Approximate Orthogonal Range Search using Patricia Tries

Bradford G. Nickerson Qingxiu Shi *

Abstract

We use Patricia tries to answer ϵ-approximate orthogonal range search on a set of n random points and rectangles in k-d space. The approximate orthogonal range search time using Patricia tries is determined theoretically to be $O(k \log n/\epsilon^{k-1})$ for cubical range counting queries, improving the best known upper bound for uniformly distributed input points and query rectangles. Patricia tries are evaluated experimentally for ϵ-approximate orthogonal range counting and reporting queries (for $2 \leq k \leq 10$ and n up to 1,000,000) using uniformly distributed random points and rectangles. For $\epsilon = 0.05$, an average of 50% fewer nodes are visited for the Patricia trie (compared to the exact range search).

1 Introduction

Range search is among the fundamental problems in computational geometry, geographical information systems, computer graphics and database applications. Given a collection of keys (each containing multidimensional attributes) and a multidimensional query rectangle, an orthogonal range search asks for all keys in the collection with attribute values each inside the given rectangle. Over the past 30 years, more than 60 data structures for range search have been presented [1][3][5][7]. The motivation of our work is to improve the speed of range search while allowing for small counting (or reporting) errors.

Chazelle [4] gives a comprehensive overview of data structures for k-d searching, including the description of a dynamic k-d range reporting algorithm requiring $O(F(\log(\frac{2n}{F})^2) + \log^{k-1} n)$ time (F = number of points found in range), which is close to the lower bound. To obtain better performance, several researchers turned to an approximate version of the range searching problem: instead of counting the points in the exact specified ranges, the data point whose distance to the boundary of the range is within ϵ times the range's diameter may or may not be included in the count. The approximate range searching problem was solved optimally by Arya and Mount [2]. With an $O(kn)$-space structure called the balanced box-decomposition tree which can be constructed in $O(kn \log n)$ time, ϵ-approximate range counting queries can be answered in $O(2^k \log n + (3\sqrt{k}/\epsilon)^k)$ time. If the ranges are convex, the query time can be strengthened to $O(2^k \log n + k^2(3\sqrt{k}/\epsilon)^{k-1})$. They also presented a lower bound of $\Omega(\log n + 1/\epsilon^{k-1})$, for the complexity of answering ϵ-approximate range counting queries assuming a partition tree approach for cubical range in fixed dimension.

The k-d trie [9] splits the search space based on the digits of the data. Each partition splits a region of the search space into two sub-regions of equal size. The splitting is not continued further when a sub-region contains one or no data points. The k-d trie has an annoying defect: there is one-way branching that leads to the creation of extra nodes in the tree. The Patricia trie, discovered by D.R. Morrison [8], avoids this problem by removing all one-child internal nodes from the k-d trie and storing the eliminated information in the nodes. Patricia tries can be preprocessed in $O(n \log n)$ time and $O(kn)$ space, and fewer internal nodes are visited for a partial match search of a Patricia trie compared to a k-d trie [6].

