Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs

Jeremy Chen
University of Waterloo
jeremy.chen@uwaterloo.ca

Yuqing Huang
University of Waterloo
y558huan@uwaterloo.ca

Mushi Wang
University of Waterloo
m358wang@uwaterloo.ca

Semih Salihoglu
University of Waterloo
semih.salihoglu@uwaterloo.ca

Ken Salem
University of Waterloo
ken.salem@uwaterloo.ca

ABSTRACT

This paper is an experimental and analytical study of two classes of summary-based cardinality estimators that use statistics about input relations and small-size joins in the context of graph database management systems: (i) optimistic estimators that make uniformity and conditional independence assumptions; and (ii) the recent pessimistic estimators that use information theoretic linear programs (LPs). We begin by analyzing how optimistic estimators use pre-computed statistics to generate cardinality estimates. We show these estimators can be modeled as picking bottom-to-top paths in a *cardinality estimation graph* (CEG), which contains subqueries as nodes and edges whose weights are average degree statistics. We show that existing optimistic estimators have either undefined or fixed choices for picking CEG paths as their estimates and ignore alternative choices. Instead, we outline a space of optimistic estimators to make an estimate on CEGs, which subsumes existing estimators. We show, using an extensive empirical analysis, that effective paths depend on the structure of the queries. While on acyclic queries and queries with small-size cycles, using the maximum-weight path is effective to address the well known underestimation problem, on queries with larger cycles these estimates tend to overestimate, which can be addressed by using minimum weight paths. We next show that optimistic estimators and seemingly disparate LP-based pessimistic estimators are in fact connected. Specifically, we show that CEGs can also model some recent pessimistic estimators. This connection allows us to adopt an optimization from pessimistic estimators to optimistic ones, and provide insights into the pessimistic estimators, such as showing that they have combinatorial solutions.

PVLDB Reference Format:
Jeremy Chen, Yuqing Huang, Mushi Wang, Semih Salihoglu, and Ken Salem. Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs . PVLDB, 15(8): 1533-1545, 2022.
doi:10.14778/3529337.3529339

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/cetechreport/CEExperiments.

Proceedings of the VLDB Endowment, Vol. 15, No. 8 ISSN 2150-8097.
doi:10.14778/3529337.3529339

1 INTRODUCTION

The problem of estimating the output size of a natural multi-join query (henceforth *join query*), is a fundamental problem that is solved in the query optimizers of database management systems to generate efficient query plans. This problem arises both in relational systems as well as those that manage graph-structured data where systems need to estimate the cardinalities of subgraphs in their input graphs. It is well known that both problems are equivalent, since subgraph queries can equivalently be written as join queries over binary relations that store the edges of a graph.

We focus on the prevalent technique used by existing systems of using statistics about the base relations or outputs of small-size joins to estimate cardinalities of joins. These techniques use these statistics in algebraic formulas that make independence and uniformity assumptions to generate estimates for queries [2, 23, 25, 28]. Cardinality estimation techniques that make such independence and uniformity assumptions are appealing because they are simple and fast, and they are widely used in practice in systems that adopt a graph-based model, where the set of small-size joins can be known in advance. We refer to these as *summary-based optimistic estimators* (optimistic estimators for short), as these estimators can make both under and overestimations. These techniques contrasts with the recent *pessimistic estimators* that are based on worst-case optimal join size bounds [1, 5, 6, 13, 17], which are based on information theoretic linear programs (LPs), and avoid underestimation, albeit using very loose estimates [31].

This paper makes three main contributions. First, we show existing optimistic estimators can be described in a common framework, which we call *cardinality estimation graphs*, or CEGs. A CEG describes a space of alternative ways in which such estimators can combine the available statistics to generate cardinality estimates. This space subsumes the choices made by existing summary-based estimators. Second, we connect the seemingly disparate pessimistic estimators with optimistic estimators and show that they can also be modeled as picking paths in a CEG (one with different edge weights than the CEG used by optimistic estimators). Third, we conduct an extensive empirical evaluation of estimators in the CEG-space we define for optimistic estimators to characterize the best way to make estimates for different types of queries and compare these estimators with pessimistic ones.

The main observation that motivates our first contribution is that in prior optimistic estimators, there is often more than one algebraic formula that can be used to make an estimate for a query, and no clear answer for which one to use. For example, consider the

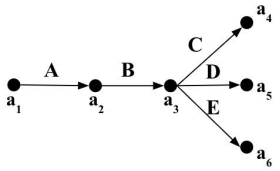
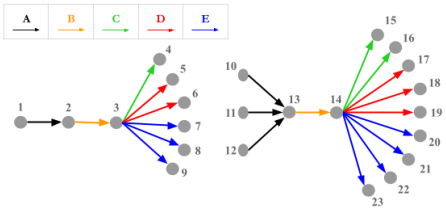


Figure 1: Example subgraph query Q_{5f}.



A.s	A.d
1	2
...	...
12	13

B.s	B.d
2	3
13	14

C.s	C.d
3	4
14	15
14	16

D.s	D.d
3	5
...	...
14	19

E.s	E.d
3	7
...	...
14	23

Figure 2: Example dataset in graph and relational formats.

query in Figure 1. Given the accurate cardinalities of all subqueries of size ≤ 2, there are 252 formulas to make an estimate. Examples of these formulas are:

- $|\overrightarrow{A}\,\overrightarrow{B}| \times \frac{|\overrightarrow{B}\,\overrightarrow{C}|}{|\overrightarrow{B}|} \times \frac{|\overleftarrow{C}\,\overrightarrow{D}|}{|\overrightarrow{C}|} \times \frac{|\overleftarrow{D}\,\overrightarrow{E}|}{|\overrightarrow{D}|}$
- $|\overrightarrow{A}\,\overrightarrow{B}| \times \frac{|\overrightarrow{B}\,\overrightarrow{D}|}{|\overrightarrow{B}|} \times \frac{|\overleftarrow{C}\,\overrightarrow{D}|}{|\overrightarrow{D}|} \times \frac{|\overrightarrow{B}\,\overrightarrow{E}|}{|\overrightarrow{B}|}$

In previous work [2, 23, 25], the choice of which of these estimates to use has either been unspecified or fixed without acknowledging possible other choices. Our paper aims to answer this question by systematically studying these alternative estimates and empirically justifying which estimates are more accurate.

We begin by showing that the algebraic formulas of prior optimistic estimators can be modeled as picking a bottom-to-top path in a weighted CEG, which we call CEG_O, for **O**ptimistic. In this CEG nodes are intermediate sub-queries and edges weights are average degree statistics that extend sub-queries to larger queries. For example, the CEG_O for the query in Figure 1 and the input dataset in Figure 2 is shown in Figure 3. Each path of this CEG corresponds to a possible algebraic formula and the corresponding estimate is the multiplication of the weights of the edges in the path. We first systematically describe a space of estimators, defined by different choices to pick a CEG path, i.e., an algebraic formula, for making an optimistic estimate. This space subsumes and extends the choices made by existing optimistic estimators.

As our second main contribution, we show that CEGs are expressive enough to model also the recent LP-based pessimistic estimators. Specifically, we show that we can replace the edge weights of CEG_O (which are average degrees) with maximum degrees of base relations and small-size joins, and construct a new CEG, which we call CEG_M. Unlike the optimistic estimators, where the choice of path is not clear, we now show that picking the minimum weight path would (provably) be the most accurate estimate and this path is indeed equivalent to solution of the LP that defines a pessimistic estimator from reference [17] called MOLP. We therefore show that

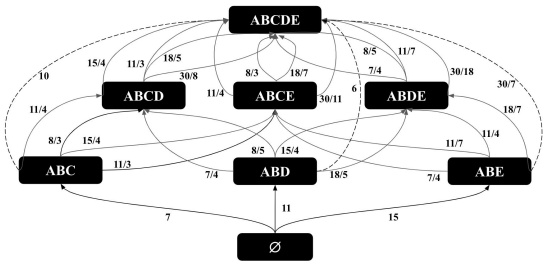


Figure 3: CEG_O for query Q_{5f} in Figure 1 when the Markov table (§4) contains joins up to size 3.

both subgraph summary-based optimistic estimators and the recent LP-based pessimistic ones, which were not known to be related, can be seen as different instances of a broader class of estimators that pick paths through CEGs. Our ability to model these two classes of estimators from literature in a common framework allows us to apply an optimization called the *bound sketch* optimization designed for the recent pessimistic estimators from reference [6] also to optimistic estimators.

As our third contribution we show empirically that the better performing optimistic estimators in the CEG space we define depend on the structure of the query. We show that on acyclic queries and queries with small-size cycles, using the *maximum-weight paths*, which correspond to choosing the highest estimating formulas, is an effective way to make accurate estimations. This is because as in the relational setting, estimators that use independence assumptions tend to underestimate the true cardinalities of queries, and picking the highest estimating formula can offset these underestimations. In contrast, we observe that on queries that contain larger cycles, optimistic estimators estimate the number of paths, rather than cycles. Therefore, unlike the case for acyclic queries and queries with small cycles, optimistic estimators make overestimates, as real-world graphs contain many more paths than cycles. Therefore, the minimum-weight paths now lead to more accurate estimates.

As a demonstration of the benefits of modeling optimistic and pessimistic estimators in a common framework, our detailed evaluations show how the bound sketch optimization used for pessimistic estimators can also improve optimistic estimators. This experiment also shows that pessimistic estimators can be highly inaccurate compared to optimistic ones. Finally, we analyze several scalability aspects of estimators that make estimates on CEGs, which can inform other CEG-based estimators future research can propose.

