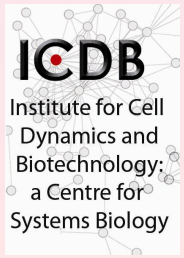
Analyzing Metric Space Indexes: What For?

Gonzalo Navarro

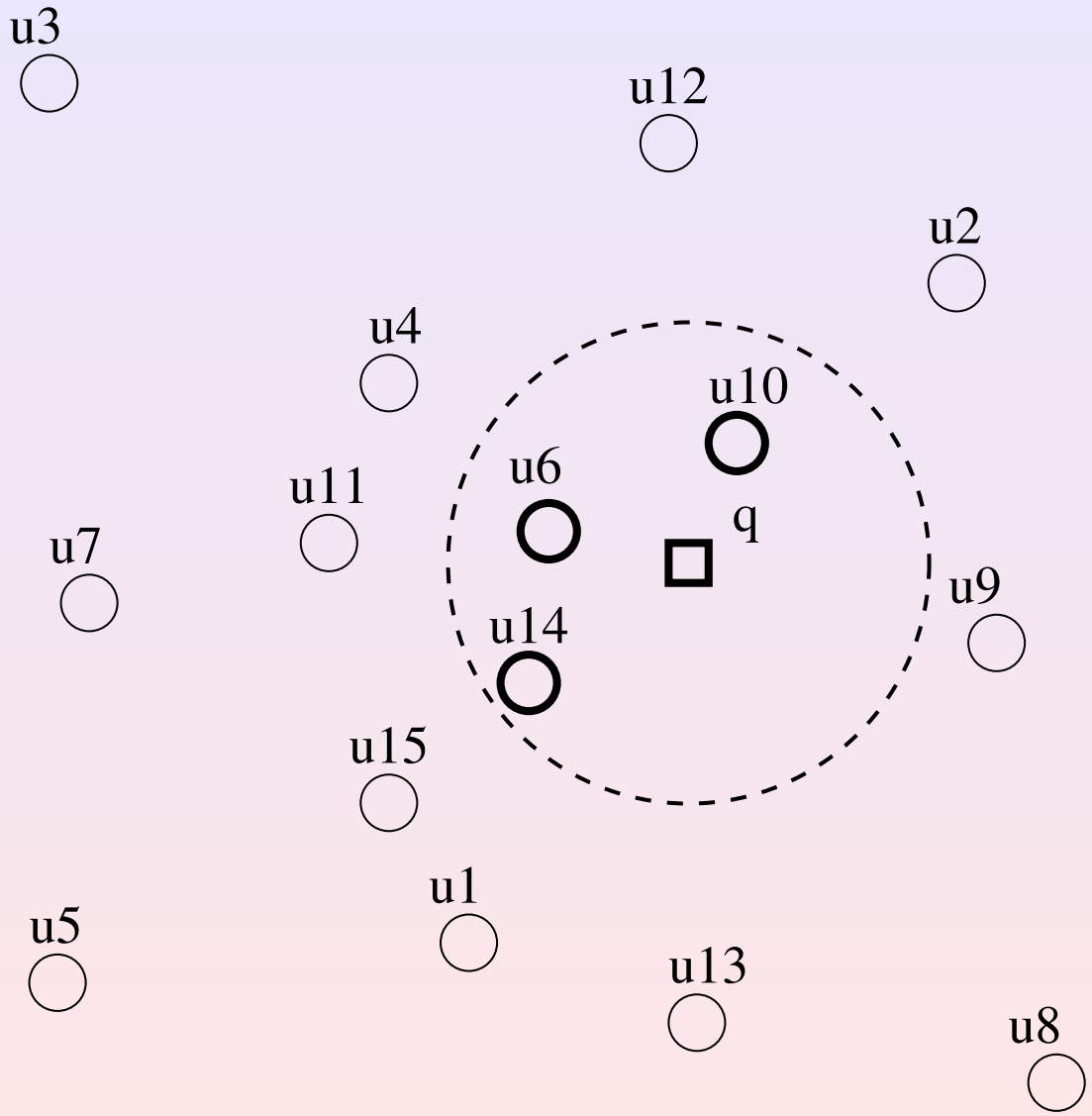
Departament of Computer Science
University of Chile
www.dcc.uchile.cl/gnavarro
gnavarro@dcc.uchile.cl



Metric Space Searching

- A universe X of black-box objects.
- A finite database $U \subseteq X$.
- A distance function $d : X \times X \to \mathcal{R}^+$.
- It respects the triangle inequality.
- Range queries (q, r), $q \in X$, $r \in \mathcal{R}^+$.
- Other queries (nearest neighbor, etc.)
- Wish to minimize the number of evaluations of d.

Metric Space Searching



What attracted me first to metric space searching?

- A paper by Ricardo Baeza-Yates et al., on Fixed Queries Trees (FQT), 1994.
- It was a tree where the search had to enter several branches.
- The analysis of the search cost was very different from the usual ones on trees (always ending up in $O(\log n)$).
- The recurrence was like this:

$$T(n) = \sum_{1 \le i \le k} q_i \, T(p_i n), \quad T(1) = 1$$

- The solution was an intriguing $T(n) = n^\alpha$...
- where α is the solution to

$$\sum_{1 \le i \le k} q_i p_i^\alpha = 1.$$

What attracted me first to metric space searching?

- Clearly, this was a different beast, defying usual assumptions on balancing, arity, etc.
- I started to work on this field, expecting to be Wonderland for an algorithmicist.
- I soon became disappointed.
- Let us see why...

In the beginning...

- Researchers tried to apply asymptotic analysis, as in any algorithmic discipline.
- Several nice results were obtained.
- But... they were difficult to verify experimentally.
- Unlike in other areas of algorithmics, reality refused to behave as it should.

What happened?

- The specificities of metric spaces were too important to ignore.
- For example the dimensionality or probabilistic distribution of the space.
- But these were either too difficult to model, or there was no consensus.

Instead...

