Computational Efficiency in Inferring Social Ties from Spatiotemporal Data

Presented to the S. Daniel Abraham Honors Program

in Partial Fulfillment of the

Requirements for Completion of the Program

Stern College for Women

Yeshiva University

January 4, 2021

Elisheva Muskat

Mentor: Professor Alan Broder, Computer Science

Abstract:

The ability to infer social ties is useful for gaining insight into matters of social influence and the propagation of information. It is also useful in a variety of other fields, such as epidemiology, marketing, and criminology. With the increasing accessibility of GPS-enabled mobile devices and the growing popularity of location-based services, a new field of research regarding inferring real-world social ties from spatiotemporal data has emerged [1-4]. Proper attention to accuracy and efficiency in inferring social ties is essential for the technique to be useful in real-world scenarios. However, computational efficiency when computing co-occurrences based on massive amounts of spatiotemporal data, has been an often overlooked or understudied factor. This paper's contribution is to survey prominent methods presented over the last two decades while analyzing how the method takes into account computational efficiency and, consequently, its usability with massive datasets. This survey then points the way towards fruitful areas for possible future research.

Introduction:

Why inferring social connection is important

Inferring social connections between individuals and identifying social networks has been important to social scientists for the past fifty years. It has become even more relevant due to its increasing feasibility in the last two decades with advances in data collection, storage capacity, and computing power. The ability to infer social ties is essential in a wide range of fields. Specifically, inferring the social ties of individuals helps researchers infer larger social networks. Gaining access to these networks offers insight into things like the propagation of information, the propagation of epidemics, and the influence of social networks on the behavior of

individuals. In addition to the advantages of identifying large social networks, even just understanding an individual's social ties can be used for targeted marketing of an individual, thorough tracing of epidemics, and advancing criminal intelligence analysis. The ability to infer social ties rose to prominence fifty years ago and has since found widely ranging applications.

The development that made spatiotemporal data available

The study of social networks became a field of interest in the 1970s when social scientists began to realize its wide range of uses [1]. Before the internet, scientists used surveys to gain extremely small, often biased datasets to determine social ties and create social networks [2,3]. In 1989, the development of the web led to internet use being pervasive. Soon after, the early 2000s brought the fast-paced growth of social media [4]. These social media platforms produced huge data sets containing explicit social graphs of millions of users which describe user's online friendships [2]. This development helped social scientists gain access to self-reported information about millions of users' online relationships. However, scientists have continued to look for ways to use the newfound datasets to identify friendships, beyond those self-reported, on social media sites. This allows researchers to expand their observation of social networks and to improve the accuracy of their models so that the observed social networks better reflect real-world social networks.

When researchers anticipated the upcoming social media platforms that were about to become available, they began to develop new methods for inferring social ties using the influx of data. Specifically, social media would bring in spatiotemporal data, or a collection of data that records the people's location over time. Preempting the rise of social media, the pioneers of using spatiotemporal data to infer social ties gathered their data by monitoring a group of people via GPS-enabled devices, or a tracking device [3,5,6] As they anticipated, in the last decade,

massive amounts of data representing users' spatiotemporal information became available as GPS-enabled mobile devices and location-based social network applications became more pervasive. Location-based services such as Twitter, Foursquare, and Gowalla each have millions of users and produce huge amounts of data every day, which can be useful for inferring social relationships. The newfound availability of huge amounts of spatiotemporal information led to the further advancement of inferring social connections using solely spatiotemporal methods.

How we can use that spatial-temporal data to infer social connections

The prominent way that spatiotemporal data is used to infer social ties is by comparing people's check-in points and finding co-occurrences. Check-in data is a type of spatiotemporal data consisting of a person's location and time at the moment of check-in. When a person surfaces on a location-based social networking service, like Facebook, the site records the individual's location and time of surfacing. A co-occurrence is recorded when two check-in points have adjacent locations within a similar time-frame. Co-occurrences aim to identify when two individuals in the real world were in the same place at the same time. Intuitively, if two people were in the same place at the same time, there is a likelihood that they have a social tie. Scientists further analyze various aspects of the co-occurrences such as the number of co-occurrences, the duration of co-occurrences, and the diversity of locations of the co-occurrences. These characteristics indicate further the likelihood that the co-occurrence is a social tie instead of a coincidence. For example, the more times people are found to be in the same place at the same time the more likely it is that they have a social tie. Scientists first extract co-occurrences between individuals from a series of check-ins and subsequently, analyze the co-occurrences found and determine which individuals have social ties or the probability that a pair of individuals has a social tie.

Benefits of Inferring Social Ties Using Spatiotemporal Data

Inferring social connection in this way can give a more complete picture of an individual's offline social network in addition to their online social network and removes biases that are present when inferring ties from surveys or an individual's social media portfolio. It can be even more helpful for the applications of inferring social ties that are specifically related to physically inferred social networks such as identifying members of criminal gangs or tracing diseases. Lastly, because spatiotemporal information is widely available, it is increasingly easy to infer social networks when using methods that rely solely on spatiotemporal data.

Focus of Prior Literature

The bulk of research thus far has focused on developing a method that uses spatiotemporal data to infer social ties with a high level of accuracy. High levels of accuracy require an understanding of which features of a co-occurrences best indicate a social tie and which features indicate a coincidental co-occurrence. As a result, researchers have investigated which attributes of people's geographic information is most revealing of their true social ties.The focus has been on the ability to infer the most information from the geographical information with the highest degree of accuracy.

What has not been researched

While considerable focus has been given to the accuracy and type of information inferred from a particular method, the computational efficiency of a method is an area that has been either understudied or entirely overlooked. For a method of inferring social ties to be useful for social scientists, criminologists, urban planners, and epidemiologists, the method must be both accurate

and efficient. The datasets being used are composed of millions of data points and help analyze the relationships of hundreds of thousands of users. Therefore, the algorithm used to infer social ties must be computationally efficient. Given the essential nature of computational efficiency when dealing with massive datasets, it's oversight in prior literature stands as a significant gap.

Unique complexities of computing co-occurrences

The problem of computing co-occurrences based on spatiotemporal data presents unique complexities because of the multidimensional nature of the data. The goal is to use the spatiotemporal information of individuals to find co-occurrences, or instances where the two individuals were in close geographical proximity at approximately the same time. In order to compute this, there must be comparisons done between individuals' spatial and temporal information, which creates a multidimensional dataset. The simplest way to solve this problem would be to compare every single spatiotemporal data point with every other spatiotemporal data point in order to find which data points are considered to have co-occurred. However, the method of comparing every single pair of data points is highly inefficient when dealing with millions of data points and even more so inefficient when the data points are multidimensional. There are common algorithms used to efficiently compare 1-dimensional data alone, like temporal data. Additionally, quadtrees are popularly used to solve specifically spatial problems in which it's necessary to compare locations. However, the difficulty of the co-occurrence problem is the need to take into account both the spatial and temporal aspects of the data when doing comparisons.

What will be addressed in this paper

This paper will survey prominent methods of inferring social ties that use spatiotemporal data, with an emphasis on the efficiency of each method and its applicability to real-world sized data sets. Given the massive data sets that are available, the problem of efficiency must be addressed to develop a feasible method of inferring social ties. However, there are only few studies that discuss the inference algorithms they used, and even fewer studies that discuss directly the efficiency of their inference algorithms or give any clear indication that they were concerned about efficiency at all. This paper will consolidate and organize the relevant information and available methods from prior research; it will be the first literature review of its kind to focus on the computing efficiency of inferring social ties from spatiotemporal data. The first section of the current paper identifies four general methods that researchers use to extract co-occurrences from spatiotemporal data and analyzes each method's computational efficiency. The four inference methods include the brute force method, partitioning the surface of the earth into grid-like cells, discrete check-in locations, and k-d trees. The subsequent section will discuss the way the MapReduce framework is utilized to optimize extracting co-occurrences. The third section will survey prominent articles and analyze the way that these authors utilized the aforementioned methods for extracting co-occurrences. Finally, this paper will present in condensed form the comparative characteristics of the top inference methods, and conclude with recommendations for future work in the field.

1. Extraction Co-occurrences

1.1 Brute Force Method

A brute force method is a problem-solving technique that tries out every possible solution until a valid solution is found. For example, a brute force algorithm to find a specific item in a list would start from the first item and iteratively check each item until the item being checked

matches the targeted item. Brute force algorithms are straightforward and simple approaches to solving a problem. However, these algorithms rely solely on computing power, to the exclusion of optimization, to solve the problem. Therefore, they can be time-consuming and inefficient when dealing with large datasets. The brute force method of finding co-occurrences compares each check-in point with every other check-in point and records the co-occurrences when found [7].

