Response to Reviewer Comments

We thank the reviewers for their comments. In the following sections, we address each comment of each reviewer. In the main text, we highlight the parts of text that we modified since our last submission in blue. We include copies of reviewers' comments below to make it easier for them to follow their comment and our responses. We also give pointers to the necessary sections to identify the main text changes that address reviewers' comments. To make space to address reviewers' comments, we shortened many paragraphs throughout the text. We do not highlight those paragraphs.

1 META REVIEWER

1.1 Writing and Presentation [R1 O1/O2, R2 O2, R3 O1]

R1 O1/O2
Introduction and motivation would benefit greatly from a major revision, i.e., the current version jumps too quickly too far into specific details that are difficult to follow without having read the paper before; the introduction merely consists of a summary of the paper rather than outlining the big picture, motivation, and use cases. My recommendation would be to separate the details into a preliminaries section and then refocus the introduction on the big picture.
Related to O1, I am missing a bit the clear positioning of this work with respect to existing graph database systems. The approach is later integrated into RDF-3X, so it is clear that RDF and SPARQL are supported but the whole paper is a bit vague about whether property graphs and Cypher could benefit as well. But I guess the applicability goes even beyond. But some more clarity and positioning is needed.

We thank the reviewer for these suggestions. We have edited our introduction to make it less technical and more high level. In particular, in the third paragraph, we start with our key observation and question that motivate our work, which is that often there are multiple algebraic formulas in optimistic estimator to make an estimate and it is unclear which one to choose. In our original submission, without stating this question and observation, we had perhaps too quickly started with our result that optimistic estimators correspond to making estimates along a path in a CEG. We have also removed some technical parts about CEG_{OCR} and instead give a higher-level ideas about our main results instead of summarizing them. We hope this has improved the positioning of our paper.

For the question about applicability of optimistic estimators to property graph models. Not only are they applicable, optimistic estimators have in fact been used in Graphflow [28], which implements subset of OpenCypher language. In Section 4, which reviews optimistic estimators, we have a new part, titled "Note on the generalization of optimistic estimators", in which we discuss both the actual and potential applicability of optimistic estimators.

R2 O2
Explanations could be improved.

- In Figures 3-4, it would be helpful if the authors explain why the CEGs are multi-graphs in the first place. The authors should also explain why some edges are solid while some are dotted.
- In the second paragraph of Section 4, the term "leftmost" is unclear in the sentence "The formula to estimate a 3-path using a Markov table with $h = 2$ is ... of the common edge."
- In Section 4.1, the sentence "In contrast, the first estimate also … condition on 2-size joins" might be confusing. Is the "uniformity assumption" in the sentence more appropriate than the "conditional independence assumption?"
- In Section 5.2.1, the rationale for excluding "extension" attributes from partitioning is unclear. The authors should clearly explain why the partitioning is performed on specific attributes only.
- In Section 5.2.1, it is also unclear whether the authors assume that the workload is known beforehand and pre-partition the tables offline, or they partition tables for each query online. The former will have generalization issues while the latter will introduce online partitioning overhead. The authors should clearly explain this.
- In Section 6, exactly which experiments use $h = 2$ and $h = 3$? It is critical to specify it since the authors mention in Section 6.2 that the path length is no longer a design choice for $h = 2$. Using $h = 2$ for the majority of the experiments can limit the contribution of the paper since they cannot tell the advantage of selecting min-hop or max-hop over another.
- A proper analysis is required for each important observation. In Section 6.2.1, the authors should clearly explain why max-hop outperforms min-hop, for example, using uniformity and independence assumptions, characteristics of the datasets and queries, and the correlation between edge labels. This also applies to Section 6.2.2, why min-hop is better for DBLP while max-hop is better for Hetionet.

We addressed the writing comments of the reviewer as follows: We give an example of why CEGs are multi graphs in Section 3. We clarified the sentences in 4.1 and address the reviewers' two comments about 5.2.1 in that same section. We explain when we use h=2 and h=3 explicitly in each subsection of Section 6. We also realized that we were inaccurately calling the estimator we use when h=2 `max-hop-max`. This was inaccurate because as the reviewer observes when h=2, the choice between min-hop and max-hop does not exist, so only the aggregator choice exists. We now call the optimistic estimator we use when h=2 `max-aggr`.

Analysis explaining why one estimator outperforms another: This is a very important question but also something we are unable to conclusively answer. As we explained in our author feedback, it is not true that one estimator outperforms another on every query in a workload. For any two estimator, we have example queries on even the same dataset, in which one outperforms the other. Indeed, on a given query Q, why estimator i, e.g., `max-hop-max`, does better than another, e.g., `min-hop-min`, depends on all the factors the reviewer

mentions: how many uniformity and independence assumptions each estimator makes, how much these assumptions hold, which in turn depends on the correlations between edge labels. However, having looked at the performances of these estimators extensively during our study, we are not convinced there will be a very simple explanation overall.

In fact we devoted significant amount of our study to studying paths in CEGs that lead to very accurate estimates and those that lead very inaccurate ones to observe conclusive patterns about the edges in these paths. For example, one promising idea we pursued was this: do the edges of paths that lead to good estimates correspond to more "uniform" extensions? That is, suppose in a path that leads to a good estimate for $\overset{A}{\longrightarrow}\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}\overset{D}{\longrightarrow}$, one edge corresponds to $\ldots \times k = |\overset{B}{\longrightarrow}\overset{C}{\longrightarrow}|/|\overset{B}{\longrightarrow}| \times \ldots$. This formula makes a uniformity assumption that each B edge extends to k many C edges. One statistic to make this question more concrete is to look at the coefficient of variance (variance/mean) of the C-degrees of B edges. Although there seems to be some pattern here, it is not conclusive. As we note in the conclusion of our paper, understanding what derives one estimator to be better than another is a very important future work direction.

> R3 O1
> The idea of CEG is not really novel. It is just a lattice of all sub-queries. In particular, the claim that using pessimistic estimators (i.e., worst-case join size bounds) in a CEG (lattice) is "surprising" is definitely an oversell. For example, [1] also uses such a lattice to derive pessimistic estimators.

As per reviewer's suggestion, we do not present our result as surprising in our revision. We also do cover the difference between CEGs and the lattice from [1] in related work in a new Section 5.3.

1.2 Restriction to Edge-labeled Graphs [R1 O3/O4]

> R1 O3/O4
> Section 2 mentions that edge-labeled subgraphs are the focus of the examples, which makes the reader immediately wonder to what extent vertex labels can be considered. In fact, the paper is very unclear about this although in real-world applications, vertex labels are extremely important. It is unclear to what extend this can be covered by the proposed approach - but since the evaluation uses RDF-3X and knowledge graphs, this it is not straightforward to understand how these things go together. Maybe introducing a "real" example graph (in addition or instead of the contained abstract graphs) could already help to a certain extent.
> Going one stop further, queries on graphs often contain restrictions on the attributes, which - since Section 2 restricts the focus of this approach on conjunctive queries - is not discussed in this paper. It would be interesting though to briefly discuss mention if/how the approach could be extended.

We added a new paragraph in Section 4, with a header "Note on the generalization of optimistic estimators" that discusses possible extensions to vertex labels, joins over ternary relations, and queries with predicates. Our discussion summarizes that these extensions are possible and that there is prior work that uses optimistic estimators for queries with vertex labels, but no prior work we are aware of has used an optimistic estimator for queries with arbitrary predicates or relations with > 2 arity.

1.3 Influence of Query Characteristics/Selectivity/Structures and Additional Issues [R1 O7]

1.3.1 *Version of Datasets.*

> R1 O7-1
> The description in the paper is not specific enough which versions of the datasets are used, YAGO, for example, is available in many versions.

We specify the exact dataset versions that we used in our experiments in Section 6.1 with exact links to where we obtained them.

1.3.2 *Original WatDiv Queries.*

> R1 O7-2
> The experiments cover a number of knowledge graphs, which leads us back to O1 and O2. But related work has already established some "standard queries" over such data, e.g., RDF-3X over YAGO, but in particular WatDiv comes with a query load generator but instead the authors created their own queries over the data, which naturally leads to the question why the original WatDiv queries could not be used.

Following the reviewer's suggestion we obtained WatDiv's original queries [43]. Although this workload contain 12400 SPARQL queries, many of them contain only 3 edges, which are very small and many others identical in shape and only differ in their vertex predicates. By a vertex predicate we are referring to SPARQL queries in which the subject or the object is not a variable and fixed with a URI, e.g., "<http://db.uwaterloo.ca/ galuc/wsdbm/ProductCategory14>". After removing the small queries and turning the subject and object URIs in others to variables (this is equivalent to removing vertex labels in our setting), there are 75 different queries left, 9 of which is cyclic and the other 64 acyclic. Note that we *did not modify* the predicates in these queries, which correspond to edge labels in our setting. Because the cyclic component contains very few queries, we only used the 64 acyclic queries. We call this the `WatDiv-Acyclic` workload. These are highly varied queries (so very few of them have the same structure) between 4 to 12 edges and contain very different depths.

We describe the `WatDiv-Acyclic` in Section 6.1 and in our primary experiment to study the space of optimistic estimators in Section 6.2. Our replaced figure can be seen in Figure 9 (the blue sub-figure). The behavior of the optimistic estimators is similar

to their behavior on Acyclic workload, except there were a few queries in which they made larger errors. In our original submission we were not commenting on WatDiv explicitly in the paragraph that discussed the experiment in Figure 9 and our revision does not as well.

We could use WatDiv-Acyclic in 3 other experiments. First is the summary-based estimator comparison experiment in Section 6.4. We replaced our previous Acyclic workload here with WatDiv-Acyclic and updated the middle sub-figure in Figure 16. Our result are unaffected here and our note in our last sentence in this sub-section that the majority max-hop-max queries are overestimates is even more visible. Second, is our new experiment where we study the effects of query depths and sizes, which is another comment the reviewer has made (see our response below in CL 1.3.5). We did not use WatDiv-Acyclic here because this workload contains very few queries compared to Acyclic workload and grouping it according to sizes and depths yields groups with very few queries. Instead, we used the Acyclic workload, which contains 360 queries. Third, is our RDF-3X experiments from Section 6.7, where we had used WatDiv on our Acyclic workload. We repeated this experiment on WatDiv-Acyclic however we only found 5 queries in which RDF-3X picked different plans on the estimators and but the runtime differences of the plans that were picked were very close to each other. We were expecting this because even in our Acyclic workload with 360 queries, we were able to find only 25 queries, in which RDF-3X would pick different plans when using different estimators. Therefore on a smaller workload, we expected even fewer queries in which we would get RDF-3X to pick different plans. Because of this, in Section 6.7 we instead still present the previous WatDiv and Acyclic query workload but we do report that we experimented with RDF-3X and WatDiv-Acyclic workload.

1.3.3 *Extended Version of RDF-3X Code.*

> R1 O7-3
> The reference to the RDF-3X code leads to an extended version, not developed by the original authors. Some more details are necessary to understand the consequences and whether this code is indeed representative for the original approach.

GH-RDF-3X indeed extends the original RDF-3X files, e.g., 177 files in GH-RDF-3X is originally authored by Thomas Neumann, the main author of the original RDF-3X. Some of these files have been edited by the two authors of GH-RDF-3X. The first commit in this repo is the original RDF-3X files. Having studied the code for our paper, it looks like almost the entire query processor operators and the "plangen" module, which is the optimizer that picks plans does not seem to have had any edits (there are some edits to the comments but not any actual code). These are the 2 modules of the system that are most important for our study. We did not use this first commit and instead used the latest commit, but we do not suspect that our results would be different.

However, we should also note that the RDF-3X that was copied in the first commit does not integrate some techniques that are mentioned in RDF-3X papers [31], e.g., the characteristic set estimator. But for our purposes what the default cardinality estimator of RDF-3X system is not important because our goal is primarily to compare our 9 estimators. We also note that to our best knowledge, this is the only open source version of RDF-3X, and we do not know if there was ever another version of RDF-3X that was released with those techniques. If there was, we do not think they are available.

1.3.4 *Filter on Runtime.*

> R1 O7-4
> In Section 6.6 some test queries are filtered out with the argument that all approaches provide "exactly the same plan or there was less than 10% difference". First, 10% difference of what? I assume runtime. Second, I could imagine that the reason for doing this is to amplify the differences but why 10%, why not keep those? Further, by filtering out some of the queries, the composition of characteristics of the remaining 15 queries for DBLP (out of 70 in the other experiments) is totally unclear.

The 10% difference *is* referring to runtime. Following the reviewer's suggestion, we updated the two charts in Figure 18 by only filtering out the queries in which all of the 10 estimators lead to exactly the same plan.

1.3.5 *Analysis on Query Types/Characteristics.*

> R1 O7-5
> In general, a more detailed analysis of the results with respect to the individual query types/characteristics is missing. Intuitively, the more joins/edges, the higher should be the error of standard approaches. However, it also makes a difference whether its path- or star-shaped queries. Unfortunately, the paper does not provide a proper analysis on this aspect.

