Utilizing the Hive Mind – How to Manage Knowledge in Fully Distributed Environments

Thomas Bach, Muhammad Adnan Tariq, Christian Mayer, and Kurt Rothermel

Institute for Parallel and Distributed Systems, University of Stuttgart, Germany,
⟨firstname.lastname⟩@ipvs.uni-stuttgart.de,
http://www.uni-stuttgart.de

Abstract. By 2020, the Internet of Things will consist of 26 Billion connected devices. All these devices will be collecting an innumerable amount of raw observations, for example, GPS positions or communication patterns. In order to benefit from this enormous amount of information, machine learning algorithms are used to derive knowledge from the gathered observations. This benefit can be increased further, if the devices are enabled to collaborate by sharing gathered knowledge. In a massively distributed environment, this is not an easy task, as the knowledge on each device can be very heterogeneous and based on a different amount of observations in diverse contexts. In this paper, we propose two strategies to route a query for specific knowledge to a device that can answer it with high confidence. To that end, we developed a confidence metric that takes the number and variance of the observations of a device into account. Our routing strategies are based on local routing tables that can either be learned from previous queries over time or actively maintained by interchanging knowledge models. We evaluated both routing strategies on real world and synthetic data. Our evaluations show that the knowledge retrieved by the presented approaches is up to 96.7% as accurate as the global optimum.

Keywords: Knowledge retrieval, Distributed knowledge, Confidence-based indexing, Query routing

1 Introduction

In many areas, such as social media or the Internet of Things (IoT) an enormous amount of data is produced. According to Cisco [7], the IoT alone will generate over 400 ZB of data annually by 2020. This data will mainly be in the form of raw observations, ranging from GPS positions to video streams, made by billions of interconnected devices, such as smart phones or myriads of ubiquitous sensors integrated in many modern devices. All these observations can be processed to generate knowledge. For example, by monitoring its GPS position over time, a smart phone can learn the working place and home of its owner [1]. Starting from such basic knowledge, more complex inferences about the owners habits

Published in R. Meersman, H. Panetto et al. OTM 2015 Conferences, LNCS ToBeDecided, pp. 1-18, 2015.

and daily routine are possible. Over time the phone can provide valuable knowledge to the user, for example, the best time to leave home dependent on the travel time to work. Greater benefit can arise if knowledge is not only generated and used locally, but also made accessible to others. In order to generate synergy effects, the devices of all users could form a hive mind. For example, by sharing knowledge about commuting habits, this hive mind can help to coordinate commuting, reduce traffic jams, save energy, and reduce CO_2 emissions [19].

One way of sharing knowledge is collecting at a central location, such as a cloud or a data center. This recent practice of massive data centralization has raised huge privacy concerns [16], as centralized data is regularly subject to breaches [11]. However, today there is nothing like one single central location to share all knowledge, the opposite is the case. Today every smart phone, every application and every company, every connected entity is gathering and analyzing as many observations as possible. In general, all these entities are not willing to share their, possibly private, observations directly [33]. For example users are uncomfortable sharing their exact GPS trajectories. However, in order to benefit from the knowledge and experience of other entities they might be willing to share gathered knowledge. This might be their knowledge about traffic at a specific road segment during the morning rush hour.

For instance, a public transport company could be interested in gathering knowledge about traveling patterns of its customers, while customers might be interested in more exact and up to date train timings and delay reports. In order to harness all these diverse sources of knowledge, we envision a scenario where many heterogeneous entities can share knowledge they have derived from many (possibly private) observations. Whenever an entity needs to acquire specific knowledge, such as knowledge about the traffic at a specific road, it can request this knowledge from other entities.

To enable such a fully decentralized system, all entities need to have the ability to search for (i.e., query) and retrieve specific knowledge generated by other entities. Locating knowledge in a fully decentralized system is a great challenge as we are facing three core difficulties. First, there exists no central index of all available knowledge. Second, knowledge is evolving fast, as previous observations may become outdated over time and new observations, made in different contexts, arise. Finally, knowledge is very heterogeneous, as each node learns individually, based on possibly high dimensional observations. Therefore, the learned knowledge is different, dependent on the number of observations and the context they are made in. For example, the traveling time between a users workplace and home might be dependent on weather, public holidays, time of the day, road works, etc. In consequence, the confidence with that a node can answer a specific query depends mainly on the context and amount of observations.

In order to tackle these issues, we propose two different routing strategies to retrieve specific knowledge by forwarding a query to the node that can answer it with high confidence. At the first glance, this problem looks as if it could be solved with a traditional peer-to-peer (P2P) approach. Traditional P2P-systems, however, cannot be applied as they are mainly designed for exact query searches.

In knowledge retrieval, many nodes may be able to answer a query. The confidence in these answers, however, may be completely different as nodes have gathered completely different knowledge. Furthermore, traditional P2P-systems assume homogeneous attributes for searching, however, knowledge is usually learned with respect to many different contexts that even differ between the devices.

In particular, our contributions are as follows: Based on our query model, we present two different, fully distributed, strategies to route a query to a node that can answer it with high confidence. To that end, we developed a confidence metric that takes the number and variance of observations into account. We evaluate both routing strategies on a synthetic- and a real-world data-set. In order to show that our algorithms can deal with heterogeneous, high dimensional data, we evaluated them in combination with an dimension selection technique to avoid the curse of dimensionality. Our results show, that the knowledge retrieved by the presented approaches is up to 96.7% as accurate as the global optimum.

2 System Model and Problem Formulation

We assume a fully distributed system of heterogeneous computing nodes $S = \{S_1, ..., S_k\}$. These nodes can join and leave the system at any time and fail temporarily or permanently. They can range from user owned portable edge computing devices, such as laptops and desktop computers, to private cloud instances or even third party services located in the nearby fog or big data centers. All nodes communicate on a peer-to-peer basis and form an acyclic, undirected topology. Maintaining such a topology, even under dynamic conditions, is a well studied problem (e.g., [6]) and is, therefore, not discussed in this paper.

