Ultra-Fast Similarity Search Using Ternary Content Addressable Memory

Yotam Harchol

The Hebrew University of Jerusalem, Israel

Joint work with:

Anat Bremler-Barr
IDC Herzliya, Israel

David Hay
Hebrew University, Israel

Yacov Hel-Or
IDC Herzliya, Israel

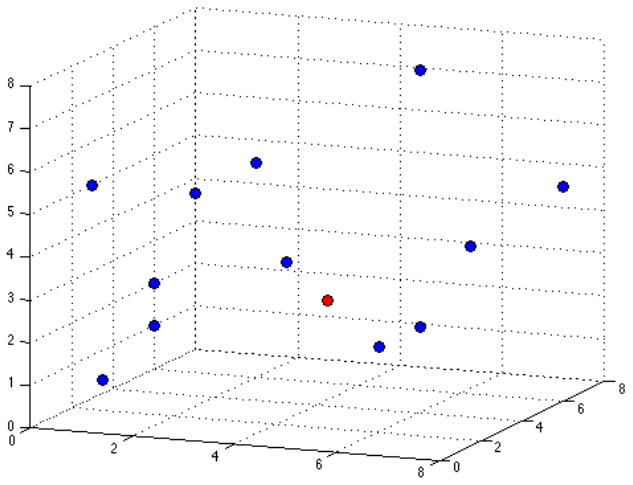
DaMoN 2015, Melbourne, VIC, Australia

This research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement no 259085.

Background

The Nearest Neighbor Problem

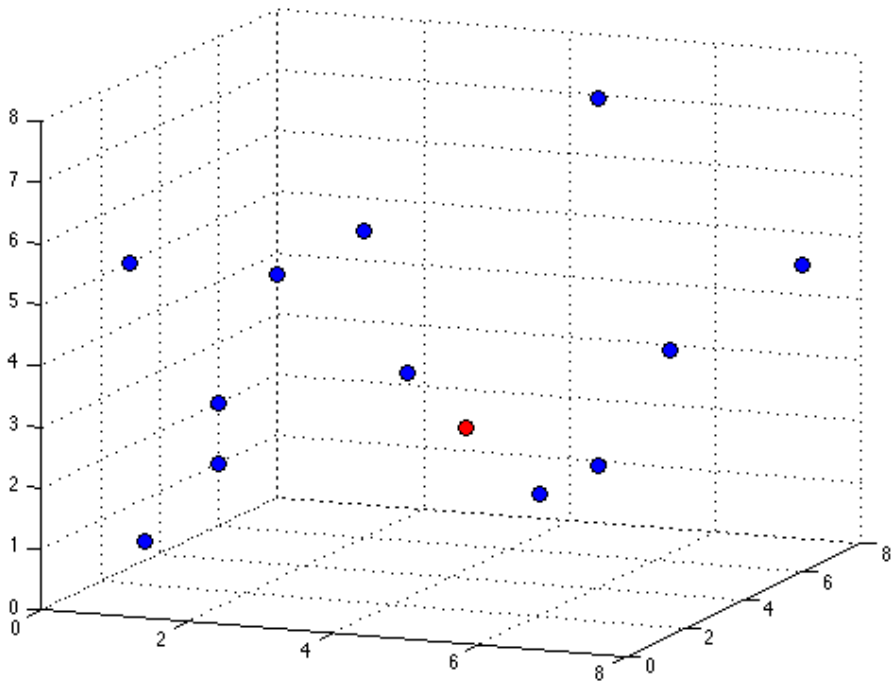
- Classic problem in computer science
  - Many applications in reality
    - Machine learning
    - Computer vision
    - Image processing
    - Spatial databases
    - Network traffic classification and QoS
  - Performance bottleneck



The Nearest Neighbor Problem

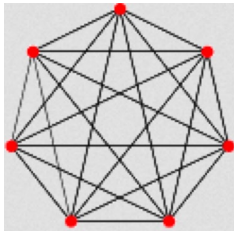
- Input:
  - A set of data points
  - A query point (or a series of those)

  (points are in a discrete space)
- Output:
  Data point closest to the query point



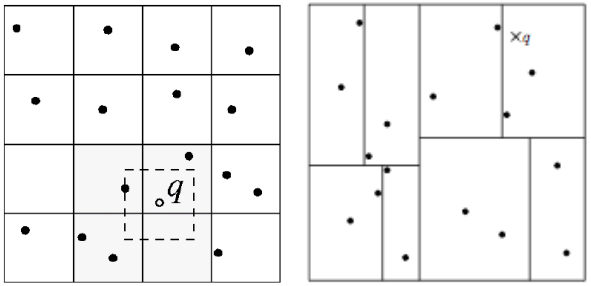
- ***The Curse of Dimensionality***: Hard problem for high dimensions (even over 10 or so)

