Abstract

Police have had issues with how to properly allocate police resources so that crime can be minimized. To solve this problem, we looked at how game theory can be applied to reduce crime, namely in Irvington, New Jersey. We constructed a simple static game with a Nash equilibrium, and found the optimal strategies for the police to use. By splitting Irvington into many data points, or cells, we were able to find the best places for the police to allocate resources. We also tested our model, and found that our game was 19 ± 1 percent more effective than a uniform distribution of police.

1 Introduction

Crime is a frequently occurring problem in every city that each town’s police force must combat. [4] Police departments all over the nation have to deal with an almost unpredictable enemy, and frequently have to deal with placement issues where they are too far away to prevent or stop crimes in action. While the allocation of police resources is by no means arbitrary, it is not necessarily optimized. In such a situation, however, a branch of mathematics can solve this major issue: game theory. Game theory can be used to model real world situations as a game between two players, and optimize strategies for those players.

While at first glance criminals may seem irrational, overall they follow a strategy to minimize their losses, thus minimizing their chances of getting caught. [6] In doing this, they act as a player, as part of a theoretical game that can be counteracted by an effective strategy on the part of the police.

Although the final call of where to allocate resources falls to the police, game theory will give the best possible mathematical strategy based on numerical data. With this, the police can then use the proposed solution, along with their own knowledge, to properly place officers as they see fit.

2 Theory

2.1 How Crime is Rational

Although the actions of individual criminals may seem irrational and unpredictable, crime as a whole often forms a distinct pattern. A number of studies, such as those by Gary Becker (1968), Marvell and Moody (1996), and Hope Corman and H. Naci Mocan (2000) demonstrate that criminals are rational beings who consider the consequence of their actions, and that increasing the number of police officers causes a reduction in crime. [6] For example, consider a situation where Player A has an intention to kill Player Bl, and Player C has an intention to kill Player D. Although individually neither Alex nor Charlie has incentive to change his target, the placement of police near Bill rather than Dan can decrease the probability that Alex follows through with his intention. Therefore when the killings are considered together, Dan, who is further away from the presence of police, is more likely to be killed. In other words, criminals are less likely to commit a crime in the presence of police. [6] As a result, the pattern of crimes can be treated as an entity that rationally avoids the police. Because of this, game theory can be used to create a strategy that effectively allocates the resources of a police force to treat this entity.

2.2 Game Theory and Nash Equilibrium

Game theory analyzes decisions in competitive situations involving multiple players. Since the pattern of crimes can be treated as rational, game theory can optimize the distribution of police by designating criminals and police as the two players, prece-
dented by the ARMOR security program in Los Angeles Airport. The ARMOR program used game theory to place checkpoints and police resources throughout the airport. The program performed far better in lab tests than other programs, and the Los Angeles Airport Police had reported very positive feedback on the program. Just as an effective strategy was found for this security issue, the same can be done for the crime in Irvington, New Jersey - the principles of game theory can be applied to find a solution that will benefit the city and minimize crime. While the methods are theoretical, they can yield success when applied practically.

One way games can be analyzed is through finding the Nash equilibrium, also called the solution. This equilibrium consists of the set of strategies that result when each player, assumed to be aware of every other player’s strategies, cannot benefit by changing his strategy while the other players keep theirs unchanged.

For example, Alice and Bob are arrested for a crime, as shown in figure 1. They are placed in separate rooms where each is offered the same deal. With no sufficient evidence to convict either of them, the police tell Bob that if he testifies against Alice and she remains silent, Bob will be released and Alice will receive a full ten-year sentence. This offer is also given to Alice except the roles are reversed. However, if they both are silent, each will be sentenced to jail for a mere six months. If both decide to testify against each other, they each will serve five years in prison. Alice and Bob must choose to either betray each other or stay silent. This example, commonly known as the Prisoner’s dilemma, contains a Nash Equilibrium. If Alice testifies against Bob and he remains silent, or vice versa, the person who remains silent will have incentive to change his or her strategy and testify—reducing his or her sentence from ten years to five years. If both remain silent, and receive six months in prison, and each have incentive to change his or her strategy and testify in order to be freed immediately instead. However, if Alice and Bob both testify and receive five years in prison, neither has incentive to change his or her strategy and receive ten years instead by revoking his or her testimony. Thus, the Nash Equilibrium is where both Alice and Bob will choose to testify.