- Researchers gave up on this line of formal analysis.
- They either resorted to pure experimentation...
- ... or to complex cost models.
- The latter were successful predicting how a particular index would perform.
- But not for understanding why the index performed in that way.

In addition...

- There were no established testbeds and datasets.
- So each researcher used his/her favorite examples.
- While the discipline has been evolving in finding practically efficient indexes...
- it has been involving in the theoretical aspect: you have an idea, you try it. It works on some dataset? Bingo!
- Why it works? Will it work elsewhere? How it scales?
- I have not seen many introspective analyses of that kind for several years.

Asymptotic analysis

- This was more or less the state of Computer Science before asymptotic analysis was invented.
- My thesis is that asymptotic analysis can give us important insights even if it is useless for precise predictions.
- In algorithmics, it is used along the design process, to understand what is going on and in which direction should we aim.
- I believe it can be used for metric indexes as well:
  - To understand why an index behaves in some way.
  - To guide the direction of optimizations and tuning.
  - To design new indexes.

For example...

- Once upon a time, Edgar Chávez and I realized that Vantage Point trees (VPTs), which were binary balanced trees, could be analyzed as a particular case of the equation I showed before.
- The solution was $O(n^\alpha)$, for

$$q_1 p^\alpha + q_2 (1-p)^\alpha = 1$$

- Here, q_1 and q_2 are the probabilities that the query enters into the first and second child, and add up more than 1.
- If $p = 1/2$ as in the VPT, you get $\alpha = \log_2(q_1 + q_2)$.
- However, if the distribution of distances is concentrated, the best p is far from $1/2$.
- This led us to design the List of Clusters (LCs), a simple and extremely efficient index for high-dimensional spaces.

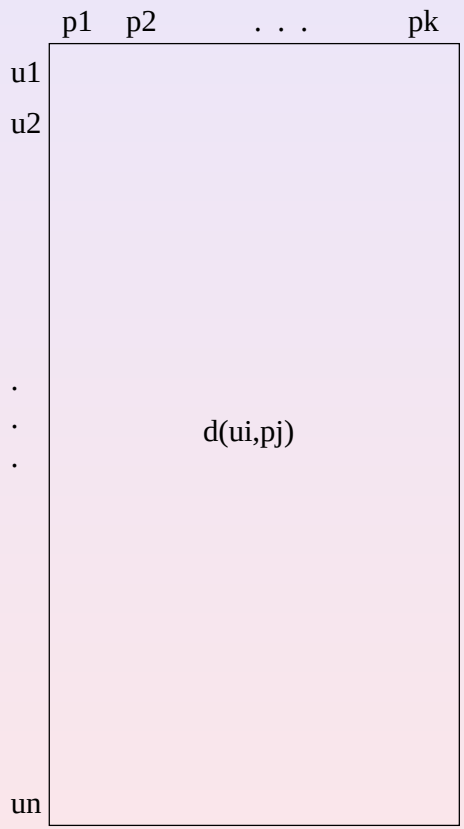
In this talk...

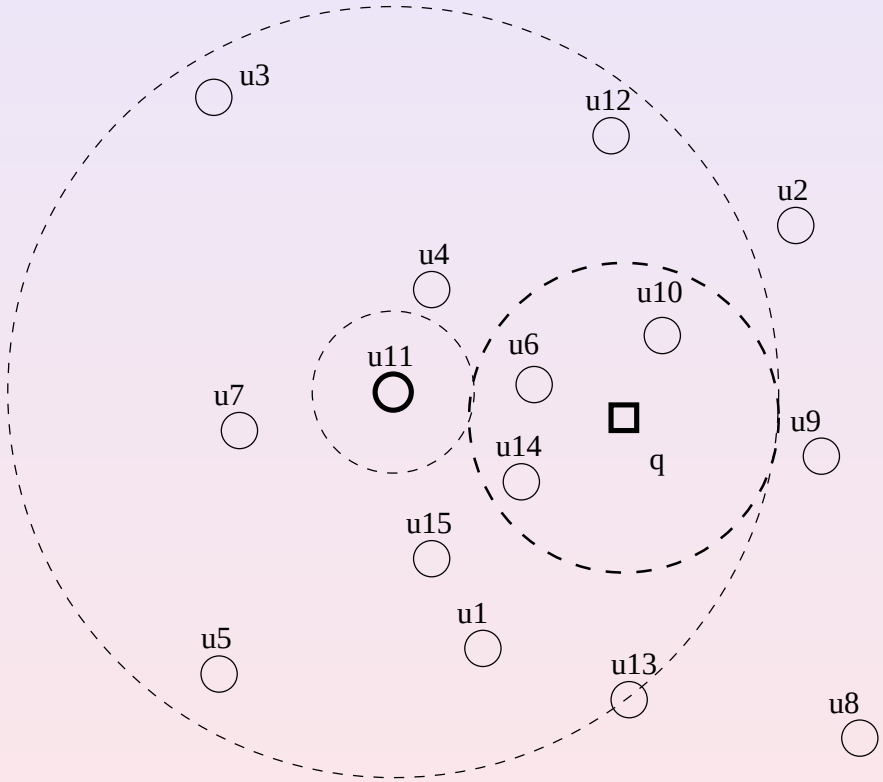
- I will review what is known about asymptotic analysis of the most common families of metric indexes.
- I will also present new results, that explains some facts experimentally verified but not totally understood.
- I will show how asymptotic analysis, even under gross simplifications (that render it useless for predicting), helps explaining behaviors that are empirically well known.
- I will argue that we need to retain some kind of introspective analysis, or our discipline will become a bunch of heuristics.

Pivot Tables: Structure

- Choose k "pivots" $\{c_1, c_2, \ldots, c_k\}$.
- Store an $n \times k$ table $d[u, c_i] = d(u, c_i)$ for all u, c_i.
- At search time:
  1. Compute every $d(q, c_i)$.
  2. If $d(q, c_i) \leq r$, output c_i.
  3. If $|d(q, c_i) - d[u, c_i]| > r$ for some i, discard u.
  4. Compare q directly with each not discarded u.