We thank the reviewer for this suggestion. As we had indicated in our original submission, we had box plots for each query template showing the performance of 9 estimators for each query template in our github repo. Across all datasets, these are over 100 figures and cannot be put in our paper. We hope interested readers can check our github repo. Following the reviewer's suggestion we did a depth and size analysis for our recommended max-hop-max estimator (doing this for all 9 of our estimators would again be over 100 figures) focusing on our acyclic workloads on all of our datasets. This analysis appears in a new Section 6.2.2 in the evaluation in our revision. Overall, we indeed observe that as the size increases the accuracy decreases (see Figure 11). This is expected as optimistic estimators start making more assumptions. For depth, we did not observe a strong pattern for the depth. Due to space limitations we put these figures in the longer version of our paper (Figure 12 in the longer version).

1.4 Scalability [R2 O1]

Please discuss the following scalability issues of CEGs, which could be critical.

1.4.1 *Space Complexity of CEG, Time Complexity of Searching.*

R2 O1-1
Since the CEGs can contain an exponential number of subqueries & paths, the overhead of searching max-weight or min-weight paths can be significant. In Section 6, the authors mainly use path-like queries (see Figure 8), where the number of subqueries and paths seem nearly linear to the query size. To allow a user to guess the scalability of CEG, it would be great if the authors discuss the space complexity of CEG, and the time complexity of searching paths using a query size, h, and a query structure (e.g., star, chain).

We give these details in the last 2 paragraphs of a new Section 6.5, in which we discuss scalability issues of both Markov tables and CEGs. In short, the number of paths depends on both the query Q and the h value of the Markov tables, where stars have the largest CEGs. We report the number of edges and paths in CEGs for different size stars between 6 to 10 edges. While the number of edges are reasonable between 180 to 11475, the number of paths is prohibitively large even for 8-star queries. We thank the reviewer for this comment especially because we believe addressing it will clarify for our readers why estimators based on `max` (and `min`) aggregator, e.g., our recommended `max-hop-max` aggregator for acyclic queries, can be practical and fast, while those based on `average` cannot. This is because for `max`-based estimators we can use the standard shortest/longest path algorithm on DAGs [8] that has a cost of $O(m)$ on a graph with m edges. So e.g., the cost of finding the `max-hop-max` estimate is in the order of number of edges in the CEG and not the number of paths. However, estimators based on `average` aggregator needs to explore every path, which will be prohibitively expensive for large queries. Related to this comment, we also found a bug in our script generating YAGO experiments. Because our implementation of `average` was timing out on 12-edge queries, we had unintentionally removed all 12-edge queries from our boxplot in Figure 9. The other estimators would finish on time but our script removed their outputs for these queries. We fixed this problem and to avoid the timeout for estimators using average, we now sample 100K paths from these CEGs. The YAGO sub-figure in Figure 9 is updated in our revision but the pattern across estimator is exactly the same as before.

1.4.2 *Scalability on Vertex/Edge Labels.*

R2 O1-2
I have some concerns about whether CEGs are scalable to a large number of vertex/edge labels. For example, YAGO initially contains 188K vertex labels, and DBpedia used in G-CARE [26] contains 39.6K edge labels, where the authors here seem to discard all vertex labels in YAGO. Please use the original YAGO dataset and DBpedia to discuss the scalability of CEG on the number of labels, and explain the overhead of building Markov tables and CEGs along with estimation accuracy and runtime. Besides, does the estimation time include the time for building CEGs?

	h = 2		h = 3	
vertex labels	**entries**	**file size**	**entries**	**file size**
2	39.4K	2 MB	12M	617 MB
4	314.4K	16 MB	197M	10 GB
8	2.5M	126 MB	3.1B	158 GB
16	19M	969 MB	50B	2.5 TB

Table 1: Markov table sizes with random vertex labels for Epinions.

As we mentioned in CL 1.2 and in the new "Note on the generalization of optimistic estimators" part of Section 4, Markov tables with vertex labels are used in reference [28] by having entries where query vertices also have labels. This naturally increases the size of the Markov tables. In particular if there are L_V many vertex labels in a graph and h = 3, the vertex-labeled Markov table could potentially be up to L_V^4 times larger than the edge-labeled one (as 3-edge sub-queries have 4 query vertices). However, as we discuss for edge label combinations in our new Section 6.5, where we discuss and evaluate the scalability of Markov tables, in practice we would expect much fewer combinations to exist because of dataset constraints (see our example about Hetionet's `treat` edges in Section 6.5). For the interest of the reviewer, we report the sizes of the Markov table for Epinions when h = 2 and h = 3 and when we randomly put 2, 4, 8, or 16 vertex labels. This represents a worst-case dataset in terms of scalability because vertex (and edge) labels are random, every combination of vertex labels can appear in every sub-query. Table 1 in this cover letter presents this information.

We did not exactly compute the possible size of a vertex-labeled Markov table for YAGO because it has a very large number of vertex labels and many vertex label combinations can exist, not because the vertex labels are random as in our Epinions experiment, but because each vertex has multiple vertex labels. Therefore, for example if two vertices u and v are connected through a single edge e and both u and v have 100 vertex labels, e would lead to 10000 unique 1-edge vertex-labeled sub-query templates. We believe this is a limitation of vertex-labeled Markov tables and they would not be scalable for datasets in which vertices have large number of vertex labels. At the same time, vertex labels in graph datasets often represent the types of entities, e.g., diseases, drugs, persons, and the more common case may be that vertices have a single label (e.g., as in the case of Hetionet).

For DBPedia: Although DBPedia is not one of our datasets, as suggested by the reviewer we did compute its edge-labeled Markov tables and we found it prohibitively large and discuss this in our revision in Section 6.5. This is because there are vertices in DBPedia with a high number of edges with a high number of different edge labels, e.g., one vertex has edges that cover 11323 different edge labels, so simply due to this vertex, there would be 11323^3 many different 3-star entries, which would already take more than 50TB storage in our implementation, which is prohibitive. At the same time on average each edge label in this graph represents a very small number of edges (on average around 10000), therefore many of the entries would have very small cardinalities.

Estimation accuracy and Markov Table sizes: For two of our datasets, we report the effects of h=2 and h=3 on estimation accuracy in Section 6.2.2 but put the figures due to space limitations only

in the longer version of our paper (in the same section). As we discussed in CL 1.4.1, in Section 6.5, we also present the edge-labeled Markov table sizes for all of our datasets.

Time for building CEGs: Yes, our estimation time includes building CEGs. However, in our implementation, the estimation phase is not separate from building the CEG. That is our estimation algorithm builds the CEG implicitly.

1.4.3 *Scalability on h.*

> R2 O1-3
> I also have concerns about h, the maximum join size in the Markov tables. The authors use $h = 2$ or 3, which seems too small. The authors should explain why $h > 3$ is not used, for example, by showing the overhead of building Markov tables for different h. Related to O1-2, it would be also helpful to explain the time complexity of building Markov tables and searching entries (to build CEGs) using h, the number of labels, etc.

Please see our response in CL 1.4.1 and CL 1.4.2 that address the comments on Markov table sizes for different h and CEG sizes. As we note, we discuss these aspects of optimistic estimators in a new Section 6.5 in detail. In the same section we also discuss the challenges of using Markov tables with h > 3 in practice.

1.4.4 *Dataset Specific Markov Table.*

> R2 O1-4
> In Section 6, the authors briefly mention that they generate workload-specific Markov tables. This indicates that some new queries cannot be supported once the tables are built, which is unrealistic in that a query might not be known beforehand. To cover all possible queries in a dataset, how large would the Markov tables be, and what is the overhead of building the tables?

We agree that in practice full Markov table sizes should be used and in our new Section 6.5 we report full Markov table sizes and not our workload-specific ones. We removed our previous worklaod-specific Markov table sizes from our paper.

1.5 Experimental Study [R2 O3]

Experiments could be improved in addition to O1.

1.5.1 *Bound Sketch Optimization.*

> R2 O3-1
> Related to O2-5, the authors should either report offline or online partitioning overhead for bound sketch optimization.

We report our offline partitioning numbers in a new sentence in Section 6.3. There is no separate online partitioning in the bound sketch optimization, though there is the time to make estimates for each partition. The time to make an estimate using a Markov table is unaffected whether the Markov table represents a partition of the input graph or the whole. Estimation numbers we report in Section 6.6 is indicative of this time. When using the bound sketch optimization, we simply make more estimates and sum these estimates.

1.5.2 *Original Job Queries.*

> R2 O3-2
> For the JOB setting, I worry that the authors have discarded the original edge labels of JOB queries and randomly added labels. Please use the original JOB queries. Besides, uniformly sampling edge labels can be biased towards CEGs since CEGs inherently use the uniformity assumption.

We thank the reviewer for this comment. We realized that our explanation of how we generated a subgraph query workload from JOBs required clarification. We think our previous explanation may have confused the reviewer. Previously, we had explained how the nodes and edges were created but we had not explained how the edge labels are assigned. In our revision we clarified our workload generation in the IMDb and JOB paragraph in Section 6.1. In particular, there are 2 types of edges we added between nodes that were generated from entity tables. The edges corresponding to foreign-key joins get a single fixed label. However, for edges that correspond to tuples of relationship tables we had a modeling choice to make. We take an example. The table' `movie_companies` is a relationship table connecting movie entities and companies and each row of it corresponds to one edge. We could label all those edges as `movie_companies` edges. We thought this was not a natural modeling choice. For example this could create this edge: (Forest Gump)-[:movie_companies]->(Paramount). We thought a more natural way was to pick these edge labels from the type value of the tuples in `movie_companies`, which take 4 values: distributor, producer, special effects, miscallenous.

Then when we created our subgraph query workload, we took the JOB queries and after removing non-join predicates, we obtained 7 different join templates, which corresponded to 7 different subgraph query templates. Many of the edges in these query templates corresponded to edges over relationship tables, and we needed to give them labels, since our study focuses on edge labeled queries. That is exactly where we involved some randomness and added one of the possible types as an edge label. We note that we are already deviating from the original JOB queries by removing non-join predicates, which we have to do. For example, JOB queries contain predicates, such as the predicate n.name LIKE '/B%', which we cannot use in our study. However, we are not deviating from the join structures and join conditions but to generate more instances of these joins we are effectively adding new type predicates to these queries.

We hope this clarifies our conversion process. We do not think this process gives CEGs a particular advantage over other cardinality estimators because we are not changing the underlying database, so the characteristics of the database stays the same, i.e., whether some values are skewed or uniform remains unchanged.

1.5.3 Comparison to More Recent Sampling Works (R2 O3-3, R3 O4).

> R2 O3-3
> In Section 6.5, comparing CEG with WanderJoin (SIGMOD'16) might seem outdated. Please compare with two recent works [1] and [2]. They are very relevant to this work. Please also use other datasets such as HPRD and Youtube, which would strengthen the experiments. [1] Yuan Qiu, Yilei Wang, Ke Yi, Feifei Li, Bin Wu, Chaoqun Zhan: Weighted Distinct Sampling: Cardinality Estimation for SPJ Queries. SIGMOD Conference 2021: 1465-1477
> [2] Kyoungmin Kim, Hyeonji Kim, George Fletcher, Wook-Shin Han: Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation. SIGMOD Conference 2021: 964-976

> R3 O4
> For cyclic queries, [b] provides a better solution than WanderJoin for join size estimation.
> [b] Yu Chen, Ke Yi: Random Sampling and Size Estimation Over Cyclic Joins. ICDT 2020.

We thank the reviewers for these three references as we had not studied them by the time of our submission. We studied these references and discuss them in the related work of our revision. First we note that our sampling experiments used our `Acyclic` workload, so we did not implement the ICDT paper on cyclic joins. We also did not implement "Weighted Distinct Sampling: Cardinality Estimation for SPJ Queries" paper because it explicitly states in its second page that their technique specifically works for queries that have only a single column in their projections and the authors leave the more general case to future work. Our queries would correspond to outputting all of the columns of our subgraph queries. The "Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation" paper was directly relevant but it describes a hybrid sampling-summary-based cardinality estimator algorithm called Alley. Although this is not a purely sampling algorithm, the authors of Alley propose and use an alternative algorithm to WanderJoin, which we call Alley-S, for Alley **s**ampling. This is described as Algorithm 3 in the Alley paper.

