Join Cardinality Estimation Graphs: Analyzing Pessimistic and Optimistic Estimators Through a Common Lens

by

Jeremy Yujui Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Join cardinality estimation is a fundamental problem that is solved in the query optimizers of database management systems when generating efficient query plans. This problem arises both in systems that manage relational data as well those that manage graph-structured data where systems need to estimate the cardinalities of subgraphs in their input graphs. We focus on graph-structured data in this thesis.

A popular class of join cardinality estimators uses statistics about sizes of small size queries to make estimates for larger queries. Statistics-based estimators can be broadly divided into two groups: (i) optimistic estimators that use statistics in formulas that make degree regularity and conditional independence assumptions; and (ii) the recent pessimistic estimators that estimate the sizes of queries using a set of upper bounds derived from linear programs, such as the AGM bound, or tighter bounds, such as the MOLP bound that are based on information theoretic bounds.

In this thesis, we introduce a new framework that we call cardinality estimation graph (CEG) that can represent the estimates of both optimistic and pessimistic estimators. We observe that there is generally more than one way to generate optimistic estimates for a query, and the choice has either been ad-hoc or unspecified in previous work. We empirically show that choosing the largest candidate yields much higher accuracy than pessimistic estimators across different datasets and query workloads, and it is an effective heuristic to combat underestimations, which optimistic estimators are known to suffer from.

To further improve the accuracy, we demonstrate how hash partitioning, an optimization technique designed to improve pessimistic estimators' accuracy, can be applied to optimistic estimators, and we evaluate the effectiveness.

CEGs can also be used to obtain insights of pessimistic estimators. We show MOLP estimator [15] is at least as tight as the pessimistic estimator [6] and are identical on acyclic queries over binary relations, and the MOLP CEG offers an intuitive combinatorial proof that the MOLP bound is tighter than the DBPLP bound.

Acknowledgements

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my masters degree!

I would like to thank my advisor, Dr. Semih Salihoglu. Without his encouragement and support, I would never reach where I am today. His enthusiasm for research and desire for knowledge make him a great advisor.

I am very grateful to Dr. Ken Salem for providing technical expertise and inputs, without which, this research would not have been possible.

I would also like to thank the members of my reading committee, Dr. Ken Salem and Dr. Yaoliang Yu, for providing very helpful suggestions and insights.

Last but not least, my parents and my wife, whose love and support over the years always keep me on the right path.

Dedication

I would like to give thanks to God for past, present, and future.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

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

A prevalent technique used by existing systems to estimate cardinalities of joins is to use statistics about the base relations or outputs of small-size joins, combined with independence and uniformity assumptions to generate estimates for larger queries. We will refer to these as *optimisitc estimators*. In the relational setting, it has been demonstrated that optimistic estimators tend to *underestimate* in practice, sometimes severely [20]. Other recent work - based on worst-case optimal join size bounds [1, 5, 6, 10, 15] - has led to the development of a class of *pessimistic estimators* that are guaranteed to *avoid* underestimation.

In this thesis, we focus on the behavior of optimistic and pessimistic estimators in the context of graph database systems, where the challenge is to estimate subgraph cardinalities. Our first contribution is an empirical study of both optimisitic and pessimistic estimators, using both real and synthetic data sets and variety of query workloads. In this study, we show that optimistic estimators tend to underestimate subgraph cardinalities, as is the case in the relational setting. We also show that the bounding estimates obtained from pessimistic estimators are typically very loose. That is, they often result in substantial overestimates, with magnitudes even greater than the magnitude of optimistic

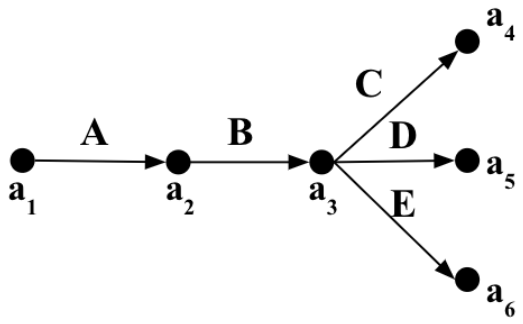


Figure 1.1: Example subgraph query Q_{5f}.

underestimates.

With these observations in hand, we revisit use of optimistic estimators for cardinality estimation. We observe that, in general, there is more than one way to generate optimistic estimates for a query, given the available statistical information. For example, consider the subgraph query in Figure 1.1. Given that we have the accurate cardinalities of all subqueries of size ≤ 2 available, there are 36 unique formulas to estimate the cardinality of the query. Examples of these formulas are:

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

In previous work on graph subgraph cardinality estimation, the choice of which of these estimates to use has either been ad hoc or has been left unspecified. Relational query optimizers have a similar problem when faced with queries involving multiple predicates, including join queries [25].

In this thesis, we consider the space of possible optimistic estimates that an optimizer could use for a given query. We represent the space as a *cardinality estimation graph* (CEG), in which each bottom-to-top path represents a different estimate. Given a CEG, a heuristic technique for combatting underestimation in optimistic estimators to base the estimate on the *longest path* through the CEG. We show empirically that the accuracy of such estimates is very good. Furthermore, these estimates are simpler to compute than pessimistic estimates, since they do not involve solving a linear program.

In addition, we review an optimization technique called *hash partitioning*. Although originally designed for pessimistic estimators, we show that the technique can also be

applied to optimistic estimators. We will show empirically the effects of hash partitioning on optimistic estimators.

Finally, we use the CEG representation to provide some insight into proposed pessimistic estimators. We show that the MOLP estimator of Joglekar and Re [15] is at least as tight as the pessimistic estimator proposed by Cai et al [6] and are identical on acyclic queries over binary relations. We also show that the MOLP bound can be interpreted as the selection of a bottom-to-top path through a CEG, although the MOLP CEG differs from the CEG used for the optimistic estimators. The MOLP CEG offers an intuitive combinatorial proof that the MOLP bound is tighter than the DBPLP bound proposed by the same authors.

The remainder of the thesis is structured as follows. Chapter 2 explains the notations we use throughout the thesis using our running example. In Chapter 3, we introduce cardinality estimation graph (CEG) and illustrate how it is used to represent cardinality estimates. The optimistic and pessimistic estimators are described, with their estimates represented in CEGs, in Chapter 4 and 5. We evaluate our approach to combat underestimation, compare optimistic and pessimistic estimators, and show the effects of the refinements empirically in Chapter 6. Other related work is reviewed in Chapter 7. Finally, we review our contributions and describe future directions for this area of research in Chapter 8.

Chapter 2

Notation and Running Example

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 and used in this thesis involve structural queries that are used frequently in graph database systems. In that setting, each R_i is a binary relation containing a subset of the edges in a graph as source/destination pairs. Figure 2.1 presents an example showing a graph with edge labels A, B, C, D, and E, shown in black, orange, green, red, and blue. This graph can be represented using five binary relations, one for each of the edge labels. These relations are also shown in Figure 2.1.

We will often represent structural queries over such relations using a graph notation. For example, consider the relations A and B from Figure 2.1. We will represent the query $Q(a_1, a_2, a_3) = A(a_1, a_2) \bowtie B(a_2, a_3)$ as

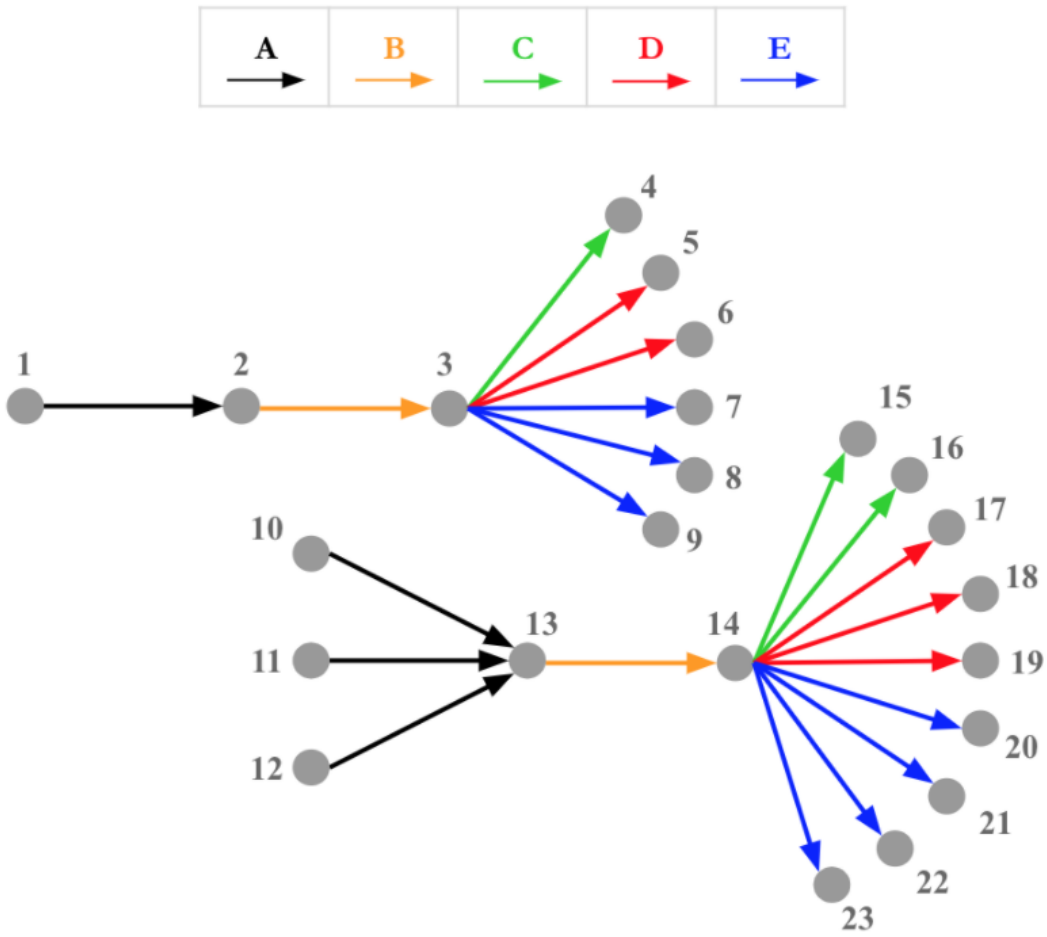
$$a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$$

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

$$a_1 \xrightarrow{A} a_2 \xleftarrow{B} a_3$$

Let $\mathcal{X}$ be a subset of the attributes $\mathcal{A}_i$ of some relation R, and let v be a possible value of $\mathcal{X}$. The *degree* of v in R is the number of times v occurs in R, i.e. $deg(\mathcal{X}(v), R) = |\{t \in R | \pi_{\mathcal{X}}(t) = v\}|$. For example, in Figure 2.1, $deg(s(14), D) = 3$ because the outgoing

D-degree of vertex 14 is 3. Similarly $deg(d(13), A)$ is also 3 because the incoming A-degree of vertex 13 is 3. We also define $deg(\mathcal{X}, R)$ to be the maximum degree in R of any value v over X, i.e., $deg(X, R) = \max_v deg(\mathcal{X}(v), R)$. So, $deg(d, A) = 3$ because vertex 13 has an incoming A-degree of 3 and this is the maximum incoming A-degree of any vertex in the dataset. The notion of degree can be generalized to $deg(X(v), Y, R)$, which refers to the "degree of a value v over attributes X in $\pi_Y R$", which counts the number of times v occurs in $\pi_Y(R)$. Similarly, we let $deg(X, Y, R) = \max_v deg(X(v), Y, R)$, i.e., $deg(X, Y, R)$ is the maximum degree of any X value in $\pi_Y R$. We will use the more general degree information when we describe pessimistic estimators.



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 2.1: Example dataset in graph and relational formats.

Chapter 3

Cardinality Estimation Graphs

We next introduce *cardinality estimation graphs* (CEGs), which we use throughout the thesis for describing cardinality estimators. Here, our goal is to give some intuition for CEGs and to illustrate how they are used to represent cardinality estimates. We will define two specific types of CEGs more precisely in Sections 4 and 5.

