EQUALGAS: Efficient Query Approximation on Large Graphs using Attribute-based Summaries

Narender Gupta, Chinmay Kulkarni, Chaitanya Datye
University of Illinois at Urbana-Champaign
{ngupta18, ckulkarn, cdatye2}@illinois.edu

1 Introduction

The goal of this project is to provide a scalable framework to efficiently answer complex queries over large-scale graphs. The framework provides an approximate result for the queries by combining information from underlying attribute-based summary graphs constructed from the original network. The results are obtained using a Naïve Bayes approximation model.

Recent rapid growth in real-world social networks has incentivized researchers to explore optimizations which can provide quick insights about the network. Due to this motivation, graph summarization and approximation has been an important research problem. Most of the work in this area has been focused on concise and informative representations of large graph. It is, however, difficult to come up with a single universal representation optimized for all purposes. Our work primarily focuses on task-based summarization of large graphs and answer queries which are computationally expensive on original graph, but have tolerance with regards to minor errors in exact results. These queries, semantically, provide the same amount of information even with approximate results. One such example can be that of an agency which wants to estimate the number of people impacted by a certain advertising budget on a social network. The agency is less likely to change its decision if the result is 4,515,736 or 4,481,973 but is mostly interested if the result is close to 4.5 Million or 8.5 Million.

Calculating the exact result of such queries can possibly take one or multiple passes of the complete original graph, which scales even up to Peta or Exa Bytes. Our contribution is a framework which can answer queries probabilistically in a highly efficient way using compact representations of original graph stored in form of summary graphs. These summary graphs are also optimized for space complexity, and only grow in terms of the number of attributes used to answer the query. One can then use a combination of these graphs to answer complex queries in an extremely efficient manner. While calculating the approximate results, the attributes are assumed to follow Naïve Bayes conditional independence. This assumption allows us to store pairs of attributes, and enables the computation of the joint probability of any combination of attributes. Thus, we can represent the entire summary and still avoid the combinatorial explosion of attribute combinations. We analyze our framework in terms of efficiency and accuracy over varying graph size and query complexity. Our results are promising and show that significant gains in runtime can be achieved using our framework without sacrificing too much on accuracy. In fact, we observe decreasing trend in error as the graph size increases.

The rest of the paper is organized as follows. Section 2 gives a brief overview of existing research. We propose our framework in Section 3, and explain the dataset in detail in Section 4. Section 5 gives an illustrative example of a simple query run on a sample of our dataset. We provide the complexity analysis in terms of time and space in Section 6. Section 7 describes evaluation metrics and experimental setup. In Section 8, we talk about the results of experiments followed by conclusion and possible future directions in Section 9.

2 Related Work

2.1 Graph Summarization

Graph summarization and network approximation using representative graphs has been an important research problem due to the ever-increasing size of real-world social networks. Since real-world graphs are massive in size and continuously evolving, a summary graph is often constructed using graph sampling techniques [1] in order to represent the entire network and study properties of the entire network. While most studies focus on graph summarization in terms of reducing the description length of graphs and graph compression [2], [3], recent developments in social network analysis give prominence to constructing the representative graph in a way that will allow studying properties of the entire network using just the representative graph. These developments offer new challenges to network summarization and extend the idea of summarization beyond just compression or minimum description length.

One such representative graph technique is constructing the supernodes using an interestingness-driven technique by Leskovec and Faloutsos given in [4]. Their work primarily focuses on computing the summary graph based on the evolving diffusion process over time. The idea is to summarize the entire network of millions of nodes by only capturing few interesting nodes or edges based on an interestingness measure defined for each node. The more interesting a node is, the higher is the probability of that node being a supernode. The work studies dynamic network properties such as diffusion rate over time on this representative graph. One of the most important strengths of this work is that it is an online algorithm which will consider changes in the graph over time. However, a very important challenge in this approach is choosing the interval of time after which the network snapshot needs to

be studied. If this interval is too small, the performance of the algorithm will deteriorate. If the interval is too large, some important diffusion properties might be missed. Also, the work fails to address whether the detected summary is actually a good summary and will capture the properties of the underlying graph and if it does, to what extent the properties are captured. Even though this work captures diffusion rate and change in information exchange, it does not describe any static measures such as degree distribution, path length distribution, etc. that can be studied on the representative graph and may represent the same behavior on the underlying network. Our approach involves storing attribute-specific summary graphs which preserve such underlying properties of the main graph.

