Discovering Functional Dependencies in Hidden Data using Approximate Formal Concept

Md. Abdullah Al Mamun

Department of Computer Science & Engineering, Collage of Engineering
Qatar University

***Abstract*—Many users issue queries to a hidden database by using the search template provided by the system. In this article, a new approach based mainly on hidden database reduction preserving functional dependencies is developed. The developed approach is applied to online market systems like OLX. Offline hidden data analysis is used to discover attributes and their domains and different functional dependencies. In this research, a comparative study between several methods for mining functional dependencies shows the advantage of conceptual methods for data reduction. In addition, approximate formal concept analysis is used to discover the functional dependencies on hidden and known datasets. The validation of the proposed designed and developed methods prove their generality and suitability for system interfacing through continuous data reductions.**

***Keywords*—**

I. Introduction

Recently, with the increased use of online systems for different scientific and/or marketing purposes, many organizations are working toward enhancing their web interfaces to facilitate user searches and provide better results. Usually, these search interfaces are correlated to hidden databases or deep web content. One way to enhance these online interfaces is based primarily on hidden data analytics.

In this research, a technique for discovering functional dependencies on hidden dataset using approximate formal concept is developed. Our technique depends on the Formal Concept Analysis theory and data reduction methods.Formal Concept Analysis (FCA) was first introduced as an applied mathematical field derived from concepts and concept hierarchy paradigms [1]. It has been widely used in different fields, like medicine, biology, computer science, electrical, chemical and civil engineering, sociology and linguistics. This mathematical tool has been used efficiently to identify functional dependencies (FDs) [2] from a relational database after transforming the database into a binary relation between the objects set and the attributes set and building a formal context (FC). It was mathematically proven that the attribute implications extracted from the FC are equivalent to functional dependencies in the relational database [3]. However, this transformation leads to a quadratic representation of the dataset compared to the original relation; thus, data reduction methods are used to minimize the number of records obtained in the FC.

In this paper, functional dependencies are extracted at this stage for the knowledge discovery (KD) process of deep web data. Real-time data analysis, mainly using conceptual data reduction, is applied to enhance the interface of the mobile application by providing different consecutive summaries of the current remaining results for a better representation of query results. Also, analyzing uncertain data is a challenging problem and data are rapidly increasing which complicates the analysis process. Data reduction techniques handling uncertainty are highly required. Displaying mostly related dataset to the users is a key concern now a days.

The objectives of this research are as follows

- Conceptually reduce uncertain formatted data without losing dependencies between different attributes with respect to the original dataset
- Reduction method based on Formal Concept Analysis Theory (FCA) are proposed:
- Approximate data reduction without loosing functional dependencies (FD)
- To what extent the reduced dataset is preserving or even improving the functional dependency of a hidden database. This is paper is organized as follows. The introduction discussed about the topic in general.

II. Background and Related Works

Galois connection is a main notion in FCA used to extract implications Crisp Galois Connection operators R and Q are defined as:

$$AR = m|\forall g, g\epsilon A : (g,m)\epsilon I, BQ = g|\forall m, m\epsilon B : (g,m)\epsilon I \tag{1}$$

Where $A \subset G$ and $B \subset M$

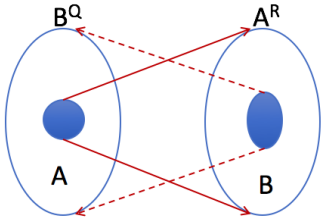

Fig. 1. Galois Connection

This section discusses the background and related work in regards to functional dependencies, algorithms for mining functional dependencies, formal concept analysis, concept lattices, the data reduction method, the Lukasiewicz implication, a deep web or hidden database, online system interfacing and