2.3 Zero-sum Games

A zero-sum game is a game in which for every combination of strategies chosen, the sum of the payoffs for the two players is zero. In other words, for every unit of payoff one player gains from a decision, the other player must lose a unit of payoff equal in value. Thus, one player’s gain results in his opponent’s direct loss. A game between police and crime would be zero sum, since when a crime is committed, the rewards and losses, which would be personal property or lives, are equivalent in value.

2.3.1 Payoff Matrix

A payoff matrix is a matrix whose elements represent the rewards for the respective players. The rows of the matrix represent the choices possible for one player, and the columns of the matrix represent the choices possible for the other player. The intersection of each row and column represent the payoffs (or rewards) from the two players’ choices. If the elements have one value each, they represent the rewards for one of the players. Otherwise, if the elements have two values each, the first value represents the reward for the rows player, and the second value represents the reward for the columns player.

2.3.2 An Example and Its Solution

Consider a zero-sum game with the payoff matrix shown in figure 2:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>3,-3</td>
<td>-1,1</td>
<td>2,-2</td>
</tr>
<tr>
<td>Y</td>
<td>1,-1</td>
<td>2,-2</td>
<td>-2,2</td>
</tr>
</tbody>
</table>

The matrix gives the payoff of every set of choices
that the players can make. In the table, the rows represent the choices for Player 1 (X or Y) while the columns represent the choices for Player 2 (A, B, or C). The players make their choices in secret first. The choices are subsequently revealed, and each player has the payoff added to his number of points.

Player 1 could use the following reasoning: "I can lose up to 2 points and win only 2 with choice Y, but with choice X I can lose only 1 and win up to 3, so choice X looks a lot better."

With similar reasoning, Player 2 would choose choice C. However, Player 2 could predict Player 1’s choice of X and go for choice B to win 1 point. Player 1 could also predict this trick in reasoning and switch to choice Y to win 2 points. If the players continue these trains of thoughts, conclusive decisions cannot be reached because they have entered a recursive loop.

To break this conundrum, Borel and Neumann discovered that the usage of probability in the players’ strategies would solve the game and find the Nash equilibrium.[1][7] The two players can assign probabilities to each choice and leave the choice to chance. The expected payoff for each player can be calculated by multiplying the value of each element in the matrix by the probability it gets chosen, and then summing the products. In this example, Player 1 should choose X with $\frac{2}{7}$ probability and Y with $\frac{5}{7}$. Player 2 should choose A with 0 probability, B with $\frac{3}{7}$ probability, and C with $\frac{4}{7}$ probability. When Player 2 uses this strategy, Player 1 would get $\frac{2}{7}$ expected points no matter which choice he makes. Player 2 cannot reduce that expected value further because in all other probability distributions, Player 1 would always be able to make a choice that gets more than $\frac{2}{7}$ expected points. Therefore Player 2 has no incentive to change his strategy. Similarly when Player 1 uses his strategy, Player 2 would lose at least an expected value of $\frac{3}{7}$ points, which is the best possible guarantee for Player 1. Thus Player 1 also has no incentive to change his strategy. This satisfies the definition of Nash equilibrium because neither player can benefit by changing his strategy.

### 2.3.3 General Solution to Zero-Sum Games

Linear programming is used to find the probabilities needed for the Nash equilibrium of a zero-sum game. It is a mathematical method used to optimize an outcome function (such as profit or cost) based on given constraints. The constraints are usually given as a system of inequalities. Linear programming finds the solution(s) that optimizes the outcome function while satisfying the system of inequalities.

In a zero-sum game, each player would try to maximize his own payoff. For the purpose of this explanation, let one player’s name be Alice, and let $c$ be the reward she can achieve at the Nash equilibrium. If the expected reward for Alice from any of her opponent’s choices is less than $c$, then her opponent will choose that strategy to decrease Alice’s reward below $c$. However, at Nash equilibrium, her opponent’s strategy is already supposed to be optimal. Therefore the expected reward for Alice from each of her opponent’s choices must be at least $c$. This sets up a system of inequalities, one for each of her opponent’s choices. These inequalities, in addition to inequalities bounding the strategies to real-world conditions (for example, probabilities must be non-negative), make up the constraints for the linear programming. One can solve this system to optimize $c$.

