Using Pessimistic Paths in Cardinality Estimation Graphs to Combat Underestimation

Anonymous Author(s)

ABSTRACT

We study 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. We show that, as in the relational setting, optimistic estimators generally underestimate but are significantly more accurate than the pessimistic estimators, which avoid underestimation. We then address how to combat the underestimation in optimistic estimators. We begin by connecting these two classes of estimators by showing that they can be seen as different instantiations of a generic estimator that picks a bottom-to-top path in a *cardinality estimation graph* (CEG) as an estimate. We outline a space of heuristics to make an optimistic estimate for a query, each corresponding to a different bottom-to-top path in a CEG, and show that in general picking the most pessimistic, i.e., maximum-weight, path is an effective technique to combat underestimation for optimistic estimators. We show that on a large suite of datasets and workloads, the accuracy of such estimates is up to three orders of magnitude more accurate in mean q-error than some ad hoc heuristics that have been proposed in prior work. Using our CEG framework allows us to connect two disparate lines of work on optimistic and pessimistic estimators, adopt an optimization from pessimistic estimators to optimistic ones, and provide insights into the pessimistic estimators, such as simplifying one of them and showing the equivalence of two of them on acyclic queries.

1 INTRODUCTION

The problem of estimating the output size of a natural multi-join query (henceforth *join query* for short), is a fundamental problem that is solved in the query optimizers of database management systems when generating efficient query plans. This problem arises both in systems that manage relational data as well 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.

A prevalent technique used by existing systems to estimate cardinalities of joins is to use statistics about the base relations or outputs of small-size joins, combined with independence and uniformity assumptions to generate estimates for larger queries [2, 17, 20, 22]. We will refer to these as *optimistic estimators*. In the relational setting, it has been demonstrated that optimistic estimators tend to *underestimate* in practice, sometimes severely [14]. Other recent work - based on worst-case optimal join size bounds [1, 4, 5, 7, 9] - has led to the development of a class of *pessimistic estimators* that are guaranteed to *avoid* underestimation.

In this paper, we focus on the behaviors of optimistic and pessimistic estimators in the context of graph database systems, where the challenge is to estimate subgraph cardinalities. Our first contribution is an empirical study of both optimisitic and pessimistic estimators, using both real and synthetic datasets and a variety of query workloads. In this study, we show that optimistic estimators tend to underestimate subgraph cardinalities, as is the case in the relational setting. Similar to some recent work [26], we also show that the bounding estimates obtained from pessimistic estimators are typically very loose. That is, they often result in substantial overestimates, with error magnitudes significantly greater than those of optimistic underestimates.

With these observations in hand, we revisit the problem of underestimation by optimistic estimators. We begin by showing that the optimistic estimators and the new pessimistic estimators from references [5, 9] are different instantiations of a generic estimator that makes a sequence of estimations for growing subqueries to derive an estimate for the given query. Specifically, we show that both optimistic and pessimistic estimators can be seen as picking a bottom-to-top path in two different weighted *cardinality estimation graphs* (CEG), where the weights are maximum degree statistics in pessimistic estimators and a form of average degree statistics in optimistic ones. This allows us to unify two disparate bodies of work in a single framework. We observe that in estimators that can be represented with CEGs, there is often more than one way to generate an estimate for a query, corresponding to different bottom-to-up paths. In the case of pessimistic estimators, each of these paths is an upper bound, so a system should use the lowest estimate, which corresponds to using the minimum-weight path as the estimate. In contrast, this decision does not have a clear answer in the case of optimistic estimators. For example, consider the subgraph query in Figure 1. Given that we have the accurate cardinalities of all subqueries of size ≤ 2 available, there are 252 formulas (or bottom-to-up paths in the CEG of optimistic estimators) to estimate the cardinality of the query. Examples of these formulas are:

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

In previous work, the choice of which of these estimates to use has either been ad hoc or has been left unspecified. We describe a space of heuristics for making an estimate for optimistic estimators and show that an effective technique for combatting underestimation is to base the estimate on the *maximum-weight path* through the CEG. We show empirically that the accuracy of such estimates is very good.

We then review a refinement called a *bound sketch*, that has been used to improve the pessimistic estimator proposed by Cai, Balazinska, and Suciu [5]. This optimization partitions a query into

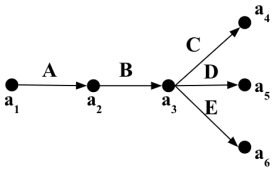


Figure 1: Example subgraph query Q_{5f}.

multiple subqueries and makes an estimate for each one to derive an estimate for the original query. Using our common CEG framework, we observe that this optimization can be directly applied to any estimator using a CEG, specifically to the optimistic estimators, and empirically demonstrate its benefits in some settings.

Finally, our use of the CEG representation of estimators provides some insight into proposed pessimistic estimators, which may be of independent interest to readers. We show that the MOLP estimator of Joglekar and Re [9] is exactly equivalent to the minimum-weight path in a CEG. This proves that MOLP has an alternative combinatorial solution than using a numeric linear program solver, and that MOLP can be simplified because some of its constraints are unnecessary. Using CEGs, we also provide several alternative combinatorial proofs to some known properties of MOLP, such as the theorem that MOLP is tighter than another bound called DBPLP proposed by the same authors. Using MOLP's CEG, we also prove that MOLP is at least as tight as the pessimistic estimator proposed by Cai et al [5] and are identical on acyclic queries over binary relations.

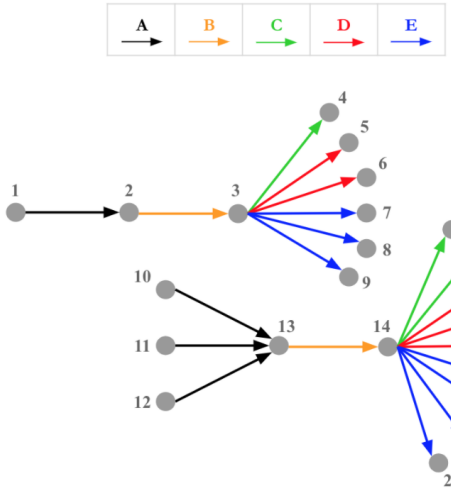
The remainder of this paper is structured as follows. Section 2 provides our query and database notation. Section 3 gives an overview of generic estimators that can be seen as picking paths from a CEG. Section 4 reviews optimistic estimators and their CEG, which we call CEG_O, and outlines the space of possible heuristics for making estimates using CEG_O. Section 5 reviews the pessimistic estimators, the CEG of the MOLP estimator, we call CEG_M, and the bound sketch refinement to pessimistic estimators. Using CEG_M, we prove several properties of MOLP and connect some of these pessimistic estimators. Section 6 presents our experiments evaluating optimistic and pessimistic estimators, the space of heuristics we outlined for optimistic estimators, and the bound sketch refinement to optimistic and pessimistic estimators. We also compare the optimistic estimators we covered against other summary-based and sampling-based techniques. Finally, Sections 7 and 8 cover related work and conclude, respectively.

2 QUERY AND DATABASE NOTATION

We consider conjunctive queries of the form

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

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, shown in black, orange, green, red, and blue. This graph can be represented using five binary relations, one for each of the edge labels. These relations are also shown in Figure 2.



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.

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). In Sections 4 and 5 we will define specific CEGs corresponding to different classes of estimators. However, all of these share a common structure for representing cardinality estimations. Specifically, a CEG for a query Q will consist 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 (and exponentiate the base of the logarithm) to compute the estimate.

Figure 3 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}. For example, the leftmost

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

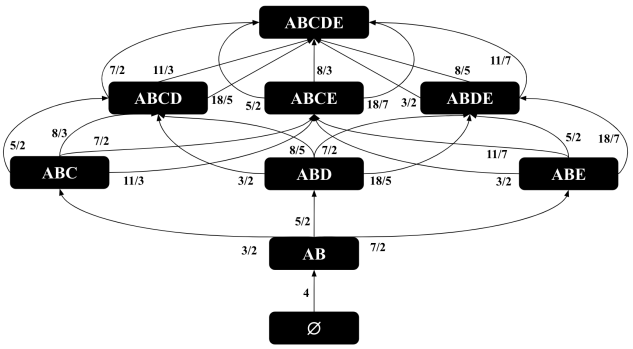


Figure 3: CEG_O for query Q_{5f} from Figure 1. The figure shows a subset of the vertices and edges of the full CEG.

path starts with $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$, then extends to $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3 \xrightarrow{C} a_4$, then to the subquery of 4-fork involving A, B, C, and D, and finally extends the 4-fork subquery to Q_{5f}. The first extension rate from $\emptyset$ to $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$ is simply the known cardinality of $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$, which is 4, and the second extension rate makes the uniformity assumption of $|a_2 \xrightarrow{B} a_3 \xrightarrow{C} a_4|/|a_2 \xrightarrow{B} a_3| = \frac{3}{2}$. The final estimate of this path is $4 \times \frac{3}{2} \times \frac{5}{2} \times \frac{7}{2} = 52.5$.

In Sections 4 and 5 we will show how some of the optimistic and pessimistic estimators from literature can be seen as instances of this generic estimator using different CEGs. We will also show that while it is clear that the minimum-weight bottom-to-top path should be the estimate chosen in the CEG of pessimistic estimators, it is not clear which path should be chosen from the CEG of optimistic estimators. We will argue that deliberately choosing the most pessimistic (i.e., maximum-weight) path from this space is an effective way to combat underestimation for optimistic estimators.

4 OPTIMISTIC ESTIMATORS

The estimators that we refer to as *optimistic* in this paper use known 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 [17] from RDF databases, and the graph catalogue estimator of the Graphflow system [20] for managing property graphs. These estimators are extensions of each other and use the statistics of the cardinalities of small-size joins. We give an overview of these estimators and then describe their CEGs, which we will refer to as CEG_O, and then describe a space of possible optimistic estimates that an optimistic estimator can make.

4.1 Overview

We begin by giving an overview of the Markov tables estimator [2], which was used to estimate the cardinalities of paths in XML documents. 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 predications 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 shown in Figure 2. The

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

Table 1: Example Markov table for h=2.