Pivot Tables: Structure

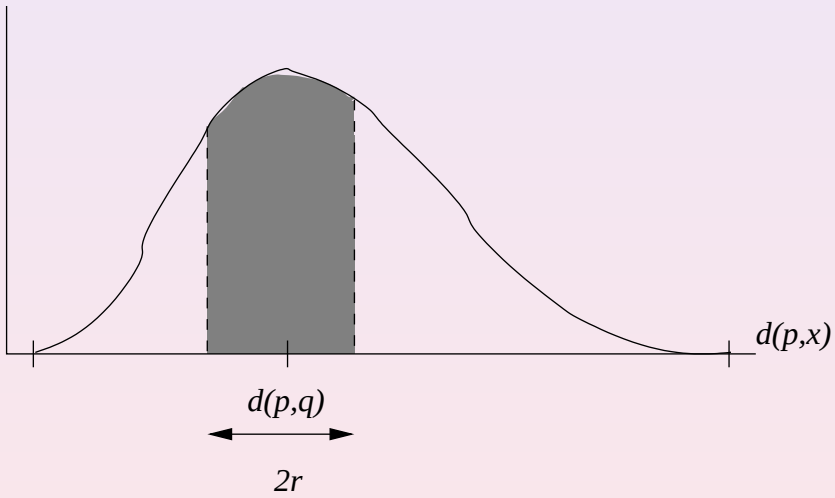


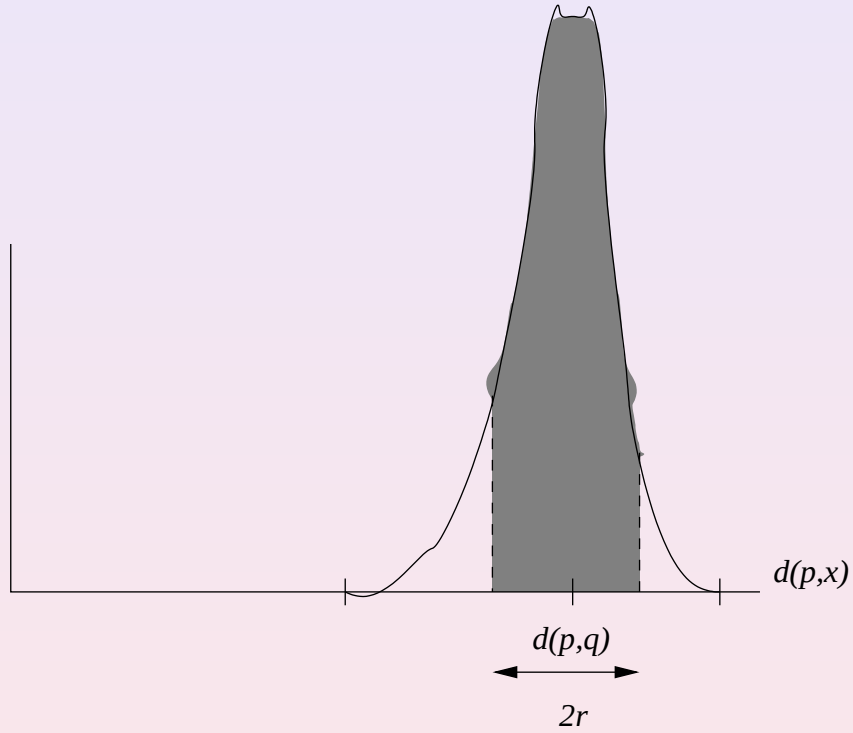


Pivot Tables: Analysis

- $f(x)$: histogram of distances of the metric space X.
- We assume points of U and queries q are chosen randomly from X.
- $f(x)$ is also the histogram of c_i if chosen at random from U.
- $F(x)$: accumulated histogram of $f(x)$.
- For random q, the probability of $d(q, c_i) = x$ is $f(x)$.
- Given that x, the probability that c_i does not discard a random $u \in U$ is that of $d(u, c_i) \in [x - r, x + r]$.
- That is, $F(x + r) - F(x - r^-)$.

Pivot Tables: Analysis





Pivot Tables: Analysis

- Then the probability that a random $q \in X$ does not discard u is

$$e = \int_0^{+\infty} f(x) \cdot (F(x+r) - F(x-r^-))\, dx$$

  where $0 \leq e \leq 1$.
- The larger r or the more concentrated $f(x)$, the closer e to 1.

Pivot Tables: Analysis

- Then the probability that u is not discarded is e^k.
- And the expected cost of the search is

$$T(n) = k + ne^k.$$

- One can achieve logarithmic search time using sufficiently many pivots:

$$k^* = \log_{1/e} n + \log_{1/e} \ln(1/e).$$

- Then the search time becomes

$$T(n) = \log_{1/e} n + \frac{1}{\ln(1/e)} = O(\log n);$$

where the constant worsens as e grows.

Pivot Tables: Analysis

- Still the $O(\log n)$-type behavior does not depend on r nor the space.
- Yet, one might have memory only for $k = \beta \cdot \log n < k^*$ pivots.
- In this case the growth rate changes drastically:

$$T(n) = \beta \log n + n e^{\beta \ln n} = O\left(n^{1-\beta \ln(1/e)}\right),$$

which is of the form n^α for some $0 \leq \alpha \leq 1$ that depends on $f(x)$, on r, and on the available memory.

