Efficient Spatio-Temporal Search of Objects Moving on a Graph

Thuy Thi Thu Le and Bradford G. Nickerson

m6839|bgn@unb.ca
Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, N.B. Canada E3B 5A3

1. Introduction

We address the challenge of indexing a large number of moving objects to improve the response time of searching for the identity of all moving objects intersecting (time × space) rectangular queries. This paper focuses on indexing historical positions of objects moving on a fixed graph $G = (V, E)$ containing $|E|$ edges and $|V|$ vertices. The graph can be non-planar as it often is when representing road networks (Eppstein and and Goodrich 2008). Existing work on indexing objects moving on a graph includes the MON-tree (de Almeida and Güting 2005), PPFI (Fang et al. 2008), FNR-tree (Frentzos 2003), and (Pfoser and Jensen 2003). The common point of these data structures is to combine several R-trees to index moving objects on a fixed network. A network is indexed by an R-tree (Leutenegger and Lopez 2005) while moving objects are indexed on a forest of R-trees, whose roots are linked to leaf nodes of the network tree. An object moving at a constant velocity on an edge is represented as a (time interval × position interval) rectangle. The disadvantage of these data structures is that the number of reported objects for a query can be many more than the exact result. Figure 1 shows an example of five rectangles representing five moving objects o_1, ..., o_5 intersecting a shaded query rectangle R. Only two moving objects o_3 and o_4, whose

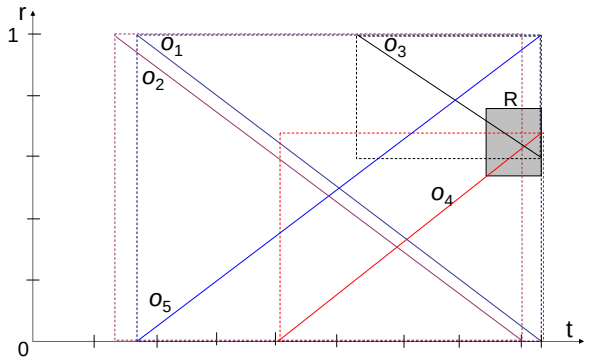


Fig. 1: Five rectangles represent five moving objects o_1, ..., o_5. Only two diagonal line segments of o_3 and o_4 intersect the shaded rectangle query R.

rectangle's diagonal line segments intersect with R, are actually in range. In the worst case, all moving object rectangles on an edge intersect the query rectangle R, but none of them is in range.

We propose a new data structure that allows us to precisely retrieve moving objects intersecting a query. Instead of using R-trees to index bounding boxes of moving objects, we index oriented line segments representing positions of moving objects at different times. We can answer a rectangle R plus time interval query in $O(\log_2 |E| + |L| \log_2(n/|L|) + k)$ time, where n is the number of moving object instances (unique entries of moving objects) on a graph with $|E|$ edges, $|L|$ is the number of edges intersected by R and k is the precise number moving object instances in range. This improves our previous search time results of (Le and Nickerson 2008) by reporting only those moving objects in range. None of the previous research reports worst case query time, but they all depend on R-tree indexing for spatial search of the graph which requires $\Omega(|E|^{\frac{1}{2}})$ time. Further analysis of previous research search complexities is given in Technical report TR10-199 (Le and Nickerson 2010).

2. Proposed Approach

We support the query $Q_2 = (R, [t_1, t_2])$ to count or report the k moving objects intersecting rectangle R at any time during time interval $[t_1, t_2]$. A spatio-temporal query $Q_2 = (R, [t_1, t_2])$ is transformed to query rectangles $Q_3 = [t_1, t_2] \times [r_1, r_2]$ by finding positions r_1, r_2 that span the query rectangle R on an edge. In Figure 2a and Figure 2b, a query Q_2 on an edge is transformed to two query rectangles Q_3.

