Shortest Path Cache

June 20, 2012

Jeppe Rishede Thomsen
Department of Computing
Hong Kong Polytechnic University
Kowloon, Hong Kong
csjrthomsen@comp.polyu.edu.hk

Man Lung Yiu
Department of Computing
Hong Kong Polytechnic University
Kowloon, Hong Kong
csmlyiu@comp.polyu.edu.hk

ABSTRACT

abstract...

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc luctus mattis pretium. Mauris egestas risus non lectus dapibus condimentum. Vestibulum at lacus sit amet elit rutrum venenatis sit amet sed turpis. Nam ullamcorper, urna aliquet venenatis tempor, dolor enim egestas neque, ac feugiat felis neque et libero. Vivamus in augue urna. Integer euismod interdum felis, a porta quam posuere at. Duis molestie leo ullamcorper mauris sodales porta. Praesent cursus, tortor a iaculis blandit, mauris tellus suscipit nibh, in condimentum elit eros nec mauris. In hac habitasse platea dictumst.

Phasellus eget tortor eget diam dignissim pretium nec ut nibh. Morbi bibendum nibh et eros tristique rhoncus. Suspendisse dolor erat, lacinia ut ullamcorper vitae, dictum nec dolor. Etiam egestas tempus fermentum. Donec commodo feugiat lorem. Donec facilisis pharetra leo, sit amet cursus turpis convallis eu. Sed fermentum sapien quam, quis porttitor est.

1. INTRODUCTION

$$\chi_{s,t} = |\{P_{s,t}\}| : \ s \in QR_{s,R}, t \in \mathcal{O}, QR_{q,R} \in \mathcal{QL}_{\mathcal{R}} \quad (1)$$

$$\chi_{s,t} = |\{P_{s,t}\}| : \{s \in QR_{s,R}, \{t \in \mathcal{O} | \mathfrak{d}_{s,t} \leq R\}, QR_{s,R} \in \mathcal{QL}_{\mathcal{R}}\} \quad (2)$$

Definition Range Search
A range search query, denoted by $Q_{q,R}$ consist of a source vertex q and a range $\mathbf{R}$. The result of $Q_{q,R}$, denoted $\mathcal{O}_{q,R}$, is a set of objects $o_i \in \mathcal{O}$ with SPdistance $d_{s,t} \leq R$ on graph $G(\mathbf{V}, \mathbf{E})$.

Definition Range Search Query Log ($\mathcal{QL}_{\mathcal{R}}$)
A range search query log $\mathcal{QL}_{\mathcal{R}}$ is a collection of timestamped queries that have been issued by users in the past. A query is on the form (q, R), where q is a query point and R is a range. The full form of the log, $\mathcal{QL}_{\mathcal{R}}$, is then: $\{(q_0, R_0), \ldots, (q_i, R_i)\}$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 38th International Conference on Very Large Data Bases, August 27th - 31st 2012, Istanbul, Turkey.
Proceedings of the VLDB Endowment, Vol. 5, No. 0
Copyright 2011 VLDB Endowment 2150-8097/11/07... $ 10.00.

Symbol	Meaning
$QR_{q,R}$	Range query from q to all $o_i : d_{q,o_i} < R$)
$\chi_{s,t}$	The frequency of a SP
$P_{s,t}$	A SP from s to t
$d_{s,t}$	The SP distance of a path $P_{s,t}$
$\mathfrak{d}_{s,t}$	$\|t - s\|_2$, the euclidean distance
$\mathcal{O}$	The set of $POI \in \mathbf{V}$, the set of vertices
o_i	Element i in $\mathcal{O}$
$\mathcal{O}_{q,R}$	The result set of $o_i \in Q_{q,R}$
$dist_{\mathcal{O}}$	Table of distances between vertexes $v_s, v_t \in P_{s,t} : P_{s,t} \in SP_q$
SP_q	A set of SP from q to all o_i
$\mathcal{QL}_{\mathcal{R}}$	Query log of range search queries
Ψ	The Cache
$G(\mathbf{V}, \mathbf{E})$	Graph representation of the Map

Table 1: Table of Symbols

Using a query log, $\mathcal{QL}_{\mathcal{R}} : \{(q_0, R_0), \ldots, (q_i, R_i)\}$, we first need to expand each query into its SPresult set containing SPs from q to each POI reachable within R distance on a road network. To find the result set of a query we use one of two algorithms: **NAIVE** or **FAIR**ns.