2.2 Graph Sampling

Ahmed, et. al. proposed different graph sampling techniques [5] which actually capture the properties of the underlying network and preserve the same in the representative graphs. The paper lists different measures that can be evaluated on the sampled graph. Evaluation metrics include degree distribution, path length distribution, clustering coefficient, etc. One of the major strengths of this paper is that these algorithms are defined taking into consideration both static as well as streaming data. However, the paper constructs the sample based on edge connectivity and topology of nodes, but doesn't take into account attribute-based similarities. Answering multi-attribute queries using such sampling techniques will not be possible without sufficient preprocessing of the summary graph which stores attributes for nodes of the summary graphs. Our approach is to store different summary graphs based on the query type and the attributes that this query is dependent on, which cannot be done using generic graph sampling.

2.3 Query Processing Techniques for Large Graphs

Previous research on query processing for large graphs generally involves introducing different methods for indexing these large graphs so that queries can be handled in a manner similar to how they are handled in Relational Database Management System. [6] focus on developing indices over super-graphs and sub-graphs so as to efficiently process queries. The main strength of this paper is that the cost of complex join operations is avoided by storing intermediary metadata about the graph using the indices built on top of the graph. Using this indexing mechanism, they translate graph queries into SQL scripts and thus leverage the computational efficiency of relational operations in SQL in order to achieve fast query processing. The main drawback of this paper is that metadata about specific queries must be recomputed irrespective of the repetitive nature of successive queries. [7] deals with developing an indexing technique called eIndex over the supergraph. The entire graph is considered as a database in their graph query processing algorithm and candidate subgraphs are generated based on the query. The index creation step makes use of all the nodes in the original graph as well as the entrire feature set. The main strength of this paper is the proposal of an efficient solution for the super-graph query problem, such that approximate algorithms are used to reduce the number of sub-graph tests. The authors make use of this result for querying super-graphs efficiently. A major weakness in this paper is that it includes a post-processing step that takes time that is polynomial in the size of the graph. Though this step is carried out only once per run of the algorithm, it will not be practical for large graphs of the order of hundreds of millions of nodes. [8] proposes a high-performance graph indexing mechanism to solve the problem of efficient querying in large graphs. The major strength of this approach as compared to other indexing mechanisms is that a shortest-path approach with respect to vertex neighborhood is taken rather than a per-vertex approach, which makes the indexing highly scalable and much more efficient computationally.

A significant weakness of all index-based approaches to graph querying is the requirement of a large amount of storage. In all cases, to store indexes for all the nodes of a super-graph, without narrowing down the set of nodes based on attributes, will require storage proportional to the size of the entire graph (the number of nodes can be in the order of millions). We thus propose a solution that generates attribute-based sub-graphs requiring storage that is considerably smaller than the size of the entire graph. These sub-graphs can furthermore be reused for queries that require the same combinations of attributes for their resolution.

2.4 Bayesian Belief Networks and Markov Networks

Bayesian Belief Networks make use of conditional probabilities across attributes to infer joint probability distributions. The exact inference task on these networks is NP-hard. In [10], Paul Dagum et. al. proved that there is no tractable deterministic algorithm that can approximate probabilistic inference to within an absolute error $\epsilon < \frac{1}{2}$. Second, they proved that there is no tractable randomized algorithm that can approximate probabilistic inference to within an absolute error $\epsilon < \frac{1}{2}$ with confidence probability greater than $\frac{1}{2}$. Even in special cases of approximate inference, computation time has been shown to be polynomial in input size. Thus, Bayesian Belief Networks are computationally expensive when it comes to query resolution. Unlike Bayesian Belief Networks which are directed graphs, Markov networks represent an undirected graph with conditional probabilities as edge weights. However, even in this case, above computational limitations hold true.

We have shown that our approach, which uses summary graphs, is computationally efficient and the runtime is independent of the underlying data size, but only depends on number of attributes and their values.

3 Proposed Methodology

Our model involves a preprocessing step which computes and stores the summary graphs followed by a query resolution step based on the Naïve Bayes approximation model.

3.1 Preprocessing

The graph has N nodes. Each node has k attributes, and each attribute can take up to v values.

