

Finding Top- k Topic-Dense Vertex Sets in Large Graphs

CS290D Project Report

Nan Li, Arijit Khan

Department of Computer Science
University of California, Santa Barbara
{nanli, arijitkhan}@cs.ucsb.edu

Abstract

Driven by large volumes of graph-structured data generated by a variety of applications, querying over graphs has been enjoying an increasing number of applications over the recent years. Meanwhile, with the evolution of more sophisticated graph structures, graph query types start to take various new forms as well. While searching a graph, what is of interest is not only the topological structure, but also the labels of features associated with the graph components. In this paper, we investigate a typical search problem in vertex-labeled graphs: finding the top vertex sets with the highest density in terms of query topics in a large graph. We propose the definition for topic density of a vertex set, as well as an ensemble framework incorporating graph dimensionality reduction to find the densest sets efficiently. In the proposed framework, a graph is first mapped to a two-dimensional Euclidean space using pairwise distance preserving mapping technique. A 2-approximation algorithm based on geometric properties is further applied in the two-dimensional space to find the densest point sets. Experiment results demonstrate the efficiency and effectiveness of the proposed framework. It is empirically proven that, compared to naive method, our method is able to reduce the run time by orders of magnitude.

1 Introduction

Graphs and networks are ubiquitous, encoding complex relationships ranging from chemical bonds to social interactions. Meanwhile graph search is becoming increasingly important in a variety of applications. For modern heterogeneous graphs, such as large-scale information networks, graph search can be conducted at various levels. To gain a deep understanding of such graphs, it is essential to investigate a variety of properties and components of the graphs, i.e. vertices, edges, connectivity, and their associated features and labels. For searching vertex sets with certain properties in large graphs, what is of interest is not only the topological structure, but also the labels associated with the vertices.

In our study, we propose the problem of local label density search, which refers to the process of finding the top- k vertex sets with the highest query label density. We can find many practical applications of this problem. For example, in an intrusion network, one might be interested to find the regions, which are severely affected by some given intrusion types. In the domain of social network, one can ask the following query. Given a set of topics, what are the groups of authors in the network that are heavily influenced by those topics? Similarly in a map with geodesic distances, it might be interesting to locate the areas that densely contain some specific features. In this project, we provide an efficient and scalable framework to find the top- k vertex sets in a large graph with the maximum query topic density. Considering that vertex sets with very small or very large diameters carry little practical significance, a diameter constraint is further incorporated. We argue such constraint ensures various compression and pruning techniques used by our framework, while maintaining the quality of returned vertex sets.

The remaining of the report is organized as follows. We define the abstract problem formulation and preliminaries in Section 2. The algorithms are discussed in Section 3. We provide the experimental results in Section 4. Related works are given in Section 5 and we conclude in Section 6.

2 Preliminaries

A labeled graph $G = (V, E)$ has a label set L and each vertex is attached with a set of labels. The label set of a vertex u in G is $L(u)$. In this work, we focus on *bidirectional* and *unweighted* graphs. However, the proposed models and algorithms can be applied to *directed* and *weighted* graphs as well. The distance between two vertices $u, v \in V$ is denoted by $d(u, v)$.

Definition 1 (Diameter) *The diameter $D(V_1)$ of a set of vertices $V_1 \subseteq V$, is defined as the maximum pair-wise distance between any two vertices in this set.*

$$D(V_1) = \max_{u, v \in V_1} d(u, v) \quad (1)$$

A topic set consists of a set of labels from the label set L in G .

Definition 2 (Topic Density) *The density of a topic set Q over a set of vertices V_1 is denoted as $\psi(Q, V_1)$ and can be defined as the sum of the number of occurrences of each label from Q in V_1 , divided by the square of the diameter of V_1 .*

$$\psi(Q, V_1) = \frac{\sum_{l \in Q} n_l}{D^2(V_1)} \quad (2)$$

Now, we shall define the problem statement as follow.

