Implementing & Evaluating Space Efficient Algoirthms for Detecting Large Neighbourhoods in Graph Streams

Dom Hutchinson

February 6, 2020

Chapter 1

Background

1.1 Motivation

For most algorithms time efficiency is viewed with primary importance, rather than space efficiency. However if the memory used for an algorithm exceeds the size of the computer's RAM then virtual memory would have be used. Virtual memory has a much longer read/write than RAM meaning that the algorithm's run-time will be much greater in this scenario. Thus for algorithms which are designed to work with large data sets space efficiency is more important.

In the early 2000s companies began to realise that data could be used as a commodity, and with this the amount of data being collected has surged in the years since. In this same time period we have nots seen the same surge in the amount of RAM typical computers have. Thus people wishing to capitalise on this increase in available data need to give greather thought to the space efficiency of the algorithms they are using. Otherwise run-times could become unusuable.

Social networks are good sources of data about people. This data can leveraged in many ways, notably within advertising. Behaviourly targeted adverts have been shown to have more than twice the click through rate of standard online advertiesments [2]. The last decade has seen developments in the space of targeted advertising, one of which is the rise of social media influencers. These are people who are deemed to have a large online following, which they are able to influence. Advertisers will often approach these influencers with deals by which the influencer will promote a product to their audience, hopefully driving sales.

The question becomes, "Who is an influencer & how do we find them?". This question can be generalised as the `Neighborhood Detection Problem`. Which is the problem I shall discuss in this paper. Although in my description of the Neighbourhood Detection Problem I validate its relevance by referring to find social media influencers, it is applicable to many other

scenarios.

- TODO

1.2 Neighbourhood Detection Problem

Graph streams are a dynamic way of representing graphs. They are simply a list of instructions for how to construct a graph and in this paper I shall discuss two types: insertion-only streams & insertion-deletion streams. There are other types which allow for updating vertex values, among other properties, but they are not relevant to the problem we are considering in this paper.

Insertion-only streams are a list of pairs of vertices, representing endpoints of edges. Each time we read an entry from an insertion-only stream we are adding the edge it describes to the graph. We apply a limitation that no duplicate edegs appear in an insertion-only stream and thus the order of the stream has no affect on the validity of the algorithms which shall be discussed.

Entries in an insertion-deletion list specify a pair of vertices, representing an edge, and a boolean which states whether this is an insertion or deletion entry. If it is an insertion entry then we are adding the described edge to the graph; if it is a deletion entry then we are removing it. Again we add a limitation that no edges are repeated & that we can not recieve a request to delete an edge from the graph which does not currently exist in the graph. This does however mean that the order of an insertion-deletion is important as the deletion of an edge must come after it.

Despite these limitations it is still possible to construct almost any type of graph from a graph stream. Suppose you wish to analyse which devices in a network are making the most request. Here vertices represent devices & edges represent requests. In this scenario you would want to allow for multiple edges (say $n > 2$) between the same pair of devices (say $\{x, y\}$). One solution would be to add n new vertices $\{v_{x,y,1}, \ldots, v_{x,y,n}\}$ each with a label stating they are just indetermiate nodes and not full devices. Creating edges from x to each of $\{v_{x,y,1}, \ldots, v_{x,y,n}\}$ & from y to each of $\{v_{x,y,1}, \ldots, v_{x,y,n}\}$ allows for an interpretation of multiple edges.

If a graph stream is ordered by time (which is common in real life applications) then it represents a dynamic-evolution of the graph over time. Allow the user to roll back the graph should they wish to make assessments during particular time preiods, rather than just assess the final graph.

The algorithms which shall be discussed in this paper are designed to work with just a single pass of a graph stream. This is not only time efficient as it does not require multiple reads of the stream, but means that should more information be gained after the algorithm is run it is easy to incorporate it into algorithm if we store the final state of the data structures

used. Single pass algorithms mean we don't necessarily have to store the graph stream, we could simply send entries from the system being modelled straight into the algorithm, without being stored in a file. This is useful when a graph stream potentially could take up multiple terra-bytes, but is unnecessary in most applications.

The `High Degree Detection Probem` is a problem in graph theory where we seek to find a vertex of sufficiently high degree, in a given graph. This node does not necessarily have to have the greatest degree in the graph, and often we cannot be sure whether it does as the greatest degree is unknown beforehand. This problem can be formally stated as

Problem 1 `High Degree Detection`.
Let $G = (A \cup B, E)$ be a bi-partite graph with vertex sets A, B, where $|A| = n$ and $|B| =$ poly n , and edge set E. Set a restriction that at least one vertex in A has degree, at least, d.
In `High Degree Detection Detection`(G, d) we are tasked with outputting a vertex from A with degree, at least, d.

This problem can be solved efficiently for both insertion-only & insertion-deletion graph streams with a fairly simple algorithm which simply runs through all the edges, incrementing or decrementing (depending on whether it is an insertion or deletion edge) a count for the degree of each vertex as it goes. Returning the first vertex it finds with degree equal to d.

Algorithm 1: Single Pass High Degree Detection

require: Stream $\{(i_0, x_0, y_0) \dots (i_n, x_n, y_n)\}$, degree bond d
$D \leftarrow \{\{\}\}$ {degree map}
for $j \in [0 \dots n]$ **do**
  if $x_j \in D$ **then**
    if i_j **then** $D[x_j] -= 1$;
    else $D[x_j] += 1$;
  else $D[x_j] = 1$;
  if $D[x_j] = d$ **then return** x_j;
  repeat with y_j
return *FAIL*

In **Algorithm 1** i is a boolean which is true if the edge is an insertion edge, and false if it is a deletion edge. x & y are the two endpoints of the edge. The algorithm makes the reasonable assumption that you will never recieve a deletion edge for an edge which is not in the graph, nor shall you recieve the same edge twice.

