Querying and Mining Graph Data Streams

February 23, 2015

1 Introduction

Graphs encode the interconnections between entities to model relations between them.They have proven to be an intuitive datastructure in modelling, querying and reasoning about real-life relationships. In websearch the graphs derived from links between webpages are used for link analysis and establishing authority of webpages. In social networks, the graph structure captures interesting insights about the influential nodes, community structures, connectivity and reachability etc. In biological networks people are interested in how diseases spread. In ontologies, they are useful to model taxonomies in the form of type hierarchies. Apart from these scenarios they find applications in road, water networks for shortest path computations and flow problems.

Traditional research has focussed on largely on *static graphs*. However, as one might quickly realize, most of the scenarios enumerated above have a evolutionary nature to them. Social networks, Web graphs, citation networks and even biological networks evolve over time. So there has been a growing need to support data management, querying, mining and analytics tasks over evolving or time-varying graphs. Moreso, to date there is little work on understanding the challenges needed to ingest, manage and query such data in large graph databases. In this proposal we attempt to scope out the problems relating to dynamic graphs and more specifically time-varying graphs with an intent to identify open issues and problems which would be worth investigating.

> ✍ **To-do**: Certain examples neededlike shortest paths, k core decomposition, matchings

2 State of the Art and Preliminary Work

We divide the related work survey into two sections: (a) algorithmic foundations on streaming graph primitives, and (b) indexing and querying for streaming graphs.

2.1 State of the Art

> ☞ **Note**: add something

2.2 Preliminary Work

> ✍ **To-do**: what we did ? How our work would be used in this project for example :
>
> - (Megha) Studied k cores , dense components, matching
>
> - (Avishek) :

2.2.1 k cores

A bipartite graph can been seen as a hypergraph. In [2] we estimate the threshold for the density of the random uniform hypergraph when the k core first appears. Such a threshold is associated with the probability of existence of perfect matchings in the associated bipartite graphs. Such estimates are quite useful ....

2.2.2 Matching in a Streaming Model

Two natural variants of this problem have been considered in the literature: (1) the edge arrival setting, where edges arrive in the stream and (2) the vertex arrival setting, where vertices on one side of the graph arrive in the stream together with all their incident edges. The latter setting has also been studied extensively in the context of online algorithms, where each arriving vertex has to either be matched irrevocably or discarded upon arrival.

The basic algorithms for finding matchings rely on the definition of augmenting paths: given a matching M in the graph, an augmenting path is an odd-length (simple) path with its edges alternating between being in M and not, and with the two end edges not in M. In order to increase the size of the matching, one starts with an unmatched vertex and walks along the augmenting path to reach another unmatched vertex. The unmached edges of the previous matching now are the matched edges and vice versa.

One can employ various methods to find these augmenting paths; namely breadth first search (BFS), depth first search (DFS) and random walk. The BFS and DFS approaches require the full knowledge of the graph and hence are not ideal for streaming models. The random walk method on the one hand chooses each vertex on the augmenting path randomly and is a stateless approach but on the other hand has no gurantees of the convergence in the worst case.

The other class of algorithms employed for finding online matchings is *greedy* approach where each edge is included in the matching if it does not destroy the matching. At present this is the best single pass algorithm providing 2 approximation when the edges are streamed in an arbitrary order. The strongest known lower bound for the single pass algorithms is $e/(e-1) \approx 1.58$ which also applies when the edges are grouped by end point [5, 3].

In the model that we consider, we are given a bipartite graph $G(U, V, E)$, in which U is known to the algorithm, vertices in V are unknown, but arrive one at a time, revealing the edges incident on them as they arrive. The algorithm has to match a vertex as soon

as it arrives. In addition each vertex in V has k incident edges and the other end of such edges are chosen uniformly randomly and independently. This model is motivated from a class of multiple choice load balancing problems in which each job ($u \in U$) chooses k machines uniformly randomly and independently. In [2] we prove that there exists a threshold such that when the density of G is below that threshold a (left-) perfect matching (i.e. all jobs are assigned to one of their choices) exists, otherwise not. In [6] we propose an algorithm which we call *local search allocation* finds maximum cardinality matching in linear time with high probability. Our approach in [6] operates by assigning labels to vertices in V. The vertices for augmenting path ae chosen based on their labels. The labels are so updated such that at any time the label of a vertex is atmost the shortest distance to an unmatched vertex. In addition to the efficiency of the approach, the algorithm never runs in loops and gives the information when the vertex cannot be matched. Emperical evidence suggests that our approach performs an order of magnitude better than the random walk method.