Suppose a zero-sum game has a payoff matrix $M$ where element $M_{i,j}$ is the payoff obtained when the minimizing player chooses pure strategy $i$ and the maximizing player chooses pure strategy $j$ (i.e. the player trying to minimize the payoff chooses the row and the player trying to maximize the payoff chooses the column). The Nash equilibrium strategy for the column (maximizing) player can be found by solving the following linear program for the vector $u$:

Minimize

$$f(u) = \sum_j u_j$$  \hspace{1cm} (1)

Subject to these constraints:

$$u \geq 0$$ \hspace{1cm} (2)

$$Mu \geq 1$$ \hspace{1cm} (3)

Then the probability for the maximizing player to choose column $j$ should be $\frac{u_j}{f(u)}$. The first constraint (equation 2) forces every element of $u$ to be positive. The second constraint (equation 3) means that no matter what strategy the row player chooses, the expected payoff is no less than $\frac{1}{f(u)}$. Since the minimizing player is rational, he will choose the row that gives the minimum expected value, or $\frac{1}{f(u)}$. Therefore to find the optimum strategy for the maximizing player, $\frac{1}{f(u)}$ needs to be maximized, so the denominator $f(u)$ needs to be minimized. We can maximize and minimize these values by solving a system of equations through linear programming.
3 Methods

Figure 3: This flow chart illustrates our method.

3.1 Data Acquisition

First, we acquired data on Irvington. The police department provided us with information in the form of a map using ArcGIS and ArcMAP software; the city was split into roughly 4000 cells, and each cell had several pieces of data. Cells contained a number corresponding to the number of shootings, the number of drug arrests, and its risk level according to the police. The risk level from the police was calculated from the existence of shootings, drug arrests, and gang residence. Working with ArcGIS, we observed patterns in the information: drug arrests were more prevalent in areas with high risk level, and that shootings were reasonably random in their distribution. One important point we observed was that the risk level of cells seemed to have a positive correlation with population density[5]; this observation implied that population would not need to be factored into our risk level calculations, since the police risk level included its effects on crime.

Since the data from the police was confidential, we constructed our own city, Rutgers City, using the Anteland Citybuilder that we designed. The distribution of risk levels and population were designed to be similar in nature to the data of Irvington. In the city we generated for this paper, the population was 10026. Figures 4 and 5 are population and crime maps of the city.
Figure 6 shows the locations of shootings in a six-month period. Figure 7 shows the locations of drug arrests in a six-month period. Shootings and drug arrests tend to be more prevalent around areas with more crime.

Figure 6: Shootings were distributed such that they were concentrated in areas of high crime

Figure 7: Drug arrests were distributed such that they were concentrated in areas of high crime

3.2 Regions Construction and Risk Calculation

To estimate the maximum number of regions, \( n \), we could analyze, we looked at the computation time for the full process of solving the game. The construction of the payoff matrix would take a computation time of \( O(n^2) \), where big-\( O \) means the time is at most a constant times the function inside the matrix. After the construction of the payoff matrix, the linear programming optimization would take a computation time of \( O(n^3) \) using most well-known algorithms. We estimate that our processors can compute constant multiples of about \( 10^{12} \) operations within an hour, so it is only realistic, considering the computation time of the linear programming step, to use \( n < 1000 \). Thus to properly analyze the data, we needed to combine cells to make regions.

We split our regions based on the scope of the effect of police presence in an area. Police can significantly reduce crimes in the same block as police presence,[6] so a reasonable region size would be the size of a city block. City blocks tend to be about 500 feet long on average, so we defined adjacent 500 feet \( \times \) 500 feet square areas as our regions. For cities like Irvington, the number of these regions is less than 400, which is within our limit.

After we decided on our new regions, we had to decide how to calculate their risk levels. Because there is little evidence suggesting that the placement of police in one block can affect crime in another block[6], we decided to calculate the initial risk levels for each region by averaging the risk levels of only the cells within the region (as opposed to including cells of adjacent regions). To continue in our approach, we added the total number of shootings in a six-month period to the risk level of each region because a prior shooting would increase the probability of another crime occurring in the same area since crimes tend to cluster in the same areas. We also added a fraction of the number of drug arrests in a six-month period into the risk level of each cell because areas with drug arrests also tend to have violent crime. [2] We used the ratio of total shoot counts to total drug arrests as the fraction so that drug arrests are not weighted more than shootings.