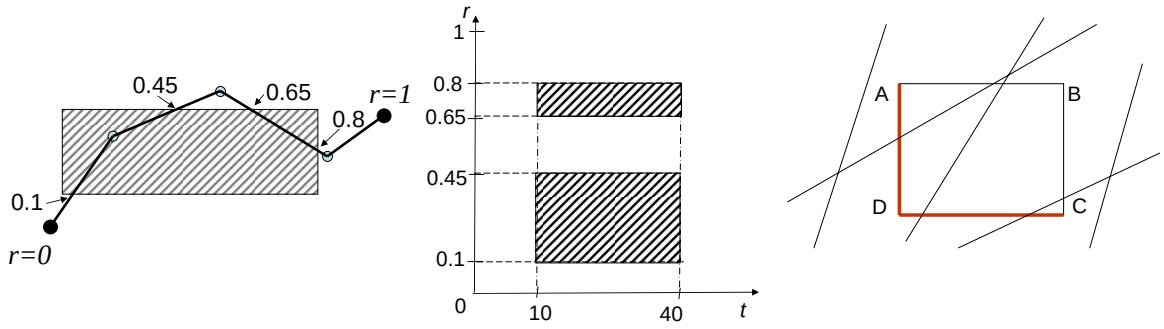


Fig. 2: (a) Spatio-temporal query $Q_2 = (R, [10, 40])$ is transformed into (b) two query rectangles Q_3: [10,40]×[0.1, 0.45] and [10,40]×[0.65, 0.8]. (c) lines with slopes $m \in (0, \infty]$ intersect rectangle $ABCD$ if they intersect AD or DC.

Each moving object is represented by a diagonal line segment of a (time interval) × (position interval) rectangle. Each point on this line segment corresponds to a position of a moving object at a specific time. If a line segment intersects a query rectangle, its corresponding moving object is in range. When objects move across an edge from $r = 0$ to $r = 1$, their corresponding path on that edge is considered as a line in a bounded plane formed by $(t \times r)$, for $t \in [0, T]$ the time domain, and $r \in [0, 1]$ positions on an edge. Our approach reduces to find the intersections of a query rectangle $ABCD$ (see Figure 2c) with a set of lines having slope $m \in (0, \infty]$. Lines intersect the query rectangle if they intersect line segments AD or DC. We divide a set of bounded lines into two subsets L_1 and L_2. L_1 contains objects moving from $r = 0$ to $r = 1$, and

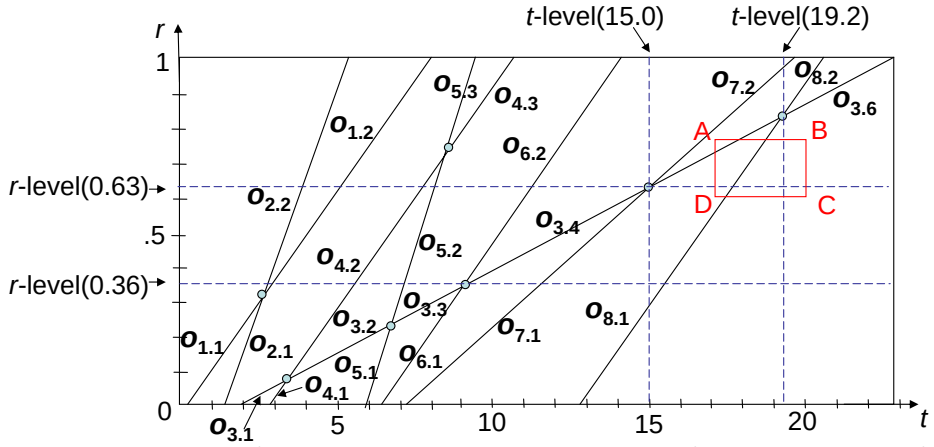


Fig. 3: Lines represent 8 objects moving at a constant velocity over an edge in the direction from $r = 0$ to $r = 1$. $o_{j.i} \in$ the line representing moving object o_j. Query rectangle $ABCD$ has points A=(17, 0.77) and C=(20, 0.6). Dashed lines show t-levels and r-levels near AD and DC. Objects o_3 and o_8 are in range.

L_2 contains objects from $r = 1$ to $r = 0$ (with slope $m \in (-\infty, 0]$). Without loss of generality, the rest of this paper focuses only on indexing and searching in L_1. Algorithms and analysis for L_1 also hold for L_2.

We use the notion t-level(i) to refer to a set of lines intersecting line $x = i$ ordered top-to-bottom. Similarly, r-level(i) refers to a set of lines intersecting line $y = i$ ordered left-to-right. Figure 3 shows an example of two t-levels: t-level(15.0) and t-level(19.2), and two r-levels: r-level(0.36) and r-level(0.63). The order of lines changes where lines intersect. Consider a set of lines and a query rectangle $ABCD$ in Figure 3, we only need to search for lines intersecting AD on t-level(15) and DC on r-level(0.36). We build a data structure for efficient search based on this idea.

