Predicting Future Crimes in Serial Cases

New Mexico
Supercomputing Challenge
Final Report
April 5, 2006

Team Number 124
Socorro High School

Team Members

-Omar Soliman
-Shaine Baldwin
-Thor Johnson
-Jayanta Majumdar

Teachers

- Martin Riggenbach
- Hanh Nguyen

Project Mentor

- Monte Mitzelfelt

Table of Contents

Executive Summary

Serial crime is getting to be a bigger problem in this day and age, so we designed a prototype model of a program that can be used to predict where a serial killer will strike next. We created our model in Java on NetBeans. It contains 20 intersections resembling streets, with 4 sites at each intersection, giving a total of 80 sites, which represent an area. We have focused our program on two types of serial crime: theft and kidnapping, which are represented by two check boxes at the top of the screen.

To begin, the user is presented with a drawn grid representing the city or subcategory thereof. There is an activation button is located at the top of the screen (used to activate the program), with the drop down menus at the right and a help bar at the bottom. The user then selects a choice from each drop down menu corresponding to what is located at each site. To aid visual awareness, a small icon is drawn at the appropriate site to signify what is there. There also checkboxes under the drop down menus to make one of the grid's streets a highway or road. The user then presses the activation button, and using the given restraints, the algorithm in the program carries out its task. Each site's gravitational value for crime is scanned in, which is added to the overall "gravity" of that site's area. The computer then locates the greatest hot spot(s), which are then shown by having red circles drawn around them.

Introduction

Serial crime has always been a serious menace to society. With all the advances of today's technology, predicting where a criminal might strike next using computers has always seemed to be close in reach. Modern initialization of projects into this subject has proved it to be complicated, with often inaccurate outcomes. In our project, we have attempted to reproduce more accurate results than the large programs on a smaller-scale prototype, using crime "gravity" as a measure of how strong it will attract the criminal. By using actual research into this subject from high-level journals, we are attempting to create a sophisticated yet basic program in Java that enables the computer to deduce logically and figure out the most probable location the next crime in a mock serial case would happen.

We chose to attempt the creation of this model because we wanted to contribute towards the first steps of the eradication of this nuisance from humanity and provide inspiration for others to attempt the same.

Approach (Scope)

There is no 100% exact way to pinpoint the next site in a serial criminal's spree (as you can't read his mind). Due to this fact, we based our project on how a generic criminal would think and act in an average situation. We decided that buildings (objects) each have their own gravity towards the acts of the criminal, either negative (repelling) or positive (attracting) and used this approach to base our various objects' gravities. The model in which our criminal has the next crime in his spree is a 5 x 4 grid with 20 intersections, each containing 4 sites, a total of 80 locations. The preprogrammed objects can then be placed at those sites (via drop down menus). Each object has 2 gravities, one for a kidnapping spree and one for a theft spree, as certain factors act differently in each case, i.e. a bank can be robbed, but is highly unlikely for a kidnapping to occur there. After the user inserts the various objects at the sites, the program compiles all the gravities to find where the most likely place to be struck is located.

Assumptions

- The bad guy acts in a way offering him the least chance of getting caught.
- He correctly acts upon the pull or push of gravity.
- He can only attack sites that can rationally be attacked in real-life (in other words, he’s not going to rob a police-station).
- All the predefined gravities are proportional to real life.
- He acts differently under the pretenses of kidnapping and theft.
- The user correctly translates the real world into the programs grid.

Procedure / Method

Our program was written in Java on the NetBeans IDE 4.1 application. We began by constructing the drop down menus. We added all the different objects to them, then placed them along with the various checkboxes in a panel and positioned to the left. Each item in the menus triggers a listening method, that when selected, draws the appropriate mini-picture of the object on the grid in the correct location to aid in visual awareness. This process also calls up the object's gravity, which is added to its intersection's total gravity. The checkboxes underneath the menus allow the option of redrawing a road to a highway, which has a different gravity. Second, we created a canvas where our grid and intersections were displayed, along with the mini-pictures. A label was placed on the bottom to help the user with the procedure. Finally, an activation button was placed at the top to activate and refresh the program.