Each node $S_i \in S$ collects raw observations o_j (e.g., GPS positions) produced by arbitrary sensors or mobile devices like smart phones. Machine learning algorithms, such as Markov Models, Conditional Random Fields (CRF), Bayesian networks, etc., can then be used to derive knowledge from these raw observations. In this paper, we use N-dimensional spaces to represent the raw observations o_j. This way, observable attributes, like time or position are represented by dimensions and every observation o_j as a point in space ($o_j \in \mathbb{R}^N$). For instance, in a traffic scenario, a concrete journey would be defined by its start-time, end-time, start-coordinates, and end-coordinates and represented as a point in N-dimensional space. Based on this representation, learning approaches, such as linear regression, multivariate adaptive regression, or time series analysis, can be applied to derive knowledge, e.g., the travel time between two locations with respect to the time of the day. In the following, we use the term knowledge model KM^{S_i} to represent the observations collected by a node S_i and the corresponding knowledge derived from these observations.

As all nodes make observations individually and, thereby, gather different observations in their knowledge models KM^{S_i}, the nodes have different confidence in the knowledge they have gathered. For example, a person traveling only in

one part of the city has intuitively higher confidence in estimated travel times in that part of the city than a person traveling mainly in another part.

To retrieve specific knowledge from the knowledge model, a query is issued. Such a query can be retrieving the travel time in the context of a journey at a specific time of the day. In an N-dimensional space ($\mathbb{R}^{\mathrm{N}}$), we define a query $\overrightarrow{q}$ as a tuple:

$$\overrightarrow{q} = (\delta, \omega, \rho_{\overrightarrow{q}}) \tag{1}$$

This tuple consists of a context (δ), a request (ω), and required confidence ($\rho_{\overrightarrow{q}}$). We define the context as a point in a subspace $\mathbb{R}^{\mathrm{C}} \subsetneq \mathbb{R}^{\mathrm{N}}$, where N is the total number of dimensions and C is the number of dimensions in the subspace of the context.

$$\delta \in \mathbb{R}^{\mathrm{C}} : \mathrm{C} < \mathrm{N}. \tag{2}$$

Similarly, the request is defined as an element in a different subspace $\mathbb{R}^{\mathrm{R}} \subsetneq (\mathbb{R}^{\mathrm{N}} \setminus \mathbb{R}^{\mathrm{C}})$, where R is the number of dimensions of the requested subspace.

$$\omega \in \mathbb{R}^{\mathrm{R}} : \mathrm{R} \leq \mathrm{N} - \mathrm{C}. \tag{3}$$

In a traffic scenario, for example, the knowledge models could consists of four dimensions (source, destination, travel time, and time of the day). A query $(\delta, \omega, \rho_{\overrightarrow{q}})$ to retrieve travel time for a specific journey can be structured as follows: context (δ) = (source, destination, time of the day) and request (ω) = (travel time) and confidence ($\rho_{\overrightarrow{q}}$) = 0.8.

Based on the above discussion, we can now define the concrete knowledge retrieval problem. Given i) a dynamic set of nodes $S = \{S_1, ..., S_k\}$, ii) continuously evolving and heterogeneous knowledge models KM^{S_i} maintained by each node $S_i \in S$ and iii) a knowledge retrieval query $\overrightarrow{q}$, our objective is to find the node $S_j \in S$ that can answer the query $\overrightarrow{q}$ with confidence $\rho > \rho_{\overrightarrow{q}}$.

In the following, we first introduce the notion of confidence and establish a metric to determine the confidence of a knowledge model KM^{S_i} to answer a specific query $\overrightarrow{q}$ (cf. Sec. 3). Afterwards, we present two algorithms that utilize the confidence metric to route a query for specific knowledge towards the node that can reply with the required confidence (cf. Sec. 4). Finally, we thoroughly evaluate and compare the performance of the proposed routing algorithms on synthetic and real-world data (cf. Sec. 5).

3 Confidence

In this section, we develop a notion of confidence and introduce a metric that reflects the confidence with that a query can be answered by a specific knowledge model KM^{s_i}. As observations in a knowledge model are usually neither concentrated in a single area of the model, nor distributed equally, it is not desirable to assign a single confidence value to a whole knowledge model. Instead, we use clustering to group similar observations and determine the confidence of these clusters individually.

3.1 Clustering

In order to cluster the observations in a knowledge model, we can use well known algorithms, such as K-means [5], spectral clustering [28], hierarchical clustering [12], etc. However, these algorithms need to know the number of clusters (denoted as K) in advance. Determining the number of clusters, however, is not easy and is usually based on preknowledge or practical experience. In our case, choosing the number of clusters K is even more difficult, as different knowledge models may witness highly heterogeneous observation patterns.

Clearly, the observations in each knowledge model are directly dependent on real world events. These real world events are usually highly chaotic as they are based on a vast amount of influence factors (Butterfly Effect). Therefore, the observations stored in the knowledge models can be treated as independent random variables. Consequently, the arithmetic mean of these observations follows a Gaussian distribution according to the central limit theorem [14]. Based on this we can conclude that the observations in each knowledge model are a combination of several Gaussian distributed clusters, located in different "areas" of the model. To determine the actual number of clusters, we use the following algorithm: We start by assuming a single cluster ($K = 1$) in a knowledge model and perform a statistical test for the hypothesis that all observations of the cluster follow a Gaussian distribution. If this test fails, we increase the assumed number of clusters by one ($K = K + 1$) and repeat the process. This process terminates, when all observations assigned to each cluster are following a Gaussian distribution. This approach is similar to the G-Means algorithm [9].