2 Approximate Orthogonal Range Search

Given n k-d points or rectangles and a k-d query rectangle, ϵ-approximate orthogonal range query counts (or reports) points in the query rectangle or rectangles intersecting the query rectangle, allowing errors near the boundary of the query rectangle. We define a k-d query rectangle $W = [L_1, H_1] \times [L_2, H_2] \times \cdots \times [L_k, H_k]$, $L_i \leq H_i$ with center $Z = (\frac{L_1+H_1}{2}, \frac{L_2+H_2}{2}, \cdots, \frac{L_k+H_k}{2})$. The edges of W have given lengths $\Delta_1, \Delta_2, \cdots, \Delta_k$, where $\Delta_i = H_i - L_i$, $\forall i \in \{1, 2, \cdots, k\}$. We define $[MIN_i, MAX_i]$, $\forall i \in \{1, 2, \cdots, k\}$, as the minimum and maximum possible data coordinate values for dimension i. Given $0 \leq \epsilon \leq 0.5$, let $W^- = [L_1 + \Delta_1\epsilon, H_1 - \Delta_1\epsilon] \times [L_2 + \Delta_2\epsilon, H_2 - \Delta_2\epsilon] \times \cdots \times [L_k + \Delta_k\epsilon, H_k - \Delta_k\epsilon]$ be the k-d inner query rectangle with center at Z, and let $W^+ = [L_1 - \Delta_1\epsilon, H_1 + \Delta_1\epsilon] \times [L_2 - \Delta_2\epsilon, H_2 + \Delta_2\epsilon] \times \cdots \times [L_k - \Delta_k\epsilon, H_k + \Delta_k\epsilon]$ be the k-d outer query rectangle with center at Z. (We assume $MIN_i \leq L_i - \Delta_i\epsilon$ and $H_i + \Delta_i\epsilon \leq MAX_i$, $\forall i \in \{1, 2, \cdots, k\}$). For an ϵ-approximate range search query, points inside W^- must be counted, and points outside W^+ must not be counted, and points between W^- and W^+ may be miscounted (see Figure 1).

We denote by T the Patricia trie constructed by inserting a set D of n k-d data points into an initially

*University of New Brunswick, Fredericton, NB, E3B 5A3, Canada {bgn,l749u}@unb.ca

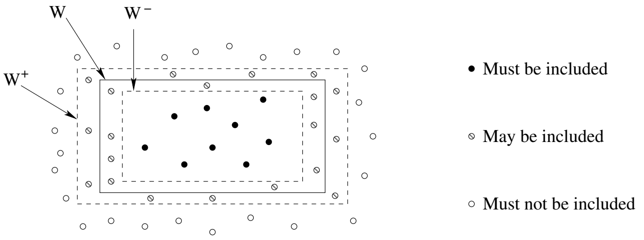


Figure 1: Approximate orthogonal range search queries for $\epsilon \approx 0.1$.

empty trie (the insertion algorithm is in [11]). There are altogether $n-1$ internal nodes and n leaves in T. The skipped bits are stored in an array SKIPSTR, and every leaf is associated with one point.

Each node in a k-d trie covers part of the k-d space, that is, every node has a cover space defined as $NC = [\mathcal{L}_1,\mathcal{U}_1] \times [\mathcal{L}_2,\mathcal{U}_2] \times \cdots \times [\mathcal{L}_k,\mathcal{U}_k]$. Arrays $\mathcal{L}$ and $\mathcal{U}$ store the lower and upper bounds of a node's cover space. The root of k-d tries cover the whole space and child nodes cover half of the search space volume of their parent. For the root, NC has $\mathcal{L}_i = MIN_i$ and $\mathcal{U}_i = MAX_i$, $\forall i \in \{1, \cdots, k\}$. The nodes on level ℓ split at attribute $p = (\ell \bmod k) + 1$ (at the root, $\ell = 0$). If a node on level ℓ has cover space $[\mathcal{L}_1,\mathcal{U}_1] \times \cdots \times [\mathcal{L}_p,\mathcal{U}_p] \times \cdots \times [\mathcal{L}_k,\mathcal{U}_k]$, then its left child's cover space is $[\mathcal{L}_1,\mathcal{U}_1] \times \cdots \times [\mathcal{L}_p, (\mathcal{L}_p + \mathcal{U}_p)/2] \times \cdots \times [\mathcal{L}_k,\mathcal{U}_k]$, and its right child's cover space is $[\mathcal{L}_1,\mathcal{U}_1] \times \cdots \times ((\mathcal{L}_p+\mathcal{U}_p)/2, \mathcal{U}_p] \times \cdots \times [\mathcal{L}_k,\mathcal{U}_k]$. For Patricia tries, ℓ is not the level of the trie, but the length of the path from root to the node plus the length of the skipped bits in the internal nodes along the path. The node cover space must take the skipped bit string stored in the nodes into consideration.