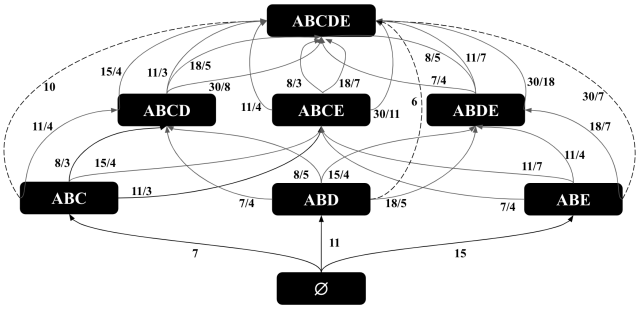


Figure 4: CEG_O for query Q_{5f} in Figure 1 when the Markov table contains up to 3-size joins (i.e., h = 3). Only a subset of the vertices and edges of the full CEG is shown.

formula to estimate a 3-path using a Markov table with $h = 2$ is to multiply the cardinality of the leftmost 2-path 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: $|\xrightarrow{A}\xrightarrow{B}| \times (|\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|)$. Observe that this formula is inspired by the Bayesian probability rule that $Pr(ABC) = Pr(AB)Pr(C|AB)$ but makes a conditional independence assumption between A and C, in which case the Bayesian formula would simplify to $Pr(ABC) = Pr(AB)Pr(C|B)$. For $Pr(AB)$ the formula uses the true cardinality $|\xrightarrow{A}\xrightarrow{B}|$. For $Pr(C|B)$ the formula makes a uniformity assumption that the number of C edges that each B edge extends to is equal for each B edge and is $r = |\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|$. Equivalently, this can be seen as an "average degree" assumption that on average the C-degree of nodes in the $\xrightarrow{B}\xrightarrow{C}$ paths is r. The result of this formula is $4 \times \frac{3}{2} = 6$, which underestimates the true cardinality of 7. The graph summaries [17] for RDF databases and the graph catalogue estimator [20] for property graphs have extended the contents of what is stored in Markov tables, respectively, to other acyclic joins, e.g., stars, and cyclic joins, e.g., triangles, but use the same uniformity and conditional independence assumptions.

4.2 Space of Possible Optimistic Estimators

We next represent such estimators using a CEG that we call CEG_O. This will help us describe the space of possible estimations that can be made with these estimators. 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 a join query 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.

In systems that adopt these optimistic estimators, the Markov table needs statistics about joins of size at least 2 to be able to make estimates. Consider the CEG_O constructed for the path query Q_{3p} from above. There would be only two $(\emptyset, Q)$ paths in this CEG. One corresponds to the estimate: $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}| \times (|\overset{B}{\rightarrow}\overset{C}{\rightarrow}| / |\overset{B}{\rightarrow}|)$. The other one corresponds to: $|\overset{B}{\rightarrow}\overset{C}{\rightarrow}| \times (|\overset{A}{\rightarrow}\overset{B}{\rightarrow}| / |\overset{B}{\rightarrow}|)$, which represents the estimate that each $\overset{B}{\rightarrow}\overset{C}{\rightarrow}$ tuple extends, on average, to $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}| / |\overset{B}{\rightarrow}|$ many A edges. Observe that for this query, both of these estimates are identical (the second formula simply swaps the positions of $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}|$ and $|\overset{B}{\rightarrow}\overset{C}{\rightarrow}|$ in the numerator. In fact, when $h = 2$ and Q is any path query (irrespective of the directions of the query edges), every $(\emptyset, Q)$ path in Q's CEG_O leads to exactly the same estimate.

However, when the query is not a path or when the Markov table contains joins of size ≥ 3, different $(\emptyset, Q)$ paths can lead to different estimates. Consider the CEG shown in Figure 3. There are 36 $(\emptyset, Q)$ paths leading to 7 different estimates. An example of these estimates are:

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

Similarly, consider the fork query Q_{5f} in Figure 1 and a Markov table that contains up to 3-size joins. The CEG of Q_{5f} is shown in Figure 4. Intuitively using the largest possible joins is advantageous, so we ignore edges that use 2-size joins in the numerator. However, there are still multiple paths in the CEG, leading to 2 different estimates:

- $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}| \times \frac{|\overset{C}{\leftarrow}\overset{D}{\rightarrow}|}{|\overset{C}{\rightarrow}|}$
- $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}| \times \frac{|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{D}{\rightarrow}|}{|\overset{A}{\rightarrow}\overset{B}{\rightarrow}|} \times \frac{|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{E}{\rightarrow}|}{|\overset{A}{\rightarrow}\overset{B}{\rightarrow}|}$

Both formulas start by using $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}|$. Then, the first "short-hop" formula makes one fewer conditional 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. We can expect this assumption to be less accurate than the two assumptions made in the long-hop estimate, which condition on 2-size joins. In general, these two formulas can lead to very different estimates.

For many queries, there can be many more than 2 different estimates. For such cases, it is unclear which of the estimates would lead to more accurate estimates in practice. Therefore, any optimistic estimator implementation needs to make decisions about which formulas to use, which corresponds to picking paths in CEG_O. We identify a space of heuristics that an optimistic estimator can adopt along two parameters:

- *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`).
- *Estimate aggregator:* Among the set of paths that are considered, each path gives an estimate. The estimator then has to aggregate these estimates to derive a final estimate, for which we identify three heuristics: (i) the largest estimated cardinality path (`max-aggr`); (ii) the lowest estimated cardinality path (`min-aggr`); or (iii) the average of the estimates among all paths (`avg-aggr`).

Any combination of these two heuristics can be used to design an optimistic estimator. In prior optimistic estimators, this decision has been either unspecified or chosen in an ad hoc manner. Specifically, the original Markov tables [2] chose the `max-hop` heuristic. In this work, each query was a path, so when the first heuristic is fixed, any path in CEG_O leads to the same estimate. Therefore an aggregator is not needed. Graph summaries [17] uses the `min-hop` heuristic and leaves the aggregator unspecified. Finally, graph catalogue [20] picks the `min-hop` and `min-aggr` aggregator. We will do a systematic empirical analysis of this space of estimators in Section 6. Although, it is unclear which of the path-length estimators would lead to more accurate estimations, given the well known problem that using independence and uniformity assumptions often lead to severe underestimations, intuitively, we expect the `max-aggr` heuristic, which picks the 'pessimistic' paths, to be an effective heuristic to combat underestimation.

5 PESSIMISTIC ESTIMATORS

Join cardinality estimation is directly related to the following fundamental question: Given a query Q and set of statistics over the relations R_i, such as their cardinalities or degree information about values in different columns, what is the worst-case output size of Q? Starting from the seminal result by Atserias, Grohe, and Marx in 2008 [4], several upper bounds have been provided to this question under different known statistics. For example the initial upper bound from reference [4], now called the *AGM bound*, used only the cardinalities of each relation, while later bounds, DBPLP [9], MOLP [9], 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. [5] in an actual estimator implementation. We refer to this as the CBS estimator, after the names of the authors. Here is the outline of this section:

- Section 5.1 reviews the MOLP bound and shows that similar to optimistic estimators, MOLP can also be seen as an estimator using a CEG. This allows us to provide several surprising insights into MOLP's properties.
- Sections 5.2 reviews the CBS estimator, which was originally described as using a subset of the inequalities of the CLLP inequalities. Using our CEG framework, we show that in fact the CBS estimator is equivalent to the MOLP bound on acyclic queries on which it was evaluated in reference [5]. We then review the bound sketch refinement of the CBS estimator from reference [5], which we show can also be applied to any estimator using a CEG, specifically the optimistic ones we cover in this paper.

To demonstrate another application of CEGs, Appendix D reviews the DBPLP bound and provides an alternative proof that MOLP is tighter than DBPLP. Specifically, we show that DBPLP bound is at least the weight of the maximum-weight path in CEG_M. Due to space constraints, in Appendix C we also review the CLLP bound, which is the tightest known pessimistic bound and evaluate its benefits over MOLP. CLLP extends MOLP with entropic sub-modularity constraints. Similarly, due to space constraints some of our proofs are provided in the appendix.

5.1 MOLP

MOLP was defined in reference [9] 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:

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 [9] has shown that 2^{m_A} is an upper bound on the size of Q. For example, in our running example, the optimal value of these inequalities is 96, which is an overestimate of the true cardinality of 78. It is not easy to directly see the solution of the MOLP on our running example. However, we will 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.

MOLP CEG (CEG_M): Let Q_Z be the projection of Q onto attributes Z, so $Q_Z = \Pi_Z Q$. Each variable s_Z in MOLP represents the maximum size of Q_Z, i.e., the tuples in the projection of Q_Z that contribute to the final output. We next interpret the two sets of inequalities in MOLP:

- *Extension Inequalities* $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i))$: 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. For example, in our running example, let X={a_2}, Y={a_2a_3} and E={a_1}. So both X and Y are subsets of $B(a_2, a_3)$. The inequality indicates that each a_1a_2 tuple, so an R_A tuple, can extend to at most $deg(\{a_2\}, \{a_2, a_3\}, B(a_2, a_3))$=$deg(a_2, B)$ $a_1a_2a_3$ tuples. This is true, because $deg(a_2, B)$ is the maximum degree of any a_2 value in B (in graph terms the maximum degree of any vertex with an outgoing B edge).
- *Projection Inequalities* $s_X \leq s_Y$ ($\forall X \subseteq Y$): These indicate that the number of tuples in Q_X is at most the number of Q_Y, if Y is a larger subquery.

With these interpretations we can now construct CEG_M.

- *Vertices:* For each $X \subseteq \mathcal{A}$ there is a vertex. This represents the subquery $\Pi_X Q$.
- *Extension Edges:* Add an edge with weight $\log(deg(X, Y, \mathcal{R}_i))$ between any $W_1 = X \cup E$ and $W_2 = Y \cup E$, for which there is $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i))$ inequality. Note that there can be multiple edges between W_1 and W_2 corresponding to inequalities from different relations.
- *Projection Edges:* $\forall X \subseteq Y$, add an edge with weight 0 from Y to X. These edges correspond to projection inequalities and intuitively indicate that, in the worst-case instances, $\Pi_Y Q$ is always as large as $\Pi_X Q$.

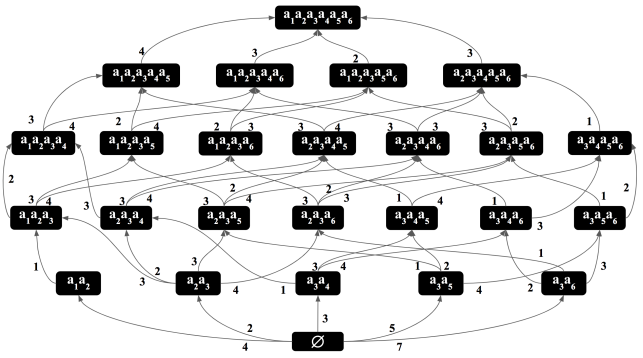


Figure 5: CEG_M for query Q_{5f} in Figure 1.

