Who to query? DispNN: A two-stage querying algorithm for identifying events with variant/unknown spatial distributions

Mai ElSherief
Dept. of Computer Science
UC Santa Barbara
mayelsherif@cs.ucsb.edu

Ramya Raghavendra
IBM T. J. Watson Research Center
rraghav@us.ibm.com

Elizabeth Belding
Dept. of Computer Science
UC Santa Barbara
ebelding@cs.ucsb.edu

ABSTRACT

The ubiquity of sensors, whether devices or humans, and the resulting explosion of critical applications have introduced a need to address a variety of new opportunities and challenges. In this paper, we propose DispNN: a two-stage node selection algorithm for resource constrained systems based on node locations to identify incidents in a $2D$ spatial environment. Given a constraint of N probes, DispNN uses a subset of N to probe nodes that maximize the dispersion of node locations. Based on the response received from the queried nodes, DispNN then probes the K nearest neighbors using the rest of the available N probes. We evaluate DispNN algorithm on three different distributions: clustered, uniform and long-tailed. We then apply the algorithm to a dataset of street harassment provided by Hollaback. The proposed algorithm outperforms a random selection approach by up to 63% and an approach of dispersion maximization by up to 68%.

1. INTRODUCTION

Sensors have become an integral part of daily life. The common smartphone includes a variety of different sensors, such as camera, microphone, GPS, accelerometer, digital compass, light sensor, and Bluetooth as a proximity sensor. The ubiquity of sensors is also prevalent in the urban environment. Examples include traffic sensors, agriculture sensors, wireless parking sensors, infrastructure sensors, weather, and pollution sensors. Data analysis from such sensors yield important observations. For instance, [9] leverages the geographic and temporal data associated with taxis in NYC to gain insight into multiple aspects of city life, from economic activity and human behavior to mobility patterns. When combined with crowdsourcing of human input, critical environmental data can be generated. One example is the application "Waze", where users can report traffic jams, accidents and other road related incidents in real-time. The work in [3] uses local workers to collect data at different events, that were mostly open to the public (e.g. a neighborhood fest, an art show, a music festival and others), and remote workers to curate the collected information and generate event reports.

One of the most pressing challenges of sensor ubiquity is energy preservation. All systems are bound by a fixed amount of resources; sensor nodes are typically the most constrained. For instance, [18] and [19] focus on eliminating redundancies among correlated sensor measurements. In this paper, we investigate the problem of how we can probe a limited subset of sensors in a particular environment in order to preserve energy or other resources. Examples of resource constrained systems include disaster and safety applications. In an emergency, communication networks tend to fail and available resources, such as bandwidth, are scarce [17].

In particular, we envision a scenario where users/sensors can be probed to collectively answer some question. An unanswered question can be related to a phenomenon that needs to be identified under the constraints of N resources. In [7], the unanswered question was related to if users felt a 5.8 magnitude earthquake which occurred on the East Coast of the United States (US) on August 23, 2011. The experiments in this work, analyzing the spatial and temporal characteristics of the twitter feed activity support the notion that people act as sensors to give us comparable results in a timely manner. In this scenario, our algorithm can be used to query human resources in the geographic area affected by the earthquake. The earthquake can affect network performance and impose a resource constraint on the number of people queried. Another example of safety applications is what happens in Tahrir Square during Egyptian revolutions in 2011 and 2012. At that time, women were discouraged from participation due to the high harassment rates [1]. This resulted in movements of men forming protective human shields [2] around female protestors to avoid assault. In this scenario the unanswered question would be "Is it safe in your location?" or "Is it harassment free in your location?". The resource constraints in this scenario would arise from avoiding disturbing the millions of people in the square and avoiding the base stations overload in this area. Our proposed algorithm can be used to query users for safe zones for women and then use the results for safe routing around the square or for identifying zones where women can safely participate in the protests.

In this paper we introduce DispNN (Dispersion-Nearest Neighbors): a generic two-stage querying/probing algorithm that

aims to maximize the *dispersion* of nodes locations in the first stage and then in the second stage, it selects the *nearest neighbors* for nodes that provide a positive feedback about incidents in their surroundings. Our contribution in this paper is three-fold. First, we propose DispNN: a two-stage matching algorithm that attempts to gain information about incidents when there is no prior information about the spatial distribution while conserving resources. Secondly, we study the performance of our proposed algorithm under three different distributions: clustered, uniform and long-tailed. The initial algorithm outperforms the random user selection by up to 63% in terms of selecting nodes that are closer (in the K nearest neighbors (KNN) sense) to the events and outperforms the dispersion maximization algorithm by up to 68%. We then test the algorithm on a real dataset of street harassment reports in different cities and show the applicability of using DispNN in detecting incidents and in locating users close to these incidents. Third, we discuss how the algorithm can be modified in the case of untrustworthy nodes based on trust variations or prior knowledge availability of the spatial distribution.

The rest of this paper is organized as follows. Section 2 surveys the related work while Section 3 describes DispNN. Section 4 experimentally evaluates DispNN, and Section 5 discusses tradeoffs and variations of DispNN. Section 6 concludes the paper.

2. RELATED WORK

Since the introduction of "crowdsourcing" as a modern business term in 2006 [12], a significant body of work has been dedicated to the study and implementation of crowdsourcing in real life applications. In particular, spatial crowdsourcing, where crowd participation is bound to a particular geographic area, has received significant attention [13, 8, 23]. For instance, [16] introduces a location-based real-time social question answering service, where users can ask temporal and geo-sensitive questions and then receive answers that are crowdsourced in a timely fashion. A crowdsensing platform was introduced in [6] to facilitate the collaboration of large groups of people participating in collective actions of urban crowdsourcing. Our work is different in the sense that it imposes a constraint on the number of nodes that can be queried in order to conserve resources. This is particularly important to avoid disturbing a large number of people in a crowded geographic area, such as a concert or protest. In addition, this prevents the querying server from entering a state of response overload.