The preprocessing step involves constructing attribute-based summary graphs of the underlying network. Since we want to

compute pairwise conditional probabilities of attributes, we construct summary graphs for every pair of attributes. Since there are *k* attributes, we will have to construct $\binom{k}{2}$ summary graphs. For constructing each of these graphs, we will have to traverse the underlying *N* nodes. Once these summary graphs are constructed and stored, each query just uses these in the query resolution step. An advantage of using pairwise combinations is that we can compute any larger combination of conditional probabilities using these graphs in an efficient way.

3.2 Query Resolution using Naïve Bayes Approximation

In this section, we formally define the query resolution step of our model which uses Naïve Bayes approximation for estimating query results over the large social network graph. Currently, we focus on queries that investigate relationship among a given set of attributes. We can model a general query to have the form - *Find the probability of co-existence of attributes* $a_1, a_2, \cdots, a_q$ *given that the attributes* $b_1, b_2, \cdots, b_r$ *are present*. Thus, we want to estimate the probability

$$\Pr(a_1, a_2, \cdots, a_q | b_1, b_2, \cdots, b_r)$$

We use Naïve Bayes independence assumption for the attributes. Thus, the above expression of probability breaks down to the following

$$\begin{aligned}\Pr(a_1, a_2, \cdots, a_q | b_1, b_2, \cdots, b_r) \\ = \Pr(a_1 | b_1, b_2, \cdots, b_r) \times \\ \Pr(a_2 | b_1, b_2, \cdots, b_r) \times \\ \cdots \times \Pr(a_q | b_1, b_2, \cdots, b_r)\end{aligned}$$

$$\begin{aligned}= \frac{\Pr(b_1, b_2, \cdots, b_r | a_1)\Pr(a_1)}{\Pr(b_1, b_2, \cdots, b_r)} \times \\ \frac{\Pr(b_1, b_2, \cdots, b_r | a_2)\Pr(a_2)}{\Pr(b_1, b_2, \cdots, b_r)} \times \\ \cdots \times \frac{\Pr(b_1, b_2, \cdots, b_r | a_q)\Pr(a_q)}{\Pr(b_1, b_2, \cdots, b_r)}\end{aligned}$$

Using Naïve Bayes Independence assumption,

$$\begin{aligned}= \frac{\prod_{i=1}^{r} \Pr(b_i | a_1)}{\Pr(b_1, b_2, \cdots, b_r)} \Pr(a_1) \times \\ \frac{\prod_{i=1}^{r} \Pr(b_i | a_2)}{\Pr(b_1, b_2, \cdots, b_r)} \Pr(a_2) \times \\ \cdots \times \frac{\prod_{i=1}^{r} \Pr(b_i | a_q)}{\Pr(b_1, b_2, \cdots, b_r)} \Pr(a_q)\end{aligned}$$

Also, the evidence attributes $b_1, b_2, \cdots, b_r$ are assumed to be independent of each other and so, we have

$$\Pr(b_1, b_2, \cdots, b_r) = \prod_{i=1}^{r} \Pr(b_i)$$

The final expression thus breaks down to

$$\boxed{\frac{\prod_{j=1}^{q} \prod_{i=1}^{r} \Pr(b_i | a_j) \Pr(a_j)}{\prod_{i=1}^{r} \Pr(b_i)^q}}$$

4 DATA

We are using the network of candidates applying to American graduate schools. Each student has multiple attributes such as standardized test scores, undergraduate university, demographic properties, and so on. Also, each student applies to multiple graduate universities, gets accepted by some and gets rejected by others. These interactions can be modeled as a social network with students and universities as entities. This dataset has entities i.e. students with several attributes, and is of considerable size to evaluate difference in performance on original graph versus summarization technique. We chose this dataset because it comes from a public forum, the complete dataset is thus available and there were no restrictions in regards to any privacy concerns or crawling limits. In case of networks like Facebook, Twitter, etc, since there are restrictions on crawling entire data, we only get a sample of data in every snapshot which might not be a representative sample of a complete graph.

There are several online resources where applicants share their admission experiences; Edulix is one commonly used resource [9]. It is an active resource which hosts applicant profiles from all over the world. GRE scores, undergraduate university name, GPA, TOEFL scores and other accomplishments such as work experience and research publications pertinent to the graduate admissions are reported in the profile. In addition, users mention the universities that they applied to and the result of each application (*Admit*, *Reject* or *Result Not Available*).