3.2 Cluster-Based Confidence

Now that the clusters are obtained, in the following we present how to determine the confidence in these clusters. Usually, machine learning algorithms are performed on a large number of observations. If not enough observations are available (for example, in the region of a query), machine learning algorithms can not derive meaningful information. Thus, we base our confidence metric on the amount and variance of observations in a cluster. The combination of both parameters is important as they can be misleading if interpreted individually. The number of observations in a cluster for example does not tell anything about the cluster's usability for learning, as the observations can be highly variant. Consider for instance, a cluster that models the waiting time at a busy intersection of a road. If the cluster consists of observations scattered over the whole day, then it might not be qualified to answer a query for the waiting time at a specific time of the day. On the other hand, low variance alone is also not adequate, as the variance can be based on a very small number of observations.

$$\rho(\overrightarrow{q}, \overrightarrow{c}_c, \Sigma_c, N_c) = N_c \cdot f_N(\overrightarrow{q}, \overrightarrow{c}_c, \Sigma_c) \tag{4}$$

This duality is reflected in our confidence metric ρ (Eq. 4) that consists of a N-dimensional Gaussian distribution function f_N, multiplied by the number

of points (N_c) in a cluster c. The Gaussian distribution function f_N is characterized by the mean value $\overrightarrow{c}_c$ (also termed as centroid) and variance Σ_c of the cluster c. This way, the confidence value behaves according to the distribution of observations and is proportional to the total number of observations in a cluster. We used a Gaussian distribution function in Eq. 4 for two reasons. First, as stated in Sec. 3.1 the observations in each cluster are following this distribution. Second, it enables the confidence in a cluster to degrade exponentially, such that queries $\overrightarrow{q}$ close to the centroid $\overrightarrow{c}$ of a cluster have high confidence, while with a continuously growing distance to the centroid, the confidence gets exponentially worse. As our knowledge models are N-dimensional spaces, clusters can have different variance in each dimension. Moreover, clusters can be clinched or rotated in different dimensions. Thus we use a multivariate Gaussian distribution function (cf. Eq. 5) that uses a covariance matrix Σ_c to take the variance in the different dimensions into account. Figure 1(a) shows the confidence for a cluster in a two dimensional knowledge model. The confidence of a cluster to answer a certain query is then the confidence value ρ of a cluster c at the point of a query $\overrightarrow{q}$, i.e. $\rho_{\overrightarrow{q}} = \rho(\overrightarrow{q}, \overrightarrow{c}_c, \Sigma_c, N_c)$.

$$f_N(\overrightarrow{q}, \overrightarrow{c}_c, \Sigma_c) = \frac{1}{(2\pi)^{0.5N}|\Sigma_c|^{0.5}} \exp^{-0.5(\overrightarrow{x}-\overrightarrow{c}_c)^T {\Sigma_c}^{-1}(\overrightarrow{q}-\overrightarrow{c}_c)} \tag{5}$$

3.3 Discussion

The above presented confidence metric can be used to compare the confidence of two knowledge models, however, the value of confidence is unbounded. This makes it difficult to determine when a knowledge model has "good enough" knowledge or is saturated (as each cluster in a knowledge model can have arbitrarily large number of observations). To solve this problem, we propose to artificially bound the confidence ρ^B as shown in Eq. 6.

$$\rho^B(\overrightarrow{q}, \overrightarrow{c}, \Sigma, N_c) = min(1, N_c \cdot f(\overrightarrow{q}, \overrightarrow{c}, \Sigma)) \tag{6}$$

The behavior of this bounded confidence function (ρ^B) is shown in Fig. 1(b). The figure shows the confidence with that a cluster can answer a query with respect to the distance between a query and the cluster center. According to Eq. 6 we assume a saturation when the confidence value exceeds 1. On the left side of Fig. 1(b), clusters with the same variance and different number of observations are compared. On the right side, clusters with the same number of observations and different variance are compared. This shows, how the saturation of a cluster is dependent on the variance and number of observations. In essence, clusters with low variance need less observations to saturate then clusters with high variance.

Another approach would be to employ domain specific knowledge to define when a cluster is saturated or has "good enough knowledge". For instance, by introducing a threshold that specifies the minimal distance between two observations in a cluster. If all the observations of a cluster obey the threshold

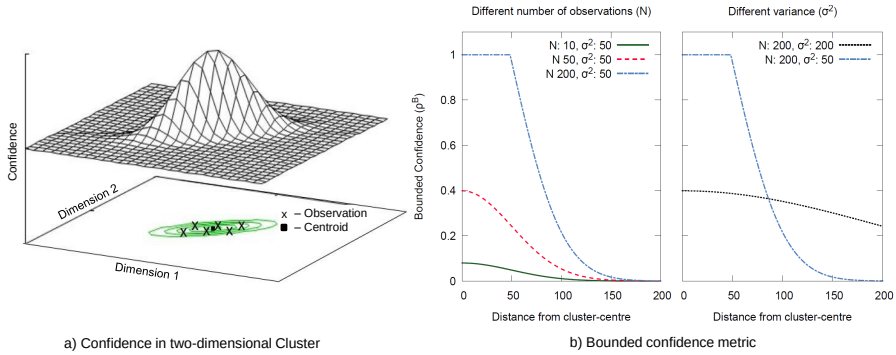

Fig. 1. Confidence dependent on distance to cluster-center, variance, and number of observations. Multivariate Gaussian distance function for all queries.

criteria, the cluster is assumed to have maximal confidence. Mathematically this threshold can be defined as the mean distance between observations in a cluster.

4 Routing

Now that the confidence metric is defined, we describe the routing of queries towards the nodes that can reply with the desired confidence. In particular, each node S_i maintains a routing model $RM_{S_n}^{S_i}$ for each direct neighbor $S_n \in S \setminus S_i$. A routing model $RM_{S_n}^{S_i}$ is an N-dimensional space (similar to the knowledge model) and summarizes the knowledge that is reachable (by forwarding the query) through the respective neighbor S_n.

On receiving a query $\overrightarrow{q}$ (cf. Sec. 2) a node S_i decides whether to answer $\overrightarrow{q}$ based on its local knowledge model KM_{S_i} or to forward $\overrightarrow{q}$ to one of its neighbor $S_n \in S \setminus S_i$. In more detail, a node S_i uses its knowledge model KM_{S_i} to calculate the confidence ρ at the point of the query $\overrightarrow{q}$ and reply $\overrightarrow{q}$ locally if $\rho > \rho_{\overrightarrow{q}}$. Otherwise, S_i uses the routing model $RM_{S_n}^{S_i}$ maintained for each of its neighbor S_n to calculate the estimated confidence with which $\overrightarrow{q}$ can be answered if forwarded to that respective neighbor S_n. The query $\overrightarrow{q}$ is finally routed to the neighbor with the highest estimated confidence. In the following, we present two different strategies to maintain the routing models and route queries.