The ARC algorithm (Figure 2) is used to perform an approximate range counting query on T. The search proceeds from the root to the leaves, accounting for possible skipped bits stored at internal nodes. Arrays $\mathcal{L}$ and $\mathcal{U}$ are the lower and upper limits of the node's cover space, and are initialized to be MIN_i and MAX_i, respectively, $\forall i \in \{1, \cdots, k\}$. If NC falls within W^+ at some node T, then all data points in the subtree attached to T are the answers of the approximate range search. If NC falls outside of W^-, the range search stops. If NC overlaps both W^- and W^+, then we recursively visit T's children. When we reach a leaf node, we determine whether the data point stored in this node is in W using the CheckNode function.

3 Approximate Range Searching Cost

Without loss of generality, the following discussions are all based on unit space $[0,1]^k$. We assume the input data and the query rectangle W are drawn from a uniform random distribution.

```
ARC(T, ℓ, L, U, W⁻, W, W⁺, Count)
if T is a leaf node
   then if T.POINT ∈ W
           then Count ← Count + 1
   else i ← 0
        while i < T.SKIPSTR.length()
        do p ← (ℓ mod k) + 1
           if T.SKIPSTR[i] = 0
              then U[p] ← (L[p] + U[p])/2
              else L[p] ← (L[p] + U[p])/2 + ε
           i ← i + 1
           ℓ ← ℓ + 1
        black = 0
        for (i = 1; i <= k; i ← i + 1)
        do if (W⁺.L[i] ≤ L[i]) and
              (U[i] ≤ W⁺.H[i])
              then black ← black + 1
              else break
        if black = k
           then Count ← Count + T.WEIGHT
                return
        for (i = 1; i <= k; i ← i + 1)
        do if (L[i] > W⁻.H[i]) or
              (U[i] < W⁻.L[i])
              then return
        p ← (ℓ mod k) + 1
        if left[T] ≠ NIL
           then U[p] ← (L[p] + U[p])/2
                ARC(left[T], ℓ + 1, L, U,
                W⁻, W, W⁺, Count)
        if right[T] ≠ NIL
           then L[p] ← (L[p] + U[p])/2 + ε
                ARC(right[T], ℓ + 1, L, U,
                W⁻, W, W⁺, Count)
```

Figure 2: Pseudo-code for the approximate range counting algorithm in the Patricia trie. $T.SKIPSTR$ is the skipped bit string stored in T and $T.SKIPSTR.length()$ is the length of $T.SKIPSTR$. $T.SKIPSTR[i]$ is the ith bit of $T.SKIPSTR$. ε is a small value to guarantee T's left and right children's cover spaces don't share any point. $T.POINT$ is the data point stored in T, and $T.WEIGHT$ is the number of points in the subtree attached to T.

Theorem 1 *Given a Patricia trie T built from n random k-d data points and a random query rectangle W of dimensions $\Delta_1 \times \Delta_2 \times \cdots \times \Delta_k$, and $0 < \epsilon \le 0.5$, ϵ-approximate range counting queries visit*

$$O(\log n \Sigma_{p=1}^{k} (\prod_{i=1, i \neq p}^{k} (2 + \frac{\Delta_i}{\Delta_p}(\frac{1}{\epsilon} - 2))))$$

nodes in T.

Proof. A node is said to be expanded if the algorithm visits the children of this node. For a node to be expanded, its node cover space must intersect with both the inner query rectangle W^- and the outer query rectangle W^+. We call the facet of the query rectangle perpendicular to the pth orthogonal axis the p-facet. According to the definition of W^- and W^+, the p-facets of W^- and W^+ are separated from each other at least by a distance of $2\Delta_p\epsilon$, $\forall p \in \{1, \cdots, k\}$. So a node in T with $|NC(p)| < 2\Delta_p\epsilon$, $\forall p \in \{1, \cdots, k\}$ is never expanded in the algorithm ARC, where $|NC(p)|$ is the length of the p-th side of node cover space NC.