For readers interested in the theory of pessimistic estimators, we further note that CEGs turn out to be very useful mathematical tools to prove properties of pessimistic estimators. For example, using CEGs in our proofs, we can derive combinatorial proofs to some properties of MOLP, e.g., that MOLP is at least as tight as the pessimistic estimator proposed by Cai et al [6] and are identical on acyclic queries over binary relations or that it is tighter than another bound called DBPLP [17].

2 QUERY AND DATABASE NOTATION

We consider conjunctive queries of the form

$$Q(\mathcal{A}) = R_1(\mathcal{A}_1), \ldots, R_m(\mathcal{A}_m)$$

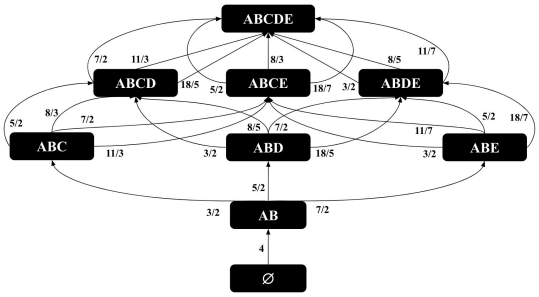


Figure 4: CEG_O for query Q_{5f} from Figure 1 when the Markov table (§4) contains joins up to size 2.

where $R_i(\mathcal{A}_i)$ is a relation with attributes $\mathcal{A}_i$ and $\mathcal{A} = \cup_i \mathcal{A}_i$. Most of the examples used in this paper involve edge-labeled subgraph queries, in which case each R_i is a binary relation containing a subset of the edges in a graph as source/destination pairs. Figure 2 presents an example showing a graph with edge labels A, B, C, D, and E. This graph can be represented using five binary relations, one for each edge label, as shown in Figure 2.

We will often represent queries over such relations using a graph notation. For example, consider the relations A and B from Figure 2. We will represent the query $Q(a_1, a_2, a_3) = A(a_1, a_2) \bowtie B(a_2, a_3)$ as $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$. Similarly, the query $Q(a_1, a_2, a_3) = A(a_1, a_2) \bowtie B(a_3, a_2)$ will be represented as $a_1 \xrightarrow{A} a_2 \xleftarrow{B} a_3$.

3 CEG OVERVIEW

Next, we offer some intution for *cardinality estimation graphs* (CEGs). A CEG for a query Q consists of:

- Vertices labeled with subqueries of Q, where subqueries are defined by subsets of Q's relations or attributes.
- Edges from smaller subqueries to larger subqueries, labeled with *extension rates* which represent the cardinality of the larger subquery relative to that of the smaller subquery.

Each bottom-to-top path (from $\emptyset$ to Q) in a CEG represents a different way of generating a cardinality estimate for Q. An estimator using a CEG picks one of these paths as an estimate. The estimate of a path is the product of the extension rates along the edges of the path. Equivalently one can put the logarithms of the extension rates as edge weights and sum the logarithms.

Figure 4 illustrates a CEG[1] for the query Q_{5f} shown in Figure 1 over the relations shown in Figure 2, assuming that statistics are available for any size-2 subqueries of Q_{5f}. Depending on the semantics of the edges in a CEG, there can be multiple edges between two sub-queries in a CEG. For example the last components of the two formulas, we present in Section 1 for Q_{5f}, $|\xleftarrow{D}\xrightarrow{E}|/|\xrightarrow{D}|$ and $|\xrightarrow{B}\xrightarrow{E}|/|\xrightarrow{B}|$, correspond to parallel edges that extend the sub-query over $ABCD$ edges with an E edge to Q_{5f}. Consider the leftmost path. The first extension rate from $\emptyset$ to $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$ is the known cardinality of $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$, which is 4, and the second extension rate has a weight 3/2, intuitively estimating that each $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$ path will extend to 3/2 many $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3 \xrightarrow{C} a_4$ paths. Continuing the extensions, the final estimate is $4 \times \frac{3}{2} \times \frac{5}{2} \times \frac{7}{2} = 52.5$.

[1]Specifically, it is a CEG_O, defined in Section 4.

Table 1: Example Markov table for h=2.

Path	|Path|
$\xrightarrow{B}$	2
$\xrightarrow{A}\xrightarrow{B}$	4
$\xrightarrow{B}\xrightarrow{C}$	3
...	...

In the rest of this paper, we will show how some of the optimistic and pessimistic estimators from literature can be modeled as instances of this generic estimator using different CEGs.

4 OPTIMISTIC ESTIMATORS

The estimators that we refer to as *optimistic* use statistics about the input database in formulas that make uniformity and independence or conditional independence assumptions. The cardinality estimators of several systems fall under this category. We focus on three estimators: *Markov tables* [2] from XML databases, graph summaries [23] from RDF databases, and the graph catalogue estimator of the Graphflow system [25] for managing property graphs. As we explain momentarily, despite being designed for systems that adopt different graph-based data models, these estimators are all extensions of each other.

We begin by giving an overview of the Markov tables estimator [2]. A Markov table of length $h \geq 2$ stores the cardinality of each path in an XML document's element tree up to length h and uses these to make predictions for the cardinalities of longer paths. Table 1 shows a subset of the entries in an example Markov table for $h = 2$ for our running example dataset from Figure 2. The formula to estimate a 3-path using a Markov table with $h = 2$ is to multiply the cardinality of one of the 2-paths with the consecutive 2-path divided by the cardinality of the common edge. For example, consider the query $Q_{3p} = \xrightarrow{A}\xrightarrow{B}\xrightarrow{C}$ against the dataset in Figure 2. The formula for Q_{3p} would be: $|\xrightarrow{A}\xrightarrow{B}| \times (|\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|)$. The formula assumes that the number of C edges that each B edge extends to is uniformly $r = |\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|$. Equivalently, this is the "average C-degree" of nodes in the $\xrightarrow{B}\xrightarrow{C}$ paths. The result of this formula is $4 \times \frac{3}{2} = 6$, which underestimates the true cardinality of 7. The graph summaries [23] for RDF databases and the graph catalogue estimator [25] for property graphs have extended the contents of what is stored in Markov tables, respectively, to other acyclic joins, such as stars, and small cycles, such as triangles.

4.1 Space of Possible Optimistic Estimators

We next represent optimistic estimators using a CEG that we call CEG_O. We assume that the given query Q is connected. CEG_O consists of the following:

- *Vertices:* For each connected subset of relations $S \subseteq \mathcal{R}$ of Q, we have a vertex in CEG_O with label S. This represents the sub-query $\bowtie_{R_i \in S} R_i$.
- *Edges:* Consider two vertices with labels S and S' s.t., $S \subset S'$. Let $\mathcal{D}$, for **d**ifference be $S' \setminus S$, and let $\mathcal{E} \supset \mathcal{D}$, for **e**xtension

be an entry in the Markov table, and let $\mathcal{I}$, for **i**ntersection, be $\mathcal{E} \cap S$. If $\mathcal{E}$ and $\mathcal{I}$ exist in the Markov table, then there is an edge with weight $\frac{|\mathcal{E}|}{|\mathcal{I}|}$ from S to S' in CEG_O.

When making estimates, we will apply two basic rules from prior work that limit the paths considered in CEG_O. First is that if the Markov table contains size-h joins, the formulas use size-h joins in the numerators in the formula. Second, for cyclic queries, which was covered in reference [25], an additional early cycle closing rule is used in the reference when generating formulas. In CEG formulation this translates to the rule that if S can extend to multiple S's and some S' contain additional cycles that are not in S, then only such outgoing edges of S to such S' are considered. Even when the previous rules are applied, there may be multiple $(\emptyset, Q)$ paths that lead to different estimates:

Example 1: Consider the CEG_O for Q_{5f} shown in Figure 4 which uses a Markov table of size 2. There are 36 $(\emptyset, Q)$ paths leading to 7 different estimates. Two examples are:

- $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}| \times \frac{|\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}|}{|\overset{B}{\longrightarrow}|} \times \frac{|\overset{B}{\longrightarrow}\overset{D}{\longrightarrow}|}{|\overset{B}{\longrightarrow}|} \times \frac{|\overset{B}{\longrightarrow}\overset{E}{\longrightarrow}|}{|\overset{B}{\longrightarrow}|} = 52.5$
- $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}| \times \frac{|\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}|}{|\overset{B}{\longrightarrow}|} \times \frac{|\overset{C}{\longleftarrow}\overset{D}{\longrightarrow}|}{|\overset{C}{\longrightarrow}|} \times \frac{|\overset{D}{\longleftarrow}\overset{E}{\longrightarrow}|}{|\overset{D}{\longrightarrow}|} = 57.6$

Example 2: Similarly, consider estimating Q_{5f} now with a Markov table with up to 3-size joins. The new CEG_O is shown in Figure 3, which contains multiple paths leading to 2 different estimates:

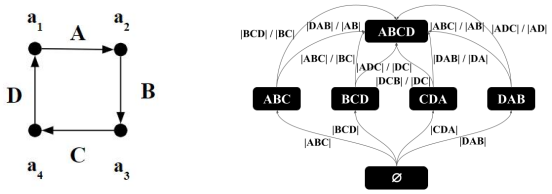
- $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}| \times \frac{|\overset{C}{\longleftarrow}\overset{D}{\longrightarrow}|}{|\overset{C}{\underset{E}{\longrightarrow}}|}$
- $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}| \times \frac{|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{D}{\longrightarrow}|}{|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}|} \times \frac{|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{E}{\longrightarrow}|}{|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}|}$

Both formulas start by $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}|$. The first "short-hop" formula makes one fewer independence assumption than the "long-hop" formula, which is an advantage. In contrast, the first estimate also makes a uniformity assumption that conditions on a smaller-size join, which might make it less accurate than the two assumptions made in the long-hop estimate, which condition on 2-size joins.