Figure 5 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 because $\Pi_{\mathcal{A}} Q = Q$.

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

Proof. Our proof consists of two steps. First we show that any feasible solution v to MOLP has a value at most the weight of any $(\emptyset, \mathcal{A})$ path. Then we show that a particular feasible solution, which we call v_{CEG}, is exactly the weight of the minimum-weight $(\emptyset, \mathcal{A})$ path. Let v be a feasible solution to the $MOLP_Q$. We refer to the value of v, so the value $s_{\mathcal{A}}$ takes in v, simply as $s_{\mathcal{A}}$. Let P be any $(\emptyset, \mathcal{A})$ path in CEG_M. Let $w(P)$ be the weight of P. Suppose w.l.o.g. that P=$(\emptyset) \xrightarrow{e_0} (E_1) ... (E_k) \xrightarrow{e_k} (E_{k+1} = \mathcal{A})$ and for the purpose of contradiction that $w(P) = w(e_0) + ... + w(e_k) < s_{\mathcal{A}}$. If this is the case, we can inductively (from i=$k+1$ down to 0) show that $w(e_0) + ... + w(e_{i-1}) < s_{E_i}$. The base case for $s_{E_{k+1}}$ holds by our assumption. Suppose $w(e_0) + ... + w(e_i) < s_{E_{i+1}}$ by induction hypothesis. Then consider the inequality in $MOLP_Q$ that corresponds

to the $(E_i) \xrightarrow{e_i} (E_{i+1})$ edge e_i. There are two possible cases for this inequality:
Case 1: e_i is a projection edge, so $w(e_i) = 0$ and we have an inequality of $s_{E_{i+1}} \leq s_{E_i}$, so $w(e_0) + ... + w(e_i) < s_{E_i+1} \leq s_{E_i}$, so $w(e_0) + ... + w(e_{i-1}) < s_{E_i}$.
Case 2: e_i is an extension edge, so we have an inequality of $s_{E_{i+1}} \leq s_{E_i} + w(e_i)$, so $w(e_0) + ... + w(e_i) < s_{E_i+1} \leq s_{E_i} + w(e_i)$, so $w(e_0) + ... + w(e_i) < s_{E_i}$, completing the inductive proof. However, this implies that $0 < s_\emptyset$, which contradicts the first inequality of MOLP, completing the proof that any feasible solution v to the MOLP is at most the weight of any $(\emptyset, \mathcal{A})$ path in CEG_M.

Next, let v_{CEG} be an assignment of variables that sets each s_X to the weight of the minimum-weight $(\emptyset, X)$ path in CEG_M. Let v_X be the value of s_X in v_{CEG}. We show that v_{CEG} is a feasible solution to $MOLP_Q$. First, note that in v_{CEG} $s_\emptyset$ is assigned a value of 0, so the first inequality of MOLP holds. Second, consider any extension inequality $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i))$, so CEG_M contains an edge from $X \cup E$ to $Y \cup E$ with weight $\log(deg(X, Y, \mathcal{R}_i))$. By definition of minimum-weight paths, $v_{Y \cup E} \leq v_{X \cup E} + \log(deg(X, Y, \mathcal{R}_i))$. Therefore, in v_{CEG} all of the extension inequalities hold. Finally, consider a projection inequality $s_X \leq s_Y$, where $X \subseteq Y$, so CEG_M contains an edge from vertex Y to vertex X with weight 0. By definition of minimum-weight paths, $v_X \leq v_Y + 0$, so all of these inequalities also hold. Therefore, v_{CEG} is indeed a feasible solution to $MOLP_Q$. Since any solution to MOLP has a value smaller than the weight of any path in CEG_M, we can conclude that $v_\mathcal{A}$ in v_{CEG}, which is the minimum-weight $(\emptyset, \mathcal{A})$ path, is equal to m_A. □

With this connection, readers can verify that the MOLP bound in our running example is 96 by inspecting the paths in Figure 5. In this CEG, the minimum-weight $(\emptyset, \mathcal{A})$ path has a length of 96 (specifically $\log_2(96)$), corresponding to the leftmost path in Figure 5. We make three observations.
Observation 1: Similar to the CEGs for optimistic estimators, 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. For example, the rightmost path in Figure 5 indicates that there are 7 a_3a_6's (so $a_3 \xrightarrow{E} a_6$ edges), each of which extends to at most 3 $a_3a_5a_6$'s (so $a_5 \xleftarrow{D} a_3 \xrightarrow{E} a_6$ edges), and so forth. This path yields 7x3x2x1x3=126 many possible outputs. 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. This is an alternative simple proof to the following proposition:

Proposition 5.1 (Prop. 2 [9]). *Let Q be a join query and OUT be the output size of Q, then $OUT \leq 2^{m_\mathcal{A}}$.*[2]

Proof. 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 algorithm, e.g., Dijkstra's algorithm, instead of a numeric LP solver.

[2]This is a slight variant of Prop. 2 from reference [9], which state that another bound, called the *MO bound*, which adds a preprocessing step to MOLP, is an upper bound of OUT.

Observation 3: Theorem 5.1 implies that we can simplify MOLP by removing the projection inequalities, which correspond to the edges with weight 0 in CEG_M. To observe this, consider any $(\emptyset, \mathcal{A})$ path $P = (\emptyset) \xrightarrow{e_0} (E_1)...(E_k) \xrightarrow{e_k} (\mathcal{A})$ and consider its first projection edge, say e_i. In Appendix A, we show that we can remove e_i and construct an alternative path with at most the same weight as P but with one fewer projection edge, showing that MOLP linear program can be simplified by only using the extension inequalities.

5.1.1 Using Degree Statistics of Small-Size Joins. MOLP can directly integrate the degree statistics from results of small-size joins. For example, if a system knows the size of $Q_{RS} = R(a_1, a_2) \bowtie S(a_2, a_3)$, then the MOLP can include the inequality that $s_{a_1a_2a_3} \leq \log(|Q_{RS}|)$. Similarly, the extension inequalities can use the degree information from Q_{RS} simply by taking the output of Q_{RS} as an additional relation in the query with three attributes a_1, a_2, and a_3. When comparing the accuracy of the MOLP bound with optimistic estimators, we will ensure that MOLP uses the degree information of the same small-size joins as optimistic estimators, ensuring that MOLP uses a strict superset of the statistics used by optimistic estimators.

5.2 CBS and Bound Sketch Optimization

We review the CBS estimator very briefly and refer the reader to reference [5] for details. CBS estimator has two subroutines *Bound Formula Generator (BFG)* and *Feasible Coverage Generator (FCG)* (Algorithms 1 and 2 in reference [5]) that, given a query Q and the degree statistics about Q, generate a subset of the CLLP 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 that can be used as an upper bound on the size of a query. In Appendix B, we show using our CEG framework that in fact the MOLP bound 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, which are the queries which reference [5] used. Therefore BFG and FCG can be seen as a method for solving the MOLP linear program on acyclic queries over binary relations, albeit in a brute force manner by enumerating all paths in CEG_M. We do this by showing that each path in CEG_M corresponds to a bounding formula and vice versa. These observations allow us to connect two worst-case upper bounds from literature using CEGs. Henceforth, we do not differentiate between MOLP and the CBS estimator.

5.2.1 Bound Sketch. We next review an optimization that was described in reference [5] to improve the MOLP bound. 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, specifically their sum is guaranteed to be more accurate than making a direct estimate for Q. We give an overview of the optimization here and refer the reader to reference [5] for details.

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 [5]), let S be the join attributes that are not extension attributes through a bounded edge. 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 5, 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 Opt. 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. The estimator can then sum the estimates for each subquery to generate an estimate for Q. Specifically, we can use a bound sketch to refine optimistic estimators and we will evaluate its benefits in Section 6.3. Intuitively, one advantage of using a bound sketch is that the tuples that hash to different buckets of the join attributes are guaranteed to not produce outputs and they never appear in the same subquery. This can make the uniformity assumptions in the optimistic estimators more accurate because two tuples that hashed to the same bucket of an attribute are more likely to join. It is interesting to note that unlike pessimistic estimators, there is no guarantee that accuracy will always improve when using bound sketch because in principle hashing can make degrees also less uniform.

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. For example, for Q_{5f}, one of the formulas that is needed is: $|a_1 \xrightarrow{A_0} a_2 \xrightarrow{B_{00}} a_3| \frac{|a_2 \xrightarrow{B_{00}} a_3 \xrightarrow{C_0} a_4|}{|a_2 \xrightarrow{B_{00}} a_3|} \frac{|a_4 \xleftarrow{C_0} a_3 \xrightarrow{D_0} a_5|}{|a_3 \xrightarrow{C_0} a_4|} \frac{|a_5 \xleftarrow{D_0} a_3 \xrightarrow{E_0} a_6|}{a_3 \xrightarrow{D_0} a_5}$, so we ensure that our Markov table has these necessary statistics.

6 EVALUATION

We next present our experiments that aim to answer four questions: (1) Which heuristic out of the space we identified in Section 4.2 leads to better accuracy for optimistic estimators? Related to this question, is choosing the maximum-weight paths in CEG_O an efficient way to combat underestimation? (2) How much does the bound-sketch optimization improve the optimistic estimators' accuracy? (3) How do optimistic and pessimistic estimators, which are both summary-based estimators, compare against each other and other baseline summary-based estimators from literature? (4) How does the best-performing optimistic estimator compare against the state-of-the-art sampling-based estimator called Wander Join [15, 26]? We also evaluated the benefits of using the CLLP estimator, which adds sub-modularity constraints on MOLP, which was not evaluated in references [5, 26]. Due to space constraints we present these experiments in Appendix C. We show that with the exception of one dataset and workload combination, the sub-modularity constraints have small accuracy benefits in our experiments. Throughout this section, except in our first experiments, where we set $h = 3$, we use a Markov table of size $h = 2$ for optimistic estimators [3]. Our code, datasets, and queries, are publicly available at https://github.com/submissionsigmod/CEExperimentsSIGMOD and https://github.com/submissionsigmod/gcare.

