Automatic Query Generation and Query Optimization for Declarative Crowdsourcing Systems

Miss. Priyanka D Sarode [1]
[1] PG Student,Computer Engineering, SND COE & RC,Yeola, Maharashtra, India

ABSTRACT

Query optimization is the way to mitigate the burden to deal with crowd and hide complexities in query. User submits the SQL query and system is then responsible to compile it to generate query execution plan and estimate in the crowdsourcing market. As per query plan there is difference in crowdsourcing cost between the best and the worst plans may be multiple orders of significance. Query optimization is important to crowdsourcing system as it provides declarative query interface in relational database management system. We have proposed a cost based query optimization approach known as, CROWDOP. It deals with the cost and latency in query optimization and make query plans which gives efficient balance between latency (time) and cost. Proposed approach optimizes three types of queries such as, select, join and complex select-join queries. As a part of contribution complex query plans are executed as well as CROWDOP is extended for more advanced SQL operators such as, aggregation and sorting. With the experimental results we have proven efficiency of system.

Keywords*: HITs, crowd sourcing, Query Optimization, user queries, query execution plans.*

1. INTRODUCTION-1

For frequent relational database management systems query optimization plays very important role. Query optimizer regulates most possible ways to evaluate query in active way. It considered most possible ways. Query optimizer cannot be getting to the user practically as queries are submitted to the database server [3]. That are parsed by parser and moved towards the query optimizer for optimization. There are some database engines available that guides to query optimizer by providing some hints. Queries are evaluated in such way they return requested information by accessing relative database. In most of databases due to complicated data structures as well as for complicated data structures required queries are bringing from different data structures in different ways by different order (e.g. Deco's data model)[9]. Therefore, each different processing time is required for different ways. Query processing time is depending upon the way that is selected for processing query. Queries having same processing time may have high variance corresponding to the selected processing way. Query optimization is an automated process for finding a way for processing a query submitted by user in very less time. Therefore our proposed system helps to analyze the query, optimize it, find query execution plans and finally predict potential query plan for execution over crowd sourced data [1]. Also to avoid this ambiguity in query execution there must be proper execution plan and query optimizer. It helps to evaluate the ways of OR execution plans from time and cost point of view and finally system should preferred possibly good plan for execution [2]. Our proposed approach should consider the cost based query optimization approach for informative crowd sourcing system. The main objective of the proposed approach is to manage cost and latency in query optimization. Practically, the query optimizer cannot be getting directly by users when queries are surrender to database server, and then that will be parsed by the parser. Furthermore they are passes to the query optimizer where optimization takes place. However, some database engines grant guiding the query optimizer with hints. Queries results are produced by accessing relative database data and evaluating it in a way that return the requested information. Each different way commonly requires different processing time. Processing times of the same query may have high variance, a second to minutes, hours, depending on the way selected. The plan of query optimization is a computerized process, is to search the way to process a user query in small amount of time.

Our proposed approach works on crowd sourcing concept where user analytical work is delegated to the system and system executes the effective query plan in crowd sourcing marketplace. In our system effective usage of query operators is achieved. Also for working environment dataset is used in which system need to evaluate the optimization approaches under many queries with varying numbers of conditions, domain sizes, etc. As a part of contribution complex query plans are executed as well as CROWDOP is extended for more advanced SQL operators such as, aggregation and sorting. With the experimental results we have proven efficiency of system.

2. RELATED WORK-2

J. Fan, M. Zhang,et al.[1], discussed about query optimization plan for automated process. It is to search a given query in minimum time. Generally, query optimization compares many common-sense options, to provide a "Correct" plan in a reasonable time which usually does not different from the best possible result.To abuse human intelligence CROWDSOURCING demanding for developing an interesting powerful tool to resolve hazards that computers cannot do well.

S. B. Davidson, T. Milo et al [2], demonstrates the difficulties to inspect maximum/top-k and group-by queries using the crowd. They have proposed a Bayesian model to represent clustering approach. In this system, for query optimization 'Ranking based' and value-based error model is considered.

J. Fan, M. Lu, B. C. Ooi, W.-C. Tan [3], proposed concept-based approach & Hybrid machine-crowd sourcing structure
It used to address difficulties in crowd souring in web table matching. Column in each table is well-developed with concept-based approach. The proposed Hybrid machine-crowd sourcing structure approach breaks human intelligence for different columns in web table. This system provides simplification for crowd to assume required answers.