Each partition of T splits a region of the search space into two equal sub-regions. Each coordinate axis gets cut in turn, in a cyclical fashion of $1, 2, \cdots, k, 1, 2, \cdots$, which results in regions such that the length of the longest side is equal to or twice that of the smallest side, and $|NC(1)| \leq |NC(2)| \leq \cdots \leq |NC(k)|$. Assume a node in T intersects both the p- facet of W^- and the p-facet of W^+, $1 \leq p \leq k$, then $|NC(p)| \geq 2\Delta_p\epsilon$, and $|NC(i)| \geq \Delta_p\epsilon$, $\forall i \in \{1, \cdots, p-1\}$, and $|NC(j)| \geq 2\Delta_p\epsilon$, $\forall j \in \{p+1, \cdots, k\}$ (see Figure 3). So there are at most

$$\left(\prod_{i=1}^{p-1}\left(1 + \left\lceil \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon} \right\rceil\right)\right)\left(\prod_{j=p+1}^{k}\left(1 + \left\lceil \frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon} \right\rceil\right)\right)$$

regions intersecting with both the p-facet of W^- and the p-facet of W^+. Each k-d rectangle has $2k$ facets, so altogether there are at most

$$2\Sigma_{p=1}^{k}\left(\prod_{i=1}^{p-1}\left(1 + \left\lceil \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon} \right\rceil\right)\right)\left(\prod_{j=p+1}^{k}\left(1 + \left\lceil \frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon} \right\rceil\right)\right)$$
$$\leq 2\Sigma_{p=1}^{k}\left(\prod_{i=1}^{p-1}\left(2 + \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon}\right)\right)\left(\prod_{j=p+1}^{k}\left(2 + \frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon}\right)\right)$$
$$\leq 2\Sigma_{p=1}^{k}\left(\prod_{i=1, i\neq p}^{k}\left(2 + \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon}\right)\right)$$

regions overlapping both W^- and W^+. The depth of the Patricia trie is $O(\log n)$ [12], so we reach the desired result. □

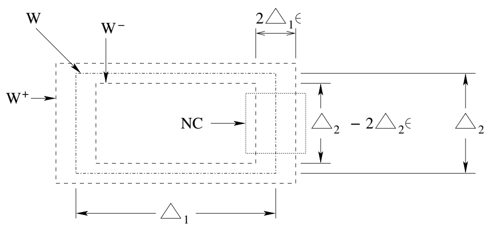


Figure 3: An illustration of a node in T with cover space NC intersecting both the 1-facet of W^- and the 1-facet of W^+; $\epsilon = 0.1$.

Corollary 2 *Given a Patricia trie T built from n random k-d data points and a random query square W, and $0 < \epsilon \leq 0.5$, ϵ-approximate range counting queries visit*

$$O(k \log n / \epsilon^{k-1})$$

nodes in T.

Besides the number of nodes visited in the *ARC* algorithm, up to $2F$ additional nodes are visited in answering the approximate range search reporting queries [10], where F is the number of points in range.

4 Experiments

Our experiments of approximate range counting and reporting were performed using uniformly and randomly distributed data from the interval $[0, 1]$ for ϵ ranging from 0 to 0.5, $2 \leq k \leq 10$, n up to 1,000,000, and the query square's volume vol ranging from 0.0001 to 0.01. The programs were run on a Sun Microsystems V880 with four 1.2 GHz UltraSPARC III processors, 16 GB of main memory, running Solaris 8. Each experimental point in the following graphs was done with an average of 300 test cases.