Using people as sensors, collective sensing, and citizen science have opened doors for interesting research problems. Some of these challenges are introduced in [5]. One important challenge in geo-crowd sensing is detecting unusual events. The work proposed in [15] leverages microblogging websites such as Twitter to detect unusual geo-social events by identifying unusually crowded regions. Another challenge is the refinement of crowd sensed data and detection of fake data. Solutions based on a user's history and reputation have been introduced in the literature. The work in [24] proposes a reputation-aware model that balances the workload between users. Another challenge is fusing untrustworthy estimates [21]. Taking into account spatial properties, [22] tackles the problem of merging multiple spatial observations reported by possibly untrustworthy users using a heteroskedastic Gaussian process model. In this paper, as opposed to detecting unusual events [5], we provide a generic framework to answer a pre-specified question by querying a subset of sensors, whether human or devices, in the spatial area. Detecting untrusted responses is not the main focus of our paper. Instead, we assume that a trust algorithm can be built on top of our algorithm to eliminate untrusted responses.

Another related body of work is sensor networks [4] that include spatially and ubiquitously distributed autonomous sensors used to monitor physical and environmental conditions. Since the sensors are typically small, low-powered nodes, resource-constrained protocols emerged to preserve the energy of these devices. Examples of work targeting energy preservation include, but are not limited to [20], where the authors achieve geographic localization using a noise tolerant acoustic ranging mechanism to meet severe resource constraints. In [14], data aggregation methods were introduced and achieved significant performance gains in comparison to end to end routing. The work proposed in [10] implements a system that analyzes sensor behaviors and uncovers misbehavior corresponding to inefficient device usage that leads to energy waste. In contrast, our work focuses on the how to choose sensors to query based on their location while constraining the number of probes to a subset of the total number of sensors hence, preserving energy.

3. DISPNN: A TWO-STAGE QUERYING ALGORITHM

In our system, we have a two-dimensional grid and a number of nodes that can sense the environment around them. These nodes can be humans, artificial sensors, mobile phones or even robotic sensors. We are interested in answering questions of the form: "What is the answer to Question X in this grid?". For instance a question can be "Is it safe around you?", or " Did you feel the earthquake?". To find the answer, one approach would be to query all the nodes in the two-dimensional space and aggregate the findings. However, in many situations, such as during emergencies or in long term sensor deployments preserving resources such as energy is critical [17]. In this paper, we investigate how to answer the aforementioned question in the case of limited resources. Hence, the question becomes: *Given M nodes and N resources, where $N < M$, which N nodes should be queried to obtain the needed answer?*

If we tackle this question from a probabilistic point of view, then the straightforward answer is to try to select nodes with the same probabilistic distribution as the phenomenon. For instance, if we know that a certain phenomenon occurs uniformly in the two-dimensional grid, then we would have no bias in selecting the users to query, i.e. each user/object should have the same probability of selection to be queried. On the other hand, if we know the phenomenon we are interested in is more prevalent in certain areas of the grid as opposed to other areas, we should take that into consideration when we are selecting the nodes such that we query devices in the area of interest and fewer devices in areas where there is a smaller probability of occurrence.

The question becomes far more challenging if the distribu-

tion is not known or if it is time variant. In this case, we inquire if there is a systematic algorithm that can be used for querying/selecting users to spatially identify a phenomenon regardless of the probabilistic distribution or time variation. In the following sections, we describe DispNN: a two-stage algorithm that queries nodes without any assumptions about the distribution of events and succeeds in locating nodes that are close to the events and in covering up to approximately 90% of the incidents.

3.1 Algorithm description

We assume that there are M users in a two-dimensional grid and that the system that selects a node to query is bounded by N resources, where $N < M$. Each of the M users has a specific location in the grid, determined by a two-dimensional system, e.g. (x, y) or a (lat, long). We also assume that the selected nodes will fully cooperate and respond to the query. If needed, a pre-selection phase can be used to eliminate users that are not likely to co-operate, such as requiring the installation of an app to facilitate querying. Here, we focus on how to select N out of M nodes, where $N < M$, to identify both events occurring in the two-dimensional grid and nodes that are close to these events.

Given N nodes, DispNN divides the selection of nodes into two stages as depicted in Algorithm 1. The number of queried nodes in the first stage is determined by $\lfloor (FSP*N) \rfloor$ where FSP denotes the first stage percentage. As indicated in Table 1, the first stage percentage represents the fraction of nodes that will be selected to be queried in the first stage of DispNN. In the first stage, our goal is to select nodes that maximize the dispersion of their locations. This is accomplished by selecting the set of points, $\mathcal{P} = \{p(i),\ i \in \{1, ..., \lfloor (FSP * N) \rfloor\}$, that maximize the average distance between each point and its nearest neighbor as follows:

$$\underset{\mathcal{P}}{\operatorname{argmax}} \sum_{i=1}^{|\mathcal{P}|} \|p(i) - NN(p(i))\|^2 \qquad (1)$$

Algorithm 1 DispNN querying algorithm

```
1: function selectNodesFromGrid (FSP, N)
2:     selectedNodes = {}
3:     firstStageCnt = ⌊(FSP * N)⌋
4:     secondStageCnt = M - firstStageCnt
5:     firstStageNodes = maximizeDisp(firstStageCnt)
       //First Stage: Maximize dispersion with firstStageCnt
6:     nodesFeedback = feedback(firstStageNodes)
       //(a) Identify pivot nodes
7:     if nodesFeedback.size == 0 then
       //Second Stage:
       //(b) Get NNs for pivot users depending on quota
       //or maximize dispersion
8:         selectedNodes = maximizeDisp(secondStageCnt)
9:     else
10:        selectedNodes.append(firstStageNodes)
11:        firstStageQuota = calculate-
   Quota(firstStageNodes)
12:        for user_i in firstStageUsers do
13:            selectedNodes.append(KNN(user_i,
   firstStageQuota_i))    //Aggregate selected nodes
14:    return selectedNodes
```

where p represents a point in the $2D$ grid and NN represents the nearest neighbor; the distance is measured as the Euclidean distance. The algorithm attempts to maximize the dispersion with a percentage of the N resources up a certain number of trials controlled by the "maximization trials" setting, as defined in Table 1.