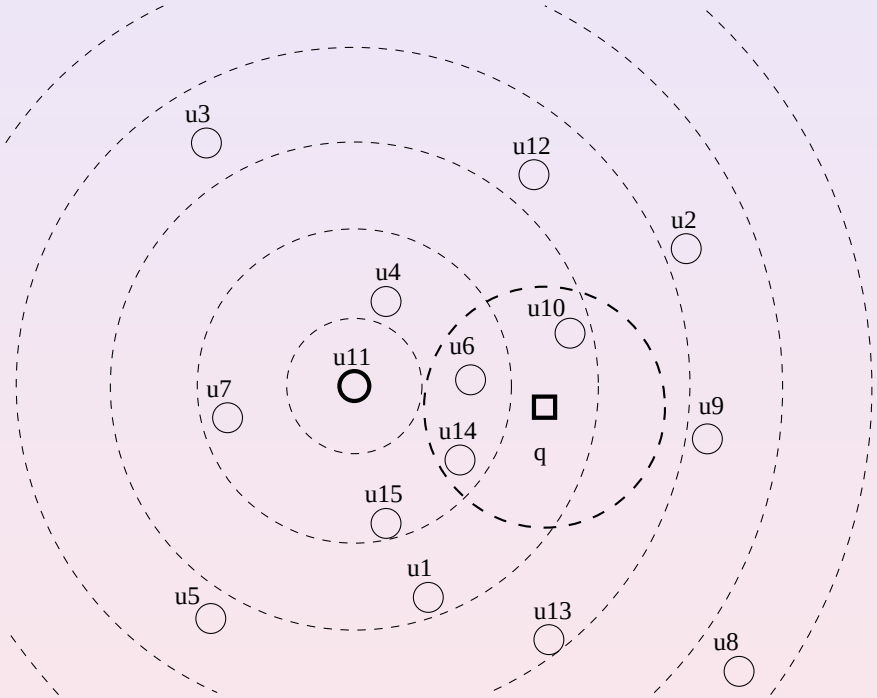
Pivot Tables: Lessons

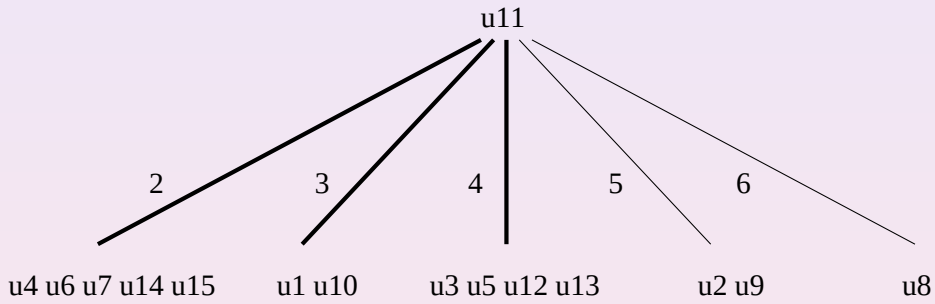
- We can achieve logarithmic search time.
- Using the right number of pivots makes a great difference.
- In many cases one does not store sufficient pivots.
- This may explain the observed results: Pivots are not so different from other intrinsically $O(n^\alpha)$-time schemes.
- On the other hand, there are analytical and empirical results about achieving $O(1)$ time by properly choosing the pivots (Faragó, AESA).
- This stresses the importance of choosing pivots correctly (e.g. LAESA).

Trees for Pivots and Rings: Structure

- A tree structure where an element $c \in U$ acts as the root.
- Each children represents a range of distances to c.
- Each subtree is organized recursively.
- At search time,
  1. Compute $d(q,c)$ and report c if $d(q,c) \leq r$.
  2. Recursively enter every subtree whose range $(\ell_i, \ell_{i+1}]$ intersects $[d(q,c) - r, d(q,c) + r]$.
- This encompasses many techniches: BKTs, MTs, FQTs, VPTs, MVPTs, VPFs, LCs.

Trees for Pivots and Rings: Structure





Trees for Pivots and Rings: Analysis

- The global histogram $f(x)$ is also that of c, if c is chosen at random.
- Let us call ℓ_i the cutting distances.
- We call $p_i = F(\ell_{i+1}) - F(\ell_i)$ the probability of an element of U being inserted at the i-th subtree.
- And $q_i = F(\ell_{i+1} + r) - F(\ell_i - r) \geq p_i$ is the probability of the query entering the i-th subtree.
- Assume those p_i values remain constant within subtrees (as in MTs, VPTs, MVPTs, LCs).
- To simplify, assume further that the q_is do not change (big assumption!).

Trees for Pivots and Rings: Analysis

- The expected number of distance computations satisfies the recurrence:

$$T(n) = 1 + \sum_{1 \le i \le k} q_i \, T(p_i n), \quad T(1) = 1$$

- The solution is of the form:

$$T(n) = \frac{s n^\alpha - 1}{s - 1} = O(n^\alpha), \quad s = \sum_{1 \le i \le k} q_i,$$

- where α is the solution to

$$\sum_{1 \le i \le k} q_i p_i^\alpha = 1.$$

Trees for Pivots and Rings: Analysis

- The solution to α is unique.
- It is smaller as the q_is are closer to the p_is.
- It is $\alpha = 0$ only for $p_i = q_i$ (exact search), where the solution is actually of the familiar form $O(\log n)$.
- This illustrates how the exact vs approximate searching drastically changes the algorithmic nature of the problem.

Trees for Pivots and Rings: Analysis

FQTs

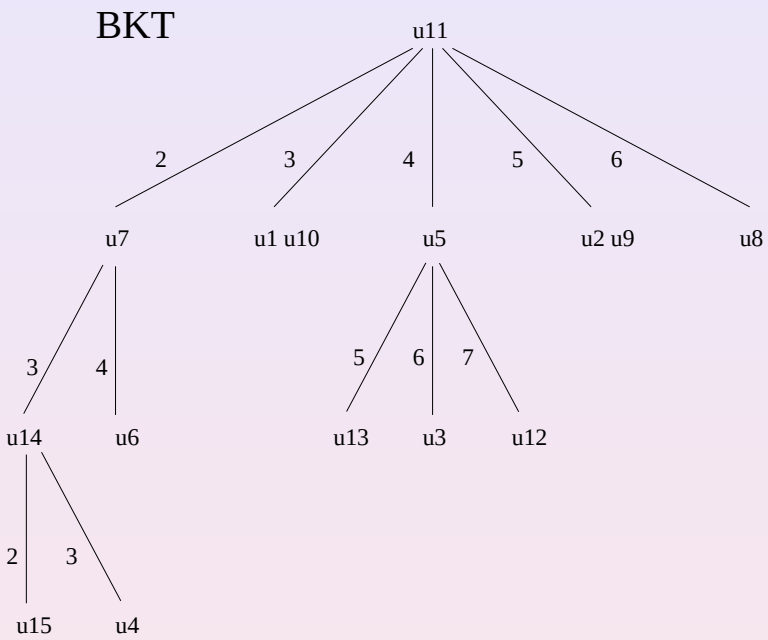
- There is only one pivot per depth, acting as the root of all those subtrees.
- The search cost is that to compare with the pivots, $O(\log_{1/p_{max}} n)$...
- ... plus a recurrence similar to the previous one, giving exactly n^α, same α.
- Thus, same analytic structure, better constants.
- Confirmed by experiments.

