Greedy Algorithm for Marginalization Ordering of Hypertree Decompositions over Relational Tables

Zezhou Huang
zh2408@columbia.edu
Columbia University

Eugene Wu
ewu@cs.columbia.edu
DSI, Columbia University

ABSTRACT

...

ACM Reference Format:
Zezhou Huang and Eugene Wu. 2021. Greedy Algorithm for Marginalization Ordering of Hypertree Decompositions over Relational Tables. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The theory of Hypertree Decompositions [7] lays the foundations for many important problems including Semiring Aggregation [8], Probabilistic Grpahical Model [14], Marginalize a Product Function[3], Factorised Database[16], Finite model Theory[4], etc. In the context of Semiring Aggregation, the insight behind Hypertree Decompositions is that many aggregation functions can be decomposed into join and marginalization operators, where the marginalization operators can be pushed down and the query time can be decreased by orders of magnitude. Consider the following toy example:

Example 1. *Consider the example tables T1(A,B) and T2(B,C) in Figure 1a and the query* $\gamma_{\text{COUNT}}(\text{T1}(\text{A},\text{B}) \bowtie \text{T2}(\text{B},\text{C}))$*, where each attribute's domain is* $\text{O}(\text{n})$*. The sizes of both relations are thus* $\text{O}(\text{n}^2)$ *and the size of join result (Figure 1b) is* $\text{O}(\text{n}^3)$*. Notice that, attribute A and C are not used for the join, and we can "marginalize" them early before join. To marginalize attribute A, we scan each table T1 and T2, and for each unique attribute value of B, we sum its count. The marginalization results are shown in Figure 1c, whose size is* $\text{O}(\text{n})$*. Then, joining the marginalized tables can be achieved in* $\text{O}(\text{n})$*. We have successfully reduced the query time of aggregation from* $\text{O}(\text{n}^3)$ *to* $\text{O}(\text{n})$ *by decomposing aggregation function and pushing down marginalization. The aggregation through marginalization can be written in algebra form* $\sum_{\text{A,B,C}} \text{T1}(\text{A},\text{B}) \bowtie \text{T2}(\text{B},\text{C})$*. The marginalization of attributes can be pushed down if attributes are not used in outer query:* $\sum_{\text{B}}((\sum_{\text{A}} \text{T1}(\text{A},\text{B})) \bowtie (\sum_{\text{C}} \text{T2}(\text{B},\text{C})))$.

The introduction of marginalization operator has exhibited new optimization opportunities for Semiring Aggregation Queries. (Fundamental question: in which order shall we marginalize atttribute)

T1

A	B	Count
a1	b1	3
a2	b1	4

T2

B	C	Count
b1	c1	3
b1	c2	4

(a) Tables to join.

T1 ⋈ T2

A	B	C	Count
a1	b1	c1	9
a1	b1	c2	12
a2	b1	c1	12
a2	b1	c2	16

(b) Join result.

$\sum_A$ T1

B	Count
b1	7

$\sum_C$ T2

B	Count
b1	7

(c) Marginalized results.

Figure 1: Example aggregation query. (a) Tables to compute total count. (b) Join result. (c) Marginalized result.

One naive optimization is to consider marginalization as generalized projection [10] and greedily push down the generalized projection over the query plan found by traditional optimizer [17]. However, this approach is suboptimal as the marginalization result is likely to be much smaller than the original relation, which will lead to better query plan.

Fractional Hypertree Width [9] has been widely used to quantify the complexity of Hypertree Decompositions for Semiring Aggregation Queries. For example, the query complexity of Semiring Aggregation Queries like Marginalize a Product Function, and Quantified Conjunctive Query is well known to be the Fractional Hypertree Width of the query hypergraph [2, 4, 12]. Similar theoretic result has shown that the size bound of factorised representation is the Fractional Hypertree Width of the query hypergraph [16]. These studies have inspired database systems like EmptyHeaded [1] to search for query plan that minimize the Fractional Hypertree Width of the query hypergraph.

However, previous works that optimize Hypertree Decompositions based on Fractional Hypertree Width are limited for two reasons. Firstly, it is well known that finding the Fractional Hypertree Width with minimum Fractional Hypertree Width is NP-hard [5]. EmptyHeaded assumes that the number of relations and attributes is small enough to exhaustively search all marginalization orders, which is not scalable. Secondly, Fractional Hypertree Width provides an upper bound of the intermediate result size, which is likely to overestimate the result size. Recent work [15] has shown that skewness can lead the real result size asymptotically smaller than the bound provided by Fractional Hypertree Width. Optimizing Hypertree Decompositions solely by Fractional Hypertree Width may lead to suboptimal query plan in practice.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

This paper presents a greedy algorithm to efficiently find marginalization ordering of Hypertree Decompositions over relational tables based on cardinality estimation. Greedy algorithm has been widely used in Probabilistic Grpahical Model [6] to search for small Tree Decompositions over large networks and have achieved great performance over real-world datasets [13]. However, unlike probabilistic tables, relational tables can be highly skewed and sparse, and naively applying the heuristics from Probabilistic Grpahical Model will lead to poor performance. To account for the characterises of relational tables, we propose simple heuristics based on cardinality estimation [11] of join and marginalization result. We demonstrate the effectiveness of the algorithm with two important applications over relational tables: finding efficient query plan for Semiring Aggregation[1] and finding efficient Factorised Representation for join query.

REFERENCES

[1] C. R. Aberger, A. Lamb, S. Tu, A. Nötzli, K. Olukotun, and C. Ré. Emptyheaded: A relational engine for graph processing. *ACM Transactions on Database Systems (TODS)*, 42(4):1–44, 2017.

[2] M. Abo Khamis, H. Q. Ngo, and A. Rudra. Faq: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 13–28, 2016.

[3] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE transactions on Information Theory*, 46(2):325–343, 2000.

[4] H. Chen and V. Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 205–214. IEEE, 2012.

[5] W. Fischl, G. Gottlob, and R. Pichler. General and fractional hypertree decompositions: Hard and easy cases. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 17–32, 2018.

[6] D. Geiger and M. Fishelson. Optimizing exact genetic linkage computations. In *7th Annual International Conf. on Computational Molecular Biology*, pages 114–121, 2003.

[7] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

[8] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

[9] M. Grohe and D. Marx. Constraint solving via fractional edge covers. *ACM Transactions on Algorithms (TALG)*, 11(1):1–20, 2014.

[10] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. 1995.

[11] H. Harmouch and F. Naumann. Cardinality estimation: An experimental survey. *Proceedings of the VLDB Endowment*, 11(4):499–512, 2017.

[12] M. Joglekar, R. Puttagunta, and C. Ré. Aggregations over generalized hypertree decompositions. *arXiv preprint arXiv:1508.07532*, 2015.

[13] K. Kask, A. Gelfand, L. Otten, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

[14] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[15] H. Q. Ngo, C. Ré, and A. Rudra. Skew strikes back: New developments in the theory of join algorithms. *ACM SIGMOD Record*, 42(4):5–16, 2014.

[16] D. Olteanu and J. Závodnỳ. Size bounds for factorised representations of query results. *ACM Transactions on Database Systems (TODS)*, 40(1):1–44, 2015.

[17] M. Schleich, D. Olteanu, M. Abo Khamis, H. Q. Ngo, and X. Nguyen. A layered aggregate engine for analytics workloads. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1642–1659, 2019.

[1]Semiring Aggregation has been shown to be general enough [2, 12, 17] to express problems including Inference in Probabilistic Grpahical Model, Marginalize a Product Function, Quantified Conjunctive Query, Machine Learning, etc.