Problem Statement 1 *Given a labeled graph G and a query topic set Q , find the top- k vertex sets from G with the maximum topic density, such that the diameter of each set lies between a lower cut-off value D_1 and an upper cut-off value D_2 .*

Note that, we set an upper and lower constraints on the diameter of the top- k result sets. This is because of the fact that vertex sets with very small or very large diameters carry little practical significance. Moreover, the lower cut-off value of diameter ensures the elimination of trivial results. On the other hand, the upper cut-off value of the diameter helps in various pruning techniques used by our framework.

3 Algorithms

In this section, we first propose a baseline approach in order to solve the current problem. Our novel framework and efficient algorithms to solve this problem will be discussed later. The baseline method is given in Algorithm 1.

Algorithm 1 Baseline Algorithm

Input: Labeled graph G , query topic set Q , lower and upper cut-off diameters D_1 and D_2 respectively, an integer k .

Output: Top- k vertex sets from G with maximum query topic density, and diameter in between D_1 and D_2 .

procedure

- 1: **for all** vertex u in G **do**
 - 2: perform Breadth First Search from u to its h -hop neighbors and form the h -hop neighborhood $N_h(u)$ of u , $D_1/2 \leq h \leq D_2$.
 - 3: **for all** $N_h(u)$ of u **do**
 - 4: count the sum of the number of each query topic occurrences in $N_h(u)$.
 - 5: calculate the diameter of the set $N_h(u)$. // can be anything between h and $2h$
 - 6: compute the query topic density of $N_h(u)$; if diameter in between D_1 and D_2 .
 - 7: **end for**
 - 8: **end for**
 - 9: output the top- K vertex sets with maximum query topic density.
-

Complexity. Let us assume that the number of vertices present in the graph is n and the average degree of each vertex is r . Hence, the number of vertex sets that we need to examine using the baseline approach

is $O(nD_2)$. On an average, each set contains $\frac{O(r^1)+O(r^2)+\dots+O(r^{D_2})}{D_2} = O(\frac{r^{D_2}}{D_2})$ vertices. Therefore, the complexity to compute the sum of occurrences of query topics in all these vertex sets (see Line 4 in Algorithm 1) is $O(nD_2 \times \frac{r^{D_2}}{D_2}) = O(nr^{D_2})$. To speed up the diameter computation of each set (see line 5 in Algorithm 1), the pairwise distance for all vertex pairs in the graph can be computed off-line, which has a complexity of $O(n^3)$ using the Floyd-Warshall algorithm. Now, for each set, we need to determine the maximum pairwise distance, which has a total complexity of $O(nD_2 \times (\frac{r^{D_2}}{D_2})^2) = O(\frac{n}{D_2}r^{2D_2})$. Thus, the baseline approach has an off-line complexity of $O(n^3)$ and an on-line complexity of $O(nr^{D_2} + \frac{n}{D_2}r^{2D_2}) = O(\frac{n}{D_2}r^{2D_2})$. Since we are dealing with very large graphs, the on-line part of the baseline approach will be very expensive for higher values of D_2 . For example, if we consider a graph containing 10^5 vertices, the average degree of each vertex being $r = 10$, and $D_2 = 10$, then $\frac{n}{D_2}r^{2D_2} = O(10^{24}) > (10^5)^{4.8}$; thus the baseline approach will have an on-line complexity which is order of $O(n^{4.8})$ in the number of vertices.

Next, we propose a novel framework to efficiently solve this problem. Our approach consists of two steps - the first step involves mapping of the vertices into a two-dimensional Euclidean space while preserving their pairwise distances. In the second step, we compute the top- k vertex sets with the maximum query topic density. While the first step can be done off-line, the second step needs to be performed on-line; however we will show that this is more efficient than the baseline approach discussed above. Also, unlike the baseline algorithm, the higher is the value of D_2 , the more pruning we can do in this on-line part of the algorithm. Later, we shall show how to use the lower cut-off diameter D_1 to compress the graph and further improve the efficiency without affecting the quality of the results significantly.

3.1 Off-Line 2-Dimensional Mapping

We first compute the pairwise distance for all pairs of vertices and apply the *Isomap* algorithm to compute the 2-dimensional embedding of the graph. Isomap uses a technique similar to Multi Dimensional Scaling (MDS) that preserves the pairwise distance among vertices. Details of the algorithm can be found in [12].