In contrast to WanderJoin, which samples a subgraph (or a join) query Q one edge at a time, Alley-S samples Q one vertex at a time. Briefly, Alley-S iteratively picks an unmatched query vertex q_{v_1} in Q and matches it to a data vertex v_1^* in the input graph. To sample q_{v_i} however, Alley-S first computes all of the data vertices that have at least one edge that would match the edge label of each query edge incident on q_{v_1} (in the appropriate direction). For example if q_{v_i} is incident on 2 edges in Q with labels A and B, say, both in the forward direction, then Alley-S first finds all data vertices that have both at least one A and one B edge. This is done by storing for each edge label L_E, all vertices that have an L_E edge in the forward and backward direction. Alley-S introduces a second technique called branching factor b, which is the main knob for adjusting how much

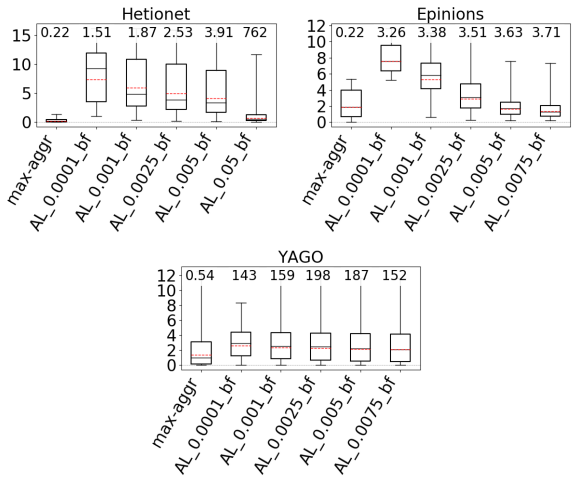


Figure 1: Comparison of `max-hop-max` and Alley-S on `Acyclic` workload on Hetionet and Epinions, and `G-CARE-Acyclic` workload on YAGO.

sampling happens. The branching factor determines the fraction of the candidate vertices that match q_{v_1} to sample. Once q_{v_1} is fixed to a vertex v_1^*, the process iteratively "extends" v_1^* to further partial matches that match a new query vertex q_{v_2} by picking from v_1^*'s neighbors (but ensuring again that these neighbors have edges that match each edge label incident on q_{v_2}).

We implemented Alley-S and provide it in our github repo (https://github.com/cetechreport/CEExperiments; see our link to our clone of G-CARE repo) for the reviewers' interest. Note that Alley-S does not implement the summary-based component of Alley, so it is not expected to match the reported performance of Alley from reference [20]. We ran Alley-S with varying branching factors. We present a subset of our experiments in Figure 1 of this cover letter. As before the boxplots show the average q-error accuracies of 5 runs and the numbers on top of the boxplots are the estimation times we obtain. Compared to WanderJoin, we find Alley-S to generate similar behavior but also find its estimation times slower at branching factors that achieves similar accuracy as `max-aggr` or WanderJoin. For example, on Epinions, while WanderJoin with an average of 0.43ms estimation time becomes competitive with `max-aggr`, which is the optimistic estimator that uses the max aggregator, Alley-S achieves this accuracy with an estimation time of 3.51ms (with a branching factor of 0.0025). The difference is significantly larger for Hetionet and YAGO[1]. An important cost of Alley-S is an initial sampling for the first query vertex, which requires intersection of large lists. Since we found WanderJoin to perform better, we still use WanderJoin in our paper but discuss our experiments with Alley-S in Section 6.6.

Finally, R2 has suggested for us to add 2 additional datasets to our workloads. We do not want to ignore any of the reviewer suggestions but at the same time, we think we have a dense paper with many figures. Each dataset leads to many new figures either

[1]The sometimes decrease in YAGO in estimation time of Alley-S when we increase branching factor is not a bug. We investigated this and this happens in cases when Alley-S cannot sample a successful match of query and estimates 0 in all of the branching factor values we tried. In this case, because all branches lead to no-matches due to randomness, sometimes some branches might return earlier in a run with a larger branching factor, leading to quicker estimates.

in our main text or the longer version of our paper and we need to scope our experimental evaluation. We also think the datasets and workload coverage is similar to the recent publications on cardinality estimation in our community, so we have left this to future work and not undertaken this study in our revision.

1.6 Updates to the Data [R2 O4]

> R2 O4
> It would be great if the authors discuss updates to the data. Time and space complexity should be analyzed if CEGs can be incrementally maintained.

First, we note what needs to be maintained are not CEGs, as CEGs are not explicitly materialized. Instead they are implicitly constructed when making an estimate when a query arrives in a system. Markov tables however can be maintained upon updates to data. None of the prior literature on optimistic estimators we are aware of discusses updates, except the very last sentence of reference [25], which mentions updates as a future work discussion. On the one hand maintaining the entries in Markov tables is related to maintaining output sizes of materialized views. However, in practice we think this would be too costly for a system. From the perspective of the optimizer of a system, Markov tables are not different than other database statistics an optimizer users, such as value frequencies and histograms and periodically updating these could be more practical. This is for example done in the popular PostgreSQL system that updates the system's statistics through explicit commands (https://www.postgresql.org/docs/9.5/planner-stats.html). We are not planning on studying this aspect of Markov tables as future work, so have not added this discussion to our future work direction, and are only providing it in our cover letter.

1.7 Join Size Bound [R3 O3]

> R3 O3
> Why do you only consider MOLP? A more general LP-based join size bound based on degree information is given in [a]. As far as I can see, the bound in [a] is better than MOLP.
> [a] Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu: What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another? PODS 2017

As suggested by the reviewer in their response to our feedback, we did include our work on CLLP and sub-modularity constraints in our revised paper. In the main text, in Section 5.3, we reviewed CLLP and our limitation or not being able to represent CLLP as a path in a CEG as we do for MOLP. We also cover a summary of our experiments in the same subsection and present the actual experiments on our `acyclic` workloads in the Appendix C of the longer version of our paper [12]. Briefly, aside from Hetionet, where we observe a small visible improvement (36% in the mean accuracy), CLLP does not improve over MOLP significantly.

1.8 Join Queries of Arity ≥ 2 [R3 O5]

> R3 O5
> Why not use your framework for general join queries instead of only graph queries (each relation has arity 2)?

Please see our response in CL 1.2 and our new paragraph in Section 4, with a header "Note on the generalization of optimistic estimators" that addresses this comment.

1.9 Sample Size for Large Cyclic Queries [R3 O6]

> R3 O6
> For handling large cyclic queries by sampling, how to choose the sample size p?

We believe this refers to the parameter p we have to fill Markov tables with cycle closing rates to estimate queries using CEG_{OCR}. We do not know a principled heuristic for picking p and our understanding here is limited. However, interestingly the reviewer has also pointed us to an ICDT paper [7] for sampling cyclic joins, which describes a sampling technique which can guarantee a constant-factor approximation to the cardinality of a cyclic join after only constant number of samples (but also building an appropriate index). Our understanding is limited but it may be possible to adapt the sampling algorithm of this technique to compute closing rates as well.

2 REVIEWER 3

2.1 Choosing Best Path in CEG [R3 O2]

> R3 O2
> The strategy of choosing best path in CEG is fully empirical. No theoretical justification or guidance is provided.

As we acknowledged in our rebuttal, this is an important limitation of our work. In our revision we acknowledge this also in our future work directions in Section 8. There, we note that although our study can give concrete advice to practitioners about which optimistic estimators to use, an important question for future work is to understand what dataset properties drive why one estimator is better than another. We further note that, as we explained in our rebuttal, prior work that proposed optimistic estimator either ignore or do not attempt to justify their choices for path-length or aggregator choices. So we think that our paper's empirical evaluation of these choices is a contribution to our field's knowledge in this area.

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

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 begin by addressing the problem of how to make accurate estimates for optimistic estimators. We model these estimators as picking bottom-to-top paths in a *cardinality estimation graph* (CEG), which contains sub-queries as nodes and edges whose weights are average degree statistics. We outline a space of optimistic estimators to make an estimate on CEGs and show that effective ones depend on the structure of the input queries. While on acyclic queries and queries with small-size cycles, using the maximum-weight path is an effective technique to address the well known underestimation problem, on queries with larger cycles these estimates tend to overestimate, which can be addressed by using minimum weight paths or an alternative CEG. We then show that CEGs can also model the recent pessimistic estimators. This surprising result connects two disparate lines of work, allows us to adopt an optimization from pessimistic estimators to optimistic ones, and provide insights into the pessimistic estimators, such as showing that they have combinatorial solutions.

1 INTRODUCTION

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

We focus on the prevalent technique used by existing systems of using statistics about the base relations or outputs of small-size joins to estimate cardinalities of joins. These techniques use these statistics in algebraic formulas that make independence and uniformity assumptions to generate estimates for queries [2, 25, 28, 30]. We refer to these as *summary-based optimistic estimators* (optimistic estimators for short), as these estimators can make both under and overestimations. This contrasts with the recent *pessimistic estimators* that are based on worst-case optimal join size bounds [1, 5, 6, 14, 19] and avoid underestimation, albeit using very loose estimates [33]. In this work, we study how to make accurate estimations using optimistic estimators using a new framework that we call *cardinality estimation graphs* (CEGs) to represent them.

The main observation that motivates our work is that in prior optimistic estimators, there is often more than one algebraic formula that can be used to make an estimate for a query, and no clear answer for which one to use. For example, consider the query in Figure 1. Given the accurate cardinalities of all subqueries of size $\leq$ 2, there are 252 formulas to make an estimate. 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 [2, 25, 28], the choice of which of these estimates to use has either been unspecified or fixed without acknowledging possible other choices. Our paper aims to answer this question by systematically studying these alternative estimates and empirically justifying which estimates are more accurate.

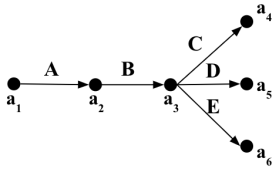
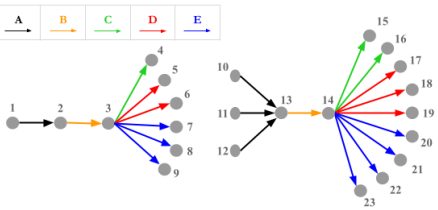


Figure 1: Example subgraph query Q_{5f}.

We begin by showing that the algebraic formulas of prior optimistic estimators can be modeled as picking a bottom-to-top path in a weighted CEG, which we call CEG_O, for **O**ptimistic. In this CEG nodes are intermediate sub-queries and edges weights are average degree statistics that extend sub-queries to larger queries. For example, the CEG_O for the query in Figure 1 and the input dataset in Figure 2 is shown in Figure 3. Each path of this CEG corresponds to a possible algebraic formula and the corresponding estimate is the multiplication of the weights of the edges in the path. As our first main contribution, we systematically describe a space of estimators, defined by heuristics to pick a CEG path, i.e., an algebraic formula, for making an optimistic estimate, and show empirically that the better performing estimators depend on the structure of the query. We show that on acyclic queries and queries with small-size cycles, using the *maximum-weight path*, which corresponds to choosing the highest estimating formula, is an effective way to make accurate estimations. This is because as in the relational setting, estimators that use independence assumptions tend to underestimate the true cardinalities of queries, and picking the highest estimating formula can offset these underestimations. In contrast, we observe that on queries that contain larger cycles, optimistic estimators tend to overestimate, and address this by proposing other heuristics and an alternative CEG that uses different edge weights.

As our second main contribution, we show that CEGs are expressive enough to model also the recent linear program (LP)-based pessimistic estimators. Specifically, we show that we can take CEG_O and replace its edge weights (which are average degrees) with maximum degrees of base relations and small-size joins, and construct a new CEG, which we call CEG_M. Unlike the optimistic estimators, we now show that picking the minimum weight path would (provably) be the most accurate estimate and this path is indeed equivalent to solution of the LP that defines a pessimistic estimator



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.

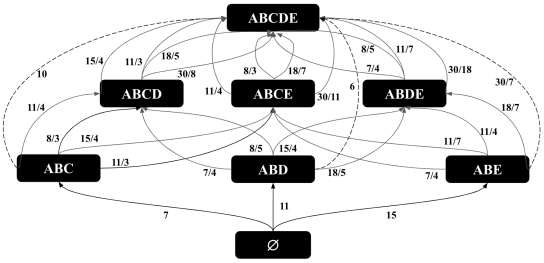


Figure 3: CEG_O for query Q_{5f} in Figure 1 when the Markov table (§4) contains joins up to size 3.

from reference [19] called MOLP. This result connects two seemingly disparate classes of estimators: *both subgraph summary-based optimistic estimators and the recent LP-based pessimistic ones can be modeled as different instances of estimators that pick paths through CEGs.* Our ability to model both of these classes of estimators in a common framework allows us to apply an optimization called the *bound sketch* optimization designed for the recent pessimistic estimators from references [6] also to optimistic estimators.

CEGs turn out to be very useful mathematical tools to prove properties of pessimistic estimators. For example, using CEGs in our proofs, we can derive combinatorial proofs to some properties of MOLP, e.g., that MOLP s at least as tight as the pessimistic estimator proposed by Cai et al [6] and are identical on acyclic queries over binary relations or that it is tighter than another bound called DBPLP [19]. These applications of CEGs may be of independent interest to readers interested in the theory of pessimistic estimators.

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. This graph can be represented using five binary relations, one for each edge label, as shown in Figure 2.

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

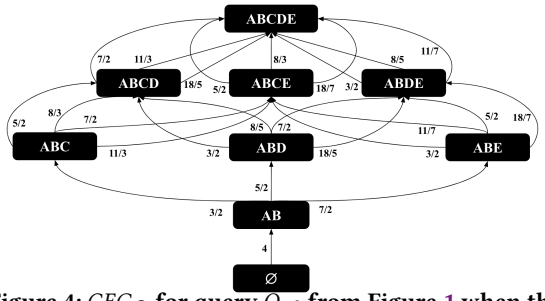


Figure 4: CEG_O for query Q_{5f} from Figure 1 when the Markov table (§4) contains joins up to size 2.

