Precomputing Hybrid Index Architecture for Flexible Community Search over Location-Based Social Networks

Precomputing Hybrid Index Architecture for Flexible Community Search Over Location-Based Social Networks

Ismail Alaqta[1,2], Junhu Wang[1], and Mohammad Awrangjeb[1]

[1] School of Information and Communication Technology, Griffith University, Brisbane, Australia
ismail.alaqta@@griffithuni.edu.au,
{j.wang,m.awrangjeb}@griffith.edu.au
[2] Department of Computer Science, Jazan University, Gizan, Saudi Arabia
ialaqta@jazanu.edu.sa

Abstract. Community search is defined as finding query-based communities within simple graphs. One of the most crucial community models is minimum degree subgraph in which each vertex has at least k neighbours. Due to the rapid development of location-based devices; however, simple graphs are unable to handle Location-Based Social Networks LBSN personal information such as interests and spatial locations. Hence, this paper aims to construct a Precomputed Hybrid Index Architecture (PHIA) for the sake of enhancing simple graphs to store and retrieve information of LBSN users. This method consists of two stages; the first is precomputing, and the second is index construction. Numerical testing showed that our hybrid index approach is reasonable because of its flexibility to combine different dimensions by adapting the wide used community model $k - core$.

Keywords: Community search · k-core · Indexing · Spatial graph.

1 Introduction

Recent statistics showed that over 1.75 billion of Facebook users are monthly active.[3] The graph data model is widespread to represent social networks. More specifically, the number of on-line social network users has been growing exponentially due to the spread of mobile internet access. Most of the work on social networks is concerned the community retrieval. Thus, It is essential to define what the community is. One of the most common definition is that the community is a subset of nodes which have more dense connections among themselves compared to the other subsets of the graph [Shang et al., 2017]. Community retrieval can generally be classified into community detection (CD) and community search (CS). However, our research interest focuses on the second type of community retrieval, which is the community search that aims to return dense communities involving query vertices [Sozio and Gionis, 2010].

The community search has been utilised in various applications within a different type of graphs. The general definitions of attributed graphs and spatial graphs are outlined as follow:

[3] http://www.statisticbrain.com/facebook-statistics

Attributed graph: is fundamentally a graph whose vertices or edges have text or keywords. For example, Figure 1 shows that each social network user, represented by a vertex in this graph, has different attributes to describe users' interests. Furthermore, the attributes can handle other personal information of the user, such as gender, education, relationship, etc. As a result, attributed graphs can make social network sites service-oriented, e.g. marketing. Technically, the node and edge attributes of simple graph model increase the possibilities of finding more cohesive groups due to the available information added by the attributes[Galbrun et al., 2014].

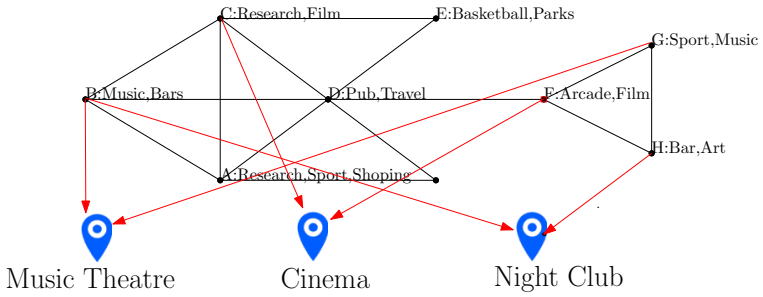

Fig. 1. Example of Spatial graph

Spatial graph: Clear examples of a spatial graph are Twitter, Facebook and Foursquare, which have become standard for their users to share locations. Due to the emergence of LBSN, lots of attention have been paid. In these LBSNs, a user has geo-locations information which can be used to indicate where the user is as shown in Figure 1. This type of network is generally defined as spatial graphs. The social graph in Figure 1 has a group of people with three different places as each person checked in a specific point.