As a starting point our proposed research focusses on the following questions.

1. Can we extend/adapt local search allocation algorithm to find approximate maximum cardinality matchings in the streaming model in which a bipartite graph is given by a stream of edges in an arbitrary order? Such a semi streaming algoritm has memory $O(n \text{ polylog } n)$, and the graph vertices are known before processing the stream of edges.

2.

state bounds for maximum cardinality matchings in bipartite graphs and general graphs explain greedy algorithm ... variants of greedy algorithm improving bounds..not much improvement theretically Introduce local search allocation for online setting....the model is different from the above considered models...a new idea based on approximate augmenting path lengths ; simple to implement scope to adapt the algorithm to general graphs ; there are of course problems in breaking the symmetry we aim to develop new algorithms based on local search method proposed in .... for computing aproximate matchings for online model

We consider a streaming model

2.3 Indexing and Query Processing

Within the streaming model, there are important applications of matching algorithms like *internet advertisements*, *load balancing in cloud computing*, *switch scheduling* etc. Based on the research problems in the previous section, we now turn our attention to how to utilize those results into building indexing frameworks to efficiently answer *matching-based queries* or simply *matching queries*. Indexing large graphs to support primitive operations like reachability [7], shortest paths [4], dense subgraphs [1] have been explored in the graph databases literature with reasonable success. As an example, a specialized index can be created over an input graph to answer *reachability queries*, i.e., if a pair of nodes are reachable from each other. Similarly, we intend to design

efficient indexing methods for streaming graph input to find matchings. Similarly for massive graphs, matchings for arbitrary subgraphs can be efficiently computed if some kind of an index is constructed over the input graph.

2.3.1 Matching-based Queries

We consider two types of queries : *standing matching queries* and *adhoc matching queries*. Standing queries (also called as publisher-subscriber queries) are a static set of queries which are checked for satisfaction in the face of changing or dynamic input data. Examples are dense-subgraph queries proposed by Angel et. al. [1] which reports dense subgraphs whenever streaming edges result in a subgraph becoming substantially dense. The query in that case is a desired density coefficient. In our scenario, we could also be interested in potentially new matchings given a stream of incoming edges/nodes. The second class of queries are the more standard adhoc queries, where user can desire a given type of matching on a given subgraph. As an example, for time-varying graphs a user could be interested in matchings for the graph which was valid in a given time interval.

2.3.2 Research Questions

In this respect we can formulate the following research questions which we intend to answer:

- [RQ I] How can we *efficiently construct* indexes specialized for answering matching queries ?

- [RQ II] What kind of index-maintenance strategies is needed to be employed to avoid partial or complete index recomp utations ?

- [RQ II] What query processing techniques need to be employed for both *standing* and *adhoc* matching queries ?

3 Objectives and Work Programme

References

[1] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB Journal*, 23(2):175–199, October 2013.

[2] N. Fountoulakis, M. Khosla, and K. Panagiotou. The multiple-orientability thresholds for random hypergraphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1222–1236, 2011.

[3] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 468–485, 2012.

[4] Andrey Gubichev, Srikanta Bedathur, Stephan Seufert, and Gerhard Weikum. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 499–508, New York, NY, USA, 2010. ACM.

[5] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1263–1282, 2015.

[6] M. Khosla. Balls into bins made faster. In *Algorithms–ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 601–612. 2013.

[7] Stephan Seufert, Avishek Anand, Srikanta Bedathur, and Gerhard Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1009–1020. IEEE, 2013.