L.Sundarrajan *et al*, International Journal of Computer Science and Mobile Applications,
Vol.6 Issue. 3, March- 2018, pg. 73-77 **ISSN: 2321-8363**
Impact Factor: 5.515

A G-TREE BASED EFFICIENT AND SCALABLE INDEX FOR SPATIAL SEARCH IN LOCATION BASED SERVICE

L.Sundarrajan[1], K.Gopinath[2], V.Praveen[3], Dr. P.Gomathi[4], A.Jenifer[5]
[1]Assistant Professor, [2]UG Scholar, [3]Assistant Professor, [4]Dean/Academics, [5]Assistant Professor
[1] mlssundar@gmail.com
[2] kgopinath579@gmail.com
[3] praveen.sfj@gmail.com
[4] gopinathk882@gmail.com
[5] jenicse02@gmail.com

Abstract: ***In the recent decades, Those have witnessed the rapidly growing popularity of location-based systems. Three types of location-based queries on road networks, single-pair shortest path, query, k nearest neighbor (KNN) query, and keyword-based KNN query, are widely used in location-based systems. Inspired by R-tree, That propose a height-balanced and scalable index, namely G-tree, to efficiently support these queries. The space complexity of G-tree is O (|V|log|V|) where |V| is the number of vertices in the road network. Unlike previous works support these queries separately, G-tree supports all these queries within one framework. The basis for this framework is an assembly-based method to calculate the short-path distances between two vertices. Based on the assembly-based method, efficient search algorithms to answer KNN queries and keyword-based KNN queries are developed. Experiment results show G-tree's theoretical and practical superiority over existing methods.***
Keywords: G-tree, KNN Queries

1. Introduction

A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to the object based on different selection criteria. The importance of spatial databases is reflected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points on a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in specific contexts. For instance, in a geographic information system, range search can be deployed to find all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address. Today, the widespread use of search engines has made it realistic to write spatial queries in a brand new way. Conventionally, queries focus on objects' geometric properties only, such as whether a point is in a rectangle, or how close two points are from each other. That have seen some modern applications That call for the ability to select objects based on both of their geometric coordinates and their associated texts.

It accesses method successfully incorporates point coordinates into a conventional inverted index with small extra space, owing to a delicate compact storage scheme. Meanwhile, an SI-index preserves the spatial locality of data points, and comes with an R-tree built on every inverted list at little space overhead. As a result, it offers two competing ways for query processing. That can (sequentially) merge multiple lists very much like merging traditional inverted lists by IDs. Alternatively, That can also leverage the Rtrees to browse the points of all relevant lists in ascending order of their distances to the query point. As demonstrated by experiments,

the SI-index significantly outperforms the IR 2-tree in query efficiency, often by a factor of orders of magnitude. That can use a single source shortest-path algorithm, e.g. Dijkstra algorithm, to compute the graph distance. It starts from each border/vertex within one G-tree node, expands the edges until if all borders of such node have been reached. Graph partitioning is an important step in G-tree construction. The optimal one should not only generate approximately equal-sized subgraphs, but also minimize the number of borders. However, it has been proven that the optimal graph partitioning is NP-Hard [5].

2. Literature Survey

1. With the proliferation of Geo-positioning and go-tagging techniques, spatio-textual objects That poses both a geographical location and a textual description are gaining in prevalence, and spatial keyword queries That exploit both location and textual description are gaining in prominence. However, the queries studied so far generally focus on finding individual objects That each satisfy a query rather than finding groups of objects where the objects in a group together satisfy a query.
2. On paper, That study the problem of can search for road networks. Given a query location and a set of candidate objects in a road network, the KNN search finds the k nearest objects to the query location. To address a problem, That propose a balanced search tree index, called a G - tree. The G-tree of a road network is constructed by recursively partitioning the road network into sub-networks and each G-tree node corresponds to a sub-network. Inspired by classical can search on metric space, That introduces a best-first search algorithm on road networks, and propose an elaborately-designed assembly-based method to efficiently compute the minimum distance from a G-tree node to the query location. G-tree only takes O (|V|log|V|) space, where |V| is the number of vertices in a network, and thus can easily scale up to large road networks with more than 20 million vertices. Experimental results on eight real-world datasets show That our method significantly outperforms state-of-the-art methods, even by 2-3 orders of magnitude.
3. Web users and content are increasingly being geopositioned, and increased focus is being given to serving local content in response to that query. It development calls for spatial keyword queries That take into account both the locations and textual descriptions of content. That study the efficient, joint processing of multiple top-k spatial keyword queries. Such joint processing is attractive during high query loads and also occurs when multiple queries are used to obfuscate a user's true query. That propose a novel algorithm and index structure for the joint processing of top-k spatial keyword queries. Empirical studies show That the proposed solution is efficient on real data sets. That also offer analytical studies on synthetic data sets to demonstrate the efficiency of the proposed solution.
4. It is quite common for networks emerging nowadays to have labels or textual contents of the nodes. On such networks, That study the problem of top-k nearest keyword (k-NK) search. In a network G modeled as an undirected graph, each node is attached with zero or more keywords, and each edge is assigned to a Thatight measuring its length. Given a query node q in G and a keyword λ, a k-NK query seeks k nodes, which contain λ and are nearest to q. K-NK is not only useful as a stand-alone query, but also as a building block for tackling complex graph pattern matching problems. The key to an accurate k-NK result is a precise shortest distance estimation in a graph. Based on the latest distance oracle technique, That builds a shortest path tree for a distance oracle and use the tree distance as a more accurate estimation. With such representation, the original k-NK query on a graph can be reduced to answering the query on a set of trees and then assembling the results obtained from the trees. That propose two efficient algorithms to report the exact k-NK result in a tree. One is query time optimized for a scenario when a small number of result nodes are of interest to users. The other handles k-NK queries for an arbitrarily large k efficiently. In obtaining a k-NK result on a graph from That on trees, a global storage technique is proposed to further reduce the index size and the query time. Extensive experimental results conform with our theoretical findings, and demonstrate the effectiveness and efficiency of our k-NK algorithms on large real graphs.
5. In the maximizing range sum (MaxRS) problem, given (i) a set P of 2D points, each of which is associated with a positive weight, and (ii) a rectangle for specific extents, we need to decide where to place are in order to maximize the covered weight of r - that is, the total weight of the data points covered by r. Algorithms solving the problem exactly entail expensive CPU or I/O cost. In practice, exact answers are often not compulsory in a MaxRS application, where slight imprecision can often be comfortably tolerated, provided that approximate answer can be computed considerably faster. Motivated by it, the present paper studies the (1 - ε) -approximate MaxRS problem, which admits the same inputs as Mars, but aims instead to return a rectangle whose covered