This paper has adapted one of the most common community models named $k-$ *core* to construct a Precomputed Hybrid Index Architecture (PHIA) for the sake of enhancing simple graphs to store and retrieve information of LBSN users. It also experimented on an LBSN dataset in order to create an efficient hybrid index that can handle different types of data. The experiment demonstrated that PHIA could be more efficient in term of retrieving meaningful query-based-communities.

In the rest of this paper, we discuss the related works in Section.2 and define the problem of our communities in Section.3 and 4. Section.5 presents the Precomputed Hybrid Index Architecture (PHIA) as well as experimental results are presented, and finally, we conclude the paper in Section.6.

2 Related Works

The attributed community that has been conducted by many works of literature is characterised by vertices associated text or keyword-named attributes. More features, for instance, interpretation and personalisation can be provided effectively by using

these attributes [Fang et al., 2017]. The same authors studied attributed community search by combining a cohesive structure and keyword. Similar work was conducted by [Shang et al., 2017] , who employed an attributed community search method that improved and enhanced by [Huang et al., 2014]. Their research was aimed to construct the graph based on topology-based and attribute-based similarities âĂŤ the constructed graph in this study named TA-graph. From TA-graph structure, an index called AttrTCP-index based on TCP-index [Huang et al., 2014] was created. Hence, queries that can be found on the new index AttrTCP-index return to communities that satisfy the queries. Another leading dimension of community search is spatial, which refers to an online social network. In this dimension, internet users can share their location information, i.e. position during check-ins. There are considerable works conducted to study a spatio-social community search, as previously reviewed works assume non-spatial graphs [Sozio and Gionis, 2010, Cui et al., 2014, Huang et al., 2014, Li et al., 2015]. A recent study named spatial-aware community (SAC) was undertaken by [Fang et al., 2016]. This study has adopted the concept of minimum degree using the k-core technique.

By the reviewed literature, K-core has been found one of the most significant graph theory techniques to model communities, especially social ones. [Li et al., 2016] argued that two significant reasons that have made the $k-core$ a sufficient model to measure the level of the user group's social acquaintance. The first reason is that $k-core$ has the minimum degree constraints which consider in social science, an important measure of group cohesiveness. The second reason is that the k-core has been extensively used in the graph problems research. Indeed, in this research, we used the k-core because of its flexibility and effectiveness to capture suitable communities within LBSN.

3 Preliminaries and problem statement

We model a location-based social network (LBSN) as an undirected graph $G = (V, E)$, where each node $u \in V$ represents a user, and the edges represent relationships between users. Each user u is associated with a tuple (l_u, W_u) where l_u is the geo-location of the user, and W_u is a set of keywords that represent the interests of the users, e.g., movie, music, martial arts. The user interests are obtained by the places she visited, i.e., the check-in information, and we assume each place has a set of interests.

Given an interest $w \in W_u$ and a user u, we define

$$RS(w, u) = \frac{f_{w,u}}{\Sigma_{w_i \in W_u} f_{w_i,u}}$$

where $f_{w,u}$ is the frequency of w that occurs in the places visited by user u. Intuitively $RS(w, u)$ represents the weight of interest w in user u.

Given a connected subgraph H of G and a node u in U, we use $nbr(u, H)$ and $deg(u, H)$ to denote the the set of neighbors of u in H and the degree of u in H respectively. Let H be a connected subgraph of G. Given an interest w, we define

$$KRS(w, H) = \frac{\Sigma_{u \in H} RS(w, u)}{|H|}$$

4 Problem statement

Given a LBSN $G = (V, E)$, an interest w, a user $u \in V$, an integer $k > 0$, and a radius $r > 0$, our problem is to find a maximal subgraph H of G such that

1. $deg(v, H) \geq k$ for all $v \in H$.
2. $d(u, v) \leq r$ for all $v \in H$, where $d(u, v)$ is the geological distance between u and v.
3. $KRS(w, H)$ is maximal among all subgraphs of G that satisfy the above conditions 1 and 2.