A significant flaw of using the brute force method to find co-occurrences is its computational inefficiency. Because the brute force method of finding co-occurrences compares every check-in point to every other check-in point, this method will include in its comparisons two check-ins that occurred on different sides of the globe. A system that necessarily compares check-in points that are across the globe for the purpose of finding co-occurrences is redundant and unintuitive. Comparing each check-in point with every other check-in point would be ((n-1)*n)/2 comparisons, n being the number of check-in points. Thus, as the number of check-ins increases the number of comparisons increases at an exponential rate. The time complexity of this algorithm can be described as $O(N^2)$, which means that the work performed to compute the co-occurrences is proportional to the square of the number of check-in points recorded. With datasets that are available today containing millions of check-in locations, performing these kinds of computations without further optimizations is inefficient and can take an exorbitant amount of time.

1.2 Surface of the Earth Split Up into Equal Cells

A second approach that eliminates many unnecessary comparisons is partitioning the surface of the earth into equally sized cells and only comparing check-ins that occur within the

same cell [8-11]. In addition to the brute force method, another method of identifying co-occurrences includes partitioning the surface of the earth into equally sized cells whose borders represent a specific location and map on to this representation all the recorded check-in points. Because each cell corresponds to a limited geographical area, all check-in points within one cell occur within relatively close proximity. These check-in points occurring inside a cell are only compared to each other [8-11]. This method cuts out many gratuitous comparisons and searches through a smaller more relevant dataset to identify co-occurrences. Because this method limits comparisons done to the check-ins recorded in each cell, smaller cell sizes, with fewer check-ins in each, result in fewer comparisons done overall. However, there is a loss of potential co-occurrences when check-ins are within the spatial threshold of co-occurrence but are in two separate cells on either side of a border as shown in Fig. 1. While the two check-ins may spatially co-occur, authors who use this method have not mentioned checking for such co-occurrences. Thus, while evenly partitioning the surface of the earth into cells increases efficiency and is more intuitive, it may miss co-occurrences that are made up of check-ins in bordering cells.

The method of partitioning the surface of the earth into grid-like cells and mapping on to the representation all recorded check-ins offers two possible strategies for identifying co-occurrences within cells, the first of which is to use a brute force method within each cell. This refers to comparing the location coordinates and time of every check-in point with every other check-in point within the cell. A co-occurrence is recorded when the location coordinates of two check-in points are within a certain spatial threshold and the times are within a certain temporal threshold. This method results in ((n-1)*n)/2 comparisons within each cell, n being the number of check-ins within a cell. In total, finding the co-occurrences from the entire dataset

would require k*(((n-1)*n)/2) comparisons, n being the number of check-ins within a cell and k being a constant that represents the number of cells that the surface of the earth was split into. The amount of comparisons is directly related to the number of check-in points within each cell and the number of comparisons grows exponentially only in relation to the number of check-ins in one cell. The time complexity of the approach where the surface of the earth is partitioned into cells is $O(n^2)$, where n is the amount of check-in points in each cell. This means as the number of check-in points in each cell grows the computational work grows exponentially. This is a big improvement compared to using a brute force method on the entire dataset, where every check-in point in the dataset is compared with every other in the entire dataset. Employing the brute force method to each cell, instead of overall, results in significantly fewer comparisons and thus greater computational efficiency.

The second strategy for identifying co-occurrences within cells assumes all check-ins within one cell are considered to be in close enough proximity to co-occur with each other [8-10]. The size of the cell corresponds to the maximum distance between two check-in points such that all check-in points within one cell are considered spatially co-occurred. In this case, a co-occurrence is recorded when two check-in points occur in the same cell and have similar timeframes. Therefore, the only comparisons that must be done to extract the overall co-occurrences are between the temporal aspect of the check-in points within each cell. Thus, the data that is left to be compared is just timestamps, which are one-dimensional data. This allows for optimized solutions for comparing the check-in points within a cell. One way researchers optimize this problem is to sort check-ins by timestamp and for each check-in point scan forward according to the size of the time-window[1]. Scanning forward includes searching through the subsequent check-in timestamps until the temporal distance between current check-in and the

[1] This information came from a personal email between myself and Dr. Crandall.

following check-in is no longer within the temporal threshold of a co-occurrence. Every pair of check-in points recorded within the time-window are considered to co-occur with each other. Sorting the check-in points by timestamp, using common sorting algorithms like Merge Sort or Quicksort, can be done with a time complexity of O(n*log(n)), n being the number of check-in points in a cell. Scanning the temporal aspect of the check-ins can on average be done with a time complexity of O(n), n being the number of check-in points in a cell. This is specifically true in a case where the check-ins are spread out over time. In a worst-case scenario where all the check-in points were within the time-window of a temporal co-occurrence, it would be necessary for every check-in point to scan through every other point to find the end of the time-window. This would require scanning through ((n-1)*n)/2 check-ins and therefore the time-complexity would be $O(n^2)$, n being the number of check-ins in a cell. However, in a real-world situation, the check-ins would most likely be spread out over time. In the average case, where time-complexity for sorting by timestamp would be O(n*log(n)) and the time-complexity scanning through the timestamps for temporal co-occurrences would be O(n), the time-complexity of the whole problem would be O(n*log(n)). Partitioning the surface of the earth and defining the boundaries of each cell as a spatial co-occurrence allows researchers to find co-occurring check-ins with a time-complexity equivalent to the time to sort the timestamps in a cell, which is efficient because it focuses solely on temporal data to identify spatiotemporal co-occurrences.

Comparing the two strategies that partition the surface of the earth into cells, the second strategy, where the boundaries of each cell define a spatial co-occurrence, is more efficient, while comparing the spatial and temporal aspects of every check-in with every other check-in within a cell offers more flexibility for the spatial threshold and the size of the cells. The time-complexity of inferring co-occurrences where the boundaries of each cell define a spatial

co-occurrence would be O(n*log(n)), n being the amount of check-ins in each cell. This strategy is more efficient compared to the time-complexity of inferring the co-occurrences within a cell where brute force method is used to compare every check-in point to every other check-in point within a cell. While using brute force method inside each cells is less efficient than assuming each cell is a spatial co-occurrence, it has a time-complexity of $O(n^2)$, n being the number of check-ins in one cell, which is much more computationally efficient than an overall brute force method's time-complexity of of $O(N^2)$, N being the amount check-ins overall. Additionally, using a brute force method within a cell offers flexibility to researchers who aim to have the threshold for a spatial co-occurrence and the cells of the partitioned the surface of the earth distinct sizes. For example, a researcher might want to count co-occurrences that happen within a close temporal and spatial proximity to each other only once. If two people were seen together at 10pm and again at 11pm on the same day ten blocks away from their initial co-occurrence, a study might want to count these two co-occurrence as only one. Therefore, it could be beneficial to have a small temporal threshold to minimize the possibility of coincidence but to have a larger cell that would define the boundaries of when multiple co-occurrences between the same pair would count only once. Assuming the boundaries of a cell define a spatial co-occurrence allows for a strategy that is on average more computationally efficient for identifying co-occurrences. However, using brute force within each cell is still significantly more efficient than using brute force on the entire dataset and offers flexibility that the other strategy does not. Thus, using the brute force method in each cell is more efficient than an overall brute method, but less efficient than assuming the boundaries of a cell define a spatial co-occurrence.

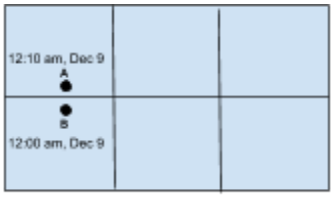

Fig 1. Illustration of how two check-ins A and B can occur within the spatial and temporal threshold of eachother but recorded in different cells and thus not counted as a co-occurrence.

1.3 Discrete Check-in Locations

One approach to the problem of extracting co-occurances from spatiotemporal data is to discretize the spatial component so that if two people check-in to a discrete check-in location within a certain time they are considered to co-occur [12-15]. Examples of using discrete check-in locations to model peoples' movements would be using data from cell phone towers [13], WiFi APs [12], or frequented locations that ask people to swipe ID cards [14] . All check-ins recorded at one check-in location are considered to spatially co-occur [12-15]. This approach has the same benefit as partitioning the surface of the earth: limiting the comparisons to only pairs of check-in points recorded in close proximity. Instead of partitioning the surface of the earth into a grid of cells, discretizing the spatial component would include a certain number of check-in locations that record the time that a person was at or near that location. This method then becomes very similar to the method of equally partitioning the surface of the earth into a grid and assuming the boundaries define a spatial co-occurrence. By employing discrete check-in locations, only the temporal aspect of the check-ins recorded at each check-in location need to be compared to identify co-occurrences.

