
        P = int(raw_input ("To print every world, press 1 or just final world, press 2 "))
        r = 0
        if P == 1 :
            while r < L :
                next_infected_world(small_world)
                print """


                """

                print_world(small_world)
                r = r + 1
                death
        elif P == 2 :
            print """""

            """
            while r < L :
                next_infected_world(small_world)
                r = r + 1
                death
            print_world(small_world)
        else :
            print "You have choosen an incorrect value"
            infect_repeat()


    infect_repeat()
    print "Goodbye and thanks for using the Sandia Prep smallpox model"
    print "Press return to exit"
    raw_input()

elif option == '2':
    print "Goodbye and thanks for using the Sandia Prep smallpox model"


else:
    print "The computer is your friend, however you have choosen an invalid command"
    input ()
```