$a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$. Similarly, the query $Q(a_1, a_2, a_3) = A(a_1, a_2) \bowtie B(a_3, a_2)$ will be represented as $a_1 \xrightarrow{A} a_2 \xleftarrow{B} a_3$.

3 CEG OVERVIEW

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

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

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

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

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

4 OPTIMISTIC ESTIMATORS

The estimators that we refer to as *optimistic* use statistics about the input database in formulas that make uniformity and independence or conditional independence assumptions. The cardinality estimators of several systems fall under this category. We focus on

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

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

Table 1: Example Markov table for h=2.

three estimators: *Markov tables* [2] from XML databases, graph summaries [25] from RDF databases, and the graph catalogue estimator of the Graphflow system [28] for managing property graphs.

We begin by giving an overview of the Markov tables estimator [2]. A Markov table of length $h \geq 2$ stores the cardinality of each path in an XML document's element tree up to length h and uses these to make predictions for the cardinalities of longer paths. Table 1 shows a subset of the entries in an example Markov table for $h = 2$ for our running example dataset from Figure 2. The formula to estimate a 3-path using a Markov table with $h = 2$ is to multiply the cardinality of one of the 2-paths with the consecutive 2-path divided by the cardinality of the common edge. For example, consider the query $Q_{3p} = \xrightarrow{A}\xrightarrow{B}\xrightarrow{C}$ against the dataset in Figure 2. The formula for Q_{3p} would be: $|\xrightarrow{A}\xrightarrow{B}| \times (|\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|)$. 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 assumes that the number of C edges that each B edge extends to is uniformly $r = |\xrightarrow{B}\xrightarrow{C}| / |\xrightarrow{B}|$. Equivalently, this is the "average C-degree" of nodes in the $\xrightarrow{B}\xrightarrow{C}$ paths. The result of this formula is $4 \times \frac{3}{2} = 6$, which underestimates the true cardinality of 7. The graph summaries [25] for RDF databases and the graph catalogue estimator [28] for property graphs have extended the contents of what is stored in Markov tables, respectively, to other acyclic joins, such as stars, and small cycles, such as triangles.

Note on the generalization of optimistic estimators: Although we are focusing on edge-labeled queries in our work, the optimistic estimator from reference [28] is generalized to both vertex- and edge-labeled queries. In this reference, the Markov table (called the catalogue) contains entries such as $L_{v1}\xrightarrow{L_{e1}}L_{v2}\xrightarrow{L_{e2}}L_{v3}$, which represents a 2-path subquery entry, where L_{vi} and L_{ej} are label restrictions of the vertices and edges, respectively. In our query notations, this could be represented as a join of 5 relations instead of 2, if we represent each vertex label as a relation L_i.

Note that our sub-graph query notation is merely a syntactic sugar to represent joins between relations represented by edge/vertex label. As such, edge or vertex labels do not necessarily need to represent full relations but can be subsets of relations with additional predicates. For example, one can model each set of edges with label L_{e1} with a particular value t_j for property p as a new relation $L_{e1}^{p=t_j}$ and have entries in a Markov table for joins that contain $L_{e1}^{p=t_j}$ edge labels, e.g., we could estimate the size of $\xrightarrow{L_{e1}^{p=t_j}}\xrightarrow{L_{e2}}\xrightarrow{L_{e3}^{p=t_k}}$ with a formula: $|\xrightarrow{L_{e1}^{p=t_j}}\xrightarrow{L_{e2}}| \times (|\xrightarrow{L_{e2}}\xrightarrow{L_{e3}^{p=t_k}}| / |\xrightarrow{L_{e2}}|)$, if the Markov table contains the cardinalities of the sub-queries in the formula.

The common setting in which optimistic estimators have been used in prior work has been on database management systems that adopt a graph model, where edge labels naturally represent binary relations between two sets of entities. However, labels do not need to represent binary relations either. For example, $\xrightarrow{L_{e1}}\xrightarrow{L_{e2}}$ can represent a binary join between ternary relations L_{e1} and L_{e2} on more than one common attribute. However, we are unaware of any prior work that uses optimistic estimators to queries with non-join predicates or joins over relations with arity > 2.

4.1 Space of Possible Optimistic Estimators

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

- *Vertices:* For each connected subset of relations $S \subseteq \mathcal{R}$ of Q, we have a vertex in CEG_O with label S. This represents the sub-query $\bowtie_{R_i \in S} R_i$.
- *Edges:* Consider two vertices with labels S and S' s.t., $S \subset S'$. Let $\mathcal{D}$, for **d**ifference be $S' \setminus S$, and let $\mathcal{E} \supset \mathcal{D}$, for **e**xtension be an entry in the Markov table, and let $\mathcal{I}$, for **i**ntersection, be $\mathcal{E} \cap S$. If $\mathcal{E}$ and $\mathcal{I}$ exist in the Markov table, then there is an edge with weight $\frac{|\mathcal{E}|}{|\mathcal{I}|}$ from S to S' in CEG_O.

When making estimates, we will apply two basic rules from prior work that limit the paths considered in CEG_O. First is that if the Markov table contains size-h joins, the formulas use size-h joins in the numerators in the formula. For example, if $h = 3$, we do not estimate the cardinality of a sub-query $\xrightarrow{A}\xrightarrow{B}\xrightarrow{C}$ by a formula $\frac{\xrightarrow{A}\xrightarrow{B}}{} \times \frac{\xrightarrow{B}\xrightarrow{C}}{\xrightarrow{B}}$ because we already store the true cardinality of $\xrightarrow{A}\xrightarrow{B}\xrightarrow{C}$ in the Markov table. Second, for cyclic queries, which was covered in reference [28], an additional early cycle closing rule is used in the reference when generating formulas. In CEG formulation this translates to the rule that if S can extend to multiple S's and some S' contain additional cycles that are not in S, then only such outgoing edges of S to such S' are considered. Even when the previous rules are applied, there may be multiple $(\emptyset, Q)$ paths that lead to different estimates:

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

- $|\xrightarrow{A}\xrightarrow{B}| \times \frac{|\xrightarrow{B}\xrightarrow{C}|}{|\xrightarrow{B}|} \times \frac{|\xrightarrow{B}\xrightarrow{D}|}{|\xrightarrow{B}|} \times \frac{|\xrightarrow{B}\xrightarrow{E}|}{|\xrightarrow{B}|} = 52.5$
- $|\xrightarrow{A}\xrightarrow{B}| \times \frac{|\xrightarrow{B}\xrightarrow{C}|}{|\xrightarrow{B}|} \times \frac{|\xleftarrow{C}\xrightarrow{D}|}{|\xrightarrow{C}|} \times \frac{|\xleftarrow{D}\xrightarrow{E}|}{|\xrightarrow{D}|} = 57.6$

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

- $|\xrightarrow{A}\xrightarrow{B}\xrightarrow{C}| \times \frac{|\xleftarrow{C}\xrightarrow{D}\xrightarrow{E}|}{|\xrightarrow{C}|}$

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

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

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

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

Any combination of these two choices can be used to design an optimistic estimator. The original Markov tables [2] chose the `max-hop` paths. In reference [2] queries were paths, so when the set of paths are chosen, any CEG_O path gives the same estimate. Therefore an aggregator is not needed. Graph summaries [25] chooses the `min-hop` paths and leaves the aggregator unspecified. Graph catalogue [28] picks the `min-hop` and `min-aggr` aggregator, without considering alternatives. Instead, we do a systematic empirical analysis of this space of estimators in Section 6 and show that the best paths and aggregator combinations depend on the query structure.

4.2 CEG_{OCR}: Handling Large Cyclic Patterns

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

Faced with a large cyclic query Q, we observe that the optimistic estimators do not actually produce estimates for Q. Instead, they produce an estimate for a similar acyclic Q' that includes all of Q's edges but is not closed. Consider estimating a 4-cycle query in Figure 5 using a Markov table with h=3. The CEG_O for this setting is shown in Figure 6a. Consider the left most path corresponding to the formula: $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}| \times |\overset{B}{\rightarrow}\overset{C}{\rightarrow}\overset{D}{\rightarrow}| / |\overset{B}{\rightarrow}\overset{C}{\rightarrow}|$. Note that this formula is in fact estimating a 4-path $\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}\overset{D}{\rightarrow}$ rather than the 4-cycle shown in Figure 5. This is true for each path in CEG_O.

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

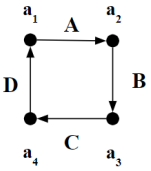
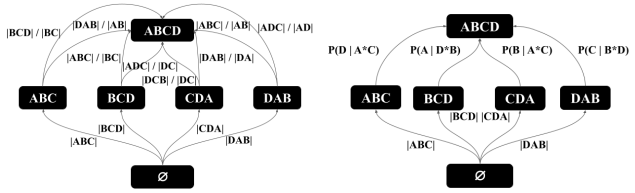


Figure 5: An example of a 4-cycle cyclic query



(a) CEG_O. **(b)** CEG_{OCR}.

Figure 6: CEG_O and CEG_{OCR} of 4-cycle query from Figure 5.

We next describe an alternative modified CEG to address this problem. Consider a query Q with a k-cycle C where $k > h$. Note that in order to not break cycles into paths, we need CEG edges whose weights capture the cycle closing effect when extending a sub-query S that contains $k-1$ edges of C to a sub-query S' that contains C. We capture this in a new CEG called CEG_{OCR}, for **O**ptimistic **C**ycle closing **R**ate, which modifies CEG_O as follows. We keep the same vertices as in CEG_O and the same set of edges, except when we detect two vertices S and S' with the above property. Then, instead of using the weights from the original CEG_O between S and S', we use pre-computed cycle closing probabilities. Suppose the last edge that closes the cycle C is E_i and it is between the query edges E_{i-1} and E_{i+1}. In the Markov table, we store the probabilities of two connected E_{i-1} and E_{i+1} edges to be connected by an additional E_i edge to close a cycle. We denote this statistic as $P(E_i|E_{i-1} * E_{i+1})$. We can compute $P(E_i|E_{i-1} * E_{i+1})$ by computing all paths that start from E_{i-1} and end with E_{i+1} of varying lengths and then counting the number of E_i edges that close such paths to cycles. On many datasets there may be a prohibitively many such paths, so we can sample p of such paths. Suppose these p paths lead to c many cycles, then we can take the probability as c/p. Figure 6b provides the CEG_{OCR} for the 4-cycle query in Figure 5. We note that the Markov table for CEG_{OCR} requires computing additional $P(E_i|E_{i-1} * E_{i+1})$ statistics that CEG_O does not require. The number of entries is at most $O(L^3)$ where L is the number of edge labels in the dataset. For many datasets, e.g., all of the ones we used in our evaluations, L is in the order of 10s or 100s, so even in the worst case these entries can be stored in several MBs. In contrast, storing large cycles with h>3 edges could require $\Theta(L^h)$ more entries.

5 PESSIMISTIC ESTIMATORS

Starting from the seminal result by Atserias, Grohe, and Marx in 2008 [5], several upper bounds have been provided for the output sizes of join queries under different known statistics. For example the initial upper bound from reference [5], now called the *AGM bound*, used only the cardinalities of each relation, while later bounds, DBPLP [19], MOLP [19], and CLLP [1] used maximum degrees of the values in the columns and improved the AGM bound.

Since these bounds are upper bounds on the query size, they can be used as *pessimistic estimators*. This was done recently by Cai et al. [6] in an actual estimator implementation. We refer to this as the CBS estimator, after the names of the authors. We next show that some of the recent pessimistic estimators [6, 19] can be modeled as making an estimate using a CEG.

In Section 5.1, we show that similar to optimistic estimators, MOLP and CBS can also be modeled as an estimator using a CEG. As a direct benefit of modeling pessimistic and optimistic estimators in a common framework, we show that the bound sketch optimization of the CBS estimator [6] can in fact be applied to any estimator using a CEG, specifically the optimistic ones. As a second benefit, the CEG modeling of pessimistic estimators allows us to derive proofs that provide insights into their properties. For example, using CEG modeling, we prove that the CBS estimator is equivalent to MOLP on acyclic queries and provide an alternative combinatorial proof that MOLP is tighter than another bound DBLP from reference [19].