4.1 Knowledge Aggregation Based Routing

With *Knowledge Aggregation Based Routing* (KAR) the routing models $RM_{S_n}^{S_i}$ that a node S_i maintains for each neighbor S_n are managed in a proactive fashion. In general, a routing model contains the summary of the knowledge reachable through a neighbor. This is done by enabling routing models to store the (subset of) N-dimensional observations that are reachable through the respective neighbors. The observations stored in a routing model are clustered similar to that of knowledge model and, thus, the calculation of confidence is alike.

To maintain routing models, each node S_n sends a compact representation of the clusters in its knowledge model KM_{S_n} to all of its neighbors. This compact representation includes the number of observations N_c, the centroid $\overrightarrow{c}_c$, and the covariance matrix Σ_c of each cluster c in the knowledge model. Moreover, the node S_n also sends to each neighbor S_i the compact representation of the observations stored in the RM of its other neighbors. This is required to ensure that the routing model not only contains the summary of the knowledge available in the knowledge model of the direct neighbor but also the knowledge reachable by routing the query through that neighbor.

On receiving the compact cluster representation from S_n, the node S_i can regenerate clusters along with all the observations in its routing model $RM_{S_n}^{S_i}$, representing the knowledge available through S_n. A complete regeneration is not very scalable for clusters containing a large number of observations, therefore, we introduce a regeneration factor $\delta \in (0, 1]$ that controls the fraction of observations that are regenerated $(N_c \cdot \delta)$.

To regenerate the clusters described by the received compact cluster representation $(N_c,\ \overrightarrow{c}_c,\ \Sigma_c)$, we use an approach described in [10]. This approach consists of three steps, shown in Fig. 2. First, we generate the desired number of observations $(N_c \cdot \delta)$ following a multivariate Gaussian distribution. Second, we skew and rotate the resulting point-cloud (the new cluster) according to the provided covariance matrix Σ_c. In the final step, we translate each point of the new cluster by the given centroid $\overrightarrow{c}_c$.

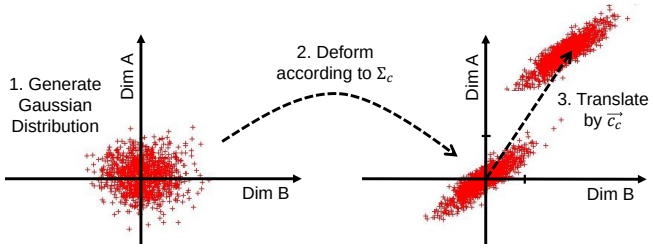

Fig. 2. Transforming a normal distributed cluster into a cluster rotated and skewed according to given compact cluster information.

Fig. 3 shows the KAR strategy for a network consisting of four nodes (S_1, S_2, S_3, S_4). To reduce complexity only relevant routing and knowledge models are shown. Initially, all routing models of node S_1 (i.e., $RM_{S_2}^{S_1}$ and $RM_{S_4}^{S_1}$) are empty, as shown in Fig. 3(a). Afterwards, S_2 and S_4 generate the compact cluster representation for their knowledge and routing models (cf. Fig. 3(b)) and send these representations to S_1 (cf. Fig. 3(c)). From the received compact representations S_1 regenerates its routing model for S_2 and S_3 (i.e., $RM_{S_2}^{S_1}$ and $RM_{S_4}^{S_1}$ respectively), as depicted in Fig. 3(d).

As the knowledge models are subject to continuous change, each node is responsible for keeping the routing models, maintained at other nodes, up to

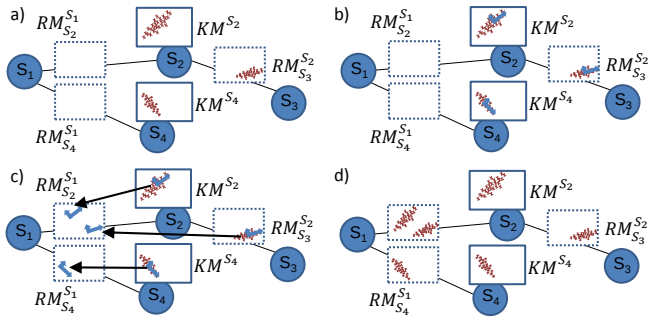

Fig. 3. Propagating the knowledge and generating the routing models.

date. If a node detects a significant change of knowledge in its own models, it generates new compact cluster representation and forwards it to its neighbors.

Routing a Query Routing of a query works as described in the beginning of this section. To determine the confidence in both models (i.e., KM and RM), we use the confidence metric described in Sec. 3.

4.2 Query Learning Based Routing

In a large-scale system, the routing models of the previously introduced KAR strategy require frequent updates. This is due to the continuously evolving knowledge models, as well as the unpredictable churn caused by joining and leaving nodes at arbitrary times. To reduce the overhead of managing the routing models, we designed a query learning based routing strategy (QLR) that continuously learns the routing models of a node, based on previously routed queries. This means, whenever a query was answered successfully, a feedback is send backwards on the routing path to the source of the query. This feedback consists of the query $\overrightarrow{q}$ and the confidence $\rho_{\overrightarrow{q}}$ with that it was answered. Whenever a node S_i receives such a feedback from a neighbor S_n, it adds the query-confidence pair $(\overrightarrow{q}, \rho_{\overrightarrow{q}})$ to its respective routing model $RM_{S_n}^{S_i}$ (cf. Fig. 4(a)).

Routing a Query With QLR, routing works exactly as described in the beginning of this section. However, as we are storing query-confidence pairs in the routing models, instead of observations, we have to estimate the confidence for an incoming $\overrightarrow{q}_{in}$ (i.e., the confidence with that the incoming query $\overrightarrow{q}_{in}$ can be answered or forwarded) a bit different. This is done in three steps: First, the query-confidence pairs in each routing model $RM_{S_n}^{S_i}$ are clustered with the weighted K-means clustering algorithm [5], where the queries represent points that are clustered and their respective confidence value is used as a weight (cf. Fig. 4). In consequence, the centroid of each cluster is not in the middle of the