An ordered polyline p_i is created by connecting line segments at intersections (with each other and with the $r = 0$, and $r = 1$ boundaries). For example, the first three ordered polylines in Figure 3 are $p_1 = \{o_{1.1}, o_{2.2}\}$, $p_2 = \{o_{2.1}, o_{1.2}\}$, and $p_3 = \{o_{3.1}, o_{4.2}, o_{5.3}\}$, ordered from left to right; they do not intersect each other. Points in an ordered polyline are monotonically increasing in both t and r. We connect points in an ordered polyline together into a list of entries, and arrange ordered polylines in a balanced search tree.

In the worst case, every line representing a moving object instance intersects the lines representing all other moving object instances on the same edge (see Figure 4). Therefore, there are $O(g_i^2)$ lines for g_i moving object instances on edge i, and each ordered polyline requires $O(g_i)$ line segments. The number of ordered polylines is still precisely g_i.

3. Indexing Moving Objects on an Edge

Ordered polylines are arranged as a balanced binary search tree, called an *ordered polyline tree*, based on each p_i dividing space ($t \times r$) into two parts. Each ordered polyline contains a list of entries. Each entry contains a point (t, r), a moving object

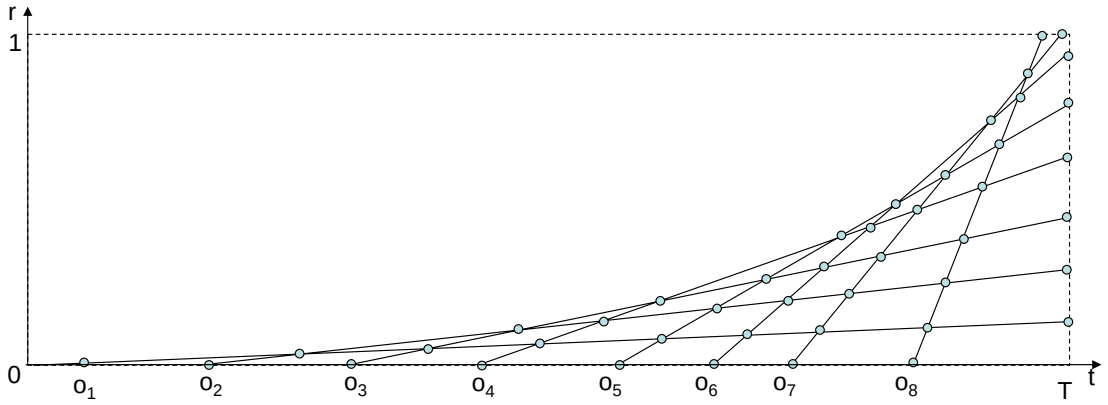


Fig. 4: Example of 8 bounded lines representing 8 moving object instances o_1, ..., o_8 in the worst case, where each object instance intersects 7 others in time.

ID corresponding to the line connecting to the point, three t-pointers, and one r-pointer. Three t-pointers point to the left, right, and next adjacent entries (belonging to the left, right, and next adjacent ordered polylines, respectively) on t-levels. One r-pointer points to the next adjacent entry belonging to the next adjacent ordered polyline on r-levels.

For a polyline p_i with t-entry t_j, the (left, right, next) pointers point to the largest t-entry in p_i's (left, right, next) node $\leq t_j$, respectively. If no t-entries in p_i's (left, right, next) nodes are $\leq t_j$, the (left, right, next) pointers point to the smallest t-entry $> t_j$. In this way, we record all line segments in the arrangement of bounded lines such that a traversal of the tree from root to leaf serves to find the polyline immediately to the left of the query point A. Following next pointers of t-entries finds segments of ordered polylines in downward order for a vertical query segment AD. Following next pointers of r-entries finds segments of ordered polylines in left-to-right order for a horizontal query segment DC. Figure 5 shows an example of an ordered polyline tree. Ordered polyline trees can be made dynamic as presented in (Le and Nickerson 2010).

4. Indexing Edges of a Graph

An edge on a fixed graph $G = (V, E)$ is considered as a polyline. We index each edge by a strip tree (Ballard 1981). Strip trees are merged bottom up in pairs to construct a *graph strip tree*. Figure 6a shows an example of strip trees created from a fixed graph containing four edges e_1, e_2, e_3, and e_4, and Figure 6b shows a graph strip tree from the merged strip trees. Leaf nodes C_i point to strip tree S_i spatially indexing e_i and to ordered polyline tree T_i indexing lines representing moving objects on e_i. Note that T_i stores the L_1 subset of the moving objects. Although not shown in Figure 6b, a second ordered polyline tree U_i storing objects in L_2 is required.