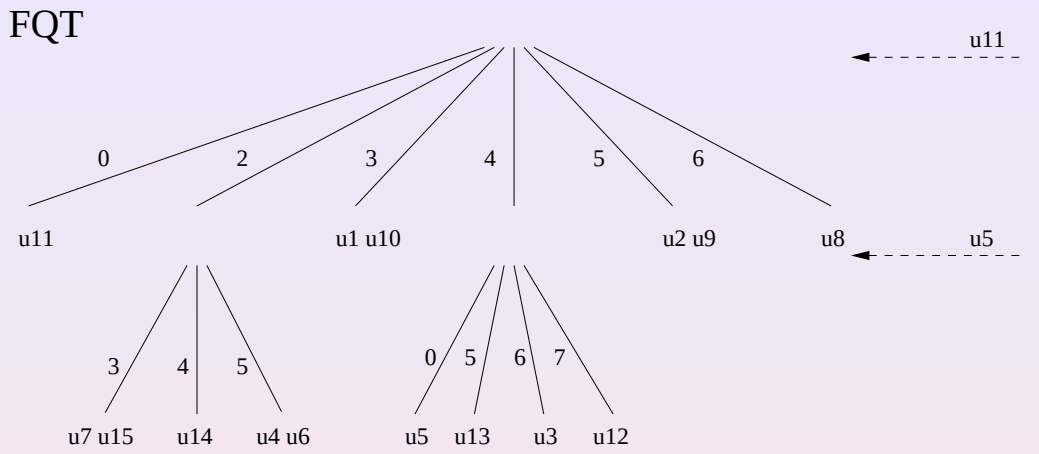
Trees for Pivots and Rings: Analysis

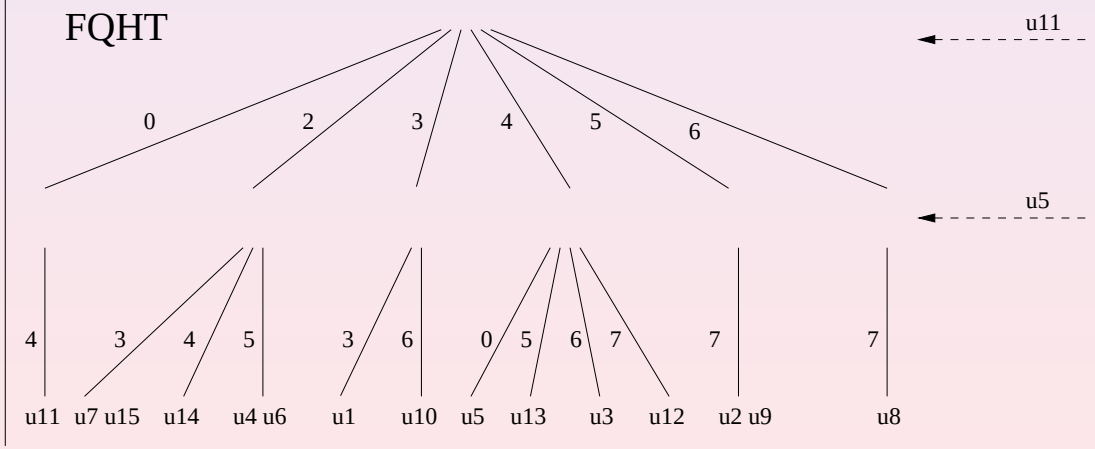
FQTs

- FQTs are so similar to pivots... why the $O(\log n)$ behavior does not show up?
- Because leaves are not compared to further pivots, but directly with q.
- FHQTs: all leaves are forced to the same depth h, even if following a unary path.
- They do behave just like pivots.

Trees for Pivots and Rings: Analysis







Trees for Pivots and Rings: Analysis

MVPTs

- They use t pivots per node and use all ranges to branch.
- The recurrence is of the form

$$T(n) = t + \sum_{1 \le i_1 \ldots i_t \le k} q_{i_1} \ldots q_{i_t} \; T(p_{i_1} \ldots p_{i_t} n)$$

- and its solution is

$$T(n) = ((t + s^t - 1) n^\alpha - t)/(s^t - 1)$$

- This is of the same order as before (same α) but the constant is $1 + t/(s^t - 1)$.
- The constant improves as t grows, suggesting maximum t.
- Partially confirmed by experiments.

Trees for Pivots and Rings: Lessons

- FQTs, FHQTs and MVPTs show that pivot tables and trees are not so different.
- Trees manage to use linear space, and this limits them to $O(n^\alpha)$-like behavior.
- Pivot tables can use $O(n \log n)$ space, at which point they can achieve $O(\log n)$-time queries.
- In general, using trees is justified only if one has not enough memory.

Trees for Pivots and Rings: Analysis & Lessons

What arity should we use?

- Assume for simplicity all $p_i = p = 1/k$.
- Assume also a flat histogram $q_i = q = p + \delta$.
- Then $s = qk$ and $\alpha = \log_k(1 + \delta k)$.
- The best α as a function of δ is achieved for higher k as r grows, yielding anyway larger α values.
- This recommends using a higher arity as the search becomes more difficult.
- Intuitively, this gives a chance of discarding at least some subtrees that do not intersect the query ball.
- Partial experiments confirm, again, this intuition.