We constructed the regions by using MATLAB to assign cells to regions based on their X and Y coordinates (longitude and latitude). We also used MATLAB to calculate the risk levels as described and the coordinates of the centers of the square regions.

3.3 Setup of the Police versus Criminals Game

We created the payoff matrix for this zero-sum game of police and crime using weighted averages for the rewards. The rewards for the police are calculated from a weighted average of the reward from preventing a crime and the loss from failing to prevent a crime.
Let $D$ be the reward of preventing a crime in a region and $P$ be the weight of that reward. Then $-D$ is the loss from failing to prevent a crime and $1 - P$ is the weight of that loss. The weighted average of the possibilities is $(2P - 1)D$.

In formal terms,

For $i, j \in \mathbb{N}; i, j \leq n$ where $n$ is the number of regions analyzed, the payoff matrix for the police is

$$[(2P_{ij} - 1)D_j]$$

where

- $i$ is row number and $j$ is column number (the police’s strategies are represented by rows, and crime’s strategies are represented by columns).
- $D_j$ is the potential of a crime in region $j$, based on risk level.
- $P_{ij}$ is a number between 0 and 1 of the ability of a police presence in region $i$ to prevent crimes in region $j$, where 0 means no ability and 1 means the police can prevent all crimes in region $j$. When $i = j$, we initially let $P_{ij} = 0.75$ because it has been shown that the police reduce car thefts, on a block by about 75% for one police officer. [6] For the purpose of this model we assume that crimes are in general similar to car thefts in their responses to police placement. However, when the population is higher than average, we reduce the $P$ value because it is difficult for the police to work in highly populated regions. For the same reason, we increase the $P$ value when the population is lower than average. When $i \neq j$, we let $P_{ij} = 0$ because there is insufficient evidence that the placement of police on one block has a significant effect on crime in neighboring blocks. [6] The effect of police response time on a region’s risk level is also not a consideration, since Irvington is a relatively small town. [5]

### 3.4 Solving the Game

To solve the game, we used MATLAB to apply the algorithms combining the regions, setting up the matrices, and then doing linear programming as described earlier. A copy of the MATLAB code used can be found in the appendix. We reapplied this approach repeatedly with different numbers of regions, up to 600. We also reapplied our algorithms with different ways to calculate the risk values of the regions. After each solution was found, we plotted it using 3D graphing to make sure the results were viable. The value of the game, the expected reward for playing, was recorded as a measurement of the performance of each attempt.

### 4 Results

The algorithm we built showed that our model was around $19\pm1\%$ more effective than a uniform distribution of police. It was also $18 \pm 1\%$ more effective than a distribution of police proportional to calculated risk levels. This percentage is calculated by dividing the improvement in expected reward by the magnitude of the reward in the distribution we compared our model against.

Figure 8 gives the result of the game, using all the previously shown factors to calculate the optimal allocation of additional police resources. The game tends to give higher weights to areas with higher likelihood of crime.

![Figure 8: A mesh graph representing our resulting police distributions.](image-url)
Figure 9 represents the same output values as figure 8, but portrayed in a 2-dimensional graph.

Figure 9: A 2D image representing our resulting police distributions.

4.1 Analysis

Our algorithm often predicts that it is most beneficial to place all the police in a few cells and leave others completely empty. This is because it uses data of the crime levels at the current status quo. It assumes that the values do not change significantly through the game. Thus our results are recommendations on the most efficient proportions for allocating small amounts of police resources, such as additions to the police force, not on the redistribution of the entire police force. Additional data would be needed after the reallocation of resources to further adjust for changes in crime levels.

Despite these setbacks, the runtime of the algorithm is relatively low for most cities. Given the data, the algorithm is simple to use and appears to consistent and predictable. The algorithm deterministically produces the same result each run for any input, and it consistently outputs high police distributions near high crime areas. This consistency indicates that the algorithm is reliable. In addition, the appearance of high values near areas of high crime indicates that the results of the algorithm are reasonable. This appears to be true in all 8 cities that we have tested on.