The **NAIVE** algorithm will not only find the result set of SPfor each query $\in \mathcal{QL}_{\mathcal{R}}$, but also the SPfrom q to all other possible POIns, before pruning and returning the result set of POIsns. **NAIVE** does so by first finding all paths from q to any POI in the cache. Afterwards it will issue SPqueries from q to the remaining POIs not already found via the cache. Once all paths are found **NAIVE** will prune the set of paths such that only the paths with distance < R remain. This is the result so **NAIVE** returns it to the user.

The idea behind the **FAIR** algorithm (see alg. 2) is to prune POIsns, which can not belong to the result set of a query, before doing range search. The **FAIR** algorithm works by, for a query (q,R), first finding the set *tmpPOIns*, of POIs which lay within euclidean distance R of the query point q. **FAIR** then invokes the **NAIVE** algorithm using *tmpPOIns* as the set of POIns. The pruning method of **FAIR** is correct as the euclidean distance R is also the longest SPns, with length R, possible. It is however important to note that this pruning technique only work if the weights of edges in a road network are distances.

Equation 1 and 2 define the frequency of a SPin Ψ, using **NAIVE** or **FAIR** respectively.

An Example of how **FAIR** might work is shown in fig. 1. In it we have 2 queries: q_1 and q_2. Each query have a range $R = 13$ associated with it, which is used to form the pruning areas (circles) of

Algorithm 1: Naive Algorithm

input :
(q, R): A Range query
$\mathcal{O}$: A set of POI
output:
A set POI $\in \mathcal{O}$

Naive$((q, R), \mathcal{O})$
  foreach $o_i \in \mathcal{O}$ **do**
    if $P_{q,o_i} \in \Psi$ **then**
      $result \leftarrow P_{q,o_i}$
    else
      Calculate P_{q,o_i} *//SPfrom q to* o_i
      if $d_{q,o_i} \leq R$ **then**
        $result \leftarrow o_i$
  return *result*

Algorithm 2: Fair Algorithm

input :
(q, R): A Range query
$\mathcal{O}$: A set of POI
output:
A set POI $\in \mathcal{O}$

Fair$((q, R), \mathcal{O})$
  foreach $o_i \in \mathcal{O} : \mathfrak{d}_{q,o_i} \leq R$ **do**
    $candidate_{\mathcal{O}} \leftarrow o_i$
  result $\leftarrow candidate_{\mathcal{O}}$)
  return *result*

the queries. The pruning areas are the shown by the 2 circles, each covering a query. The white circles are all vertices. $v_1, \ldots, v_{11}$ are the regular vertices. $o_1, \ldots, o_{10}$ represent the POIsns, and q_1, q_2 are the query points. Vertices ($\mathbf{V}$), and connecting lines,edges ($\mathbf{E}$), model a road network, $G(\mathbf{V}, \mathbf{E})$.

In our problem the range R is constrained on a road network, meaning that after pruning, when the vertexes more than R euclidean distance away are no longer considered, the remaining vertexes may still be further than R distance in the road network. We can see an example of this with q_2, where only one out of the 3 POI in the gray region is actually within R SPdistance (o_9, the reader should visually easily be able to confirm this using the edge weights in fig 1).

When we consider what to put into the cache, then we will include SPs for all POI within the pruning area, regardless of whether the POIs are contained in the result set or not. The reasoning behind this, is that if we only add the exact result, then we will not be able to determine if the entire result is in the cache or not, except in the special case that all vertexes in the pruning region is part of the result, as with query q_1.

