PLOCCK: Police Location Optimization of Chicago Crimes using K-Means

Sara Awid, Inah Canlapan, Jovan El Hoayek, Jack Moore, David Natta

Introduction & Problem Definition

Established in 1835, the Chicago Police Department has existed longer than Chicago has been recognized as a city (Chicago Police Department - History). Over almost 2 centuries, the city of Chicago has created districts and built police stations in response to their growing population and expanding city bounds. With historical crime data now available for the city and given that the city has not expanded their boundaries since the 1960's (Population & Annexation), we believe there is a more sophisticated way of determining where Chicago's police stations should be located.

We will use Chicago's crime data to find ideal locations for their police stations. We intend to cluster the crime data to show the city's predicted crime hotspots relative to where the existing police stations are. By visualizing the distance, we can clearly see if the existing stations are optimized to be in the center of the crime zones. We aim to give Chicago's municipal government, urban planners and the Superintendent of Police a tool that will show them where they should consider placing police stations given the existing and predicted crime hotspots. By building stations closer to where crime occurs, they can deploy their forces in a more timely and efficient manner and in turn, reduce their overall crime rate.

Literature Survey

Liberatore et al. introduce the objective of the Police Districting Problem (PDP) as the "optimal grouping of blocks into 'homogeneous' patrol sectors in such a way that all the territory is partitioned and that no sector is empty." Our project objective is similar but we want group crime, not patrol sectors. This paper provides a generalized problem definition and references other literature to summarize possible approaches. It does not present a new solution of their own.

Chow et al. views PDP as an optimization problem and proposes using the maximum coverage and p-median approach as a way to minimize the distances and maximize the coverage of police stations over potential crime spots. This approach heavily relies on ward boundaries and other GIS input that are specific to the region they chose; thus, the solution cannot be extended beyond the chosen region.

Similarly, Curtin et al. also views PDP as an optimization problem but includes a geographic optimization approach, borrowing concepts from the Maximal Covering Location Problem where the objective is to maximize the number of calls covered by a fixed number of patrol areas that are limited to a specified size. It uses the number of calls as an estimate of officer workload and does not consider response time. It also assumes the number of patrol areas are known in advance, rather than provide feedback on how many patrol areas there should be.

De Gusmão et al. propose a solution to PDP that uses k-means clustering. This solution closely aligns with our project objective and provides us with a framework we can use to evaluate the performance of our model. However, their solution input parameters were tailored specifically for Brazil and is not a scalable solution we can easily apply to other geographies.

Liao & Guo use an adapted k-means algorithm and frames PDP as a Capacitated Facility Location Problem where facilities are police stations and capacity is the number of police officers at a station. Their solution requires that all facilities be movable and swappable and that each demand has the same cost in capacity. In real life, this is not the case - police stations are not easily movable or swappable and the type of crime (demand) may require more than once police officer (capacity).

Hochstetler et al. also uses k-means to find crime hotspots on an hourly basis. They thoroughly explain issues with the data set (i.e. invalid coordinates, incomplete data), which gives us insight into what kind of issues we could encounter with our own data. They do not consider existing police departments and use real-time traffic data to calculate distance between crimes. This data is very hard and expensive to obtain.

Mitchell uses clustering to define police patrol areas (beats) for a city and compares his results with real police beats. The comparison between police and computer designed beats proved that computers outperform police officers in reducing travel distance. We can use a similar approach in our solution, comparing the algorithm results vs the existing locations of police stations. Mitchell's solution, however, does not use longitude and latitude coordinates which are more precise and can likely yield better results.

D'Amico et al., used queuing models and simulation as part of their solution and provided a stochastic optimization method that could help inform our team's approach. The objective of this paper, however, was to minimize workload disparity among policing districts and does not necessarily maximize response time or coverage as we aim to provide as part of our solution.

