Alternative Thoughts About Processing Queries on Large Scale Networks

Motivation

We are seeking more efficient methods in solving queries on contact networks and simulation results produced by EpiHiper than those that need to load in the entire time series of contact networks and create full indexes on node/edge properties. And, if possible, we want to generalize these methods to arbitrary networks. In addition, we also want to understand how much and what information is required to fulfill different types of queries, and when only partial information is provided whether we could attain acceptable approximate query results.

Note that sometimes the intermediate contact networks produced by EpiHiper are called `transmission trees`. To avoid confusion, we reserve the name `initial contact network` to the input contact network for EpiHiper, and we call those intermediate networks produced by EpiHiper `intermediate contact networks` or more briefly `intermediate networks`.

Beforehand, let us review why graph DBMS or relational DBMS would take a long time at the loading and indexing stages.

(1) Neo4j: Too Many Edge Copies

When using Neo4j, since Neo4j does not support friendly array-like data types for node/edge properties which can be indexed, it becomes difficult for us to stuff the temporal information (i.e. time stamps of edges) into a single edge property while keeping executing temporal queries efficiently.

Limited by this issue, we trade off the space consumption, and load in all edges in all contact networks into the Neo4j DB. And each edge is associated with a simple integer property indicating the time point at which this edge occurs. Doing this enables us bypass the difficulty of using array-like data types for storing the temporal information on edges. And an index is created on that integer edge property.

However, this method significantly increases the total number of edges involved in the indexing on edge properties. And, in fact, for the edge properties other than the time stamp, it may not be necessary to create indexes on all edges over time. Rather, an index for each time point can be sufficient for queries.

(2) Eliminating Edge Copies May Not Solve the Problem

First, when temporal networks are stored a database, creating an index on an edge property will usually take into account at least all unique edges, which, again, may not be necessary for fulfilling queries.

Second, usually indexes are created individually, and B-tree data structures are used. Constructing a B-tree index can cost $O(nt\log_t n)$, where n is the size of data and t is the minimum degree of the B-tree. Also, sorting the data beforehand benefits insertions, which also $O(n\log n)$. Empirically, sorting an edge file containing 1.4 billion records takes about 1 hour on a Rivanna computing node (Intel Xeon Gold 6148 @ 2.40GHz x 40 threads, 376GB memory). In other words, for each edge property, creating an index may take a few hours. And, when the number of edge properties requiring indexing increases, the running time of creating all indexes may be prolonged due to the limits of computing resources.

To sum up, no matter graph databases or relational databases are in use, it is likely that we may not be able to gain any solution with satisfactory running time for data loading and indexing without utilizing distributed computing resources and advanced parallelization.

On the other hand, it is also interesting to us what queries can be acceptably fulfilled with limited computing resources, if any. In other words, the following question is asked:

> **what types of queries do not need all information of the entire time series of contact networks?**

More specifically, the queries, in the worst case, that can be fulfilled without the need of accessing every node and edge in all contact networks are wanted.

Let us take a look at a counter example first:

> What are PIDs of individuals who were infected by individuals whose ages are between 21 and 45?

For a specific simulation run, the results for this query may not include the entire set of nodes at each simulated time point; however, in the worst case, at each simulated time point, every individual may have been infected by the individuals whose ages are between 21 and 45, and thus every node and edge in all contact networks are needed to produce the results. This type of queries is not in our primary consideration.

Now, let us modify the above query to the following:

> What is the age distribution of individuals who were infected by individuals whose ages are between 21 and 45?

To provide an acceptable answer to this question, in the worst case, do we have to consider every node and edge in all contact networks? Intuitively, first, we can sample from the individuals whose ages are between 21 and 45, second, we can further sample from the infected individuals, and finally, we compute an approximate age distribution based on the samples. If the resulting approximate distribution is acceptably close to the desired/empirical distribution, then we say that we do not have to consider every node and node in all contact networks.

From these two examples, we learned that:

- **Queries inquiring about distributions** may be acceptably fulfilled by accessing only a limited portion of data.
- Sampling is a key in delivering acceptable results.
- Evaluating the quality of query results is an important problem.

Note that other types of queries that do not require all information of the entire time series of contact networks may also exist. And we are in discovery of other possibilities.