Any optimistic estimator implementation needs to make choices about which formulas to use, which corresponds to picking paths in CEG_O. We systematically identify a space of choices that an optimistic estimator can make along two parameters that capture the choices made in prior work:

- *Path length:* The estimator can identify a set of paths to consider based on the path lengths, i.e., number of edges or hops, in CEG_O, which can be: (i) maximum-hop (`max-hop`); (ii) minimum-hop (`min-hop`); or (iii) any number of hops (`all-hops`). Let $\mathcal{P}$ be the set of paths an estimator picks.
- *Aggregator:* To derive a final estimate, the estimator has to aggregate the estimates in $\mathcal{P}$. We identify three aggregators: (i) the path with the largest estimate (`max-aggr`); (ii) the path with the lowest estimate (`min-aggr`); or (iii) the average of all the estimates in $\mathcal{P}$ (`avg-aggr`).

Any combination of these two choices can be used to design an optimistic estimator. The original Markov tables [2] chose the `max-hop` paths. In reference [2] queries were paths, so when the path length is chosen, any CEG_O path gives the same estimate. Therefore an aggregator is not needed. Graph summaries [23] chooses the `min-hop` paths and leaves the aggregator unspecified. Graph catalogue [25] picks the `min-hop` and `min-aggr` aggregator. None of these estimators consider alternative choices an estimator can make. Instead, we do a systematic experimental analysis of this space of estimators in Section 6 and show that the best choices depend on the query structure, as optimistic estimators behave differently on different structures. In particular our experiments indicate that, as observed in the relational setting for estimators that make independence assumptions [19], optimistic estimators suffer from the well known under-estimation problem on acyclic queries and queries with small cycles. However, on queries with larger cycles, they tend to over-estimate. We next make an observation to explain this difference.



(a) 4-cycle query. **(b)** CEG_O.

Figure 5: A 4-cycle query and its CEG_O.

4.2 Large Cyclic Patterns

Recall that a Markov table stores the cardinalities of patterns up to some size h. Given a Markov table with $h \geq 2$, optimistic estimators can produce estimates for any acyclic query with size larger than h. But what about large *cyclic* queries with size larger than h?

Faced with a large cyclic query Q, we observe that the optimistic estimators do not actually produce estimates for Q. Instead, they produce an estimate for a similar acyclic Q' that includes all of Q's edges but is not closed. Consider estimating a 4-cycle query in Figure 5a using a Markov table with h=3. The CEG_O for this setting is shown in Figure 5b. Consider the left most path corresponding to the formula: $|\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}| \times |\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}\overset{D}{\longrightarrow}| / |\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}|$. Note that this formula is in fact estimating a 4-path $\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}\overset{D}{\longrightarrow}$ rather than the 4-cycle shown in Figure 5a. This is true for each path in CEG_O.

More generally, when queries contain cycles of length $> h$, CEG_O breaks cycles in queries into paths. Therefore, estimates over CEG_O can lead to *over*estimates for queries with large cycles, as there are often significantly more paths than cycles in real-world graphs. We note that this problem does not exist if a query contains a cycle C of length $> h$ that contains smaller cycles in them, such as a clique of size $h+1$, because the early cycle closing rule from Section 4.1 will avoid formulas that estimate C as a sub-query.

5 PESSIMISTIC ESTIMATORS

Starting from the seminal result by Atserias, Grohe, and Marx in 2008 [5], several upper bounds have been provided for the output sizes of join queries under different known statistics. For example the initial upper bound from reference [5], now called the *AGM bound*, used only the cardinalities of each relation, while later bounds, DBPLP [17], MOLP [17], and CLLP [1] used maximum degrees of the values in the columns and improved the AGM bound.

Since these bounds are upper bounds on the query size, they can be used as *pessimistic estimators*. This was done recently by Cai et al. [6] in an actual estimator implementation. We refer to this as the CBS estimator, after the names of the authors. We next show that some of the recent pessimistic estimators [6, 17] can also be modeled as making an estimate using a CEG.

5.1 MOLP

MOLP was defined in reference [17] as a tighter bound than the AGM bound that uses additional degree statistics about input relations that AGM bound does not use. We first review the formal notion of a degree. Let $\mathcal{X}$ be a subset of the attributes $\mathcal{A}_i$ of some relation $\mathcal{R}_i$, and let v be a possible value of $\mathcal{X}$. The *degree* of v in $\mathcal{R}_i$ is the number of times v occurs in $\mathcal{R}_i$, i.e. $deg(\mathcal{X}(v), \mathcal{R}_i) = |\{t \in \mathcal{R}_i | \pi_{\mathcal{X}}(t) = v\}|$. For example, in Figure 2, $deg(s(3), E) = 3$ because the outgoing E-degree of vertex 3 is 3. Similarly $deg(d(2), A)$ is 1 because the incoming A-degree of vertex 2 is 1. We also define $deg(\mathcal{X}, \mathcal{R}_i)$ to be the maximum degree in $\mathcal{R}_i$ of any value v over X, i.e., $deg(X, \mathcal{R}_i) = \max_v deg(\mathcal{X}(v), \mathcal{R}_i)$. So, $deg(d, A) = 3$ because vertex 13 has 3 incoming A edges, which is the maximum A-in-degree in the dataset. The notion of degree can be generalized to $deg(X(v), Y, \mathcal{R}_i)$, which refers to the "degree of a value v over attributes X in $\pi_Y \mathcal{R}_i$", which counts the number of times v occurs in $\pi_Y(\mathcal{R}_i)$. Similarly, we let $deg(X, Y, \mathcal{R}_i) = \max_v deg(X(v), Y, \mathcal{R}_i)$. Suppose a system has stored $deg(X, Y, \mathcal{R}_i)$ statistics for each possible $\mathcal{R}_i$ and $X \subseteq Y \subseteq \mathcal{A}_i$. MOLP is the output of this LP:

Maximize $s_{\mathcal{A}}$

$s_{\emptyset} = 0$

$s_X \leq s_Y, \ \forall X \subseteq Y$

$s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i)), \forall X, Y, E \subseteq \mathcal{A}, X \subseteq Y \subseteq \mathcal{A}_i$

The base of the logarithm can be any constant and we take it as 2. Let m_A be the optimal value of MOLP. Reference [17] has shown that 2^{m_A} is an upper bound on the size of Q.

MOLP CEG (CEG_M): It is not easy to directly see the solution of the MOLP on our running example. However, we next show that we can represent the MOLP bound as the cost of minimum-weight $(\emptyset, Q)$ path in a CEG that we call CEG_M.

- *Vertices:* For each $X \subseteq \mathcal{A}$, the variable s_X in MOLP represents an upper bound on the size of $Q_X = \Pi_X Q$. Therefore, for each $X \subseteq \mathcal{A}$ there is a vertex in CEG_M.
- *Extension Edges:* For each $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i))$ inequality, there is an edge with weight $\log(deg(X, Y, \mathcal{R}_i))$ between any $W_1 = X \cup E$ and $W_2 = Y \cup E$. These inequalities intuitively indicate the following: each tuple $t_{X \cup E} \in Q_{X \cup E}$ can extend to at most $deg(X, Y, \mathcal{R}_i)$ $Q_{Y \cup E}$ tuples.
- *Projection Edges:* For each $s_X \leq s_Y$ inequality (i.e., $\forall X \subseteq Y$), add an edge with weight 0 from Y to X. These indicate that the size of Q_X is at most the size of Q_Y, if Y is a larger subquery.

Figure 6 shows the CEG_M of our running example. For simplicity, we use actual degrees instead of their logarithms as edge weights and omit the projection edges in the figure. Below we use $(\emptyset, \mathcal{A})$ instead of $(\emptyset, Q)$, to represent the bottom-to-top paths in CEG_M.

Theorem 5.1. *Let Q be a query with degree statistics $deg(X, Y, R_i)$ for each $\mathcal{R}_i$ and $X \subseteq Y \subseteq \mathcal{A}_i$. The optimal solution m_A to the MOLP of Q is the weight of the minimum-weight $(\emptyset, \mathcal{A})$ path in CEG_M.*

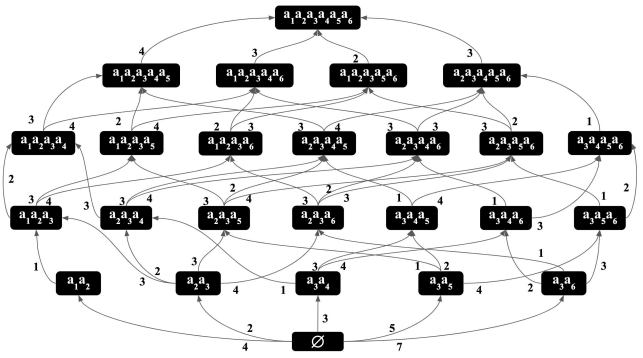


Figure 6: CEG_M for query Q_{5f} in Figure 1.

The proof of Theorem 5.1 can be found in the longer version of our paper [7]. With this connection, readers can verify that the MOLP bound in our running example is 96 by inspecting the paths in Figure 6. In this CEG, the minimum-weight $(\emptyset, \mathcal{A})$ path has a weight of 96, corresponding to the leftmost path in Figure 6. We make two observations.

Observation 1: Reference [17] proves through a numeric LP-duality argument that $2^{m_{\mathcal{A}}}$ is an upper bound on the the output size of the query (OUT), i.e., $OUT \leq 2^{m_{\mathcal{A}}}$ (see Prop. 2 [17]). Our CEG formulation of MOLP provides arguably a simpler proof of this property. Note that each $(\emptyset, \mathcal{A})$ path in CEG_M corresponds to a sequence of extensions from $\emptyset$ to Q and is an estimate of the cardinality of Q. Since we are using maximum degrees on the edge weights, each $(\emptyset, \mathcal{A})$ path is by construction an upper bound on Q. So any path in CEG_M is a pessimistic estimator. Since for any $(\emptyset, \mathcal{A})$ path P in CEG_M, $OUT \leq 2^{w(P)}$ and by Theorem 5.1, $m_{\mathcal{A}}$ is equal to the weight of the minimum-weight $(\emptyset, \mathcal{A})$ path in CEG_M, $OUT \leq 2^{m_{\mathcal{A}}}$.