Complexity. Pairwise distance calculation for all pairs of vertices requires a time complexity of $O(n^3)$ using the Floyd-Warshall algorithm. Isomap has a time complexity of $O(n^2 \log n)$. Therefore, the complexity of the off-line part of our framework is dominated by the pairwise distance calculation.

3.2 On-Line Density Finding

In this section, we develop a technique to find the top- k vertex sets embedded in 2-dimensional Euclidean space with the highest query topic densities. Before going into details of our algorithm, we shall briefly introduce the k -Diameter problem as described in [1].

k -Diameter Problem. The k -Diameter problem can be stated as follows. Given a set of points in the Euclidean space, find the smallest diameter set enclosing k points, where the diameter of a set is defined as the maximum pairwise distance between any two points in that set. The algorithm proposed in [1] works in the following way. For each pair of points, say p and q , consider the set of points enclosed in the region, *lune*(p, q) (see the shaded region in Figure 1) and observe if either of the followings occurs.

1. If the number of points is greater than $2k$, then one half of *lune*(p, q) must contain at least k points and diameter of these k points is defined by $d(p, q)$.
2. Otherwise, define a graph $G'(p, q)$ with the points enclosed in *lune*(p, q), such that an edge exist between two points only if their pairwise distance is more than $d(p, q)$. Note that, $G'(p, q)$ will be a bipartite graph. Now, if there exist k points in $G'(p, q)$, which form an independent set, then the diameter of these set of k points is also defined by $d(p, q)$.

Therefore, the algorithm considers the vertex pairs (p, q) in increasing order of their distances $d(p, q)$. For each pair, it verifies by using the two previous methods, whether there exist k points whose diameter can be defined by $d(p, q)$. In this way, the algorithm determines the smallest diameter set containing k points. It has been proved in [] that the algorithm has a time complexity of $O(n^3 + k^{1.5}n^2 \log k)$, where n the total number of points in the space.

Based on the above idea, our on-line density finding technique is given in Algorithm 2.

Approximation Ratio. The maximum count of occurrences of query labels in the bipartite graph $G'(p, q)$ for the pair (p, q) can be $t_1 + t_2$. However, to find the exact maximum count, we need to find the maximal

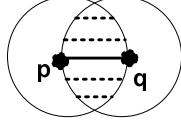


Figure 1: k-Diameter Problem

Algorithm 2 On-Line Density Finding

Input: Set of points R in 2-D, query topic set Q , lower and upper cut-off diameters D_1 and D_2 respectively, an integer k .

Output: Top- k sets with maximum query topic density, and diameter in between D_1 and D_2 .

procedure

- 1: **for all** pairs of point (p, q) in R **do**
 - 2: **if** $D_1 \leq d(p, q) \leq D_2$ **then**
 - 3: count t_1, t_2 , the total occurrences of all query topics in two half-lunes of $\text{lune}(p, q)$.
 - 4: compute the density of the points in those half-lunes as $\frac{t_1}{d^2(p, q)}$ and $\frac{t_2}{d^2(p, q)}$ respectively.
 - 5: **end if**
 - 6: **end for**
 - 7: return the top- k sets with highest query-topic density.
-

independent set of points in $G'(p, q)$, which will increase the complexity of the on-line part of our algorithm. Rather we report two sets with occurrences t_1 and t_2 . Thus, in the worst case, the maximum density found by our approximation algorithm will be half the maximum original density present in the graph.

Complexity. The total number of point pairs is $O(n^2)$ and for each pair, we need to consider all the n points. Therefore, the worst case time complexity is $O(n^3)$. However, a significant amount of pruning occurs in line 2 of Algorithm 2. Assume, there are n_1 point pairs that satisfy the diameter constraint; then the complexity reduces to $O(n^2 + n_1 \cdot n)$. Usually, $n_1 \ll n$ and the complexity is dominated by $O(n^2)$.

In the next section, we shall discuss how to further reduce the complexity without affecting the quality significantly.