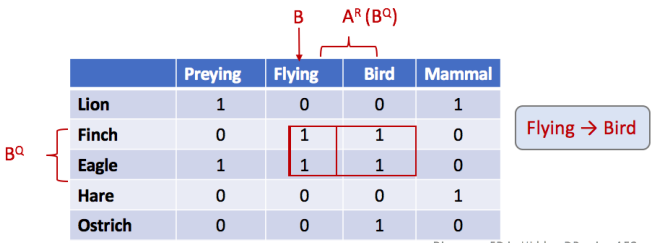

Fig. 2. Example of Galois Connection

interfacing with a small screen design . In a relational database, different types of dependencies exist. Among these different types of dependencies, functional dependencies are the most important and have been widely studied in research [2], [3], [4], [5], [6], [7]. Characterizing functional dependencies from a relational database is an important database topic and has many applications in different fields, such as database management, query optimization, database normalization and reverse engineering. In a given relation, functional dependency is usually used to express the relationship between the attributes. A functional dependency can be expressed as $X \rightarrow Y$, which means that the value of attribute X functionally determines the value of attribute Y and that any two tuples that share attribute X's value also share attribute Y's value.

In the literature, there have been many research studies done in the field of discovering the functional dependencies of a database. Many algorithms have been developed for the goal of extracting functional dependencies from a given set (or sets) of data. Among these algorithms, there were seven that were the most cited and important algorithms [3]. These algorithms were categorized into three main categories, which are lattice traversal algorithms, difference and agree-set algorithms and dependency induction algorithms. In the lattice traversal algorithms, a power set lattice of attribute combinations was built and traversed either by using a bottom-up traversal strategy or by using a depth-first random walk. It produces FD candidates and validates it using a stripped partition approach. TANE, FUN, FD_MINE and DFD algorithms are all lattice traversal algorithms. Two algorithms belong to the category of difference- and agree-set algorithms, which are DEP-MINER and FASTFDs. In this category, difference and agree sets are generated to characterize minimal functional dependency by looking for sets of attributes that have the same values in some tuples. After obtaining the agree sets, valid FDs are extracted from them. The dependency induction algorithm category, such as FDEP, begins by assuming that each attribute is functionally determining other attributes and then by using observation on the data set the FDs were either validated or removed. The TANE algorithm introduced by Huhtala et al. was used for extracting FDs and approximate FDs [8]. The search space in this algorithm is represented as a Hasse diagram of its attributes. Its functionality to detect FDs is based on three main principles to detect. First, it uses partition refinement to determine whether an FD is holding or not. Second, it uses apriori-gen functions to verify that only minimal functional dependencies are discovered. Finally, to reduce the search space of the lattice, pruning rules are used. In the TANE algorithm, the input lattice is divided into levels of attributes, where each level number represents the attribute combination size. Detecting FDs starts from level 1 of the lattice and continues upward, level by level. Each attribute combination in every level is tested for functional dependency. Then, the supersets of the detected FDs are pruned using pruning rules. At the last step of the algorithm, the apriori-gen function tests the attribute combinations that were left unchecked from the previous level.

The FUN algorithm, which is proposed by Novelli and Cichetti [9], is a level-wise algorithm that traverses the attribute input lattice relation level after level bottom-up and uses the partition refinement techniques to characterize functional dependencies. It uses the concept of free sets and non-free sets to validate FD candidates that result in non-minimal FDs. By free sets, we mean, "the sets of attributes that do not include attributes that are functionally dependent on another subset of attributes. Other attributes that are not in the free sets are members of non-free sets. In this algorithm, the attribute sets are validated incrementally, level by level, based on their length, after considering the knowledge acquired from the previous level. For testing to identify if a candidate is a free set or not, a comparison is done between its cardinality and the cardinality of its entire maximal subset. If it is found that it is a free set, then it might be a possible candidate for FD. Otherwise, its entire superset of attributes are discarded because they are not candidates for FDs. The candidate generation mechanism follows the Apriori algorithm [10]. Similar to the TANE and FUN algorithms, in the DFD algorithm [11], a powerset lattice is built, which represents all attribute combinations. In order to characterize FDs, the powerset lattice has to be traversed in depth-first random walk to test valid FDs. This power set lattice is considered to be multiple lattices that are traversed one after anotherusing a decidable path.

The FD_MINE algorithm proposed by Yao et al. is another algorithm belonging to the lattice traversal category that traverses the attribute lattice level-wise bottom up by using stripped partitions and partition intersection techniques to mine the FDs. In addition to the latter techniques, it uses the equivalence classes concept of attribute sets to minimize the number of sets and FDs to be validated. If two attribute sets are functionally dependent on each other, then they are considered equivalent. Each level in the attribute lattice is visited and validated to detect the FDs and the equivalent FDs. Whenever equivalent attributes are discovered, the algorithm prunes only one from the lattice because they are functionally dependent.