6.1 Setup, Datasets and Workloads

For all our experiments, we use a single machine that has two Intel E5-2670 at 2.6GHz CPUs, each with 8 physical and 16 logical cores, and 512 GB of RAM. We represent our datasets as labeled graphs and queries as edge-labeled subgraph queries but our datasets and queries can equivalently be represented as relational tables, one for each edge label, and SQL queries. We focused on edge-labeled queries for simplicity. Estimating queries with vertex labels can be done in a straightforward manner both for optimistic and pessimistic estimators e.g., by extending Markov table entries to have vertex labels as was done in reference [20]. Several prior work [12, 14] on cardinality estimation, including reference [5] has focused primarily on the IMDb dataset and the *Join Order Benchmark* (JOB) query workload (both reviewed momentarily). We use the IMDb and JOB workload, a dataset and workload from the G-CARE benchmark [26], and several new datasets and workloads consisting of acyclic and cyclic queries of various sizes.

6.1.1 Datasets. We used 6 real-world datasets: (i) IMDb: a movie dataset; (ii) YAGO: knowlege graph; (iii) Hetionet: a biological network; (iv) DBLP: a real knowledge graph; (v) WatDiv: a synthetic knowlege graph; and (vi) Epinions: a real-world social network graph. Except for IMDb, all of our datasets are by default in graph formats. IMDb is a relational database, which we converted into a property graph. Epinions 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 guaranteed to not have any correlations between edge labels. For reference our datasets are summarized in Table 2. We next describe how we converted the IMDb dataset into property graph.

IMDb: IMDb contains three groups of tables: (i) *entity tables* representing entities, such as actors (e.g., `name` table), movies, and

[3] We generated workload-specific Markov tables, which required less than 0.2MB memory for any workload-dataset combination for $h = 2$ or $h = 3$.

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

Table 2: Dataset descriptions.

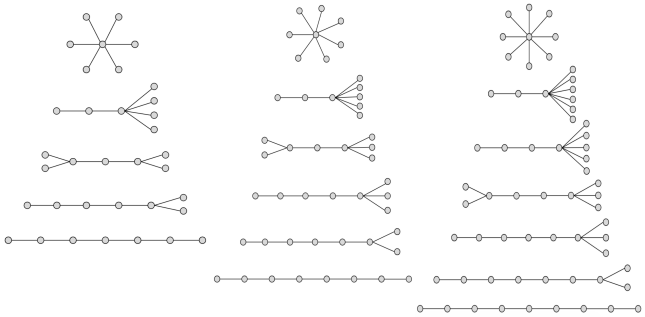


Figure 6: Our full acyclic query templates. The directions of the edges are neglected in the figure.

companies; (ii) *relationship tables* representing many to many relationships between the entities (e.g., `movie_companies` table represents relationships between movies and companies); and (iii) *type tables* that 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. We ignored vertex types because many queries in the JOB workload have no predicates on entity types. 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; (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.

6.1.2 Query Workloads. We used four workloads:

JOB: We transformed the JOB query workload from reference [14] into equivalent subgraph queries on our transformed graph. We removed non-join predicates in the queries, since we are focusing on join cardinality estimations and encoded the equality predicates on types directly on edge labels. We obtained 33 join query templates, whose subgraph versions included four 3-edge, four 4-edge, two 5-edge, one 6-edge query templates (we removed 2-edge query templates). Each of these queries were acyclic. We generated up to 100 query instances from each template by putting one edge label uniformly at random on each edge, ensuring that the output of the query was non-empty. The final workload contained 609 specific query instances. We use this workload only on the IMDb dataset [5].

Acyclic Query Workloads: We used two other acyclic query workloads. First, we took the acyclic query workload that was used in the G-CARE benchmark. This set contains query templates with 3-, 6-, 9-, and 12-edge star and path queries, and randomly generated trees. We will refer to this workload as `G-CARE`. Because the `G-CARE` workload does not ensure that every possible query depth is covered and different query templates may generate different number of query instances, we also created a more controlled workload of acyclic queries containing 6-, 7-, or 8-edge templates. We ensured that for each query size k, we had patterns of every possible depth. Specifically for any k, the minimum depth of any query is 2 (stars) and the maximum is k (paths). For each depth d in between, we picked a specific pattern. Our full patterns are shown in Figure 6. 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. We will refer to this workload as `Acyclic`.

Cyclic: We generated a workload of cyclic queries from templates used in prior work: a 5-edge diamond with a crossing edge [20], 6-edge two triangles [20], and a 5-edge lollipop query [24]. We generated 20 instances for each template.

6.2 Space of Optimistic Estimators

In our first set of experiments, we answer the question: *Which optimistic estimator in the space we defined leads to more accurate estimates?* We ran two sets of experiments. First, we used the JOB workload on the IMDb dataset and the `Acyclic` workload on DBLP, Hetionet, WatDiv, and Epinions. Then, we used the `Cyclic` workload on each of these datasets. This was to ensure that our observations are not affected by the cyclicity of queries. In order to set up an experiment in which we could test all of the 9 possible optimistic estimators, we used a Markov table that contained up to 3-size joins (i.e., h=3), which also contained all the necessary triangle subqueries. The default Markov table that contains up to 2-size joins does not allow us to test different estimators based on different path-length heuristics as each path in CEG_O has the same length, which is equal to the number of edges in the query minus 1. Also observe that when $h = 2$, we also cannot estimate cyclic queries because each entry in the Markov table is a path, so an optimistic estimator can only estimate acyclic queries.

In order to compare 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.

Our results are shown in Figures 7 and 8. We make several observations. First, regardless of the path-length heuristic chosen, 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 is further more accurate than `min`. This is true across all acyclic and cyclic experiments and all datasets. For example, on IMDb and `JOB` workload, the `all-hops-min`, `all-hops-avg`, and `all-hops-max` estimators have mean q-errors (after removing top 10 percent outliers), respectively, of 6.49, 1.65, and 1.01. The

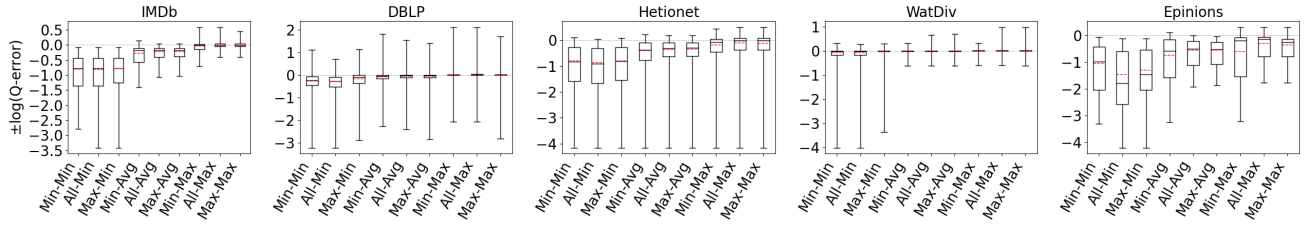


Figure 7: Evaluation of the space of optimistic estimators on JOB and Acyclic workloads.

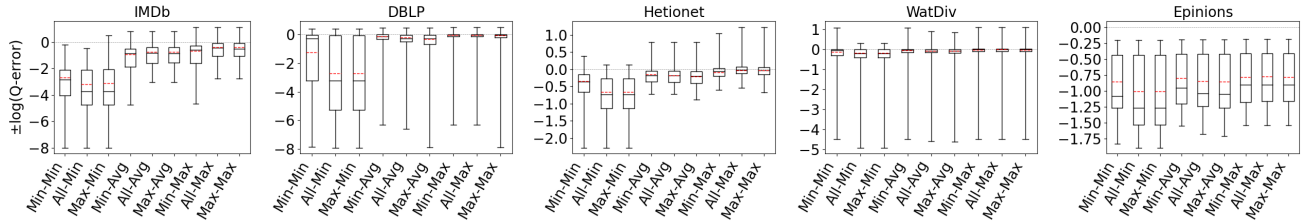


Figure 8: Evaluation of the space of optimistic estimators on Cyclic workload.

differences are larger on the Cyclic workload, where their q-errors, respectively, are 1967.62, 5.54, and 2.36. Therefore, using the pessimistic of the optimistic estimates leads to significantly more accurate estimations in our evaluations than the ad hoc heuristics used in prior work (up to three orders of magnitude in mean accuracy). Reference [20] had provided an intuition for using the min aggregator in their optimistic estimator. The intuition was based on an extreme example: if one possible extension, i.e., path in CEG_O, gives an estimate of 0 (i.e., when one subquery has empty output), then the full query is guaranteed to be empty. Although this intuition holds in this extreme case, in our evaluations which contain queries with non-empty outputs, this intuition does not hold.

We next analyze the path-length heuristics. 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 effectively picks one of the max-hop paths in majority of the queries in our workloads. Since max-hop considers fewer paths, it is more efficient to implement than all-hops. We conclude that systems implementing the optimistic estimators we consider should use the max-hop-max estimator. Interestingly, for these experiments, we can conclude that making more conditional independence assumptions that make uniformity assumptions by conditioning on larger joins (what max-hop does) is a better heuristic than making fewer conditional independence assumptions that condition on smaller joins (what min-hop does). Recall that intuitively, making a uniformity assumption that conditions on larger joins are more likely to hold.

Finally, Table 3 reports the number of over- and under-estimations that max-hop-max, all-hops-max, all-hops-avg, and all-hops-min estimators make on the Acyclic and Cyclic workloads. all-hops-max and all-hops-min are, respectively, the most pessimistic and optimistic of the optimistic estimators. Observe that not only does using the max aggregator yield more accurate results, it also underestimates significantly fewer number of queries on our datasets. As we hypothesized in Section 4.2, using max aggregator can be effective in combatting underestimation. For example, on DBLP and Acyclic workload, all-hops-max overestimates on 75% of queries while all-hops-min overestimates on only 8% (and is significantly less accurate overall).

Acyclic				
Dataset	**max-max**	**all-max**	**all-avg**	**all-min**
IMDb	156/213	156/213	1/368	0/369
DBLP	211/98	233/76	88/231	26/293
Hetionet	137/183	148/172	32/288	10/310
WatDiv	212/104	236/80	96/224	42/275
Epinions	29/291	47/273	1/319	0/320

Cyclic				
Dataset	**max-max**	**all-max**	**all-avg**	**all-min**
IMDb	12/48	12/48	8/52	0/60
DBLP	17/41	18/40	12/46	8/50
Hetionet	25/35	28/32	9/51	3/57
WatDiv	22/38	22/38	13/47	11/49
Epinions	0/60	0/60	0/60	0/60

Table 3: Number of over-/under-estimations for Acyclic (or JOB for IMDb) and Cyclic workload. We abbreviate the estimator names by removing the hop(s) from the name (e.g., max-max stands for max-hop-max).