A CEG for a query $Q(\mathcal{A}) = R_1(\mathcal{A}_1), \ldots, R_m(\mathcal{A}_m)$ consists of the following:

- Nodes representing sub-queries of Q. For optimistic estimators (Section 4), the sub-queries will correspond to subsets of the relations involved in Q. For the pessimistic estimators (Section 5), the sub-queries will correspond to projections of Q onto subsets of its attributes. Not all possible sub-queries will be represented in a query's CEG, although it will always include $\emptyset$ and Q.
- Weighted edges from "sub" nodes to "super" nodes, e.g., for optimisitic CEGs the edges run from a given node to some or all of the nodes representing its super-queries. Edges are labled with *extension rates*, which represent the cardinality of the "super" node relative to that of the "sub" node.

Figure 3.1 illustrates a CEG for the query Q_{5f} shown in Figure 1.1 over the relations shown in Figure 2.1, assuming that statistics are available for any size-2 subqueries of Q_{5f}. Each bottom-to-top path, represents a different way of generating a cardinality estimate for Q_{5f} by combining statistics about its subqueries. For example, the leftmost path starts with the known cardinality of $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3$. Then, it extends to $a_1 \xrightarrow{A} a_2 \xrightarrow{B} a_3 \xrightarrow{C} a_4$ and then extends to the subquery of 4-fork involving the relations of A, B, C, and D, by incorporating information about the cardinality of $a_2 \xrightarrow{B} a_3 \xrightarrow{C} a_4$ and $a_2 \xrightarrow{B} a_3 \xrightarrow{D} a_5$, respectively. Finally, it extends to the full query by utilizing the cardinality of $a_2 \xrightarrow{B} a_3 \xrightarrow{E}$

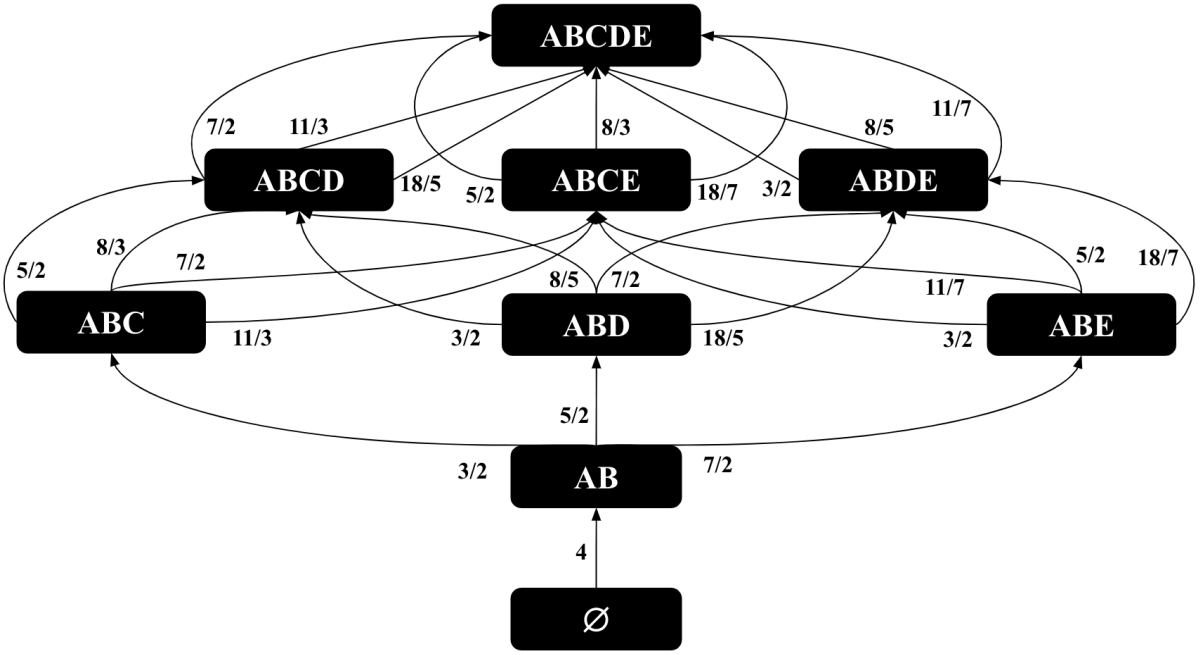


Figure 3.1: A CEG for query Q_{5f} from Figure 1.1. Each node is labeled with the relations involved in its (sub)query.

a_6. In CEGs for optimisitic estimators, the cardinality estimate along each path is the product of the edge weights along the path.

For optimisitic estimators, we will show in Section 4 that CEGs are a convenient representation of a space of candidate cardinality estimates that can be generated for Q given some summary statistics about Q's sub-queries. We will also argue that deliberately choosing a pessimistic (i.e., large) candidate from this space is an effective way to combat the underestimation problem. In Section 5, we will also show that, perhaps surprisingly, the pessimistic estimator described by Cai et al. [6] can be interpreted in terms of estimates along paths in a CEG.

Chapter 4

Optimistic Estimators

The estimators that we refer to as *optimistic* in this paper are based on formulas that use known statistics about the input database and estimate the cardinalities of a sequence of sub-queries, where the last sub-query is the original query. These formulas often make uniformity and independence or conditional independence assumptions. The cardinality estimators of many systems fall under this category. We focus on three estimators: *Markov tables* [2] from XML databases, graph summaries [24] from RDF databases, and the graph catalogue estimator of the Graphflow system [27] for managing property graphs. These estimators are extensions of each other and use as statistics the cardinalities of small-size joins. We give an overview of these estimators and then describe their CEGs, which we will refer to as CEG_{opt}, and then describe a space of possible optimistic estimates that an optimistic estimator can make. Related work (Section 7) covers other estimators that also fall under optimistic estimators, such as *characteristic sets* [29] from the RDF-3X system or those from some relational systems [38].

4.1 Overview

We begin by giving an overview of the Markov tables estimator [2], which was used to estimate the cardinalities of paths in XML documents. A Markov table of length $h \geq 2$ stores the cardinality of each path in an XML document's element tree up to length h and uses these to make predications for the cardinalities of longer paths. Table 4.1 shows a subset of the entries in an example Markov Table for $h = 2$ for our running example dataset shown in Figure 2.1. The formula to estimate a 3-path using a Markov Table with $h = 2$ is to multiply the cardinality of the leftmost 2-path with the consecutive 2-path divided by the

Path	**|Path|**
...	...
$\overset{B}{\rightarrow}$	2
$\overset{A}{\rightarrow}\overset{B}{\rightarrow}$	4
$\overset{B}{\rightarrow}\overset{C}{\rightarrow}$	3
...	...

Table 4.1: Example Markov Table for h=2.

cardinality of the common edge. For example, consider the query $Q_{3p} = \overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}$ against the dataset in Figure 2.1. The formula for Q_{3p} would be: $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}| \times \frac{|\overset{B}{\rightarrow}\overset{C}{\rightarrow}|}{|\overset{B}{\rightarrow}|}$. 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 $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}|$. For $Pr(C|B)$ the formula makes a uniformity assumption that the number of C edges that each B edge extends to is equal for each B edge and is $\frac{|\overset{B}{\rightarrow}\overset{C}{\rightarrow}|}{|\overset{B}{\rightarrow}|}$. The result of this formula is $4 \times \frac{3}{2} = 6$, which is an underestimation of the true cardinality of 7. The graph summaries [24] for RDF databases and the graph catalogue estimator [27] for property graphs have extended the contents of what's stored in Markov tables, respectively, to other acyclic joins, e.g., stars, and cyclic joins, e.g., triangles, but use the same uniformity and conditional independence assumptions.

4.2 Space of Possible Optimistic Estimators

We next represent the estimates of optimistic estimators in a CEG that we call CEG_{opt}. This will help us describe the space of possible estimations that can be made with these optimistic estimators. We assume that the given query Q is connected. CEG_{opt} consists of the following:

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

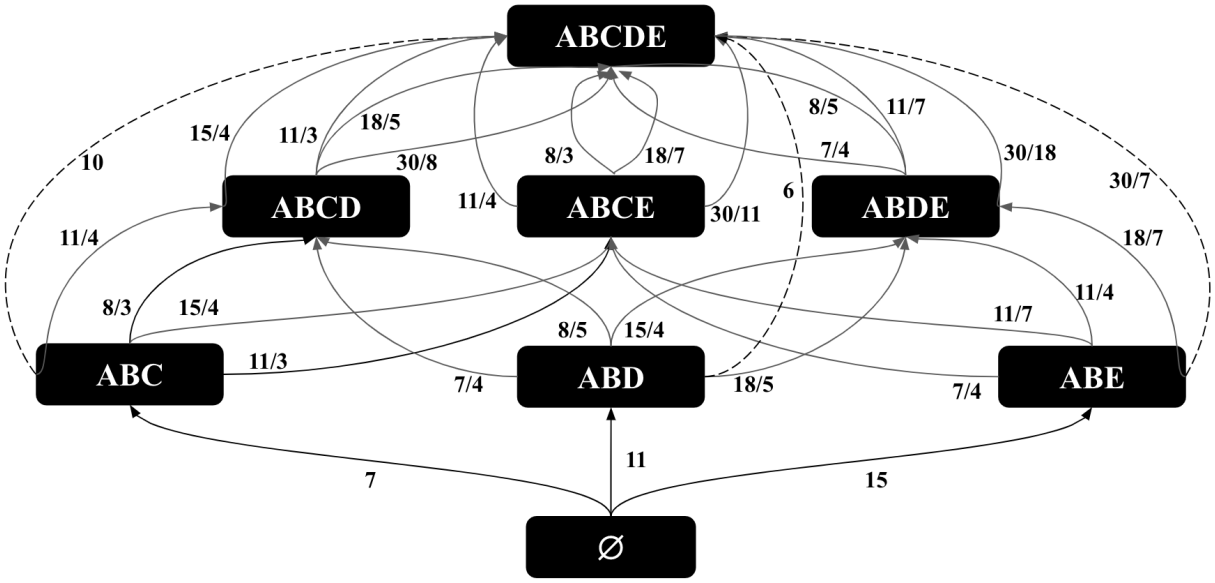


Figure 4.1: CEG_{opt} for catalogue $h = 3$ for query Q_{5f} in Figure 1.1.

intersection, be $\mathcal{E} \cap S$. Then there is an edge with weight $\frac{|\mathcal{E}|}{|\mathcal{I}|}$ from S to S' in CEG_{opt}.

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

However, when the query is not a path and/or Markov table contains joins of size ≥ 3, different $(\emptyset, \mathcal{R})$ paths in CEG_{opt} can lead to different estimates. Consider our running example's CEG shown in Figure 3.1. There are 36 $(\emptyset, \mathcal{R})$ paths leading to 7 different

estimates. An example of these estimates are:

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

Similarly, consider the fork query Q_{5f} in Figure 1.1 and a Markov table that contains up to 3-size joins. The CEG of Q_{5f} is shown in Figure 4.1. Intuitively using the largest possible joins is advantageous, so we ignore edges that use 2-size joins in the numerator. However, there are still multiple paths in the CEG, leading to 2 different estimates:

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

Both formulas start by using $|\overset{A}{\rightarrow}\overset{B}{\rightarrow}\overset{C}{\rightarrow}|$. Then, the first "short-hop" formula makes one fewer conditional independence assumption than the "longer-hop" estimate, which is an advantage. In contrast, the first estimate also makes a uniformity assumption that conditions on a smaller size join. We can expect this assumption to be less accurate than the two assumptions made in the longer-hop estimate, which condition on 2-size joins. In general, these two formulas can lead to very different estimates. For many queries, there can be many more than 2 different estimates. For example, a star query with 4 edges using a Markov table of size 2 can have as many as 11 different estimates. For such cases, it is unclear which of the estimates would lead to more accurate estimates in practice. Therefore, any optimistic estimator implementation needs to make decisions about which formulas to use, which corresponds to picking paths in CEG_{opt}.