The machine learning application of predicting crime hotspots have been extensively reviewed with several works exploring ways to identify crime patterns and map crime density. A more modern approach to use deep learning and advanced analytics was discussed by Zhang et al. (2020), Kang et al. and Kennedy et al. The common approach was to use advanced techniques like Risk Terrain Modeling, Long Short-Term Memory Neural Networks and Deep Neural Networks to identify crime hotspots and used non-conventional data such as Google images and social, physical and behavioural aspects of neighbourhoods to create features for these models. Although these data sources can enhance the final results of our project, they do require a great deal of complexity in terms of data collection. The complexity of these advanced techniques also makes it difficult to explain our methodology to our potential stakeholders.

Zhang et al. (2010) & Garima & Alaid provide history and background on existing methods like spatial mining and k-means clustering for crime spot detection. Garima & Alaid concluded that spatial clustering is indeed a useful method while Zhang et al. (2010) discussed its limitations and suggested new attribute-oriented clustering to overcome them. Both approaches are useful in providing us background information, however the real-world effectiveness of their methodologies is not measurable. Wang et al. introduces a solution that uses spatial mining and includes crime related factors using geospatial discriminative patterns to predict crime hotspots. They provide a heatmap which can be useful for our visualization but their methodology is very complex and difficult to scale.

Nath used k-means to identify crime patterns or crime sprees within a police jurisdiction. Some of the attributes used required crime analyst input and recommendations from subject matter experts on the importance of different attributes which is not possible in our application.

Zhu & Xie provided a novel approach using natural language processing to extract features from raw text in police data to identify crime patterns. Those features were then mapped to measure similarities among crimes and could predict if such crimes are being committed by the same criminals. This approach relies on clear, written descriptions of crime in order for us to accurately extract patterns from data.

Proposed Method

In our proposed method, crime data is used in a novel weighted K-Means algorithm that takes user input into consideration as parameters to the algorithm. Our approach assumes that all locations within Chicago city bounds are available and that the crime location distribution stays constant over time. The user is able to select the number of desired police stations, specify the date range of the data used to create the clusters and assign weights to different crime types based on how important it is to have police present at the crime scene. They can also pin certain locations if there are locations they want to fix and have as part of the suggested police station list returned to them.

Innovation #1

The ability for our solution to consider user input is unique among the solutions presented in our literature review; it allows our solution to be interactive and flexible based on the expertise of the person using it. The dynamic time frame selection also gives the user the ability to see what the locations would be like at different points in time, instead of static snapshots that the analyses in the papers we reviewed had given. Furthermore, the ability to apply weights to certain crimes types and pin locations were not explored in related works, making our solution truly innovative.

The user interface is a website built using React.js and the Mapbox library. Once the user sets their parameters, they are sent to a Flask app hosted on an EC2 instance in AWS. The Flask app runs the customized k-means algorithm using crime data housed in a MySQL database. Once completed, a list of latitude/longitude coordinates are passed to the front end and are rendered on the map.

Innovation #2

By hosting our back-end in AWS, we can easily package our entire solution in a single EC2 instance and leverage the scalable computing power that it offers. Having the crime data stored in a MySQL database in the EC2 instance also makes it simple to add new data, making it possible to get a real time assessment of where crime hotspots are.

User Interface

The final user interface can be seen in Figure 1 and Figure 2. In the settings menu under “Select Date Range”, the user can drag the pins to indicate which years of crime they want to use in the analysis. Under “Number of Departments”, the user can drag the pin to indicate how many police stations they want returned to them. By clicking “Set Crime Weights”, a pop-up menu appears where the user can drag and drop crime types into crime weight buckets based on how urgent police presence is required at the scene. Crimes not placed in buckets will not be considered in the analysis. The user can double click any point on the map and a purple pin will be placed, indicating that the location has been pinned. To unpin the location, the user can simply double click the purple pin the remove it. Once the user parameters have been sent, they can hit “Apply” to kick

off the machine learning algorithm. The suggested locations will appear as the blue pins on the map. The user can hover over the pins to see its exact longitude/latitude coordinates. Under "Layers", we've given the user options to toggle different map layers on or off. Users will have the option to see where existing police stations are located as red pins, the existing police district boundaries and the suggested district boundaries returned by the analysis.