6.3 Effects of Bound Sketch

Our next set of experiments aim to answer: *How much does the bound-sketch optimization improve the optimistic estimators' accuracy?* This question was answered for the CBS pessimistic estimator in reference [5]. We reproduce the experiment for MOLP in our context as well. To answer this question, we tested the effects of

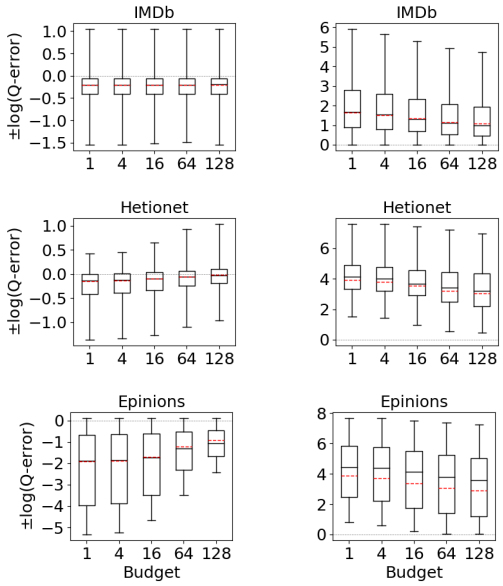


Figure 9: Effects of bound sketch on `max-hop-max` (left column) and MOLP (right column) estimators.

bound sketch on the `JOB` workload on IMDb and `Acyclic` workload on Hetionet and Epinions. We excluded WatDiv and DBLP as the `max-hop-max` estimator's estimates are already very close to perfect on these datasets and there is no room for significant improvement (recall Figure 7). Then we applied the bound sketch optimization to both `max-hop-max` and MOLP estimators and measured the q-errors of the estimators under partitioning budgets of 1 (no partitioning), 4, 16, 64, and 128.

Our results are shown in Figure 9. As demonstrated in reference [5], 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-hop-max` estimator though the results are data dependent. On Hetionet and Epinions, partitioning improves the mean accuracy at similar rates: by 25% and 89%, respectively. In contrast, we do not observe significant gains on IMDb. We note that the estimations of 68% of queries in Hetionet and 93% in Epinions strictly improved, so bound sketch is highly robust for optimistic estimators. We did not observe significant improvements in IMDb dataset for `max-hop-max`, although 93% of their q-errors strictly improve, albeit by a small amount.

6.4 Summary-based Estimator Comparison

The optimistic and pessimistic estimators we consider in this paper fall under summary-based estimators. Our next set of experiments compare `max-hop-max` and MOLP against each other and other baseline summary-based estimators. A recent work [26] has compared MOLP against two other summary-based estimators, Characteristic Sets (CS) [22] and SumRDF [30]. We reproduce and extend this comparison in our setting with our suggested `max-hop-max` optimistic estimator. We first give an overview of CS and SumRDF.

CS: CS is a cardinality estimator that was used in the RDF-3X system [23]. CS is primarily designed to estimate the cardinalities of stars in an RDF graph. The statistics it uses is 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, such as the number of nodes that belong to a characteristic set. Then, using these statistics, the estimator makes estimates for the number of distinct matches of stars. For a non-star query Q, Q is decomposed into multiple stars $s_1, ..., s_k$, then the estimate for each s_i is multiplied, which corresponds to an independence assumption.

SumRDF: SumRDF 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. Then, it returns the average cardinality of Q across all possible instances. This is effectively another form of uniformity assumption: each possible world has the same probability of representing the actual graph on which the estimate is being made. Note that the pessimistic estimators can also be seen as doing something similar, except they consider the most pessimistic of the possible worlds and return the cardinality of Q on that instance.

We measured the q-errors of `max-hop-max`, MOLP, CS, and SumRDF on the `JOB` workload on IMDb, the `Acyclic` workload on Hetionet, WatDiv, and Epinions, and the `G-CARE` workload on YAGO. We did not use bound sketch for MOLP and `max-hop-max`. However, we ensured that MOLP uses the cardinalities and degree information from 2-size joins, which ensures that the statistics MOLP uses is a strict superset of the statistics `max-hop-max` estimator uses.

Our results are shown in Figure 10. We omit CS from all figures except the first one, because it was not competitive with the rest of the estimators and even when y-axis is in logarithmic scale, plotting CS decreases the visibility of differences among the rest of the estimators. SumRDF timed out on several queries on YAGO and Hetionet and we removed those queries from `max-hop-max` and `MOLP`'s distributions as well. We make two observations. First, these results confirm the results from reference [26] that although MOLP does not underestimate, its estimates are very loose. Second, across all summary-based estimators, unequivocally, `max-hop-max` generates significantly more accurate estimations, often by several orders of magnitude in mean estimation. For example, on the IMDb and `JOB` workload, the mean q-errors of `max-hop-max`, SumRDF, MOLP, and CS are 1.8, 193.3, 44.6, and 333751, respectively. We also note that both CS and SumRDF perform underestimations in virtually all queries, whereas there are datasets, such as WatDiv and YAGO, where the majority of `max-hop-max`'s estimates are overestimations.

6.5 Comparison Against WanderJoin

Although we studied summary-based estimators in this paper, an alternative technique that has been studied is based on sampling. Sampling-based techniques are fundamentally different and based on using unbiased samplers of the query's output. As such, their primary advantage is that by enough sampling they are guaranteed to achieve estimations at any required accuracy. However, because they effectively perform the join on a sample of tuples, they can be slow, which is why they have seen little adoption in practice for

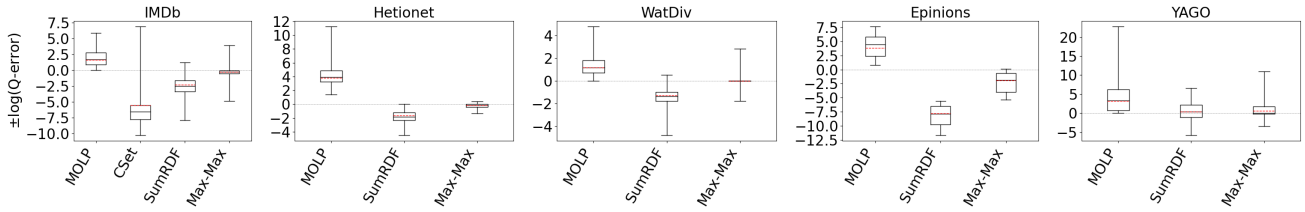


Figure 10: Summary-based estimation technique comparison.

join estimation. This time-vs-accuracy tradeoff is fundamentally different in summary-based estimators, which can give more accurate estimates only by storing more statistics, (e.g., larger join-sizes in Markov tables), so with more memory and not time. For completeness of our work, we compare `max-hop-max` estimator with WanderJoin (WJ) [15, 26], which is a sampling-based estimator that was identified in reference [26] as the most efficient sampling-based technique out of a set of techniques the authors experimented with. Our goal is to answer: *What is the sampling ratio at which WJ's estimate outperforms* `max-hop-max` *in accuracy and how do the estimation speeds of WJ and* `max-hop-max` *compare at this ratio?* We first give an overview of WJ as implemented in reference [26].

WanderJoin: WJ is similar to the index-based sampling described in reference [13]. Given a query Q, WJ picks one of the query edges e_q of Q to start the join from and picks a sampling ratio $r \in (0, 1]$, which is the fraction of edges that can match e_q that it will sample. For each sampled edge e_q^* (with replacement), WJ computes the join one query-edge at-a-time by picking one of the possible edges of a vertex that has already been matched uniformly at random. Then, if the join is successfully computed, a correction factor depending on the degrees of the nodes that were extended is applied to get an estimate for the number of output results that e_q^* would extend to. Finally, the sum of the estimates for each sample is multiplied by $1/r$ to get a final estimate.

We used the G-CARE's codebase [26]. We integrated the `max-hop-max` estimator into G-CARE and used the WJ code that was provided. We compared WJ and `max-hop-max` with sampling ratios 0.01%, 0.1%, 0.25%, 0.5%, and 0.75% on the `JOB` workload on IMDb, the `Acyclic` workload on Hetionet, WatDiv, and Epinions, and the `G-CARE` workload on YAGO. We ran both estimators five times (each run executes inside a single thread) and report the averages of their estimation times. We report the distribution of average q-error that WJ makes on each query. However, we can no longer present under- and over-estimations in our figures, as WJ might under and over-estimate for the same query across different runs.

The box-plot q-error distributions of `max-hop-max` and WJ are shown in Figure 11. We identify the sampling ratios in which the mean accuracy of WJ is better than the mean accuracy of `max-hop-max`, except in DBLP and Hetionet, where both `max-hop-max` and WJ's mean estimates are generally close to perfect, so we look at the sampling ratio where WJ's maximum q-errors are smaller than `max-hop-max`. We find that this sampling ratio on IMDb is 0.1%, on DBLP is 0.5%, on Hetionet is 0.75%, on Epinions is 0.5%, and on YAGO is 0.75%. However, the estimation time of WJ is between 15x and 212x slower, so one to two orders of magnitude, than `max-hop-max` except on our smallest dataset Epinions, where the difference is 1.95x. Observe that `max-hop-max`'s estimation times are very stable and consistently in sub-milliseconds, between 0.18ms and 0.54ms. This is because `max-hop-max`'s estimation time is independent of the dataset's size. In contrast, WJ's estimation times get slower as the datasets get larger, because WJ performs more joins. For example, at 0.25% ratio, while WJ takes 0.28ms on our smallest dataset Epinions, it takes 35.4ms on DBLP.

We emphasize that these comparisons are not perfect because it is difficult to compare distributions and these are two fundamentally different classes of estimators, providing systems with different tradeoffs. However, we believe that our 'competitive sampling ratio' analysis (more than runtime numbers) is instructive for interested readers.

7 RELATED WORK

There are decades of research on cardinality estimation of queries in the context of different database management systems. We cover a part of this literature focusing on work on graph-based database management systems, specifically XML and RDF and on relational systems. We covered Characteristic Sets [22], SumRDF [30], and WanderJoin [15, 26] in Section 6. We cover another technique, based on maximum entropy that can be used with any estimator that can return estimates for base tables and or small-size joins. We do not cover work that uses machine learning techniques to estimate cardinalities and refer the reader to several recent work in this space [11, 16, 34] for details of these techniques.