We identify a space of heuristics that an optimistic estimator can adopt along two parameters:

- *Path length:* The estimator can identify a set of paths to consider based on the path lengths, i.e., number of edges or hops, in CEG_{opt}, which can be: (i) maximum-hop (`max-hop`); (ii) minimum-hop (`min-hop`); or (iii) any number of hops (`all-hops`).
- *Estimate aggregator:* Among the set of paths that are considered, each path gives an estimate. The estimator then has to aggregate these estimates to derive a final estimate, for which we identify three heuristics: (i) the largest estimated cardinality path (`max-aggr`); (ii) the lowest estimated cardinality path (`min-aggr`); or (iii) the average of the estimates among all paths (`avg-aggr`).

Any combination of these two heuristics can be used to design an optimistic estimator. In prior optimistic estimators, this decision has been either unspecified or chosen in an ad-hoc manner. Specifically, the original Markov tables [2] chose the `max-hop` heuristic. In this work, each query was a path, so when the first heuristic is fixed any path in CEG_{opt} leads to the same estimate. Therefore an aggregator is not needed. Graph summaries [24] uses the `min-hop` heuristic and leaves the aggregator unspecified. Finally, graph catalogue [27] picks the `max-hop` and `min-aggr` aggregator.

4.3 Combatting Underestimation

Given the well known problem that using independence and uniformity assumptions often lead to severe under-estimations, intuitively, the most effective way to combat under-estimations would be to use the `all-hops-max-aggr` heuristic in the space we described. This estimator considers all possible estimates that can be made in CEG_{opt} and picks the one that gives the largest estimate. Therefore it is the most pessimistic of the optimistic estimators. We will test this hypothesis empirically in Section 6.

Chapter 5

Pessimistic Estimators

Join cardinality estimation is directly related to the following fundamental question: Given a query Q and set of statistics over the relations R_i, such as their cardinalities or degree information about values in different columns, what is the worst-case output size of Q? Starting from the seminal result by Atserias, Grohe, and Marx in 2008 [5], several upper bounds have been provided to this question 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 the DBP [15], MO [15] and CLLP bounds [1][1] also used maximum degrees of the values in the columns. As an example, if $R_1(a_1, a_2)$ is one of the input relations, an example degree constraint could be that any a_1 value in R_1 appears in at most 10 tuples in R_1, so each a_1 value matches with at most 10 a_2 values[2]. Instead of asking for an estimate on the size of Q on the actual input relations $R_1, ..., R_m$, the above question asks for the maximum possible size of Q across all possible input relations that are consistent with the known statistics. Therefore, any of these bounds can be used as *pessimistic estimators*. In particular, this was done in a recent work [6] in an actual estimator implementation. We refer to this as the WBS estimator, after the names of the authors.

All of these bounds use numerical solutions to linear programs (LPs) that consist of inequalities that use the known cardinality and degree statistics. We ignore the AGM bound, which is the loosest of all these bounds and refer to the other LPs as DBPLP, MOLP, and CLLP. In this section, we review these bounds and the WBS estimator and

[1] CLLP bound is a generalization of the GLVV bound [10], which focused only on functional dependencies, i.e., where degree constraints are 1, to general degrees.

[2] Note that degree constraints generalize functional dependencies, which can be seen as degree constraints of 1.

derive insights into these bounds using our CEG framework. Here is the outline of this section:

- Section 5.1 reviews the MOLP bound and shows that similar to optimistic estimators, MOLP can be represented as finding a path in a particular CEG. This shows that, surprisingly, the MOLP bound has a combinatorial solution. Using our CEG framework, we provide an alternative proof that MOLP is an upper bound to the actual output size. Our proof is combinatorial and significantly simpler than the numerical proof from reference [15].
- Section 5.2 describes how MOLP can utilize the known cardianlity and degree statistics from the results of small-size joins.
- Section 5.3 briefly reviews the CLLP bound.
- Section 5.4 reviews the WBS estimator, which was originally described as using a subset of the inequalities of the CLLP inequalities. Using our CEG framework, we show that in fact the WBS estimator is equivalent to the MOLP bound on acyclic queries on which it was evaluated in reference [6]. We also review a hash partitioning refinement of the WBS estimator from reference [6]. In our evaluations, we will show that this refinement step can also be applied to optimistic estimators.

To demonstrate another application of CEGs, Appendix C reviews the DBPLP bound provides alternative proofs that MOLP is tighter than DBPLP. Specifically, we show that while MOLP bound is equal to the length of shortest $(\emptyset, \mathcal{A})$ path in a CEG, we call CEG_M, DBPLP bound is at least the length of the longest path in CEG_M. Due to space constraints, some of our proofs are provided in the appendix. It is known $CLLP \leq MOLP \leq DBPLP$. We will use tightest of these two bounds, MOLP and CLLP, in our evaluations in Section 6.

5.1 MOLP

MOLP was defined in reference [15] as a worst-case bound on join sizes that is tighter than the AGM bound. As we will show, the WBS estimator is in fact based on this bound. Suppose a system has stored $deg(X, Y, R_i)$ statistics, for each pair of attribute subsets $X \subseteq Y \subseteq \mathcal{A}_i$ in a relation R_i. MOLP is:

Maximize $s_{\mathcal{A}}$

$s_{\emptyset} = 0$

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

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

Although the base of the logarithm was taken to be the total number of tuples across all relations in reference [15], this is not necessary and we will take it as 2. Let m_A be the optimal value of MOLP. Reference [15] has shown that 2^{m_A} is an upper bound on the size of Q. An exponent over 2 is taken because the inequalities use the logarithms of the known degree constraints, so the optimal value m_A is the logarithm of the upper bound. For example, in our running example, the optimal value of these inequalities is 96, which is an overestimate of the true cardinality of 78. It is not easy to directly see the solution of the MOLP on our running example. However, we will next show that we can represent the MOLP bound as the cost of shortest $(\emptyset, \mathcal{R})$ path in a CEG, we call CEG_M.

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

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

  With these interpretations we can now construct CEG_M.

- *Vertices:* For each $X \subseteq \mathcal{A}$ have a node. This represents the sub-query $\Pi_X Q$.
- *Extension Edges:* Add an edge with cost $\log(deg(X, Y, R))$ between any $W_1 = X \cup E$ and $W_2 = Y \cup E$, for which there is $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, R))$ inequality. Note that there can be multiple edges between X and Y corresponding to multiple degree constraints of X and Y in different relations.
- *Projection Edges:* $\forall X \subseteq Y$, add an edge with cost 0 from Y to X. These edges

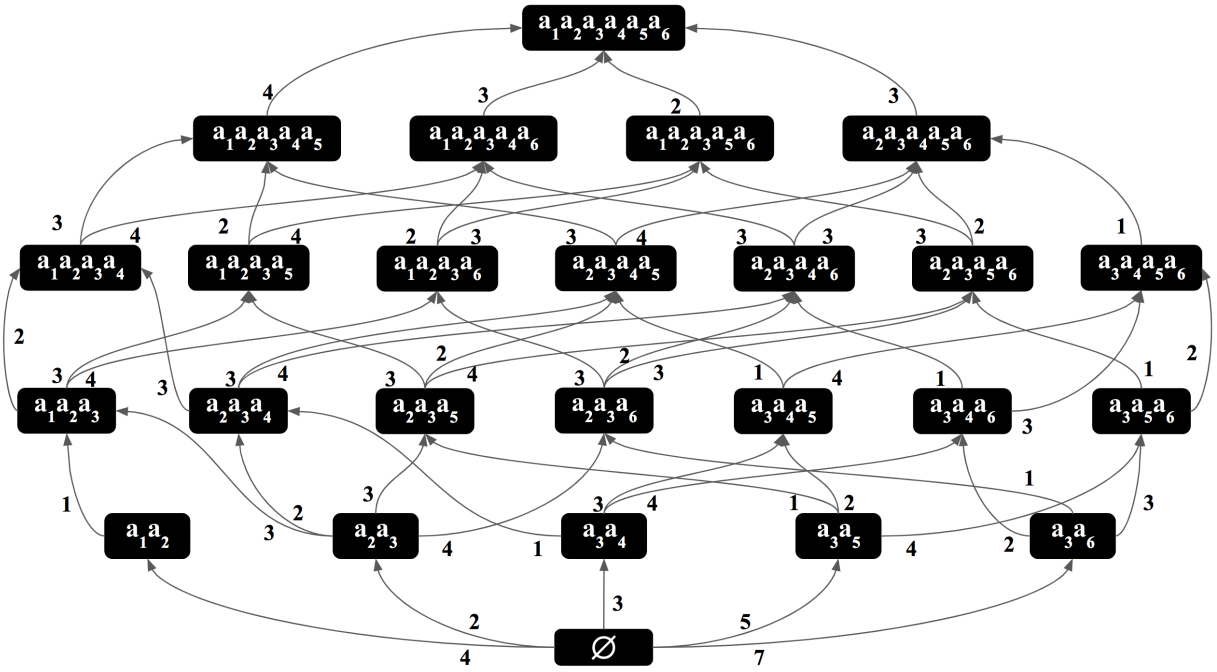


Figure 5.1: CEG_M for query Q_{5f} in Figure 1.1.

correspond to projection inequalities and intuitively indicate that, in the worst-case instances, $\Pi_Y Q$ is always as large as $\Pi_X Q$.

Figure 5.1 shows the CEG_M of our running example. We omit the projection edges for simplicity.

Theorem 5.1.1. *Let Q be a query whose degree statistics $deg(X, Y, R_i)$ for each $X \subseteq Y \subseteq \mathcal{A}_i$ is known. Let m_A be the optimal solution to the MOLP of Q. Then m_A is equal to the length of the shortest $(\emptyset, \mathcal{A})$ path in CEG_M, where length of paths is the sum of the weights on the edges.*

Proof. Our proof consists of two steps. First we show that any feasible solution v to MOLP has a value at most the length of any $(\emptyset, \mathcal{A})$ path. Then we show that a particular feasible solution, which we call v_{CEG}, is exactly the length of the shortest $(\emptyset, \mathcal{A})$ 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 $|P|$ be the length of P. Suppose w.l.o.g. that $P{=}(\emptyset)\xrightarrow{e_0}(E_1)...(E_k)\xrightarrow{e_k}(\mathcal{A})$ and for the purpose of contradiction that $|P|{=}\ w(e_0){+}....{+}w(e_k){<}s_{\mathcal{A}}$. Let $\mathcal{A}$ be E_{k+1}. If this is the case, using induction from $i{=}k{+}1$

down to 0, we can show that $w(e_0) + ... + w(e_{i-1}) < s_{E_i}$ and arrive at a contradiction. 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 length 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 length of the shortest $(\emptyset, X)$ 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 $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, R))$, so CEG_M contains edge from $X \cup E$ to $Y \cup E$ with length $\log(deg(X, Y, R))$. By definition of shortest paths, $v_{Y \cup E} \leq v_{X \cup E} + \log(deg(X, Y, R))$. 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 node Y to X with weight 0. By definition of shortest 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 weight smaller than the length of any path, $v_\mathcal{A}$ in v_{CEG}, which is the shortest $(\emptyset, \mathcal{A})$ path in CEG_M is the optimal solution to MOLP. □

With this connection, readers can verify that the MOLP bound in our running example is 96 by inspecting the paths in Figure 5.1. As we will justify momentarily, the CEG_M's we use in our figures do not include the projection edges. In this CEG, the shortest $(\emptyset, \mathcal{A})$ path has a length of 96 (specifically $\log_2(96)$), corresponding to the $(\emptyset) \xrightarrow{4} (a_1a_2) \xrightarrow{1} (a_1a_2a_3) \xrightarrow{2} (a_1a_2a_3a_4) \xrightarrow{3} (a_1a_2a_3a_4a_5) \xrightarrow{4} (a_1a_2a_3a_4a_5a_6)$ path (i.e. the leftmost path). Recall that in Figure 5.1 we put as edge weights the degrees of values, instead of their logarithms, so when inspecting shortest paths, readers should multiply the weights on the edges instead of summing them. We make three observations.