Observation 2: Theorem 5.1 implies that MOLP can be solved using a combinatorial shortest-path algorithm instead of a numeric LP solver.

5.2 CBS and Bound Sketch Optimization

We review the CBS estimator very briefly and refer the reader to reference [6] for details. CBS estimator has two subroutines *Bound Formula Generator (BFG)* and *Feasible Coverage Generator (FCG)* (Algorithms 1 and 2 in reference [6]) that, given a query Q and the degree statistics about Q, generate a set of *bounding formulas*. A coverage is a mapping (R_j, A_j) of a subset of the relations in the query to attributes such that each $A_j \in \mathcal{A}$ appears in the mapping. A bounding formula is a multiplication of the known degree statistics and is an upper bound on the size of a query. In the long version of our paper [7], we prove that each path in CEG_M corresponds to a bounding formula and vice versa. With this observation, we show that MOLP is at least as tight as the CBS estimator on general acyclic queries and is exactly equal to the CBS estimator over acyclic queries over binary relations. Henceforth, we do not differentiate between MOLP and the CBS estimator. [2]

[2] A similar connection between MOLP and CBS cannot be established for cyclic queries. This is because, although not explicitly acknowledged in reference [6], on cyclic queries, the CBS estimates are not guaranteed to be pessimistic. We provide a counter example in the long version of our paper [7]. In contrast, MOLP generates a pessimistic estimate for arbitrary, so both acyclic or cyclic, queries.

5.2.1 Bound Sketch. We next review the bound sketch optimization from reference [6] to improve the CBS/MOLP estimates. We describe bound sketch using the CEG framework (see reference [6] for the original description). Given a partitioning budget K, for each bottom-to-top path in CEG_M, the optimization partitions the input relations into multiple pieces and derives K many subqueries of Q. Then the estimate for Q is the sum of estimates of all K subqueries. Intuitively, partitioning decreases the maximum degrees in subqueries to yield better estimates whose sum is guaranteed to be more accurate than making a direct estimate for Q.

We divide the edges in CEG_M into two. Recall that each edge $W_1 \xrightarrow{e_j} W_2$ in CEG_M is constructed from an inequality of $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, R_i))$ in MOLP. We call e_j (i) an unbound edge if $X = \emptyset$, i.e., the weight of e_j is $|R_i|$; (ii) a bound edge if $X \neq \emptyset$, i.e., the weight of e_j is actually the degree of some value in a column of R_i. Note that unbound edge extends W_1 exactly with attributes $\mathcal{A}_i$, i.e., $W_2 \setminus W_1 = \mathcal{A}_i$ and a bound edge with attributes Y, i.e., $W_2 \setminus W_1 = Y$. Below, we refer to these attributes as "extension" attributes.

Step1: For each $p = (\emptyset, \mathcal{A})$ path in CEG_M (so a bounding formula in the terminology used in reference [6]), let S be the join attributes that are not extension attributes through a bounded edge[3]. For each attribute in S, allocate $K^{1/|S|}$ partitions. For example, consider the path $P_1 = \emptyset \xrightarrow{|B|} a_2a_3 \xrightarrow{deg(a_3,C)} a_{2-4} \xrightarrow{deg(a_2,A)} a_{1-4} \xrightarrow{deg(a_3,E)} a_{1-4}a_6 \xrightarrow{deg(a_3,D)} a_{1-6}$ in the CEG_M of Q_{5f} from Figure 6, where a_{i-j} refers to $a_ia_{i+1}...a_j$. Then both a_2 and a_3 would be in S. For path $P_2 = \emptyset \xrightarrow{|A|} a_1a_2 \xrightarrow{deg(a_2,B)} a_{1-3} \xrightarrow{deg(a_3,C)} a_{1-4} \xrightarrow{deg(a_3,D)} a_{1-5} \xrightarrow{deg(a_3,E)} a_{1-6}$, only a_2 would be in in S.

Step2: Partition each relation R_i as follows. Let PA_i, for **p**artition **a**ttributes, be $PA_i = S \cap \mathcal{A}_i$ and z be $|PA_i|$. Then partition R_i into $K^{z/|S|}$ pieces using z hash functions, each hashing a tuple $t \in R_i$ based on one of the attributes in PA_i into $\{0, ..., K^{1/|S|} - 1\}$. For example, the relation B in our example path P_1 would be partitioned into 4, B_{00}, B_{01}, B_{10}, and B_{11}.

Step3: Then divide Q into K components $Q_{0...0}$, to $Q_{K^{1/|S|}-1,...,K^{1/|S|}-1}$, such that $Q_{j_1,...,j_z}$ contains only the partitions of each relation R_i that matches the $\{j_1, ..., j_z\}$ indices. For example, in our example, $Q_{0...0}$ is $A_0 \bowtie B_{0,0} \bowtie C_0 \bowtie D_0 \bowtie E_0$. This final partitioning is called the bound sketch of Q for path p.

5.2.2 Implementing Bound Sketch in Optimistic Estimators. Note that a bound sketch can be directly used to refine any estimator using a CEG, as it is a general technique to partition Q into subqueries based on each path p in a CEG. Specifically, we can use a bound sketch to refine optimistic estimators, which is a direct benefit of using a common framework to model both optimistic and pessimistic estimators. We will evaluate benefits of bound sketch on optimistic estimators in Section 6.3. We implemented the bound sketch optimization for optimistic estimators as follows. Given a partitioning budget K and a set of queries in a workload, we worked backwards from the queries to find the necessary subqueries, and for each subquery the necessary statistics that would be needed are stored in the Markov table. Bound sketches are query-specific, so computing the right Markov table entries (or degree statistics) requires pre-knowledge of the workloads. We are unaware of a workload-agnostic technique to generate all necessary statistics for arbitrary queries.

[3]These correspond exactly to the attributes that are "unconditionally" covered by a relation in a bounding formula (see Section 3.4 of reference [6]).

Table 2: Dataset descriptions.

Dataset	Domain	|V|	|E|	|E. Labels|
IMDb	Movies	27M	65M	127
YAGO	Knowledge Graph	13M	16M	91
DBLP	Citations	23M	56M	27
WatDiv	Products	1M	11M	86
Hetionet	Social Networks	45K	2M	24
Epinions	Consumer Reviews	76K	509K	50

5.3 Use of CEGs in Proofs & A Note on CLLP

Another application of CEGs is that they can be useful mathematical tools to provide proofs for properties of some existing bounds. As we mentioned earlier, our proof in the longer version of our paper [7] that shows that CBS is equivalent to MOLP on acyclic queries over binary relations uses an argument analyzing CEGs. Similarly, the longer version of our paper [7] reviews another bound called DBPLP bound and provides an alternative proof that MOLP is tighter than DBPLP using CEGs.

We end this section with a note on the CLLP bound [1], which is the tightest known join query size bound. CLLP extends MOLP with a set of sub-modularity constraints. With the addition of these constraints, we can no longer map the solution of CLLP to a path in a CEG because sub-modularity constraints represent a relationship between 4 sub-queries and do not seem to have an interpretation as weighted edges as the other constraints of MOLP, which are between 2 sub-queries. We also note that CLLP generalizes another LP from reference [13] called GLVV, which was introduced to study join output sizes under functional dependencies. As part of deriving CLLP, reference [1] also uses lattices, which might look similar to our CEGs. Reference [1] uses lattices as a mathematic tool to determine queries and functional dependencies under which GLVV is tight. Instead, we use CEGs *to solve an LP combinatorially.*

6 EVALUATION

We next present extensive experiments that: (i) evaluate the accuracies of the space of optimistic estimators we described on a large suite of datasets and workloads; (ii) demonstrate the benefits of applying the bound sketch optimization to optimistic estimators, where we also compare the accuracies of optimistic and pessimistic estimators; and (iii) analyze several scalability aspects of estimators that make estimates by picking paths on CEGs. Our code, datasets, and queries are available at https://github.com/cetechreport/CEExperiments.

6.1 Setup, Datasets and Workloads

For all of our experiments, we use a single machine with two Intel E5-2670 at 2.6GHz CPUs, each with 8 physical and 16 logical cores, and 512 GB of RAM. We used a total of 6 real-world datasets, shown in Table 2, and 6 workloads on these datasets. Our dataset and workload combinations are as follows.

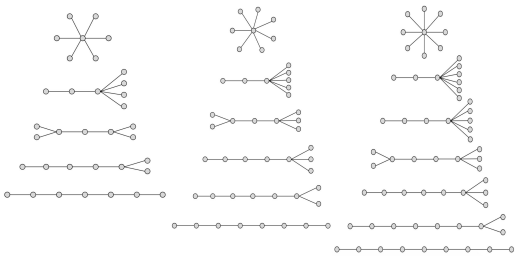


Figure 7: Query templates for the `Acyclic` workload. Edge directions are omitted in the figure.