The data used here was collected from this website. Since the data is self reported, it had some erroneous reports, which were identified and removed by completely deleting any such record. Any application that was not classified as either *Admit* or *Reject* was also excluded. It is observed that GRE and TOEFL scores have undergone various changes in grading scale over the years. Also, undergraduate institutions all over the world follow different scales for reporting GPA. As a standardization measure, these fields are mapped linearly to a scale of 0-100. An undergraduate university might be referred to by the differing names due to reasons such as usage of a popular acronym or spelling errors. This problem was mitigated by mapping the university names to their unique website URL. The dataset has students from various undergraduate departments and applying to multitude of graduate departments. Some of the statistics of a sample of this dataset, candidate applying to Computer Science graduate department, are in Table I. Although Computer Science candidates make the plurality of our dataset, it is rich enough to accommodate students applying to various other engineering departments, business, humanities as well as biological sciences.

TABLE I: Features of Data

General Features	
Total number of users before sanitization	36,207
Total number of users after sanitization	26,148
Features for CS related dataset	
Number of users	10,788
Application year range	[2001 2015]
Median Application Year	2013
Most Frequent Application Term	Fall
Number of universities with reported data	313
Number of applications per student (Mean)	6
Number of applications per university (Mean)	51
Number of undergraduate universities	2353
Degrees sought	[MS, PhD]

5 Illustrative Example

This section details out on an illustrative example to show the working of our framework. The example chosen consists of data sampled randomly from our dataset, which consists of 75 candidates. We ran simple queries on our model to observe the quality of results. We will go through the intuition and explanations of our framework by providing a walk-through of this query example. Based on the sample dataset, we have considered a query in which we only consider four attributes, namely Undergraduate School, Program. Decision and Graduate School. For the Undergraduate School, we have *A* and *B* as the possible values. For the Program, we have *MS* and *PhD* as the possible values, for the Decision, we have *Ad* and *Rj* as possible values to denote an admit or a reject, and for the Graduate School, we have *S*, *C* and *U* as the possible values.

The Fig 1, 2, 3, 4, 5 and 6 show the summary graphs constructed from the preprocessing done on the sample dataset. An edge in a graph from node *i* to node *j* corresponds to the number of applications having *i* and *j* as their corresponding attribute values.

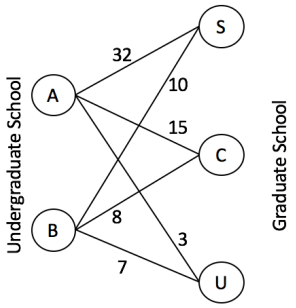

Fig. 1: Undergraduate School to Graduate School mapping

The sample query which we chose to evaluate our model on, asks - *If a person is from an undergraduate school A, what are her chances of getting admitted to the MS program at school S?* This query is important for students on the forum to get an idea of whether they can expect an admit from a university for a given program. Though the admission decision and graduate school are actually dependent attributes, we observe

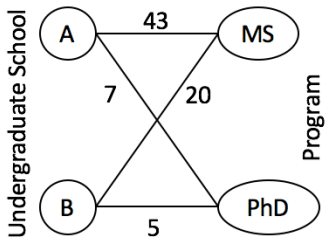

Fig. 2: Undergraduate School to Program mapping

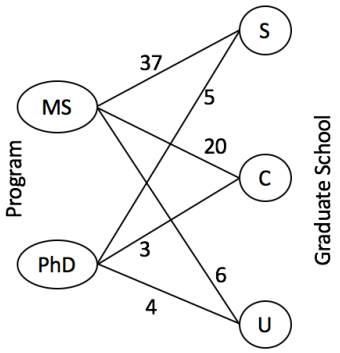

Fig. 3: Program to Graduate School mapping

that assuming conditional independence by applying Naïve Bayes rule gets us approximate results which are very close to the actual probability. This slight compromise in accuracy is worth it since our framework provides query resolution in much less time than the actual approach which runs in time that is linear in the data size.

Ideally, if we were to look at the entire graph, we would have looked at all nodes that have *Undergraduate School = A, Program = MS, Decision = Ad, Graduate School = S* as their attribute value tuple. However, if we just consider the above summary graphs, from Fig 1, 2, 3, 4, 5 and 6 we can estimate the probability of the person coming from Undergraduate School *A* and getting admitted to the *MS* program at Graduate School *S*.