3.3 Graph Compression

Although the on-line approach discussed in Algorithm 2 is very efficient compared to the baseline method in Algorithm 1, the complexity is still quadratic in number of vertices. However, the complexity would be further reduced if we can compress the graph in such a way that does not affect the effectiveness significantly. We shall use the lower cut-off diameter D_1 to intelligently compress the graph as follow. The technique is given in Algorithm 3.

Algorithm 3 Graph Compression

Input: Input graph $G = (V, E)$, lower cut-off diameter D_1 .

Output: Compressed graph $G_c = (V_c, E_c)$ with weighted edges.

procedure

- 1: $i \rightarrow 0$
 - 2: **while** there exist some vertex $u \in V$ **do**
 - 3: combine u and all its D_1 -hop neighbors $v \in V$ into supervertex $C_i \in V_c$.
 - 4: **for all** vertex $w \in C_i$ **do**
 - 5: $V \rightarrow V \setminus \{w\}$
 - 6: **end for**
 - 7: $i \rightarrow i + 1$
 - 8: **end while**
 - 9: add (C_i, C_j) to E_c if there exist $u \in C_i, v \in C_j$ such that $(u, v) \in E$.
 - 10: weight of $(C_i, C_j) = \max\{d(u, v) : u, v \in \{C_i, C_j\}\}$
-

Here we note that, the number of vertices in the compressed graph will be $O(n/r^{D_1})$. Therefore, when we apply the previously discussed 2-dimensional mapping and density finding algorithms on this

compressed graph, the efficiency is further improved. Moreover, the diameter of each dense set found from this compressed graph will still be the same because of the way we have defined edge weights (see line 10 in Algorithm 3). Also, the total number of vertices in D_2 neighborhood is very large compared to the total number of vertices in D_1 neighborhood of any vertex. Therefore, by combining the vertices till D_1 -hops using Algorithm 3 will not significantly affect the top- k dense sets found by our algorithm.

Complexity. Using the previous argument, the complexity of our compression algorithm is $O(nr^{D_1})$. Note that D_1 is typically very small and also, this compression is performed off-line.

4 Experimental Results

In order for empirical validation, we establish a comparative study between the proposed framework and the aforementioned baseline method. Performance of the algorithms is measured by both the query processing time and the average density of the top- k returned vertex sets. Our results have shown that our method is able to reduce the run time by orders of magnitude, while still providing competitive results.

4.1 Experiment Setup

Last.fm data. The experiments are done on a data set we collected from Last.fm web site. We crawled a local network consisting of 6,899 users from www.last.fm. Last.fm is a music web site where users listen to their favorite tracks and communicate with each other based on their choice of music. For each user, we crawled the most recent communications among them. These communications recommend songs. We treat them as edges. There are total 58,179 edges. For each user, we also crawled the name of 3 artists (or bands) of the most recently listened tracks by that user. There are total 6,340 artists and bands crawled. We consider a set of artists and bands as the query labels and want to determine the top- k dense groups of users in the graph containing those labels.

Machine configuration. All experiments are run on a Intel Xeon 2.5GHz and 32G RAM server with Fedora 8.

4.2 Graph Mapping

A well-known drawback of spectral methods for dimensionality reduction is their poor scalability to large data sets. Due to the large size of the graph, we face problem in terms of efficiency and memory requirement. Thus, we first compress the graph to form a smaller representation containing only 1,891 super-nodes. Each super node is a collection of nodes from the original graph which were very close to each other. The distance between two super-nodes is considered as the maximum pairwise distance between their content nodes and thus, we preserve the definition of diameter of a region. The labels are also aggregated at each super-node. Then, we perform Isomap on this smaller representation of the graph. The running time for graph compression is 39 seconds and the Isomap requires 60 seconds.

4.3 Query Processing

Performance of query processing is measured by both the query processing time and the average density of the top- k returned vertex sets. Ten subsets are randomly sampled from the label set to form our queries.

As aforementioned, in order to show the efficiency and effectiveness of the proposed method, we establish comparative study with the baseline method. Table 1 shows the average top-10 vertex set density generated by the proposed method and the baseline method, for $d = 1$. It is shown that our method is able to propose a 3-approximation. The reason why we did not achieve an exact 2-approximation is because the mapping did not precisely preserve the order of vertex pairs according to their pair-wise distances. We argue that if such order is preserved, our framework is ensured to produce a 2-approximation.