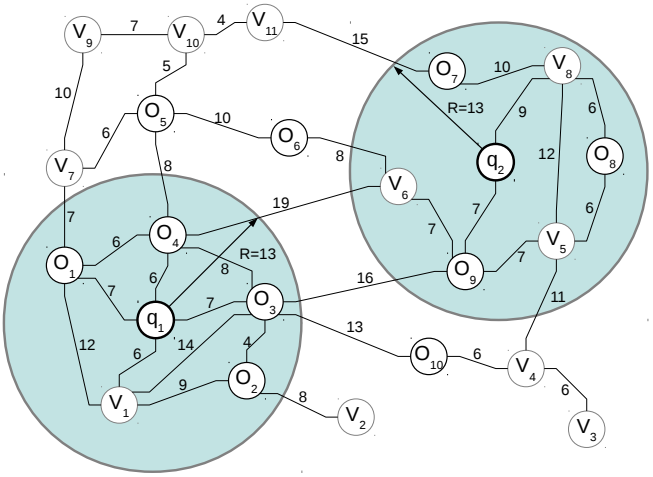

Figure 1: Query points q_1, q_2, q_3, with region defined by R. White points are POIns

2. RELATED WORK REFERENCE

Reference support for related work section.

2.0.1 *On effective presentation of graph patterns: a structural representative approach*

They develop an approach that combine two focuses when mining patterns in graphs. 1. they introduce a method to relax the tightness of the pattern subgraph pattern matching, so they can have high support for subgraphs which are very similar, but not exact. 2. as many mining approaches return allot (often very similar) patterns, they propose a method to collapse similar patterns so the user is presented with something that is easier to get an overview of and gain an understanding of the data. [2]

2.1 Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments

They develop two cache replacement and invalidation techniques for mobile clients communicating with a LBS. They argue that in the setting of spatial data and LBS then it is important to consider more than just the access time when doing cache replacement. They look at the spatial area where an object in the cache is valid as well as the direction the user is moving. They do this besides calculating the probability that this object will be accessed again.

Assumes all POI objects are fixed size and no updates will be made. [11]

2.2 Nearest-Neighbor Caching for Content-Match Applications

[8]

2.3 Caching Content-based Queries for Robust and Efficient Image Retrieval

They study how to do caching with Content-based Image Retrieval, and they support range and kNN queries. They focus on how to do caching when many of the queries are similar, but not the same (e.g. picture cropped or color changes) without polluting the cache. Their approach works in metric space and they develop an approximate method to check if the result can be satisfied by the cache. They archive good results, getting few direct cache hits, but still satisfying many queries from similar queries in the cache.

[4]

2.4 Caching Complementary Space for Location-Based Services

They develop the notion of Complementary Space(CS) to help better use a cache on a mobile client. CS is different levels for representing the objects on a map within MBRs. At the lowest level they just show the object, and as the levels go up they include more and more objects within MBRs, looking at the trade of in communication up/down link from a mobile client. They always have the entire world represented within the clients cache, at different levels, and offer no solution to how they will handle server updates to the map.

This is very similar to [5], although the approach does not formally depended on an R-tree, they still use one and offer no viable alternative, which lessens the difference even more. Their results are better than their competitors, including [5], though it seems that they stop their graphs just before [5] beats them.

2.5 Proactive Caching for Spatial Queries in Mobile Environments

They develop an approach which uses the index of an R-tree to add context to a cache of spatial object on a mobile client. They develop several communication and space saving techniques by representing less important parts of the R-tree in more compact ways, or just not storing the lower nodes/leaves of the tree. They also formally prove the asymptotic bounds of their algorithms.

[5]

2.6 Aggregate Aware Caching for Multi-dimensional Queries

They develop a method to re-use items in a cache for a data werehouse. They take advantage of the levels of aggregation which exist in OLAP query results and come up with a way where they can get aggregated results using full or partial more detailed data from the cache. The use most of the paper on showing and proving that their algorithms have a good running time and admit them selfes that the approach is not very mature, though they still manage to get resonable results. [3]