The DEP_MINER algorithm was proposed by Lopes et al. Its basic idea works on computing the minimal FDs from the agreed sets of attributes and their inverse difference agreed sets. Agreed sets can be defined as sets of attributes that have the same value in some tuples, and their inverses are the difference agreed sets. Executing the algorithm can be divided into five stages. In the first stage, the algorithm computes the stripped partition for each attribute in a given relation to be used in the second stage for calculating the agreed sets of attributes. Partitions can be defined as sets of equivalence classes that contain tuples that have the same values in given attributes. A stripped partition can be defined as a partition that groups equivalence classes having a size greater than one. In the third stage, the agreed set is converted to maximal sets of attributes, which are sets of attributes with no supersets that have the same value in any two given tuples. Next, in the

fourth stage, complement sets are calculated from the agreed sets. In the last stage, the minimal FDs are extracted using a level-wise search on the complement sets.

The FASTFDs algorithm proposed by Wyss et al. uses the concept of agree sets and difference agree sets of attributes to detect minimal FDs. This algorithm is considered an improvement upon the DEP_MINER algorithm, and it was proven more efficient than DEP_MINER [12]. The FDEP algorithm was proposed by Flash and Savnik [13]. Unlike the previously mentioned algorithms, FDEP uses a special approach to extract the FDs. This approach is based on the comparison of every two tuples in any given relation to find the minimal FDs. There are three mechanisms for implementing the traversal of the FDEP search tree; they are top-down, bidirectional and bottom-up. Among the three versions of the FDEP algorithm, the bottom-up approach was proved to be the best in performance. It consists of two phases, which are calculating the negative cover construction and the negative cover inversion. The negative cover set refers to the set of all non-FDs for a given relation and contains all non-FDs that do not hold in a given relation, while the negative cover inversion refers to the set of all minimal FDs. Thus, the algorithm identifies the set of all FDs after transforming the negative cover into the positive cover of FDs (i.e. it represents the set of all minimal FDs).

Since many databases contain massive number of records and attributes, it is essential to reduce the size of the database either by eliminating unnecessary attributes, removing rows, reducing both database dimensions while preserving some relevant information or keeping a good prediction accuracy. There have been many database reduction methods in the literature. In [14], a technique was developed to exact or approximate the reduction of database attributes. This technique was implemented successfully using Structured Query Language (SQL) in a database. The approximate reduction technique is beneficial for dirty data, while the exact reduction method is used when no dirty data exist. In the exact reduction of [14], all the information is preserved by eliminating only superfluous attributes, which are attributes or sets of attributes that are not required for the correct classification of database objects.

Data reduction methods are used to reduce the size of the database without losing knowledge. By knowledge, we mean the association rules of the database. In the literature, there have been many data reduction methods implemented; however, most of these methods are not efficient, inaccurate and do not fit for fuzzy data [15]. The Lukasiewicz implication, which is based on a fuzzy Galois connection at different precision levels, is one of the efficient reduction methods used for reducing the size of objects or attributes of fuzzy or crisp formal context.

Deep web content represents a huge amount of structured data on the web. By the term deep web or hidden database, we mean the data content that is hidden behind HTML forms [16]. To retrieve data or information from the deep web, which is represented by the expression crawling deep web, the user enters a valid query on the form provided by the organization or institute, and then the deep web returns the result that matches the query.

Crawling deep web content is used in many tasks such as data integration and web indexing [17]. Two methods of accessing deep web content have been studied and implemented in the literature. The idea of the first method is to build for each specific domain a search engine such as books, mobiles, etc. To implement this method, a mediator form was developed for each domain, and then each mediator form is mapped to its data source individually. By mediator form, we mean the form that is provided by any mediator system [18], which is a system that provides the user with access to information from heterogeneous resources. However, implementing this method has several disadvantages, such as the high cost of creating the mediator form and the mapping to its data sources. In addition, matching each input query to its domain is considered a challenging problem. Moreover, defining data on different domains is not easy, since the nature and the boundaries of the data on the web are not clear.