Other Summary-based Estimators: The estimators we studied in this paper fall under the category of summary-based estimators. 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, 21, 29], wavelets [19], or probabilistic and statistical models [6, 31] 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., $|R_i| = deg(\emptyset, \mathcal{R}_i)$).

Several estimators for subgraph queries have proposed summary-based estimators that compute a sketch of an input graph. SumRDF, which we covered in Section 6, falls under this category. In the context of estimating the selectivities of path expressions, XSeed [38] and XSketch [27] 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

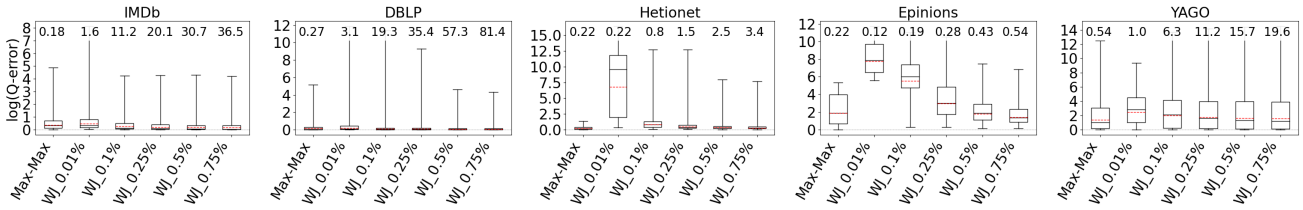


Figure 11: Comparison of `max-hop-max` and WJ, with average estimation times (in milliseconds) indicated at the top of boxes.

sub-queries, the question of which decomposition to use does not arise for these estimators.

Several work have used data structures that are adaptations of histograms from relational systems to store selectivities of path or tree queries in XML documents. Examples include, *positional histograms* [36] and *Bloom histogram* [33]. 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* [25] databases) for purposes other than cardinality estimation. For example, *TreeSketch* [28] produces a summary of large XML documents to provide approximate answers to queries.

Sampling-based Estimators: Another class of important estimators are based on sampling tuples [8, 13, 15, 32, 35]. These estimators either sample input records from base tables offline or during query optimization, and they evaluate queries on these samples to make estimates. Research has focused on different ways samples can be generated, such as independent or correlated sampling, or sampling through existing indexes. Wander Join, which we covered in Section 6 falls under this category. As we discussed, by increasing the sizes of the samples these estimators can be arbitrarily accurate but they are in general slower than summary-based ones because they actually perform the join on a sample of tuples. We are aware of systems [14] that use sampling-based estimators to estimate the selectivities of predicates in base tables but not on multiway joins. Finally, sampling-based estimators have also been used to estimate frequencies of subgraphs relative to each other to discover *motifs*, i.e. infrequently appearing subgrahs, [10].

The Maximum Entropy Estimator: Markl et al. [18] has proposed another approach to make estimates for conjunctive predicates, say $p_1 \wedge \ldots \wedge p_k$ given a set of ℓ selectivity estimates $s_{i_{11},\ldots,i_{1k}}$, $s_{i_{\ell 1},\ldots,i_{\ell k}}$, where $s_{i_{j1},\ldots,i_{jk}}$ is the selectivity estimate for predicate $p_{i_{j1}} \wedge \ldots \wedge p_{i_{jk}}$. Markl et al.'s maximum entropy approach takes these known selectivities and uses a constraint optimization problem to compute the distribution that maximizes the entropy of the joint distribution of the 2^k possible predicate space. Reference [18] has only evaluated the accuracy of this approach for estimating conjunctive predicates on base tables and not on joins, but they have briefly described how the same approach can be used to estimate the cardinalities of join queries. Multiway join queries can be modeled as estimating the selectivity of the full join predicate that contains the equality constraint of all attributes with the same name. The statistics that we considered in this paper can be translated to selectivities of each predicate. For example the size of $|(a_1) \xrightarrow{A} (a_2) \xrightarrow{B} (a_3)|$ can be modeled as $s_i = \frac{|(a_1)\xrightarrow{A}(a_2)\xrightarrow{B}(a_3)|}{|A||B|}$, as the join of A and B is by definition applying the predicate $A.src = B.dst$ predicate on the Cartesian product of relations A and B. This way, one can construct another optimistic estimator using the same statistics. We have not investigated the accuracy of this approach within the scope of this work and leave this to future work.

8 CONCLUSIONS AND FUTURE WORK

We studied the behaviors of optimistic and pessimistic estimators, which are summary-based estimators that use degree statistics about base relations and small-size joins in estimating the cardinalities of subgraph queries. Similar to some recent work [26], we showed that while pessimistic estimators are guaranteed to not underestimate by design, their estimates are very loose compared to optimistic ones, which can underestimate. We then addressed the problem of underestimation for optimistic estimators. We observed that both of these estimators are in fact specific instances of a more generic estimator that makes estimates by picking a path in a weighted CEG. We then showed that there is a space of possible paths that an estimator using a CEG can pick as an estimate and for optimistic estimators, an effective way to combat underestimation is to use the maximum-weight path, i.e., the most pessimistic one, in the CEG. When statistics about 3-size or larger joins are known, a more practical way is to use the maximum-weight path among only the paths with the maximum number of edges (hops). We also showed empirically that the estimates made by this heuristic is significantly more accurate compared to both pessimistic estimators and other summary-based estimators from literature. Our CEG framework has also allowed us to apply the bound sketch optimization for pessimistic estimators to optimistic ones and provide insights into the pessimistic estimators from prior literature, e.g., that MOLP estimator's linear program can be simplified and has a combinatorial solution.

The CEGs for optimistic and pessimistic estimators we focused on are only three specific examples of CEGs that has been used in prior work. By using different statistics or mixing statistics from different estimators, one can design a variety of other CEGs, which can be used to develop new estimators. For example, one can use the entropies of the distributions of small-size joins as edge weights in a CEG and pick the minimum-weight, i.e., "lowest entropy", paths, assuming that degrees are more regular in lower entropy edges. 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 heuristics on these CEGs for picking paths, to understand which statistics lead to more accurate results in practice.

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] A. Atserias, M. Grohe, and D. Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SICOMP* 42, 4 (2013).
[5] Walter Cai, Magdalena Balazinska, and Dan Suciu. 2019. Pessimistic Cardinality Estimation: Tighter Upper Bounds for Intermediate Join Cardinalities. In *SIGMOD*.
[6] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation Using Probabilistic Models. In *SIGMOD*.
[7] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *JACM* 59, 3 (2012).
[8] 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).
[9] Manas Joglekar and Christopher Ré. 2018. It's All a Matter of Degree - Using Degree Information to Optimize Multiway Joins. *TOCS* 62, 4 (2018).
[10] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. 2004. Efficient Sampling Algorithm for Estimating Subgraph Concentrations and Detecting Network Motifs. *Bioinformatics* 20, 11 (2004).
[11] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *CIDR*.
[12] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really?. In *VLDB*.
[13] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *CIDR*.
[14] 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).
[15] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In *SIGMOD*.
[16] Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, and Calisto Zuzarte. 2015. Cardinality Estimation Using Neural Networks. In *CASCON*.
[17] Angela Maduko, Kemafor Anyanwu, Amit Sheth, and Paul Schliekelman. 2008. Graph Summaries for Subgraph Frequency Estimation. In *ESWC*.
[18] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tam. 2007. Consistent Selectivity Estimation via Maximum Entropy. *VLDBJ* 16 (2007).
[19] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD*.
[20] Amine Mhedhbi and Semih Salihoglu. 2019. Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins. *PVLDB* 12, 11 (2019).
[21] M. Muralikrishna and David J. DeWitt. 1988. Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries. In *SIGMOD*.
[22] Thomas Neumann and Guido Moerkotte. 2011. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *ICDE*.
[23] Thomas Neumann and Gerhard Weikum. 2008. RDF-3X: A RISC-Style Engine for RDF. In *VLDB*.
[24] Dung Nguyen, Molham Aref, Martin Bravenboer, George Kollias, Hung Q. Ngo, Christopher Ré, and Atri Rudra. 2015. Join Processing for Graph Patterns: An Old Dog with New Tricks. In *GRADES*.
[25] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. 1995. Object Exchange Across Heterogeneous Information Sources. In *ICDE*.
[26] 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*.
[27] Neoklis Polyzotis and Minos Garofalakis. 2002. Statistical Synopses for Graph-Structured XML Databases. In *SIGMOD*.
[28] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. 2004. Approximate XML Query Answers. In *SIGMOD*.
[29] Viswanath Poosala and Yannis E. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *VLDB*.
[30] Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev. 2018. Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation. In *WWW*.
[31] 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*.
[32] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P. Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. In *VLDB*.
[33] Wei Wang, Haifeng Jiang, Hongjun Lu, and Jeffrey Xu Yu. 2004. Bloom Histogram: Path Selectivity Estimation for XML Data with Updates. In *VLDB*.
[34] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. 2019. Cardinality Estimation with Local Deep Learning Models. In *aiDM*.
[35] Wentao Wu, Jeffrey F. Naughton, and Harneet Singh. 2016. Sampling-Based Query Re-Optimization. In *SIGMOD*.
[36] Yuqing Wu, Jignesh M. Patel, and H. V. Jagadish. 2002. Estimating Answer Sizes for XML Queries. In *EDBT*.
[37] C. T. Yu and M. Z. Ozsoyoglu. 1979. An Algorithm for Tree-query Membership of a Distributed Query. In *COMPSAC*.
[38] Ning Zhang, M. Tamer Ozsu, Ashraf Aboulnaga, and Ihab F. Ilyas. 2006. XSEED: Accurate and Fast Cardinality Estimation for XPath Queries. In *ICDE*.