From our model, we get an approximate result of 0.3853 for the query. A walk-through of this estimated result for the query is given below.

$$\begin{aligned}\Pr(S, Ad, MS|A) &= \Pr(S|A) \times \Pr(Ad|A) \times \Pr(MS|A) \\ &= \frac{32}{50} \times \frac{35}{50} \times \frac{43}{50} \\ &= 0.3853\end{aligned}$$

From this we can say that if a person were to finish her undergraduate education at school *A*, there is a 38.53% chance that she will get admitted to the *MS* program at graduate school *S*. The graphs in corresponding figures are generated for all 75 candidates that are contained in our sample dataset. In general, for a subgraph between attributes a_1 and a_2, we can see the number of candidates out of the total 75 that map to different values of a_1 and a_2.

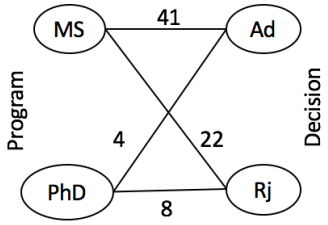

Fig. 4: Program to Decision mapping

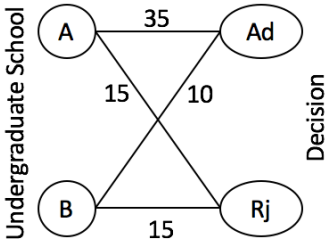

Fig. 5: Undergraduate School to Decision mapping

In order to prove the correctness of our estimated value, we find extreme case bounds for the estimated result.

One extreme case will occur when the maximum number of applicants that get admitted to the *MS* program to graduate school *S* are from the undergraduate school *A*. This means that out of the 32 people that applied to *S* from *A*, 30 got admits (since we can see that 30 people got admits to university *S* overall). We make sure that this is consistent with the number of people that apply for the *MS* program from undergraduate school *A* (since $30 < 43$). This is consistent with the number of people that applied for *MS* to graduate school *S* (since $30 < 37$) and is also consistent with the number of *MS* applicants that got admits (since $30 < 35$). We also note that the total number of applicants from undergraduate school *A* is 50.

Thus, now the probability,

$$\Pr(S, Ad, MS|A) = 30/50 = 0.60$$

The opposite extreme case will occur when as many applicants from undergraduate school *A* that apply to graduate school *S* for the *MS* program get rejected as possible. We have 32 applicants from undergraduate university *A* that apply to graduate school *S*. If we assume the maximum number of these applicants to be *PhD* applicants, we are left with 27 applicants since graduate school *S* has received only 5 *PhD* applications. Of these 27 *MS* applicants to graduate school *S* from undergraduate university *A*, we assume that the maximum possible number of these applicants get rejected. Thus we are left with 15 applicants from undergraduate university *A* that got admitted to the *MS* program at graduate school *S* (there are 12 applications that got rejected by graduate school *S*). Again, we note that the total number of applicants from undergraduate school *A* is 50.

Thus, the probability,

$$Pr(S, Ad, MS|A) = 15/50 = 0.30$$

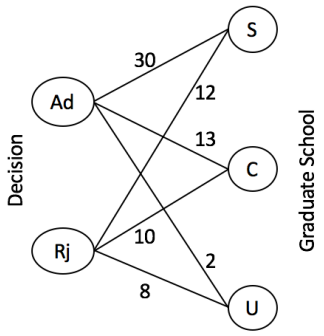

Fig. 6: Graduate School to Decision mapping

These extreme case scenarios provide the bounds within which the actual answer of query (probability) should lie. It can be seen that our approximation model estimates a result within this range formed by the bounds.

$$0.30 \leq 0.3853 \leq 0.60$$

Calculating the exact result on the original graph will take processing of at least 50 nodes (75 in the worst case), connected to university *A*. For our estimation, we needed only 3 nodes from Fig 1, 2 nodes from Fig 2 and 2 nodes from Fig 5 resulting in only 7 nodes. Also, if the same graph were to scale to 50 million nodes, our model wouldn't need 5 million nodes, but most likely require nodes in the order of thousands (based on distinct values of undergraduate and graduate universities). Also, as the size of our original graph increases, the number of attributes participating in a query, and the categorical values of each attribute increase, the bounded window keeps on decreasing, and hence, our algorithm provides even better estimates.