After the first stage of dispersion maximization, stage 2 consists of querying the sensors that were selected and evaluating the response of these sensors. The response provided is application dependent. For some applications, the query can be in the form of probing for some measurement and if that measurement exceeds/is less than a certain threshold, the system marks this as a positive response to the incident under investigation. In other applications, the query could be a simple yes-no question. An example is in [7], when an earthquake occurred on the East Coast in the US in 2011, the query was the question "Did You Feel It?".

Based on the node response in the first stage, we then proceed to a more fine-grained selection. The nodes that provide a positive response, where the definition of a positive response is application dependent, are called the *pivot nodes*. After response inspection, we divide the rest of the resources $(N - \lfloor FSP*N \rfloor)$ among the nearest neighbors for each of the pivot users. If no nodes provide a positive response, the second stage maximizes the dispersion of the location of nodes with the rest of the available resources. In our analysis, we test the cases of setting $FSP = 20\%, 40\%, 60\%$ and 80% of the N resources in the first stage.

This initial algorithm assumes that the first stage nodes will respond with unfalsified responses. To relax this assumption, we explore dividing the selection of the second stage users into two groups: a group that consists of the KNN of trusted pivot users, and another group that aims to maximize the dispersion. In the next section, we focus on studying DispNN with the assumption of having full trust in the nodes and discuss other variants of the algorithm in Section 5.

4. EXPERIMENTS

To quantify the performance of our algorithm, we perform multiple experiments with three types of data spread: clustered, uniform and long-tailed distributions. We also perform experiments on a real harassment dataset. In our experiments, we compare our algorithm in the selection of users to two alternative approaches as follows:

- Random user selection: For this approach, we select N nodes randomly based on a uniform distribution.
- Dispersion maximization (DispMax) selection: In this approach, N nodes are selected from the resources/crowd such that the dispersion of their locations is maximized.

4.1 Experiment variables

There are multiple variables that can be controlled to test the behavior of DispNN. Table 1 summarizes the most important. The environment settings are related to the size of the $2D$ matrix, the number of incidents, the distribution

Environment settings: • *matrix dimension:* the length and width of the $2D$ spatial matrix. We model the spatial area under investigation as a $2D$ square matrix. • *incident count:* number of incidents distributed across the cells of the spatial matrix • *resources or crowd count:* the M resources from which N will be chosen to query, where $N < M$.
Query settings: • *N:* the number of resources the system is limited by to query/sense • *first stage percentage (FSP):* the percentage of users/sensors of the N resources that will be selected to query in the first stage. • *k:* used to identify the KNN crowd individuals/sensors to an incident
Approximation settings: • *maximization trials:* number of attempts to maximize the dispersion of selected individuals/sensors from the crowd

Table 1: Parameters of DispNN.

of incidents across the matrix, and the number of resources from which to choose. In all of our experiments, except the case study, we use a 10 by 10 matrix. We show results for incident count of 50 and number of resources (M) of 100. We varied the environment settings, by changing the incident count and node count (M), in our experiments and DispNN still performs better than the random selection approach and the DispMax approach. We omit the results of these variations because of their consistency. Instead, we focus on varying the query settings to better understand DispNN. In this section, we vary the first stage percentage and leave the variation of the k setting to the following section. We also show results for $N = 30$, which constitutes 30% of the available resources (M). We notice that the gap between the performance of our algorithm and the other approaches increases when N decreases, and the performance of all approaches converges when N approaches M.

To compare the performance of our algorithm to random selection and dispersion maximization, we utilize two different metrics: count of nodes queried in the KNN of incidents, and the number of incidents covered by the nodes queried. For $i \in \{1, ..., \mathcal{I}\}$, let KNN_i be the KNN of the ith incident, where $\mathcal{I}$ denotes the total number of incidents. The two metrics are defined as:

- CNC: the absolute number of nodes in the KNN of each incident for all incidents. This is formally represented as follows:

$$Close\ node\ count = \sum_{i=1}^{\mathcal{I}} |(KNN_i \cap QN)| \quad (2)$$

  where QN (the "Queried Nodes" set) is the set of nodes selected for querying.

- Coverage: the number of incidents covered out of the total number of incidents that occur in the $2D$ matrix. We define an incident as covered if at least one of the nodes in the incident's KNN is queried. This is formally measured as:

$$Coverage = \sum_{i=1}^{\mathcal{I}} Coverage_i \quad (3)$$

where,

$$Coverage_i = \begin{cases} 1, & \text{if } (KNN_i \cap QN) \neq \phi \\ 0, & \text{otherwise} \end{cases}$$

In our experiments, we simulate the positive response. A node will provide a positive response if it is in the kNN of one or more incidents in the grid. Choosing a small value of k simulates a fine-grained phenomenon like street harassment. In contrast, a large value of k simulates a phenomenon with a larger range like an earthquake.

4.2 Clustered data experiments

Geographer Waldo R. Tobler's stated in the first law of geography: "Everything is related to everything else, but near things are more related than distant things." In this set of experiments, we assume that the incidents are related to each other, i.e. they form clusters across the $2D$ spatial matrix as seen in Fig 1. Our goal in these experiments is to study the performance of the different query algorithms when the events are clustered.

For this set of experiments, we vary the number of clusters in our $2D$ matrix from one to ten clusters while fixing the incident count to be 100. To enforce data variability, we model the size of each cluster as a random variable while ensuring that the aggregated size of all the clusters is equal to the crowd count. For each case of number of clusters, we average over 100 different configurations. Our objective is to measure the effect of variation of the first stage percentage on our performance metrics.