Trees for Pivots and Rings: Analysis

Should we balance the trees?

- Assume we fix arity k.
- Assume for now a flat histogram $q_i = p_i + \delta$.
- We seek the best partitioning p_i, minimizing the α that solves $\sum p_i^\alpha (p_i + \delta) = 1$.
- If we set all $p_i = p = 1/k$, the equation becomes

$$kp^\alpha(p + \delta) = k^{1-\alpha}(1/k + \delta) = 1$$

- with solution $\alpha = \log_k(1 + \delta k)$, as we have seen.

Trees for Pivots and Rings: Lessons

Should we balance the trees?

- If we perturb just two p_i values, by ϵ, the new equation is

$$(k-2)p^\alpha(p+\delta)+(p+\epsilon)^\alpha(p+\epsilon+\delta)+(p-\epsilon)^\alpha(p-\epsilon+\delta)=1$$

- Implicit differentiation reveals that α increases as ϵ deviates from zero.
- Thus the balanced partition is the best for a flat histogram, i.e. an easy space.

Trees for Pivots and Rings: Analysis

Should we balance the trees?

- Assume now that we are in dimension D, that is, $F(x) = x^D$, $x \in [0, 1]$.
- To simplify, assume $k = 2$.
- We want the x such that $p = F(x)$ minimizes the α that solves

$$F(x)^\alpha F(x + r) + (1 - F(x))^\alpha (1 - F(x - r)) = 1$$

- We use implicit differentiation to obtain the optimum α as a function of x and r, yet there is no closed-form expression.

Trees for Pivots and Rings: Lessons

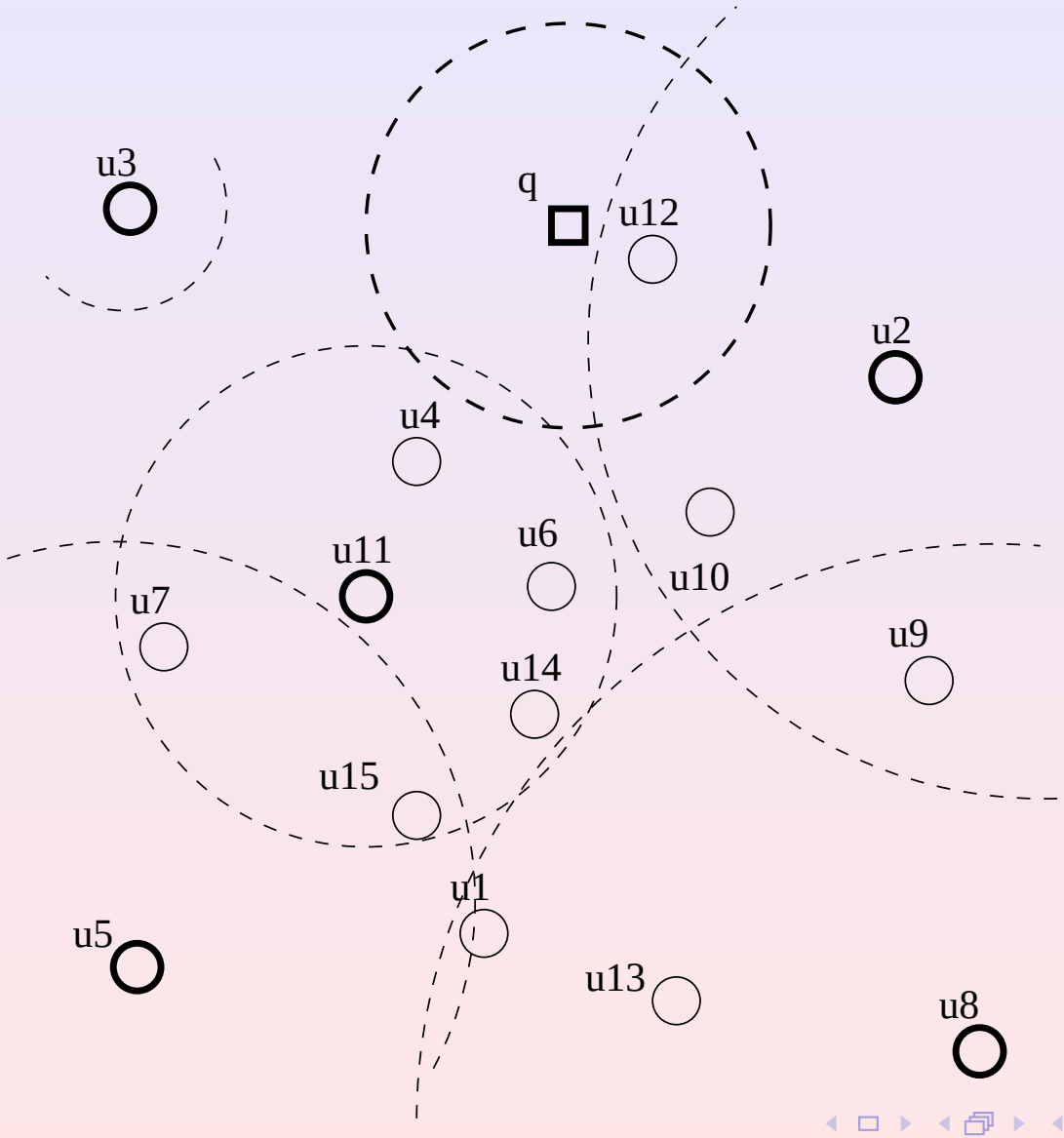
Should we balance the trees?

- Still we can handle it numerically to gain intuition.
- Already low dimensions such as $D = 2$, for retrieving 0.01% of the data set, recommend leaving $p_1 = 0.24$ of the histogram to the left and $p_2 = 0.76$ to the right, so as to obtain $\alpha = 0.035$.
- As we let D or r grow, the analysis recommends more unbalanced partitions to obtain the optimum α, which is nevertheless higher.
- Widely backed by experiments
- Intuition: we have many many pivots per element!