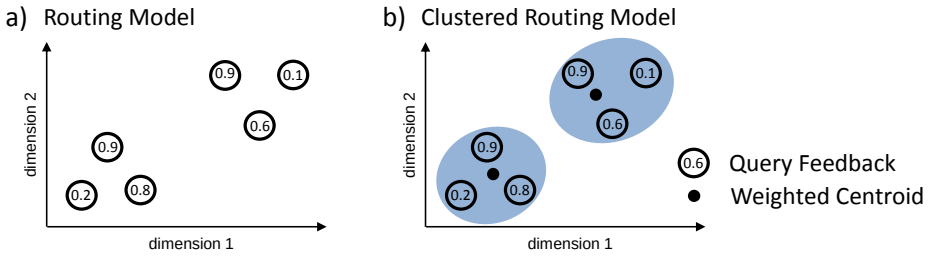

Fig. 4. a) A routing model containing query feedback with confidence values. b) Routing model after clustering.

cluster, but rather drawn towards the query-confidence pairs that have high confidence values (cf. Fig. 4(b)). We named such a centroid *weighted centroid*, $\overrightarrow{c}_w$. Second, we calculate the euclidean distance d between the incoming query $\overrightarrow{q}_{in}$ (for which the confidence has to be estimated) and the weighted centroid $\overrightarrow{c}_w$ (i.e., $d = \overrightarrow{q}_{in} - \overrightarrow{c}_w$). Third, we estimate the confidence of incoming query $\rho_{\overrightarrow{q}_{in}}$ by weighing this distance d according to the distribution of confidence values in the query-confidence pairs of the cluster, represented by the distribution function $f_c(d)$ (i.e., $\rho_{\overrightarrow{q}_{in}} = f(d)$).

To determine this distribution function $f_c(d)$, we start with the generic form of a Gaussian function (cf. Eq. 7), dependent on the parameters a_1 and a_2.

$$f_c(x) = a_1 exp(\frac{-x^2}{2a_2^2}) \tag{7}$$

To calculate these parameters we need two additional equations that specify our requirements for the distribution function f_c. The first requirement states that if the distance between the weighted centroid $\overrightarrow{c}_w$ and the query $\overrightarrow{q}_{in}$ is zero, the confidence should be equivalent to the confidence value of the weighted centroid μ_{conf} (Eq. 8).

$$f_c(0) = \mu_{conf} \tag{8}$$

The second requirement states that the confidence of a cluster should decrease with increasing distance between incoming query $\overrightarrow{q}_{in}$ and $\overrightarrow{c}_w$, proportional to the standard deviation of the confidence values σ_{conf} in the query-confidence pairs, i.e., for high standard deviation in the confidence values of the query-confidence pairs in the cluster, f_c should decrease slower than for low standard deviation. This is expressed by Eq. 9, where σ_c is the standard deviation of the cluster, μ_{conf} the confidence value of the weighted centroid $\overrightarrow{c}_w$ and σ_{conf} the standard deviation of the confidence values of the query confidence pairs.

$$f_c(\sigma_c) = |\mu_{conf} - \sigma_{conf}| \tag{9}$$

Based on above two requirements we can now determine a_1 and a_2 and get the following distribution function.

$$f_c(d) = \mu_{conf}(\frac{\mu_{conf} - \sigma_{conf}}{\mu_{conf}})^{\frac{d^2}{\sigma_c^2}} \tag{10}$$

Discussion As described above, QLR uses past queries to keep routing models up-to-date. However, sometimes, it might be necessary to proactively explore the knowledge reachable through different neighboring nodes. This is, for example, necessary to bootstrap the system or when not enough queries are routed through a node. In this case, a node issues exploration queries. This can be done randomly or interest-based, i.e., a node can either try to learn about an area of interest or just a randomly selected area. An exploration query is a regular query with additional selectivity and hop count parameters, that control the magnitude of exploration. The *selectivity* parameter controls the number of neighbors to which a query is sent. The *hop count* parameter controls the depth of exploration and is decreased on reception by every node.

Furthermore, in dynamic environments, the query-confidence pairs stored in the routing models may become stale and should be removed. We use a *decay based* approach, where the confidence values of the query-confidence pairs $(\overrightarrow{q}, \rho_{\overrightarrow{q}})$ stored in the routing models decay according to a specified decay factor α over time $\rho_{new} = \rho \cdot \alpha$. When the confidence value has become lower than a certain threshold, the query-confidence pair is removed from the routing model.

5 Evaluation

In this section, we evaluated the knowledge retrieval strategies (KAR and QLR), presented in Sec. 4, w.r.t. network size, message overhead and retrieval quality. We also evaluated the influence of protocol specific parameters such as hop count and selectivity for QLR and the regeneration factor δ for KAR. Furthermore, we show that our routing strategies can work, even when a large number of dimensions in the knowledge models contain noise. To evaluate the performance in the presence of noise, we used a dimension selection algorithm in combination with QLR.

We implemented our retrieval strategies on top of the network simulator PeerSim [18] and performed our evaluations on a compute cluster, consisting of 24 Intel®Xeon®, 3.00GHz CPUs, with a total of 377.8GB of RAM. We measured the retrieval quality θ as retrieved confidence divided by global maximum confidence ($\theta = \frac{\rho_{ret}}{\rho_{max}}$).

We performed our evaluations on synthetic and real world data. To produce synthetic data we randomly generated clusters in the knowledge models of each node in a way, that neighboring nodes contained similar observations. Such a localization is realistic in many scenarios, for example, when neighboring devices collect similar observations, or can be created by an overlay network. We plan to cover the creation of such an overlay network in future work. To evaluate our approach on real world data, we used the GeoLife data set [32, 30, 31]. This data set represents mobility data and contains 17,621 GPS trajectories gathered by 178 people, mainly around Bejing.

5.1 Knowledge Aggregation Routing (KAR)

In the following, we evaluated the Knowledge Aggregation Routing (KAR) strategy presented in Sec. 4.1. In Fig. 5, we measured the retrieval quality (θ) for different network sizes, real world, and synthetic data. We compared our results to a random routing strategy, also using the real world data set, because this strategy can be seen as a lower bound for routing performance. During this experiment we used a regeneration factor of 1 ($\delta = 1$) (cf. Sec. 4.1). The reason for the reduced retrieval quality of the real world data is due to massively overlapping and highly similar knowledge in the knowledge models. This leads to suboptimal routing decisions, because a node cannot clearly distinguish the best neighbor to forward the query. However, as we can see, the retrieved quality is still significantly better than the random approach.