M. J. Franklin, et al. [4], focused on basic building blocks, as well as an algorithm to filter a set of data items. The term filtering is used for each of the properties author wish to check This system "Image shows a scientist," and "Image of people in which people looking towards the camera." Filters are used. As effective solutions for filtering strategy a heuristic algorithm is proposed.

X. Liu, et al. [5], suggested two types of models, known as prediction model, it is also known as "Economic Model" in AMT, Voting-based Prediction and Verification model also called as, Probability-based Verification, Online Processing. These models are used to show high-quality answers while keeping the total cost low. The natural expertise of human workers to perform complex tasks that are very challenging for computers is permitted by Crowdsourcing techniques.

A. Marcus, et al [6], discovered coordinated attacks from multiple workers. Accuracy of system is depends on count-based approach. Therefore, they mainly focused on search text-based counts as well as they searched that the label-based approach has better accuracy.

A. Marcus, et al. [7], compare items for sorting and joining data, two of the most common operations in DBMSs. MTurk platform is used Qurk, runs on top of crowdsourcing.

A. G. Parameswaran et al, [8], suggested, Deco's data model, query language prototype. Crowdsourcing and Databases Crowdsourcing Algorithms are used for practical as well as principled approach. It is used for accessing crowd data and also integrating it with conventional data.

H. Park and J. Widom[9], proposed, Deco's cost-based query optimizer, building on Deco's data model, query language, and query execution engine. Main objective of Deco's is to search best query plan to answer a query. It represents Deco's cost-based query optimizer. The Primary goal Deco's is to find the best query plan to answer a query.

V. C. Raykar, et al [10], discovered the probabilistic approach for supervised learning. It is used to evaluate different experts and also gives an estimate of the actual hidden labels. Result of proposed method indicates that, it is superior to the commonly used majority voting baseline. In this two key assumptions are defined as, 1. Performance of each annotator does not depend on the feature vector for a given instance and 2. Conditional on the truth the experts are independent, that is, they make their errors independently.

A. D. Sharma, A. Parameswaran, H. Garcia-Molina, and A. Halevy [11], discussed the problem of CROWDFIND using the metrics of cost and time, and design optimal algorithms that span the skyline of cost and time, Authors studied the fundamental CROWDFIND of problem, relevant in many crowdsourcing applications. They build an approach that lies on the skyline of cost and latency for two settings: when humans answer correctly, and when they may make errors. Also for simplification of assumption that all workers are equally capable, identifying spam workers and learning accuracies of workers over time while solving CROWDFIND problems are also interesting extensions.

P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis[12] implements a methods for accessing maximum item from a set in a crowdsourcing environment. They build parameterized families of algorithms to access maximum item and proposed strategies to tune these algorithms under various human error and cost models. This system utilizes Plurality Rule, Bubble Max Algorithms and Tournament Max Algorithms. Implemenatation of this system is based on Human Error Model.

J. Wang, T. Kraska, M. J. Franklin, and J. Feng [13], studied the problem of crowdsourcing entity resolution. Authors described how machine-only approaches often fall short on quality, while brute force people only approaches are too slow and expensive. Thus, they proposed a hybrid human-machine workflow to address this problem. In the context of this hybrid approach, In particular, the results indicated that, two-tiered approach generated fewer cluster-based HITs than existing algorithms, Hybrid human-machine workflow significantly reduced the number of HITs compared to human-based techniques, and achieved higher quality than the state-of-the-art machine based techniques and cluster-based HITs can provide lower latency than a pair-based approach.

J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng,[14],uses a technique for query processing on dirty data such as,

1. Sampling Error
2. Data Error
3. SampleClean Framework

They proposed SampleClean, a novel framework that requires users to clean a sample of data, and utilizes the cleaned sample to obtain unbiased query results with confidence intervals.

They also identify three types of data errors that may affect query results, and develop Normalized SC and Raw SC to estimate query results for the data with these errors.

S. E. Whang, P. Lofgren, and H. Garcia-Molina [15], determines the problem of enhancing Entity Resolution (ER) with the help of crowdsourcing. Algorithm: Brute-force" algorithm: For deriving the best question that has the highest expected accuracy. GCER algorithm to produce an approximate result within polynomial time, Half algorithm.