By applying Game Theory to reduce crime rates in Irvington, we were able to predict potential payoffs. The knowledge of the police force is also strategically important to the percent allocation of resources. Since too many random variables exist in real life, our results are merely suggestions that should guide the police, not supersede their authority. Police resources can be allocated in percentages according to our results, but how that is done is up to the department. As shown by the improvement in expected reward, our model can be reliably used to optimize police resources to combat crime.

4.2 Limitations

Our data was limited by what the police could provide us with, so for security reasons, some of the more recent data was withheld. Our data is therefore slightly out of date. However, the police can use this model on up-to-date data to find an efficient way to allocate additional police resources.

Computation time was also an issue. A 600 cell game took roughly 3 minutes to solve, and the algorithm takes \( O(n^4) \) time, so a 4000 cell game would take about 2000 times as long as the 600 cell game, or 100 hours. Thus, we had to decrease the number of cells we had, in order to shorten computation time to a reasonable time frame.

Our model used an assumption that police response time would not be an issue for regions with large distances between them. For a city like Irvington, this may hold, but when applied to cities with larger areas, our model may need to be modified such that \( P \) values are slightly negative for regions \( i \) that are far away from regions \( j \). This modification can easily be done in the MATLAB code by adding in a section to precreate a \( n \times n \) matrix of values for \( 2P - 1 \). Then our reward matrix creation section can be changed to just multiplying every column of the precreated matrix with the calculated risk for the corresponding region of the column.

Our results were also limited by the fact that our model can only be accurately applied for small amounts of police resources. This can also be accounted for by increasing the reward for the cells where the police have a success. Replacing \((2P − 1)D\) with \((k + 1)P − 1)D\) can provide a reasonable approximation, where \( k \) is a constant proportional to the amount of police resources. Another way to overcome this limitation is by rerunning our model numerous times, once for every unit of police resource to be assigned. Each successive run can take into account the effect of the allocation of police resources in the previous runs by multiplying each risk level from the previous run for each region by \((1 − P)^p\), where \( p \) is the proportion of resources assigned there. The results could be added up from every single run. We have conducted primitive simulations that have shown that both methods produce similar proportions.
5 Conclusion

We have built an effective algorithm, as shown by our comparison to a uniform distribution of police. With a 19 percent increase in effectiveness, our model is one that can reduce crime a great deal if applied in Irvington.

The results we have are promising. In theory, the game is the optimal placement of the police in Irvington, using the data we have. Therefore, it could prove to decrease crime significantly, since the police will be in the best positions to deter crime. Steps will need to be taken, such as studies to see whether or not this model actually functions optimally. If it does, it can be used to build more games in the future, including more effective dynamic games. These games may perhaps turn out to be an asset for the police in their fight against criminals.

5.1 Recommendations for Future Work

Our first and foremost recommendation is the creation of a dynamic game. A dynamic game is a game in which decisions are made more than once, over and over again, where each player takes into account previously made decisions. In the situation of the police and criminals, this would allow the police to determine how to change allocation of police resources as crime adapts to the initial placements. Our static game is only useful for initial placements of the police. If the police department felt the need to change how police resources are distributed, they would need a dynamic game to model that situation. Therefore, we believe that a dynamic game would be very helpful to the police in the future.

Furthermore, more controlled studies need to be conducted. With crime, it is very difficult to hold controlled studies; however, if an attempt was made, data on several key variables can be acquired. Such pertinent information as the effect of police on a city block, and how they effect drug trade and shootings in their vicinity, would be very useful. More data would substantially help in constructing games in the future, because it improves overall accuracy and provides a real-world scale to evaluate our results.

Data on other types of crime are also helpful. While shootings and drug trade are important to prevent, preventing crimes such as shoplifting is also essential. Optimizing police placements can possibly prevent shoplifting and other crimes; however, there is not enough data to support this, and if those studies can be done, the game can be reconstructed to account for those variables.

A major recommendation is also to test our game in both the lab and in the field. While our game is determined to be effective with the data that we have given it, it may function very differently if implemented in real life. Studies need to be done on the viability of this game. If the game is viable, then it can be fully implemented and eventually expanded upon. If the game is not viable, then it can be adjusted so that it can be implemented properly or abandoned entirely to create a new game. It is up to independent parties and the police department to test the game extensively.