Intuitively, the conditions 1,2,and 3 represent structural cohesiveness, location cohesiveness and interest cohesiveness respectively.

5 Precomputing and Index Architecture

In this section, the design of precomputing architecture is presented. First, An overview of the architecture is given. Then each section elaborates each component, (1) indexing, (2) Precomputed results structure and (3) Query processing.

5.1 Indexing

Here we propose our index structure to solve a community search problem over LBSN effectively and efficiently. Our approach consists of two stages, precomputing and query processing. In the precomputing stage, the attributed social graph is the input that needs to be processed offline in order to decompose and store subgraphs, which interestingly, can be employed when processing a query. Secondly, the index construction.

Precomputing Stage The precomputing stage of our proposed index structure depends on an interesting relationship between query-based community and core decomposition. Moreover, a key observation that makes our index structure relies on *k-core* is that cores are nested. Thus, in this stage, it is straightforward to decompose the graph recursively. Specifically, as illustrated in Figure 2, the *0-core*, which is basically the whole graph, is generated. Then, the $(k + 1) - cores$ are consequently generated. The idea behind precomputing the core decomposition of the input graph is to sort the $k - cores$ in order to efficiently locate the $k - core$, which includes answers to the issued queries. Moreover, this phase is crucial to extract our final hybrid index named *Attri-Spatial Core-based Index*.

In following algorithm 1, the *k-core* decomposition is precomputed and stored in an oriented-document database. In an iterative manner, our algorithm starts with initialising $k = 0$ and an empty collection to host the decomposed *k-cores*. Each *k-core* is stored in a document whose key holds k and the value of k represents a *k-core* subgaph. Each subgraph has connected components. Each component of vertices C_i^k

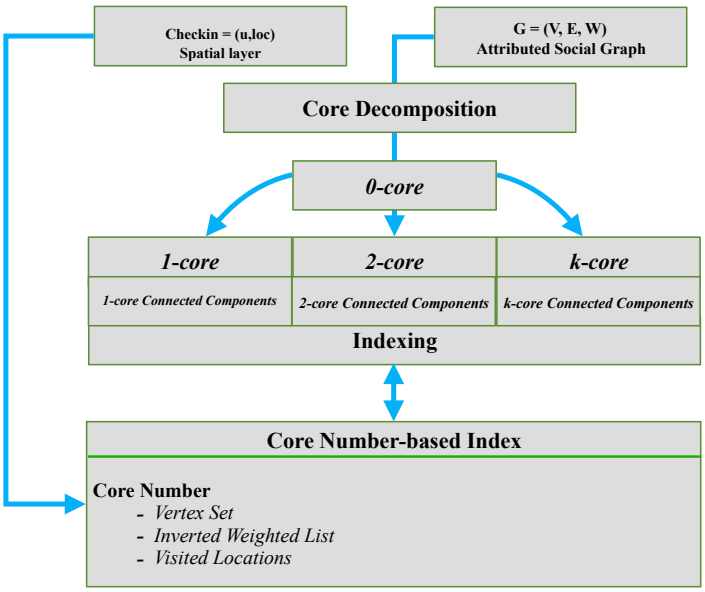

Fig. 2. Core-based Index

is represented by an array as demonstrated by (Line 3).

Data: $G = (V, E, W)$
Result: *Cores Collection*
begin
1 Initialization: $k \leftarrow 0$, $KcoreCollection \leftarrow \phi$
2 **while** $k_{max} \geq k$ **do**
3 KcoreCollection.insert{"k" : k : $[H(V)] \leftarrow CoreComp\,(k, G)$}
4 **for** $i = 0$ **to** $|H|$ **do**
5 **for** $j = i{+}1$ **to** $|H|$ **do**
6 **if** $(H[i], H[j]) \in E$ **then**
7 KcoreCollection.Update
($\{$"k" : $k\}$, $\{$"H_E" : $H_E.add((H[i], H[j])\}$)
end
end
end
8 $k++$
end
9 **return** KcoreCollection
end

Algorithm 1: k-core Precomputation

In following algorithm 2, the connected components are precomputed for each $k-core$ subgraph.