Observation 1: Similar to the CEGs for optimistic estimators, each $(\emptyset, \mathcal{A})$ path in CEG_M corresponds to a sequence of extensions from $Q_\emptyset$ to Q and is an estimate of the cardinality of Q. For example, the rightmost path $(\emptyset) \xrightarrow{7} (a_3a_6) \xrightarrow{3} (a_3a_5a_6) \xrightarrow{2} (a_3a_4a_5a_6) \xrightarrow{1} (a_2a_3a_4a_5a_6) \xrightarrow{3} (a_1a_2a_3a_4a_5a_6)$ in Figure 5.1 indicates that there are 7 a_3a_6's, each of which extends to

at most 3 many $a_3a_5a_6$'s, each of which extends to at most 2 many $a_3a_4a_5a_6$'s, each of which extends to at most 1 $a_2a_3a_4a_5a_6$'s, and finally each of which extends to at most 3 many $a_1a_2a_3a_4a_5a_6$. This yields 7x3x2x1x3=126 many possible outputs. Since we are using maximum degrees on the edge costs, each $(\emptyset, \mathcal{A})$ path is by construction an upper bound on Q. So any path in CEG_M is a pessimistic estimator. This observation is an alternative proof that MOLP is an upper bound on the actual output size, i.e., is a pessimistic estimator:

Proposition 5.1.1 (Prop. 2 [15])**.** *Let Q be a join query and OUT be the output size of Q, then $OUT \leq 2^{m_{\mathcal{A}}}$.*[3]

Proof. As argued above, for any $(\emptyset, \mathcal{A})$ path P in CEG_M, $OUT \leq 2^{|P|}$. By Theorem 5.1.1, $m_{\mathcal{A}}$ is equal to the length of the shortest $(\emptyset, \mathcal{A})$ path in CEG_M, so $OUT \leq 2^{m_{\mathcal{A}}}$. □

Observation 2: Theorem 5.1.1 implies that MOLP can be solved using a combinatorial algorithm, e.g., Dijkstra's algorithm, instead of a numeric LP solver.

Observation 3: Theorem 5.1.1 implies that we can simplify MOLP by removing the projection inequalities, which correspond to the edges with weight 0 in CEG_M. To observe this, consider any $(\emptyset, \mathcal{A})$ path $P{=}(\emptyset)\xrightarrow{e_0}(E_1)...(E_k)\xrightarrow{e_k}(\mathcal{A})$ and consider its first projection edge, say e_i. We can remove e_i and replace the rest of the edges e_{i+1} to e_k with (possibly) new edges e'_{i+1} to e'_k with exactly the same weights and construct $P'{=}(\emptyset)\xrightarrow{e_0}(E_1)...\xrightarrow{e_{i-1}}(E_i)\xrightarrow{e'_{i+1}}(E'_{i+2})....(E'_k)\xrightarrow{e'_k}(\mathcal{A})$, where $E'_{i+2} \supseteq E_{i+2}$, ..., $E'_k \supseteq E_k$. This can be seen inductively as follows. We know $E_i \supseteq E_{i+1}$ because e_i is a projection edge. Then if $(E_{i+1})\xrightarrow{e_{i+1}}(E_{i+2})$ edge was an extension edge that extended E_{i+1} to $N_{i+1} = E_{i+2} \setminus E_{i+1}$ attributes, then by construction, there is another edge $(E_i)\xrightarrow{e'_{i+1}}(E'_{i+2} = E_i \cup N_i)$ in CEG_M with the same weight as e_{i+1}. If instead e_{i+1} was a projection edge that removed a set of attributes from E_{i+1}, similarly there is another projection edge e'_{i+1} that removes the same set of attributes from E_i. So inductively, we can find an alternative sub-path from E_{i+1} to $\mathcal{A}$, $(E_{i+1})\xrightarrow{e'_{i+1}}...\xrightarrow{e'_k}(\mathcal{A})$ with the same length as the sub-path $(E_{i+1})\xrightarrow{e_{i+1}}...\xrightarrow{e_k}(\mathcal{A})$. This justifies, why we do not draw the projection edges in our CEG_M figures.

[3]This is a slight variant of Prop. 2 from reference [15], which state that another bound, called the *MO bound*, which adds a preprocessing step to MOLP, is an upper bound of OUT.

5.2 Using Degree Statistics From Results of Small Size Joins in MOLP

MOLP can directly integrate the degree statistics from results of joins. For example, if a system knows the size of the join $Q_{RS} = R(a_1, a_2) \bowtie S(a_2, a_3)$, then the MOLP can include the inequality that $s_{a_1a_2a_3} \leq \log|Q_{RS}|$. Similarly the extension inequalities can use the degree information from Q_{RS} simply by taking the output of Q_{RS} as an additional relation in the query with three attributes a_1, a_2, and a_3. For example, one would add $s_{Y \cup E} \leq s_{X \cup E} + \log(deg(X, Y, Q_{RS}))$ extension inequalities.When comparing the accuracy of the MOLP bound with optimistic estimators, we will ensure that MOLP uses the known cardinalities of the same small-size joins and the degree information from the results of these joins. This ensures that the statistics used by MOLP is a strict superset of the statistics used by optimistic estimators.

5.3 CLLP

CLLP bound extends MOLP with entropic sub-modularity inequalities (also known as the *Shannon inequalities*):

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

With the addition of sub-modularity inequalities, we cannot map the solution of CLLP to a path in the CEG framework. In Section 6, we will empirically evaluate how much extra accuracy the refinement of sub-modularity constraints give on top of the MOLP using numerical solvers.

5.4 WBS Estimator and Hash Partitioning Optimization

We review the WBS estimator very briefly and refer the reader to reference [6] for details. WBS 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 subset of the CLLP bounding formulas. A coverage

is a mapping (R_j, A_j) of a subset of the relations in the query to attributes such that each $A_j \in \mathcal{A}$ appears in the mapping. A bounding formula is a multiplication of the known degree statistics that can be used as an upper bound on the size of a query. In Appendix A, we show using our CEG framework that in fact, that the MOLP bound is at least as tight as the WBS estimator on general acyclic queries and is exactly equal to the WBS estimator over acyclic queries over binary relations, which are the queries we focus on in this paper. Therefore BFG and FCG can be seen as a method for solving the MOLP linear program on acyclic queries over binary relations, albeit in a brute force manner by enumerating all paths in CEG_M. We do this by showing that each path in CEG_M corresponds to a bounding formula and vice versa.These observations allow us to connect two worst-case upper bounds from literature using CEGs: the WBS estimator's bound and the earlier MOLP bound by Joglekar and Ré.[4]

5.5 Hash Partitioning

We next review an optimization that was described in reference [6] that is guaranteed to improve the MOLP buond. We refer to this optimization as *hash partitioning of queries*. We give an overview of the steps of the hash partitioning optimization using CEG terminology. For details, we refer the reader to reference [6]. Below, let B be the partitioning budget for the entire query. We also 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. For each attribute in S, allocate $B^{1/|S|}$ partitions. For example, consider the path $P_1 = (\emptyset) \xrightarrow{|B|} (a_2a_3) \xrightarrow{deg(a_3,C)} (a_2a_3a_4) \xrightarrow{deg(a_2,A)} (a_1a_2a_3a_4) \xrightarrow{deg(a_3,E)} (a_1a_2a_3a_4a_6) \xrightarrow{deg(a_3,D} (a_1a_2a_3a_4a_5a_6)$ in the CEG_M of Q_5f from Figure 5.1. Then both a_2

[4]Although not explicitly mentioned in reference [6], on cyclic queries, the covers that FCG generates may not be safe, i.e., the output of BFG may not be a pessimistic output. We show an example in Appendix B.

and a_3 would be in S. However for path $P_2 = (\emptyset) \xrightarrow{|A|} (a_1a_2) \xrightarrow{deg(a_2,B)} (a_1a_2a_3) \xrightarrow{deg(a_3,C)} (a_1a_2a_3a_4) \xrightarrow{deg(a_3,D)} (a_1a_2a_3a_4a_5) \xrightarrow{deg(a_3,E)} (a_1a_2a_3a_4a_5a_6)$, only a_2 would be in in S.

Step2: Partition each relation R_i as follows. Let PA_i, for partition attributes, be $PA_i = S \cap \mathcal{A}_i$ and z be $|PA_i|$. Then partition R_i into $B^{z/|S|}$ pieces using z hash functions, each hashing a tuple $t \in R_i$ based on one of the attributes in PA_i into $\{0, ..., B^{1/|S|} - 1\}$. 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 B components $Q_{0...0}$, to $Q_{B^{1/|S|}-1,...,B^{1/|S|}-1}$, 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 the join of $A_0 \bowtie B_{0,0} \bowtie C_0 \bowtie D_0 \bowtie E_0$.

The WBS estimator assumes that for each relation R_i, statistics about partitioning R_i into a set of possible partition sizes (e.g., 2, 4, 8, 16, ...) have been computed. Then, for different paths in the CEG_M's of different queries, the estimator looks up at the necessary statistics for the pre-partitioned version of R_i.

5.6 Implementing Hash Partitioning For Optimistic Estimators

Hash partitioning can also be used to refine optimistic estimators and we will evaluate its benefits in Section 6.4.1. Intuitively, one advantage of partitioning is that the tuples that hash to different buckets of the join attributes are guaranteed to not produce outputs and they never appear in the same sub-query. This can make the uniformity assumptions in the optimistic estimators more accurate because two tuples that hashed to the same bucket of the an attribute are more likely to join. However, unlike pessimistic estimators, there is no guarantee that accuracy will always improve.

In order to test whether hash partitioning helps optimistic estimators, we implemented hash partitioning for optimistic estimators as follows. Given a partitioning budget B and a set of queries in a workload, we worked backwards from the queries to find the necessary subqueries, and for each sub-query the necessary statistics that would be needed and stored them in the Markov table. For example, for the query Q_{5f} in our running example, one of the formulas that will be needed is this (using $h = 2$ Markov table): $|(a_1) \xrightarrow{A_0} (a_2) \xrightarrow{B_{00}} (a_3)| \frac{|(a_2) \xrightarrow{B_{00}} (a_3) \xrightarrow{C_0} (a_4)|}{|(a_2) \xrightarrow{B_{00}} (a_3)|} \frac{|(a_4) \xleftarrow{C_0} (a_3) \xrightarrow{D_0} (a_5)|}{|(a_3) \xrightarrow{C_0} (a_4)|} \frac{|(a_5) \xleftarrow{D_0} (a_3) \xrightarrow{E_0} (a_6)|}{(a_3) \xrightarrow{D_0} (a_5)}$, so we ensure that our Markov Table has these necessary statistics.

Chapter 6

Evaluation

6.1 Datasets and Workloads

We represent our datasets as labeled graphs and queries as subgraph queries. As discussed in Section 2, this is simply a modeling choice that makes our presentation simpler. Our datasets and queries can equivalently be represented as relational tables and SQL queries. Several prior work [18, 20] on cardinality estimation, including reference [6] has focused primarily on the IMDb dataset and the *Join Order Benchmark* (JOB) query workload (both reviewed momentarily). We used a more comprehensive suite of datasets and workloads, subsuming the IMDb dataset and join queries in the JOB workload.

6.1.1 Datasets

We used 5 real world datasets: (i) IMDb: a movie dataset; (ii) Hetionet: a biological network; (iii) DBLP a real knowledge graph; (iv) WatDiv: a synthetic knowlege graph; and (v) Epinions, a real-world social network graph. Except for IMDb, all of our datasets are by default in graph formats. IMDb is a relational database, which we converted into a property graph. Epinions by default does not have any edge labels. We added a random set of 50 edge labels to Epinions. Our goal in using Epinions was to test whether our experimental observations also hold on a graph that is guranteed to not have any correlations between edge labels. For reference our datasets are summarized in Table 6.1. We next describe how we converted the IMDb dataset into property graph.

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

Dataset	Domain	|V|	|E|	|Edge Labels|
IMDb	Movie	27,495,959	64,617,375	127
DBLP	Knowledge	23,312,916	55,586,971	27
Hetionet	Biology	45,158	2,250,197	24
WatDiv	Knowledge	1,052,571	10,916,457	86
Epinions	Social Network	75,879	508,837	50

Table 6.1: Dataset descriptions.

many to many relationships between the entities (e.g., `movie_companies` table represents relationships between movies and companies); and (iii) *type tables* that denormalize the entity or relationship tables to indicate the types of entities or relationships. We converted each row of an entity table to a vertex. We ignored vertex types because many queries in the JOB workload have no predicates on entity types. Let u and v be vertices representing, respectively, rows r_u and r_v from tables T_u an T_v. We added two sets of edges between u and v: (i) a *foreign key edge* from u to v if the primary key of row r_u (that u represents) is a foreign key in row r_v; (ii) a *relationship edge* between u to v if a row r_ℓ in a relationship table T_ℓ connects row r_u and r_v. The edge label was taken from the type attribute in r_ℓ. We picked the edge direction based on the meaning of the type. For example, if r_ℓ is a row in `movie_link` with type `follows` connecting a movie u to movie v, then we add a `follows` edge from u to v.