6 Acknowledgements

We would like to thank our project mentors, Zhe Duan and Professor Melike Gursoy, for allowing us work on this project. We would also like to thank the Governor’s School staff for guiding us throughout this amazing journey. We would like to thank Rutgers University for providing us with the resources we needed. We would also like to extend our thanks to the police department of Irvington, New Jersey for providing us with data. Finally, we would like to thank the Governor’s School of Engineering and Technology, for giving us the opportunity to work on this project.

References


7 Appendix

Region Divider and Game Solver:

```
%input city data into "data" with columns "ID X Y Population Risk
%Level Shoot Count Drug Count"
%CombinedCells is combined cell data with "ID X Y Population CellCount
%Avg Risk Level Total Shoot Count Total Drug Count" as Colheaders
%Answer is ZeroSum Game result with ID, X, Y, Police, Crime
%rskLvlMat is Reward Matrix for Police with Police choosing Rows

%set m for number of 100feet per cell,
%p for average probability for police to prevent crime
%Process XY Data
m=5; %5 per block
p=0.75;

X=data(:,2);
Y=data(:,3);
XStart=min(X);
YStart=min(Y);
X=(X-XStart)./100;
Y=(Y-YStart)./100;
ydim=(floor(max(Y)/m)+1);
xdim=(floor(max(X)/m)+1);

%Set up Combined Data
RegionsLocs=zeros(xdim*ydim,3);
RegionsData=zeros(xdim*ydim,5);

%Combined XY coordinates
k=0;
for i=1:xdim
    for j=1:ydim
        k=k+1;
        RegionsLocs(k,:)=k-1 (i*m-m/2)*100+XStart (j*m-m/2)*100+YStart;
    end
end

%Combine rest of the data
for i=1:size(data,1)
    RegionsData(floor(X(i)/m)*ydim + floor(Y(i)/m) + 1,:)= ...
    RegionsData(floor(X(i)/m)*ydim + floor(Y(i)/m) + 1,:)+horzcat([1] , data(i,4:7));
end
clear CombinedCells;
CombinedCells=horzcat(RegionsLocs,RegionsData);

i=0;
while (i<size(CombinedCells,1))
    i=i+1;
    %Remove Empty Regions
    while (i<=size(CombinedCells,1) && CombinedCells(i,4)==0)
        CombinedCells(i,:)= [ ];
    end
    %Average risk level and assign IDs
    if (i<=size(CombinedCells,1))
```
CombinedCells(i,6)=CombinedCells(i,6)/CombinedCells(i,4);
CombinedCells(i,1)=i-1;
end
m=size(CombinedCells,1);

%Apply game

%Calculate New Risk Levels
shootdrugRatio=sum(CombinedCells(:,7))/sum(CombinedCells(:,8));
%Use 3x Avg RskLv1 + shootings + drugcount*shootdrugRatio
NewRskLv1=CombinedCells(:,6).*3+CombinedCells(:,7)+CombinedCells(:,8).*shootdrugRatio;

%Build Reward Matrix
denom=sum(CombinedCells(:,5))/m; %Make p based on population --> 0: p=1, avg: p=preset p.
A=transpose(CombinedCells(:,5)).*1;
rskLvlMat=A;
rskLvlMat(1,1)=rskLvlMat(1,1)*(1-2*(p^(CombinedCells(1,5)/denom)));
for i=2:m
    rskLvlMat=vertcat(rskLvlMat,A);
    rskLvlMat(i,i)=rskLvlMat(i,i)*(1-2*(p^(CombinedCells(i,5)/denom)));
end

%Solve Zerosum game
A=rskLvlMat.*100;
[m,n]=size(A);
X_a=linprog([-1;zeros(m,1)],[ones(n,1) -A'],[],ones(1,m),[]);X_a(1,:)=[];
X_b=linprog([1;zeros(n,1)],[-ones(m,1) A],[],ones(1,n),[]);X_b(1,:)=[];
C=roundn(X_a,-6);
D=roundn(X_b,-6);
Answer=horzcat(CombinedCells(:,1:3),C,D);
Citybuilder:

%Anteland Citybuilder v1.0
%Output data columns are ID, X, Y, Population, Risk Level, Shoot Count, Drug Count

%Lowerleft position
Xstart=575000+rand*16000;
Ystart=634000+rand*16000;

