Bulk-Insertions into R-Trees Using the Small-Tree-Large-Tree Approach *

Li Chen, Rupesh Choubey and Elke A. Rundensteiner
Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609
{lichen|rupesh|rundenst}@cs.wpi.edu

Abstract. Previous work on *bulk loading* data focussed at building multidimensional index structures *from scratch*, while the problem of *bulk insertions* into *existing* index structures has been largely overlooked. In this paper, we now propose the STLT (Small-Tree-Large-Tree) technique, which as opposed to the current technique of inserting data one by one, bulk inserts entire new datasets into an active R-tree. STLT considers the new dataset as an R-tree itself (small tree), identifies and prepares a suitable location in the original R-tree (large tree) for insertion, and lastly performs the insert of the small tree into the large tree. Our experiments confirm that for skewed datasets STLT consistently outperforms existing techniques (by up to 65%) for data insertion, while generating an index structure with comparable query retrieval performance.

Keywords: Multidimensional Index Structures, Bulk-loading, Bulk-insertion, Spatial Databases.

1 Introduction

Enormous amount of spatial data needs to be stored and retrieved in diverse applications including Cartography, Computer-Aided Design, computer vision, robotics. Thus alternate index structures have been proposed in recent years for spatial data specifically [Sam89] and for multi-dimensional data more generally. The R-tree structure was initially proposed by Guttman [Gut84] and various variations of the original structure have been suggested ever since. Traditionally, the basic *insert* operation of the R-tree was to use an OBO (one-by-one) technique to load sets of data into an R-tree in a sequential way. Various *bulk loading* strategies have been proposed in recent years aiming at improving the loading efficiency by constructing the complete R-tree all at once *from scratch* [LLE97, KF93]. However, no work has been done thus far on *bulk insertions* of new data into an already existing R-tree.

The importance of this problem for numerous real-world applications is apparent. For example, new data points collected by satellite covering a particular region of the galaxy, coming in in bunch and rather skewed compared to the whole space region, may need to be quickly loaded in order to allow for scientific exploration. Driven by these real-world requirements, this paper investigates the problem of bulk loading of new (possibly *skewed*) datasets into an existing *active* R-tree. By active, we mean an R-tree which already contains (large) datasets and would continue functioning while being minimally impacted by the insertion of new data. By skewed, we mean that the dataset to be inserted is localized to some portion of the region covered by the existing index. We propose a new technique, called STLT (for Small-Tree-Large-Tree), the key idea of which is to consider the new dataset as an R-tree itself (small tree), identify and prepare a suitable location in the original R-tree (large tree) for insertion, and insert the small tree into the large tree.

Contributions of this work [CCR98] are four-fold. One, we identify the problem of *bulk insertions* of spatial data into *existing* index structures. Two, we design STLT as specially tuned solution to address this problem. Three, we implement STLT and the conventional techniques in an integrated environment in order to establish a uniform testbed. Four, we conduct experimental studies both using real-world as well as synthetic datasets that demonstrate the performance gains achievable by STLT. Due to space limitations, our results related to STLT are briefly outlined below, while full details can be found in [CCR98].

2 The STLT Approach

The STLT technique is designed to solve the problem of inserting (skewed) data into an existing R-tree. Our goals are to *minimize the loading time* of new data while *preserving the retrieval performance* of the resulting R-tree at a comparable level. STLT builds a new R-tree (*small tree*) from the new (skewed) datasets, thus requiring no down time of the existing index; and then it inserts the tree as a data object

* This work was supported in part by the University of Michigan ITS Research Center of Excellence grant (DTFH61-93-X-00017-Sub) sponsored by the U.S. Dept. of Transportation and by the Michigan Dept. of Transportation.

into an already existing R-tree (*large tree*); thus just locking as few portions as possible of the R-tree for a short time. time. The four steps of STLT are:

```
1. Build an R-tree (small tree) from scratch
   from new dataset.

2. Find suitable position in original tree (big tree)
   for insertion of small tree.

3. Handle unavailability of entry slot space in large
   tree using heuristics.

4. Insert small tree into identified location
   (or location created as result of Step 3).
```