Figure 2 illustrates the results for CNC when varying the first stage percentage (FSP) from 20% to 80%. DispNN always outperforms the Random node selection and the Dispersion Maximization selection. Table 2 depicts the performance gain for CNC in comparison to the Random and Dispersion maximization approaches. As the amount of nodes

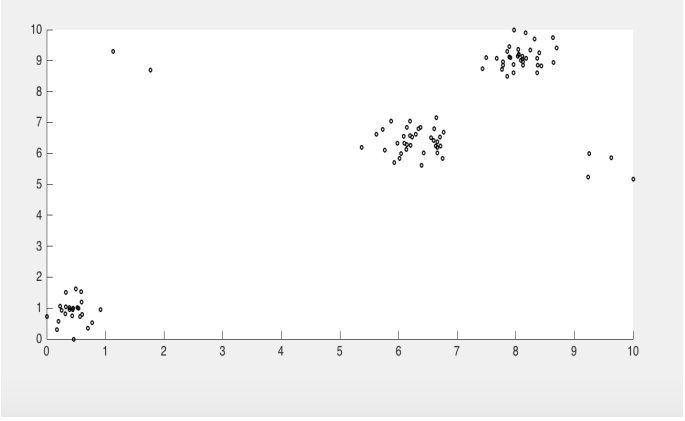


Figure 1: An example of a $2D$ spatial matrix with 5 clusters.

queried in the first stage decreases, CNC increases. This is due to the fact that when the first stage percentage decreases, the second stage resources increase under limited resources constraints, which focuses on nodes close to incidents detected in the first stage. On the other hand, incident coverage tends to increase as the first stage count increases. This is shown in Figure 3. Both CNC and Coverage tend to increase with the number of clusters until the number of clusters is four or five. Then, they decrease. The reason is that as the number of clusters increases in the grid, so does the probability of success of the first stage in identifying more incidents. On the other hand, as the number of clusters increase, the cluster size decreases, which results in fewer incidents per cluster.

4.3 Uniformly distributed data experiments

In the next set of experiments, the probability of occurrence of incidents is uniform across the grid, i.e. $P_I(j) = P_I(k)$ where $j \neq k$ and P_I denotes the probability of an incident occurring at a specific cell. We randomly generate 100 different matrices and average the results. We note that if we know that the distribution of the incidents is uniform, the best we can do is to choose N nodes uniformly. Using DispNN, we select N nodes without assuming any distribution about the incidents and compare the performance to the two alternative approaches. Figure 4 shows that DispNN with $FSP = 20\%$ achieves the highest CNC while Figure 5 shows that DispNN with $FSP = 80\%$ achieves higher coverage that the uniform random policy and it is close to the maximum coverage by an average of 1.32 incidents. Hence our algorithm achieves better node selection than the uniform scheme in terms of CNC when $FSP = 20\%$ and achieves better coverage than the uniform scheme when $FSP = 80\%$.

4.4 Long tail distribution

In this section, the incidents are generated according to a special case of the Long Tail distribution called the "Pareto principle". According to the Pareto principle, we assume that 20% of the matrix cells are home to 80% of the incidents, while 80% of the matrix cells are home to 20% of the incidents. We generate 100 different matrices applying the Pareto Principle randomly on the cells. We use a random uniform distribution to select 20% of the cells and generate 80% of the incidents uniformly for these cells and vice versa. Figure 6 shows that DispNN outperforms both the random and the dispersion maximization algorithms by up to 10.2% and 12.1%, respectively, in terms of CNC. This is due to the clustering of events in only 20% of the grid, which means that more than one incident is likely to occur in the same cell. So, if DispNN reaches a node close to an incident in one cell, this same node will cover more than one incident in the same cell. Dispersion maximization achieves the best incident coverage as shown in Figure 7. DispNN approaches the maximum incident coverage when $FSP = 80\%$ with a difference of 1.24 incidents on average. This shows that DispNN with $FSP = 80\%$ can still achieve a balance between CNC and incident coverage.

4.5 Case study: Hollaback harassment data set

After applying DispNN to the previously mentioned three distributions (clustered, uniform and long-tail), we wish to examine the algorithm under real incident distributions. To do that, we test our querying algorithm on a global street harassment dataset provided by Hollaback [11].

First stage percentage (FSP)	20%	40%	60%	80%
Performance gain over Random	63%	59%	35%	20%
Performance gain over Dispersion Maximization	68%	62%	40%	21%

Table 2: Performance gain of DispNN for clustered data in comparison to Random and Dispersion Max approaches averaged over all clusters.

4.5.1 Data overview

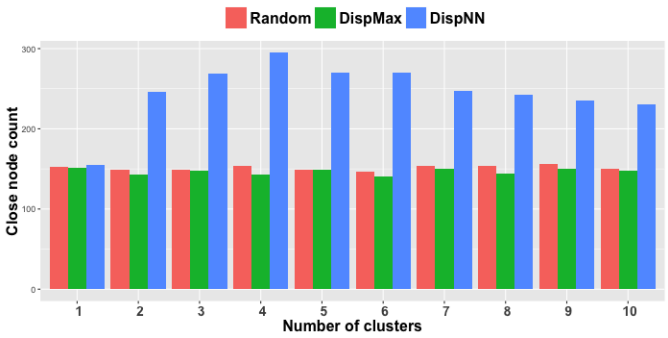
Hollaback is a non-profit movement powered by local activists in 92 cities and 32 countries to end street harassment. The Hollaback project collects data on street harassment events worldwide. Through the Hollaback phone app and the online platform, users can report stories of street harassment to share with the Hollaback community. This empowers victims to speak out about harassment and spread the word about the prevalence of these events. In some communities, local governments are informed in real-time about street harassment so that there is a system-wide level of accountability. In addition, the Hollaback app uses GPS to record a data set of street harassment event locations as a means of improving the collective understanding of street harassment and how it can be prevented. As of January 2016, over 8000 street harassment incidents have been recorded in the dataset since February 2011. It is on this data set that we wish to test DispNN.