6 Complexity Analysis

6.1 Time Complexity Analysis

The preprocessing step involves constructing $\binom{k}{2}$ summary graphs. For constructing each of these graphs, we will have to traverse the underlying N nodes. Thus, the complexity involved in the preprocessing step is $\mathcal{O}(k^2N)$ The preprocessing step also involves computing aggregate values for each attribute. This will take additional $\mathcal{O}(kN)$ time. Thus the total preprocessing time is $\mathcal{O}(k^2N + kN) = \mathcal{O}(k^2N)$.

Let's now look at the time complexity to process a query once we have the summary graphs in place. For each query, of type $\Pr(a_1, a_2, \cdots, a_q|b_1, b_2, \cdots, b_r)$, there can be a maximum of k^2 graphs that we need to look at. In each of these graphs, we need to look at v values at maximum. Thus, the complexity of processing each query is $\mathcal{O}(k^2v)$. Without the summary graphs, if we were to answer the query based on the entire underlying network, we would require to look at all possible $\binom{k}{2}$ combinations of attributes with v values each over N nodes resulting in a $\mathcal{O}(k^2Nv)$ time complexity. Thus, the preprocessing step reduces the query latency by a factor of N.

6.2 Space Complexity Analysis

As far as the space complexity is concerned, we have $\binom{k}{2}$ summary graphs, each consisting of $\mathcal{O}(v)$ nodes and $\mathcal{O}(v^2)$ edges. Thus, the space complexity can be given by $\mathcal{O}(k^2v^2)$.

7 Experimental Evaluation

7.1 Evaluation Metrics

Our evaluation metrics are based on both the efficiency and accuracy of our algorithm in answering queries of varying complexity. The complexity of a query is based on the number of attributes for which we need to store sub-graphs. Based on this observation, we evaluate our algorithm on the basis of the following metrics:

1) **Per-Query Runtime**: Per-query runtime is defined as the amount of time required to resolve a query as a function of data size. The resolution time of a query according to our model depends on the square of the number of attributes as we have shown in previous sections and independent of the data size. So, we expect the runtime to remain almost constant even when data size changes. In order to calculate the speed-up that we achieve, we have compared our per-attribute query time to the corresponding time required for retrieving the exact answer by processing the entire graph. The Results section shows that this speed-up is significant (in polynomial order of data size).
2) **Per-Query Error**: Since the resolution of a query depends on the Naïve Bayes assumption, we evaluate the accuracy of our query resolution as compared to the exact solution after having examined the entire graph. We define per-query error as the metric of accuracy. It can be defined as:

$$Error = |Accurate\ result\ -\ Estimation|$$

As the proposed query approximation framework is efficient, we are able to achieve results that are within some error margin of the actual solution. This error margin is sufficiently small given the use case for our specific queries.

By evaluating the results obtained by our model on the above two metrics, we can observe the effect of query complexity (based on the number of attributes) on the accuracy of our model. Similar readings can be obtained for the speed-up that we achieve when compared to the scenario when the entire graph must be processed and a series of joins be taken in order to derive the exact solution to a query.

7.2 Experimental Setup

We evaluate both of the above metrics on simple as well as complex queries. The complexity of a query is defined by the number of attributes involved. As the number of attributes increases, more summary graphs are needed to obtain the estimation and thus the query complexity increases. For both types of queries, we evaluate per-query runtime and per-query error for different data sizes. The results of our experiments are discussed in the next section.

8 Results

8.1 Per-Query Runtime

Fig 7 and 8 show the results of our experiments on simple queries. Fig 9 shows the results on a complex query. The plots are per-query runtime as a function of data size. We can see from these figures that the query estimation using summary graph remains almost constant as the data size increases while the time taken for actual query result calculation increases linearly. In Fig 8, our algorithm performs so efficiently that the time is very close to zero. This is because, the number of categorical values for each attribute i.e. v is very small. Even when this v is significantly large, as is the case in Fig 7 and 9, the run time is still constant for our algorithm.

In Fig 7, we can see that initially EQUALGAS is outperformed by the linear processing of data because of compiler and runtime optimizations. The cost of finding and accessing relevant summary graphs causes the extra time because of hashing operations. However, this cost remains constant irrespective of the graph size.