The primary difference between the strategy that partitions the surface of the earth into cells where all check-ins in one cell spatially co-occur and the strategy that discretizes space into a number of check-in locations is the type of data used for both. The first method, partitioning

the surface of the earth into cells, uses social media data to identify co-occurrences. Using social media can be beneficial because of its widespread use, its mobile nature, and its accessibility. Additionally, social media data has ground truth associated with it, meaning that social media sites often have their own self-identified list of friendships. Therefore, a researcher looking to accurately identify social ties can confirm the accuracy of the observed social ties by comparing them to the social media's self reported database of friends. However, if the database of choice is cell-phone calls or a connection to specific WiFi, the intuitive strategy for collecting this spatiotemporal data would be to use a discrete number of check-in locations where all check-in points are considered to spatially co-occur. This strategy works best because cell-phone calls or connection to WiFi is automatically recorded at the nearest cell-phone tower or WiFi AP, respectively. While this data is harder for researchers to access, the benefit of using cell-phone calls or WiFi APs includes more dense and widespread data that social media check-ins. The amount of people who check social media varies and can be irregular while the use of cell-phone calls and the use of WiFi is more essential and consistent. Similarly, when identifying social ties of a specific population, setting up check-in locations in popular social spaces can offer a more populated dataset. For example, researchers studying social ties on college campuses collected spatiotemporal data by recording requiring students to scan their ID to complete a transaction at food courts and supermarkets. Since almost all students use these services multiple times each day the data acquired from this kind of method is very dense. However, this strategy of setting up check-in locations in popular areas would be less effective and less feasible if the goal was to identify co-occurrences from the entire population. While there are many computational similarities between the strategy that partitions the surface of the earth into cells with each cell defined as spatial co-occurrence and the strategy that discretizes check-in locations, the primary

difference between them is the source of the spatiotemporal data being used to determined co-occurrences.

Discretizing check-in locations is similar to partitioning the surface of the Earth into cells where each cell is a spatial co-occurrence, in that the computational efficiency of identifying co-occurrences is limited to the identification of temporal co-occurrences. Thus, in the case of discrete check-in locations, two check-ins are considered a co-occurrence when they are recorded at one check-in location within the temporal threshold to be considered a co-occurrence. There are three general approaches for identifying temporal co-occurrences: the fixed time slicing approach [14,15], the sliding window approach [14,15], and the clustering approach [12,15].

The first approach is the fixed time slicing method, where the check-ins ordered by timestamp are sliced into consecutive and non overlapping time slots [14,15]. The time slots are of equal size and correspond to the temporal threshold necessary for a co-occurrence [14,15]. All check-ins at one location within a time slot co-occur with each other. The way to compute co-occurrences with a fixed time slicing method is to order the check-ins by timestamp and scan through the ordered time stamps, slicing after the amount of time of the temporal threshold has passed. For example, if the temporal threshold is 10 minutes, then this method would slice the check-ins ordered by timestamp at every ten minute interval. The time-complexity for the slicing of this method is always linear because it is only necessary to pass through the data once, cutting at the end of each time interval and the sorting can be done with a logarithmic time-complexity. Therefore the method would have a guaranteed logarithmic time-complexity. The downside with a fixed time slicing method is that if two people were within a temporal threshold they could still be in two different time slots [14,15]. This leads to a loss of co-occurrences. Thus, the fixed time

slicing method can sort the check-ins into time intervals with a linear time-complexity because the data is only passed through once, but it overlooks co-occurrences that border each other.

A second approach is the sliding time window method, which corrects for the fixed time slicing method and captures all co-occurrences by looking at each check-in point's time window separately, such that no temporal co-occurrences get overlooked [14, 15]. There is a fixed time interval for the sliding time window method, which is similar to the fixed time slicing method. However, the ordered timestamps are not sliced at the end of each fixed time interval like in the previous method. Instead, for each check-in *c* which occurs at time *t,* all the other check-ins within the temporal threshold, Δt, of check-in c, are considered a co-occurrence [14, 15]. There are two ways to find for each check-in *c* all the check ins that occur between *t* and $t+\Delta t$. The first is to use a sort of brute method approach and compare every check-in's timestamp to every other check-in's timestamp at a given check-in location [14]. This approach requires ((n-1)*n)/2 passes through the data, and has a time complexity of $O(n^2)$, n being the number of check-ins at the given location. The second approach for finding for each check-in *c* with time *t,* all the points which occur between *t* and $t+\Delta t$, is to first order the check-ins by timestamp or assume the timestamps are stored in order [15]. Then, for each check-in, the ordered timestamps are scanned forward according to the size of the time-window[2]. This way, the check-in points are not separated into several discrete time slots, but rather, every check-in point has its own time-window that encompasses all the following check-in points that occur within the temporal threshold. The time-complexity for this approach is linear in the usual case where the times of the check-ins are spread out. However, in a case where all the check-ins occur within the time of the temporal threshold, or close to that spread, the time-complexity would be quadratic because

[2] Neither article [14] nor [15] articulated the approach to sort first. However, it was implied in article [15] that the check-ins were already sorted, and in an email from Dr. Crandall this method of identifying co-occurrences with check-ins ordered by timestamp was fully elaborated upon.

for every check-in, all the following check-ins will be scanned. The number of check-ins scanned would be ((n-1)*n)/2, n being the number of check-in points at each location, which would translate to a time-complexity of $O(n^2)$. Whether the data is ordered by timestamp initially or not, because each check-in point is examined separately for temporal co-occurrences, co-occurrences between pairs of check-ins recorded at one location within the temporal threshold would not be lost. The sliding time window approach solves the problem of the fixed time interval method by accounting for co-occurrences that would appear on the border of each slice.

A third approach that researchers use to identify co-occurrences is to identify social events and assume that all check-ins that appear together at that event co-occurred [12,15]. Social events are identified using clustering algorithms, which determine increased activity during blocks of time at a given location. In an article studying social ties of birds, the Gaussian mixture model, which groups points of a single distribution together, was used to detect windows of increased activity [15]. In another article which studied social ties of college students based on students' connection to WiFi APs, they used a clustering algorithm called CLL to identify time-windows where a high number of devices were connecting to one WiFi AP for a long enough duration of time [12]. Social events can be good indicators of meaningful interaction [15]. The logic is that two people who attended a social event together are more likely to have a social tie than two people who co-occurred in an empty hallway. Another advantage to this method is that researchers don't have to guess arbitrarily what an appropriate time-window would be to consider two individuals to co-occur [15]. Excessively limiting time-windows tend to omit important co-occurrences, while larger time-windows lead to an overestimation of social ties. Using social events to define the temporal parameters of a co-occurrence reduces the arbitrary nature of identifying meaningful co-occurrences [15]. Thus, the clustering algorithm

would consider the cluster to be an event and all the check-ins at that event are recorded as co-occurring [14,15]. Clustering algorithms can run with a time-complexity of O(N). This means that the time-complexity of identifying social events and defining co-occurrences based on appearances at those identified events O(n), n being the number of check-ins at each location. Using social events, which are identified by clustering algorithms, to infer co-occurrences is both computationally efficient and indicative of more meaningful social ties.

1.4 K-D Trees

K-D trees are multidimensional binary search trees, meaning they are an efficient data structure to store k-dimensional points in space. A binary search tree is a node-based tree data structure that stores 1-dimensional keys such that the keys in the left subtree of each node are less than the node above it and the keys in the right subtree are greater than the node. This structure allows for searches to be done with logarithmic time complexity, or in O(log(n)) time. This means that the algorithm runs proportionally to the logarithm of the size of the binary tree. K-d trees were developed by Bentley in 1975 to create a binary search tree structure that could efficiently store points of multiple dimensions like latitude and longitude points [16]. K-d trees integrate dimensions into the different layers of the tree so that each level is associated with a particular dimension and the children of a particular node are organized to the left or the right subtrees based on if the value of the relevant dimension is greater or less than the parent node. The tree can be depicted as a graphic, reflecting the number of dimensions associated with the tree. As points are inserted in the graphic, partitions are added along varying axes such that denser areas are partitioned into smaller spaces. This allows researchers to minimize the number of points they search through to find a point in a particularly dense area, which contrasts if the

entire space was split up evenly. As data points are added in one area in the k-d tree, the area is subdivided more finely and smaller, more targeted sub-divisions can be searched. Searching for a point or points near a certain value in a k-d tree can be done on average with a logarithmic time complexity, or in O(log(n)) time, just as with a binary tree. K-d trees are data structures based on the binary search tree structure that can store multi-dimensional data and search for points in logarithmic time.