An ordered polyline tree uses $O(g_i + \lambda_i)$ space to index a set of g_i lines with λ_i intersections among them. A graph strip tree with $|E|$ edges uses $O(|E| + n + \lambda)$ space to index a set of n moving object instances with λ intersections among lines

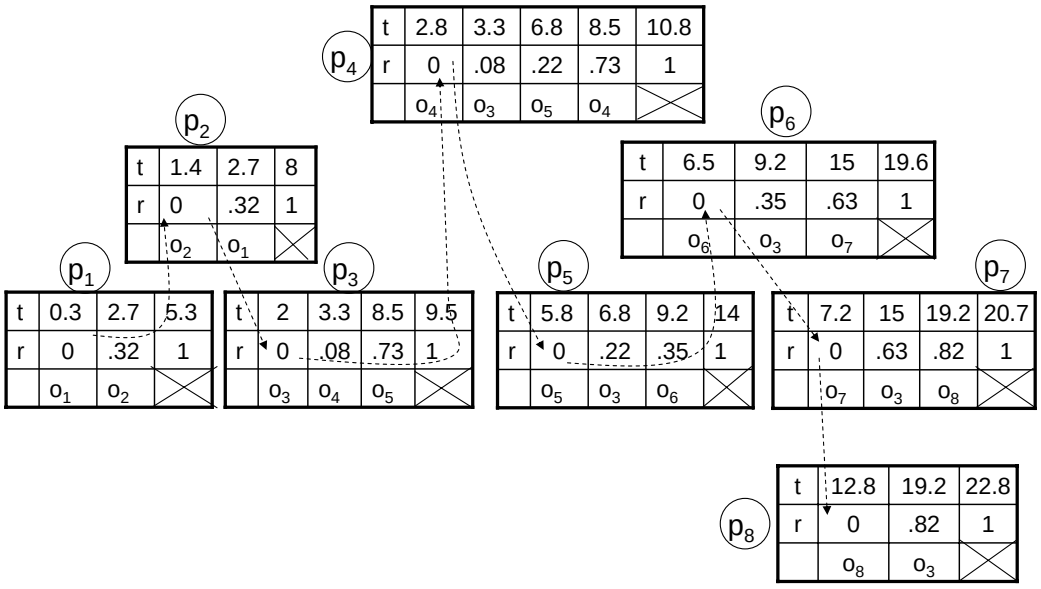


Fig. 5: Ordered polyline tree indexing the 8 lines from Figure 3. A three-row rectangle represents an ordered polyline, where each column is a point represented as an entry. A dashed line represents a pointer of an entry to its adjacent entry for r-level(0).

representing moving objects (i.e., $\lambda = \sum_{i=1}^{|E|} \lambda_i$). Proof of the space complexity is given in Technical report TR10-199 (Le and Nickerson 2010).