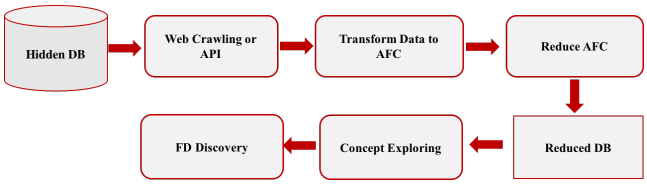

Fig. 3. FD Discovery Steps

Deep web sites or hidden web sites contents can be categorized into two categories: document- oriented textual contents and entity-oriented content. The document-oriented textual content represents popular websites such as Wikipedia, Twitter, etc. The entity-oriented content of deep web can be represented by online shopping websites such as OLX, Amazon, etc. Each deep web category has different crawling techniques or algorithms. The following is a description of an entity-oriented crawl system that is specified to crawl entity-oriented deep web content and use it for advertisement purposes. Each structured entity in such deep web content represents a specified product. The objective behind implementing this system is to get the representative coverage of a specified item for a user. This system consists of the following main components, as illustrated in Diagram 2: URL template generation, query generation and URL generation, empty page filter (or web page filter) and URL extraction and duplication.

III. Methodology

Sampling methods might offer a good alternative for that [19]. In this chapter, we describe an original approach for creating a conceptual sample from data (i.e., reduced database), which can preserve functional dependencies between the different attributes existing in the initial search result. To assess the quality of the reduced database, we select a benchmark of a database of objects, and we check the prediction accuracy of the reduced database. This chapter describes in detail the conceptual approach used in this thesis towards achieving our objectives and goal, which consists of converting a database into an approximate formal context, applying an incremental reduction process and extracting functional dependency or mapping the formal context back into a database instance, as shown below in Fig. 3:

The transformation is done using a pairwise comparison between the tuples of the obtained database. In the exact formal context analysis, a value of 1 is assigned whenever there is an exact match between the same values of an attribute for the compared tuples; otherwise, a value of 0 is given. As exact equality is too restrictive, so we will propose different similarity functions to consider approximate functional dependencies. Data reduction methods play a vital role in FCA due to the large obtained objects in the context, which negatively imposes excessive time and storage needs. The Lukasiewicz reduction is implemented in a new way in this proposed conceptual method, which is incremental on packages of a certain number of FC objects. Thus, this leads us to avoid having an FC with n2 objects compared to the number of database objects and waiting until the entire formal context is build. Another advantage of implementing the Lukasiewicz reduction incrementally is that it enables the dynamic reduction of formal context as long as new objects are available. After that, the Lukasiewicz reduction was further applied iteratively in the obtained reduced FC to further reduce its size in terms of objects and attributes without loss (attribute implications that are equivalent to functional dependencies). In fact, the reduction is implemented in the approximate FC iteratively multiple times for columns and rows until reaching stability, which means that no more columns or rows can be reduced. The precision level used was $\delta = 1$ to guarantee information preservation while reducing rows and attributes.

In this section, we explain how we could extract functional dependencies from the dataset of search results, as we need these to define some order between attributes. As the FDs in the initial dataset are equivalent to the implications in the corresponding FC, we have advantageously used a tool (ConExp) to indirectly find FDs. The functional dependencies were extracted from the approximate reduced formal context. This process is done using the ConExp tool, which is an Open Source Java application used to extract different dependencies that exist between the attributes of the FC. The ConExp tool provides the following functionalities for the user: context editing, building concept lattices for a context, extracting attribute implications and association rules that are applicable in the context and attribute exploration. In our case, we will use the approximate reduced context obtained in the previous step as an input to the tool for extracting attribute implications that are equivalent to the functional dependencies in the original database.

IV. Experimentation

The implemented algorithm has been conducted an extensive experimentation with three dataset. The known datasets are Abalone and ncvoter. The Abalone dataset contains the physical measurements of abalones, which are large, edible sea snails. This dataset has 4177 rows and 9 columns. The columns include 1 categorical predictor (sex) and 7 continuous predictors such as Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight. Tools and language are used are Java, Metanome, ConexExp. Metonome for evaluate the seven algorithm with hidden dataset. Experimentation has also been conducted for the a unknown datasets such as OLX (online shopping site). The experimental steps are as follows: first we need to transform and reduce a dataset to formal approximate formal concept formate using java code.