A PROJECTION INEQUALITIES CAN BE REMOVED FROM MOLP

Consider any $(\emptyset, \mathcal{A})$ path P=$(\emptyset)\xrightarrow{e_0}(E_1)...(E_k)\xrightarrow{e_k}(\mathcal{A})$ and consider its first projection edge, say e_i. We show that we can remove e_i and replace the rest of the edges e_{i+1} to e_k with (possibly) new edges e'_{i+1} to e'_k with exactly the same weights and construct P'=$(\emptyset)\xrightarrow{e_0}(E_1)...\xrightarrow{e_{i-1}}(E_i)\xrightarrow{e'_{i+1}}(E'_{i+2})$ ....$(E'_k)\xrightarrow{e'_k}(\mathcal{A})$, where $E'_{i+2} \supseteq E_{i+2}$, ..., $E'_k \supseteq E_k$. This can be seen inductively as follows. We know $E_i \supseteq E_{i+1}$ because e_i is a projection edge. Then if $(E_{i+1})\xrightarrow{e_{i+1}}(E_{i+2})$ edge was an extension edge that extended E_{i+1} to $N_{i+1} = E_{i+2} \setminus E_{i+1}$ attributes, then by construction, there is another edge $(E_i)\xrightarrow{e'_{i+1}}(E'_{i+2} = E_i \cup N_i)$ in CEG_M with the same weight as e_{i+1}. If instead e_{i+1} was a projection edge that removed a set of attributes from E_{i+1}, similarly there is another projection edge e'_{i+1} that removes the same set of attributes from E_i. So inductively, we can find an alternative sub-path from E_{i+1} to $\mathcal{A}$, $(E_{i+1})\xrightarrow{e'_{i+1}}...\xrightarrow{e'_k}(\mathcal{A})$ with the same length as the sub-path $(E_{i+1})\xrightarrow{e_{i+1}}...\xrightarrow{e_k}(\mathcal{A})$.

B CBS ESTIMATOR'S CONNECTION TO MOLP ON ACYCLIC QUERIES

We first show that on acyclic queries MOLP is at least as tight as the CBS estimator. Our proof is based on showing that for each bounding formula generated by BFG and FCG (respectively, Algorithms 1 and 2 in reference [5]), there is a path in $MOLP_C$. For a detailed overview of these algorithms, we refer the reader to reference [5]. We then show that if the queries are further on binary relations, then the standard MOLP bound, which only contains degree statistics about subsets of attributes in each relation, is exactly equal to the CBS estimator.

For each bounding formula generated by BFG and FCG, there is a path in CEG_M: Let C be a coverage combination enumerated by FCG. Consider a bounding formula F_C. We can represent F_C as a set of (R_i, X_i) triples, where $X_i \subseteq \mathcal{A}_i$ is the set of attributes that R_i covers and is of size either 0, $|\mathcal{A}_i|-1$, or $|\mathcal{A}_i|$. Let $Y_i = \mathcal{A}_i \setminus X_i$. Then the bounding formula generated for F_C will have exactly cost $\sum_i \log deg(Y_i, R_i)$ (recall that $deg(Y_i, R_i) = deg(Y_i, \mathcal{A}_i, R_i)$). This is because there are 3 cases: (i) if $|X_i|= 0$, then the BFG ignores R_i and $deg(Y_i, R_i) = 0$; (ii) if $|X_i|= |\mathcal{A}_i - 1|$, then BFG uses in its formula the degree of the single uncovered attribute a in $\mathcal{A}_i$, which is equal to $deg(Y_i, R_i)$ as Y_i only contains a; and (iii) if $|X_i|= |\mathcal{A}_i|$, then BFG uses $|R_i|$ in its formula, and since $Y_i = \emptyset$, $deg(Y_i, R_i) = |R_i|$.

We next show that CEG_M contains an $(\emptyset, \mathcal{A})$ path with exactly the same weight as the cost of F_C. We first argue that if Q is acyclic, then there must always be at least one relation R_i in the coverage

C, that covers all of its attributes. Assume for the purpose of contradiction that each relation $R_i(\mathcal{A}_i) \in Q$ either covers 0 attributes or $|\mathcal{A}_i|-1$ attributes. Then start with any $R_{i1}(\mathcal{A}_{i1})$ that covers $|A_{i1}|-1$ attributes. Let $a_{i1} \in \mathcal{A}_{i1}$ be the attribute not covered by R_{i1}. Then another relation $R_{i2}(\mathcal{A}_{i2})$ must be covering a_{i1} but not covering exactly one attribute $a_{i2} \in \mathcal{A}_{i2}$. Similarly a third relation R_{i3} must be covering a_{i2} but not covering another attribute $a_{i3} \in \mathcal{A}_{i3}$, etc. Since the query is finite, there must be a relation R_j that covers an a_{j-1} and whose other attributes are covered by some relation R_k, where $k < j$, which forms a cycle, contradicting our assumption that Q is acyclic.

We can finally construct our path in CEG_M. Let's divide the relations into $\mathcal{R}_C$, which cover all of their attributes, i.e., $\mathcal{R}_C = \{R_i(\mathcal{A}_i) | R_i$ covers $|\mathcal{A}_i|$ attributes$\}$, and $\mathcal{R}_E$, which cover all but one of their attributes, $\mathcal{R}_E = \{R_i(\mathcal{A}_i)$ s. t. R_i covers $|\mathcal{A}_i|-1$ attributes$\}$. We ignore the relations that do not cover any attributes. Let relation in $\mathcal{R}_C = R_{C1}(\mathcal{A}_{C1}), ..., R_{Ck}(\mathcal{A}_{Ck})$ and those in $\mathcal{R}_E = R_{E1}(\mathcal{A}_{E1}), ..., R_{Ek'}(\mathcal{A}_{Ek'})$. Then we can construct the following path. The first of the path uses the cardinalities or relations in $\mathcal{R}_C$ in an arbitrary order to extend $(\emptyset)$ to $U = (\cup_{i=1,...k} \mathcal{A}_{Ci})$. For example this path can be: $P_1 = (\emptyset) \xrightarrow{\log(|R_{C1}|} (\mathcal{A}_{C1}) \xrightarrow{\log(|R_{C2}|} (\mathcal{A}_{C1} \cup \mathcal{A}_{C2}) ... \xrightarrow{\log(|R_{Ck}|} (U)$. Now to extend U to $\mathcal{A}$, observe that for each uncovered attribute $T = \mathcal{A} \setminus U$, there must be some relation $R_{Ej}(\mathcal{A}_{Ej}) \in \mathcal{R}_E$, such that all of the $|\mathcal{A}_{Ej}|-1$ attributes are already bound in U. This is because $|T| = k'$ and if each R_{Ej} has at least 2 attributes in T, then Q must be cyclic. Note that this is true because by definition of acyclicity [37] any "ear" that we remove can be iteratively covered by at most one relation, which means by the time we remove the last ear, we must be left with a relation R_{Ej} and one attribute, which contradicts that R_{Ej} had at least 2 uncovered attributes in T. So we can iteratively extend the path P_1 with one attribute at a time with an edge of weight $\log(deg(Y_{Ej}, R_{Ej}))$ until we reach $\mathcal{A}$. Note that this path's length would be exactly the same as the cost of the bounding formula generated by BFG and FCG for the coverage C.

When relations of an acyclic query are binary the CBS estimator is equal to MOLP: Ideally we would want to prove that when relations are binary that any path in CEG_M corresponds to a bounding formula. However this is not true. Consider the simple join $R(A, B) \bowtie S(B, C)$. CEG_M contains the following path, $P = (\emptyset) \xrightarrow{\log deg(\{B\},\{B\},R} (\{B\}) \xrightarrow{\log deg(C,\{B,C\},S)} (\{B, C\}) \xrightarrow{\log deg(A,\{A,B\},R)} (\{A, B, C\})$. There is no bounding formula corresponding to this path in the CBS estimator because the first edge with weight $\log deg(\{B\}, \{B\}, R)$ uses the cardinality of projection of R. However, the CBS estimator does not use cardinalities of projections in its bounding formulas and only uses the cardinalities of relations. Instead, what can be shown is that if a path P in CEG_M uses the cardinalities of projections, then there is another path P' with at most the same length, for which the CBS estimator has a bounding formula.

First we show that given an acyclic query over binary relations, if a path P in CEG_M contains cardinalities of projections, then there is an alternative path P' that has at most the length of P and that contains at least one more edge that uses the cardinality of a full relation. We assume w.l.o.g. that Q is connected. Note that in P any edge from $(\emptyset)$ in CEG_M must be using the cardinality of a relation or a projection of a relation (the only outgoing edges from $(\emptyset)$ are these edges). Let us divide the edge in P into multiple groups: (i) *Card* are those edge that extend a subquery with two attributes and use the cardinality of a relation; (ii) $Ext - Card$ are those edges that bound an attribute in a relation R_i in *Card* and extend a subquery to one new attribute a using the degree of a in R_i; (iii) *Proj* are those edges that extend a subquery by a single attribute a, without bounding any attributes in the subquery, i.e., using the cardinality of the projection of a relation R_i onto a (so the weight of these edges look like $\log(deg(\{a\}, \{a\}, R_i))$; and (iv) *Rem* are the remaining edges that extend a subquery by one attribute either bounding another attribute in *Proj* or some other attribute in *Rem*.

We first note that we can assume w.l.o.g., that if any relation $R_i(a_1, a_2)$ is used in an edge $e_p \in Proj$ to extend to, say, a_2, then a_1 cannot be an attribute covered by the edges in *Card* or *Ext*. Otherwise we can always replace e_p, which has weight $\log(\Pi_{a_2} R_i)$ with an edge we can classify as *Ext* with weight $\log(deg(a_2, \{a_1, a_2\}, R_i))$ because $|\Pi_{a_2} R_i| \geq deg(a_2, \{a_1, a_2\}, R_i)$. Next, we argue that we can iteratively remove two edges from *Proj* and possibly *Rem* and instead add one edge to *Card* without increasing the length of P. First observe that if *Rem* is empty, we must have a relation $R_i(a_1, a_2)$ whose both attributes are in set *Proj*, in which case, we can remove these two edges and replace with a single *Card* edge that simply has weight $\log(\emptyset, \{a_1, a_2\}, R_i)$ and reduce P's length because $|R_i| \leq \Pi_{a_1} R_i \times \Pi_{a_2} R_i$. Note that if *Rem* is not empty, then at least one of the edges e_r must be bounding an attribute a_1 and extending to a_2 using a relation R_p, where a_1 must be extended by an edge e_p in *Proj* using the same relation R_p. Otherwise there would be some edge e in *Rem* that extended a subquery W_1 to $W_1 \cup \{a_j\}$ without bounding the attribute that appears in the weight of e. This is because note that if we remove the relations that were used in the edges in *Card* and *Ext* then we would be left with an acyclic query, so have t relations and $t + 1$ attributes that need to be covered by *Proj* and *Rem*. If no edge e_r is bounding an attribute a_i in *Proj*, then one of the t relations must appear twice in the edges in *Rem*, which cannot happen because the relations are binary (i.e., this would imply that the attributes of a relation $R_x(a_{x1}, a_{x2})$ were covered with two edges with weights $\log(deg(\{a_{x1}\}, \{a_{x1}, a_{x2}\}, R_x))$ and $\log(deg(\{a_{x2}\}, \{a_{x1}, a_{x2}\}, R_x))$, which cannot happen). Therefore such an e_r and e_p must exist. Then we can again remove them and replace with one edge to *Card* with weight $\log(deg(\emptyset, \{a_1, a_2\}, R_p))$ and decrease the weight of P. Therefore from any P we can construct a P' whose length is at most P and that only consist of edges *Card* and *Ext*. Readers can verify that each such path P' directly corresponds to a bounding formula generated by BFG and FCG (each relation R_i used by an edge in *Card* and *Ext*, respectively corresponds to a relation covering exactly $|\mathcal{A}_i|$ and $|\mathcal{A}_i|-1$ attributes).