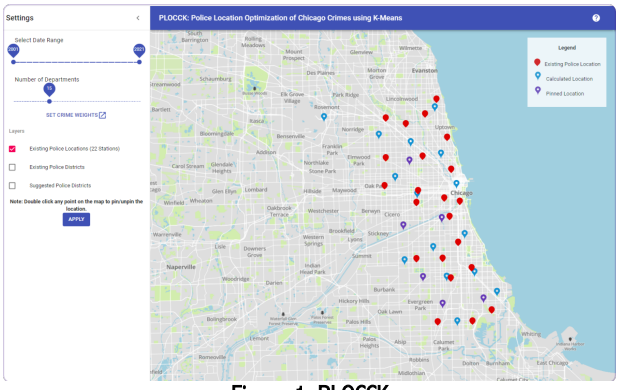

Figure 1: PLOCCK

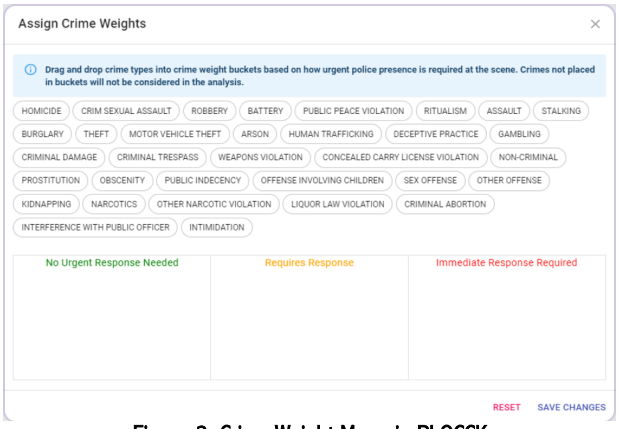

Figure 2: Crime Weight Menu in PLOCCK

Experiments & Evaluation

For the purposes of this analysis, we assume that the Manhattan distance between a police station and reported crime scene is an accurate proxy for response time; the closer a police station is to the reported crime scene, the faster the police response time will be. We had initially used the Euclidean distance, but decided to use the Manhattan distance since city streets are typically organized in a grid.

Our key performance metric is the average distance of crime to the police station. Our solution will be successful if the average distance from crimes to our algorithm's suggested police stations is lower than that of existing police stations. To find the average distance of crimes to the suggested police stations, we set the user parameters to use a sample size of almost 72,000 crimes (~1% of the crime data) from all years with equal crime weighting and returned 22 locations, the number of existing police stations. To find the average distance of crimes to existing police stations, we measured the distance between the crime location and the location of the police station of the ward in which the crime occurred. We assume that if the crime occurred in a certain ward, the police station in the same ward would dispatch an officer. Based on the average Manhattan distances calculated, the stations generated by PLOCCK reduce the average distance to crime by 27.2%, from 2.4 km to 1.7 km. (See Figure 3)

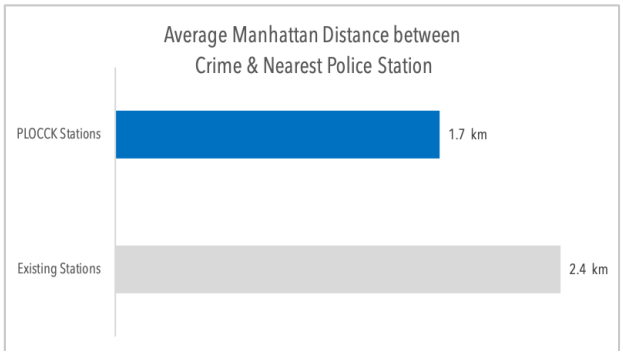

Figure 3: Difference in Average Manhattan Distance between PLOCCK and Existing Stations to Crime

Testing the User Experience

In testing the user experience, we wanted to make sure that the results were returned in a reasonable amount of time; the less time a user is waiting for results, the better. We had run some preliminary tests with the MVP (minimum viable product) of our solution that allows for simple k-means clustering that uses an EMR instance in AWS. Using the EMR, the user would have to wait 2-3 minutes after selecting the number of clusters before seeing the results displayed on the map. In an attempt to return results faster, we tested a version of our solution where we did not use the EMR instance and ran the machine learning code directly in the EC2 instance. The second version was much faster (See Figure 4), but requires us to increase the RAM of the EC2 instance in order to accommodate the size of the data, which is around 7 million rows (See Figure 5).