When started, the program displays all of the above objects mentioned. Next, the user selects the objects to be in the mock "city" from the drop down menus. As each object is added, a miniature representation of it is drawn on the map to aid in visual coordination. When the user is finished, they press the activation button at the top to initiate the program's "finding" algorithm.

The program first scans in all the objects selected in the menus and stores a representational value of each as an integer. These values are then compared with default ones to act as a “fault checker”, making sure that there is enough information to work with. If there isn’t, it halts the procedure and changes the help label to reflect this. If the input is acceptable, it proceeds with gravity compilation.

First, the computer divides the objects that can be hit and those that just affect other sites, into 2 groups. This is used as a reference later. Next, gravity is amassed from the objects at each intersection to produce a sum of 20 points of gravity. The algorithm exposes these points to each other, and each one alters the sites near it to produce a “ripple effect”. These altered values are stored in different variables in order to keep original information and to allow comparison after sorting. The altered values are then sorted to find the one with greatest positive gravity. This one value is compared to the original values, allowing the original to be identified. Once identified, the program pulls up the 4 objects at that site and finds the ones that can be attacked. The one with the highest gravity from there is identified as the most likely place to be struck.

The paint method is called up along with the time method to communicate the findings to the user. Three red circle are drawn at

the target object, each at a different time via the time method, to produce a homing in effect and allow the site to be easily seen. The activation button then has its properties changed into a refresh button. After the user has recorded the findings, or wants to try again, they simply click the button. A method is called that refreshes all values and clears the screen except for the items that appear when it starts up. The program becomes ready to use again.

Results

Data from our program seems to be consistent with each run. The serial criminal was predicted to rob or kidnap from only the sites where this was allowed. The refresh method successfully resets all values and erases additional objects on the screen.

Conclusion

In a staged scenario, our program would successfully predict where the criminal would strike next. As there is no real way to compare these results to the real world unless used in the field, we can't say that our program would be 100% accurate when actually used. However, we believe we have accomplished our goal.

Significant Original Achievement

Our most significant original achievement was producing a program that has the same purpose as ones only experimented on by universities (but much simpler with more direct results) that can directly affect all of humanity and possibly leading to the stoppage of crime altogether.

Code

Our code is currently being reformatted into a more concise and exact formatting. It does, however, work in its present stage. Screenshots are also unavailable at submittal time.

Recommendations

If this code is implemented into a supercomputer, millions of sites in thousands of cities in dozen of states could be used as the grid. Many more variables that give off gravity could be imported and the gravities could be tailored to a specific criminal whose psychological profile is known, thus producing much more accurate results. However, at our scope, we believe that it’s accurate for our purpose.

Acknowledgements

We would like to thank all who helped us, including our teachers, mentor, Dr. Soliman, Dr. Baldwin, and Dr. Majumdar. Special thanks to Mr. Jackson and Mr. Banks III for help in our Java terminology and our code.

References

[1] Australian Institute of Criminology © 2005
http://www.aic.gov.au/publications/crm/crm030t.html

[2] Stephen Schneider, Ph.D.
School of Justice Studies Ryerson University ©2002
http://www.justice.gc.ca/en/ps/rs/rep/2002/rr2002-7/rr2002-7_002.html

[3] Dr. Derek J Paulsen
Director, institute for the Spatial Analysis of Crime
©2005 UK Crime mapping Conference (electronic)
http://www.jdi.ucl.ac.uk/downloads/pdf/third_mapping_conference/presentations/derek_paulsen.pdf

[4] Diana Ehlers, Senior Researcher
Published in Nedbank ISS Crime Index (electronic)
Volume 2 1998 Number 2, March - April
http://www.iss.co.za/pubs/CrimeIndex/98Vol2No2/PREDICTION.HTML

[5] Ralph Morelli, Author
Prentice Hall; May 5, 2002 ©2002
Java, Java, Java: Object-Oriented Problem Solving