Note for Caution

Identifying classes of queries that do not need the full information of the time series of contact network does not necessarily mean that in general we would be able to process queries more efficient than using DBMS. When doing sampling, the full information may be also needed depending on what types of sampling is performed. For example, when sampling from imbalanced data, clustering information is usually needed, which may require sorting. In this case, if all node/edge properties are considered in queries, then the overall running time may not be better than that using DBMS, if not worse, because it may need a sorting on each property. On the other hand, if a sampling method with a lower time complexity than $O(n \log n)$ can fulfill a non-trivial class of queries acceptably and it only needs to be run once, then this class is outstanding.

Queries Inquiring About Distributions (QuID)

This is a class of queries.

> **Definition**: **QuID**
> A QuID query requires a distribution as its desired result.

It is necessary to understand what are the items upon which the desired distributions are computed. And the following is a collection (may not be the full list):

- Node properties (e.g. age and gender)

- Edge properties (e.g. activity type and duration)
- Properties of graph structures (e.g. degree and shortest path length)

Note that it may not be true that we can find an acceptable approximate result for every QuID query. As mentioned before, the quality of an approximate result needs to be evaluated by the distance/divergence between this approximate result and the desired/empirical result. If the distance/divergence falls into a predetermined range of acceptance, the approximate result is said to be *accepted*.

Also note that there are quite a number of candidates for the distance/divergence such as f-divergence, JS-divergence, Bhattacharyya distance, Wasserstein distance and energy distance. It is necessary to clarify which distance/divergence is in use when accepting an approximate result.

> **Definition**: **QuID-AAR**
> We define a sub-class of QuID in which each member query corresponds to at least one acceptable approximate result w.r.t. at least one distance/divergence with a predetermined acceptance range. And we name this sub-class **QuID with Acceptable Approximate Results (QuID-AAR)**. In addition, for clarification, a member in QuID-AAR should be said to be **α-Φ-accepted**, where α is the value of distance/divergence and Φ is the name of the distance/divergence (in abbr.) (e.g. 0.1-Wasserstein-accepted).

Note that QuID itself is a QuID-AAR if the acceptance range is arbitrary.

We are also interested in how the values of distance/divergence worsens when the size of samples for output decreases regardless of the sampling methods and how the final result is computed. In other words, for each QuID query, we seek a function to reflect such relationships:

> **Definition**: **Sample-Size Approximation-Quality Function (SSAQF)**
> $\rho : |\hat{G}| \mapsto \inf_{\varphi, \hat{G}} d(\varphi(\hat{G}), f)$, where $\hat{G}$ is a sample of the time series of contact networks, φ is a method computing the approximate result, f is the desired/empirical distribution and d denotes the selected distance/divergence.

SSAQFs of QuID queries may have significantly different characteristics (e.g. linear, sublinear, superlinear, quadratic, exponential and etc.). These characteristics provide an approach to measure the "hardness" of the queries.

In addition, seeking a SSAQF is an optimization problem. Fully solving this problem can be difficult, and thus imposing constraints to this problems may be necessary.

Sampling for QuID

As mentioned above, node properties, edge properties and graph structure properties are three typical items on which distributions may be desired. This also means that these items can also appear as conditions of given queries. Thus, to fulfill a QuID query, the sampling may be performed multiple times on different items.

For queries exclusively requiring node/edge properties, when multiple properties are involved, a sampling from the multivariate distribution is needed. In this case, the multivariate data can be imbalanced and clustering information may be needed (i.e. one or multiple sorting may be needed and each costs $O(n \log n)$). Sampling from multivariate distributions is an art of its own. And it can be done more efficiently only if some special structures or correlations exist among the dimensions.

On the other hand, sampling on graph structures is less straightforward (e.g. sampling shortest paths). Such sampling may need to be indirectly performed on nodes and edges. Let us take a look at an example.

> We want to retrieve the distribution of shortest path lengths.

Suppose the sampling is performed on edges uniformly. In this case, the expected length of shortest paths in the sample may be lower than that in the original graph, and thus this sampling would result in a biased estimator, which may lead to misleading results.

Seeking appropriate sampling methods on graph structures is a non-trivial problem, especially when the graph model of networks is unknown.

Mapping From Queries to Graph Properties

Since nodes and edges can have arbitrary numbers of properties, it may be helpful if the relationships between sampling on the properties using specific methods and the variation of some graph properties. Such relationships can help cluster queries into groups. And it would be more generic studying the variation of graph properties and the variance of the quality of approximate results.