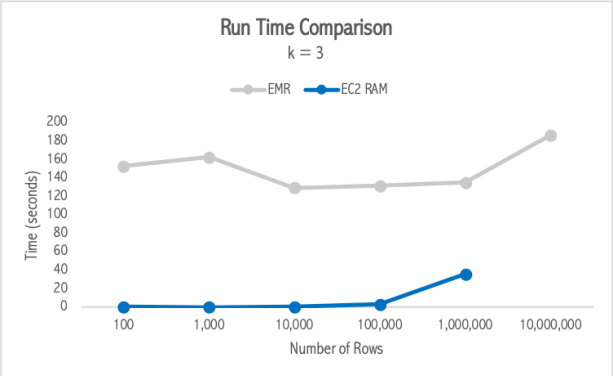

Figure 4: Run Time Comparison of EMR vs EC2 RAM

EC2 RAM takes 1-2 seconds to return results when less than 1 million records are used. No results were returned for the EC2 RAM using 10 million rows because of a memory error.

Number of Rows	Clusters	Clustering Time (seconds)
100	3	0.07
1,000	3	0.05
10,000	3	0.12
10,000	10	0.26
100,000	10	1.67
1,000,000	10	25
4,000,000	10	MemoryError

Figure 5: Clustering Time using EC2 RAM

Using more than 4 million rows results in a memory error.

At the time, we thought that we needed more time to set up a new EC2 instance with more memory and reconfigure all the connections to the front end. This would have also meant a significant increase in cost if we chose to do this. Using the results from our test that uses a sample of the crime data instead of using the entire data set (see "Testing the Algorithm"), we had determined that we didn't have to increase the amount of data in order to pin locations or implement crime weights so we no longer had to consider memory issues when choosing between the two implementations.

When we tested our final algorithm using all the features (selecting the number of locations, selecting the start and end year, apply weights by crime type, and pinning locations) and a sample size of 100,000 appropriate data points, the EMR version took 218 seconds to run on average and the EC2 instance only took 41 seconds (See Figure 6). Based on this difference, we decided to use the existing EC2 instance as it returns results 5 times faster than the EMR.

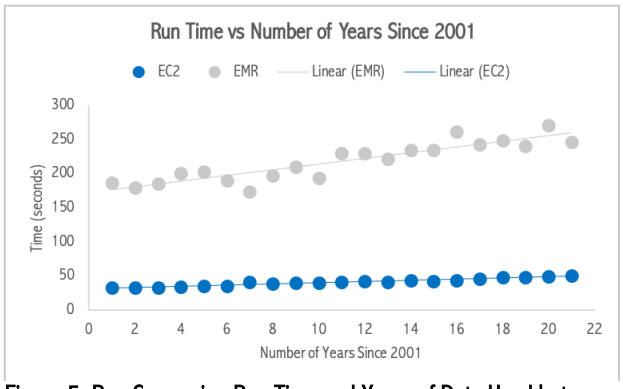

Figure 5: Run Comparing Run Time and Years of Data Used between EC2 and EMR

Even though we chose the EC2 implementation for the purposes of this project, we wanted to extrapolate the run time data and see if choosing the EMR implementation would have been better at any point in time. Using first year of data as the baseline, we plotted the relative percentage increase in run time as more years of data is being used (See Figure 6). It's clear from this graph that EMR scales better than EC2 since the rate at which run time increases as the quantity of data also increases is less than that of EC2. Thus, if more data becomes available it will, at some point, be better to use EMR rather than EC2.