weight is at least (1-ε) m*, where m* is the optimal covered weight, and ε can be an arbitrarily small constant between 0 and 1. We present fast algorithms that settle it problem with strong theoretical guarantees.

6. It paper investigates the MaxRS problem in spatial databases. Given a set O of weighted points and a rectangular region are of a given size, the goal of the MaxRS problem is to find a location of r such that the sum of the weights of all the points covered by r is maximized. Its problem is useful in many location-based applications such as finding the best place for a new franchise store with a limited delivery range and finding the most attractive place for a tourist with a limited reachable range. However, the problem has been studied mainly in theory, particularly, in computational geometry. The existing algorithms from the computational geometry community are in-memory algorithms which do not guarantee the scalability. In its paper, That propose a scalable external-memory algorithm (ExactMaxRS) for the MaxRS problem, which is optimal in terms of the I/O complexity.

3. Existing System

Spatial queries with keywords have not been extensively explored. In the past years, the community has sparked enthusiasm in studying keyword search in relational databases. There are a large number of studies on can search for road networks [3, 8, 9, 14, 15, 16, 20, 23]. However, existing methods still cannot support very large road networks (e.g. The whole USA road network). The main limitation of these approaches is either high memory consumption or heavy search over head g, which has very poor scalability and efficiency of large road networks. For example, for the whole USA dataset (24M vertices), That estimate That ROAD needs over 105 days for pre-processing, and SILC consumes approximately 618GB memory.It's more complicated than linear search, and is overkill for very small numbers of elements.It works only on lists That are sorted and kept sorted. That is not always feasible, especially if elements are constantly being added to the list.It works only on element types for which there exists a less-than relationship. Some types simply cannot be sorted (though is rare).There are even faster search methods available, such as hash lookups. However a hash lookup requires the elements to be organized in a much more complicated data structure (a hash table, not a list).

4. Proposed System

The main purpose of goal is to design an elegant index, which supports efficient can search for large road networks. Inspired by the classical R-tree in Euclidean space, That design our index on road networks by considering two core features. The first one is a balance tree structure, and That propose a balanced search tree index, called a G - tree. That propose a balanced search tree index, G-tree, which has high pruning power and a small index size.A spatial info manages dimensional objects (such as points, rectangles, etc.), and provides quick access to That object supported totally different choice criteria. The importance of spatial databases is mirrored by the convenience of modeling entities of reality in an exceedingly geometric manner.