Data: *Cores Collection*
Result: List of Connected Components for each *k-Core*
Initialization: $CCList[\,] \leftarrow \phi, Subgraph\, H = (V_H, E_H)$ **begin**
  while K **do**
    $k \leftarrow K.pop()$
    for $cc \in H.CC$ **do**
      for $u \in cc$ **do**
        CCList.append(u)
      end
      $ColCores.update(\{"k" : k\}, \{"CC\_Users" : CCList\})$
      $CCList.clear()$
      $H.clear()$
    end
    return Updated Cores Collection
  end
end

Algorithm 2: K-Core Connected Components Precomputation

Index Construction After the core decomposition occurred $C = \{C_1, \cdots, C_k\}$, we utilise the core number as an index for those users who have the same core order in the graph G which is already decomposed into $k-cores$. In other words, the core number for each user is kept as $C(u) = k$ as the following:

- For all users $u \in V$, the core index $C(u)$ of u, that is as the index of the highest order core containing u: $C(u) = max\{k \in [0 \cdots k] \,|u \in C_k\}$.

Underneath each core index as illustrated in Figure 3, we encapsulate the other core-based index components to make our hybrid index more effective by employing the inverted weighted index for the graph attributes. To summarise, each index, which is referred to as a core number, contains three elements:

- *VertexSet* : a set of graph vertices that represent users;
- *InvertedWeightedList* : a list of $\langle key, value \rangle$ pairs, where the *key* is a keyword that is contained by *VertexSet* and the *value* is an array of weights. Each element represents a weight, given by Eq 3, which corresponds to each user in *VertexSet*.
- *VisitedLocations* : a set of coordinates $\langle x, y \rangle$ of places that have been visited by u. More effectively, all spatial objects, that contain the interest w, are mapped to the interest w.

The main idea of the structure in Figure 3 is to enhance query processing by taking the advantages of the inverted index data structure. After the precomputing process, Algorithm 3 is proposed to construct our index named Attri-Spatial Core-based Index. To tackle this proposed algorithm, the technique of the inverted index has been adopted inside the MongoDB. Specifically, users V, interests W and places P are grouped into a single collection called $CoreNum\_Col$. In details, for each

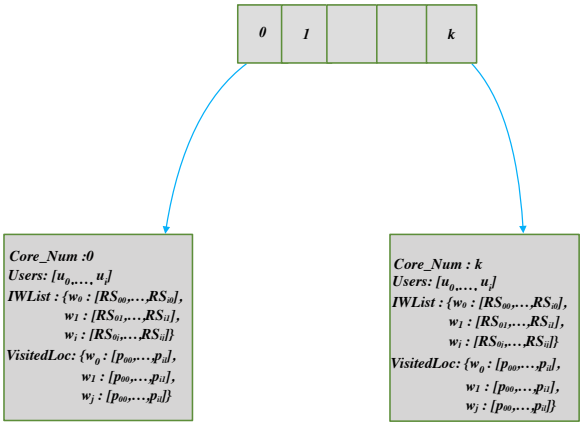

Fig. 3. Attri-Spatial Core-based Index

interest $w_j \in W$ we build an inverted weighted list whose elements represent weights RS_{ij} that correspond to $Users$. Besides, the places P, which has been visited by users, are also inverted by each interest w_j.