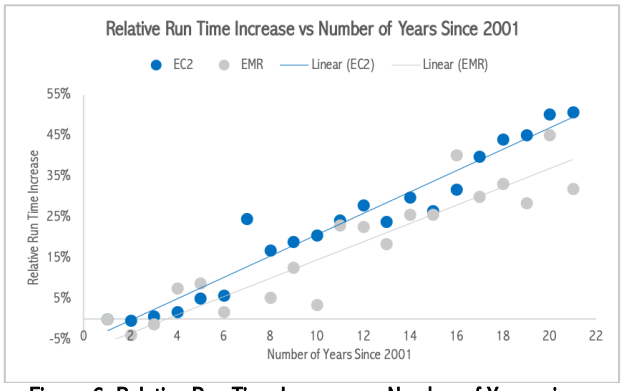

Figure 6: Relative Run Time Increase vs Number of Years since 2001

Testing the Algorithm

In the early stages of developing our algorithm, we had thought that in order to emulate a fixed location and to implement the crime weights, we needed to add multiples of the corresponding crime data in order to force a cluster in the chosen spot or increase the impact of certain crimes over others. When we tried to implement this, we ran into memory errors using both the EMR version and the EC2 version of our backend – this was not a feasible implementation for our project.

After more research, we decided to utilize the sample_weight parameter in the sklearn.k_means module and gave pinned locations a weight of 1,000,000,000 and data points in the "No Urgent Response Needed", "Requires Response" and "Immediate Response Required" buckets a weight of 1, 2 and 3, respectively. If a crime type was left out of a bucket, all crimes of that type are given a weight of 0. The weight for the pinned locations were easy to visually verify; it is harder to test the crime type weights as the k-means algorithm handles them by itself. Since the weights are passed properly in the code and assuming that PySpark and SKLearn have no bugs, we are confident that the crime weights are applied properly.

In an effort to decrease user run time and manage memory issues in both versions of our implementation mentioned in "Testing the User Experience", we wanted to test if taking a random sample of the crime data would give close enough results to using the whole data set. K-means was run on the full data set with k=15 to get the best police department locations. K-means was then run using 14 different sample sizes with k=15 and centers initialized at the best locations determined by the entire data set. 1000 trials were executed for each sample size. In each trial, the distance error – the distance between the centers found in the full data set and the centers found for the sampled data set – was calculated. These distance errors were then averaged across all 15 stations to get an average distance error from the ideal location for each trial in each sample size. Detailed results for this test are seen in Figure 7 and 8. We assume that that similar results would be achieved by executing this study with different centroid initializations or different number of cluster centers when running the initial algorithm on the full dataset.

In our final results, we found that a sample size of 5000 will result in sampled cluster centers to be within 500 meters of the population cluster centers. Sample clusters generated with sample size of 100,000 with be within 100 meters. We also tested different sample sizes and measured their run time (See Figure 9). A sample size of 100,000 results in a run time of 54 seconds. Increasing the sample size to 250,000 may increase the accuracy of the cluster center but significantly increases the run time to 143 seconds. We felt comfortable using a sample size of 100,000 in our algorithm because the difference between sample and population cluster centers is quite small and the return time to the user is not too long.

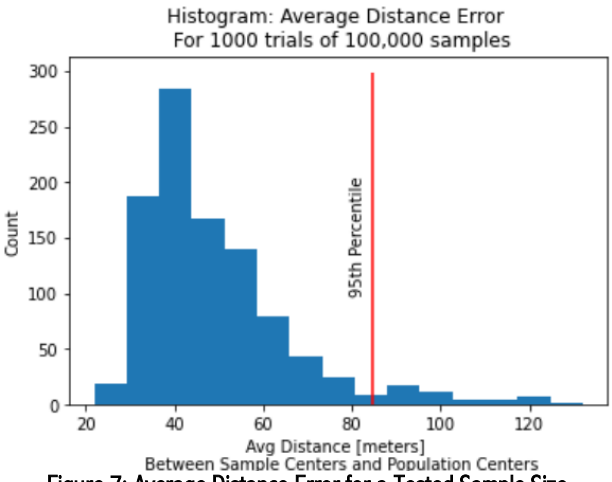

Figure 7: Average Distance Error for a Tested Sample Size

The histogram shows non-normality so we used the 95th percentile to represent an upper error limit, as opposed to using standard deviation.