3. PROBLEM DEFINITION-3

Recently crowd system is user friendly as user gets the results on firing query according to his/her expectations. Respective system analyze the query, generate the execution plan, executes it, get results, resolve errors. While this query may execute in various ways and its effective execution time will vary according to given query execution plan. This ambiguity is the congestion problem for system. These query execution ways must be analyzed as well as effective query plans should be executed only. For the sake of effective query execution from time and cost point of view proper strategy should be defined. In crowd sourcing environment to hide query execution complexity and to encapsulate the execution phases there must be system that executes the user query with effective execution plans. System should recognize the best query execution plans using proposed algorithm in optimizer from cost and execution time point of view. Also operator involve in query statement should have priority of execution else time of execution may vary and this is one of the critical point of crisis. Hence there must be a need of such system that works on these issues.

4. PROPOSED SYSTEM-4

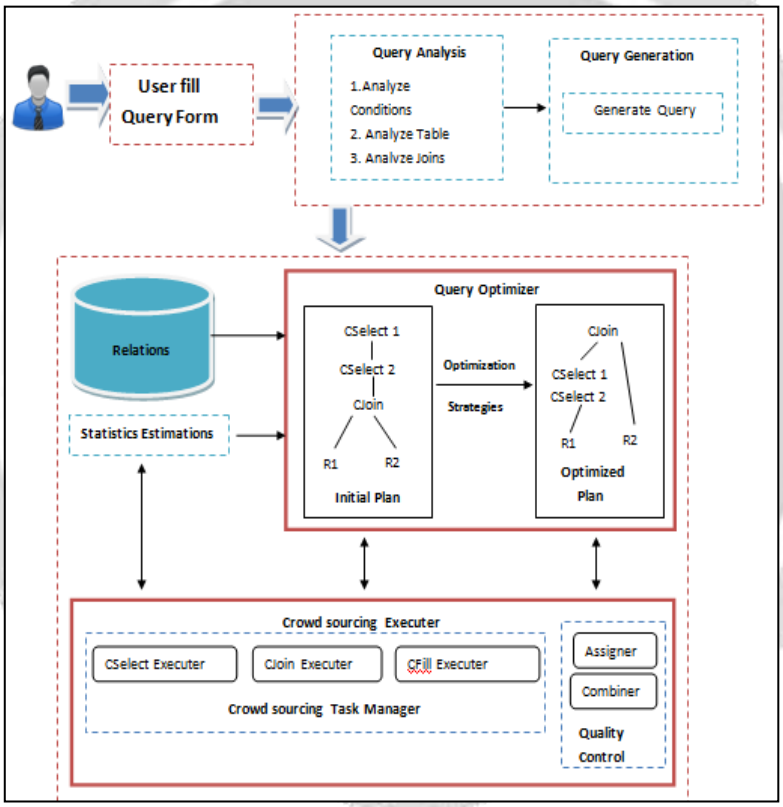


Fig-1: Proposed System Architecture

1. Query optimizer
a) CSELECT (Crowd-Powered Selection)
b) CJOIN (Crowd-Powered Join)
c) CFILL (Crowd-Powered Fill)
2. Crowdsourcing executor

a) CSELECT (Crowd-Powered Selection): A CSELECT operator abstracts the human operation of selecting objects satisfying certain conditions. Formally, the input of a CSELECT operator oS consists of a set T of tuples and a collection C of selection conditions, and the output is a subset T.
b) CJOIN (Crowd-Powered Join): A CJOIN operator leverages the crowd to combine objects from two sources according to certain constraints. Formally, the input of a CJOIN operator oJ consists of two tuple sets T1 and T2 as well as a collection C of join conditions, and the output is a set t1; t2.

c) CFILL (Crowd-Powered Fill): A CFILL operator crowd sources the tuple attributes that are unknown to machines but can be identifed by humans. Formally, the input of a CFILL operator oF is a collection of pairs of tuple sets and attributes T1;A1; T2;A2; : : :, and the output is a collection of tuple sets T1; T2; : : :.
2. Crowd sourcing executor: CROWDSOURCING EXECUTOR iteratively executes the operators by a bottom-up traversal of the plan tree.