IMDb and JOB [21]: The IMDb relational database, together with a workload called JOB, has been used for cardinality estimation studies in prior work [6, 21]. We created property graph versions of the this database and workload as follows. IMDb contains three groups of tables: (i) *entity tables* representing entities, such as actors (e.g., `name` table), movies, and companies; (ii) *relationship tables* representing many-to-many relationships between the entities (e.g., the `movie_companies` table represents relationships between movies and companies); and (iii) *type tables*, which denormalize the entity or relationship tables to indicate the types of entities or relationships. We converted each row of an entity table to a vertex. Let u and v be vertices representing, respectively, rows r_u and r_v from tables T_u an T_v. We added two sets of edges between u and v: (i) a *foreign key edge* from u to v if the primary key of row r_u is a foreign key in row r_v with a fixed edge label E_{T_u,T_v}; (ii) a *relationship edge* between u to v if a row r_ℓ in a relationship table T_ℓ connects row r_u and r_v. The label of the edge between u and v in this case would come from the type value of the r_ℓ tuple that "connected" u and v. For example if a movie entity u is joined with company entity v through the `movie_companies` table, then there can be 4 different labels this edge can have: distributor, producer, special effects, miscallenous. These are the four values in the `company_type` table that denormalizes the type information in `movie_companies`.

We then transformed the JOB workload [21] into equivalent subgraph queries on our transformed graph. We removed non-join predicates in the queries since we are focusing on join cardinality estimations. This resulted in 7 join query templates, including four 4-edge queries, two 5-edge queries, and one 6-edge query. Because we are focusing on edge labeled queries, for any query edge that corresponds to a join over a relationship table T_ℓ, we generated a random edge label by picking uniformly at random one of the type values of T_ℓ. We generated 100 query instances this way from each query template and removed the ones whose outputs were empty. The final workload contained 369 queries.

WatDiv [40] and `WatDiv-Acyclic` Workload: WatDiv [4] is a synthetic knowledge graph that has its own workload in SPARQL format with 12400 original queries. We converted these queries into equivalent subgraph queries by removing their vertex predicates and we then removed queries with at most 3 edges. After removing duplicates among the remaining queries there were 75 different queries left, 9 of which is cyclic and the other 64 acyclic. Because 9 queries is very small for a workload, we use only the 64 acyclic queries call this the `WatDiv-Acyclic` workload.

YAGO 1 [45] and `G-CARE-Acyclic` and `G-CARE-Cyclic` Workloads: G-CARE [31] is a recent cardinality estimation benchmark for subgraph queries. From this benchmark we took the YAGO knowledge graph dataset and the acyclic and cyclic query workloads for that dataset. The `acyclic` workload contains 382 queries generated from query templates with 3-, 6-, 9-, and 12-edge star and path queries, as well as randomly generated trees. We will refer to this workload as `G-CARE-Acyclic`. The cyclic query workload contains 240 queries generated from templates with 6-, and 9-edge cycle, 6-edge clique, 6-edge flower, and 6- and 9-edge petal queries. We will refer to this workload as `G-CARE-Cyclic`.

DBLP [10], WatDiv [40], Epinions [11], and Hetionet [16] and `Acyclic` and `Cyclic` Workloads: We used three other datasets: (i) Hetionet: a biological network; (ii) DBLP: a real knowledge graph; and (iii) Epinions: a real-world social network graph. Epinions is a dataset that by default does not have any edge labels. We added a random set of 50 edge labels to Epinions. Our goal in using Epinions was to test whether our experimental observations also hold on a graph that is guranteed to not have any correlations between edge labels. For these datasets we created one acyclic and one cyclic query workload, which we refer to as `Acyclic` and `Cyclic`. The `Acyclic` workload contains queries generated from 6-, 7-, or 8-edge templates, shown in Figure 7. These templates are systematically picked to ensure that for each query size k, there is a pattern of every possible depth. Then, we generated 20 non-empty instances of each template by putting one edge label uniformly at random on each edge, which yielded 360 queries in total. The `Cyclic` workload contains queries generated from templates used in reference [25]. We then randomly generated instances of these queries by randomly matching each edge of the query template one at a time in the datasets. Because the WatDiv's original queries contained only acyclic queries, we used the `Cyclic` workload also on WatDiv. We generated 70 queries for DBLP, 212 queries for Hetionet, 129 queries for WatDiv, and 394 queries for Epinions.

6.2 Space of Optimistic Estimators

We begin by comparing our 9 optimistic estimators on CEG_O, we defined. In order to set up an experiment in which we could test all of the 9 possible optimistic estimators, we used a Markov table with h=3. A Markov table with only 2-size joins can not test different estimators based on different path-length choices or any cyclic query.

To compare the accuracies of different estimators, for each query Q in our workloads we make an estimate using each estimator and compute its q-error. If the true cardinality of Q is c and the estimate is e, then the q-error is $\max\{\frac{c}{e}, \frac{e}{c}\} \geq 1$. For each workload, this gives us a distribution of q-errors, which we compare as follows. First, we take the logs of the q-errors so they are now ≥ 0. If a q-error was an underestimate, we put a negative sign to it. This allows us to order the estimates from the least accurate underestimation to the least accurate overestimation. We then generate a box plot where the box represents the 25th, median, and 75th percentile cut-off marks. We also compute the mean of this distribution, excluding the top 10% of the distribution (ignoring under/over estimations) and draw it with a red dashed line in box plots.

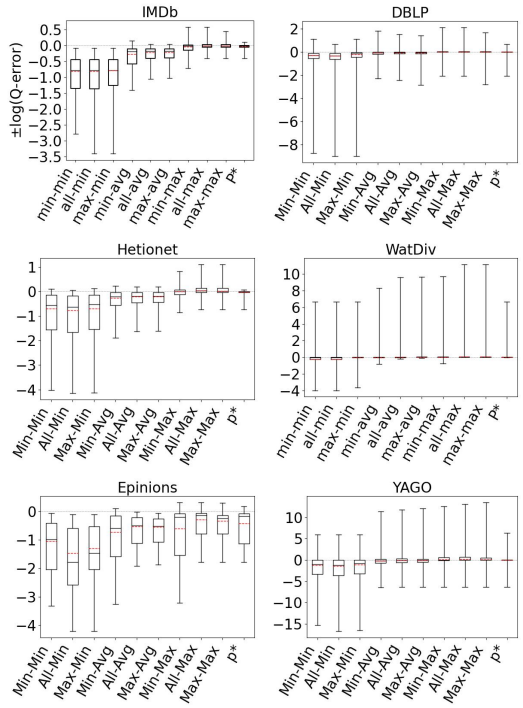


Figure 8: Evaluation of the optimistic estimators on CEG_O on acyclic queries. Estimators are labeled"P-A": P is the path length (one of max-hop, min-hop, or all-hops) and A the aggregator (one of max-aggr, min-aggr, or avg-aggr).

6.2.1 Acyclic Queries and Cyclic Queries With Only Triangles. Our first question is: Which of the 9 possible optimistic estimators leads to most accurate estimates on acyclic queries and cyclic queries that contain ≤3 edges on CEG_O? We compare our 9 estimators on CEG_O for each acyclic query workload in our setup (for IMDb, WatDiv, and YAGO, using JOB, `WatDiv-Acyclic`, and `G-CARE-Acyclic` workloads). Because estimators that use avg aggregator cannot finish 12-size queries in YAGO in a reasonable time, we randomly sampled 100K paths for these estimators and take their average. As we demonstrate in our scalability experiments in Section 6.4 even the number of paths in a 9-star query is prohibitively large.

We then compare our 9 estimators on each cyclic query workload, but only using the queries that only contain triangles as cycles. All except one clique-6 query in `GCARE-Cyclic` contained cycles with more than 3 edges, so we omit `GCARE-Cyclic` combination.

Our results are shown in Figure 8 (ignore the P* column for now). We make several observations. First, regardless of the path-length choice, the `max` aggregator (the last 3 box plots in the figures) makes significantly more accurate estimates (note that the y-axis on the plots are in log scale) than `avg`, which in turn is more accurate than `min`. This is true across all acyclic experiments and all datasets. For example, on IMDb and `JOB` workload, the `all-hops-min`, `all-hops-avg`, and `all-hops-max` estimators have log of mean q-errors of 6.5 (underestimation), 1.7 (underestimation), and 1.02 (understimation), respectively. *Therefore on acyclic queries, when there are multiple formulas that can be used to make an estimate, using the pessimistic ones is an effective technique to combat the well known underestimation problem.* This can give up to three orders of magnitude lower mean q-errors than, e.g., the `min-hop-min` estimator used in prior work.

We next analyze the effects of path-length choices. Observe that across all experiments, if we ignore the outliers and focus on the 25-75 percentile boxes, `max-hop` and `all-hops` do at least as well as `min-hop`. Further observe that on IMDb, Hetionet, and on the `Acyclic` workload on Epinions, `max-hop` and `all-hops` lead to significantly more accurate estimates. Finally, the performance of `max-hop` and `all-hops` are comparable across our experiments. We verified that this is because `all-hops` picks one of the `max-hop` paths in majority of the queries in our workloads. Therefore we observe that the advantage of long-hop paths that condition on 2-size joins when making uniformity assumptions is generally stronger than its disadvantage of making more independence assumptions. Since `max-hop` enumerates strictly fewer paths than `all-hops` to make an estimate, we conclude that on acyclic queries, systems implementing the optimistic estimators can prefer the `max-hop-max` estimator.

Figure 9 shows the accuracies of the 9 estimators on cyclic query workloads with only triangles. Our observations are similar to those for acyclic queries, and we find that the `max` aggregator yields more accurate estimates than other aggregators, irrespective of the path length. When using the max aggregator, we also observe that `max-hop` performs at least as well as `min-hop`. Therefore, as we observed for acyclic queries, *we find* `max-hop-max` *estimator to be an effective way to make accurate estimations for cyclic queries with only triangles.*

6.2.2 Effects of Query Templates, Sizes, and h. For the above experiments, we also performed a more detailed analysis studying the effects of different query templates (i.e., shapes). That is, we grouped the queries in each of our workloads into different isomorphic templates, e.g., a 5-star, a 5-star with a two-path tail, etc., and re-generated the accuracies of our 9 estimators as in Figures 8 and 9 by only plotting the accuracies of one group. These constitute over 100 figures and can be found in our github repo. Broadly these figures verified that our conclusions generally hold for different acyclic and cyclic query template we used in our workloads.