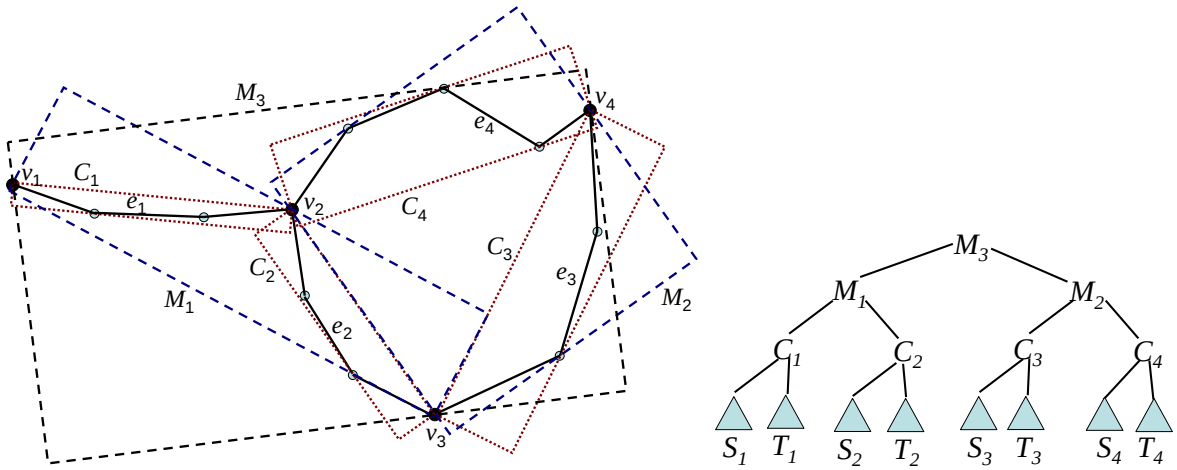
5. Search Complexity

Given query rectangle Q_3 with four vertices A, B, C, D in the clockwise direction with $D = (x, y)$, searching finds lines in t-level(x) intersecting AD and lines in r-level(y) intersecting DC. For a single edge, and assuming objects move at a constant velocity, the time to report moving objects intersecting a query rectangle Q_3 on the ordered polyline tree T_i is $O(\log_2(g_i) + k_i)$, where g_i is the number of moving object instances stored in T_i, and k_i is the number of moving object instances in range.

A graph strip tree indexing n moving object instances on $|E|$ edges of a graph answers a Q_2 query in time $O(\log_2 |E| + |L| \log_2(\frac{n}{|L|}) + k)$, for k the number of moving object instances in range, and $|L|$ the number of edges on the graph intersecting Q_2. Theorems proving these results are in Technical report TR10-199 (Le and Nickerson 2010). If the number of moving object instances is much greater than the number of edges on the graph (i.e., $n \gg |E|$), we expect the search time to be dominated by the time $O(|L| \log_2(\frac{n}{|L|}) + k)$ to search $|L|$ ordered polyline trees.

6. Conclusion

We present a new data structure for efficient search of objects moving on a graph. Our data structure is a combination of strip trees and ordered polyline trees. Strip trees are used for spatial indexing of the graph edges. Ordered polyline trees are used to index moving objects on edges. For n moving object instances on a graph with $|E|$ edges, we



(a) (b)

Fig. 6: (a) Four edges of graph G are represented as four strip trees, with $C_1, ..., C_4$ representing the root bounding boxes for each strip tree. Bounding boxes are merged bottom up in pairs to construct (b) a graph strip tree.

show that $O(\log_2 |E| + |L| \log_2(\frac{n}{|L|}) + k)$ time is required to answer a Q_2 query, for $|L|$ the number of edges intersecting Q_2, and k the number of moving objects in range. For λ intersections among paths of n moving object instances, the required space for the graph strip tree is $O(n + \lambda + |E|)$.

We are working on experimental validation of the results presented here using Canadian road networks. An open problem is how to efficiently index moving object instances to achieve an I/O-efficient worst case optimal search complexity.

Acknowledgements

This research is supported, in part, by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the UNB Faculty of Computer Science, and the government of Vietnam. We gratefully acknowledge this support.

References

Ballard D. H., 1981, Strip trees: a hierarchical representation for curves. *Communications of the ACM*, 24(5):310-321.

de Almeida V. T. and Güting R. H., 2005, Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33-60.

Eppstein D. and Goodrich M. T., 2008, Studying (non-planar) road networks through an algorithmic lens. In Proc. of the 16th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM GIS 2008), Irvine, CA, USA, Nov.5-7, 1-10.

Fang Y., Cao J., Peng Y., and Wang L., 2008, Indexing the past, present and future positions of moving objects on fixed networks. In Proc. of the Int. Conf. on Computer Science and Software Engineering (CSSE 2008), Wuhan, China, Dec. 12-14, IEEE Computer Society Press, 524-527.

Frentzos E., 2003, Indexing objects moving on Fixed networks. In Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD 2003), Santorini, Greece, Jul. 24-27, 289-305.

Le T. T. T. and Nickerson B. G., 2008, Efficient Search of Moving Objects on a Planar Graph. In Proc. of the 16th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM GIS 2008), Irvine, CA, USA, Nov. 5-7, 367-370.

Le T. T. T. and Nickerson B. G., 2010, Towards a Dynamic Data Structure for Efficient Bounded Line Range Search. In Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG 2010), Winnipeg, Manitoba, Canada, Aug. 9-11, in press.

Le T. T. T. and Nickerson B. G., 2010, Graph Strip Tree for Efficient Search of Objects Moving on a Graph. Technical report, TR10-199, Faculty of Computer Science, UNB, Fredericton, Canada, 15 pages.

Leutenegger S. and Lopez M. A., 2005, *Handbook of Data Structures and Applications*, Chapter 12.

Pfoser D. and Jensen C. S., 2003, Indexing of network constrained moving objects. In Proc. of the 11th ACM Int. Symposium on Advances in Geographic Information Systems (ACM GIS 2003), New Orleans, Louisiana, USA, Nov. 7-8, 25-32.