6.1.2 Query Workloads

We picked a set of acyclic queries that subsumes those used in reference [6] and a set of cyclic queries. We used three workloads:

JOB: We transformed the JOB query workload from reference [20] into equivalent subgraph queries on our transformed graph. We removed non-join predicates in the queries, since we are focusing on join cardinality estimations and encoded the equality predicates on types directly on edge labels. We obtained 33 join query templates, whose subgraph versions included four 3-edge, four 4-edge, two 5-edge, one 6-edge query templates (we removed 2-edge query templates). Each of these queries was acyclic. We generated up to 100 query instances from each template by putting one edge label uniformly at random on each edge, ensuring that the output of the query was non-zero. The final workload contained 609 specific query instances. We use this workload only on the IMDb dataset [6].

Acyclic: Because the queries in JOB were limited to up to 6-size joins, we created a

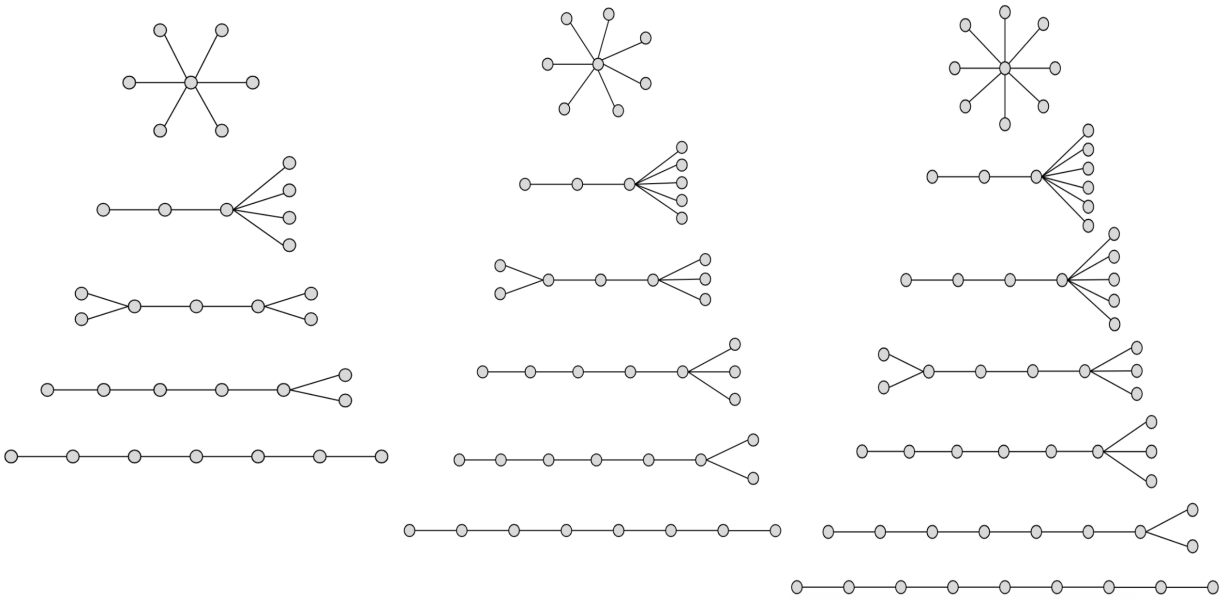


Figure 6.1: Our full acyclic query templates. The directions of the edges are neglected in the figure.

separate workload of acyclic queries containing 6-, 7-, or 8-edge templates. We ensured that for each query size k, we had patterns of every possible depth. Specifically for any k, the minimum depth of any query is 2 (stars) and the maximum is k (paths). For the set of experiments that evaluate the space of optimistic estimators, we excluded the 2- and 3-depth of the 8-edge query templates, which were quite slow to evaluate. For each depth d in between, we picked a specific pattern. Our full patterns are shown in Figure 6.1. 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.

Cyclic: We generated a workload of cyclic queries from templates used in prior work: a 5-edge diamond with a crossing edge [27], 6-edge two triangles [27], and a 5-edge lollipop query [31]. We generated 20 instances of each template.

6.2 Space of Optimistic Estimators

In our first set of experiments, we answer the question: *Which optimistic estimator in the space we defined leads to more accurate estimates?* We ran two sets of experiments. First,

we used the `JOB` workload on the IMDb dataset and the `Acyclic` workload on the rest of our datasets. Then we used the `Cyclic` workload on each of our datasets. This was to ensure that our observations are not affected by the cyclicity of queries. In order to set up an experiment in which we could test all of the 9 possible optimistic estimators, we used a Markov Table that contained up to 3-size joins (i.e., h=3), which also contained all the necesssary triangle sub-queries. The default Markov Table that contains up to 2-size joins does not allow us to test different estimators based on different path-length heuristics as each path in CEG_{opt} has the same length, which is equal to the number of edges in the query minus 1. Also observe that when $h = 2$, we also cannot estimate cyclic queries because each entry in the Markov table is a path, so an optimistic estimator can only estimate acyclic queries.

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

Our results are shown in Figures 6.2 and 6.3. We make several observations. First regardless of the path-length heuristic chosen, the `max` aggregator (the last 3 box plots in the figures) makes significantly more accurate estimates (note that the y-axis on the plots are in log scale) than `avg`, which is further more accurate than `min`. This is true across all acyclic and cyclic experiments and all datasets. For example, on IMDb and `JOB` workload, the `all-hops-min`, `all-hops-avg`, and `all-hops-max` estimators have mean q-errors (after removing top 10 percent outliers), respectively, of 6.49, 1.65, and 1.01. The differences is larger on the `Cyclic` workload, where their q-errors, respectively, are 1967.62, 5.54, and 2.36. Therefore, using the pessimistic of the optimistic estimates leads to significantly more accurate estimations in our evaluations. Reference [27] had provided an intuition for using the `min` aggregator in their optimistic estimator. The intuition was based on an extreme example, where if one possible extension, i.e. path in CEG_O, gives an estimate of 0 (i.e., when one sub-query has empty output), then the full query is guaranteed to be empty. Although this is certainly true in this extreme case, in our evaluations which contains queries with outputs, this intuition does not hold.

We next analyze the path-length heuristics. Observe that across all experiments, if

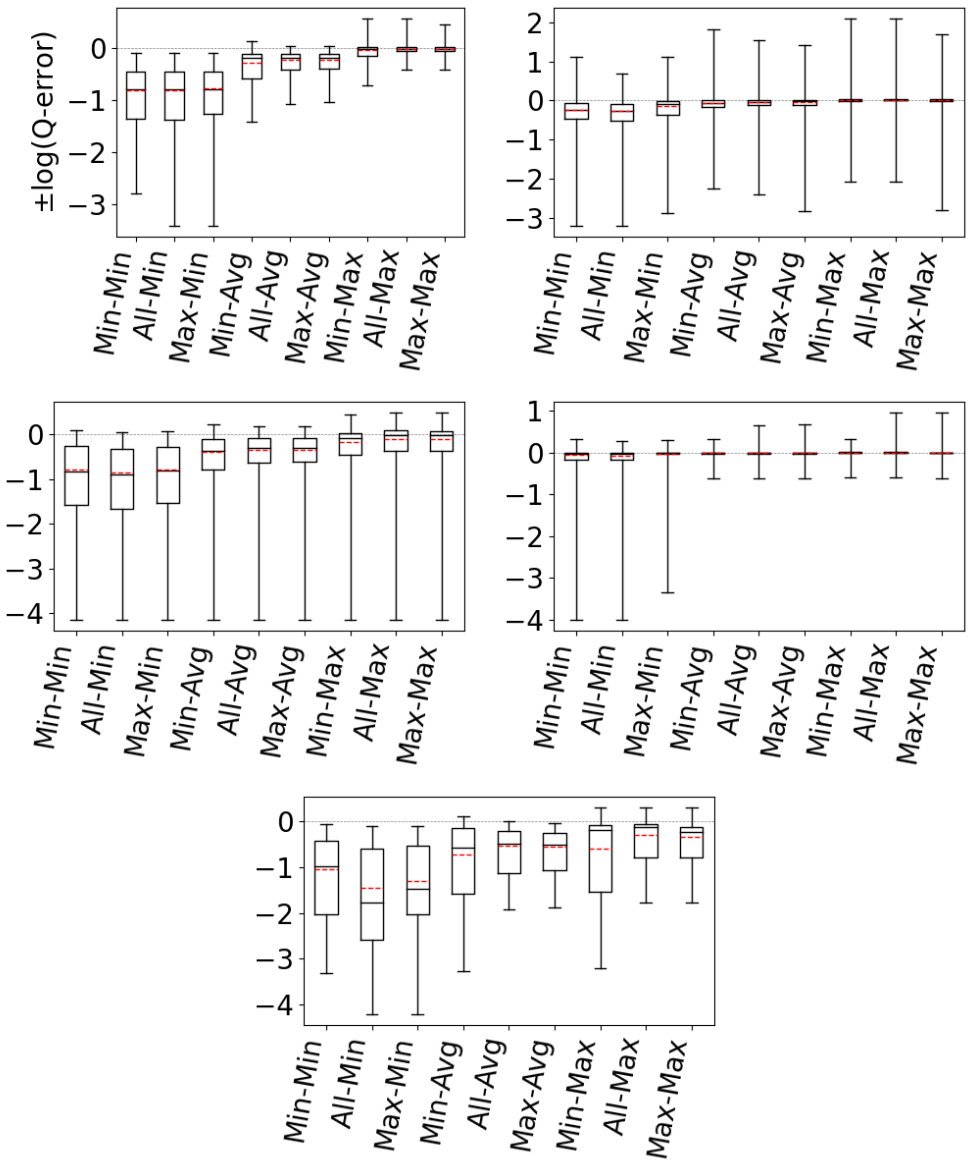


Figure 6.2: `Acyclic` workload: comparison between optimistic estimators. Note that the x-axis labels are shortened using the format of `(hop)-(aggr)`. For example, `min-hop-max-aggr` is shortened to `min-max`. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