4.5.2 Analysis

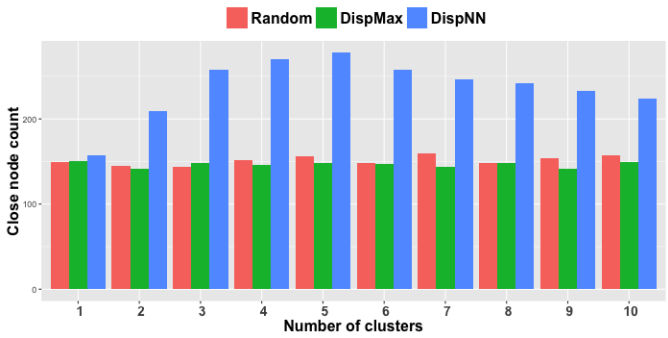
From the Hollaback dataset, we select cities for which we have enough harassment samples for statistical significance (i.e. more than 30 samples). We test the performance of Random selection, Dispersion maximization selection, and DispNN on six different cities: Paris, Brussels, Berlin, Baltimore, Buenos Aires and Istanbul. These cities were in the top ten cities with respect to the number of harassment reports in this dataset. In this paper, we show results for Paris, Brussels, and Istanbul. The results for Berlin, Baltimore, and Buenos Aires were consistent with the results shown in this paper.

As a first step, we parse the Hollaback dataset such that incidents reports are grouped by city. To do so, we use bounding box coordinates. We then draw the border lines for the different cities and remove any outliers from our datasets. Figure 8 shows the resulting distribution of events for the three cities. The Paris dataset contains 197 harassment incidents and covers an area of 28.2 mi^2, while the Brussels dataset contains 154 incidents covering a geographic area of 28.4 mi^2. For Istanbul, we divide the coverage into two areas: 87 reported incidents over an area of 138 mi^2 on the left of the Bosporus Strait and 69 mi^2 on the right.

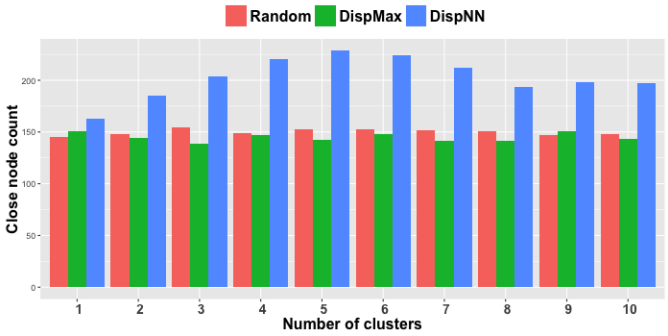
For each of the cities, we generate different variations of uniformly distributed crowds ($M = 100$) across the city. In this analysis, the parameters, matrix dimension and incident count, are taken directly from the dataset. In this case, we update the distance metric in Equation 1 and use



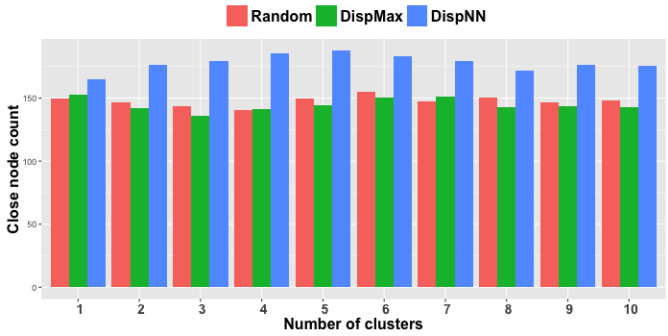
(a) CNC with $FSP = 20\%$.



(b) CNC with $FSP = 40\%$.



(c) CNC with $FSP = 60\%$.



(d) CNC with $FSP = 80\%$.

Figure 2: Average number of nodes close to the incidents (CNC) as FSP varies.

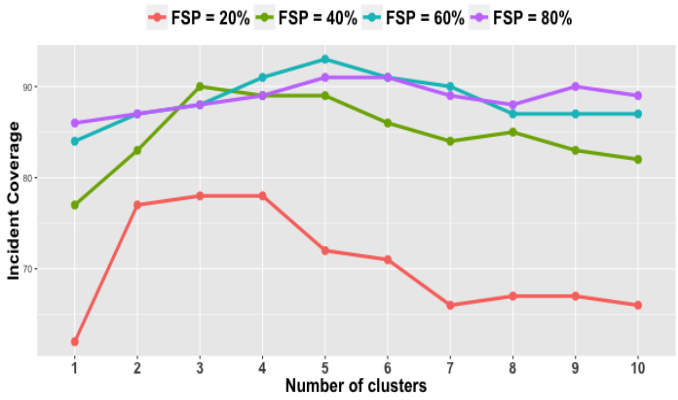


Figure 3: Incident coverage for different values of FSP.