4.1 k-d points

The experimental results with k-d query square's volume $vol = 0.001$ for Patricia tries are found to be consistent with the theoretical analysis (see Figure 4). We find that there are significant improvements in running time when ϵ grows from 0 to 0.05. As ϵ increases, the running times tend to converge, irrespective of k. Figure 5 shows $f = \frac{y_{\epsilon=x}}{y_{\epsilon=0}}$, where $y_{\epsilon=0}$ is the number of nodes visited for an exact range counting query, and $y_{\epsilon=x}$ is the number of nodes visited for an ϵ-approximate range counting query. The average improvement of the approximate range counting queries when $\epsilon = 0.05$ and $k \leq 5$ is more dramatic than that of the range reporting queries.

For query squares with fixed side length w, we define the fraction of points miscounted as $\delta_{\epsilon=x} = \frac{|F_{\epsilon=x} - F_{\epsilon=0}|}{F_{\epsilon=0}}$, where $F_{\epsilon=x}$ is the number of points counted as in range for an ϵ-approximate range query, and $F_{\epsilon=0}$ is the number of points in the exact range query. The experimental results show that when $k = 2$, $n = 1,000,000$ and $vol = 0.001$, $\delta_{\epsilon=x}$ ranges from 0.002 ($\epsilon = 0.05$) to 0.05 ($\epsilon = 0.5$) [10], so relatively few points are miscounted.

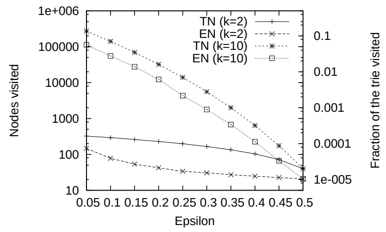
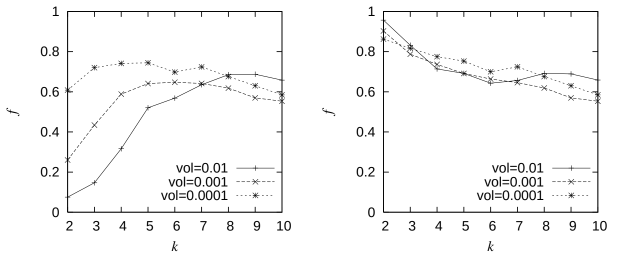


Figure 4: The theoretical (TN) and experimental (EN) number of nodes visited in Patricia trie for counting k-d points with k-d query square volume $vol = 0.001$ for different k ($n = 1,000,000$).



(a) Range counting (b) Range reporting

Figure 5: The fraction f of nodes visited when $\epsilon = 0.05$ for query squares with fixed volume vol for different k ($n = 1,000,000$).

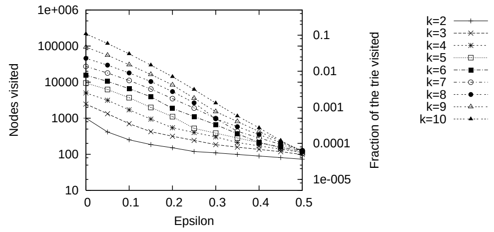


Figure 6: Number of nodes visited versus ϵ in Patricia trie for counting k-d rectangles with k-d query square's volume $vol = 0.001$ ($n = 1,000,000$ and $maxsize = 0.001$).