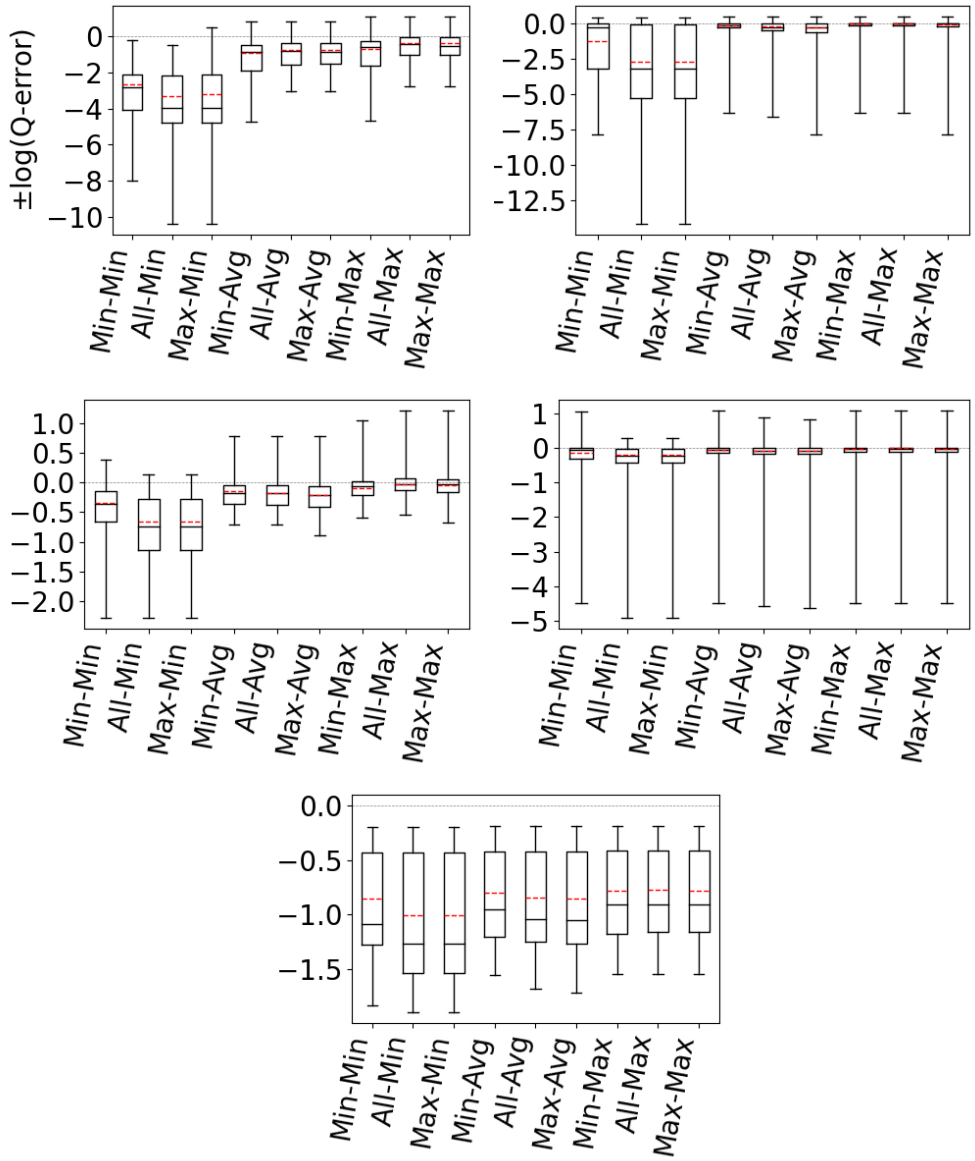


Figure 6.3: `Cyclic` workload: comparison between optimistic estimators. Note that the x-axis labels are shortened using the format of `(hop)-(aggr)`. For example, `min-hop-max-aggr` is shortened to `min-max`. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

we ignore the outliers and focus on the 25-75 percentile boxes, `max-hop` and `all-hops` do at least as good as `min-hop`. Further observe that on IMDb, Hetionet, and on the `Acyclic` workload on Epinions, `max-hop` and `all-hops` lead to significantly more accurate estimates. Finally, the performance of `max-hop` and `all-hops` are comparable across our experiments. We verified that this is because `all-hops` effectively picks one of the `max-hop` paths in majority of the queries in our workloads. Since `max-hop` considers fewer paths, it is more efficient to implement than `all-hop`. We conclude that systems implementing optimistic estimators we consider should use the `max-hop-max` estimator.

Finally, Table 6.2 reports the number of over- and under-estimations that `max-hop-max`, `all-hops-max`, `all-hops-avg`, and `all-hops-min` estimators make on the `Acyclic` and `Cyclic` workloads. `all-hops-max` and `all-hops-min` are, respectively, the most pessimistic and optimistic of the optimistic estimators. Observe that not only does using the `max` aggregator yield more accurate results, it also underestimates significantly fewer number of queries on our datasets. As we hypothesized in Section 4.3, using `max` aggregator can be effective in combatting underestimation. For example, on DBLP and `Acyclic` workload, `all-hops-max` overestimates on 75% of the queries while `all-hops-min` overestimates on only 8% (and is significantly less accurate overall).

6.3 Optimistic vs. Pessimistic Estimators

Our second set of experiments aim to answer: *How do the accuracies of `max-hop-max` optimistic and the MOLP pessimistic estimators compare*? To answer this question, we compared the q-errors of `max-hop-max` and MOLP on `JOB` workload on IMDb, the `Acyclic` workload on the rest of our datasets, and `Cyclic` on all of our datasets. Our results are shown in Figures 6.4 and 6.5 (ignore the CLLP boxes for now). First, these results confirm the results from reference [33] that the accuracy of the estimates of MOLP estimator is very loose. In addition to the estimators studied in reference [33], these estimates are also several magnitudes worse than the optimistic estimators we consider. For example, on the Hetionet dataset and `Acyclic` workload, the mean q-error of `max-hop-max` and MOLP, are 1.45 and 7987, respectively.

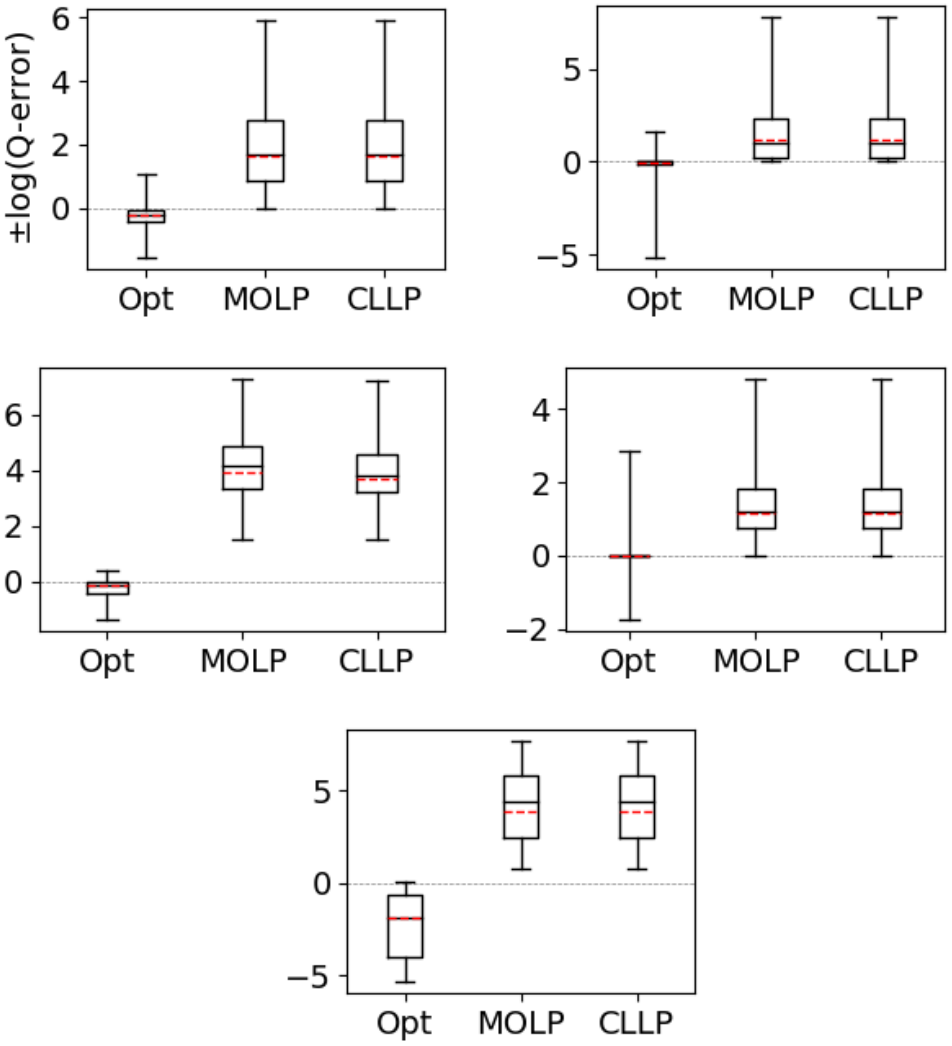


Figure 6.4: Q-error distribution of `max-hop-max` optimistic, MOLP, and CLLP on `JOB` on IMDb and `Acyclic` on other datasets. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

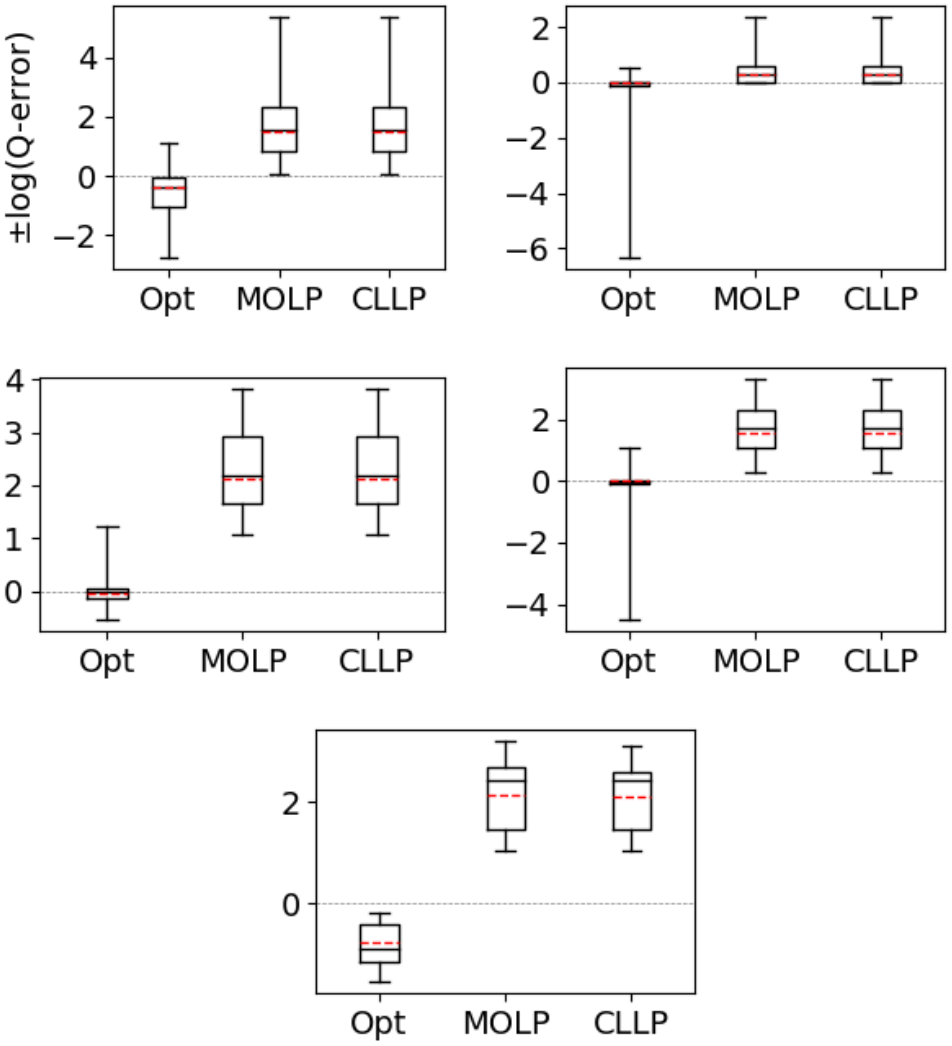


Figure 6.5: Q-error distribution of `max-hop-max` optimistic, MOLP, and CLLP on `Cyclic`. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

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

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

Table 6.2: Number of over- and under-estimations for `Acyclic` (or `JOB` for IMDb) and `Cyclic` workload. Note that the x-axis labels are shortened using the format of `(hop)-(aggr)`. For example, `all-hops-max-aggr` is shortened to `all-max`.

6.4 Refinements to Optimistic and Pessimistic Estimators

We next study the effects of the hash partitioning refinement to optimistic estimators. We also study the effect of adding the sub-modularity constraints to the MOLP estimator.

Dataset	% Improved	% Degraded
IMDb	92.94%	7.06%
DBLP	54.17%	45.55%
Hetionet	67.5%	32.5%
WatDiv	44.44%	55.28%
Epinions	92.5%	7.5%

Table 6.3: Percentage of `Acyclic` (or `JOB`) queries are improved/degraded with hash partitioning.