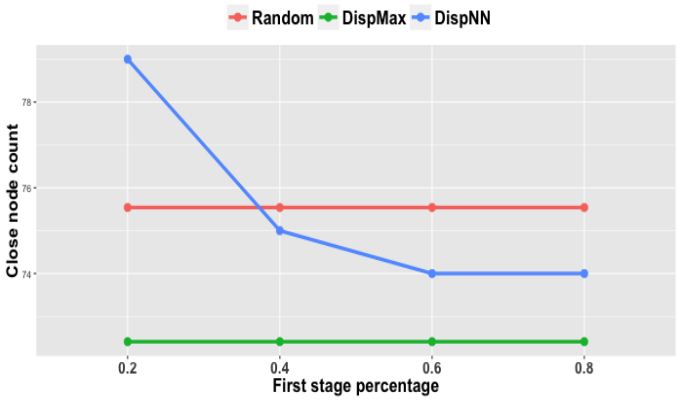


Figure 4: CNC for different values of FSP.

the Haversine formula to calculate the great-circle distance between two points as follows:

$$d = 2R * atan2(\sqrt{a}, \sqrt{1-a}) \quad (4)$$

where a is calculated as $\sin^2((\Delta\phi)/2) + \cos(\phi_1)\cos(\phi_2) * \sin^2((\Delta\lambda)/2)$; $\Delta\phi$ and $\Delta\lambda$ are calculated as the radian difference between the latitudes and longitudes, respectively; and R is the Earth's radius (mean radius = 6,371km). We measure CNC and Coverage for all three querying approaches and plot the results in Figures 9 and 10, respectively. DispNN outperforms both the Random and Dispersion Maximization in terms of CNC for all three cities. In terms of incident coverage, Figure 10 shows that dispersion maximization achieves maximum incident coverage. The figure also shows that DispNN can achieve this maximum by setting the first stage percentage to be 80%. Figures 9 and 10 suggest that there is an inherent tradeoff between accuracy and coverage under constrained resources which we discuss in detail in later sections. The figures also suggest that DispNN with $FSP = 80\%$ can achieve a balance between accuracy and coverage.

4.6 Stressing DispNN (k=1)

After applying DispNN to different datasets, we are interested in checking which schemes were able to query nodes that were closest to the incidents i.e. the first nearest neigh-

(a) Paris (b) Brussels (c) Istanbul

Figure 8: Distribution of harassment incidents across representative city datasets.

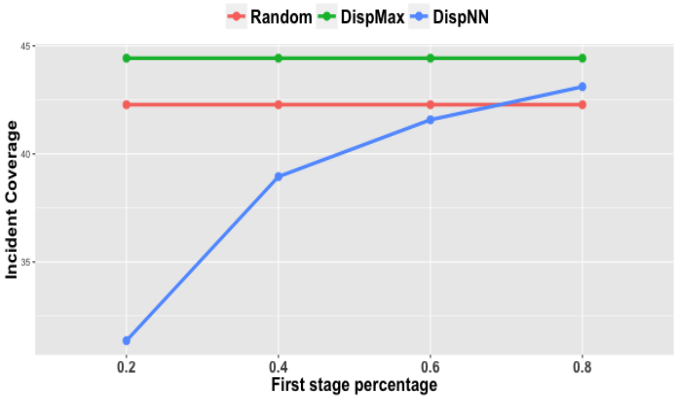


Figure 5: Incident coverage for different values of FSP.

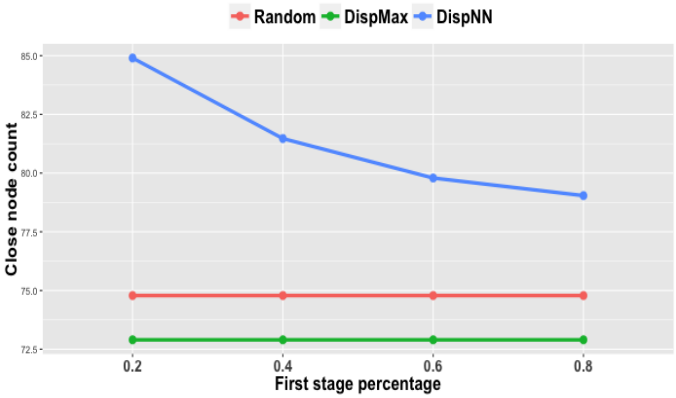


Figure 6: CNC for different values of FSP in the case of a long tail distribution.

bor to the incidents. This is beneficial, for example, when targeting first responders in an emergency scenario or in a spatial task distribution where we want to select the nearest neighbors to maximize spatial task assignment. This can be viewed as stressing the selection policies in order to determine which achieves a higher number of first nearest neighbors.

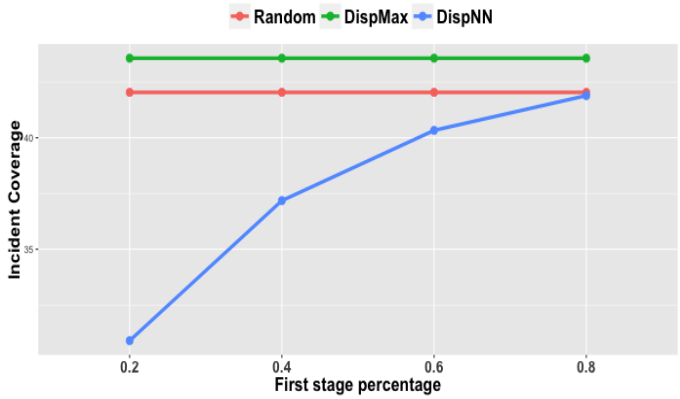


Figure 7: Incident coverage for different values of FSP in the case of a long tail distribution.

To study first nearest neighbors, we examine the Hollaback datasets for Paris, Brussels, and Istanbul. We examine the total CNC and for each incident, and determine whether we selected the first nearest neighbor in the queried users set. These results are shown in Table 3, where NN denotes nearest neighbors. For Paris, DispNN achieves a 9.5% increase in performance on average in selecting nearest neighbors in comparison to dispersion maximization and a 19% increase in comparison to random selection. For Brussels (Bruss), the performance gain is 14.2% and 21.35% in comparison to dispersion maximization and random selection, respectively. For Istanbul (Istan), the performance gains were 26.7% and 36.5%. These results show that DispNN is able to locate more first nearest neighbors to incidents. Locating more first nearest neighbors, in case of human involvement, can lead better spatial task assignment through minimizing the cost of travel to incidents.