The `High Degree Detection Probem` is limited in its applications as it only returns a vertex with high degree & gives you no information as to

its neighbourhood. Solving this problem would not be very helpful when it comes to choosing influencers as it would only return people who have large followings, but not who their followers are. Thus an advertisier would not be able to evaluate whether their followers represented the audience they wished to market to.

The `Neighbourhood Detection Problem` is an extension of the `High Degree Detection Problem`. The task is to return a vertex with sufficiently high degree & a certain proportion that vertex's neighbourhood. This is formally stated as

Problem 2 `Neighbourhood Detection`.
Let $G = (A \cup B, E)$ be a bi-partite graph with vertex sets A, B, where $|A| = n$ and $|B| =$ poly n, and edge set E.
In `Neighbourhood Detection`(G, d, c) we are tasked with outputting a vertex from A and at least d/c of its neighbours in B.
Here d is a threshold parameter & c is an approximation parameter.

Solving the `Neighbourhood Detection Problem` allows for a much more general assessment of the influence of the returned vertex. In the context of finding social media influences the returned neighbourhood can be assessed to determine whether the returned person is suitable for a given advertising campaign. When assessing the members of the neighbourhood you could look at the demographics of the members & how active the members are (among other features) in order to determine whether they fit the target of the campaign. For cases where $c \neq 1$ you only get a fraction of a vertices whole neighbourhood but for sufficiently high d you an implement many statistical tools to still be able to draw meaningful inferences. Later it shall be shown that there are meaningful advantages to increasing the value of c.

1.3 Why Space Efficiency?

Simply solving the `Neighbourhood Detection Problem` is fairly trivial. **Algorithm 2** is an algorithm for solving the `Neighbourhood Detection Problem` for insertion-only graph streams by recording the vertices in the neighbourhood of every A-vertex and then returning the first neighbourhood we enounter with $\frac{d}{c}$ members.

Algorithm 2: Naïve Single-Pass Insertion-Streaming Algorithm for Neighbourhood Detection

require: Stream $\{(s_0, t_0) \dots (s_n, t_n)\}$, degree bound d, precision bound c

$N \leftarrow \{\{\}\}$ {neighbourhoods}
for $i = 0 \dots n$ **do**
  append t_i to $N[s_i]$
  if *size*$(N[s_i]) \geq \frac{d}{c}$ **then return** $(s_i, N[s_i])$;
return *FAIL*

Algorithm 2 requires $O\left(n^2\right)$ space. Thus the space used to run the algorithm grows very quickly which could easily lead to the computer's RAM overflowing, increasing execution time. This is undesirable, especially given the run time complexity of **Algorithm 2** is alread bad, $O(n^2)$. For solutions which are space efficient run times become unmanagable for large graph streams. In 2014 in the UK, Facebook had 36.68 million users [3] each with an average of 155 friend connections [4]. In order for this to be represented in a bi-partite graph for `neighbourhood detection` we would need a graph stream with ∼5.6 billion edges. **Algorithm 2** would not be suitable for analysing this graph.

Instead we must look for much more space efficient algorithms for `neighbourhood detection`. In the rest of this paper I shall discuss & test an algorithm which can solve `neighbourhood detection` for insertion-only streams with space $O(n \log n + n^{\frac{1}{c}} d \log^2 n)$; and, two approaches to solving for `neighbourhood detection` for insertion-deletion streams with space $\tilde{O}\left(\frac{xd}{c}\right)$ & $\tilde{O}\left(\frac{nd}{c}\left(\frac{1}{x} + \frac{1}{x}\right)\right)$ respectively.

1.4 Plan

In this paper I shall implement and test the two algorithms presented by Dr Chrisian Konrad in [1]. These algorithms solve the *Neighbourhood Detection Problem* for insertion-only & insertion-deletion graph streams in a space-efficient manner. Both algorithms require only a single pass of the stream.

I shall use the naïve algorithm described in **Algorithm 2** as a benchmark for both time & space efficiency of these two algorithms. I shall use a four different graph streams, described in **Table 1.1**, for my tests. These graphs were acquired from the *Stanford Network Analysis Project*, [5].

The theoretical space requirements for these two algorithms both depend on the degree variable, d, and precision variable, c. I shall perform tests where I vary each of these in order to see what the space requirements are in practice, and then compare these results to the theory.

Table 1.1: Test Graph Streams

Name	# Edges	# Vertices	Max Degree	File Size, KB
`facebook_small`	292	52	36	3
`facebook`	60,050	747	586	587
`gplus`	1,179,613	12,417	5,948	52,839
`gplus_large`	30,238,035	120,100	104,947	1,328,820

Bibliography

[1] Christian Konrad *Streaming Frequent Items with Timestamps and Detecting Large Neighborhoods in Graph Streams.* November 2019

[2] Howard Beales *The Value of Behavioral Targeting.*

[3] https://www.statista.com/statistics/553538/predicted-number-of-facebook-users-in-the-united-kingdom-uk/

[4] https://www.telegraph.co.uk/news/science/science-news/12108412/Facebook-users-have-155-friends-but-would-trust-just-four-in-a-crisis.html

[5] http://snap.stanford.edu/data/