Because of a check-in points three dimensional nature, k-d trees are a suitable data structure to optimize the spatiotemporal co-occurrence identification [17-18]. Check-in points are three dimensional in that they contain latitude and longitude coordinates as well as the time the check-in occurred. Pham et al. was the first to use the k-d tree data structure to store spatiotemporal check-in points to optimize the co-occurrence identification [17]. Since then, only one other study used this data structure to identify co-occurrences [18]. The k-d tree used to store check-in data for the purpose of identifying co-occurrences is a three-dimensional tree with a two dimensional (x,y) plane representing the location that a check-in took place, specifically latitude and longitude, and a t-axis representing the time that the check-in was recorded. The way a k-d tree optimizes implementations, as described earlier, is by partitioning areas of the tree as points are added so that even dense areas of the tree can be quickly traversed. When data points are inserted, the tree or sub tree is partitioned further into smaller sub-trees. As explained earlier, the trees can be partitioned in accordance with different possible dimensions. In this case, either the two dimensional (x,y) plane or the t-axis can be split. The goal of splitting the trees into sub-trees is to limit the amount of points that would have to be searched within a sub-tree. Therefore, it is most beneficial to partition the tree or sub-tree in such a way that the data points in the tree or subtree are being partitioned into a smaller sub-tree [17]. For example in a case where all the

check-ins in a sub-tree are recorded around the same time but are spatially spread out, partitioning that subtree on the time-axis could result in all the points remaining together in one of the smaller areas instead of being split up. Whereas, if in this example, the two dimensional location plane is split up, the points will be spread out into different sub-trees, which will result in a more efficient search as seen in Fig. 2 [17]. One strategy for identifying the best direction to partition the subtree is to calculate the potential Shanon entropy of subspaces with a partition along the (x,y) plane and compare that to the potential Shanon entropy of subspaces with a partition along the t-axis. The outcome that lends itself to greater entropy is favored because a higher entropy in the ensuing sub-spaces indicates that the points are more spread out, such that partitioning the space on that dimension would likely split up the space more efficiently. As was explained in the previous paragraph, a search through an efficient K-D tree has a time complexity of O(log(N)), N being the number of check-in in the dataset. Because the check-ins can be stored in an efficient k-d tree, for any given check-in searching for a co-occurrence has a time complexity of O(N*log(N)), N being the amount of check-ins in the dataset [18]. K-d trees are an ideal method of storing spatiotemporal data for the purpose of co-occurrence identification, because they are multi-dimensional and allow for efficient traversing of check-in data.

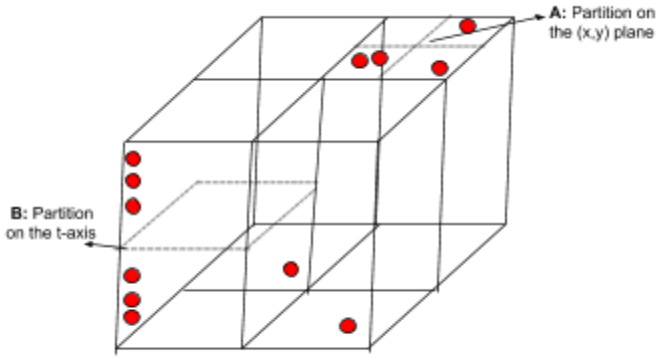

Figure 2: Illustration of a 3-D tree storing check-in data and the two possible partitions: A. On the (x,y) plane and B. on the t-axis.

2. MapReduce and Parallelization

MapReduce is a programming model and supporting framework that optimizes the processing of big data by partitioning the computing work into multiple processes that work in parallel to one another. The MapReduce model is made up of two functions: map and reduce. The map function processes the input data, generating a (key, value) pair for each relevant data point. For example, if the goal was to find the frequency of each name on a list, the list would be split up and passed to different mappers. The reduce function then aggregates pairs with the same key. In our example, if one mapper found a name twice and a parallel mapper found the same name once, the reduce function aggregates the lists. However, before starting the map function, the input data is partitioned into multiple blocks of data such that each block is then passed to one mapper, or a processor that applies the map function. Each mapper would then generate a count for each name in that particular section of the list. The (key, value) pairs, or each name and the number of times it appears, are then shuffled so that (key, value) pairs with the same keys that were in different processors for the mapping stage are passed to one reducer for the reduce stage. Finally, the reduce function is applied to each processor and the values of the (key, value) pairs with the same key are combined or aggregated. Running a process in a MapReduce format, which splits up the data and processes them independently before aggregating the results with multiple processors at the end, optimizes the process by allowing multiple processors to be running in parallel.

MapReduce has been used by researchers to optimize the process of identifying co-occurrences, speeding up the process by a factor k, k being the amount of processors used [7,17,18]. Overall, MapReduce works in a similar way for all methods of identifying co-occurrences. The map function identifies the co-occurrences from the chunk of the dataset that it is given and the reduce function aggregates for each pair of users all the co-occurrences that were found. Each mapper receives a chunk of check-in data and independently analyzes its own chunk of data to identify co-occurrences between individuals. All the co-occurrences recorded between one pair of individuals identified by multiple mappers gets passed to a single reducer and the reducer aggregates the list of co-occurrences for each pair [7,17,18]. The MapReduce framework can be used to optimize the identification co-occurrences by running multiple processors in parallel to one another.

However, for identifying co-occurrences, data partitioning in the MapReduce program can be tricky and is approached differently depending on how the data is stored. As explained previously, in order to apply MapReduce, the data needs to be partitioned such that each chunk of data can be processed independently. When it comes to identifying co-occurrences it is not simple to partition the check-in dataset such that the chunk of data can be processed independently. If the entire dataset was split up arbitrarily, there would potentially be two points which co-occur but were split into two different processors and were never compared. The way to split up the data to correct for these losses would differ depending on the method used to store data points and infer co-occurrences. If the researcher is partitioning the surface of the earth into equal cells where only check-ins within a cell are compared, it would make sense to partition the data by cells such that all the check-ins in one cell were processed together on one mapper. This helps to avoid a situation where co-occurring check-ins are passed to different mappers and

never checked for a co-occurrence. Similarly, in the case where check-ins were recorded by discrete check-in locations, check-ins are only compared to other check-ins within the same location. Thus, if the data is partitioned by check-in location and all check-ins within one check-in location are processed together there is no further loss of co-occurrences. As explained in section 1.4, sub-trees in a k-d tree contain points that are spatially and temporally close. Therefore, partitioning the data by sub-tree, so that all points in a sub-tree are processed together in a mapper maximizes the identification of co-occurrences [17, 18]. However, this is imperfect because partitioning k-d trees by sub-tree could create a situation where two points co-occur but appear in different sub-trees and are passed to different mappers. These co-occurrences would not be identified during the Map function, whereas they would not have been lost if the processes were not parallelized. If the check-ins are stored simply as a list of data and brute force method is used, there would be no possible partitioning of the data that would leave intact all the comparisons. Instead the way to make use of multiple processors would be to split up the data into chunks and first find co-occurrences in each chunk and then process every chunk with every other chunk in the co-occurrence identification stage [7]. To ensure the optimization of the MapReduce framework results in the fewest possible co-occurrences lost, splitting the data into chunks must be considered carefully and independently for each method of co-occurrence inference.

3. Practical Use of Methodologies as Seen in Prominent Articles

3.1 Inferring social ties from geographic coincidences 2010 [8]

David J. Crandall et al. wrote the first paper *Inferring Social Ties from Geographic Coincidences*, which used a massive spatiotemporal dataset to identify the probability that two

people have social ties based on co-occurrences in time and space [8]. The purpose of *Inferring Social Ties from Geographic Coincidences* is to understand what the likelihood is that people who were in close spatial and temporal proximity have a social tie, meaning to what extent a co-occurrence implies a social connection. Furthermore, Crandall et al. attempts to answer how that likelihood changes based on the spatial and temporal proximity of the co-occurrence [8].

In order to answer these questions, Crandall et al. computes co-occurrences with various spatial and temporal thresholds from a large spatiotemporal database and computes the probability that the pairs who co-occurred at various spatial and temporal proximities have a social tie [8]. The co-occurrences are checked against Flicker's public social network where Flickr users self report their social ties. The authors show that if two users are found to co-occur only a few times, the probability of social ties between them grows significantly. Furthermore, the studies found that smaller temporal and spatial thresholds indicate a much higher probability of social ties. This means that if the co-occurrences are limited to when the pair checked-in within a smaller window of time in closer physical proximity it is even more likely the pair shares a social tie. Additionally, using Bayes' Law, they expressed the probability that two users are friends given that they co-occur in varying locations on a certain number of consecutive days. By analyzing thousands of check-in points and identifying spatiotemporal co-occurrences, the authors were able to confirm that based on available spatiotemporal data of individuals it is possible to predict social ties with high probability [8].