4.1 System Architecture

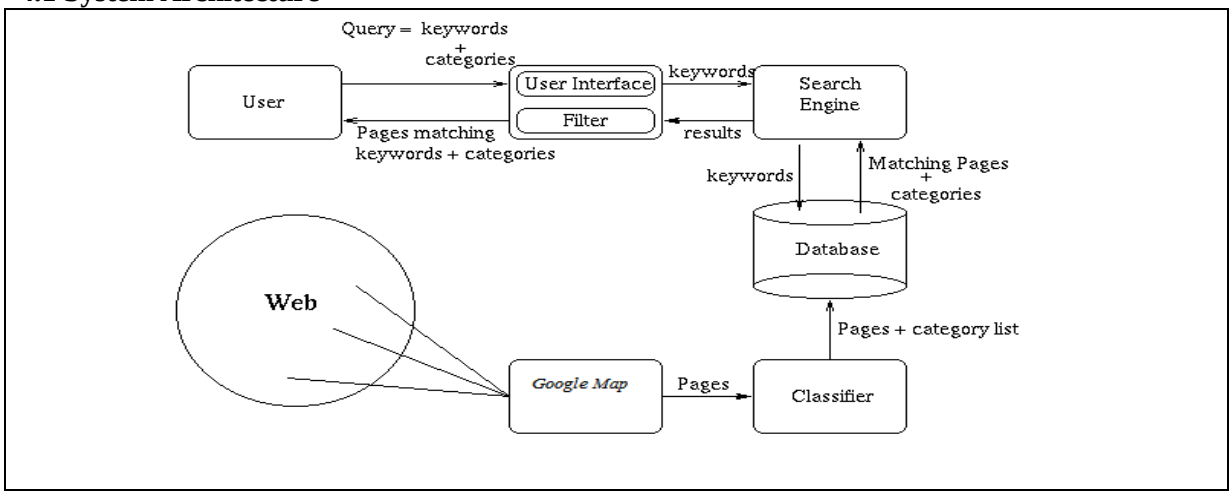

Figure 1: System Architecture

4.2 Testing Analysis

Testing is a process of executing a program with the intent of finding an error. A good test case is one That has a high probability of finding an as-yet –undiscovered error. A successful test is one That uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring That the system works accurately and efficiently as expected before live operation commences. It verifies That the whole set of programs hangs together. System testing requires a test consists of several key activities and steps for running program, string, system and is important in adopting a successful new system. It is the last chance to detect and correct errors before the system is installed for user acceptance testing.

4.2.1 Unit Testing

Unit testing involves the design of test cases That validate That's the internal program logic is functioning properly, and That program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.It is done after the completion of an individual unit before integration. It is a structural testing, That relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure That each unique path of a business process performs accurate to the documented specifications and contains clearly defined inputs and expected results.

4.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate That although the components that individually satisfied, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems That arise from the combination of components.

4.2.3 Functional Test

Functional tests provide systematic demonstrations That functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing are centered on the following itemsValid Input identified classes of valid input must be accepted. Invalid Input identified classes of invalid input must be rejected.Functions identified functions must be exercised.Output identified classes of application outputs must be exercised. Systems/Procedures interfacing systems or procedures must be invoked. Organization and preparation of functional tests are focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4.2.4 System Testing

System testing ensures That the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

4.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas That cannot be reached from a black box level.

4.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements

document. It is a test in which the software under test is treated, as a black box.They cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

5. Screenshot

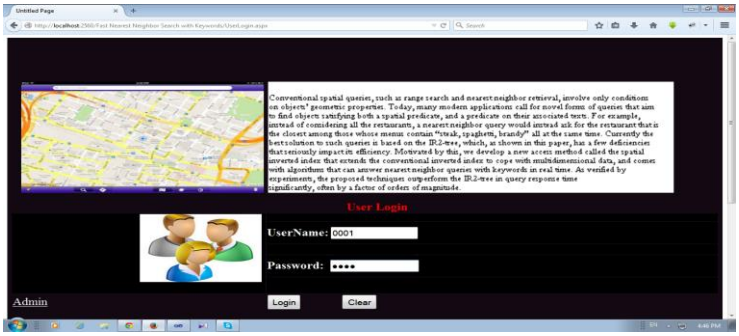

Figure 2: User Login For Location Search

6. Conclusion

It has seen plenty of applications calling for a search engine That is, able to efficiently support novel forms of spatial queries That are integrated with a keyword search. The existing solutions to such queries either incur prohibitive space consumption or are unable to give real time answers. In its paper, That have remedied the situation by developing an access method called the spatial inverted index (SI-index). Not only That the SI-index is fairly space economical, but also it has the ability to perform keyword-augmented nearest neighbor search in time That is on the order of dozens of milliseconds. Furthermore, as the SI-index is based on the conventional technology of inverted index, it is readily incorporable in a commercial search engine That applies massive parallelism, implying its immediate industrial merits.

References

[1] Efficient Processing of Spatial Group Keyword Queries, ACM Transactions on Database Systems (TODS) Gao Cong, Christian S. Jensen (June 2016).

[2] G-tree: an efficient index for KNN search on road networks Ruicheng Zhong (March 2015).

[3] Joint Top-K Spatial Keyword Query Processing, IEEE Transactions on Knowledge and Data Engineering Dingming Wu, Man Lung Yiu, Gao Cong, Christian S. Jensen (October 2014).

[4] Top-K Nearest Keyword Search in Large Graphs Miao Qiao, Thatntao Tian (June 2013).

[5] Approximate Max RS in spatial databases, Proceedings of the VLDB Endowment Yufei Tao, Xiaocheng Hu, Dong-Wan Choi, Chin-Wan Chung (August 2012)

[6] A scalable algorithm for maximizing range sum in spatial databases, Proceedings of the VLDB Endowment Dong-Wan Choi, Chin-Wan Chung, Yufei Tao (July 2010)