We further divided the queries in our `Acyclic` workloads based on their sizes, and analyzed the accuracy distribution of our overall recommended estimator `max-hop-max`. Figure 10 shows our boxplots for size analysis. Overall, we see that as the size of the queries increases accuracy decreases. This can be seen by the boxplots getting taller as the size increases. This is expected because optimistic estimators make more independence and uniformity assumptions in their formulas for larger queries. One exception is that the accuracy on YAGO 8-size queries is better than all other sizes, but this is mainly because this boxplot contains very few, only 8, queries (shown on top of boxplots), while other boxplots contains many more, e.g., over 150 queries.[4]

[4]We did a similar analysis studying the effect of query depths on accuracy, i.e., for example whether star queries, which have a depth of 2 irrespective of size were harder to estimate than paths, but we did not observe a very clear pattern as we can see for size and omit these experiments.

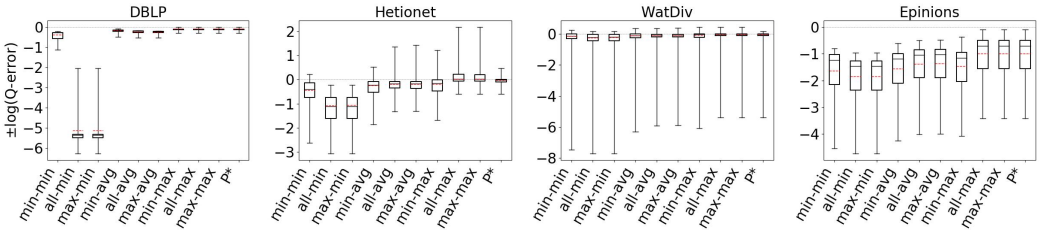


Figure 9: Evaluation of the space of optimistic estimators on CEG_O on `Cyclic` workload on queries with only triangles.

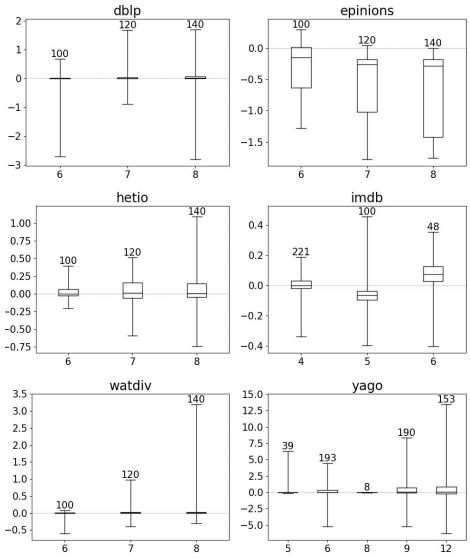


Figure 10: Accuracy of `max-max` for different query sizes (x-axis). The number on top of boxplots is the number of queries in the boxplot.

Finally, on two of our datasets, DBLP and Hetionet, we verified that increasing h from 2 to 3 improves accuracy in optimistic estimators. These figure appear in the longer version of our paper [7]. For example, the `max-hop-max` estimators' median accuracy improves from 1.37 to 1.02 in Hetionet.

6.2.3 Cyclic Queries With Cycles of Size >3. Recall our observation that when faced with queries that contain large cycles, existing optimistic estimators estimate paths instead of cycles, which we expect to yield highly inaccurate overestimates as real-world graphs contain many more paths than cycles. Therefore, unlike the case for acyclic queries and queries with small cycles, we now expect that estimates based on CEG_O will be pessimistic. To verify this hypothesis, our next experiments compare the performance of optimistic estimators for each dataset-cyclic query workload combination, but only using queries that contain cycles of size > 3.

Figure 11 shows our results. As we expected, we now see that the optimistic estimators yield overestimates for the majority of the queries. This can be seen by observing that except for a few exceptions the 25-75 percentile boxes of the boxplots are above 0. In addition, estimators using the `min` aggregator perform generally better, sometimes with several orders of magnitude difference, which can be seen by observing the mean accuracy lines of the boxplots. For example, on YAGO, the mean accuracy lines of `max-hop-min` and `max-hop-max` are, respectively, 0.20 and 3.61 on logarithmic scale, which correspond to absolute q-errors of 1.58 (overestimation) and 4043.86 (overestimation). We also observe that there is less sensitivity to the path-length choice when using the `min` aggregator, and any of the path-length choices perform reasonably well.

6.2.4 P^* Estimator and Room for Improvement. Our next question is: How much room for improvement is there for the space of optimistic estimators? To answer this, we consider a thought experiment in which, for each query in our workloads, an oracle picks the most accurate path in CEG_O. The accuracies of this oracle-based estimator are shown as P^* bars in our bar charts in Figures 8, 9, and 11. We compare the P^* bars in these figures with the `max-hop-max` estimator on on acyclic queries and cyclic queries with only triangles, and `max-hop-min` estimator on queries with larger cycles. We find that on acyclic queries, shown in Figure 8, we generally see little room for improvement, though there is some room in Hetionet and YAGO. For example, on Hetionet, the [25-75] percentile cutoffs for `max-hop-max` and P^* are [1.1 (underestimation), 1.4 (overestimation)] and [1.1 (underestimation), 1.0] in absolute q-error, respectively. We see more room for improvement on cyclic query workloads with large cycles, shown in Figures 11, e.g., the [25, 75] percentile cutoffs in absolute q-error for `max-hop-min` and P^* on DBLP are [346.4 (underestimation), 1.9 (overestimation)] and [2.0 (underestimation), 1.04 (overestimation)], respectively. This indicates that future work on CEG-based estimators can focus on workloads with large cycles to find opportunities for improvement.

As we discuss in Section 8, we hope CEGs can be the foundation for proposing other estimators for future work. As a demonstration of the flexibility of our CEG-framework to design new CEG-based estimators, in the longer version of our paper, we present one possible approach (though others can be proposed) to remedy the overestimates of the estimators on CEG_O for queries with large cycle queries. Specifically, we describe a new CEG, that modifies CEG_O with new edge weights that incorporate a cycle closing effect and show that estimates on this new CEG can be more accurate than those on CEG_O.

6.3 Effects of Bound Sketch

The bound sketch optimization was designed and shown to significantly improve the accuracy of the CBS/MOLP pessimistic estimator in reference [6]. Recall from Section 5.2, that as a direct benefit of modeling optimistic and pessimistic estimators in our CEG framework, we had observed that this optimization can also be applied to

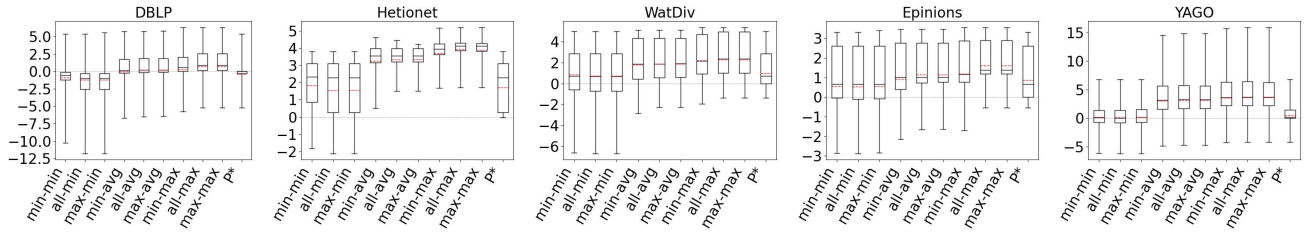


Figure 11: Evaluation of optimistic estimators on CEG_O on cyclic queries with cycles of 4 or more edges.

optimistic estimators (and in fact any CEG-based estimator). Our next experiments aim to demonstrate that indeed the bound sketch optimization can improve optimistic estimators. As in reference [6], we focus on acyclic workloads and apply the bound sketch optimization to our suggested `max-max` optimistic estimator. We also reproduce the result from reference [6] that bound sketch improves CBS/MOLP.

Specifically, we tested the effects of bound sketch on the `JOB` workload on IMDb and `Acyclic` workload on Hetionet and Epinions. We use h=2 in our Markov tables. Note that when h=2 only the aggregator heuristic is relevant because max-hop vs min-hop choice does not exist, so we refer to `max-max` simply as `max-aggr`. Then we applied the bound sketch optimization to both `max-aggr` (on CEG_O) and MOLP estimators and measured the q-errors of the estimators under partitioning budgets of 1 (no partitioning), 4, 16, 64, and 128. For the largest partitioning size of 128, our offline partitioning code that also generates Markov table entries for each partition, took 18 minutes for IMDb, 41 minutes for Hetionet, and 58 seconds for Epinions. We did not optimize this code, so these numbers can be improved with more performance-oriented implementations.

Our results are shown in Figure 12. We make two observations. First, as demonstrated in reference [6], our results confirm that bound sketch improves the accuracy of MOLP. The mean accuracy of MOLP increases between 15% and 89% across all of our datasets when moving between 1 and 128 partitions. Similarly, we also observe significant gains on the `max-aggr` estimator. On Hetionet and Epinions, partitioning improves the mean accuracy at similar rates: by 25% and 89%, respectively. We observe more modest gains on IMDb though 93% of their q-errors strictly improve under `max-aggr`.

Second, we observe that MOLP overall produces highly inaccurate, i.e., very pessimistic, estimates, which was also observed in reference [31]. This can be seen by observing that for each of our datasets, the scale of the (logarithmic q-error) y-axis in MOLP bars indicate q-errors that are several orders of magnitude larger than the q-errors for `max-aggr`. For example on Hetionet, across all partitioning budgets, the absolute values of the mean accuracy lines in the box plots for max-aggr are below 0.15 (so an absolute q-error less than 1.4), while the accuracy lines are consistently above 3.0 for MOLP (so an absolute q-error greater than 1042).