In order to identify co-occurrences from the massive dataset of check-ins, Crandall et al. used the method of partitioning the surface of the earth into equal cells and assuming that the boundaries of each cell defined the spatial threshold for a co-occurrence [8]. Flicker, social media platform used to collect data, has 490,000 users and stores thirty-eight million geotagged

photos, which means that every time a photo is uploaded the time and latitude–longitude coordinates indicating the location on surface of the earth where the photo was taken is shared as well. In order to identify pairs of individuals who were in close proximity within a certain window of time, this paper partitions the surface of the earth into equal square cells. This way the time and coordinates of geotagged photos uploaded in different cities are not compared, which significantly improves computational efficiency. Crandall et al. demonstrate in practice how partitioning the surfaces of the earth into equal cells is a more efficient method than the brute force method of finding co-occurrences [8].

Because Crandall et al. defined the borders of each cell to be a spatial threshold for co-occurrence, they only had to compare temporal data to find co-occurrences. In this case, the temporal data was reflected in the time that each photo was uploaded [8]. The cells in this study, which defined a spatial co-occurrence, have side lengths of 80 km and longer. The size of the cells indicate that they each contain very large datasets of check-ins and comparing the times of every check-in with every other check-in in each cell would be computationally inefficient. The paper itself does not contain further elaboration on how they computed temporal co-occurrences. In a personal email, Crandall explained that they sorted the check-ins by timestamp and for each check-in, scanned forward until the time-stamp reached was no longer within the temporal threshold. For each check-in, all the following check-ins that were scanned until reaching the end of the time-window. This is the sliding time window method, described in sections 1.2 and 1.3, which eliminates loss of co-occurrences that would result from slicing the ordered time-stamps into fixed intervals. Crandell did not share the algorithm his research team used to sort the check-ins in each cell by timestamp. As explained in section 1.2, the time complexity for this method would be equivalent to the time complexity of this sorting algorithm. It is most likely

that Crandall used an algorithm like Mergesort or Quicksort which sorts with a time-complexity of O(n*log(n)), n being the number of check-ins within a cell. Crandall et al. orders check-ins within a cell by timestamp and uses a sliding time window to efficiently capture all the temporal co-occurrences.

Crandall et al.'s especially efficient method identifies co-occurrences by partitioning the surface of the earth into cells and using the sliding time window method; however, it does not account for lost co-occurrences resulting from check-ins that occur in different cells. The method used by Crandall et al. to partition the surface of the earth into cells and assume that all check-ins within one cell spatially co-occur is far more efficient and intuitive than the brute force method of comparing every check-in or every user with every other. Crandall's method of checking for co-occurrences within each cell by sorting the check-ins by timestamp before using the sliding-window method runs with a time-complexity on average of O(n*log(n)), the input value being just the amount of check-ins in a cell. Whereas, brute force method would have a time-complexity of $O(N^2)$, the input value being the entire dataset of check-ins. Crandall et al. 's method even has a good time complexity in relation to the use of a k-d tree which has a time-complexity of O(N*log(N)), N being the entire dataset. However, Crandall et al. partitions the surface of the earth into large cells so while the time-complexity is technically better, partitioning the cells equally in areas like New York City where there is a dense population will lead to a particularly long run time for identifying co-occurrences in those cells. In contrast, the benefit of a k-d tree is that in more dense areas the tree is subdivided into smaller trees so that the search is more efficient. Additionally, a benefit of Crandall's approach is the use of the sliding time-window dentifies co-occurrences that a fixed time-window approach would lose because the check-ins may occur within the temporal threshold but in different time intervals. However,

the loss of co-occurrences between check-ins that occur with-in the spatial threshold but in different cells is overlooked. While Crandall et al. optimized the co-occurrence identification process by partitioning the surface of the earth into cells while assuming all check-ins in one cell spatially co-occur and used a sliding time window to reduce co-occurrence loss, the use of large cell sizes can lead to costly searches for temporal co-occurrences and the loss of co-occurrences of bordering check-ins is overlooked.

3.2 EBM: an entropy-based model to infer social strength from spatiotemporal data 2013 [17]

Pham et al. in their paper titled, *EBM: an entropy-based model to infer social strength from spatiotemporal data* develops EBM as a model of inferring social ties based on spatiotemporal data that more accurately distinguishes social ties from coincidences and corrects for data sparseness, all while improving computational efficiency of co-occurrence identification [17]. The EBM model uses frequency and diversity co-occurrences to more accurately quantify the strength of a social connection thereby reducing the number of coincidences that are miscatagorization as social ties. The measurement of frequency refers to the number of times a pair co-occurred [17]. The more times two individuals co-occur, the higher the likelihood that there is a social tie between the two people. However, this measurement can lead to an overestimation of a connection between two individuals whose lives overlap coincidentally [17]. For example, if two individuals share a route to work or study in the same library, the frequency measure would indicate a high likelihood of a social tie. For this reason Pham et al. introduced the measurement of diversity into the EBM model [17]. The diversity measurement takes into account the diversity of the distribution of co-occurrence locations that a pair of individuals

share. The underlying theory being that if a pair of individuals co-occur multiple times in different locations, it is less likely that their co-occurrences are by chance. However, when the availability of spatiotemporal data is low, certain social ties can be underestimated or lost, which is a problem that is further exacerbated by taking the diversity measurement into account [17]. To correct for data sparseness the EBM model takes into account weighted frequency, which is the measurement of the popularity of the location of the co-occurrence [17]. If a location is not popular or generally uncrowded, a co-occurrence is more likely an indicator of a social tie between individuals. The authors tested this model by first identifying co-occurrences from a real-world dataset collected by a location-based social network called Gowalla. They then inferred social ties by analyzing the both types of frequency and the diversity of the co-occurrences. Finally, they compared the inferred friendships against the Gowalla's given social network [17]. EBM's model of inferring social ties takes into account frequency and diversity of co-occurrences to quantify the strength of a social connection with greater accuracy and reduce the number of coincides misinterpreted as social ties.

The EBM model used a K-D tree as the data structure to store the check-ins and parallelizes the process of identifying the co-occurrences using the MapReduce framework [17]. Researchers used Gowalla, the location-based social network containing close to 200,000 users and almost 6.5 million check-ins, as the platform to gather check-in data. Pham et al. explicitly recognizes the challenge of efficiently inferring co-occurrences from extremely large spatiotemporal datasets. In response to the challenge, the authors use a K-D tree with three dimensions to store the 3-D spatiotemporal data points. As explained in section 1.4, as check-ins are added the K-D tree is partitioned either along the time-axis or the location coordinate plane. Specifically, the authors explain that trees or sub-trees are partitioned when the tree or sub-tree

reaches its storage capacity. At that point, Shanon Entropy is used to predict whether splitting along the time axis or the location plane will result in sub-trees with check-ins more evenly divided amongst them. If the check-in points are not evenly divided between the sub-trees, and instead remain bunched together in one sub-tree, the computational efficiency is reduced when traversing the tree in search for specific data points. The authors note that when the Shanon Entropy for both partitions are approximately equal the default partition is a partition along the time axis. They explain that splitting the time axis is more cost efficient, since it requires only one partition to split the time axis in half . This contrasts with the partitioning of the location plane, which requires a partition along the x-axis as well as the y-axis as seen in Fig. 2 [17]. The time-complexity to search for points that are within the spatial and temporal threshold of every point is O(N*log(N)), N being the number of check-ins in the dataset. The authors further optimize the co-occurrence identification by running the program with multiple processors using the MapReduce framework [17]. Because the check-in points are stored in K-D trees, the check-ins are partitioned by sub-tree and chunks of subtrees are distributed to different mappers. The mappers identify co-occurrences based on the complete sub-trees they were given. In the reduce step, all the co-occurrences belonging to one pair of individuals are aggregated [17]. This parallelization does not affect the time-complexity, O(N*logN), but does speed up the operation by a constant factor of k, k being the amount of processors used. Pham et al. recognizes the complexity of identifying co-occurrences from large datasets and optimizes the inference algorithm by storing the data using a K-D tree and using the MapReduce framework to parallelize the process.