TABLE I. Number of functional dependency found in 3 datasets

DB	Abalone	ncvoter	OLX
Column	9	19	7
Row	4177	1000	118
25% FC	56	78	33
FD	55	3752	101
50% FC	64	113	42
FD	50	4023	116
75% FC	65	160	57
FD	54	4035	117
100% FC	66	174	59
FD	59	4133	119

Then, using ConExp, we can discover the hidden functional dependency.

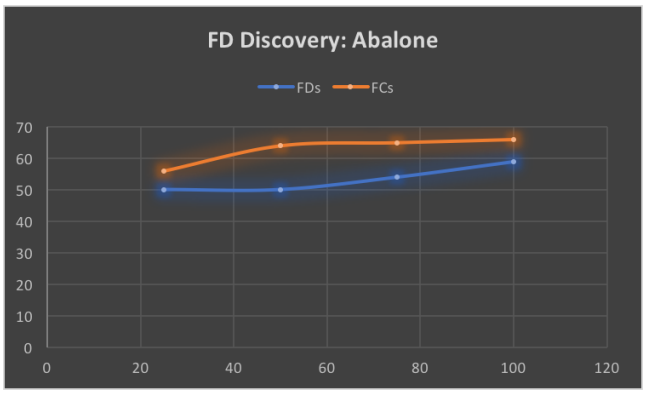

Fig. 4. FD Discovery for Abalone Dataset

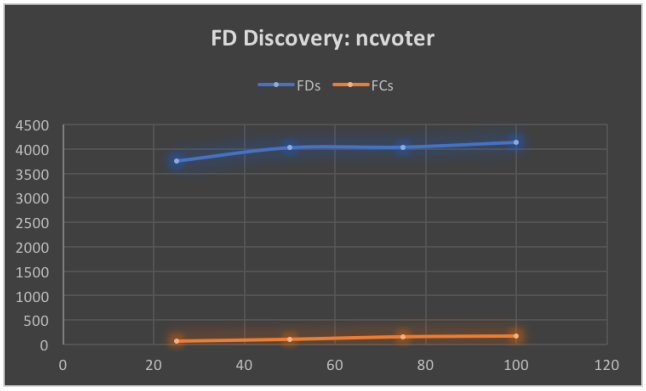

Fig. 5. FD Discovery for ncvoter Dataset

V. Results and Discussions

There are two ways to create a hidden dataset, one is to make web crawling and another is using API. In our experiment, we have collected the hidden data using web crawling from the famous shopping website OLX. Before doing our experiment with a complete hidden dataset, we divided a known dataset into 4 parts as if it is unknown to us. So first run with the hidden dataset has been done with the 25% tuples of the Abalone dataset which has 4177 rows. After reduction, we got only 56 rows of data. Among 56 rows we find 55 functional dependency (FD) which is listed in Table I. The relationship between number of reduced concept and number of functional dependency for the three different datasets are depicted in Fig

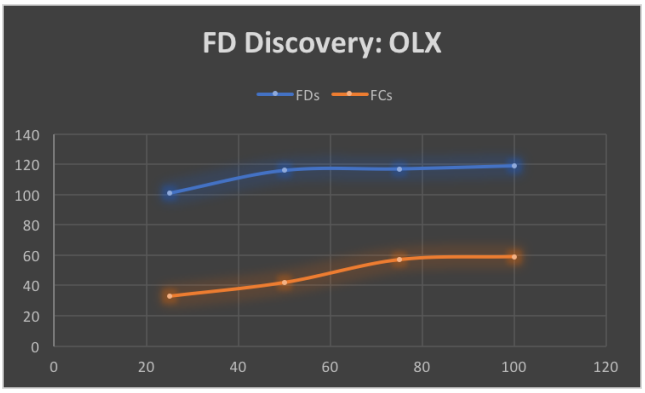

Fig. 6. FD Discovery for OLX Dataset

4 to Fig. 6 respectively. For the Abalone dataset, the algorithm found 55 functional dependencies among 25% of total tuples, when the number of objects is increased to double, the number of functional dependencies is still 50 shown in Fig 4 where the x-axis indicates the reduces number of formal concepts and y-axis indicates the number of functional dependencies are found. However, this total has a little increment when the tuples are 75% and 100% respectively. The increment is not noticeable. The same pattern is also found for the other two datasets such as ncvoter and OLX. Thus, we can conclude that the number of functional dependencies can be settled after a certain level of objects and we can find approximately all functional dependencies on even half of the objects. So, this experiment and empirical results shows that discovering functional dependency is possible even if the datasets are hidden.