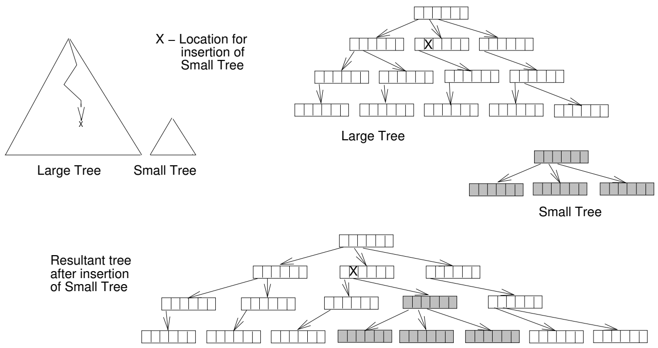

Figure 1: Example of the STLT Process.

For step 1, we build the small R-tree using any of the existing bulk loading techniques. Step 2 determines a suitable place to insert the newly built R-tree into the existing R-tree. For this, we treat the small tree as if it were an individual data object by performing an insert of its root node using the minimum bounding rectangle (MBR) of the root. The main twist here is to assure that the bottom level of the small R-tree is on the same level as that of the original R-tree (Figure 1). We achieve this by forcing termination of the insert step once a level is reached in the large tree that is equal to height of large tree minus height of small tree. If the identified insertion point is found to be full, step 3 of STLT incorporates heuristics for handling this overflow problem to allow for insertion of the whole small tree all at once (as it is assumed to be compact due to being built using highly skewed data). These heuristics attempt some local reorganization of the large tree, and its cost is minimal (in practical settings, a small constant). Step 4 completes the final insertion.

3 Results and Conclusions

Using both synthetic and real datasets, we have conducted two major classes of tests to evaluate the performance of STLT over OBO in terms of *loading time* and the *quality* of the resultant trees for different parameters [CCR98]. Our experimental results show that STLT consistently and many-fold outperforms alternate techniques in terms of loading time. On the other hand, the quality of the resulting tree constructed by STLT in terms of query performance deteriorates as the spatial region covered by the small tree is increased. However, even under the testing scenario of a not-so-localized small tree area (the ratio reaches up to 40% of small tree area versus large tree area), STLT performs as well as OBO. And, when considering tree quality, we note that STLT does especially well (over 65% of improvement in I/O costs over the OBO) for skewed datasets as well for large size ratios (when the small tree is kept sufficiently small, i.e., 0.5% size, as compared to the large tree size), while achieving trees identical in terms of retrieval performance to those generated by OBO. All in all, the STLT *bulk insertion* strategy has thus been found to be a viable and significant optimization over the conventional one-by-one insertion approach.

Acknowledgments. The authors would like to thank Yun-Wu Huang, Ning Jing, and Matthew Jones who have developed most of the experimental system that served as platform for our work. We also thank students at the Database Systems Research Group at WPI for their feedback on this research.

References

[CCR98] L. Chen, R. Choubey, and E.A. Rundensteiner. Bulk insertions into R-trees. *WPI, Tech. Rep. CS-WPI-98-05*, Feb. 1998.

[Gut84] A. Guttman. R-trees : A dynamic index structure for spatial searching. *Proceedings of SIGMOD*, pages 47–57, 1984.

[HJR97a] Y.W. Huang, N. Jing, and E.A. Rundensteiner. Integrated query processing strategies for spatial path queries. *IEEE Int. Conf. on Data Eng.*, April 1997.

[HJR97b] Y.W. Huang, N. Jing, and E.A. Rundensteiner. Spatial joins using R-trees: Breadth-first traversal with global optimizations. *Int. Conf. on Very Large Data Bases*, 1997.

[HNR97] Y.W. Huang, N.Jing, and E.A. Rundensteiner. Cost model for estimating performance of spatial joins using R-tree. *SSDBM*, Aug. 1997.

[KF93] I. Kamel and C. Faloutsos. On packing R-trees. *Proceedings of International Conference on Information and Knowledge Management*, pages 490–499, November 1993.

[LLE97] S. Leutenegger, M. Lopez, and J. Edgigton. STR: A simple and efficient algorithm for R-tree packing. *IEEE Int. Conf. on Data Eng.*, pages 497–506, 1997.

[Sam89] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1989.