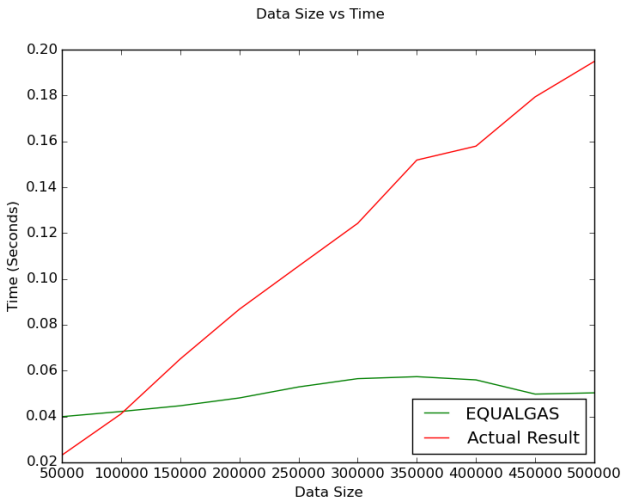

Fig. 7: Data Size vs Runtime for Simple Query: $Pr(MS, NCSU|BITS)$

8.2 Per-Query Error

Fig 10 and 11 show the results of our experiments on simple and complex queries respectively. It can be seen from both these graphs that there is generally a decreasing trend in error as the data size increases. This is expected since, as the data size grows, the bounds for extremities narrow down and our estimate always lies within these bounds.

The first peak for both the graphs is because the data is not yet large enough for probabilities to make sense. But as data size increases, probabilistic results model the actual data much better. The second peak in Fig 11 is because data in the last few chunks is extremely skewed in terms of Graduate School attribute distribution. However, over all error has a decreasing trend. In spite of the skewed nature of data, EQUALGAS always produces results within a 0.025 tolerance window of the actual probability.