2.7 DynaMat: A Dynamic View Management System for Data Warehouses

abstract: Pre-computation and materialization of views with aggregate functions is a common technique in Data Warehouses. Due to the complex structure of the warehouse and the different profiles of the users who submit queries, there is need for tools that will automate the selection and management of the materialized data. In this paper we present DynaMat, a system that dynamically materializes information at multiple levels of granularity in order to match the demand (workload) but also takes into account the maintenance restrictions for the warehouse, such as down time to update the views and space availability. DynaMat unifies the view selection and the view maintenance problems under a single framework using a novel ï£¡ï£¡goodnessï£¡ï£¡ measure for the materialized views. DynaMat constantly monitors incoming queri and materializes the best set of views subject to the space constraints. During updates, DynaMat reconciles the current materialized view selection and refreshes the most beneficial subset of it within a given maintenance window. We compare DynaMat against a system that is given all queries in advance and the pre-computed optimal static view selection. The comparison is made based on a new metric, the Detailed Cost Savings Ratio introduced for quantifying the benefits of view materialization against incoming queries. These experiments show that DynaMat's dynamic view selection outperforms the optimal static view selection and thus, any sub-optimal static algorithm that has appeared in the literature. [7]

2.8 Cache-Oblivious Data Structures and Algorithms for Undirected Breadth-First Search and Shortest Paths

[1]

2.9 Cached Shortest-Path Tree: An Approach to Reduce the Influence of Intra-Domain Routing Instability

They assume a network setting and try to reduce the time and computational load it takes when network topology changes, as well as prevent any links from being unreachable if the topology changes often. The propose a cache with shortest-path trees, arguing that even if the topology changes often, then it is mostly between the same configurations (e.g. a computer/router is turned off/on) meaning that a cache with the most common seen configurations will be able to drastically reduce the amount of computation needed to recalculate routing tables.

[10]

2.10 On Designing a Shortest-Path-Based Cache Replacement in a Transcoding Proxy

[6]

2.11 Optimizing Graph Algorithms for Improved Cache Performance

[9]

3. REFERENCES

[1] G. Brodal, R. Fagerberg, U. Meyer, and N. Zeh. Cache-oblivious data structures and algorithms for undirected breadth-first search and shortest paths. In T. Hagerup and J. Katajainen, editors, *Algorithm Theory - SWAT 2004*, volume 3111 of *Lecture Notes in Computer Science*, pages 480–492. Springer Berlin / Heidelberg, 2004.

[2] C. Chen, C. X. Lin, X. Yan, and J. Han. On effective presentation of graph patterns: a structural representative approach. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 299–308, New York, NY, USA, 2008. ACM.

[3] P. Deshpande and J. F. Naughton. Aggregate aware caching for multi-dimensional queries. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '00, pages 167–182, London, UK, 2000. Springer-Verlag.

[4] F. Falchi, C. Lucchese, S. Orlando, R. Perego, and F. Rabitti. Caching content-based queries for robust and efficient image retrieval. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 780–790, New York, NY, USA, 2009. ACM.

[5] H. Hu, J. Xu, W. S. Wong, B. Zheng, D. L. Lee, and W.-C. Lee. Proactive caching for spatial queries in mobile environments. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 403–414, Washington, DC, USA, 2005. IEEE Computer Society.

[6] H.-P. Hung and M.-S. Chen. On designing a shortest-path-based cache replacement in a transcoding proxy. *Multimedia Systems*, 15:49–62, 2009.

[7] Y. Kotidis and N. Roussopoulos. Dynamat: A dynamic view management system for data warehouses. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 371–382. ACM Press, 1999.

[8] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii. Nearest-neighbor caching for content-match applications. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 441–450, New York, NY, USA, 2009. ACM.

[9] J.-S. Park, M. Penner, and V. K. Prasanna. Optimizing graph algorithms for improved cache performance. *IEEE Trans. Parallel Distrib. Syst.*, 15(9):769–782, 2004.

[10] S. ZHANG, K. IIDA, and S. YAMAGUCHI. Cached shortest-path tree : An approach to reduce the influence of intra-domain routing instability. *IEICE transactions on communications*, 86(12):3590–3599, 2003-12-01.

[11] B. Zheng, J. Xu, and D. L. Lee. Cache invalidation and replacement strategies for location-dependent data in mobile environments. *IEEE Trans. Comput.*, 51(10):1141–1153, 2002.