%lay city
CellPos=zeros(73,86);
CrimeMat=zeros(73,86);
PopMat=zeros(73,86);
ShootMat=zeros(73,86);
DrugMat=zeros(73,86);

%lay blocks
for i=1:15
   CellPos(i*5-4,:)=zeros(1,86)+1;
end
for i=1:17
   CellPos(:,i*5-4)=zeros(73,1)+1;
end
k=1.85;

%Set Boundaries (Cut corners)
ul=tan(pi/4*rand+pi/4*rand);
ulc=[rand^(1/2)*7.3*k 86-rand^(1/2)*8.6*k];
ulb=ulc(2)-ul*ulc(1);
ur=-tan(pi/4*rand+pi/4*rand);
urc=[73-rand^(1/2)*7.3*k 86-rand^(1/2)*8.6*k];
urb=urc(2)-ur*urc(1);
lr=tan(pi/4*rand+pi/4*rand);
lrc=[73-(rand^(1/2))*7.3*k (rand^(1/2))*8.6*k];
lrb=lrc(2)-lr*lrc(1);
ll=-tan(pi/4*rand+pi/4*rand);
llc=[rand^(1/2)*7.3*k rand^(1/2)*8.6*k];
llb=llc(2)-ll*llc(1);
for i=1:73
   for j=1:86
      if not(((i-ul+j)<ulb) & & ((i*ur+j)<urb) & & ((i*ll+j)>llb) & & ((i*lr+j)>lrb) )
         CellPos(i,j)=0;
      end
   end
end

%Seed population centers
for i=1:17
   x=floor(rand*73+1);
   y=floor(rand*86+1);
   m=12.5+(rand^(1/3))*12.5;
   hx=0.0105+rand*0.004;
   hy=0.0105+rand*0.004;
   for j=1:73
      CellPos(i,j)=m;
   end
end
for k=1:86
    PopMat(j,k)=PopMat(j,k)+(0.70+0.60*rand)*m* ... 
    exp(-(0.80+0.40*rand)*(1.95*hx*((j-x)^2)^0.75+1.95*hy*((k-y)^2)^0.75));
end
end
end
PopMat=1.55+PopMat./5.4;

%Seed crime centers
for i=1:12
    x=floor(rand*73+1);
    y=floor(rand*86+1);
    m=12.5+(rand^(1/3))*12.5;
    hx=0.0135+rand*0.004;
    hy=0.0135+rand*0.004;
    for j=1:73
        for k=1:86
            CrimeMat(j,k)=CrimeMat(j,k)+0.83*(0.70+0.60*rand)*m* ... 
            exp(-(0.80+0.40*rand)*(1.95*hx*((j-x)^2)^0.75+1.95*hy*((k-y)^2)^0.75));
        end
    end
end
avg=sum(sum(CrimeMat))/73/86;
CrimeMat=CrimeMat.*(25/avg);
CrimeMat=CrimeMat.^0.5;

%Place population and crime on city blocks
for i=1:73
    for j=1:86
        PopMat(i,j)=floor(PopMat(i,j)*CellPos(i,j));
        CrimeMat(i,j)=CrimeMat(i,j)*CellPos(i,j);
    end
end

%prepare Result Matrix
clear CellData;
colheaders={'ID', 'X', 'Y', 'Population', 'Risk Level', 'Shoot Count', 'Drug Count'};

%Combine data into Result Matrix
k=0;
for i=1:73
    for j=1:86
        if (CellPos(i,j)>0)
            k=k+1;
            ShootMat(i,j)=floor((rand^7.5)*(3800+CrimeMat(i,j)^4)/3726);
            DrugMat(i,j)=floor((rand^5)*(1100+CrimeMat(i,j)^4)/1070);
            CellData(k,:)=[k-1 i*100+Xstart j*100+Ystart PopMat(i,j) ... 
                            CrimeMat(i,j) ShootMat(i,j) DrugMat(i,j)];
        end
    end
end
population=sum(sum(PopMat))
%Graph City
figure('Name','Population Distribution')
imagesc(PopMat);
figure('Name','Risk Levels')
imagesc(CrimeMat);
figure('Name','Shoot Counts')
imagesc(ShootMat);
figure('Name','Drug Counts')
imagesc(DrugMat);

%Place data into "data" for CrimeGame analysis
clear data;
data=CellData;