6.4.1 Effects of Hash Partitioning

Our next set of experiments aim to answer: *How much do the optimistic (and pessimistic) estimators benefits from the hash partitioning optimization?* To answer this question, we tested the effects of hash partitioning on the `JOB` workload on IMDb and `Acyclic` workload on our other datasets. Then we applied the hash partitioning optimization to both `max-hop-max` and MOLP estimators and measured the q-errors of the estimators under partitioning budgets of 1 (no partitioning), 4, 16, 64, and 128. Our results are shown in Figures 6.6 and 6.7. As demonstrated in reference [6], our results confirm that partitioning improves the accuracy of MOLP. The mean accuracy of MOLP increases between 15% and 89% across our datasets when moving between 1 and 128 partitions. The results for `max-hop-max` are data dependent. On DBLP and WatDiv, we cannot expect visible improvements because the estimates of `max-hop-max` on these datasets are very close to perfect and there is no room for significant improvement. On two of the remaining three datasets, Hetionet and Epinions, partitioning improves the mean accuracy significantly by 25% and 89%, respectively. On IMDb, we see little improvements.

Recall that unlike the MOLP estimator, this optimization is not guaranteed to improve the accuracy of the optimistic estimators. We show in Table 6.3 the percentage of queries whose accuracy improved and degraded. Note that on IMDb and Epinions almost all of the queries in our workload improved. This percentage of queries that improved varies between 44% and 67% on the remaining datasets.

6.4.2 Effects of Submodularity Constraints

Recall that reference [6]'s pessimistic estimator enumerates a subset of the CLLP bounds, which we showed were equal to MOLP on binary acyclic queries. CLLP is the tightest of the

known worst-case optimal bounds and extends MOLP with sub-modularity constraints. In our next set of experiments, we addressed the question of: *How much do the submodularity constraints improve the accuracy of the MOLP estimator?* Since with the sub-modularity constraints, we cannot use a CEG, we implemented the CLLP estimator with a numerical solver. We ran CLLP on the same workloads we had used to compare the `max-hop-max` and MOLP. Our results are shown in Figures 6.4 and 6.5. Overall, we see negligible improvement in accuracy except in Hetionet on `Acyclic`, where the improvement is 36%. This observation is important as it provides that systems using a strictly pessimistic estimator might prefer using a fast combintorial solver for MOLP (using CEG_M) instead of using a slow numerical solver for CLLP, which is a more complex linear program, as they should not expect significant improvements from the addition of sub-modularity constraints.

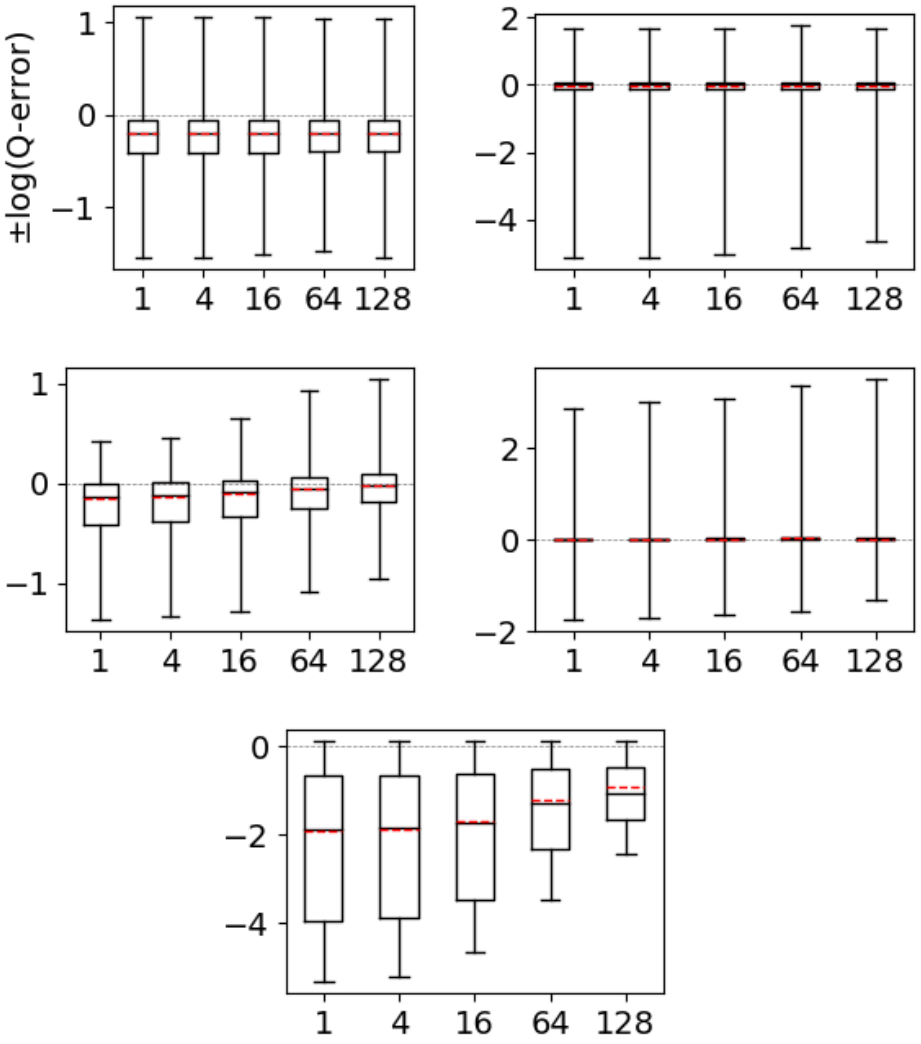


Figure 6.6: Effects of hash partitioning on `max-hop-max` estimator. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

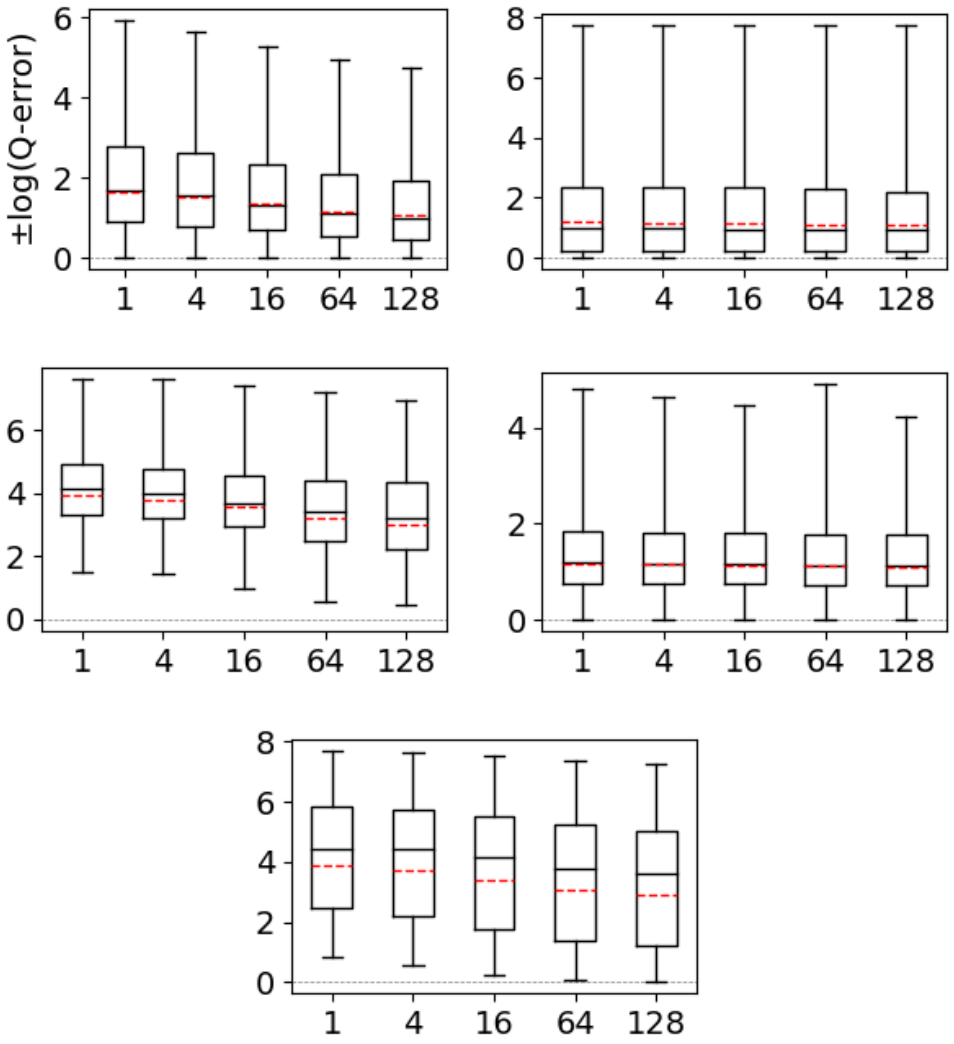


Figure 6.7: Effects of hash partitioning on the MOLP estimator. The red dashed line indicates the mean of the q-errors, excluding the highest 10% outliers. The charts, left-to-right and top-to-bottom, correspond to IMDB, DBLP, Hetionet, WatDiv, and Epinions.

Chapter 7

Related Work

There is decades of research on cardinality estimation of queries in the context of different database management systems. We cover a part of this literature focusing on work on graph-based database management systems, specifically XML and RDF and on relational systems. We also cover another technique, based on maximum entropy that can be used with any estimator that can return estimates for base tables and or small-size joins. We do not cover more work that uses machine learning techniques to estimate cardinalities and refer the reader to several recent work in this space [17, 23, 43] for details of these techniques.

Other Summary-based Estimators: The estimators we studied in this paper fall under the category of summary-based estimators. Many relational systems, including commercial ones such as PostgreSQL, use summary-based estimators. We do not provide an extensive review of this work and refer the reader to several references for details. Example summaries include the cardinalities of relations, which is also used by all of the estimators we evaluate the number of distinct values in columns, or histograms [3, 37, 28], wavelets [26], or probabilistic and statistical models [8, 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.

We studied three summary-based estimators that have been introduced in the context of managing graph-structured data. All of these three estimators use as statistics cardinalities of small-size queries. Several estimators from literature have proposed summary-based cardinality estimators for subgraph queries that compute a sketch of an input graph. We refer to these as sketch-based techniques though in the common use of the term, these can be considered summary-based techniques as well. In the context of estimating the

selectivities of path expressions, XSeed [47] and XSketch [34] build 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 contain 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 in the context of XML databases have used data structures that are adaptations of histograms from relational systems to store selectivities of path or tree queries in XML documents. Examples include, *positional histograms* [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 stores 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) data for purposes other than cardinality estimation. For example, DataGuides [9] was used in the Lore system to summarize object exchange model databases to be used by users to browse the database and formulate queries. Similarly, *TreeSketch* [35] produces a summary of large XML documents to provide approximate answers to queries.

In the context of RDF graphs, *SumRDF* is similar to XSeed and others and builds a summary graph S of an RDF graph. SumRDF adopts a holistic approach to making estimates. Given the summary S, SumRDF considers all possible RDF graphs G that could have the same summary S. Then they return the average cardinality of Q across all possible instances. This is effectively another form of uniformity assumption: each possible world has the same probability of representing the actual graph on which the estimate is being made. Similar to the sketch Note that the pessimistic estimators can also be seen as doing something similar, except they consider the most pessimistic of the possible worlds and return the cardinality of Q on that instance.

Characteristic sets [29] is another summary-based cardinality estimator that was used in the RDF-3X [30] system. Characteristic sets is primarily designed to estimate the cardinalities of stars in an RDF graph. As statistics it uses the *characteristic set* of each vertex v in an RDF graph, which is the set of distinct outgoing labels v has. Then statistics about all nodes with the same characteristic set are stored. These include the number of nodes that belong to a characteristic set and the total number of edges with a particular label. Then using these statistics the estimator makes estimates for the number of distinct matches of stars. For example, a star query with three labels A, B, C would be estimated by using the statistics from each characteristic set that contains these three labels and

summing the estimates from each. For example, the estimate from the characteristic set {A,B,C} is the number of distinct nodes with this set multiplied by the average A, B, and C edges of these nodes. Therefore this formula uses a uniformity assumption. For non-star queries, a query is decomposed into multiple stars $s_1, ..., s_k$, then the estimate for each s_i is multiplied, which corresponds to an independence assumption. Similar to the problem we focused on in this paper, for many queries, there can be multiple ways to decompose a query into multiple stars, and the question of which decomposition to pick can also be modeled as picking a path in a CEG. The CEG for characteristic sets would consists of unions of stars of a query as nodes as a set of edges. Then there would be an edges from a node W_1 to W_2 with a weight of the estimate for the star represented by $W_2 \setminus W_1$, using the estimation technique described above. In our preliminary experiments for non-star queries, characteristic sets did not perform as well as the optimistic estimators we covered in this work for query patterns other than stars. As a result, we omitted an evaluation of characteristic sets in our work.