5.1 MOLP

MOLP was defined in reference [19] as a tighter bound than the AGM bound that uses additional degree statistics about input relations that AGM bound does not use. We first review the formal notion of a degree. Let $\mathcal{X}$ be a subset of the attributes $\mathcal{A}_i$ of some relation $\mathcal{R}_i$, and let v be a possible value of $\mathcal{X}$. The *degree* of v in $\mathcal{R}_i$ is the number of times v occurs in $\mathcal{R}_i$, i.e. $deg(\mathcal{X}(v), \mathcal{R}_i) = |\{t \in \mathcal{R}_i | \pi_{\mathcal{X}}(t) = v\}|$. For example, in Figure 2, $deg(s(3), E) = 3$ because the outgoing E-degree of vertex 3 is 3. Similarly $deg(d(2), A)$ is 1 because the incoming A-degree of vertex 2 is 1. We also define $deg(\mathcal{X}, \mathcal{R}_i)$ to be the maximum degree in $\mathcal{R}_i$ of any value v over X, i.e., $deg(X, \mathcal{R}_i) = \max_v deg(\mathcal{X}(v), \mathcal{R}_i)$. So, $deg(d, A) = 3$ because vertex 13 has 3 incoming A edges, which is the maximum A-in-degree in the dataset. The notion of degree can be generalized to $deg(X(v), Y, \mathcal{R}_i)$, which refers to the "degree of a value v over attributes X in $\pi_Y\mathcal{R}_i$", which counts the number of times v occurs in $\pi_Y(\mathcal{R}_i)$. Similarly, we let $deg(X, Y, \mathcal{R}_i) = \max_v deg(X(v), Y, \mathcal{R}_i)$. Suppose a system has stored $deg(X, Y, \mathcal{R}_i)$ statistics for each possible $\mathcal{R}_i$ and $X \subseteq Y \subseteq \mathcal{A}_i$. MOLP is the output of this LP:

Maximize $s_{\mathcal{A}}$

$$s_\emptyset = 0$$

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

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

The base of the logarithm can be any constant and we take it as 2. Let m_A be the optimal value of MOLP. Reference [19] has shown that 2^{m_A} is an upper bound on the size of Q.

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

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

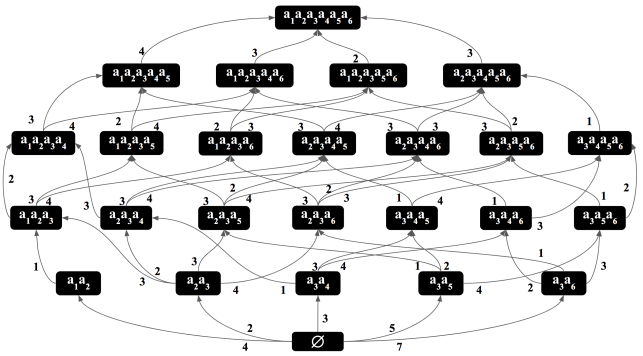


Figure 7: CEG_M for query Q_{5f} in Figure 1.

- *Projection Edges:* For each $s_X \leq s_Y$ inequality (i.e., $\forall X \subseteq Y$), add an edge with weight 0 from Y to X. These indicate that the size of Q_X is at most the size of Q_Y, if Y is a larger subquery.

Figure 7 shows the CEG_M of our running example. For simplicity, we use actual degrees instead of their logarithms as edge weights and omit the projection edges in the figure. Below we use $(\emptyset, \mathcal{A})$ instead of $(\emptyset, Q)$, to represent the bottom-to-top paths in CEG_M.

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

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 7. In this CEG, the minimum-weight $(\emptyset, \mathcal{A})$ path has a weight of 96, corresponding to the leftmost path in Figure 7. We make two observations.

Observation 1: Reference [19] proves through a numeric LP-duality argument that $2^{m_{\mathcal{A}}}$ is an upper bound on the the output size of the query (OUT), i.e., $OUT \leq 2^{m_{\mathcal{A}}}$ (see Prop. 2 [19]). We next observe that our CEG formulation of MOLP provides arguably a much simpler proof of this property. As in the 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 7 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 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. 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.

5.2 CBS and Bound Sketch Optimization

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

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

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

Step1: For each $p = (\emptyset, \mathcal{A})$ path in CEG_M (so a bounding formula in the terminology used in reference [6]), let S be the join attributes that are not extension attributes through a bounded edge[3]. For each attribute in S, allocate $K^{1/|S|}$ partitions. For example, consider the path $P_1 = \emptyset \xrightarrow{|B|} a_2a_3 \xrightarrow{deg(a_3,C)} a_{2-4} \xrightarrow{deg(a_2,A)} a_{1-4} \xrightarrow{deg(a_3,E)} a_{1-4}a_6 \xrightarrow{deg(a_3,D)} a_{1-6}$ in the CEG_M of Q_{5f} from Figure 7, 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. Specifically, we can use a bound sketch to refine optimistic estimators and we will evaluate its benefits in Section 6.3. We implemented the bound sketch optimization for optimistic estimators as follows. Given a partitioning budget K and a set of queries in a workload, we worked backwards from the queries to find the necessary subqueries, and for each subquery the necessary statistics that would be needed are stored in the Markov table. Bound sketches are query-specific, so computing the right Markov table entries (or degree statistics) requires pre-knowledge of the workloads. We are unaware of a workload-agnostic technique to generate all necessary statistics for arbitrary queries.

5.3 Discussion on CLLP

MOLP is not the tightest known upper bound in literature. The tightest known upper bound is CLLP [1], which extend the MOLP

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

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

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

Table 2: Dataset descriptions.

with a set of *sub-moduliarity constraints* (also known as the *Shannon inequalities*). With the addition of these constraints to MOLP, we can no longer map the solution of CLLP to a path in a CEG. In fact, we believe this cannot be done as sub-modularity constraints represent a relationship between 4 sub-queries and do not seem to have an interpretation as weighted edges as the other constraints of MOLP, which are between 2 sub-queries. We review CLLP in Appendix C, where we also compare the accuracy benefits of making estimates using CLLP on our acyclic query workloads. Instead of using a CEG and a combinatorial solution, we used a numeric LP solver to solve CLLP formulations of our queries. Except for Hetionet, where the obtained 36% improvement in mean accuracy, we saw negligible improvements.

We end this section with an important note on reference [1]. CLLP is a generalization of another LP from reference [15] called GLVV, which was introduced to study join output sizes under functional dependencies (a restricted form of degree-constraints). As part of deriving CLLP, reference [1] also uses lattices, which might look similar to our CEG, as a mathematic tool to determine queries and functional dependencies under which GLVV is tight. Instead, we use CEGs to *solve* an LP.

6 EVALUATION

We next present our experiments evaluating and analyzing the optimistic and pessimistic estimators we study and the bound sketch optimization. Our code, datasets, and queries are available at https://github.com/cetechreport/CEExperiments.

6.1 Setup, Datasets and Workloads

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

IMDb and JOB [23]: The IMDb relational database, together with a workload called JOB, has been used for cardinality estimation studies in prior work [6, 23]. We created property graph versions of the this database and workload as follows. IMDb contains three groups of tables: (i) *entity tables* representing entities, such as actors (e.g., `name` table), movies, and companies; (ii) *relationship tables* representing many-to-many relationships between the entities (e.g., the `movie_companies` table represents relationships between movies and companies); and (iii) *type tables*, which denormalize the entity or relationship tables to indicate the types of entities or relationships. We converted each row of an entity table to a vertex. Let u and v be vertices representing, respectively, rows r_u and r_v from tables T_u an T_v. We added two sets of edges between u and v: (i) a *foreign key edge* from u to v if the primary key of row r_u is a foreign key in row r_v with a fixed edge label E_{T_u,T_v}; (ii) a *relationship edge* between u to v if a row r_ℓ in a relationship table T_ℓ connects row r_u and r_v. The label of the edge between u and v in this case would come from the type value of the r_ℓ tuple that "connected" u and v. For example if a movie entity u is joined with company entity v through the `movie_companies` table, then there can be 4 different labels this edge can have: distributor, producer, special effects, miscallenous. These are the four values in the `company_type` table that denormalizes the type information in `movie_companies`.

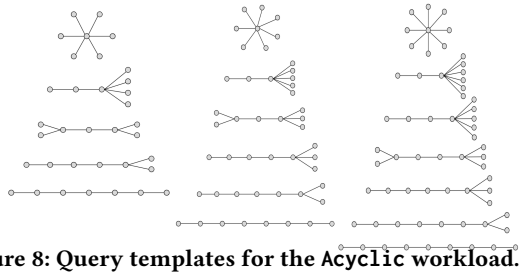


Figure 8: Query templates for the `Acyclic` workload. Edge directions are omitted in the figure.

We then transformed the JOB workload [23] into equivalent subgraph queries on our transformed graph. We removed non-join predicates in the queries since we are focusing on join cardinality estimations. This resulted in 7 join query templates, including four 4-edge query templates, two 5-edge query templates, and one 6-edge query template. Because we are focusing on edge labeled queries, for any query edge that corresponds to a join over a relationship table T_ℓ, we generated a random edge label by picking uniformly at random one of the type values of T_ℓ. We generated 100 query instances this way from each query template and removed the ones whose outputs were empty. The final workload contained 369 queries.

WatDiv [43] and `WatDiv-Acyclic` Workload: WatDiv [4] is a synthetic knowledge graph that has its own workload in SPARQL format with 12400 original queriees. We converted these queries into equivalent subgraph queries by removing their vertex predicates and we then removed queries with at most 3 edges. After removing duplicates among the remaining queries there were 75 different queries left, 9 of which is cyclic and the other 64 acyclic. Because 9 queries is very small for a workload, we use only the 64 acyclic queries call this the `WatDiv-Acyclic` workload.

YAGO 1 [46] and `G-CARE-Acyclic` and `G-CARE-Cyclic` Workloads: G-CARE [33] is a recent cardinality estimation benchmark for subgraph queries. From this benchmark we took the YAGO knowledge graph dataset and the acyclic and cyclic query workloads for that dataset. The `acyclic` workload contains 382 queries generated from query templates with 3-, 6-, 9-, and 12-edge star and path queries, as well as randomly generated trees. We will refer to this workload as `G-CARE-Acyclic`. The cyclic query workload contains 240 queries generated from templates with 6-, and 9-edge cycle, 6-edge clique, 6-edge flower, and 6- and 9-edge petal queries. We will refer to this workload as `G-CARE-Cyclic`.

DBLP (2012-11-28 dump) [9], WatDiv v.0.6 [43], Epinions [11], and Hetionet v.1.0 [18] Datasets and `Acyclic` and `Cyclic` Workloads: We used three other datasets: (i) Hetionet: a biological network; (ii) DBLP: a real knowledge graph; and (iii) Epinions: a real-world social network graph. Epinions is a dataset that by default does not have any edge labels. We added a random set of 50 edge

labels to Epinions. Our goal in using Epinions was to test whether our experimental observations also hold on a graph that is guranteed to not have any correlations between edge labels. For these datasets we created one acyclic and one cyclic query workload, which we refer to as `Acyclic` and `Cyclic`. The `Acyclic` workload contains queries generated from 6-, 7-, or 8-edge templates, shown in Figure 8. These templates are systematically picked to ensure that for each query size k, there is a pattern of every possible depth. Then, we generated 20 non-empty instances of each template by putting one edge label uniformly at random on each edge, which yielded 360 queries in total. The `Cyclic` workload contains queries generated from templates used in reference [28]. We then randomly generated instances of these queries by randomly matching each edge of the query template one at a time in the datasets. Because WatDiv's original queries contained few cyclic queries, we used the `Cyclic` workload also on WatDiv. We generated 70 queries for DBLP, 212 queries for Hetionet, 129 queries for WatDiv, and 394 queries for Epinions.

6.2 Space of Optimistic Estimators

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

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

6.2.1 Acyclic Queries and Cyclic Queries With Only Triangles. Our first question is: Which of the 9 possible optimistic estimators leads to most accurate estimates on acyclic queries and cyclic queries that contain ≤3 edges on CEG_O? We compare our 9 estimators on CEG_O for each acyclic query workload in our setup (for IMDb, WatDiv, and YAGO, using JOB, `WatDiv-Acyclic`, and `G-CARE-Acyclic` workloads). Because estimators that use avg aggregator cannot finish 12-size queries in YAGO in a reasonable time, we randomly sampled 100K paths for these estimators and take their average. As we demonstrate in our scalability experiments in Section 6.5 even the number of paths in a 9-star query is prohibitively large.

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

Our results are shown in Figure 9 (ignore the P* column for now). We make several observations. First, regardless of the path-length choice, the `max` aggregator (the last 3 box plots in the figures) makes significantly more accurate estimates (note that the y-axis on the plots are in log scale) than `avg`, which in turn is more accurate than `min`. This is true across all acyclic experiments and all datasets. For example, on IMDb and `JOB` workload, the `all-hops-min`, `all-hops-avg`, and `all-hops-max` estimators have log of mean q-errors of 6.5 (underestimation), 1.7 (underestimation), and 1.02 (understimation), respectively. *Therefore on acyclic queries, when there are multiple formulas that can be used to make an estimate, using the pessimistic ones is an effective technique to combat the well known underestimation problem.* This can give up to three orders of magnitude lower mean q-errors than, e.g., the `min-hop-min` estimator used in prior work.