Data: $CoresCol\ C$,
$Social\ Attributed\ Weighted\ Graph\ Col$,$CK = \{\langle u_i, p_k \rangle\}$
Result: *Core-based Index*
begin
  Initialization: $CoreNum\_Col \leftarrow \phi, Core\_num \leftarrow 0$
  foreach $w_j \in W$ **do**
    foreach $u_i \in V$ **do**
      if $|CoreNum\_Col| < |KcoreCollection.find(u_i)|$ **then**
        $CoreNum\_Col.insert($
        $"Core\_Num" : |KcoreCollection.find(u_i)|,$
        $"Users" : VertexSet[\ ].append(u_i),$
        $"IWList"\{: w_j : weight[\ ].append(RS_{ij})\},$
        $"VisitedLocations" : \{w_j : places[\ ].append(p_{k_u})\})$
      end
      else
        $CoreNum\_Col.update(\ "Users" : VertexSet[\ ].append(u_i),$
        $"IWList" : w_j : \{weight[\ ].append(RS_{ij})\},$
        $"VisitedLocations" : \{w_j : places[\ ].append(p_{k_u})\})$
      end
    end
  end
  return Core-based Index
end

Algorithm 3: Attri-Spatial Core-based Index

5.2 Dataset and Numerical Testing

Weeplaces is a dataset [Liu et al., 2014] that has been collected from a website named Weeplaces, in which users' check-in activities can be visualised in LBSN. It has been integrated using the API of other well-known LBSNs, e.g. Facebook Places, Foursquare, and Gowalla. The dataset contains more than 7.5 million check-ins by 15,799 users across 971,309 geolocations.

col_cores > k

Core	k	H_E	CC_Users
[163 elements]	15	[2026 elements]	[1 elements]
[119 elements]	14	[472 elements]	[1 elements]
[188 elements]	13	[707 elements]	[3 elements]
[206 elements]	12	[654 elements]	[6 elements]
[202 elements]	11	[337 elements]	[15 elements]
[415 elements]	10	[1329 elements]	[9 elements]
[327 elements]	9	[489 elements]	[23 elements]
[626 elements]	8	[1173 elements]	[33 elements]
[803 elements]	7	[1356 elements]	[61 elements]
[912 elements]	6	[1090 elements]	[98 elements]
[1252 elements]	5	[1207 elements]	[171 elements]
[1613 elements]	4	[1281 elements]	[229 elements]
[2367 elements]	3	[1385 elements]	[421 elements]
[3251 elements]	2	[1359 elements]	[597 elements]
[3578 elements]	1	[481 elements]	[387 elements]

Fig. 4. The Pre-computation Collection Structure

In Figure 4, algorithms 1,2 have been implemented and then stored inside a MongoDB collection called Col_cores. This collection has four fields named Core, k, H_E, and CC_Users. For instance, at *k=1*, there are 3578 users stored in field Core and 387 connected components of users stored in CC_Users. These components belong to the subgraph edges H_E.

Core_num	1	Int32
Core_Users	[3578 elements]	Array
IWLL	[474 elements]	Array
0	{ 2 fields }	Object
kw	Hot Dogs	String
Score	[3 elements]	Array
0	[6 elements]	Array
0	daniel-poulsen6	String
1	doug-hanna	String
2	john-moccia	String
3	justin-street	String
4	joe-landor	String
5	marc-ribo	String
1	[6 elements]	Array
0	0.08038326515710895	Double
1	0.057496363272098755	Double
2	0.017920944396498314	Double
3	0.11499272654419751	Double
4	0.0275982543706074	Double
5	0.029359845075114258	Double
2	[8 elements]	Array
0	[2 elements]	Array
1	[2 elements]	Array
2	[2 elements]	Array
3	[2 elements]	Array
4	[2 elements]	Array
5	[2 elements]	Array
6	[2 elements]	Array
7	[2 elements]	Array

$C(u) = max\{k \in [0 \cdots k] \mid u \in Ck\}$

a list of ⟨key, value⟩ pairs, where the key is a keyword kw that is contained by VertexSet and the value is an array of weights

a set of coordinates ⟨x, y⟩ of places that have been visited by u w.r.t the keyword kw

Fig. 5. Attri-Spatial Core-based Index inside MongoDB

In Figure 5, algorithm 3 has been implemented in order to index our LBSN graph G by employing the core number as illustrated in Figure 3.