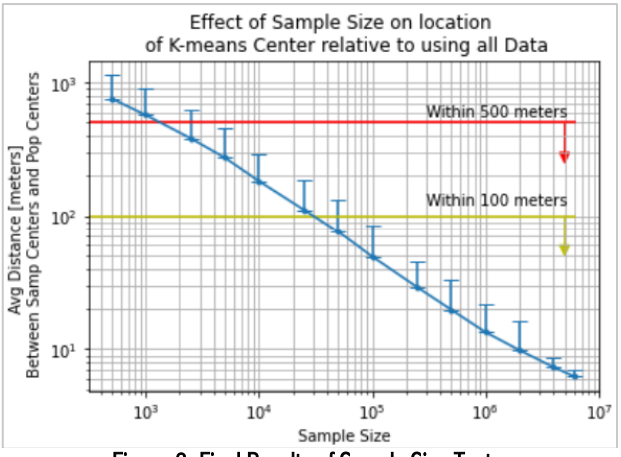

Figure 8: Final Results of Sample Size Tests

A sample size of 5000 will result in sampled cluster centers to be within 500 meters of the population cluster centers. Sample clusters generated with sample size of 100,000 with be within 100 meters.

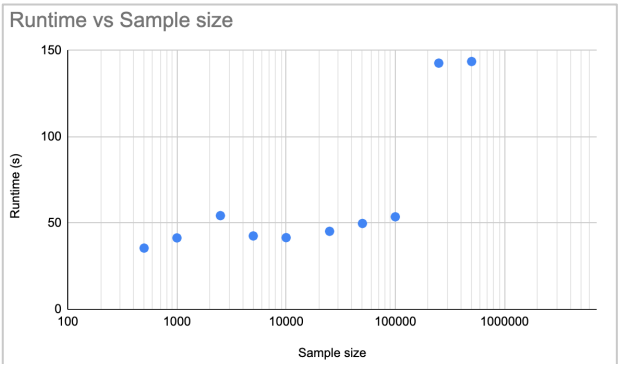

Figure 9: Sample Size vs Runtime

There is a significant increase in run time when the sample size is 250,000. Any sample size greater than 1,000,000 will results in memory errors in our existing infrastructure.

Conclusion and Discussion

PLOCCK is truly the first solution of its kind, combining scalable computing, machine learning and the expertise of the user to reduce crime in Chicago. It successfully combines an underlying k-means based algorithm with an intuitive user interface. The tool can be deployed as a web app, operating off of a low cost (free tier) AWS EC2 instance, requiring only a browser. More importantly, using the tool does not require background knowledge on how it works or is set up. While there are other factors like budget, location size and site availability that are important to consider when deciding where to place police stations, PLOCCK provides invaluable insights that would be rather difficult or complex to collect and/or understand without it.

Future Considerations

Given more time, we would have liked explore different machine learning methods (i.e. – different clustering methods or optimization implementations) that are more accommodating of the data we have. In the front-end visualization, we would have liked to add more detail in the tool tip that appears when the user hovers over a pin. It would be useful to add insights like the top types of crime in the district or notable crimes that may have happened in the given timeframe. To improve the user experience, it would have been ideal to include a step-by-step tutorial when a user first uses the tool. In the interest of time, we have included a help menu in the top right corner with instructions on how to use PLOCCK instead.

For the purposes of this project, we felt comfortable taking a sample of the data as it allows for a faster run time without impacting accuracy too much. Should more financial resources and more data become available, it may be worth it to further test the EMR implementation with different configurations as it scales better and provides better precision. The EMR implementation could also be useful should the stakeholders want to connect to real time data. Should a connection to real time data become available, this would slightly change the objective of PLOCCK; perhaps it would instead provide the optimal police officer location as it would be able to predict crime hotspots based on crimes in the last 24 hours or even the last hour.