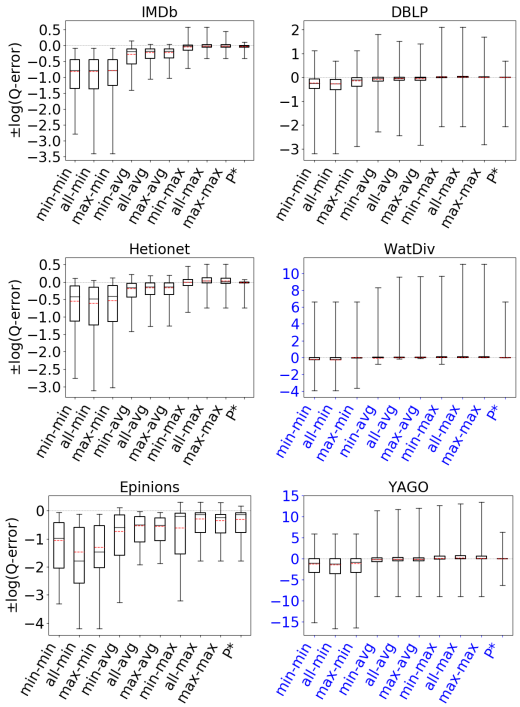


Figure 9: Evaluation of the optimistic estimators on CEG_O on acyclic queries. Estimators are labeled"P-A": P is the path length (one of max-hop, min-hop, or all-hops) and A the aggregator (one of max-aggr, min-aggr, or avg-aggr).

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

Figure 10 shows the accuracies of the 9 estimators on cyclic query workloads with only triangles. Our observations are similar to those for acyclic queries, and we find that the `max` aggregator yields more accurate estimates than other aggregators, irrespective of the path length. When using the max aggregator, we also observe that `max-hop` performs at least as well as `min-hop`. Therefore, as

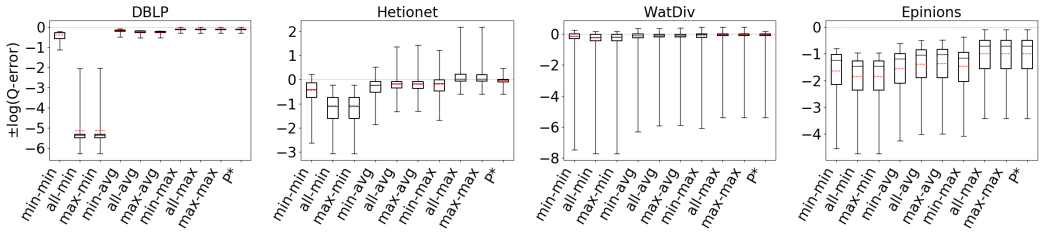


Figure 10: Evaluation of the space of optimistic estimators on CEG_O on `Cyclic` workload on queries with only triangles.

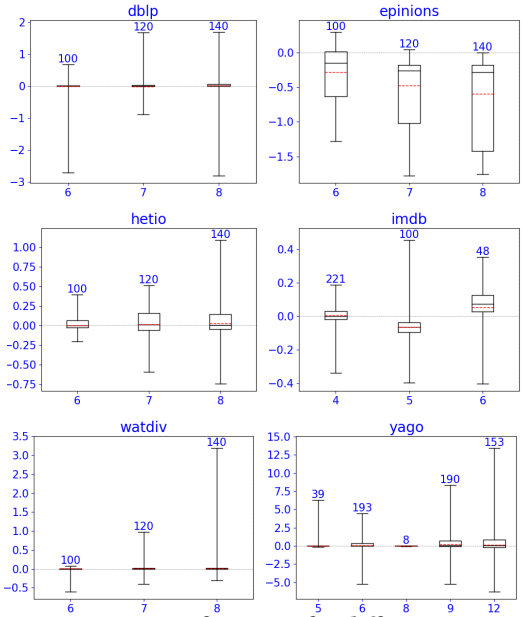


Figure 11: Accuracy of `max-max` for different query sizes (x-axis). The number on top of boxplots is the number of queries in the boxplot.

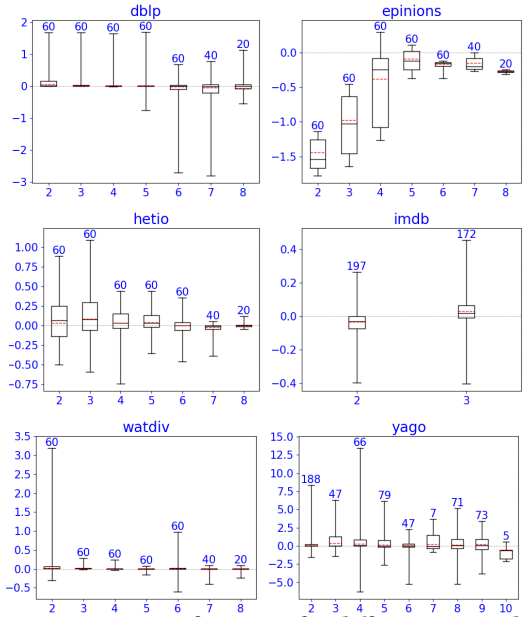


Figure 12: Accuracy of `max-max` for different query depths (x-axis). The number on top of boxplots is the number of queries in the boxplot.

we observed for acyclic queries, *we find* `max-hop-max` *estimator to be an effective way to make accurate estimations for cyclic queries with only triangles.*

6.2.2 Effects of Query Templates, Sizes, Depth, and h. For the above experiments, we also performed a more detailed analysis studying the effects of different query templates (i.e., shapes). That is, we grouped the queries in each of our workloads into different isomorphic templates, e.g., a 5-star, a 5-star with a two-path tail, etc., and re-generated the accuracies of our 9 estimators as in Figures 9 and 10 by only plotting the accuracies of one group. These constitute over 100 figures and can be found in our github repo. Broadly these figures verified that our conclusions generally hold for different acyclic and cyclic query template we used in our workloads.

We also did two further detailed analysis. First, we divided the queries in our `Acyclic` workloads based on their sizes, and analyzed the accuracy distribution of our overall recommended estimator `max-hop-max`. Figures 11 and 12 show our boxplots. Overall, we see that as the size of the queries increases accuracy decreases. This can be seen by the boxplots getting taller as the size increases. This is expected because optimistic estimators make more independence and uniformity assumptions in their formulas for larger queries. One exception is that the accuracy on YAGO 8-size queries is better than all other sizes, but this is mainly because this boxplot contains very few, only 8, queries (shown on top of boxplots), while other boxplots contains many more, e.g., over 150 queries. For the depth, no clear pattern emerges and the trend seems dependent on the dataset and workload. On Hetionet, we see a general trend that as the depth of queries increases from 2 to 8 (from stars to paths), the accuracy gets better. We do not observe a similar trend in other datasets. For example, in Epinions a similar trend exists between depth 2 and 5, but not after depth 5, where accuracies seem to get worse.

Finally, on two of our datasets, DBLP and Hetionet, we verified that increasing h from 2 to 3 improves accuracy in optimistic estimators. Figure 13 shows our results. As expected increasing h improves the accuracy of each of our estimators. For example, in Hetionet, the mean accuracy improves from 9.52 to 5.11 for `min-hop-min` and from 1.62 to 1.26 for `max-hop-max`.

6.2.3 Cyclic Queries With Cycles of Size >3. Our second question is: Which of the 9 estimators lead to most accurate estimates for cyclic queries with cycles of size > 3 on CEG_O and CEG_{OCR}? We compare our 9 estimators on CEG_O and CEG_{OCR} for each dataset-cyclic query workload combination in our benchmark, but only using queries that contain cycles of size > 3. We now expect estimates on CEG_O to be pessimistic as the formulas corresponding to bottom to top paths breaks large cycles into paths. We also expect

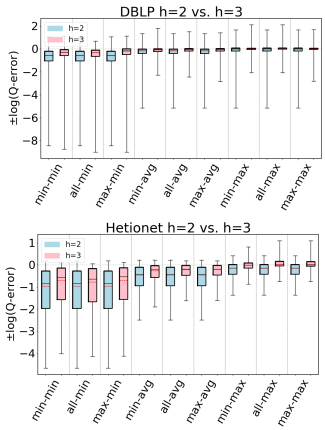


Figure 13: Accuracy of all 9 estimators on DBLP Hetionet both when h=2 and h=3.

the edge weights in CEG_{OCR} to fix this pessimism, so estimates on CEG_{OCR} can still be optimistic.

Figure 14 shows our results. As we expected, we now see that across all of the datasets, our 9 estimators on CEG_O generally overestimate. In contrast to our observations for acyclic queries and cyclic queries with only cycles, *we now see that the most accurate results are achieved when using the* `min` *aggregator (instead of* `max`*).* For the `min` aggregator, any of the path-length choices seem to perform reasonably well. On DBLP we see that `min-hop`, while on Hetionet `max-hop` and `all-hops` lead to more accurate results. The results are comparable in other datasets.

In contrast, on CEG_{OCR}, similar to our results from Figures 9 and 10, we still observe that `max` aggregator yields more accurate results, although the `avg` aggregator also yields competitive results on Hetionet and YAGO. *This shows that using cycle closing rates avoids the pessimism of CEG_O and results in optimisitic estimates and countering this optimism using the pessimistic paths is effective in making accurate estimates.* In addition, we also see that any of the path-length choices perform reasonably well.

Our third questions is: Which of CEG_O and CEG_{OCR} lead to most accurate estimations under their best performing estimators? We take `min-hop-min` as CEG_O's best estimator and take `max-hop-max` for CEG_{OCR}. We see that, except for DBLP and YAGO where the estimates are competitive, the estimates on CEG_{OCR} are more accurate. For example, on Hetionet, while the median q-error for `min-hop-min` on CEG_O is 213.8 (overestimation), it is only 2.0 (overestimation) for `max-hop-max` CEG_{OCR}. *Therefore, we observe that even using the most optimistic estimator on CEG_O may not be very accurate on cyclic queries with large cycles and modifying this CEG with cycle closing rates can fix this shortcoming.*

6.2.4 P^* Estimator and Room for Improvement. Our next question is: How much room for improvement is there for the space of optimistic estimators we considered on CEG_O and CEG_{OCR}? To do so, we consider a thought experiment in which, for each query in our workloads, an oracle picks the most accurate path in our CEGs. The accuracies of this oracle-based estimator are shown as P^* bars in our bar charts in Figures 9, 10, and 14. We compare the P^* bars in these figures with the `max-hop-max` estimator on CEG_O on acyclic queries and cyclic queries with only triangles, and `max-hop-max` estimator on CEG_{OCR} for queries with larger cycles. We find that on acyclic queries, shown in Figure 9, we generally see little room for improvement, though there is some room in Hetionet and YAGO. For example, although the median q-errors of `max-hop-max` and P^* are indistinguishable on Hetionet, the 75 percentile cutoffs for `max-hop-max` and P^* are 1.52 and 1.07, respectively. We see more room for improvement on cyclic query workloads that contain large cycles, shown in Figures 14. Although we still find that on DBLP, WatDiv and Epinions, `max-hop-max` estimator on CEG_{OCR} is competitive with P^*, there is a more visible room for improvement on Hetionet and YAGO. For example, on Hetionet, the median q-errors of `max-hop-max` and P^* are 1.48 (overestimation) and 1.02 (underestimation), respectively. On YAGO the median q-errors are 39.8 (overestimation) and 1.01 (overestimation), respectively. This indicates that future work on designing other techniques for making estimations on CEG-based estimators can focus on workloads with large cycles on these datasets to find opportunities for improvement.

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/MOLP pessimistic estimator in reference [6] and we reproduce that experiment in our context as well. To answer the question for optimistic estimators, we tested the effects of bound sketch on the `JOB` workload on IMDb and `Acyclic` workload on Hetionet and Epinions. We use h=2 in our Markov tables. Note that when h=2 only the aggregator heuristic is relevant because max-hop vs min-hop choice does not exist. We used the max aggregator and call this estimator as `max-aggr`. Then we applied the bound sketch optimization to both `max-aggr` (on CEG_O) and MOLP estimators and measured the q-errors of the estimators under partitioning budgets of 1 (no partitioning), 4, 16, 64, and 128. For the largest partitioning size of 128, our offline partitioning code that also generates Markov table entries for each partition, took 18 minutes for IMDb, 41 minutes for Hetionet, and 58 seconds for Epinions. We did not optimize this code, so these numbers can be improved with more performance-oriented implementations.

Our results are shown in Figure 15. As demonstrated in reference [6], our results confirm that bound sketch improves the accuracy of MOLP. The mean accuracy of MOLP increases between 15% and 89% across all of our datasets when moving between 1 and 128 partitions. Similarly, we also observe significant gains on the `max-aggr` estimator 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. although 93% of their q-errors strictly improve under `max-aggr`, albeit by a small amount.

6.4 Summary-based Estimator Comparison

The optimistic and pessimistic estimators we covered are summary-based estimators. A recent work [33] has compared MOLP against two other summary-based estimators, Characteristic Sets (CS) [30] and SumRDF [39]. We next reproduce and extend this comparison in our setting with the `max-aggr` optimistic estimator (so h=2) focusing on `Acyclic` workloads. CS was not competitive with SumRDF or `max-aggr` so we omit its coverage here.