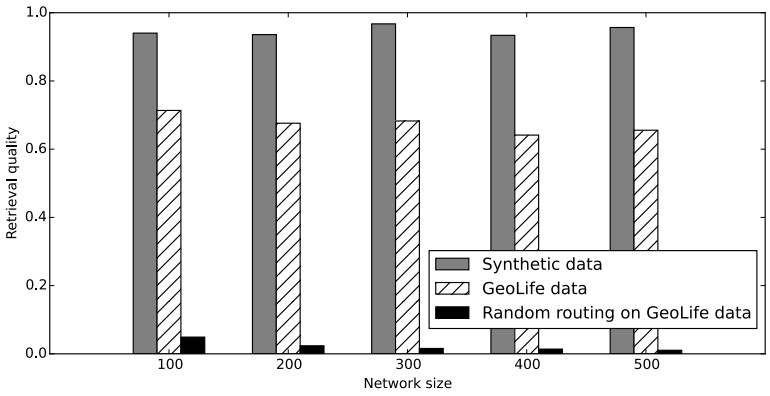

Fig. 5. Retrieval quality of KAR for different network sizes.

In Fig. 6 (left) we evaluated the retrieval quality for different regeneration factors δ (cf. Sec. 4.1). Overall, the routing quality decreases only slightly with the regeneration factor, as long as all clusters have a certain minimal size which is the case for the synthetic data and the synthetic data with GeoLife sized clusters, where we used the average cluster size of the GeoLife data set. However, the GeoLife data set also contains many clusters with very few observations. If the number of observations in these small clusters is further reduced the retrieval quality becomes worse.

To get an indicator, if the real world data provided by the GeoLife data set can be modeled as a multivariate Gaussian distribution, we performed the Kolmogorov Smirnov goodness-of-fit test. The subject of this test is the hypothesis, that we can represent a real world cluster by its respective multivariate Gaussian distribution. In statistical testing, the so called p-value is calculated to indicate if the hypothesis under test needs to be rejected. A large p-value provides evi-

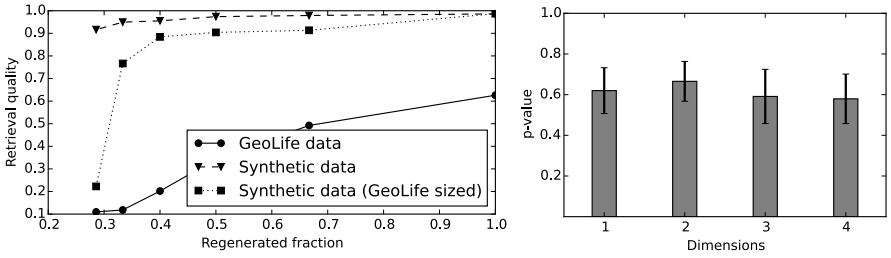

Fig. 6. Left: Routing quality for different regeneration factors (δ). Right: The p-values for clusters of different dimensionality.

dence that the hypothesis is true. We tested 14000 real world clusters, obtained from the GeoLife data set and plotted the mean of the p-value for up to four dimensional clusters in Fig. 6 (right). With a p-value significantly above the usual threshold of 0.05, we can conclude that it is reasonable to use multivariate Gaussian distributions to represent real world clusters.

5.2 Query Learning based Routing (QLR)

Similar to the evaluations for KAR, we determined the retrieval quality of QLR for different network sizes, real and synthetic data. As we can see in Fig. 7, the retrieval quality of QLR on synthetic data is not as high as with KLR. The reason is that in KAR every node regenerates clusters similar to the "original" clusters of its neighbors in its routing models. This leads to a precise representation of the clusters in the routing models. However, the advantage of QLR lies in its higher retrieval quality for real world data. Because QLR is building its routing models based on the success of previous queries, it can better reflect the real world distribution of observations in the knowledge models of its neighbors.

In QLR, each node has to gather a number of queries in its routing model, before reasonable routing performance can be reached. This *learning phase* is clearly visible in Fig. 8, where we compared the bootstrapping of QLR with KAR. We can see, that the retrieval quality for QLR is gradually increasing with an increasing amount of queries, while the retrieval quality of KAR immediately jumps from low to high after all routing models have been successfully generated.

We use exploration queries in QLR to probe the knowledge available in the network and keep routing models up-to-date. In our system, two parameters (selectivity and hop count) controls the spread of these exploration queries. In Fig. 9, we evaluated the influence of these parameters on the retrieval quality and message overhead. We measure the exploration query overhead as the number of exploration queries per "retrieval" query. Comparing message overhead and retrieval quality, we can see that it is not necessary to choose high values for selectivity and hop count. Instead, it seems adequate if each node explores its closer neighborhood. For instance, setting selectivity and hop count to two results in high retrieval quality and low message overhead.

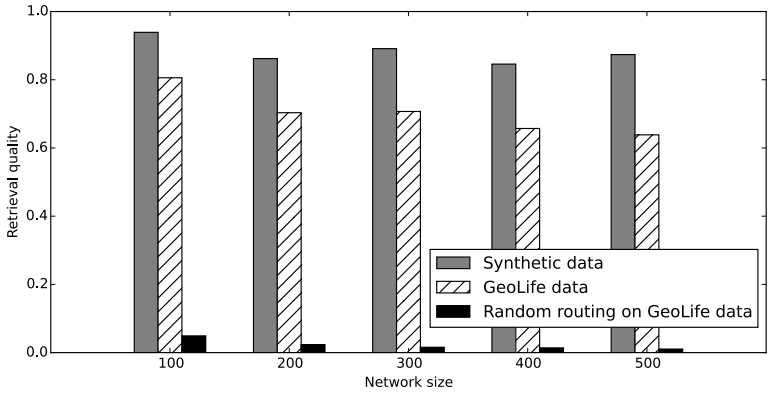

Fig. 7. Retrieval quality for different network sizes.

