A Tractable Approach to Finding Closest Truncated-hitting-time Neighbors in Large Graphs

Purnamrita Sarkar (presenter) PSARKAR@CS.CMU.EDU
CALD,School of Computer Science,Carnegie Mellon University, Pittsburgh, PA 15213

Andrew Moore AWM@GOOGLE.COM
Google Inc. Pittsburgh, PA 15213

Recently there has been much interest in graph-based learning problems. This has applications in collaborative filtering in recommender networks, link prediction in social networks or fraud detection. These networks can be consisting of millions of entities, and henceforth it is very important to develop highly efficient techniques. We are specially interested in accelerating random walk approaches to compute some very interesting measures of these kinds of graphs. These metrics have been shown to do well empirically by authors in (Liben-Nowell & Kleinberg, 2003), (Brand, 2005).

Let G be an undirected graph of vertices V, and edges E. Each edge has a real-valued weight. In a random walk the probability of moving from node i to neighboring node j is proportional to the weight of the edge w_{ij}. The hitting time (Aldous & Fill,) $H(i,j)$ or h_{ij} is defined as the expected time to hit node j for the first time starting at node i. The commute time c_{ij} between a pair of nodes is $h_{ij} + h_{ji}$. These measures decrease if there are many short paths between the nodes. On the other hand , as observed by (Liben-Nowell & Kleinberg, 2003), two major setbacks are that they tend to be small whenever one of the nodes have a high degree, and they are also sensitive to parts of the graph far away from the nodes, even when there are short paths between the nodes.

To avoid these problems we define a T truncated hitting time, where we only consider paths of length less than T. The measures described above can be computed in closed form from the pseudo inverse of the graph laplacian(Saerens et al.,). In most applications this cubic computation is avoided by using sparse matrix manipulation techniques (Brand, 2005). But it is still expensive beyond a few thousand nodes. We present a very novel algorithm to compute all-pairs of approximate nearest neighbors in hitting times , without computing the full matrix.

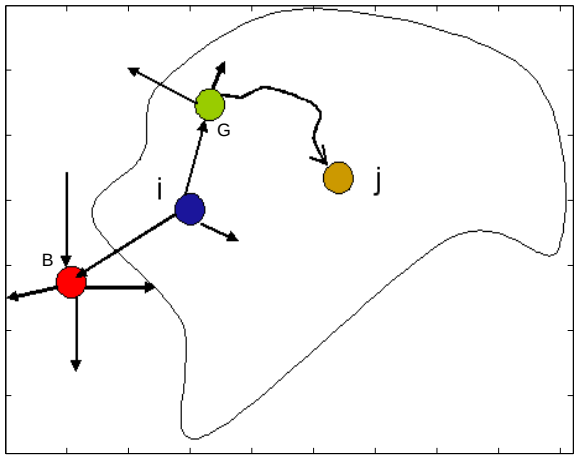

Figure 1. Neighborhood of node j, we have drawn a directed graph for clarity.

Lets start with the hitting times matrix H. The truncated hitting time h_{ij}^T can be defined recursively as $h_{ij}^T = 1 + \sum_k p_{ik} h_{kj}^{T-1}$, where h is defined to be zero if $i = j$, or if $T = 0$.

We compute upper and lower bounds of the h values, and use them to return k ϵ-approximate nearest neighbors. Given k, and ϵ we want to return any k neighbors j of i, s.t. $H(i,j) \leq H(i, kth\_true\_nearest\_nb)(1+\epsilon)$. This is interesting since in all applications hitting and commute times are used for ranking entities, e.g. recommend the k best movies based on choices already made by an user; or, find the k most likely co-authors of an author in a co-authorship network.