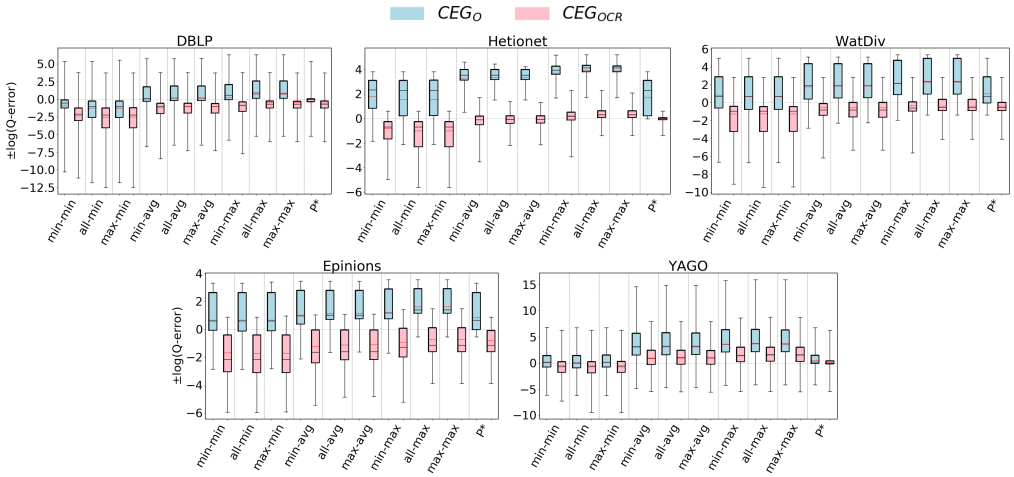


Figure 14: Evaluation of optimistic estimators on CEG_O, CEG_{OCR} on cyclic queries with cycles of 4 or more edges.

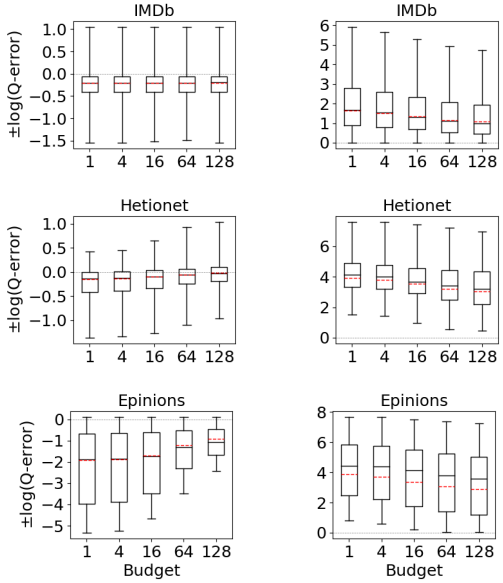


Figure 15: Effects of bound sketch on `max-aggr` estimator (left column) and MOLP (right column) estimators.

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 a form of uniformity assumption: each possible world has the same probability of representing the actual graph on which the estimate is being made.

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

Our results are shown in Figure 16. SumRDF timed out on several queries on YAGO and Hetionet. We make two observations. First, these results confirm the results from reference [33] that although MOLP does not underestimate, its estimates are very loose. Second, across all summary-based estimators, unequivocally, `max-aggr` 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-aggr`, SumRDF, MOLP, and CS are 1.8, 193.3, 44.6, and 333751, respectively. We also note that both CS and SumRDF underestimate in virtually all queries, whereas there are datasets, such as WatDiv and YAGO, where the majority of `max-aggr`'s estimates are overestimations.

6.5 Scalability of Markov tables and CEGs

We next discuss several scalability aspects of Markov tables and CEGs. Table 3 presents the size of full Markov tables for $h = 2$ and $h = 3$ for each of our datasets, which is at most 39MBs. The sizes of Markov tables for a given dataset depends on: (1) how many subgraph topologies exist in the graph for a given size, which depends on the structure of the input graph; and (2) for each subgraph topology S, which edge combinations exist. For example, if $h = 3$ and a graph has K many 3-size subgraph topologies and L many edge labels, in the worst-case, $O(KL^3)$ many entries can exist in the Markov table. The number of edge labels in our datasets are between 24 to 127, so the number of entries are not prohibitively large. In addition, on many datasets many edge combinations do not exist because of the constraints in the datasets. For example, Hetionet models a biological network about diseases and by construction there is no 2-path with both `treats` labels, because drugs treat diseases and diseases do not further treat any other entity.

We note that at least for h=2 and h=3, these Markov tables are very space-efficient for these datasets. Amongst our datasets, Epinions has the largest Markov tables. This is expected because the edge labels in this graph are synthetic and randomly assigned so more edge combinations exist for each subgraph topology. We also tried another dataset, DBPedia [10], which we do not use in our study but with two orders of magnitude more edge labels (39.6K). This is a popular knowledge graph extracting encyclopedic information from Wikipedia. Constructing a Markov table of size 3 is not practical. For example there is a vertex in DBPedia whose edges cover 11323 different labels, so simply due to this vertex, there

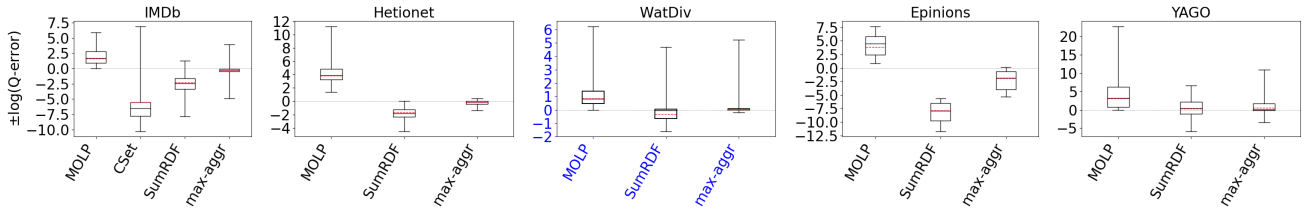


Figure 16: Summary-based estimation technique comparison.

would be 11323[3] many different 3-star entries, which would already take more than 50TB storage in our implementation, which is prohibitive. At the same time on average each edge label in this graph represents a very small number of edges (on average around 10000), therefore many of the entries would have very small cardinalities. Reference [2] has described methods to increase the scalability of Markov tables, by omitting subgraph topologies with small counts or merging multiple entries into one entry. These technique or their modifications can be used to mitigate these scalability issues.

We next discuss the complexity of searching an entry E in a Markov table. For h=2 and h=3, ignoring edge directions, there are only three possible entry topologies, which are stars, paths, or triangles. We have implemented a specialized search algorithm for h=2 and h=3, which first searches for the topology and then matches the sequence of edge directions, and then the edge labels, so entry searching is constant time. The only general solution we are aware of for indexing and searching entries in larger Markov tables appears the source code [16] for the (graph catalogue) optimistic estimator of reference [28][4]. This solution creates a key from each entry E, by combining the degree information of the query vertices in E, the edge directions and then edge labels and uses this key to index entries. However, multiple entries, E_1 and E_2, in general can have the same key, in which case the code does an isomorphism test, which is not constant time. We think using Markov Tables with size h=4 or larger will have practicality issues both because the size of the Markov tables can get much larger and searching entries will be slower.

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

A critical point about the practicality of optimistic estimators is that the complexity of making an estimate depends highly on the aggregator that is used. For `max` and `min` aggregators, the complexity is commensurate *with the number of edges* in the graph. That is because CEGs are DAGs, and there is linear time dynamic-programming-based text-book shortest (or longest) path finding algorithm in DAGs [8], which is what we use in our `max-hop-max` implementation. However for the `average` aggregator, the complexity is commensurate *with the number of paths*, as it needs to investigate each path. As can be seen in Table 4, for large queries. the number of CEG paths is prohibitively large, and this aggregator is not practical.

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

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

Table 3: Markov table sizes for our datasets.

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

Table 4: CEG edges and paths for 6- to 10-stars.

6.6 Comparison Against WanderJoin

Although we studied summary-based estimators in this paper, an alternative technique 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. For completeness of our work, we compare `max-aggr` (so h=2) estimator with WanderJoin (WJ) [24, 33], which is a sampling-based estimator that was identified in reference [33] as the most efficient sampling-based technique out of a set of techniques the authors experimented with. A more recent publication [20] describe a hybrid sampling and summary-based cardinality estimation algorithm called Alley. A submodule in Alley describes an alternative sampling algorithm to WanderJoin, which we call Alley-S. In contrast to WanderJoin (described momentarily), which samples subgraphs one edge at a time, Alley-S samples subgraph patterns vertex at a time and considers the labels of all edges incident on each vertex when sampling vertices. This requires intersections of lists of vertices. We implemented Alley-S in the G-CARE framework but we found that to achieve different accuracy levels, its estimation time is slower that WanderJoin. We therefore only present our results against WanderJoin in this section.

Our goal is to answer: *What is the sampling ratio at which WJ's estimate outperforms* `max-aggr` *(on* CEG_O*) in accuracy and how do the estimation speeds of WJ and* `max-aggr` *compare at this ratio?* We first give an overview of WJ as implemented in reference [33].

WanderJoin: WJ is similar to the index-based sampling described in reference [22]. 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^*, WJ computes the join one query-edge at-a-time by picking one of the edges of a vertex that has already been matched uniformly at random. Then, if the join is successful, a correction factor depending on the degrees of the nodes that were extended is applied to get an estimate for the number of outputs 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 [33]. We integrated the `max-hop-max` estimator into G-CARE and used the WJ code that was provided. We compared WJ and `max-aggr` 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-Acyclic` 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 also report the average q-error of WJ. 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.

Our results are shown in Figure 17. We identify the sampling ratios in which the mean accuracy of WJ is better than the mean accuracy of `max-aggr`, except in DBLP and Hetionet, where both `max-aggr` 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-aggr`. 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-aggr` except on our smallest dataset Epinions, where the difference is 1.95x. Observe that `max-aggr`'s estimation times are very stable and consistently in sub-milliseconds, between 0.18ms and 0.54ms. This is because `max-aggr`'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 the goal of our experiment is not to suggest `max-aggr` or WJ is better than the other. These are two fundamentally different classes of estimators, providing systems with different tradeoffs. However, we believe that our 'competitive sampling ratio' analysis is informative for interested readers.

6.7 Impact on Plan Quality

Reference [21] established that cardinality estimation is critical for optimizers to generate good plans for RDBMSs as it leads to better plan generation. Several other work has verified this in different contexts, in RDBMSs [6] and in RDF systems [33]. In our final set of experiments we set out to verify this in our context too by comparing the impact of our estimators on plan quality. We used the RDF-3X system [31] and its source code available here [38]. We issued our `Acyclic` workload as join-only queries to RDF-3X on the DBLP and WatDiv datasets. We then ran the query under 10 configurations: first using RDF-3X's default estimator and then by injecting the cardinality estimates of our 9 optimistic estimators to the system. We used h=3 in our Markov tables to be able to differentiate between max-hop and min-hop path choices in these estimators. The cardinalities are injected inside the system's dynamic programming-based join optimizer. We then filtered out the queries in which all of the 10 estimators lead to exactly the same plan. We were left with 41 queries for DBLP and 25 queries for WatDiv. We ran each query 5 times and report the best execution time.

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

Figure 18 shows the runtimes of the system under each configuration where the y-axis shows the log-scale speedup or slow down of each plan under each estimator compared to the plans under the default RDF-3X estimator. Although a visible improvement is not identified in DBLP, on WatDiv, observe that the median lines of the 9 estimators are above 0, indicating the each of these estimators, which have more accurate estimates than RDF-3X's default estimator, leads to better plan generation. In addition, observe that the box plot of estimators with the `max` aggregators are generally better than estimators that use the `min` or `avg` aggregator. This correlates with Figure 9, where we showed these estimators lead to more accurate estimations. We then performed a detailed analysis of the `max-hop-max` and `min-hop-min` estimator as representative of, respectively, the most and least accurate of the 9 estimators. We analyzed the queries in which plans under these estimators differed significantly. Specifically, we found 10 queries across both datasets where the runtime differences were at least 1.15x. Of these, only in 1 of them `min-hop-min` lead to more efficient plans and by a factor of 1.21x. In the other 9, `max-hop-max` lead to more efficient plans, by a median of 2.05x and up to 276.3x, confirming our expectation that better estimations generally lead to better plans.

7 RELATED WORK

We cover a part of the extensive database literature on cardinality estimation focusing on work on graph-based database management systems, specifically XML and RDF systems.

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

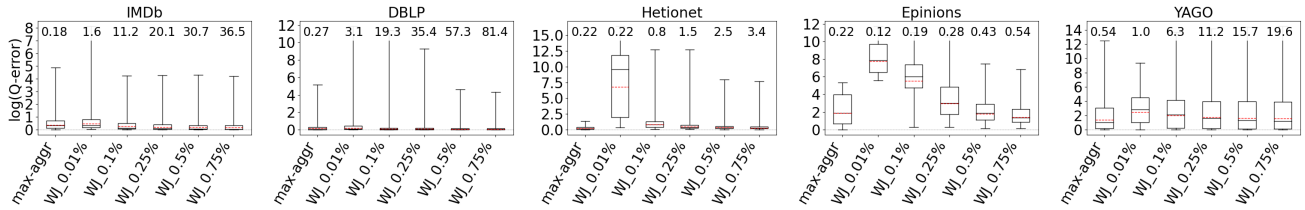


Figure 17: Comparison of `max-aggr` and WJ, with average estimation times (in milliseconds) indicated at the top of boxes.