Project Work Distribution

All team members have contributed a similar amount of effort to the project in its entirety. David was responsible for the development of the user interface. Jovan led the development of the customized k-means algorithm and worked closely with David to ensure seamless communication between the front and back ends of our solution. Jack was responsible for the data cleaning and storage and provided the sample size analysis. He worked closely with David and Jovan to build the project infrastructure in AWS. Sara provided the initial designs of the user interface, assisted Inah in writing the proposal document and provided the project's key performance metric analysis of the average distance to crime. Inah wrote and submitted all the project deliverables, including the proposal document, proposal presentation, progress report, poster and final report.

References

Chicago Police Department - History. (n.d.). Retrieved March 08, 2021, from https://home.chicagopolice.org/about/history/

Chow, A. H., Cheung, C. Y., & Yoon, H. T. (2015). Optimization of police facility locationing. Transportation research record, 2528(1), 60-68.

Curtin, K. M., Hayslett-McCall, K., & Qiu, F. (2010). Determining optimal police patrol areas with maximal covering & backup covering location models. Networks & spatial economics, 10(1), 125-145.

D'Amico, S. J., Wang, S. J., Batta, R., & Rump, C. M. (2002). A simulated annealing approach to police district design. Computers & Operations Research, 29(6), 667-684.

De Gusmão, A. P., Da Costa Borba, B. F., & Clemente, T. R. (2020). Management information system for POLICE facility location. Decision Support Systems X: Cognitive Decision Support Systems & Technologies, 86-98

Garima, A., & Alaiad, A. (2019, June). Crime analysis in Chicago city. In 2019 10th International Conference on Information & Communication Systems (ICICS) (pp. 166-172). IEEE.

Hochstetler, J., Hochstetler, L., & Fu, S. (2016, December). An optimal police patrol planning strategy for smart city safety. In 2016 IEEE 18th International Conference on High Performance Computing & Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science & Systems (HPCC/SmartCity/DSS) (pp. 1256-1263). IEEE.

Kang, H. W., & Kang, H. B. (2017). Prediction of crime occurrence from multi-modal data using deep learning. PloS one, 12(4), e0176244.

Kennedy, L. W., Caplan, J. M., & Piza, E. (2011). Risk clusters, hotspots, & spatial intelligence: risk terrain modeling as an algorithm for police resource allocation strategies. Journal of quantitative criminology, 27(3), 339-362

Liao, K., & Guo, D. (2008). A Clustering-Based Approach to the Capacitated Facility Location Problem 1. Transactions in GIS, 12(3), 323-339.

Liberatore, F., Camacho-Collados, M., & Vitoriano, B. (2020). Police districting problem: Literature review & annotated bibliography. Optimal Districting & Territory Design, 9-29.

Mitchell, P. S. (1972). Optimal selection of police patrol beats. J. Crim. L. Criminology & Police Sci., 63, 577.

Nath, S. V. (2006, December). Crime pattern detection using data mining. In 2006 IEEE/WIC/ACM International Conference on Web Intelligence & Intelligent Agent Technology Workshops (pp. 41-44). IEEE.

Population & Annexation. (2003, March 17). Retrieved March 09, 2021, from https://chicagology.com/population/

Wang, D., Ding, W., Lo, H., Stepinski, T., Salazar, J., & Morabito, M. (2013). Crime hotspot mapping using the crime related factors—a spatial data mining approach. Applied intelligence, 39(4), 772-781.

Zhang, X., Hu, Z., Li, R., & Zheng, Z. (2010, June). Detecting & mapping crime hot spots based on improved attribute oriented induce clustering. In 2010 18th International Conference on Geoinformatics (pp. 1-5). IEEE.

Zhang, X., Liu, L., Xiao, L., & Ji, J. (2020). Comparison of Machine Learning Algorithms for Predicting Crime Hotspots. IEEE Access, 8, 181302-181310.

Zhu, S., & Xie, Y. (2018, April). Crime incidents embedding using restricted boltzmann machines. In 2018 IEEE International Conference on Acoustics, Speech & Signal Processing (ICASSP) (pp. 2376-2380). IEEE.