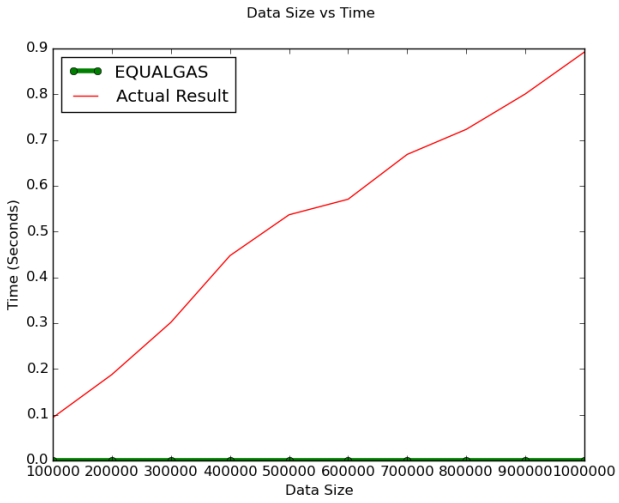

Fig. 8: Data Size vs Runtime for Simple Query: $Pr(MS, CS|Fall)$

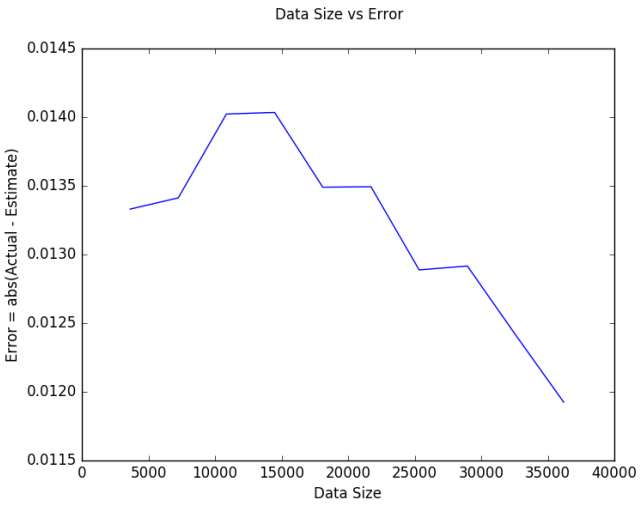

Fig. 10: Data Size vs Error for Simple Query: $Pr(MS, CS|Fall)$

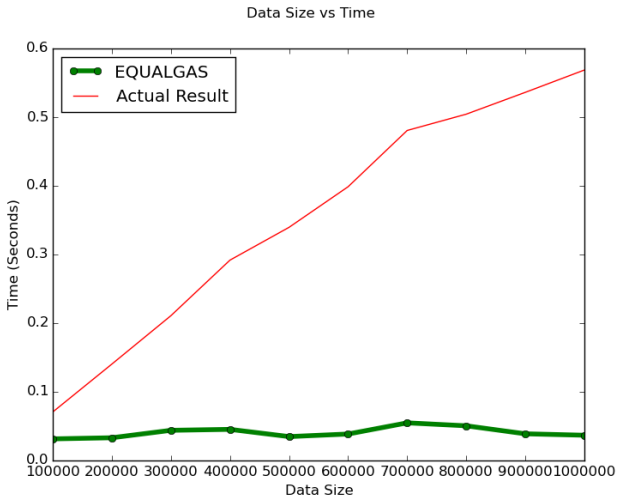

Fig. 9: Data Size vs Runtime for Complex Query: $Pr(MS, CS, UTDallas|Fall, MumbaiUniversity)$

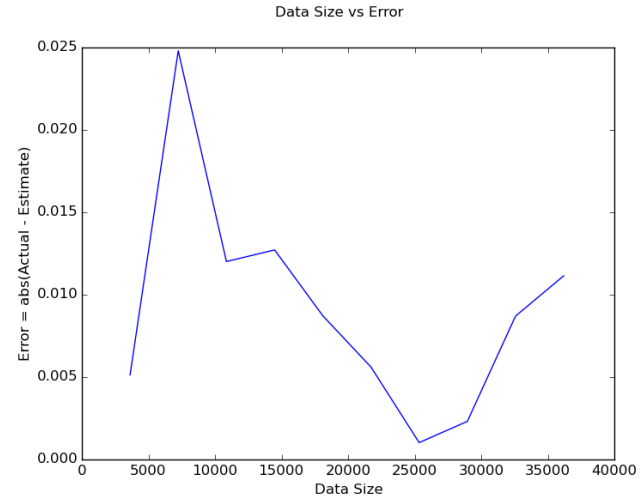

Fig. 11: Data Size vs Error for Simple Query: $Pr(MS, CS, NCSU|Fall, BITS)$

9 Conclusion

In this paper, we have provided an efficient framework for query resolution over large graphs. Our summary graph approach does not depend on the size of the graph. By computing the summary graphs only once, we provide efficient calculations for subsequent queries. We achieve reductions in runtime in order of input data size. Moreover, even for varying complexity of the query and varying data size, EQUALGAS provides probability results within 0.025 of the actual. Our framework is computationally and spatially efficient and can be applied to large scale social networks. Our results are promising and as part of future work, we wish to apply these to other large scale graphs like Facebook and Twitter.

Some of the probable future extensions to our work are lazy initialization of summary graphs using heuristics to predict probable and frequent queries, in order to reduce computations even further. We also plan to extend our framework to capture various social relationships and compute the probabilistic estimates based on these relationships.

References

[1] Leskovec, J. and Faloutsos, C. 2006. Sampling from large graphs. In Proceedings of the 12^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD06). ACM Press, New York, NY, USA, 631636.

[2] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In WWW, 595601, 2004

[3] Wei Liu et al. On Compressing Weighted Time-evolving Graphs. In: (2012)

[4] Qu, Q., Liu, S., Jensen, C. S., Zhu, F., & Faloutsos, C. (2014). Interestingness-Driven Diffusion Process Summarization in Dynamic Networks. In Machine Learning and Knowledge Discovery in Databases (pp. 597-613). Springer Berlin Heidelberg

[5] Ahmed, N. K., Neville, J., and Kompella, R., 2013. Network sampling: from static to streaming graphs. TKDD

[6] S.Sakr and G. Al-Naymat. Efficient Relational Techniques for Processing Graph Queries
[7] M. Saber, M. Aref, T. Gharib. An Efficient Filtering Technique for Super-Graph Query Processing
[8] P. Zhao and J. Han. On Graph Query Optimization in Large Networks
[9] S. Kassegne. Edulix - premier site for scholars - 'education crowd-sourced'.
[10] Dagum, Paul and Luby, Michael, Approximating Probabilistic Inference in Bayesian Belief Networks is NP-hard, Artif. Intell., March 1993 Elsevier Science Publishers Ltd. Essex, UK.