6.4 Scalability of Markov Tables and CEGs

We next analyze several scalability aspects of Markov tables and CEGs. These scalability aspects were not discussed in detail in prior work and can inform future research that can use CEGs to design

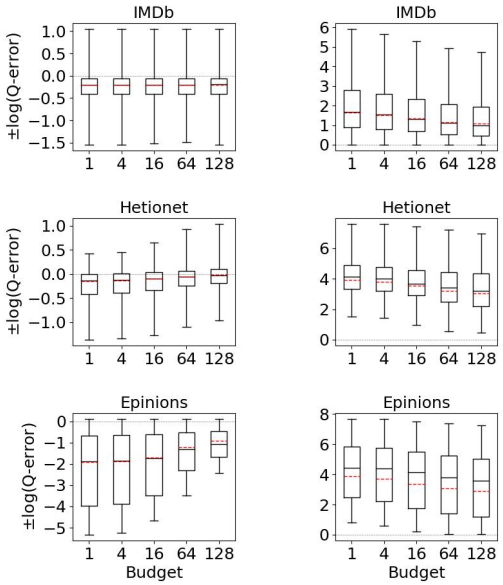


Figure 12: Effects of bound sketch on `max-aggr` estimator (left column) and MOLP (right column) estimators.

and implement new estimators. The sizes of Markov tables for a given dataset depends on: (1) how many subgraph topologies exist in the graph for a given size, which depends on the structure of the input graph; and (2) for each subgraph topology S, which edge combinations exist. For example, if $h = 3$ and a graph has K many 3-size subgraph topologies and L many edge labels, in the worst-case, $O(KL^3)$ many entries can exist in the Markov table. The number of edge labels in our datasets are between 24 to 127, so the number of entries are not prohibitively large even in the worst-case. In addition, on many datasets many edge combinations do not exist because of the constraints in the datasets. For example, Hetionet models a biological network about diseases and by construction there is no 2-path with both `treats` labels, because drugs treat diseases and diseases do not further treat any other entity.

Table 3 reports the size of full Markov tables for $h = 2$ and $h = 3$ for each of our datasets, which is at most 39MBs. At least for h=2 and h=3, these Markov tables are very space-efficient for these datasets. Amongst our datasets, Epinions has the largest Markov tables. This is expected because the edge labels in this graph are synthetic and randomly assigned so more edge combinations exist for each subgraph topology. However on graphs with thousands of edge labels, the sizes of Markov tables can still become very large.

Table 3: Markov table sizes for our datasets.

	h = 2		h = 3	
Dataset	**entries**	**file size**	**entries**	**file size**
IMDb	8K	0.4 MB	597K	30 MB
YAGO	4K	0.2 MB	95K	5 MB
DBLP	0.8K	0.04 MB	27K	1 MB
WatDiv	2K	0.09 MB	45K	2 MB
Hetionet	0.3K	0.02 MB	6K	0.3 MB
Epinions	5K	0.3 MB	774K	39 MB

Reference [2] has described methods to increase the scalability of Markov tables, by omitting subgraph topologies with small counts or merging multiple entries into one entry. These technique or their modifications can be used to mitigate these scalability issues.

We next discuss the complexity of searching an entry E in a Markov table. For h=2 and h=3, ignoring edge directions, there are only three possible entry topologies, which are stars, paths, or triangles. We have implemented a specialized search algorithm for h=2 and h=3, which first searches for the topology and then matches the sequence of edge directions, and then the edge labels, so entry searching is constant time. The only general solution we are aware of for indexing and searching entries in larger Markov tables appears the source code [14] for the (graph catalogue) optimistic estimator of reference [25][5]. This solution creates a key from each entry E, by combining the degree information of the query vertices in E, the edge directions and then edge labels and uses this key to index entries. However, multiple entries, E_1 and E_2, in general can have the same key, in which case the code does an isomorphism test, which is not constant time. We think using Markov Tables with size h=4 or larger will have practicality issues both because the size of the Markov tables can get much larger and searching entries will be slower.

Finally, we discuss the complexity of making estimates in the CEG for a query Q with different estimators. The number of edges and paths in a CEG depends on Q and the h value of the Markov table. In terms of queries, the CEGs of stars are largest, as every possible choice of k query-edges of Q forms a valid sub-query and can be distinct (e.g., if every edge label is distinct so there are no isomorphic sub-queries), and every k-edge sub-query can in the worst-case extend to every (k+1)- and (k+2)-size sub-query. Under this worst-case assumption, Table 4 shows the number of edges and $\emptyset$ to Q paths when using a Markov table of size h=2 and h=3 for stars between 6 and 10 edges .

A critical point about the practicality of optimistic estimators is that the complexity of making an estimate depends highly on the aggregator that is used. For `max` and `min` aggregators, the complexity is commensurate *with the number of edges* in the graph. That is because CEGs are DAGs, and there is a standard linear time dynamic-programming-based shortest (or longest) path finding algorithm in DAGs [9], which is what we use in our `max-hop-max` implementation. However for the `avg` aggregator, the complexity is commensurate *with of number of paths*, as it needs to investigate each path. As shown in Table 4, for large queries. the number of CEG paths is prohibitively large, and this aggregator is not practical.

[5]See the Catalog.java and QueryGraph.java files.

Table 4: CEG edges and path counts for 6- to 10-stars. Overflow indicates a number greater than $2^{64} - 1$.

	h = 2		h = 3	
Shape	**|edges|**	**|paths|**	**|edges|**	**|paths|**
6-Star	180	486000	225	51M
7-Star	434	153M	651	30B
8-Star	1008	90B	1746	30T
9-Star	2286	90T	4572	48998T
10-Star	5100	Overflow	11475	Overflow

6.5 Impact on Plan Quality

Reference [19] established that cardinality estimation is critical for optimizers to generate good plans for RDBMSs as it leads to better plan generation. Several other work has verified this in different contexts, in RDBMSs [6] and in RDF systems [31]. In our final set of experiments we set out to verify this in our context too by comparing the impact of our estimators on plan quality. We used the RDF-3X system [29] and its source code available here [35]. We issued our `Acyclic` workload as join-only queries to RDF-3X on the DBLP and WatDiv datasets. We then ran the query under 10 configurations: first using RDF-3X's default estimator and then by injecting the cardinality estimates of our 9 optimistic estimators to the system. We used h=3 in our Markov tables to be able to differentiate between max-hop and min-hop path choices in these estimators. We then filtered out the queries in which all of the 10 estimators lead to exactly the same plan. We were left with 41 queries for DBLP and 25 queries for WatDiv. We ran each query 5 times and report the best execution time.

The open source version of RDF-3X uses a simple cardinality estimator that based on basic statistics about the original triple counts and some 'magic' constants. We analyzed the final estimates of the RDF-3X estimator on the WatDiv queries and compared with the other estimators. We omit the full results but while the RDF-3X estimator had a median q-error of 4.001 underestimation, the worst-performing of the 9 optimistic estimators had a median q-error of only 1.760 underestimation. So we expect RDF-3X's estimator to lead to worse plans than the other estimators. We further expect that the more accurate of the optimistic estimators, such as `max-hop-max`, yield more efficient plans than the less accurate ones, such as `min-hop-min`.

Figure 13 shows the runtimes of the system under each configuration where the y-axis shows the log-scale speedup or slow down of each plan under each estimator compared to the plans under the default RDF-3X estimator. Although a visible improvement is not identified in DBLP, on WatDiv, observe that the median lines of the 9 estimators are above 0, indicating the each of these estimators, which have more accurate estimates than RDF-3X's default estimator, leads to better plan generation. In addition, observe that the box plot of estimators with the `max` aggregators are generally better than estimators that use the `min` or `avg` aggregator. This correlates with Figure 8, where we showed these estimators lead to more accurate estimations. We then performed a detailed analysis of the `max-hop-max` and `min-hop-min` estimator as representative of, respectively, the most and least accurate of the 9 estimators. We analyzed the queries in which plans under these estimators differed significantly. Specifically, we found 10 queries across both datasets

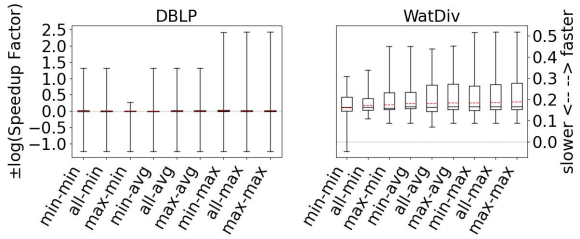


Figure 13: RDF-3X runtimes on Acyclic workload.

where the runtime differences were at least 1.15x. Of these, only in 1 of them `min-hop-min` lead to more efficient plans and by a factor of 1.21x. In the other 9, `max-hop-max` lead to more efficient plans, by a median of 2.05x and up to 276.3x, confirming our expectation that better estimations generally lead to better plans.

7 RELATED WORK

Within the scope of our work, we studied two classes of summary-based estimators, optimistic and pessimistic ones, in the context of graph database management systems. We showed these estimators can all be modeled as instances of picking paths on CEGs and experimentally studied how to make efficient estimates on the CEGs for optimistic estimators.

There is decades of extensive research on cardinality estimation techniques that can be used for estimating sizes of join queries. This literature has proposed several different classes of estimators, including other summary-based ones, as well as sampling-based ones and novel machine learning-based ones. Sampling-based estimators [8, 15, 18, 20, 22, 38, 43] sample input records from base tables and evaluate queries on these samples to make estimates. Sampling-based estimators are fundamentally different than summary-based ones as by increasing the sizes of the samples they can be arbitrarily accurate, while summary-based ones can be more accurate by keeping more statistics. Another class of estimators on which there is active work is machine-learning-based ones that learn to make estimates from example queries or predicates, e.g., example range predicates on columns [27, 41, 42]. Our goal in this paper is not to demonstrate that optimistic or pessimistic estimators can be more or less accurate than these other classes of estimators and a detailed comparison against them is beyond the scope of our work. In the following, we cover other summary-based estimators primarily focusing on graph-based database management systems.