Table 1: Average top-10 Density Comparison, $d = 1$

Average Density	$Q = \{471\}$	$Q = \{3709\}$
Proposed Method	0.08037	0.02469
Baseline Method	0.24755	0.06250

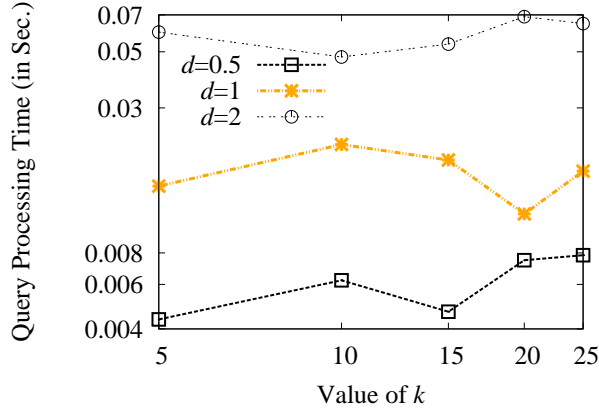


Figure 2: Query Processing Time (in Seconds) vs. k for Various Diameter Constraints

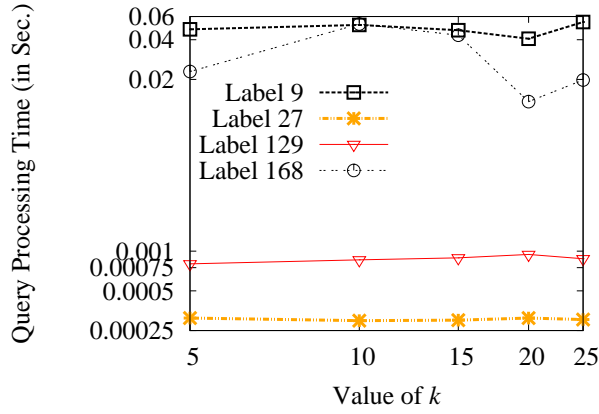


Figure 3: Query Processing Time (in Seconds) vs. k for Various Queries

Table 2 and Table 3 demonstrate the time used to find the top-10 answers for various d values, for query 471 and query 3709, respectively. As shown, the proposed method is able to reduce the runtime by orders of magnitude.

Table 2: Query Processing Time (in Seconds) Comparison for Query 471, $k=10$

Runtime	$d = 0.5$	$d = 1$	$d = 2$
Proposed Method	0.000757	0.00855	0.00293
Baseline Method	675	780	820

Table 3: Query Processing Time (in Seconds) Comparison for Query 3709, $k=10$

Runtime	$d = 0.5$	$d = 1$	$d = 2$
Proposed Method	0.000297	0.000299	0.000366
Baseline Method	550	675	760

We also observed the impacts of various parameters used in our framework, towards the query processing performances. Figure 2 visualizes the change of query processing time (in seconds) over various values of k , where diameter constraint changes from 0.5 to 2 in 2-D space, averaged over all ten queries. It is shown that the average query processing time increases with the value of d , which is expected.

Figure 3 shows the query processing time (in seconds) changing with the value of k , for 4 different query labels, where diameter constraint d is set to 1. As shown, there is no direct relation observed between

query time and k . For example, for query 129, the query time slightly increases over the increment of k , whereas for queries 9 and 168, there is a noticeable drop in query processing time for $k = 20$.

Figure 4 visualizes the average density of the top-10 sets in 2-D space, for 4 different query labels, for d values ranging from 0.5 to 2. It is observed that for three of the queries, the average density levels off for d , whereas one of the query has a lower average density for higher value of d . For the three queries, it is primarily because the top- k sets concentrate within $d \leq 0.5$ to each other, thus relaxing the values of d does not include more answers. For the one query, it is mainly because $d = 0.5$ is too small to include all the top- k answers, whereas relaxing d helps include more sets whose density is not as high as the one with $d \leq 0.5$ diameter, therefore relaxing d actually decreases the average vertex set density.