While Pham et al.'s method to identify co-occurrences is computationally efficient and innovative in its use of k-d trees to store check-ins and search for co-occurrences and the

MapReduce framework to optimize the process, the issue of loss of co-occurrences due to the parallel programming is not discussed. Pham et al. were the first to use a k-d tree to store spatiotemporal data in order to more efficiently search for co-occurrences [17]. The k-d tree method has a time-complexity of O(N*log(N)), N being the number of check-ins in the entire dataset [18]. The time complexity of Crandall's method is O(n*log(n)), n being the number of check-ins within a cell, which seems relatively more time efficient than the k-d tree method. However, with a k-d tree, more densely populated areas are subdivided more finely than sparse areas, such that searching for co-occurrences in more crowded areas like New York City would not lead to a substantially longer run time the way it would if the space was subdivided evenly in Crandall et al. 's method. Pham et al. further optimizes the co-occurrence identification using a MapReduce framework to parallelize the process. Using this optimization does not change the time-complexity but does speed up run time by a factor of the number of processors running in parallel. The data is partitioned by sub-tree, so that all points in a sub-tree are processed together in one mapper. However, if two co-occurring points appear in different subtrees, they will be passed to different mappers and won't be identified as a co-occurrence in the map function. While Pham et al. do not mention this potential loss of co-occurrences its possible that it is partially accounted for in that the subtrees are partitioned specifically on the basis of the check-in points being spread out. The fact that Shanon entropy is used to determine the partitioning of the subtrees, such that the resulting subtrees have check-ins that are specifically spread out, might also imply that the check-ins in different subtrees less often occurred within close spatial and temporal proximity to each other in contrast to check-ins within one subtree[3]. Although, the method used by Pham et al. to identify co-occurrences is computationally efficient and their partitioning method potentially increases the probability that check-ins stored on one subtree

[3] Analysis suggested by Professor Broder.

co-occur more than check-ins on neighboring subtrees, the processing of a chunk of subtrees independently from other chunks of subtrees still can lead to co-occurrence loss.

3.3 Exploring check-in data to infer social ties in location-based social networks 2017 [7]

Njoo et al. developed a novel Social Connection Inference framework (SCI framework) to infer social ties from geographic co-occurrences, which extracts the co-occurrences that are analyzed using a brute force algorithm [7]. The SCI framework, developed in 2017, analyzes the stability and duration of co-occurrences and looks more deeply at what time of the week the co-occurrence takes place. This helps predict more accurately the users' social link based on the available mobility data [7].

Njoo et al. tested their method on two location-based social network datasets: Gowalla and Brightkite [7]. There were 158,498 users and over 10 million check-ins collected. While this dataset is huge, Njoo and Hsu acknowledge in their work titled *Exploring Check-in Data to Infer Social Ties in Location Based Social Networks* that they use a brute force method, checking every combination of every user pair [7]. They explain that they split up the dataset of all of the check-ins into check-in sequences that correspond to each user's check-in history [7]. Each check-in sequence is ordered by the timestamp aspect of the check-in. Every user check-in sequence is compared with every other user check-in sequence [7]. This approach, which compares every user sequence with every other user sequence is considered a brute force approach. As explained earlier, this kind of method will include in the comparisons users that recorded check-ins at opposite ends of the globe.

The comparisons between check-in sequences were optimized such that the amount of comparisons that were done at most equal the sum of the check-ins in each users' check-in

sequence [7]. This was accomplished by first ordering each user's check-ins by timestamp. When a pair of users' check-ins were compared, two iterators were used, one for each user. Both iterators initially select the first check-ins in each users' list of check-ins. The check-ins selected by each iterator would be compared. The second iterator increments by one comparing each check-in selected with the check-in selected by the first iterator until the timestamp of the check-in chosen by the second iterator is smaller than the timestamp of the check-in selected by the first iterator. At this point the first iterator is incremented. Using this algorithm to compare check-ins between users reduces the amount of check-ins compared to at most equal the sum of the number of check-ins in each user's check-in sequence [7]. While this optimization allows for a faster co-occurrence generation in a case where there are fewer users and more check-ins per user, still it is inefficient because every user's check-in sequence is compared with every other user's check-in sequence. Thus, while not every check-in point is necessarily compared to every other check-in point, there still are ((N-1)*N)/2 comparisons being done, with N being the number of users that have any check-in points. This means that the algorithm is still running proportional to the square of the amount of users, and that as the amount of users grows the run time of the algorithm grows exponentially [7]. For this type of data where there are 158,498 users and 10 million check-ins, it is advantageous that the exponential time complexity is proportional to the amount of users, which is a much smaller number than the amount of check-in points. However, as the data grows, and specifically the number of users grows, this brute force algorithm, which runs with a time complexity of $O(N^2)$, N being the amount of users, becomes increasingly less feasible. While the method is optimized such that the run time grows with the number of users as opposed to the number of check-ins, this method still leads to unnecessary comparisons by way of brute force method.

Njoo et al. also optimized the brute force process by implementing a MapReduce framework, which can speed up run time by a factor of *k, k* being the number of processors [7]. As explained in section 2 the mappers identify co-occurrences based on the data received and the reducers aggregate for each pair their full set of co-occurrences. As explained in section 2, using a brute force method to identify co-occurrences does not lend itself to partitioning the data into sections that can be mapped independently because a brute force method asks for every point to be compared to every other point across the entire dataset. Njoo et al. explains that the dataset is separated into chunks of data that are processed separately meaning that the co-occurrences in each chunk are identified separately [7]. However, mappers identify co-occurrences in each chunk independently, and the authors do not explain how they partition the data such that every user's check-ins are compared with every other user's check-ins. One must assume that the researchers maintained a brute force method by splitting the data into a certain number of chunks, and after each chunk is processed for co-occurrences, each chunk is compared with every other chunk for additional co-occurrences[4]. For example, if user 1 is in chunk *a* and user 5 is in chunk *b*, after all the users in chunk *a* are compared with each other, chunk *a* and chunk *b* would be processed together and user 1's check-ins and user 5's check-ins would be compared. Because all the users are still compared with every other user, the time-complexity remains $O(N^2)$, but there is a speed up by a factor of *k,* with k being the number of processors [7]. Despite the time-complexity remaining the same, Njoo et al. notes that the speed up is evident in the co-occurrence identification process [7]. The authors of this paper use MapReduce to further optimize the brute force co-occurrence identification by splitting up the data into chunks of users and then processing with multiple processors every chunk of users with every other chunk of

[4] This understanding of the co-occurrence identification method used in *Exploring Check-in Data to Infer Social Ties in Location Based Social Networks* was a contribution from Professor Broder

users so that the check-ins of every user from each chunk is checked against the check-ins of every other user.

While the brute force method used by Njoo et al. is computationally inefficient, the two optimizations implemented to speed up run time help to make this method more feasible. As previously noted, using brute force method carries the usual inefficiencies of unnecessary comparisons between unrelated check-ins. This method runs with a quadratic time complexity, in line with the rate of growth of the entire dataset. In a later article written by some of the same authors of this paper they use a k-d tree to solve the same problem of identifying co-occurrences, clearly demonstrating that this earlier method suffered from inefficiency [18]. However, Njoo et al. implemented two optimizations that sped up run time. The first was that instead of comparing every single check-in with every other check-in, the check-ins of a single user were grouped together and sorted within a user. Subsequently, only check-ins which align temporally were compared [7]. Because of this optimization, comparing two users on average runs linearly with the number of the check-ins. Therefore, the time-complexity of identifying co-occurrences from the entire dataset has a time-complexity of $O(N^2)$, with N being the number of users instead of the number of check-ins. The second optimization is the use of the MapReduce framework which speeds up the run time by a factor equal to the number of processors used [7]. While the method used here is computationally inefficient, the two optimizations improve the time complexity and increase speed up resulting in a more feasible method.

3.4 Finding College Student Social Networks by Mining the Records of Student ID Transactions 2019 [14]

In a paper titled Finding College Student Social Networks by Mining the Records of Student ID Transactions, Xu et al. presents a method that inferres social networks of college students from spatiotemporal data with increased accuracy by using a sliding-time window method and accounting for the homophily effect [14]. The homophily effect describes a phenomenon where people with similar interests tend to co-occur more often than those with dissimilar interests [14]. This paper was published in 2019, when a good amount of research had been done on the topic of inferring ties based on spatiotemporal data. Xu et al. specifically aimed to improve the accuracy of the inference regarding college students' social networks [14]. The social life of college students is particularly important both in the realm of their mental health and overall happiness. However, as Xu et al. points out, previous research studying social networks of college students have overlooked the homophily effect . Because of the homophily effect, students who share a major or an interest in a specific extra-curricular will present with particularly strong ties in comparison to close friends who have different interests [14]. Xu et al. first uses a sliding time-window method to infer co-occurrences from check-ins recorded at a single discrete check-in location. Subsequently, they account for the homophily bias by analyzing the inferred co-occurrences using a hierarchical encounter model based on association rules. Association rules apply different association analysis to intra-group students, or students with the social ties *and* the same interest, and inter-group students, students with social ties who do not have the same interests [14]. Based on co-occurrence analysis on both these levels the social network of each student is constructed with more accuracy than previous studies.