5. ALGORITHMS-5

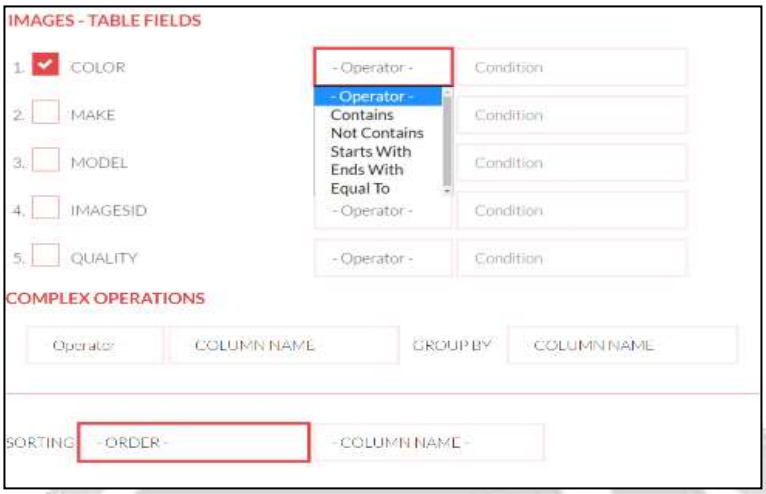
Algorithm 1: OPT FRAMEWORK
Input: Q: A query;
C: A cost budget;
Output: P: An optimized query plan Process:
Process:
1. If c = null
P* = cost Opt (Q)
Else
2. Calculate: Lmin = COMPUTEMINLATENCY (Q)
3. Calculate: Lmax = COMPUTEMAXLATENCY (Q)
4. While (Lmin = Lmax) do
L = (Lmin+ Lmax)/2
P= LATENCYBOUNDOPT (Q; L)
IF P: cost is less than C then
P* = P
Lmax =L-1
Else
Lmin = L+1
End If
End while
5. return P*

Algorithm 2: LATENCY BOUND OPT
This algorithm is used to calculate latency bound for each query plan and find out the optimal query plan.
Input: Os =: Select segments;
Oj: Join segments;
R: Relations;
L: Latency constraint
Output: P: An optimized query plan
Processing:
Step 1: for each Qs E Qs do
for L = 1 : : : min {L }
Calculate: PS =OPTSELECT (QS: C; L)
SAVEPLAN ({QS: R}; PS)
Step 2: for k = 1:: mod QJ do
{⟨Ri;Rj⟩}-- FINDCANDIDATES(R; QJ; k)
Step 3: for each ⟨Ri;Rj⟩ join do
Find plan: Pi -- FINDPLAN (Ri:L − l)
Find plan: Pj --FINDPLAN (RJ; L − l)
Find Optimal Strategy: PJ ← OPTJOIN (Pi:T ; Pj :T ;QJ; l)
Calculate cost: costPJ-- PJ13: cost + Pi:cost + Pj :cost
Step4: Calculate Minimum Cost: PJ --argPJ min costPJ
Pd: GENPLAN (P; Pi; Pj)
P: SAVEPLAN (Ri U Rj; Pd)
return P

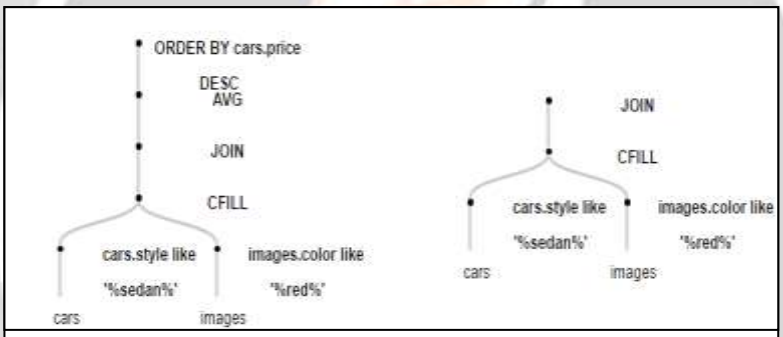
6. EXPERIMENTAL RESULTS-6

The system is developed in java using jdk1.7. For database we used mysql5.3. For testing synthetic data are generated using existing vehicle dataset [16] and a java program. UCI Automobile Dataset: Auto [16] is used for system testing. This is VEHICLE Details dataset contains specification of 205 cars. We have collected images from [17] and create mapping of image with dataset using color and quality attribute. To create images dataset, each tuple in vehicle dataset is duplicated 20 times with random values of car image, color and quality attribute. To create review dataset, each tuple in vehicle dataset is duplicated 10

times to generate review table. We have written a java program to randomly assign reviews to the vehicle and map image color quality attribute with respect to the vehicle entry in dataset.
In the system design query generation panel is provided to the user using which user can generate a query with operations like select, select with condition, join operation, complex aggregate functions and sorting.
Following screenshot displays the query generation panel.