Suppose we want to estimate H_{ij} for pairs of nodes i, j which have relatively low H_{ij} values, and suppose we want to avoid n^2 time or space. So we cannot do anything that requires representing or iterating over all pairs of nodes. We in fact cannot even afford to iterate over all pairs of nodes that are less than T hops

Category : Data Mining
Preference : Oral

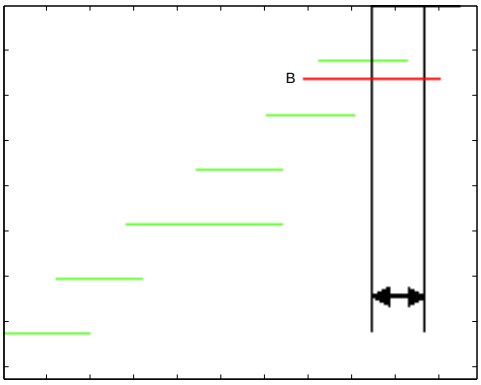

Figure 2. Upper and lower bounds of $H(i,*)$

apart. So suppose we restrict computation to iterate only over some subset of pairs of nodes, which we will call the APset. The interesting question is how good are the upper and lower bounds we can get on H_{ij}, $\forall i,j$ using only $O(|AP|)$ work ?

For each node we consider its own neighborhood. Each neighborhood has a boundary. Lets first assume that the neighborhood is given. Its clear from the expression for truncated hitting time that h_{ij} can be computed from the hitting time of i's neighbors to j. In Figure 1 the random walk from i can hit either of its neighbors G or B. Since G is inside the neighborhood of j we already have an estimate of h_{Gj}. However B is outside the neighborhood of j , and hence we do an optimistic and pessimistic approximation of h_{Bj}, resulting into an upper(HP) and lower(HO) bound of h_{ij}. The optimistic bound is obtained by allowing the random walk to jump to the boundary node with closest optimistic hitting time to j, and the pessimistic bound arises from the fact that the walk might never come back to the neighborhood after leaving it, i.e. takes T time.

Lets now look at how to obtain the k ϵ approximate nearest neighbors for each node. From figure 2 lets assume that we want to find the 7 closest nearest neighbors of node i. Now the 8^{th} nearest neighbor's hitting time is greater than the 8^{th} largest HO value. Now we also allow a small error margin of ϵ. Hence the new upper bound becomes $8^{th}\_largest\_HO(i,*)(1+\epsilon)$. All entities with upper bound less than this are guaranteed to be ϵ approximate. We can see that in figure 2 all nodes but B are within ϵ error of the true 8^{th} nearest neighbor of i. Hence the neighborhood of B needs to be expanded in order to tighten h_{iB}.

Now lets come to the topic of neighborhood expansion. We start will all nodes within 1 or 2 hops of the destination. We use different heuristics to find the best nodes to put into the neighborhood of j. The two most intuitive ones are to put all neighbors of the boundary node with smallest HO value to j. This ends up resulting in uniform surfaces around the destination in the H space. We can also put the nodes m s.t. $\sum_{k \in AP(*,j)} p_{km}$ is large. These nodes will tighten loose bounds , since we effectively end up including more and more probability mass within the neighborhood.

Now lets look at how the AP set grows for different graphs. We quantify a graph in terms of its intrinsic dimensionality(R. Krauthgamer, 2003). Given a ball of radius r this tells us how fast the size of the ball grows w.r.t. r. For example a grid has dimensionality 2, where as a k dimensional hypercube has a dimensionality of k.

We include some initial experimental results on simulated graphs. We generate graphs of dimensionality k, by assigning k dimensional euclidian coordinates to the nodes, and adding links between close neighbors, and some long distance links. This brings about the very famous small-world phenomenon (Kleinberg, 2000). Initial experiments show that we need relatively small fractions of all pairs to get a good estimate of the hitting times. E.g. for 1000 nodes, $T = 10$, we can find 10 nearest neighbors for each node using around 40000 pairs for $\epsilon = .001$, 20000 pairs for $\epsilon = .01$, 14000 pairs for $\epsilon = .05$. This shows the tradeoff between time complexity and accuracy of the algorithm.

References

Aldous, D., & Fill, J. A. Reversible markov chains.

Brand, M. (2005). A Random Walks Perspective on Maximizing Satisfaction and Profit. *SIAM '05.*

Kleinberg, J. (2000). The Small-World Phenomenon: An Algorithmic Perspective. *Proceedings of the 32nd ACM Symposium on Theory of Computing.*

Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. *CIKM '03 Proceedings of the twelfth international conference on Information and knowledge management.*

R. Krauthgamer, J. R. L. (2003). The intrinsic dimensionality of graphs. *Proceedings of the 35th ACM Symposium on Theory of Computing.*

Saerens, M., Fouss, F., Yen, L., & Dupont, P. The principal component analysis of a graph and its relationships to spectral clustering.