To infer the co-occurrences Xu et al. set up several discrete check-in locations in popular places on the college campus and used a sliding time-window to calculate which pairs of check-ins occur at a spatial co-occurrence within the temporal threshold [14]. Because this study

was collecting the spatiotemporal data of a specific population, namely college students, recording peoples check-ins at popular places can result in a more dense and all-inclusive dataset. There were 17 check-in locations located at various food courts and supermarkets, which record students' ID and their transaction information, including the location and time. In two years, there were 662 students and almost 400,000 check-ins recorded [14]. Every check-in is stored as an ID: a number between 1 and 17 which represents the check-in location and the time of the transaction [14]. All check-ins within one check-in location are analyzed to find which pairs co-occurred. Since the space has been discretized into 17 locations, all check-ins that occur at one location are considered to spatially co-occur. Therefore, only the temporal aspect of the check-ins at each location need to be considered to identify a co-occurrence. The authors of the paper explain that they identified check-ins that have occurred within the temporal threshold using a sliding time-window approach to eliminate loss of co-occurrences that results from using a fixed time-window as explained in section 1.3. As explained earlier, the sliding time-window approach would mean that for each check-in which occurs at time t, all check-in points that occur within the window $t+\Delta t$ are considered co-occurring [14]. In contrast to a fixed-time window approach, where the ordered time-stamps would be cut into consecutive, non-overlapping time-windows of Δt. From the algorithm described in the paper, it seems that the authors did not sort the check-ins by timestamp before applying the sliding time-window approach. Therefore, as explained in section 1.3, for each check-in c occurring at time t at location l, the timestamp of all check-ins at location l are checked. Those that occur between t and $t+\Delta t$ are recorded as co-occurrences with check-in c [14]. Because for every check-in the algorithm scans through all the other check-ins, the time-complexity of this approach is $O(n^2)$, n being the number of check-ins at each location. Because there are only 17 locations, the 400,000 check-ins are not

divided into such small groups and using an exponential time algorithm could be computationally inefficient. Sorting the check-ins with an O(n*log(n)) algorithm would have been more computationally efficient as explained in section 1.3.

While Xu et al. 's chosen method to identify co-occurrences can be used for a dense and full dataset and reduces the loss of co-occurrences with the use of a sliding time window, this approach overlooks certain efficiency considerations. Because the goal of the study is to understand the social ties of college age students, using discrete check-in locations in popular college hotspots to collect check-in data offers a more comprehensive and meaningful dataset. While this type of data collection is less feasible when collecting data from people across the world in different stages of life, the collection method for college students ensures that almost every student will check-in at least once a day which is not the case with social media check-in data. Additionally, because the spots chosen are popular areas, they are spaces of increased socialization, which could indicate that check-ins recorded in these places are more likely to signify a social tie. However, the discrete check-in location approach lacks flexibility in terms of the spatial breakdown of the group of check-ins. With the method that entails partitioning the surface of the earth into equal cells, the earth can be partitioned into very small cells when the dataset is large, which helps speed up the co-occurrence identification process. Furthermore, with a k-d tree, areas that have large populations of check-ins can be partitioned further to speed up the search for co-occurrences. In contrast, discrete check-in locations don't have that flexibility, and instead the set number of check-in locations doesn't allow for speeding up the search for co-occurrences. In terms of the co-occurrence identification process, the use of a sliding time-window is another strength of Xu et al's overall method because as explained previously it reduces the loss of co-occurrences [14]. However, the algorithm used to implement

the sliding time-window lacks the optimization of Crandall et al.'s algorithm, which sorts the check-ins by timestamp before using a sliding time window to find the temporal co-occurrences for each check-in. Instead of sorting the check-ins at each check-in location and then scanning forward for each check-in to find all the other check-ins within the temporal threshold, which can run linearly with the number of check-ins at each location, Xu et al. uses an algorithm which scans for each check-in *c*, every other check-in's timestamp to determine whether those other check-ins are within the temporal threshold of check-in *c* [14]. The latter application of the sliding time-window used by Xu et al. is far less computationally efficient with a time complexity of $O(n^2)$, n being the number of check-ins at each location. While Xu et al.'s method allows for the use of a comprehensive dataset of college students, captures instances that likely indicate social interaction, and prevents loss of co-occurrences, the computational efficiency of the method is close to a brute force method with the data split up into large inflexible chunks and the co-occurrence identification at each check-in location running a quadratic time complexity.

5. Chart with comparative characteristics of the top inference methods

Article	Nature of Data	# of Subjects	Total Observations	Method for Computing Co-occurrences	Computational Complexity
Reality mining: sensing complex social systems 2006 [3]	Mobile phone data from GPS enabled bluetooth	100 subjects	450,000 hours of information about users location	Not mentioned	Unknown
Inferring friendship network structure by using mobile phone data 2009 [5]	Mobile phone data from GPS enabled bluetooth	94 subjects	Approximately 4 million events	Not mentioned	Unknown
Bridging the gap between physical location and online	Location sharing social network,	489 Subjects	2 million observations	Partitioning the surface of the earth into cells. The method then used to calculate the	Between $O(n*\log(n))$ and $O(n^2)$, n being the number of check-in in a cell.

social networks 2010 [11]	called Locaccino			co-occurrences within a cell is unknown.	
Inferring social ties from geographic coincidences 2010 [8]	Geo-tagged photographs from Flicker	490,000 users	38 million geotagged photos	Partitioning the surface of the earth into cells whose borders define the threshold for a spatial co-occurrence. Sorting the check-ins by timestamp and applying a sliding time-window to identify the temporal co-occurrence within a cell.	The computational complexity of the sorting algorithm used. Most likely O(n*log(n)), n being the number of check-ins in one cell
Ebm: an entropy-based model to infer social strength from spatiotemporal data 2013 [17]	Location-based social network, called Gowalla	196,591 users	6,442,890 check-ins	Using a k-d tree to store check-ins and a MapReduce framework to parallelize the co-occurrence identification process.	O(N*log(N)), N being number of check-ins in the entire dataset because the time-complexity of co-occurrence search for each check-in would be O(log(N)). The process is sped up by the parallelization by a factor of the number of processors.
PGT: Measuring Mobility Relationship Using Personal, Global and Temporal Factors 2014 [20]	Gowalla and Brightkite, two location-based social networks	Gowalla: 107,092 users Brightkite: 58,228 users	Gowalla: 6,442,890 check-ins Brightkite: 4,491,143 check-ins	Not Mentioned	Unknown
Inferring friendship from check-in data of location-based social networks 2015 [9]	Location-based social network, called Gowalla	54,622 users	3,672,646 check-ins	Partitioning the surface of the earth into cells whose borders define the threshold for a spatial co-occurrence. Splitting up the check-ins into users and comparing timestamps between users.	The time-complexity is $O(n^2)$, n could be the number of users in a cell or the number of check-ins in a cell depending on the method used to compare the timestamps.
Inferring social structure from temporal data 2015 [15]	The population of birds at 35 different feeding stations	"Large proportion of wild birds"	More than half a million recordings of birds at feeders	Discretizing the space in 35 locations and identifying the co-occurrences using a fixed time window, a sliding time-window and the Gaussian mixture model to identify events. This paper seems to indicate that the check-ins were ordered by timestamp before the temporal co-occurrences were identified.	If the data is not sorted the time complexity will most likely be O(n*log(n)) to sort for all 3 models. If the data is sorted: FTW: O(n), STW: O(n) best $O(n^2)$ worst, GMM: O(n), n being the population of birds at each of the 35 locations

Geographical impacts on social networks from perspectives of space and place: an empirical study using mobile phone data 2016 [13]	Mobile phone records from 1529 mobile base towers around Northeast China	3,361,860 mobile phone users	145,693,888 voice call records with record the time and which tower picked up the call	Temporal co-occurrences are identified from the check-ins recorded at each phone tower.	Depending on which method of defining co-occurrences is used and whether the data is ordered by timestamp first the time complexity could be between O(n) and $O(n^2)$, n being the number of voice calls recorded at each of the 1529 phone towers
Event-Based Social Network Discovery (ESONED) Using WiFi Access Points 2016 [12]	Activity logs of over 16,000 WiFi access points	32,000 users	Over 5,400,000 data points	Events are identified from the datastream of check-ins recorded at a single WiFi AP using a clustering algorithm. All pairs of check-ins that appear at one check-in location are considered co-occurrences.	It depends on the computational complexity of the clustering algorithm but clustering algorithms can run linearly so the time-complexity most likely is O(n), n being the number of check-ins at a given location.
Exploring check-in data to infer social ties in location-based social networks 2017 [7]	Gowalla and Brightkite, two location-based social networks	158,498 users	Over 10 million check-ins	Brute force method was used to compare every user's check-ins with every other user's check-ins and the MapReduce framework was implemented so that the process could run on multiple processors.	Because the algorithm used, which compares the check-ins between two users, runs linearly, the time-complexity is $O(N^2)$, N being the number of users. The MapReduce framework speeds up the process by a factor equal to the number of processors.
Distinguishing friends from strangers in location-based social networks using co-location 2018 [18]	Gowalla, Brightkite, and foursquare, three location-based social networks	Gowalla: 99.2k Brightkite: 43.6k Foursquare: 11.3k	Gowalla: 6.43M Brightkite: 4.74M Foursquare: 2.29M	Using a k-d tree to store check-ins and a MapReduce framework to parallelize the co-occurrence identification process.	O(N*log(N)), N being number of check-ins in the entire dataset because the time-complexity of co-occurrence search for each check-in would be O(log(N)). The process is sped up by the parallelization by a factor of the number of processors.
Finding College Student Social Networks by Mining the Records of Student ID Transactions 2019 [14]	Location of undergraduate student transaction	662 students	2012: 203,247 2013:194,161	Comparisons are done only within the recorded transactions at each of the 17 check-in locations	$O(n^2)$, n being the number of student transactions at different check-in locations