5. DISCUSSION

5.1 Tradeoffs

In the previous section, we examined the performance of the DispNN querying algorithm in a variety of incident and node configurations; we observed that as FSP decreases, CNC tends to increase. We also observed that as CNC increases, when FSP decreases, the same node can be in the

K nearest neighbors for multiple incidents. This means that the algorithm tends to select central nodes that are in proximity to other incidents. This is beneficial in cases where the centrality of nodes is important to the problem, e.g. minimizing trip costs to these incidents and maximizing task assignments. This observation can be used to diversify feedback to improve accuracy i.e., instead of relying on a small number of nodes close to the incidents, we have a greater sample that can contribute to the measurement. On the other hand, as FSP increases so does the probability of catching more incidents in the spatial environment, which is crucial in applications where coverage is important and where a false positive is less expensive than a false negative. This is due to the fact that more nodes are selected in the first stage and fewer nodes in the second stage. It is no surprise, under resource constrained conditions, there is a tradeoff between accuracy and coverage.

5.2 Algorithm variants

Trust-based responses:
In the second stage of DispNN, we select users based on the pivot nodes that provide positive feedback in the first stage. To incorporate trust into DispNN, trust-based algorithms can provide feedback about certain nodes and the trustworthiness of their response. If some of the nodes queried in the first stage of the algorithm are deemed trust unworthy, the second stage can be divided into two querying steps. The first step is the KNN for the trustworthy-nodes, and the second is determination dispersion maximization.

Prior knowledge availability:
DispNN does not assume any knowledge about the distribution of events. Given some prior information about the distribution, the algorithm can be tailored to take this information into account. The idea is to divide the spatial area into bounding regions. For each region, we give a specific weight that reflects the probability of occurrence in that region. For example, Figure 11 shows Baltimore divided into four bounding regions denoted as br1, br2, br3, and br4. Using the knowledge that br3 has more incidents than br2, which has more incidents than br1 and br4, a higher weight should be given to br3. In particular, $br3_w > br2_w > br4_w > br1_w$ where the subscript w denotes the weight assigned to the bounding region. The next step would be to apply the algorithm on the different bounding regions taking into account the weights assigned when allocating resources as shown in Algorithm 2.

Algorithm 2 Bounding regions two-stage variation.

```
1: function SELECTUSERSBRS (BRs[], w[], N, FSP)
2:     selectedUsers = {}
3:     BRQuota = calculateQuota(BRs[], w[], N)
4:     for br in BRs do
5:         brQuota = BRQuota(br)
6:         brUsers = selectUsersFromGrid(FSP, brQuota)
7:         brUsers.append(brUsers)
8:     return selectedUsers
```

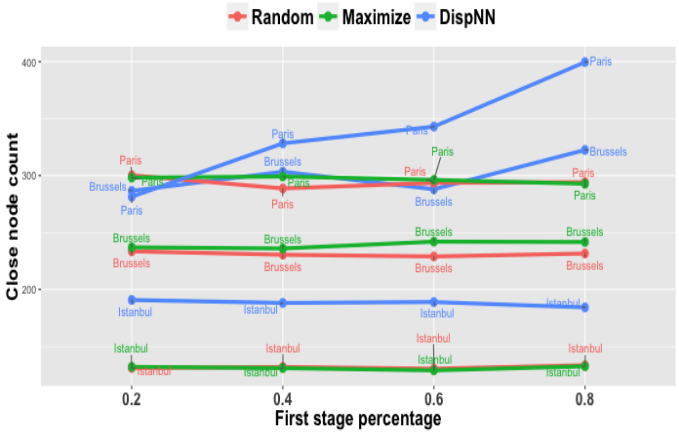


Figure 9: CNC for different values of FSP.

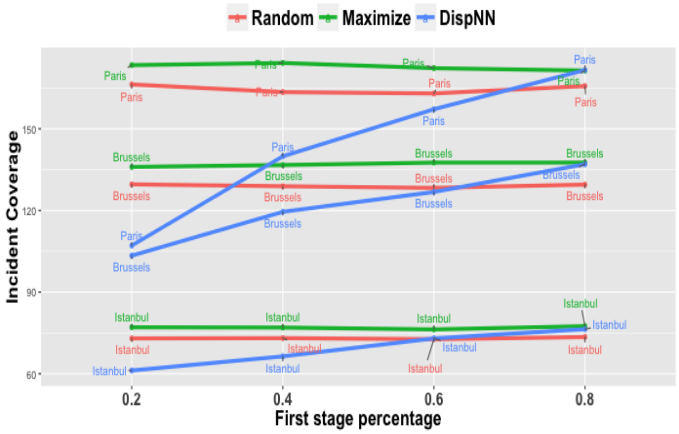


Figure 10: Incident coverage for different values of FSP.

City	Random	DispMax	DispNN: $FSP = 20\%$	DispNN: $FSP = 40\%$	DispNN: $FSP = 60\%$	DispNN: $FSP = 80\%$
Paris-NN	58	63	54	69	69	84
Paris-CNC	300	298	281	328	343	399
Bruss-NN	48	51	55	58	55	65
Bruss-CNC	233	237	286	303	288	322
Istan-NN	26	28	36	37	35	36
Istan-CNC	131	132	190	188	188	184

Table 3: First nearest neighbors (NN) count and CNC for Paris, Brussels, and Istanbul.