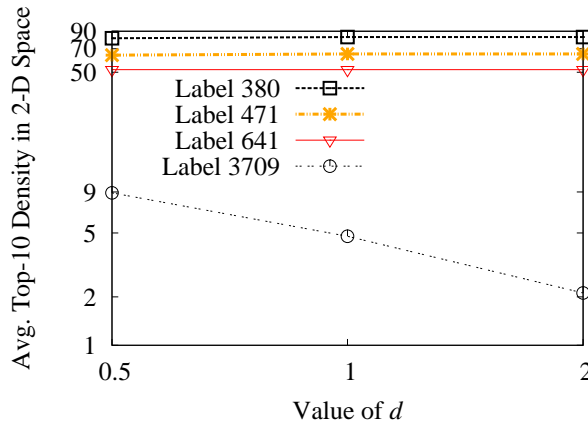


Figure 4: : Average Density of the Top-10 Sets in 2-D Space vs. Diameter Constraint

5 Related Works

In this section, we briefly review related works in the following perspectives.

Dense subgraph finding. Density subgraph search has been extensively studied in the current literature [9, 10]. However, these density search techniques mainly focus on the edge density of a particular subgraph in graphs, while ignoring the label information.

Enclosing circle problem. Finding vertices in a certain proximity was also studied in Euclidean space [1, 5, 7]. Problems take the form of finding the smallest circle, rectangle or some other geometrical shape, that encloses k points.

Dimensionality reduction. There are various dimensionality reduction techniques, that can be used to map a graph into a metric space while approximately preserving the pairwise vertex distances, i.e., ISOMAP, Maximum Variance Unfolding, Locally Linear Embedding and Laplacian Eigen Maps [2] [12]. However, most of these techniques require polynomial running time and thus, they are not very scalable for large graphs containing millions of vertices and edges.

Top- k queries in RDBMS. Top- k query processing has been studied for both RDBMS [3, 6] and middleware [4, 8, 11]. Supporting top- k queries in SQL was proposed in [3] and algorithms for top- k personalized pagerank queries were proposed in [6]. In the middleware scenario, top- k query is abstracted as getting objects with the top- k aggregate ranks from multiple data sources.

6 Conclusion

In this paper, we investigate finding the top vertex sets with the highest density in terms of query topics in a large graph. We propose the definition for topic density of a vertex set, and an ensemble framework incorporating graph dimensionality reduction to find the densest sets efficiently. In the proposed framework, a graph is first mapped to a two-dimensional Euclidean space using pairwise distance preserving mapping technique. A 2-approximation algorithm based on geometric properties is further applied in the two-dimensional space to find the densest point sets. Experiment results demonstrate the efficiency and effectiveness of the proposed framework. In the future, we intend to further apply max-flow-based technique

to find the exact vertex set within a 2-D lune. We would also like to generate more faithful mapping in terms of pairwise distances.

References

- [1] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with minimum diameter and related problems. *J. Algorithms*, 12(1):38–56, 1991.
- [2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14:585–591, 2001.
- [3] M. Carey and D. Kossmann. On saying “enough already!” in sql. *SIGMOD Rec.*, 26(2):219–230, 1997.
- [4] K.C. Chang and S.W. Hwang. Minimal probing: supporting expensive predicates for top- k queries. In *SIGMOD*, pages 346–357, 2002.
- [5] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k -enclosing circle and related problems. *Comput. Geom. Theory Appl.*, 4(3):119–136, 1994.
- [6] M. Gupta, A. Pathak, and S. Chakrabarti. Fast algorithms for topk personalized pagerank queries. In *WWW*, pages 1225–1226, 2008.
- [7] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k -enclosing circle. *Algorithmica*, 41(3):147–157, 2005.
- [8] I.F. Ilyas, G. Beskales, and M.A. Soliman. A survey of top- k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):1–58, 2008.
- [9] S. Khuller and B. Saha. On finding dense subgraphs. In *In ICALP 09*, pages 597–608, 2009.
- [10] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 639–650, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [11] A. Marian, N. Bruno, and L. Gravano. Evaluating top- k queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.
- [12] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.