4. Conclusion

The current paper outlines four general methods for identifying co-occurrences, focusing on the computational efficiency of each method and various optimizations that improve on the

computational efficiency of the methods. The first method outlined is the brute force method which compares every check-in or user with every other check-in or user [7]. This is the least efficient method. This method is represented in my analysis of Njoo et al.'s work to improve on the accuracy of inferring social ties from spatiotemporal data [7]. The second method entails partitioning the surface of the earth into equal cells and mapping on to this representation all the recorded check-in points [8-11]. In this method only check-ins within one cell are compared when identifying co-occurrences [8-11]. One implementation of partitioning the surface of the earth into cells is to use a brute force method within the cell to compare every check-in with every other check-in to find co-occurrences while another assumes the borders of the cell define a co-occurrence and only check for temporal co-occurrences within a cell [8-11]. Crandall et al. partitions the surface of the earth and assumes each cell is a spatial co-occurrence in order to better understand to what extent co-occurrences imply a social connection [8]. A third approach uses discrete check-in locations spread out over larger spaces to record check-in data [12-15]. This method has a similar computational efficiency to the method of partitioning the surface of the earth and assuming that the borders of the cells define a spatial co-occurrence. Both require only checking for temporal co-occurrences between check-ins corresponding to a single location. The paper here outlines three approaches to identifying temporal co-occurrences: the fixed time window [14,15], the sliding time window [14,15], and identifying social events using a clustering algorithm [12, 15]. Xu et al.'s work to infer social networks of college students with increased accuracy employs discrete check-in locations because it captures a more comprehensive and meaningful dataset [14]. Finally, the fourth method presented in the current paper stores the 3-dimensional check-in data in a k-d tree, which allows for computationally efficient traversing to find co-occurrences [17, 18]. Pham et al. uses k-d trees in their work to

discern social ties from coincidental co-occurrences [17]. In addition to the four methods, the current paper discusses MapReduce as a strategy for optimization, as seen in the work of Njoo et al. and Pham et al [7,17,18]. The current paper organizes, consolidates, and analyzes the primary methods and papers that aim to identify co-occurrences with greater efficiency. The ability to identify social ties based on spatiotemporal data has a wide range of applications namely targeted marketing, thorough tracing of epidemics, advancing criminal intelligence analysis and providing insight into propagation of information and progression of epidemics. These applications are useful and often urgent. They span matters of safety, economics, and health. Computational efficiency when inferring social ties from co-occurrences is essential for the feasibility of this process.

Future research should empirically test the co-occurrence methods, solve the problem of co-occurrences being lost across the border of a cell, and synthesize properties from the various methods presented. Building on the theoretical work done in the current paper, testing the different methods empirically on huge data sets will help determine how to best pair methods with optimizations to create the overall most efficient computation. Future empirical work can also offer practical understanding of how lost co-occurrences in the various methods affect accuracy. Another important direction for future research is to find solutions to the problem of co-occurrences lost in the various methods. This is especially pertinent in the case where the surface of the earth is partitioned into different cells and check-ins that co-occur are in different cells. Finally, work should be done to synthesize these different methods, borrowing properties from each to develop new methods that highlight each individual method's strengths. For example, developing a k-d tree,-partitioned cell hybrid method would limit searches within certain dense areas while maintaining the time complexity advantages of only searching within a

smaller cell. Since the articles raised in this paper did not themselves discuss computational efficiency and only briefly summarized their methods, this paper is limited to theoretical analysis, and reflects the vast need for deeper and more focused research on this topic.

Bibliography

[1] Rogers, Everett M., and F. Floyd Shoemaker. “Communication of Innovations; A Cross-Cultural Approach.” *Eric*, 1971.

[2] Chen, Wei, et al. “Information and Influence Propagation in Social Networks.” *Synthesis Lectures on Data Management*, vol. 5, no. 4, 2013, pp. 1–177., doi:10.2200/s00527ed1v01y201308dtm037.

[3] Eagle, Nathan, and Alex (Sandy) Pentland. “Reality Mining: Sensing Complex Social Systems.” *Personal and Ubiquitous Computing*, vol. 10, no. 4, 2005, pp. 255–268., doi:10.1007/s00779-005-0046-3.

[4] Sen, Arnab. “The Real and Virtual Worlds Are Melting Together.” Mashable, Mashable, 9 Jan. 2012, mashable.com/2012/01/09/real-world-digital-world/.

[5] Eagle, N., et al. “Inferring Friendship Network Structure by Using Mobile Phone Data.” *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, 2009, pp. 15274–15278., doi:10.1073/pnas.0900282106.

[6] Li, Quannan, et al. “Mining User Similarity Based on Location History.” Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '08, 2008, doi:10.1145/1463434.1463477.

[7] Njoo, Gunarto Sindoro, et al. “Exploring Check-in Data to Infer Social Ties in Location Based Social Networks.” *Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science*, 2017, pp. 460–471., doi:10.1007/978-3-319-57454-7_36.

[8] Crandall, D. J., et al. “Inferring Social Ties from Geographic Coincidences.” *Proceedings of the National Academy of Sciences*, vol. 107, no. 52, 2010, pp. 22436–22441., doi:10.1073/pnas.1006155107.

[9] Cheng, Ran, et al. “Inferring Friendship from Check-in Data of Location-Based Social Networks.” *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, doi:10.1145/2808797.2808884.

[10] Pham, Huy, et al. “Towards Integrating Real-World Spatiotemporal Data with Social Networks.” *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11*, 2011, doi:10.1145/2093973.2094046.

[11] Cranshaw, Justin, et al. “Bridging the Gap between Physical Location and Online Social Networks.” *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, 2010, doi:10.1145/1864349.1864380.

[12] Fang, Le, et al. *Event-Based Social Network Discovery (ESONED) Using WiFi Access Points*, 2016, www.semanticscholar.org/paper/Event-Based-Social-Network-Discovery-(ESONED)-Using-Fang-Guan/005cfab60b6f0dd2f3da94ce725a45775c3a09e6.

[13] Shi, Li, et al. "Geographical Impacts on Social Networks from Perspectives of Space and Place: an Empirical Study Using Mobile Phone Data." *Journal of Geographical Systems*, vol. 18, no. 4, 2016, pp. 359–376., doi:10.1007/s10109-016-0236-8.

[14] Xu, Jing-Ya, et al. "Finding College Student Social Networks by Mining the Records of Student ID Transactions." *Symmetry*, vol. 11, no. 3, 2019, p. 307., doi:10.3390/sym11030307.

[15] Psorakis, Ioannis, et al. "Inferring Social Structure from Temporal Data." *Behavioral Ecology and Sociobiology*, vol. 69, no. 5, 2015, pp. 857–866., doi:10.1007/s00265-015-1906-0.

[16] Bentley, Jon Louis. "Multidimensional Binary Search Trees Used for Associative Searching." *Communications of the ACM*, vol. 18, no. 9, 1975, pp. 509–517., doi:10.1145/361002.361007.

[17] Pham, Huy, et al. "Ebm." *Proceedings of the 2013 International Conference on Management of Data - SIGMOD '13*, 2013, doi:10.1145/2463676.2465301.

[18] Njoo, Gunarto Sindoro, et al. "Distinguishing Friends from Strangers in Location-Based Social Networks Using Co-Location." *Pervasive and Mobile Computing*, vol. 50, 2018, pp. 114–123., doi:10.1016/j.pmcj.2018.09.001.

[19] Wang, Hongjian, et al. "PGT: Measuring Mobility Relationship Using Personal, Global and Temporal Factors." *2014 IEEE International Conference on Data Mining*, 2014, doi:10.1109/icdm.2014.111.