VI. Conclusion and Future Work

In sum, we can say that the proposed AFC preserved the functional dependency. Formal concept can be reduced without losing any information compared to the primary stage of the database. Finally, discovering Functional dependency is possible even if database is hidden.

References

[1] T. PARVEEN and H. D. ARORA, "Congestion pricing, motorcycle, marginal-health cost, generalized cost, emme-2." *Walailak Journal of Science & Technology*, vol. 15, no. 1, 2018.

[2] K. T. J. Rancz and V. Varga, "A method for mining functional dependencies in relational database design using fca," *Studia Universitatis Babes-Bolyai Cluj-Napoca, Informatica*, vol. 53, no. 1, pp. 17–28, 2008.

[3] J. Baixeries, M. Kaytoue, and A. Napoli, "Characterizing functional dependencies in formal concept analysis with pattern structures," *Annals of mathematics and artificial intelligence*, vol. 72, pp. 129–149, 2014.

[4] H. Mannila and K.-J. Räihä, "Algorithms for inferring functional dependencies from relations," *Data & Knowledge Engineering*, vol. 12, no. 1, pp. 83–99, 1994.

[5] H. Yao and H. J. Hamilton, "Mining functional dependencies from data," *Data Mining and Knowledge Discovery*, vol. 16, no. 2, pp. 197–219, 2008.

[6] A. Silberschatz, H. F. Korth, S. Sudarshan *et al.*, *Database system concepts*. McGraw-Hill New York, 1997, vol. 4.

[7] S. Lopes, J.-M. Petit, and L. Lakhal, "Efficient discovery of functional dependencies and armstrong relations," in *International Conference on Extending Database Technology*. Springer, 2000, pp. 350–364.

[8] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann, "Functional dependency discovery: An experimental evaluation of seven algorithms," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1082–1093, 2015.

[9] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *The computer journal*, vol. 42, no. 2, pp. 100–111, 1999.

[10] N. Novelli and R. Cicchetti, "Fun: An efficient algorithm for mining functional and embedded dependencies," in *ICDT*, vol. 1. Springer, 2001, pp. 189–203.

[11] Y. Ye and C.-C. Chiang, "A parallel apriori algorithm for frequent itemsets mining," in *Software Engineering Research, Management and Applications, 2006. Fourth International Conference on*. IEEE, 2006, pp. 87–94.

[12] Z. Abedjan, P. Schulze, and F. Naumann, "Dfd: Efficient functional dependency discovery," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 949–958.

[13] C. Wyss, C. Giannella, and E. Robertson, "Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2001, pp. 101–110.

[14] A. Kumar, "New techniques for data reduction in a database system for knowledge discovery applications," *Journal of Intelligent Information Systems*, vol. 10, no. 1, pp. 31–48, 1998.

[15] S. Elloumi, J. Jaam, A. Hasnah, A. Jaoua, and I. Nafkha, "A multi-level conceptual data reduction approach based on the lukasiewicz implication," *Information Sciences*, vol. 163, no. 4, pp. 253–262, 2004.

[16] J. Madhavan, D. Ko, Ł. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy, "Google's deep web crawl," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1241–1252, 2008.

[17] Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, "Crawling deep web entity pages," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 355–364.

[18] R. Yerneni, C. Li, H. Garcia-Molina, and J. Ullman, "Computing capabilities of mediators," in *ACM Sigmod Record*, vol. 28, no. 2. ACM, 1999, pp. 443–454.

[19] S. T. Buckland, E. A. Rexstad, T. A. Marques, and C. S. Oedekoven, *Distance sampling: methods and applications*. Springer, 2015.