Sampling-based Estimators: Another class of important estimators that have been used in relational systems are based on sampling tuples [12, 19, 22, 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. In the context of estimating subgraph queries, a recent work [33] has shown that some sampling based estimators designed for relational systems, can return highly accurate estimates compared to two summary-based ones (this work however has not considered the Markov Table-based estimators we studied in this paper). One advantage of sampling-based estimators is that they are often unbiased estimators, so by increasing the sizes of the samples, one can always make highly accurate estimate. In the context of graph data, sampling-based estimators have been used to estimate frequencies of subgraphs relative to each other to discover *motifs*, i.e. infrequently appearing subgrahs, [16] and lately also estimate the cardinalities of subgraph queries [7, 33].

Across relational systems, sampling based estimators have seen adoption in systems to estimate the selectivities of predicates in base tables but not on multiway joins. For example they HyPer system uses samples for predicates on base tables and uses independence assumptions. A study by the authors of the Hyper system observed that none of the 4 commercial database management systems, one of which is Postgres (the names of others are not provided) use sampling and instead resort to independence assumptions to calculate estimates for join queries. One shortcoming of sampling-based estimators is that they are often less efficient than summary-based estimators, even those that compute samples offline, as they effectively perform the actual join on the samples, and possibly each sub-query of

a given query as well, during query optimization. This is a slower process than making an estimate based on small size joins.

The Maximum Entropy Estimator: Markl et al. [25] has proposed another approach to making estimates for conjunctive predicates, say $p1 \wedge ... \wedge p_k$ given a set of ℓ selectivity estimates $s_{i_{11},...,i_{1k}}$, $s_{i_{\ell 1},...,i_{\ell k}}$, where $s_{i_{j1},...,i_{jk}}$ is the selectivity estimate for predicate $p_{i_{j1}} \wedge ... \wedge p_{i_{jk}}$. Markl et. al.'s maximum entropy approach take these known selectivities and using a constraint optimization problem, compute the distributions that maximizes the entropy of the joint distribution of the 2^k possible predicate space. Reference [25] 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.

Chapter 8

Conclusions and Future Work

We showed that we can represent both LP-based pessimistic estimators and the seemingly unrelated optimistic bounds that use multiplications of formulas as instances of CEGs with different edge weights. We can directly represent the estimations made by MOLP-based estimators and the optimistic bounds as instances of finding paths in their corresponding CEGs. We showed that we can also use CEGs to understand several properties of the DBPLP, e.g., why DBPLP is more pessimistic than MOLP and why these bounds are pessimistic. We believe the alternative combinatorial proofs of the LP-based pessimistic bounds we provided using CEGs are simpler and more direct than the original proofs provided in prior literature. We then showed that while it is clear which path MOLP uses to make an estimation (i.e., the shortest $(\emptyset, \mathcal{A})$ one, in its CEG, in the optimistic bounds it is often not clear which paths to use. We then empirically studied which paths in the CEGs of optimistic bounds lead to better estimates. Finally we compared the accuracies of both the pessimistic and optimistic bounds using exactly the same statistics and same preprocessing techniques that have been described for pessimistic estimators.

Our work opens up a new set of interesting questions, which instructs venues for future works:

- Other CEGs: Pessimistic estimators use maximum degrees as weights, while optimistic ones use estimates of average degrees. The weights one can put on CEGs, e.g., are not limited to these. For example, one can put in the entropies of different extensions as weights. Future work can introduce and study different classes of estimators that put different weights on the edges of CEGs.
- We did not focus on the speed with which estimations can be made. Future work can study the speed with which these estimations can be made. A possible approach to

speed up estimation is to sparsify the number of edges in CEGs, which correspond to removing some of the constraints in pessimistic estimators, or some pattern-based formulas in optimistic estimators.

- Finally, our work focused primarily on acyclic queries (with only a few cyclic query templates), which are known to be the predominant class of queries in practice. Future work can study the performance of pessimistic and optimistic estimators on larger classes of queries.

References

[1] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. Computing join queries with functional dependencies. In *PODS*, 2016.

[2] Ashraf Aboulnaga, Alaa R. Alameldeen, and Jeffrey F. Naughton. Estimating the selectivity of xml path expressions for internet scale applications. In *VLDB*, 2001.

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

[4] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. Diversified stress testing of rdf data management systems. In *The Semantic Web – ISWC 2014*, 2014.

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

[6] Walter Cai, Magdalena Balazinska, and Dan Suciu. Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. In *SIGMOD*, 2019.

[7] Xiaowei Chen and John C. S. Lui. Mining Graphlet Counts in Online Social Networks. *ACM TKDD*, April 2018.

[8] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. In *SIGMOD*, 2001.

[9] Roy Goldman and Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *VLDB*, 1997.

[10] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. Size and treewidth bounds for conjunctive queries. *JACM*, 59(3), 2012.

[11] Marc Graham. On the universal relation. Technical report, University of Toronto, September 1979.

[12] Haas, Peter J. and Naughton, Jeffrey F. and Seshadri, S. and Swami, Arun N. Selectivity and Cost Estimation for Joins Based on Random Sampling. *Journal of Computer and System Sciences*, 52(3), 1996.

[13] Hetionet. https://het.io/, 2020.

[14] Yannis Ioannidis. The history of histograms (abridged). In *VLDB*, 2003.

[15] Manas R. Joglekar and Christopher M. Ré. It's all a matter of degree: Using degree information to optimize multiway joins. In *ICDT*, 2016.

[16] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics (Oxford, England)*, 20, 08 2004.

[17] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *CIDR*, 2019.

[18] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proc. VLDB Endow.*, 9(3):204–215, November 2015.

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

[20] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. Query optimization through the looking glass, and what we found running the join order benchmark. In *The VLDB Journal*, 2018.

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

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

[23] Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, and Calisto Zuzarte. Cardinality Estimation Using Neural Networks. In *CASCON*, 2015.

[24] Angela Maduko, Kemafor Anyanwu, Amit Sheth, and Paul Schliekelman. Graph Summaries for Subgraph Frequency Estimation. In *The Semantic Web: Research and Applications*, 2008.

[25] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tam. Consistent selectivity estimation via maximum entropy. *VLDBJ*, 16, 2007.

[26] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD*, 1998.

[27] Amine Mhedhbi and Semih Salihoglu. Optimizing subgraph queries by combining binary and worst-case optimal joins. *PVLDB*, 12(11), 2019.

[28] M. Muralikrishna and David J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *SIGMOD*, 1988.

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

[30] Thomas Neumann and Gerhard Weikum. Rdf-3x: A risc-style engine for rdf. In *VLDB*, 2008.

[31] Dung Nguyen, Molham Aref, Martin Bravenboer, George Kollias, Hung Q. Ngo, Christopher Ré, and Atri Rudra. Join Processing for Graph Patterns: An Old Dog with New Tricks. In *GRADES*, 2015.

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

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

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

[35] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. Approximate xml query answers. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 263–274, New York, NY, USA, 2004. Association for Computing Machinery.

[36] Neoklis Polyzotis, Minos Garofalakis, and Yannis Ioannidis. Approximate xml query answers. In *SIGMOD*, 2004.

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

[38] PostgreSQL. https://www.postgresql.org/, 2020.

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

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

[41] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P. Chakkappen. Join Size Estimation Subject to Filter Conditions. 2015.

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

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

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

[45] Yuqing Wu, Jignesh M. Patel, and H. V. Jagadish. Estimating answer sizes for xml queries. In *EDBT*, 2002.

[46] Clement Yu and M. Z. Ozsoyoglu. An algorithm for tree-query membership of a distributed query. In *COMPSAC*, 1979.

[47] Ning Zhang, M. Tamer Ozsu, Ashraf Aboulnaga, and Ihab F. Ilyas. Xseed: Accurate and fast cardinality estimation for xpath queries. In *ICDE*, 2006.

APPENDICES

Appendix A

WBS Estimator's Connection to MOLP On Acyclic Queries

We first show that on acyclic queries MOLP is at least as tight as the WBS 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 WBS 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 can be seen exactly as $\sum_i \log deg(Y_i, R_i)$ (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. 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, $\mathcal{R}_E = \{R_i(\mathcal{A}_i) | R_i \text{ covers } |\mathcal{A}_i|-1 \text{ attributes}\}$. We ignore the relations that do not cover any attributes.

Let relation in $\mathcal{R}_C = R_{C1}(\mathcal{A}_{C1}), ..., R_{Ck}(\mathcal{A}_{Ck})$ and those in $\mathcal{R}_E = R_{E1}(\mathcal{A}_{E1}), ..., R_{Ek'}(\mathcal{A}_{Ek'})$. Then we can construct the following path. The first of the path uses the cardinalities or relations in $\mathcal{R}_C$, in an arbitrary order, to extend $(\emptyset)$ to $U = (\cup_{i=1,...k} \mathcal{A}_{Ci})$. For example this path can be: $P1 = (\emptyset) \xrightarrow{\log(|R_{C1}|} (\mathcal{A}_{C1}) \xrightarrow{\log(|R_{C2}|} (\mathcal{A}_{C1} \cup \mathcal{A}_{C2}) ... \xrightarrow{\log(|R_{Ck}|} (U)$. Now to extend U to $\mathcal{A}$, observe that for each uncovered attribute $T = \mathcal{A} \setminus U$, there must be some relation $R_{Ej}(\mathcal{A}_{Ej}) \in \mathcal{R}_E$, such that all of the $|\mathcal{A}_{Ej}|-1$ attributes are already bound in U. This is because $|T| = k'$ and if each R_{Ej} has at least 2 attributes in T, then Q must be cyclic. Note that this is true because by definition of acyclicity [11][46] 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 $P1$ 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 WBS 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 WBS estimator because the first edge with weight $\log deg(\{B\},\{B\},R)$ uses the cardinality of projection of R. However, the WBS 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 WBS 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 sub-query 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 sub-query to one new attribute a using the degree of a in R_i; (iii) $Proj$ are those edges that extend a sub-query by a single attribute a, without bounding any attributes in the sub-query, 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 sub-query 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 $Proj$ 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, otherwise 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 sub-query 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 and we can again remove them add one edge to $Card$ with weight $\log(\emptyset, \{a_1, a_2\}, R_p)$ 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).

Appendix B

Counter Example for Using the WBS Estimator on Cyclic Queries

Consider the triangle query $R(a,b) \bowtie S(b,c) \bowtie T(c,a)$. FCG would generate the cover $a \to R$, $b \to S$, and $c \to T$. 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.

Appendix C

DBPLP

We end this section by demonstrating another application of CEGs and provide alternative combinatorial proofs to some properties of DBPLP, which is another worst-case output size bound from reference [15]. DBPLP is not as tight as MOLP (or CLLP) (which our proofs demonstrate through a path-analysis of CEGs). Therefore, we will not use DBPLP in our evaluations. Readers interested in our evaluations can skip over this section. 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:

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

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

DBPLP CEG (CEG_D):

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

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 C.0.1. *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 $d_{\mathcal{A}}$, has to have a value greater than $|P|$. □

Corollary C.0.1. *Let $m_{\mathcal{A}}$ and $d_{\mathcal{A}}$ be the solutions to MOLP and DBPLP, respectively. Then $m_{\mathcal{A}} \leq d_{\mathcal{A}}$ for any cover C used in DBPLP.*

Proof. Directly follows from Theorems 5.1.1 and C.0.1 and the observation that CEG_D contains the same vertices and a subset of the edges in CEG_M. □

Corollary C.0.1 is a variant of Theorem 2 from reference [15], which compares refinements of MOLP and DBPLP. Our proof serves as an alternative combinatorial proof to the inductive proof in reference [15] 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.