Trees for Centers and Clusters: Structure

- They choose k centers $c_1 \ldots c_k$, with their subtrees $T_1 \ldots T_k$.
- Other elements are inserted into those subtrees with some criterion.
- They try that each c_i is spatially compact.
- They compute covering radii $cr_i = \max_{u \in T_i} d(c_i, u)$.
- The construction proceeds recursively inside each T_i.
- Queries q are compared to each c_i, which is reported if $d(q, c_i) \leq r$.
- Then we enter each subtree where $d(q, c_i) \leq cr_i + r$.
- BSTs, M-trees, Antipole Trees, LCs.

Trees for Centers and Clusters: Structure



Trees for Centers and Clusters: Analysis

- Assume we choose centers at random.
- Thus $p_i = F(cr_i)$ is the probability of a random element falling within the area covered by T_i.
- And $q_i = F(cr_i + r)$, that of a query q having to enter T_i.
- If the cluster balls cover the space, it must hold $\sum_{1 \leq i \leq k} p_i \geq 1$.
- Equality is desirable, holding only if the balls are disjoint.
- This is usually impossible, but can be approached by a good clustering.

Trees for Centers and Clusters: Analysis

- Assume we achieve roughly the same probability mass for clusters, $p_i = p$, $k \geq 1/p$.
- Assume this arity and relative mass is maintained in subtrees.
- The probability mass of a cluster at depth h is p^h.
- If elements are inserted at random inside any suitable cluster, the expected leaf depth is $h^* = \log_k n$.
- At depth $h \leq \log_k n$ there are k^h clusters.
- The probability of the query ball intersecting a cluster at depth h is $F(F^{-1}(p^h) + r)$.
- The overall query cost is

$$T(n) = \sum_{h=1}^{h^*} k^h F(F^{-1}(p^{h-1}) + r)$$

Trees for Centers and Clusters: Analysis

- To gain intuition, assume a flat histogram in $[0, 1]$, $F(x) = x$.
- Thus $F(F^{-1}(p^{h-1}) + r) = p^{h-1} + r$, and

$$T(n) = \sum_{h=1}^{h^*} k^h (p^{h-1} + r)$$

- If $kp = 1$ (perfect clustering, very unlikely), the solution is

$$T(n) = k \log_k n + \frac{k}{k-1}(n-1)r$$

- If $kp > 1$, the solution is

$$T(n) \leq \frac{k}{kp-1} n^{1-\log_k(1/p)} + \frac{k}{k-1} r n$$

- Again, of the form $O(n^\alpha)$.

Trees for Centers and Clusters: Lessons

- A good clustering is essential: α increases with kp.
- Indeed, a perfect clustering achieves logarithmic time.
- Increasing k is good when the search is difficult: it reduces the overhead associated to nr, which is dominant in this case.
- Decreasing k is good when the search is easy: it reduces the term that multiplies $O(n^{\alpha})$.
- Similar conclusions can be derived in dimension D.
- Clearly validated experimentally, e.g. LCs.

Trees for Centers and Clusters: Analysis

What about balancing?

- Consider now that we have different probabilities $p_1 + p_2 + \ldots + p_k > 1$.
- At level h we have

$$\sum_{1 \le i_1, i_2, \ldots, i_h \le k} F(F^{-1}(p_{i_1} p_{i_2} \ldots p_{i_h}) + r)$$

- Assuming again $F(x) = x$, this simplifies to

$$(p_1 + p_2 + \ldots + p_k)^h + rk^h$$

- This is essentially the same as before, except that now the depth of the tree is $h^* = \log_{p_{max}} n$.

Trees for Centers and Clusters: Lessons

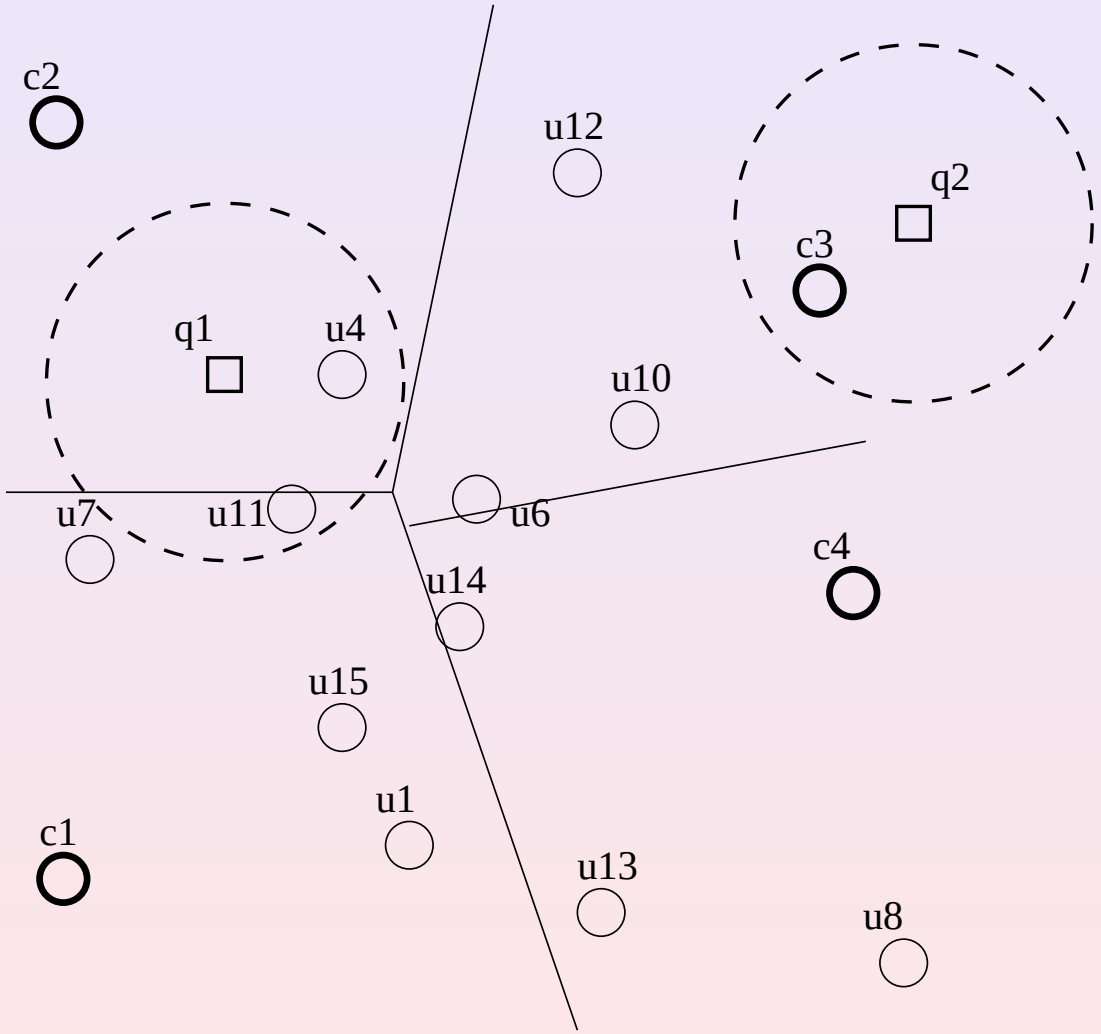
What about balancing?