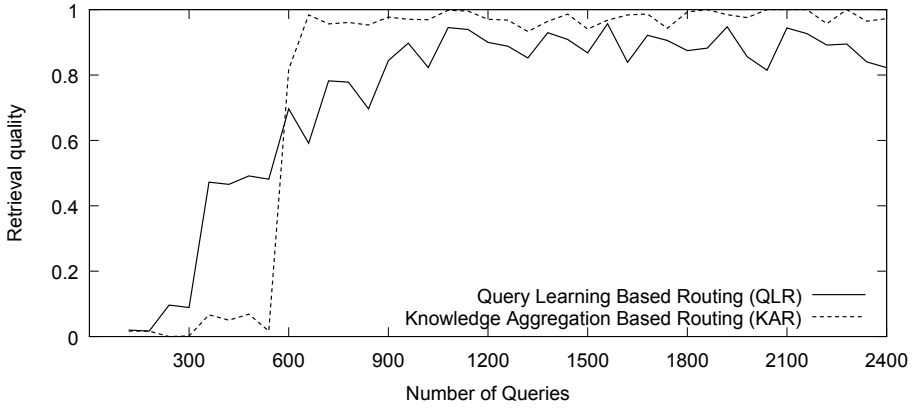

Fig. 8. The retrieval quality w.r.t. the number of queries for KAR and QLR.

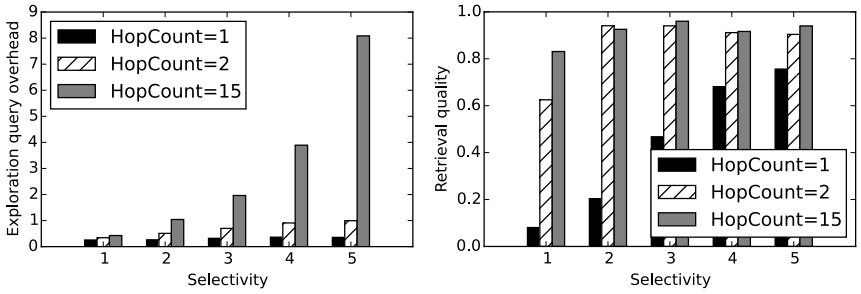

Fig. 9. Impact of hop-count and selectivity on message overhead and retrieval quality.

As already mentioned in Sec. 1, the knowledge of the different nodes can be heterogeneous and high dimensional, which may result in sub optimal routing. In order to deal with this challenge, literature proposes various dimension selection approaches [29, 15]. We extended QLR to select the most important dimensions for query routing using a simple variance based dimension selection approach. In a nutshell, our approach determines the important dimensions of a routing model based on the variance of the observations in each dimension. If the observations in one dimension exceed a certain variance threshold, the dimension is marked as not important. To evaluate this, we started with four dimensional observations and added up to 20 highly variant (noisy) dimensions. Fig. 10 compares the retrieval quality of QLR with dimension selection to QLR without dimension selection. The figure clearly shows, that QLR with dimension selection can perform almost without loss of retrieval quality, even in the presence of 20 highly variant dimensions.

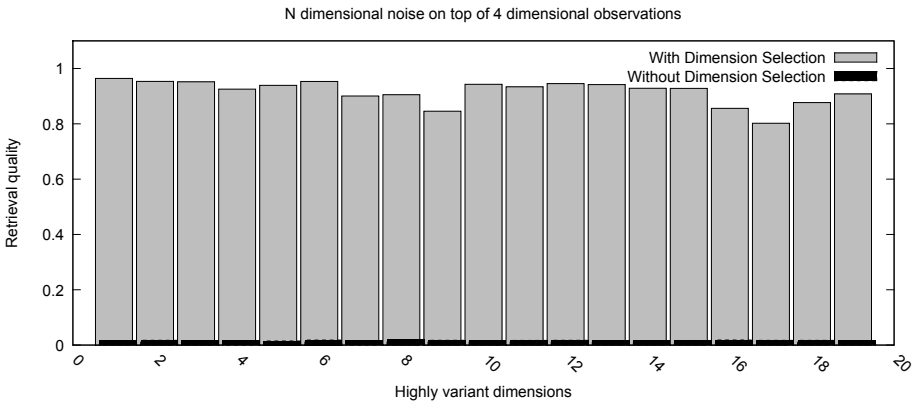

Fig. 10. Querying high dimensional knowledge with and without dimensionality reduction.

6 Related Work

Retrieving information from a distributed system is a well-studied research problem [20, 25, 21, 13, 22, 24, 28, 27]. In this section, we review the related work in a chronological order.

An early bulk of work is centered around the question of how to find and retrieve specific data items or files in a distributed P2P system, given by exact keys, such as hash values. Examples of such systems are Chord [25], CAN [20] and Pastry [21]. While Chord allows only one dimensional keys for the distributed search, routing in CAN is performed based on a multidimensional, euclidean key

space. These systems are, however, only suitable for the retrieval of items that can be identified by an unique index.

The second wave of data retrieval systems identified the need for more complex queries, such as multidimensional range queries (e.g. *retrieve all people with* $age \geq 18$ *and* $income \geq 50k$). Examples include Mercury [3], Squid [23] and Znet [24]. These systems enable clients to retrieve data within a specified range of values of certain attributes. The attribute space has to be partitioned and mapped to the participating peers. Data locality in the multidimensional space can be achieved with dimensionality reduction techniques such as space-filling curves (e.g. [8]). However, range-queries are often too restricted, for instance, if there is sparse or no data in the specified range. For example, a person with $income = 49,99k$ might be interesting for the query issuer, too.

To fill this gap, another branch of research enabled semantic and similarity search in a P2P-environment [22]. Two examples are pSearch [26] and Semantic Small World [13]. Both provide routing mechanisms to find similar data items to a certain multidimensional query. In Semantic Small World, a node maintains links to nodes with similar data items and some long range contacts to distant nodes. This enables efficient routing of arbitrary multidimensional queries to retrieve semantically related data items.

Other approaches [17] [4] [2] provide nearest neighbor search, i.e., finding the k closest data items in a large collection of high dimensional data, where a predefined similarity measure, such as euclidean distance defines *closeness* of data items. These systems aim to find peers with data that is similar to the query, while ignoring the quality and confidence of retrieved information. To the best of our knowledge, there is no approach in the literature enabling scalable, quality-oriented knowledge retrieval in a distributed environment.