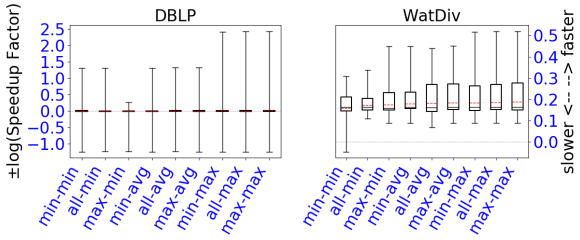


Figure 18: RDF-3X runtimes on Acyclic workload.

in columns, or histograms [3, 29, 36], wavelets [27], or probabilistic and statistical models [13, 40] 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 works 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 [48] and XSketch [34] build a sketch S of the input XML Document. The sketch of the graph effectively collapses multiple nodes and edges into supernodes and edges with metatadata on the nodes and edges. The metadata contains statistics, such as the number of nodes that was collapsed into a supernode. Then given a query Q, Q is matched on S and using the metadata an estimate is made. Because these techniques do not decompose a query into smaller sub-queries, the question of which decomposition to use does not arise for these estimators.

Several work use data structures that are adaptations of histograms from relational systems to store selectivities of paths or trees in XML documents. Examples include, *positional histograms* [45] and *Bloom histogram* [42]. 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* [32] databases) for purposes other than cardinality estimation. For example, *TreeSketch* [35] 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 [17, 22, 24, 41, 44]. 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 [23] that use sampling-based estimators to estimate the selectivities of predicates in base tables but not on multiway joins.

Reference [7] describes a sampling algorithm for cyclic join queries with theoretically guarantees on estimation time and accuracy. In another reference [37], one of the authors of reference [7] along with others have designed a sampling algorithm for select-project-join queries with for the case when the projection contains a single attribute with theoretical guarantees. As we briefly discussed in Section 6.6, Alley [20] is another novel hybrid algorithm that contains an alternative vertex-at-a-time (in relational terms column at a time) sampling for sampling results of join queries but enhances it with a summary-based index that stores information about some sub-queries. As we mentioned in Section 6.6, we implemented the sampling component of Alley in G-CARE and call it Alley-S and compared it against WanderJoin but found that WanderJoin achieves different accuracy levels with faster estimation times. That is why we use WanderJoin in our experiments.

The Maximum Entropy Estimator: Markl et al. [26] 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 [26] 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

Aside from capturing existing optimistic and pessimistic estimators, we believe the CEG framework can be the foundation to develop

novel estimators. In addition to the three CEGs we considered here (and a fourth one that is used for theoretical purposes in Appendix D) many other CEGs using different statistics can be defined and paired with different techniques to pick paths to develop new estimators. For example, one can use variances or entropies of the distributions of small-size joins as edge weights, possibly along with degree statistics, and pick the lowest entropy paths. An important research direction is to systematically study a class of CEG instances that use different combination of statistics as edge weights, as well as techniques for picking paths, to understand which statistics lead to more accurate results in practice. An important limitation of our work is that our evaluation of different optimistic estimators is purely empirical. A second important direction is to theoretically understand the dataset properties, e.g. types of edge-label correlations, that explain why one estimator outperforms another is an important future work direction.

REFERENCES

[1] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2016. Computing Join Queries with Functional Dependencies. In *PODS*.
[2] Ashraf Aboulnaga, Alaa R. Alameldeen, and Jeffrey F. Naughton. 2001. Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. In *VLDB*.
[3] Ashraf Aboulnaga and Surajit Chaudhuri. 1999. Self-Tuning Histograms: Building Histograms Without Looking at Data. In *SIGMOD*.
[4] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified Stress Testing of RDF Data Management Systems. In *ISWC*.
[5] A. Atserias, M. Grohe, and D. Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SICOMP* 42, 4 (2013).
[6] Walter Cai, Magdalena Balazinska, and Dan Suciu. 2019. Pessimistic Cardinality Estimation: Tighter Upper Bounds for Intermediate Join Cardinalities. In *SIGMOD*.
[7] Yu Chen and Ke Yi. 2020. Random Sampling and Size Estimation Over Cyclic Joins. In *ICDT*, Vol. 155.
[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms* (2 ed.). The MIT Press.
[9] dblp 2012. dblp 2012-11-28 Dump. https://dblp.org/. (2012).
[10] DBPedia 2015. DBPedia 3.5. https://downloads.dbpedia.org/3.5/en/. (2015).
[11] Epinions 2003. Epinions. https://snap.stanford.edu/data/soc-Epinions1.html. (2003).
[12] [Omitted for double-blind submission]. 2021 https://rb.gy/b9sbwy. *Accurate Summary-based Cardinality Estimation Through the Lens of Cardinality Estimation Graphs.* Technical Report.
[13] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation Using Probabilistic Models. In *SIGMOD*.
[14] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *JACM* 59, 3 (2012).
[15] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *J. ACM* 59, 3 (June 2012).
[16] Graphflow 2019. Graphflow Source Code. https://tinyurl.com/wyuzb9pr. (2019).
[17] 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).
[18] Hetionet 2015. Hetionet v1.0. https://het.io/. (2015).
[19] Manas Joglekar and Christopher Ré. 2018. It's All a Matter of Degree - Using Degree Information to Optimize Multiway Joins. *TOCS* 62, 4 (2018).
[20] Kyoungmin Kim, Hyeonji Kim, George Fletcher, and Wook-Shin Han. 2021. Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation. In *SIGMOD*.
[21] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really?. In *VLDB*.
[22] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *CIDR*.
[23] 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).
[24] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In *SIGMOD*.
[25] Angela Maduko, Kemafor Anyanwu, Amit Sheth, and Paul Schliekelman. 2008. Graph Summaries for Subgraph Frequency Estimation. In *ESWC*.
[26] 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).
[27] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD*.
[28] Amine Mhedhbi and Semih Salihoglu. 2019. Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins. *PVLDB* 12, 11 (2019).
[29] M. Muralikrishna and David J. DeWitt. 1988. Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries. In *SIGMOD*.
[30] Thomas Neumann and Guido Moerkotte. 2011. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *ICDE*.
[31] Thomas Neumann and Gerhard Weikum. 2008. RDF-3X: A RISC-Style Engine for RDF. In *VLDB*.
[32] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. 1995. Object Exchange Across Heterogeneous Information Sources. In *ICDE*.
[33] 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*.
[34] Neoklis Polyzotis and Minos Garofalakis. 2002. Statistical Synopses for Graph-Structured XML Databases. In *SIGMOD*.
[35] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. 2004. Approximate XML Query Answers. In *SIGMOD*.
[36] Viswanath Poosala and Yannis E. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *VLDB*.
[37] Yuan Qiu, Yilei Wang, Ke Yi, Feifei Li, Bin Wu, and Chaoqun Zhan. 2021. Weighted Distinct Sampling: Cardinality Estimation for SPJ Queries. In *SIGMOD*.
[38] rdf3x 2020. RDF-3X Source Code. https://github.com/gh-rdf3x/gh-rdf3x/. (2020).
[39] Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev. 2018. Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation. In *WWW*.
[40] 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*.
[41] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P. Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. In *VLDB*.
[42] Wei Wang, Haifeng Jiang, Hongjun Lu, and Jeffrey Xu Yu. 2004. Bloom Histogram: Path Selectivity Estimation for XML Data with Updates. In *VLDB*.
[43] WatDiv 2014. WatDiv v.0.6. https://dsg.uwaterloo.ca/watdiv/. (2014).
[44] Wentao Wu, Jeffrey F. Naughton, and Harneet Singh. 2016. Sampling-Based Query Re-Optimization. In *SIGMOD*.
[45] Yuqing Wu, Jignesh M. Patel, and H. V. Jagadish. 2002. Estimating Answer Sizes for XML Queries. In *EDBT*.
[46] YAGO 2008. YAGO 1. https://yago-knowledge.org/downloads/yago-1. (2008).
[47] C. T. Yu and M. Z. Ozsoyoglu. 1979. An Algorithm for Tree-query Membership of a Distributed Query. In *COMPSAC*.
[48] Ning Zhang, M. Tamer Ozsu, Ashraf Aboulnaga, and Ihab F. Ilyas. 2006. XSEED: Accurate and Fast Cardinality Estimation for XPath Queries. In *ICDE*.

A 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 [6]), there is a path in $MOLP_C$. For a detailed overview of these algorithms, we refer the reader to reference [6]. 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., $\mathcal{R}_C = \{R_i(\mathcal{A}_i) | R_i \text{ covers } |\mathcal{A}_i| \text{ attributes}\}$, and $\mathcal{R}_E$, which cover all but one of their attributes, $\mathcal{R}_E = \{R_i(\mathcal{A}_i) \text{ s. t. } R_i \text{ covers } |\mathcal{A}_i|-1 \text{ attributes}\}$. We ignore the relations that do not cover any attributes. Let relation in $\mathcal{R}_C = R_{C1}(\mathcal{A}_{C1}), \ldots, R_{Ck}(\mathcal{A}_{Ck})$ and those in $\mathcal{R}_E = R_{E1}(\mathcal{A}_{E1}), \ldots, R_{Ek'}(\mathcal{A}_{Ek'})$. 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 $U = (\cup_{i=1,\ldots k} \mathcal{A}_{Ci})$. For example this path can be: $P_1 = (\emptyset) \xrightarrow{\log(|R_{C1}|)} (\mathcal{A}_{C1}) \xrightarrow{\log(|R_{C2}|)} (\mathcal{A}_{C1} \cup \mathcal{A}_{C2}) \ldots \xrightarrow{\log(|R_{Ck}|)} (U)$. 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 [47] 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).

B COUNTER EXAMPLE FOR USING THE CBS ESTIMATOR ON CYCLIC QUERIES

Consider the triangle query $R(a,b) \bowtie S(b,c) \bowtie T(c,a)$. FCG would generate the cover $a \rightarrow R$, $b \rightarrow S$, and $c \rightarrow T$. For this cover, BFG would generate the bounding formula: $h(a,b,c) \leq h(a|b) + h(b|c) + h(c|a)$, which may not be a pessimistic bound. As an example, suppose each relation R, S, and T contains n tuples of the form $(1,1)...(n,n)$. Then this formula would generate a bound of 0, whereas the actual number of triangles in this input is n.

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 used our JOB workload on IMDb and `Acyclic` worklaod on DBLP, Hetionet, WatDiv, and Epinions. We used 128 partitions when using the bound sketch optimization. We re-present the performance of the optimistic `max-max` estimator for reference in the figure. Our results are shown in Figure 19. Overall, we see small improvement in accuracy except in Hetionet, where we see 36% improvement in the mean accuracy compared to MOLP. This observation is important as it provides evidence that systems using a pessimistic estimator might prefer using a fast combintorial solver for MOLP (using CEG_M) instead of using a slow numerical solver for CLLP, 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 [19]. DBPLP is not as tight as MOLP, 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, 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 $P = \emptyset \xrightarrow{e0} X_0 \xrightarrow{e1} ... \xrightarrow{ek} \mathcal{A}$. 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 $\sum_{a\in\mathcal{A}} \geq |P|$. In other words, each $(\emptyset, \mathcal{A})$ path identifies a subset of the constraints $c_1, ..., c_k$ that do not share the same variable v_a twice, so summing these constraints yields the constraint $\sum_{a\in\mathcal{A}} \geq |P|$. 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.*

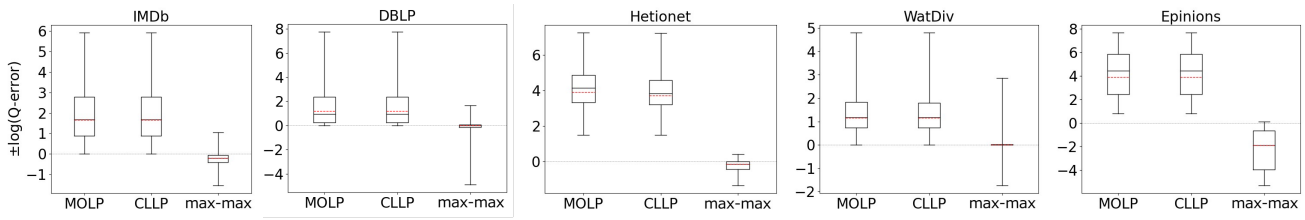


Figure 19: Q-error distribution of `max-hop-max` optimistic, MOLP, and CLLP on `JOB` on IMDb and `Acyclic` on the other datasets.

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 [19], which compares refinements of MOLP and DBPLP. Our proof serves as an alternative combinatorial proof to the inductive proof in reference [19] 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.