C CLLP AND EFFECTS OF SUB-MODULARITY CONSTRAINTS

CLLP is the tightest of the known worst-case optimal bounds and extends MOLP with sub-modularity constraints (also known as the *Shannon inequalities*):

$$s_{X \cup Y} + s_{X \cap Y} \leq s_X + s_Y, \ \forall X, Y$$

The effects of sub-modularity constraints on MOLP (or CBS estimator) have not been evaluated in prior work. We next provide a set of experiments to address the question: *How much do the submodularity constraints improve the accuracy of the MOLP estimator?* With the addition of sub-modularity constraints, we cannot map the solution of CLLP to a CEG. Therefore, we solved the CLLP estimator with a numerical solver. We ran CLLP on the same workloads we had used to compare the `max-hop-max` and MOLP. Our results are shown in Figures 12 and 13. Overall, we see small improvement in accuracy except in Hetionet, where we see 36% improvement in the mean accuracy of using 128 partitions compared to 1 partition. This observation is important as it provides evidence that systems using a strictly pessimistic estimator might prefer using a fast combintorial solver for MOLP (using CEG_M) instead of using a slow numerical solver for CLLP, which is a more complex linear program, as they should not expect significant improvements from the addition of sub-modularity constraints.

D DBPLP

We demonstrate another application of CEGs and provide alternative combinatorial proofs to some properties of DBPLP, which is another worst-case output size bound from reference [9]. DBPLP is not as tight as MOLP (or CLLP), which our proofs demonstrate through a path-analysis of CEGs. We begin by reviewing the notion of a cover of a query.

Definition 1. *A cover C is a set of (R_j, A_j) pairs where $R_j \in \mathcal{R}$ and $A_j \in \mathcal{A}_j$, such that the union of A_j in C "cover" all of $\mathcal{A}$, i.e., $\cup_{(R_j,A_j)\in C} A_j = \mathcal{A}$.*

DBPLP of a query is defined for a given cover C as follows:

Minimize $\Sigma_{a\in\mathcal{A}} v_a$

$$\Sigma_{a\in A_j \setminus A'_j} v_a \geq \log(deg(A'_j, \Pi_{A_j} R_j)),\ \forall (R_j, A_j) \in C, A'_j \subseteq A_j$$

Note that unlike MOLP and CLLP, DBPLP is a maximization problem and has one variable for each attribute $a \in \mathcal{A}$ (instead of each subset of $\mathcal{A}$). Similar to the MOLP constraints, we can also provide an intuitive interpretation of the DBPLP constraints. For any (R_j, A_j) and $A'_j \subseteq A_j$, let $B = A_j \setminus A'_j$. Each constraint of the DBPLP indicates that the number of B's that any tuple that contains A'_j can extend to is $deg(A_j, \Pi_{A_j} R_j)$, which is the maximum degree of any A'_j value in $\Pi_{A_j} R_j$. Each constraint is therefore effectively an extension inequality using a maximum degree constraint. Based on this interpretation, we next define a DBPLP CEG, CEG_D, to provide insights into DBPLP.

DBPLP CEG (CEG_D):

- *Vertices:* For each $X \subseteq \mathcal{A}$ we have a vertex.
- *Extension Edges:* Add an edge with weight $deg(A'_j, \Pi_{A_j} R_j)$ between any W_1 and W_2, such that $A'_j \subseteq W_1$ and $W_2 = W_1 \cup (A_j \setminus A'_j)$.

We note that DBPLP is not the solution to some path in this CEG, but defining it allows to provide some insights into DBPLP. Observe that DBPLP and MOLP use the same degree information and the condition for an extension edge is the same. Therefore CEG_D contains the same set of vertices as CEG_M but a subset of the edges of CEG_M. For example CEG_D does not contain any of the projection edges of CEG_M. Similarly, CEG_D does not contain any edges that contain degree constraints that cannot be expressed in the cover C, because in the (R_j, A_j) pairs in C, A_j may not contain every attribute in $\mathcal{A}_j$. Consider our running example and the cover C=$\{(\{a_1, a_2\}, R_A), (\{a_3, a_4\}, R_C)\}$. The DBPLP would contain, 6 constraints, 3 for $(\{a_1, a_2\}, R_A)$ and 3 for $(\{a_3, a_4\}, R_C)$. For example one constraint would be that $v_{a_1} + v_{a_2} \geq \log(deg(\emptyset, \Pi_{\{a_1,a_2\}R_A}) = \log(|R_A|) = \log(4)$.

The following theorem provides insight into why DBPLP is looser than MOLP using CEG_D.

Theorem D.1. *Let P be any $(\emptyset, \mathcal{A})$ path in CEG_D of a query Q and cover C of Q. Let $d_{\mathcal{A}}$ be the solution to the DBPLP of Q. Then $d_{\mathcal{A}} \geq |P|$.*

Proof. We first need to show that there is always an $(\emptyset, \mathcal{A})$ path in CEG_D. We can see the existence of such a path as follows. Start with an arbitrary (R_j, A_j) pair in C, which has an inequality for $A'_j = A_j$ which leads to an $\emptyset \rightarrow A_j$ edge. Let $X = A_j$. Then take an arbitrary (R_i, A_i) such that $Z = A_i \setminus X \neq \emptyset$, which must exist because C is a cover. Then we can extend X to $Y = X \cup Z$, because by construction we added an $X \rightarrow Y$ edge for the constraint where $A'_i = A_i \setminus Z$ in DBPLP (so the constraint is $\Sigma_{a\in Z} v_a \geq \log(deg(A_i \setminus Z, \Pi_{A_i}(R_i))))$.

Now consider any $(\emptyset, \mathcal{A})$ path $P = \emptyset \xrightarrow{e0} X_0 \xrightarrow{e1} \ldots \xrightarrow{ek} \mathcal{A}$. Observe that by construction of CEG_D each edge e_i extends an X to $Y = X \cup Z$ and the weight of e_i comes from a constraint $\Sigma_{a\in Z} v_a \geq \log(deg(A_j \setminus Z, \Pi_{A_j} R_j))$ for some (R_j, A_j). Therefore the variables that are summed over each edge is disjoint and contain every variable. So we can conclude that $\Sigma_{a\in\mathcal{A}} \geq |P|$. In other words, each $(\emptyset, \mathcal{A})$ path identifies a subset of the constraints $c_1, \ldots, c_k$ that do not share the same variable v_a twice, so summing these constraints yields the constraint $\Sigma_{a\in\mathcal{A}} \geq |P|$. Therefore, any feasible solution v^* to DBPLP, in particular the optimal solution $d_{\mathcal{A}}$ has to have a value greater than $|P|$. □

Corollary D.1. *Let $m_{\mathcal{A}}$ and $d_{\mathcal{A}}$ be the solutions to MOLP and DBPLP, respectively. Then $m_{\mathcal{A}} \leq d_{\mathcal{A}}$ for any cover C used in DBPLP.*

Proof. Directly follows from Theorems 5.1 and D.1 and the observation that CEG_D contains the same vertices and a subset of the edges in CEG_M. □

Corollary D.1 is a variant of Theorem 2 from reference [9], which compares refinements of MOLP and DBPLP. Our proof serves as an alternative combinatorial proof to the inductive proof in reference [9] that compares the LPs of the bounds. Specifically, by representing MOLP and DBPLP bounds as CEGs and relating them, respectively, to the lengths of shortest and longest $(\emptyset, \mathcal{A})$ paths, one can directly observe that MOLP is a tighter than DBPLP.

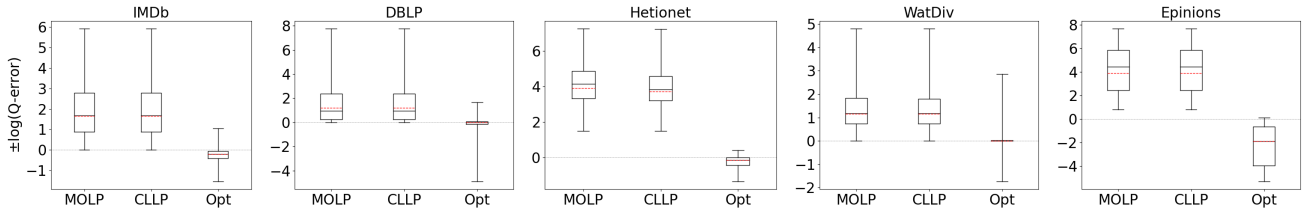


Figure 12: Q-error distribution of `max-hop-max` optimistic, MOLP, and CLLP on `JOB` on IMDb and `Acyclic` on the other datasets.

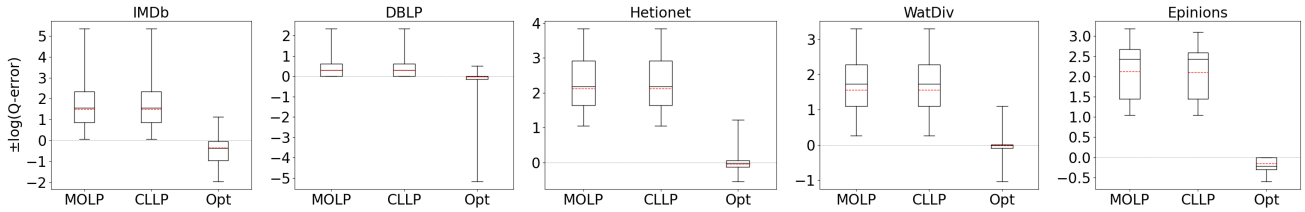


Figure 13: Q-error distribution of `max-hop-max` optimistic, MOLP, and CLLP on `Cyclic`.