- At levels $\log_k n < h \leq h^*$, there are n clusters.
- Doing the summation, we obtain the same terms as before, plus some extra terms in this area.
- This shows that it is good idea to have clusters of similar size, that is, a balanced partition.
- Having such a partition is, e.g., a design goal of the M-tree, now analytically justified.
- Note that, on trees for pivots, unbalancing was the answer to high dimensions; here the answer is higher arity.

Trees for Hyperplanes and Voronoi Regions: Structure

- They choose k centers $c_1 \ldots c_k$, with their subtrees $T_1 \ldots T_k$.
- Other elements are inserted into the subtree of their closest center.
- The construction proceeds recursively inside each T_i.
- Queries q are compared to each c_i, which is reported if $d(q, c_i) \leq r$.
- Let c_i be the closest to q.
- Then we enter each subtree j where $d(q, c_j) - d(q, c_i) \leq 2r$.
- VTs, GHTs, GNATs, SATs.
- Note regions do not overlap.

Trees for Centers and Clusters: Structure



Trees for Hyperplanes and Voronoi Regions: Analysis

- Let $D_i = d(q, c_i)$.
- Let $G_i = D_i - \min(D_1, \ldots, D_k)$.
- Let $g_i(x)$ be the histogram of G_i.
- Let $G_i(x)$ be the cumulative histogram of $g_i(x)$.
- The probability g_i of entering T_i is that of $G_i \leq 2r$, that is, $g_i = G_i(2r)$.
- The recurrence for the search cost is

$$T(n) = k + \sum_{1 \leq i \leq k} p_i \left(T(p_i n) + \sum_{j \neq i} g_j \, T(p_j n) \right)$$

- This is

$$T(n) = k + \sum (p_i + g_i(1 - p_i)) \, T(p_i n)$$

where p_i is the probability of being closer to c_i than to any other center.

Trees for Hyperplanes and Voronoi Regions: Analysis

- We assumed that the g_is stay the same within the clusters.
- The solution to the recurrence is

$$T(n) = \frac{(k+s-1)n^\alpha - k}{s-1}, \quad s = 1 + \sum_{1 \le i \le k} g_i(1-p_i),$$

  where α is the solution of

$$\sum_{1 \le i \le k} (p_i + g_i(1-p_i))\, p_i^\alpha = 1$$

- This has a unique solution, that improves for smaller g_i's.

Trees for Hyperplanes and Voronoi Regions: Lessons

Which is the best arity?

- Assume the partitions are similar in mass, $p_i = 1/k$.
- Then $\alpha = \log_k s$.
- Assume a relatively flat histogram, $g_i = p_i + \delta$, so $s = 1 + k(p + \delta)(1 - p) = 2 - 1/k + (k - 1)\delta$.
- The best α as a function of k has no closed form expression.
- Numerically, one can see that the best k grows with δ (i.e., with the search difficulty).
- A subtler effect is that δ decreases as k increases (since $G_i(2r)$ decreases), but only up to some point since $G_i \geq F_i$.
- Experimentally verified on GNATs vs GHTs, and in SATs.

Trees for Hyperplanes and Voronoi Regions: Lessons

Balancing or unbalancing?

- These trees are not naturally unbalanced, but balancing is difficult to ensure.
- Assume all $p_i = p = 1/k$ except for two, $p + \epsilon$ and $p - \epsilon$, associated with g, g^+ and g^-.
- Now the formula that defines α is

$$\ldots+(p+\epsilon)^\alpha((p+\epsilon)(1-g^+)+g^+)+(p-\epsilon)^\alpha((p-\epsilon)(1-g^-)+g^-) = 1$$

- Since $(p(1-g)+g)$ is a convex combination between p and 1 with weight g, the sum is convex on ϵ.
- Thus increasing ϵ increases the sum and forces α to raise.
- Hence the best is to have balanced partitions.

Conclusions

- Asymptotic analysis is, admittedly, hopeless for predicting performance of metric space indexes.
- It has been mostly abandoned after a few initial attempts.
- I have argued, however, that it has a great potential to aid in understanding performance, designing new indexes, and guiding the direction of optimizations.
- I have shown how asymptotic analysis explains the known general behavior of existing indexes.
- And also how it has helped us design extremely successful indexes.

What is missing?

- Most indexes have an $O(n^{\alpha})$-type behavior.
- Analysis tells us what to expect if we move parameters within an indexing scheme.
- We do not have clues to compare across indexes.
- This would be a great success for asymptotic analysis.
- Furthermore, we should combine it better with the characteristics of the space.
- Probabilistic methods and nearest neighbor searches are likely to be amenable of (probably cleaner!) asymptotic analysis as well, and maybe it would help in their understanding and design of improved methods.

Thanks!