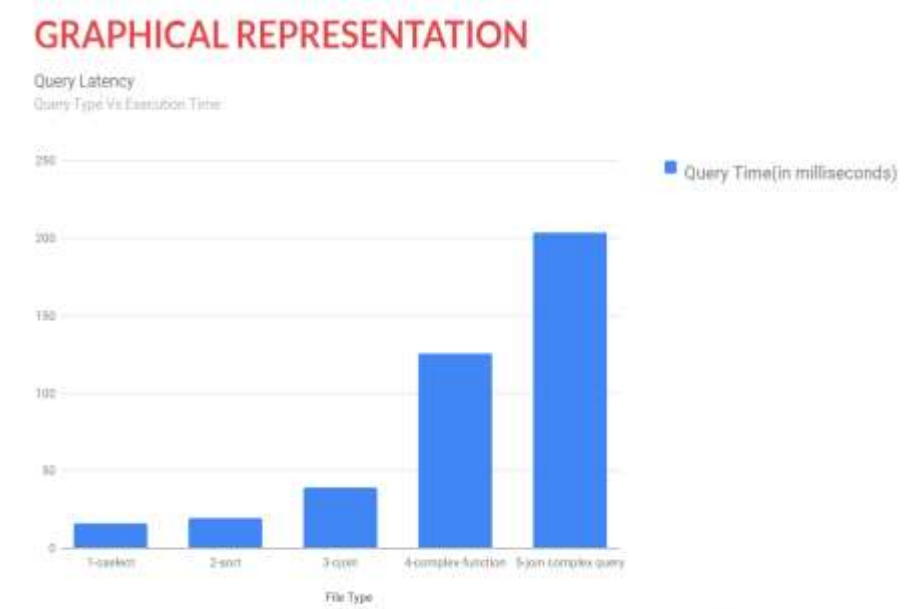
Efficient Query execution plan is generated using cost and latency evaluation. Following fig shows the partition tree for join query and complex query.



Query1: Select images.color, cars.style, cars.make, images.make, cars.model, images.model from cars INNER JOIN images on cars.make = images.make and cars.model = images.model WHERE images.color like '%red%' and cars.style like '%sedan%'

Query2: Select images.color, cars.style, cars.make, images.make, cars.model, images.model, AVG(cars.price) From cars INNER JOIN images on cars.make = images.make and cars.model = images.model WHERE images.color like '%red%' and cars.style like '%sedan%' group by cars.make ORDER BY cars.make DESC

Latency of query is evaluated based on execution time required for different types of queries. For testing, randomly 20 queries are generated of each type and evaluated average time of execution for each type of query. Following graph shows the latency of query with respect to various query type. The result shows that c-join has highest latency as compared to other operations.



For cost comparison we have used the price function:
S = b+wx. Where b is the base charge and w is incremental charge set to $0:005 and x is selectivity cost for single attribute.
X can be calculated as:

$$X=|R|.\Pi^{i-1}_{j=1}\, s\,(o^{s}_{j}) \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where |R| represents the number of conditions and o^{s}_{j} represents conditions selectivity cost in order of execution.
Following table shows the cost evaluation for multiple select conditions and selection with sort operations. Sort operation requires extra time that regular select query.
Generally sorting function requires n(logn) hits for evaluation.
Hence sorting cost can be calculated as:
Sorting cost = S+n(logn)
Where, n represents total number of records with Condition S.
For aggregate function with group by clause groups the distinct values together and hence based on number of HIT cost can be evaluated as :
Aggregate function cost: S+n(log(Dn)
Where, Dn represents distinct records.