6. CONCLUSION

This paper proposed DispNN; a two-stage spatial querying algorithm that attempts to probe nodes that are closer to the incidents in a $2D$ spatial environment. The algorithm maximizes the dispersion of location in the first stage and selects the K nearest neighbors in the second stage based on node response. The experimental evaluation confirms the appli-

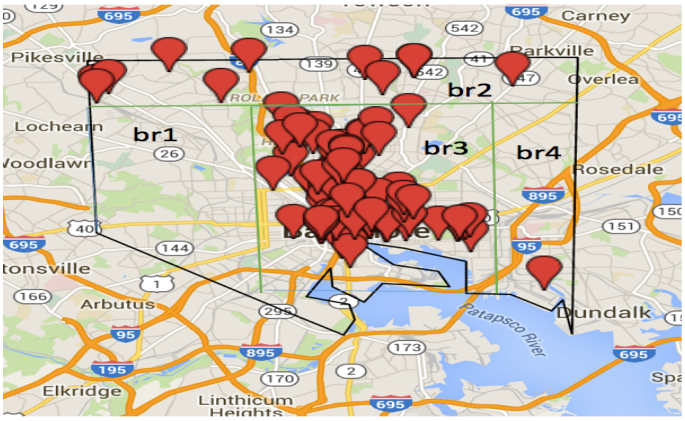


Figure 11: Baltimore divided into bounding regions depending on prior knowledge of harassment occurrence.

cability of proposed approach. Important aspects that need to be taken into consideration that could affect the choice of N include the sensing range and the sensing accuracy of the nodes. Nodes with more accurate sensing and larger sensing ranges will provide more robustness to DispNN and will lead to detecting more incidents with a smaller value of N. Another important parameter that can impact the performance of the algorithm is the distribution of the available nodes in the spatial environment. To ensure optimal performance, nodes should ideally be placed uniformly in the environment if the incident distribution is not known or following an approximate distribution of the incidents. The node placement can be controlled for some scenarios such as traffic or agriculture sensors. In other cases such control is more difficult particularly when human participation is involved. In that case, there are no guarantees that humans will be uniformly distributed across the spatial area under investigation. When choosing the number of nodes to query, another important factor is the granularity of the incidents. For instance, when using human as sensors, incidents such as disasters can likely be sensed/detected by numerous people in the range of the incident. On the other hand, incidents such as street harassment require the presence of the human geographically close to the incident; otherwise, the incident is unlikely to be detected. This work is applicable in many scenarios especially when preserving resources is important and when querying all nodes in range will cause disturbance to nodes or response implosions.

7. ACKNOWLEDGMENTS

The authors would like to thank Hollaback for sharing their collected dataset and taking the time to answer questions.

8. REFERENCES

[1] Eighty sexual assaults in one day - the other story of Tahrir Square. `http://www.theguardian.com/world/2013/jul/05/egypt-women-rape-sexual-assault-tahrir-square`, 2016. [Online; accessed March-2016].

[2] Human Shield Formed In Tahrir Square To Protect Women From Sexual Assault . `http://www.huffingtonpost.com/2013/07/03/human-shield-tahrir-square-egypt-sexual-violence_n_3540970.html`, 2016. [Online; accessed March-2016].

[3] E. Agapie, J. Teevan, and A. Monroy-Hernández. Crowdsourcing in the field: A case study using local crowds for event reporting. In *Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP'15)*, September 2015.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002.

[5] T. Blaschke, G. J. Hay, Q. Weng, and B. Resch. Collective sensing: Integrating geospatial technologies to understand urban systems-an overview. *Remote Sensing*, 3(8):1743–1776, 2011.

[6] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, 2013.

[7] A. Crooks, A. Croitoru, A. Stefanidis, and J. Radzikowski. # earthquake: Twitter as a distributed sensor system. *Transactions in GIS*, 17(1):124–147, 2013.

[8] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *the 21st International Conference on Advances in Geographic Information Systems (SIGSPATIAL GIS'13)*, pages 324–333, November 2013.

[9] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[10] R. Fontugne, J. Ortiz, N. Tremblay, P. Borgnat, P. Flandrin, K. Fukuda, D. Culler, and H. Esaki. Strip, bind, and search: a method for identifying abnormal energy consumption in buildings. In *the 12th International Conference on Information Processing in Sensor Networks (IPSN'13)*, pages 129–140, April 2013.

[11] Hollaback. Read and Share Stories. When it comes to street harassment, you are not alone. `http://www.ihollaback.org/share/`, 2015. [Online; accessed July-2015].

[12] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[13] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL GIS'12)*, pages 189–198, November 2012.

[14] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *the 22nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCS'02)*, pages 575–578, July 2002.

[15] R. Lee and K. Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *the 2nd ACM SIGSPATIAL International Workshop on Location based Social Networks*, pages 1–10, November 2010.

[16] Y. Liu, T. Alexandrova, and T. Nakajima. Using stranger as sensors: temporal and geo-sensitive question answering via social media. In *the 22nd international conference on World Wide Web (WWW'13)*, pages 803–814, May 2013.

[17] B. S. Manoj and A. H. Baker. Communication challenges in emergency response. *Communications of the ACM*, 50(3):51–53, 2007.

[18] D. Marco, E. J. Duarte-Melo, M. Liu, and D. L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *Information Processing in Sensor*

Networks, pages 1–16. Springer, 2003.

[19] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(4):24, 2008.

[20] J. Sallai, G. Balogh, M. Maroti, A. Ledeczi, and B. Kusy. Acoustic ranging in resource-constrained sensor networks. In *International Conference on Wireless Networks (ICWN '04)*, page 467, June 2004.

[21] M. Venanzi, A. Rogers, and N. R. Jennings. Trust-based fusion of untrustworthy information in crowdsourcing applications. In *the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13)*, pages 829–836, May 2013.

[22] M. Venanzi, A. Rogers, and N. R. Jennings. Crowdsourcing spatial phenomena using trust-based heteroskedastic gaussian processes. In *First AAAI Conference on Human Computation and Crowdsourcing (HCOMP'13)*, November 2013.

[23] H. Yu, C. Miao, Z. Shen, and C. Leung. Quality and budget aware task allocation for spatial crowdsourcing. In *the 2015 IFAAMAS International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, pages 1689–1690, May 2015.

[24] H. Yu, Z. Shen, C. Miao, and B. An. A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems. In *the 2013 IFAAMAS international conference on Autonomous agents and multi-agent systems*, pages 1315–1316, May 2013.