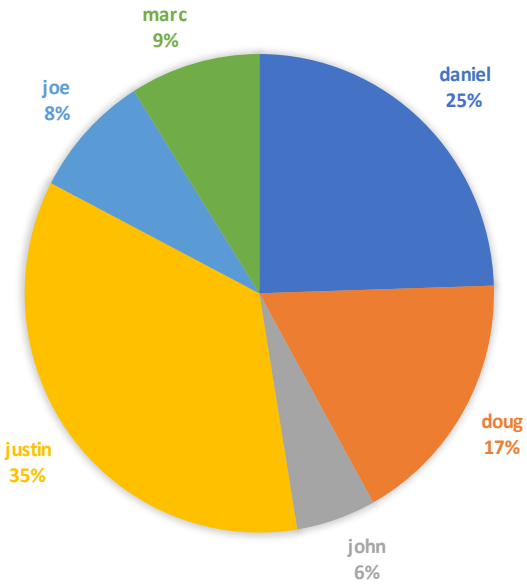

Fig. 6. Weights of users' Interest (HOT DOGS)

The pie chart in Figure 6 shows the result of weights for interest (HOT DOGS) in which users can be retrieved at core number $k = 1$ as illustrated in Figure 5. From the pie chart it is clear that user named *justin* is highly interested in *HOT DOGS* compared with other users in the core number $k = 1$.

6 Conclusion and Future work

This paper has discussed the possibilities of spatio-attributed community search to enrich the simple graphs model. It can be done by constructing a precomputed Hybrid Index Architecture method (PHIA) which could assist for the sake of enhancing simple graphs to retrieve and store users' information over LBSN. The proposed method has consisted of two phases; the first is precomputing, where the second is index construction. In both stages, the attributed social graph is the input that needs to be processed offline to store and compose and then the hybrid index can be employed when processing a query. Together, Numerical testing showed that our hybrid index approach is promising due to its flexibility to combine different dimensions by adapting the wide used community model $k - core$. Future research will focus on conducting extensive experiments on an LBSN dataset in order to enhance the efficiency of the developed hybrid index that can handle different types of data in order to demonstrate that PHIA can be more efficient in term of retrieving meaningful query-based-communities. Moreover, the query processing, which is involved of two elements, a retrieval and ranking function algorithm, will be further designed; hence, the PHIA will be efficiently utilised.

Bibliography

Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. Local search of communities in large graphs. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, 1:991–1002, 2014. ISSN 07308078.

Yixiang Fang, Reynold Cheng, Wenbin Tang, Silviu Maniu, and Xuan Yang. Scalable algorithms for nearest-neighbor joins on big trajectory data. In *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, pages 1528–1529, 2016. ISBN 9781509020195.

Yixiang Fang, Reynold Cheng, Yankai Chen, Siqiang Luo, and Jiafeng Hu. Effective and efficient attributed community search. *VLDB Journal*, 26(6):803–828, 2017. ISSN 0949877X.

Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. Overlapping community detection in labeled graphs. *Data Mining and Knowledge Discovery*, 28(5-6):1586–1610, 2014. ISSN 13845810.

Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying k-truss community in large and dynamic graphs. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, 2:1311–1322, 2014. ISSN 07308078.

Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. Influential community search in large networks. *Proceedings of the VLDB Endowment*, 8(5):509–520, 2015. ISSN 2150-8097.

Yafei Li, Rui Chen, Jianliang Xu, Qiao Huang, Haibo Hu, and Byron Choi. Geo-Social K-Cover Group queries for collaborative spatial computing. *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, 27(10):1510–1511, 2016. ISSN 10414347.

Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 739–748, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2598-1. https://doi.org/10.1145/2661829.2662002.

Jingwen Shang, Chaokun Wang, Changping Wang, Gaoyang Guo, and Jun Qian. An attribute-based community search method with graph refining. *The Journal of Supercomputing*, pages 1–28, 2017. ISSN 1573-0484.

M Sozio and a Gionis. The community-search problem and how to plan a successful cocktail party. *Proceedings Of The Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*, pages 939–948, 2010.