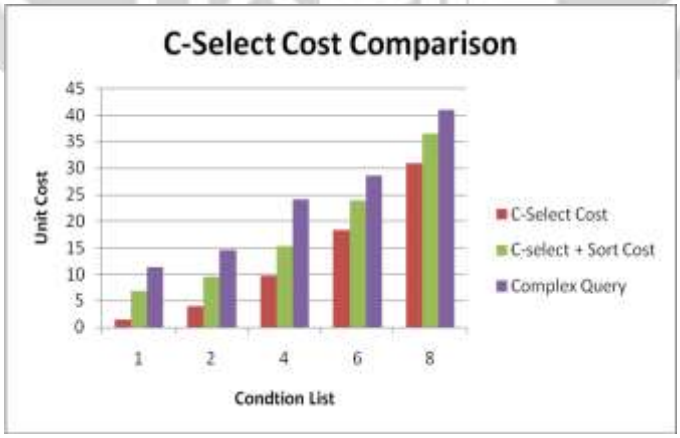


Chart-1: Graph of cost of comparison

Following graph represents the cost evaluation for Fill condition with select query, query with sort and query containing aggregate and sort functions.
The query cost is evaluated as : Q cost = Select cost + fill cost + join cost + Sort Cost + aggregate Cost

Chart-2: Graph of query execution cost comparison

7. CONCLUSION

The proposed system works on crowd sourcing concept. In this system user analytical work is delegated to the system and system executes the effective query plan in crowd sourcing marketplace. This system achieves an effective usage of query operators. Also for working environment dataset is used in which system need to evaluate the optimization approaches under many queries with varying numbers of conditions, domain sizes, etc. As a part of contribution complex query plans are executed as well as CROWDOP is extended for more advanced SQL operators such as, aggregation and sorting. With the experimental results i have proven efficiency of system. The systems analyzes the multiple way or approaches that help to resolve the user queries in crowd a sourcing system which supports cost based query optimization, multiple crowd sourcing operators and allow tradeoff in between cost from monetary point of view and latency.

8.ACKNOWLEDGMENT

I am amazingly glad to represent my sentiments of gratitude to all who rendered their valuable guidance to us. I would like to express my appreciation and also thankful to Prof. I. R. Shaikh, Head of Department, Computer Engineering, S.N.D. College of Engineering and Research Center, Yeola. I also thank the anonymous reviewers for their comments.

9. REFERENCES

[1] Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi, CrowdOp:Query Optimization for Declarative Crowdsourcing Systems
[2] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, Using the crowd for top-k and group-by queries, in Proc. 16th Int. Conf. Database Theory, 2013, pp. 225236.
[3] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang, A hybrid machinecrowdsourcing system for matching web tables, in Proc.IEEE 30th Int. Conf. Data Eng., 2014, pp. 976987.
[4] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, CrowdDB: Answering queries with crowdsourcing, in Proc. ACM SIGMOD Int. Conf. Manage Data, 2011, pp. 6172.
[5] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, CDAS: A crowdsourcing data analytics system, Proc. VLDB Endowment, vol. 5, no. 10, pp. 10401051, 2012.
[6] A. Marcus, D. R. Karger, S. Madden, R. Miller, and S. Oh, Counting with the crowd, Proc. VLDB Endowment, vol. 6, no. 2, pp. 109120, 2012.
[7] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, Human-powered sorts and joins, Proc. VLDB Endowment, vol. 5, no. 1, pp. 1324, 2011.
[8] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom, Deco:Declarative crowdsourcing, in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage., 2012.
[9] H. Park and J. Widom, Query optimization over crowdsourced data, Proc. VLDB Endowment, vol. 6, no. 10, pp. 781792, 2013
[10] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, Learning from crowds, J. Mach. Learn. Res., vol. 11, pp. 12971322, 2010.
[11] A. D. Sharma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, Crowd-powered find algorithms, in Proc. IEEE 30th Int. Conf. 2014.
[12] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis, Max algorithms in crowdsourcing environments, in Proc. 21st Int. Conf. World Wide Web, 2012.
[13] OJ. Wang, T. Kraska, M. J. Franklin, and J. Feng, Crowder: Crowdsourcing entity resolution, Proc. VLDB Endowment, vol. 5, no. 11, pp. 14831494, 2012.

[14] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, Leveraging transitive relations for crowdsourced joins, in Proc. SIGMOD Int. Conf. Manage. Data, 2013.
[15] S. E. Whang, P. Lofgren, and H. Garcia-Molina, Question selection for crowd entity resolution, Proc. VLDB Endowment, vol. 6,no. 6, pp. 349360, 2013.SN
[16] https://archive.ics.uci.edu/ml/datasets/Automobile
[17] http://ai.stanford.edu/~jkrause/cars/car_dataset.html