7 Conclusion

Today, many devices, such as smart phones, gather an enormous amount of observations. Machine learning algorithms can be used to derive knowledge from these observations. In order to benefit from this enormous amount of distributed knowledge, we need methods to search and retrieve specific knowledge. To accomplish this, we first developed a metric to determine the confidence in the local knowledge of each device. This confidence metric takes the number and variance of observations made by each device into account. Based on this confidence metric, we proposed two methods to route a query for specific knowledge to a device that can answer it with high confidence. To that end, each device maintains routing models that can either be learned from previous queries or maintained actively by exchanging knowledge models. Our evaluations show, that the knowledge retrieved by our approaches is up to 96.7% as accurate as the global optimum.

Acknowledgment

The authors would like to thank the European Union's Seventh Framework Programme for partially funding this research through the ALLOW Ensembles project (project 600792).

References

1. Ashbrook, D., Starner, T.: Using gps to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing 7(5) (2003)
2. Batko, M., Gennaro, C., Zezula, P.: A scalable nearest neighbor search in p2p systems. In: Databases, information systems, and peer-to-peer computing, pp. 79–92. Springer (2005)
3. Bharambe, A.R., Agrawal, M., Seshan, S.: Mercury: supporting scalable multi-attribute range queries. In: ACM SIGCOMM Computer Communication Review. vol. 34, pp. 353–366. ACM (2004)
4. Chen, D., Zhou, J., Le, J.: Reverse nearest neighbor search in peer-to-peer systems. In: Larsen, H., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) Flexible Query Answering Systems, Lecture Notes in Computer Science, vol. 4027, pp. 87–96. Springer Berlin Heidelberg (2006)
5. Chen, X., Yin, W., Tu, P., Zhang, H.: Weighted k-means algorithm based text clustering. In: Proceedings of the 2009 International Symposium on Information Engineering and Electronic Commerce. pp. 51–55. IEEC '09, IEEE Computer Society, Washington, DC, USA (2009), http://dx.doi.org/10.1109/IEEC.2009.17
6. Cheng, C., Cimet, I., Kumar, S.: A protocol to maintain a minimum spanning tree in a dynamic topology. In: Symposium Proceedings on Communications Architectures and Protocols. SIGCOMM '88, ACM (1988)
7. Cisco: Cisco global cloud index: Forecast and methodology, 2013–2018. Online (2014), http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html
8. Ganesan, P., Yang, B., Garcia-Molina, H.: One torus to rule them all: multi-dimensional queries in p2p systems. In: Proc. of the 7th Int. Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004. ACM (2004)
9. Hamerly, G., Elkan, C.: Learning the k in k-means. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16, pp. 281–288. MIT Press (2004)
10. Hernádvölgyi, I.: Generating random vectors from the multivariate normal distribution. Tech. rep., University of Ottawa, Canada, Ottawa, ON (1998)
11. informationisbeautiful.net: World's biggest data breaches. Online (2015), http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/
12. Johnson, S.: Hierarchical clustering schemes. Psychometrika 32(3), 241–254 (1967)
13. Li, M., Lee, W.C., Sivasubramaniam, A.: Semantic small world: An overlay network for peer-to-peer search. In: Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on. pp. 228–238. IEEE (2004)
14. Lindley, D.V.: Introduction to probability and statistics from bayesian viewpoint. part 1 probability, vol. 1. CUP Archive (1965)
15. Lu, Y., Cohen, I., Zhou, X.S., Tian, Q.: Feature selection using principal feature analysis. In: Proc. of the 15th int. conf. on Multimedia. pp. 301–304. ACM (2007)

16. Madden, M.: Privacy and cybersecurity: Key findings from pew research. Online (2015), http://www.pewresearch.org/key-data-points/privacy/
17. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems 45, 61–68 (2014)
18. Montresor, A., Jelasity, M.: Peersim: A scalable p2p simulator. In: Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth Int. Conf. on. pp. 99–100. IEEE (2009)
19. Ogallo, H.G., Jha, M.K., Marroquin, O.: Studying the impacts of vehicular congestion and offering sustainable solutions to city living. Proceedings of the 2nd International Conference on Sustainable Cities, Urban Sustainability and Transportation (SCUST '13) (2013)
20. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. ACM SIGCOMM Computer Communication Review 31, 161–172 (2001)
21. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware. Springer (2001)
22. Sahin, O.D., Emekci, F., Agrawal, D., El Abbadi, A.: Content-based similarity search over peer-to-peer systems. In: Databases, Information Systems, and Peer-to-Peer Computing, pp. 61–78. Springer (2005)
23. Schmidt, C., Parashar, M.: Squid: Enabling search in dht-based systems. Journal of Parallel and Distributed Computing 68(7), 962–975 (2008)
24. Shu, Y., Ooi, B.C., Tan, K.L., Zhou, A.: Supporting multi-dimensional range queries in peer-to-peer systems. In: Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on. pp. 173–180. IEEE (2005)
25. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review 31(4), 149–160 (2001)
26. Tang, C., Xu, Z., Mahalingam, M.: psearch: Information retrieval in structured overlays. ACM SIGCOMM Computer Communication Review 33(1), 89–94 (2003)
27. Tariq, M.A., Koldehofe, B., Bhowmik, S., Rothermel, K.: Pleroma: A sdn-based high performance publish/subscribe middleware. In: Proceedings of the 15th International Middleware Conference. pp. 217–228. Middleware '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2663165.2663338
28. Tariq, M.A., Koldehofe, B., Koch, G.G., Rothermel, K.: Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In: Proceedings of the 6th ACM int. Conf. on Distributed Event-Based Systems. pp. 213–224. ACM (2012)
29. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: ICML. vol. 3, pp. 856–863 (2003)
30. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.Y.: Understanding mobility based on gps data. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 312–321. ACM (2008)
31. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. 33(2), 32–39 (2010)
32. Zheng, Y., Zhang, L., Xie, X., Ma, W.Y.: Mining interesting locations and travel sequences from gps trajectories. In: Proceedings of the 18th international conference on World wide web. pp. 791–800. ACM (2009)
33. Ziegeldorf, J.H., Morchon, O.G., Wehrle, K.: Privacy in the internet of things: threats and challenges. Security and Communication Networks 7(12), 2728–2742 (2014)