4.2 k-d rectangles

A k-d rectangle R can be represented as a $2k$-d point $(x_1^{min}, x_1^{max}, x_2^{min}, x_2^{max}, \cdots, x_k^{min}, x_k^{max})$ [11]. Given a k-d query rectangle $W = [L_1, H_1] \times [L_2, H_2] \times \cdots \times [L_k, H_k]$, R intersects W iff $x_i^{min} \in [MIN_i, H_i]$ and $x_i^{max} \in [L_i, MAX_i]$, $\forall i \in \{1, \cdots, k\}$. Each node in T covers part of the $2k$-d space, that is, every node has a cover space defined as $NC = [\mathcal{L}_1, \mathcal{U}_1] \times [\mathcal{L}_2, \mathcal{U}_2] \times \cdots \times [\mathcal{L}_{2k}, \mathcal{U}_{2k}]$. We define the query rectangle W's cover space $WC = [MIN_1, H_1] \times [L_1, MAX_1] \times \cdots \times [MIN_k, H_k] \times [L_k, MAX_k]$, the inner query rectangle W^-'s cover space $WC^- = [MIN_1, H_1 - \Delta_1\epsilon] \times [L_1 + \Delta_1\epsilon, MAX_1] \times \cdots \times [MIN_k, H_k - \Delta_k\epsilon] \times [L_k + \Delta_k\epsilon, MAX_k]$, and the outer query rectangle W^+'s cover space $WC^+ = [MIN_1, H_1 + \Delta_1\epsilon] \times [L_1 - \Delta_1\epsilon, MAX_1] \times \cdots \times [MIN_k, H_k + \Delta_k\epsilon] \times [L_k - \Delta_k\epsilon, MAX_k]$. The ARC algorithm (Figure 2) can be used for k-d rectangles with some modifications: all ks are changed to $2k$s, and WC^-, WC and WC^+ are used instead of W^-, W and W^+.

For our experiments, rectangle centers were uniformly distributed and the lengths of their sides uniformly and independently distributed between 0 and $maxsize$ ($0 \leq maxsize \leq 0.01$). Figure 6 shows the average number of nodes visited versus ϵ with k-d query square's volume $vol = 0.001$ for $maxsize = 0.001$. An average of 40% fewer nodes are visited for the Patricia trie when $0.05 \leq \epsilon \leq 0.5$, compared to the exact range query.

5 Conclusions

We show that Patricia tries can be used to answer orthogonal range counting queries visiting $O(k \log n/\epsilon^{k-1})$ nodes for cubical range queries on uniform random k-d points. Can the result be improved closer to the lower bound of $\Omega(\log n + 1/\epsilon^{k-1})$ in fixed dimension? Experimental results show that if we allow small relative errors, the number of nodes visited for range counting can be reduced on average by 1/2 for query squares with volume ranging from 0.0001 to 0.01, $\epsilon = 0.05$, $2 \leq k \leq 10$ and $n = 1,000,000$. Range reporting queries have a less dramatic improvement when $k \leq 5$, because of the additional $O(F)$ nodes visited, which is the dominating term in the number of the nodes visited during range reporting. Another open question is how to perform combined textual and spatial data approximate range search.

References

[1] P. Agarwal. *Handbook of Discrete and Computational Geometry*, chapter Range Searching, pages 575–598. CRC Press LLC, Boca Raton, FL, 1997.

[2] S. Arya and D. Mount. Approximate range searching. *Computational Geometry: Theory and Applications*, 17:135–163, 2000.

[3] J. Bentley and J. Friedman. Data structures for range searching. *ACM Computing Surveys*, 11(4):397–409, December 1979.

[4] B. Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal of Computing*, 17(3):427–462, June 1988.

[5] V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

[6] P. Kirschenhofer and H. Prodinger. Multidimensional digital searching-alternative data structures. *Random Structures and Algorithms*, 5(1):123–134, 1994.

[7] D. Knuth. *The art of computer programming: sorting and searching*, volume 3, pages 492–512. Addison-Wesley, Reading, Mass., 2nd edition, 1998.

[8] D. Morrison. Patricia - practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 14(4):514–534, October 1968.

[9] J. Orenstein. Multidimensional tries used for associative searching. *Information Processing Letters*, 14(14):150–156, June 1982.

[10] Q. Shi and B. G. Nickerson. Approximate orthogonal range search using patricia tries. Technical report, TR05-172, Faculty of Computer Science, University of New Brunswick, April 2005, 30 pages.

[11] Q. Shi and B. G. Nickerson. k-d range search with binary patricia tries. Technical report, TR04-168, Faculty of Computer Science, University of New Brunswick, December 2004, 35 pages.

[12] W. Szpankowski. Patricia tries again revisited. *Journal of the ACM*, 37(4):691–711, 1990.