Many relational systems, including commercial ones such as PostgreSQL, use summary-based estimators. Example summaries include the cardinalities of relations, the number of distinct values in columns, or histograms [3, 26, 34], wavelets [24], or probabilistic and statistical models [12, 37] that capture the distribution of values in columns. These statistics are used to estimate the selectivities of each join predicate, which are put together using several approaches, such as independence assumptions. In contrast, the estimators we studied store degree statistics about base relations and small-size joins (note that cardinalities are a form of degree statistics, e.g., $|R_i| = deg(\emptyset, \mathcal{R}_i)$).

Characteristic Sets (CS) [28] is a summary-based estimator primarily designed to estimate the cardinalities of stars in an RDF graph. It uses the so-called *characteristic set* of each vertex v in an RDF graph, which is the set of distinct outgoing labels v has. CS keeps statistics about the vertices with the same characteristic set. Then, using these statistics, CS makes estimates for the sizes of star queries. For a non-star query Q, Q is decomposed into multiple stars $s_1, ..., s_k$ in a greedy manner, by removing the largest stars first, and the estimates for each s_i is multiplied. However, unlike the optimistic estimators we considered, this decomposition procedure does not lead to multiple possible decompositions.

Several works have proposed summary-based estimators that compute a sketch of an input graph. SumRDF [36] builds a summary graph S of an RDF graph and adopts a holistic approach to make an estimate. Given the summary S, SumRDF considers all possible RDF graphs G that could have the same summary S. In the context of estimating the selectivities of path expressions, XSeed [46] and XSketch [32] build a sketch S of the input XML Document. The sketch of the graph effectively collapses multiple nodes and edges into supernodes and edges with metatadata on the nodes and edges. The metadata contains statistics, such as the number of nodes that was collapsed into a supernode. Then given a query Q, Q is matched on S and using the metadata, an estimate is made. Because these techniques do not decompose a query into smaller sub-queries, the question of which decomposition to use does not arise for these estimators either.

Several work use data structures that are adaptations of histograms from relational systems to store selectivities of paths or trees in XML documents. Examples include, *positional histograms* [44] and *Bloom histogram* [39]. These techniques do not consecutively make estimates for larger paths and have not been adopted to general subgraph queries. For example, instead of storing small-size paths in a data structure as in Markov tables, Bloom histograms store all paths but hashed in a bloom filter. Other work used similar summaries of XML documents (or its precursor the *object exchange model* [30] databases) for purposes other than cardinality estimation. For example, *TreeSketch* [33] produces a summary of large XML documents to provide approximate answers to queries.

8 CONCLUSIONS AND FUTURE WORK

Aside from capturing existing optimistic and pessimistic estimators, we believe the CEG framework can be the foundation to develop novel summary-based estimators. In addition to the two CEGs we considered here, CEG_O and CEG_M, other CEGs using different statistics can be defined and paired with different techniques to pick paths. To demonstrate an example, in the longer version of our paper [7] we describe another CEG we call CEG_{OCR} as a possible approach to remedy the overestimation problem of optimistic estimator on CEG_O for queries with large cycles. CEG_{OCR} uses new edge weights that capture the closing of large cycles between sub-queries. However, many other CEGs can naturally be defined and studied. For example, one can use variances or entropies of the distributions of small-size joins as edge weights, possibly along with degree statistics, and pick the lowest entropy paths. An important research direction is to systematically study a class of CEG instances that use different combination of statistics as edge weights, as well as techniques for picking paths, to design more accurate CEG-based estimators. In addition, in this work, we focused only on join-only queries and ignore other non-join predicates. An additional future work direction is a principled mechanisms to integrate filters on the queries that can be estimated using CEG.

REFERENCES

[1] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2016. Computing Join Queries with Functional Dependencies. In *PODS*.

[2] Ashraf Aboulnaga, Alaa R. Alameldeen, and Jeffrey F. Naughton. 2001. Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. In *VLDB*.

[3] Ashraf Aboulnaga and Surajit Chaudhuri. 1999. Self-Tuning Histograms: Building Histograms Without Looking at Data. In *SIGMOD*.

[4] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified Stress Testing of RDF Data Management Systems. In *ISWC*.

[5] A. Atserias, M. Grohe, and D. Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SICOMP* 42, 4 (2013).

[6] Walter Cai, Magdalena Balazinska, and Dan Suciu. 2019. Pessimistic Cardinality Estimation: Tighter Upper Bounds for Intermediate Join Cardinalities. In *SIGMOD*.

[7] Jeremy Chen, Yuqing Huang, Wang Mushi, Salihoglu Semih, and Salem Ken. 2022. *Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs https://cs.uwaterloo.ca/~ssalihog/papers/ceg-long.pdf*. Technical Report.

[8] Yu Chen and Ke Yi. 2020. Random Sampling and Size Estimation Over Cyclic Joins. In *ICDT*.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms* (2 ed.). The MIT Press.

[10] DBLP 2012. DBLP 2012-11-28 Dump. https://dblp.org/.

[11] Epinions 2003. Epinions. https://snap.stanford.edu/data/soc-Epinions1.html.

[12] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation Using Probabilistic Models. In *SIGMOD*.

[13] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *JACM* 59, 3 (2012).

[14] Graphflow 2019. Graphflow Source Code. https://tinyurl.com/wyuzb9pr.

[15] Haas, Peter J. and Naughton, Jeffrey F. and Seshadri, S. and Swami, Arun N. 1996. Selectivity and Cost Estimation for Joins Based on Random Sampling. *JCSS* 52, 3 (1996).

[16] Hetionet 2015. Hetionet v1.0. https://het.io/.

[17] Manas Joglekar and Christopher Ré. 2018. It's All a Matter of Degree - Using Degree Information to Optimize Multiway Joins. *TOCS* 62, 4 (2018).

[18] Kyoungmin Kim, Hyeonji Kim, George Fletcher, and Wook-Shin Han. 2021. Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation. In *SIGMOD*.

[19] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *PVLDB* 9, 3 (2015).

[20] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *CIDR*.

[21] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2018. Query Optimization Through the Looking Glass, and What We Found Running the Join Order Benchmark. *VLDBJ* 27, 5 (2018).

[22] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In *SIGMOD*.

[23] Angela Maduko, Kemafor Anyanwu, Amit Sheth, and Paul Schliekelman. 2008. Graph Summaries for Subgraph Frequency Estimation. In *ESWC*.

[24] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD*.

[25] Amine Mhedhbi and Semih Salihoglu. 2019. Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins. *PVLDB* 12, 11 (2019).

[26] M. Muralikrishna and David J. DeWitt. 1988. Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries. In *SIGMOD*.

[27] Parimarjan Negi, Ryan Marcus, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2020. Cost-Guided Cardinality Estimation: Focus Where it Matters. In *ICDEW*.

[28] Thomas Neumann and Guido Moerkotte. 2011. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *ICDE*.

[29] Thomas Neumann and Gerhard Weikum. 2008. RDF-3X: A RISC-Style Engine for RDF. *PVLDB* 1, 1 (2008).

[30] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. 1995. Object Exchange Across Heterogeneous Information Sources. In *ICDE*.

[31] Yeonsu Park, Seongyun Ko, Sourav S. Bhowmick, Kyoungmin Kim, Kijae Hong, and Wook-Shin Han. 2020. G-CARE: A Framework for Performance Benchmarking of Cardinality Estimation Techniques for Subgraph Matching. In *SIGMOD*.

[32] Neoklis Polyzotis and Minos Garofalakis. 2002. Statistical Synopses for Graph-Structured XML Databases. In *SIGMOD*.

[33] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. 2004. Approximate XML Query Answers. In *SIGMOD*.

[34] Viswanath Poosala and Yannis E. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *VLDB*.

[35] RDF3X 2020. RDF-3X Source Code. https://github.com/gh-rdf3x/gh-rdf3x/.

[36] Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev. 2018. Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation. In *WWW*.

[37] Wei Sun, Yibei Ling, Naphtali Rishe, and Yi Deng. 1993. An Instant and Accurate Size Estimation Method for Joins and Selections in a Retrieval-Intensive Environment. In *SIGMOD*.

[38] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P. Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. *PVLDB* 8, 12 (2015).

[39] Wei Wang, Haifeng Jiang, Hongjun Lu, and Jeffrey Xu Yu. 2004. Bloom Histogram: Path Selectivity Estimation for XML Data with Updates. In *VLDB*.

[40] WatDiv 2014. WatDiv v.0.6. https://dsg.uwaterloo.ca/watdiv/.

[41] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. 2019. Cardinality Estimation with Local Deep Learning Models. In *aiDM*.

[42] Lucas Woltmann, Dominik Olwig, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. 2021. PostCENN: PostgreSQL with Machine Learning Models for Cardinality Estimation. *PVLDB* 14, 12 (2021).

[43] Wentao Wu, Jeffrey F. Naughton, and Harneet Singh. 2016. Sampling-Based Query Re-Optimization. In *SIGMOD*.

[44] Yuqing Wu, Jignesh M. Patel, and H. V. Jagadish. 2002. Estimating Answer Sizes for XML Queries. In *EDBT*.

[45] YAGO 2008. YAGO 1. https://yago-knowledge.org/downloads/yago-1.

[46] Ning Zhang, M. Tamer Ozsu, Ashraf Aboulnaga, and Ihab F. Ilyas. 2006. XSEED: Accurate and Fast Cardinality Estimation for XPath Queries. In *ICDE*.