Known Solutions



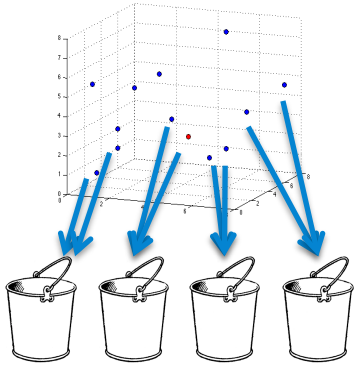
Brute force search

Too long to compute



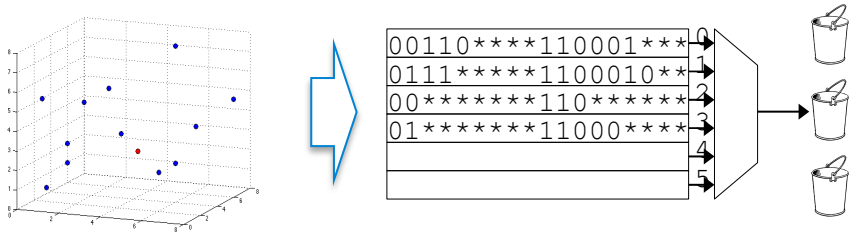
Space partitioning techniques

Time exponential in dimension



Locality sensitive hashing (LSH)

More efficient, but probabilistic
and still takes long time



Ternary locality sensitive hashing

First to show similarity search on TCAM
Probabilistic, requires more time and
space than our technique [SIGMOD '10]

Our Contribution

- ***Single TCAM lookup → Approximate solution***
  - ***Four orders of magnitude faster than LSH***
    - ***Also faster than Ternary LSH***
  - ***Deterministic***
  - ***Smaller (and practical) TCAM space required***

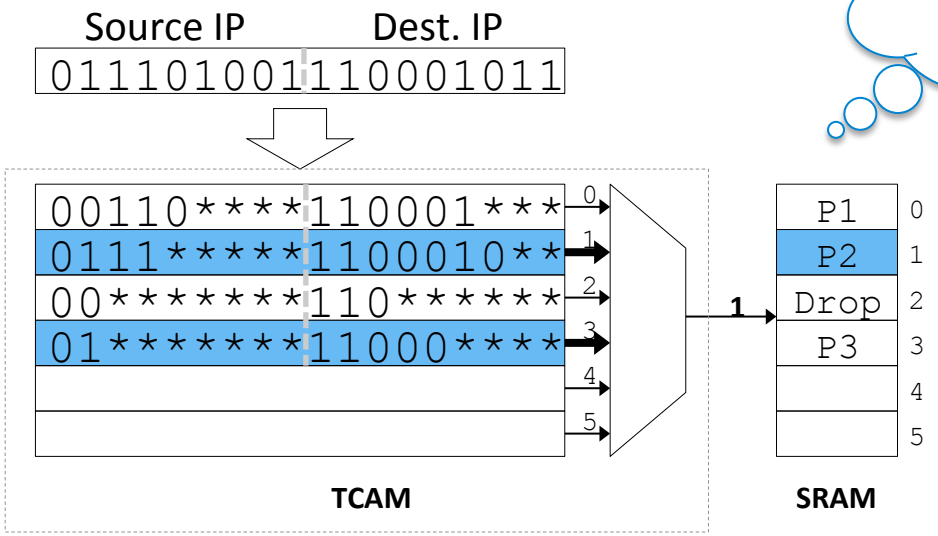
- Ultra-fast algorithms for:
  - Exact nearest neighbor search
  - k-nearest neighbors search
  - More related problems…

  (Not covered in this talk but shown in the paper)

TCAM – Ternary Content Addressable Memory

- TCAM is an associative memory module
  - Useful for parallel multidimensional prefix lookup
- Each entry may consist of 0, 1, and * bits
- Widely used in high-speed networking devices
- *360 Million* queries per second and higher!

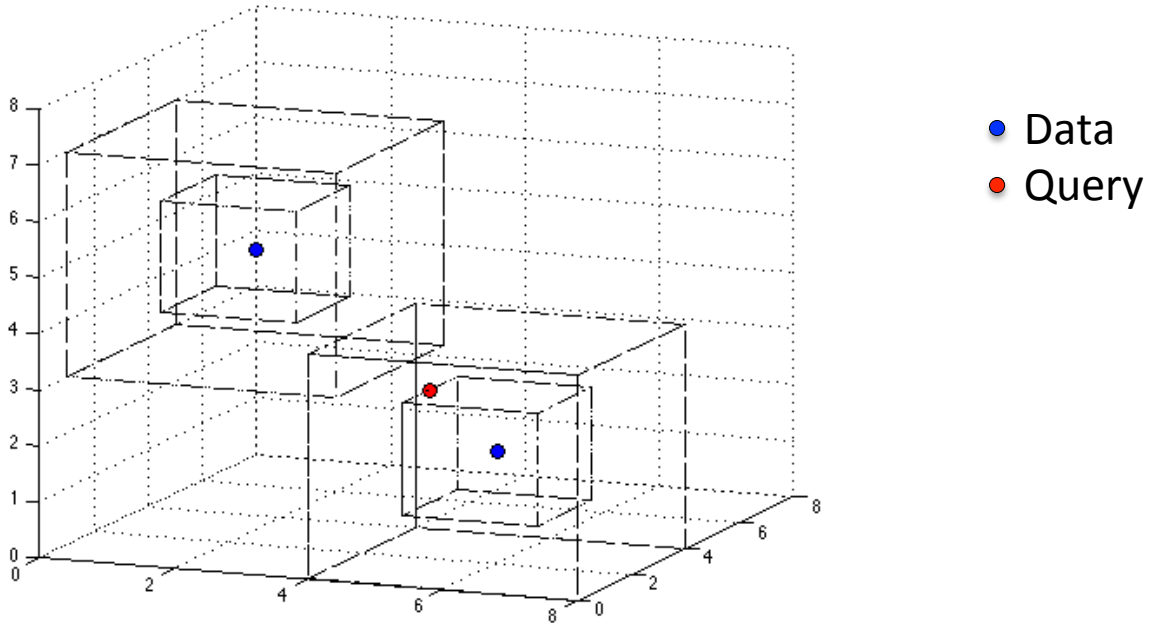
Example: 2-dimensional IP lookup:



Memory chip
or
Coprocessor?

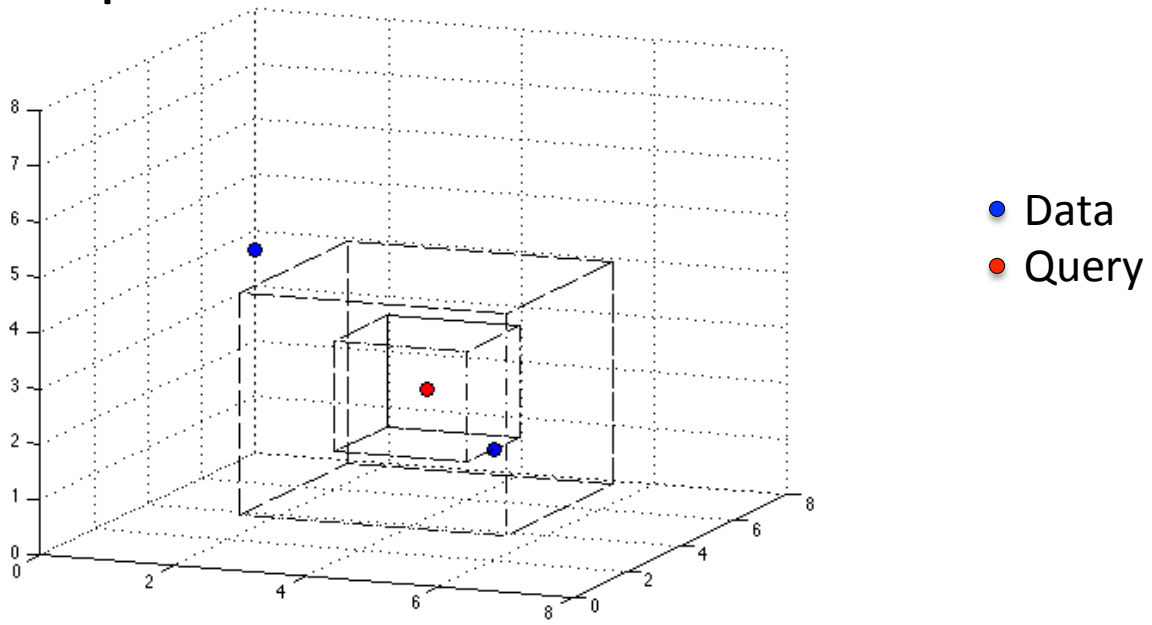
General Idea

- Encode cubes on data points
  - Given a query point, *single TCAM lookup* returns the smallest cube that contains it
  - Goal: no TCAM entry expansion in high dimension



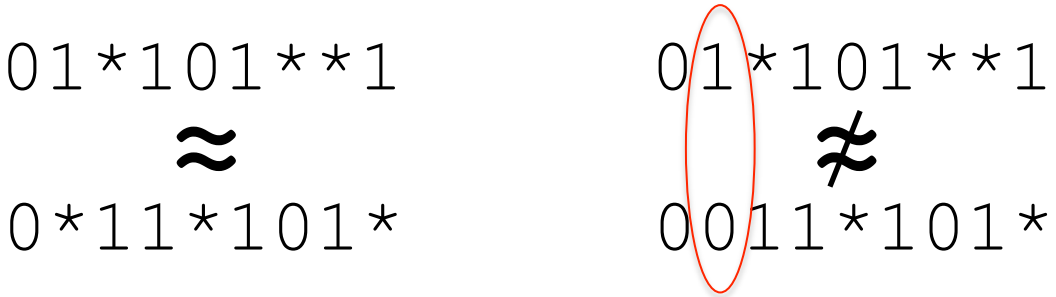
General Idea

- Or instead, save TCAM space
  - Encode cubes on the query point
  - Query TCAM with growing cubes
  - Find first data point to match



Encoding Scheme

Ternary Match

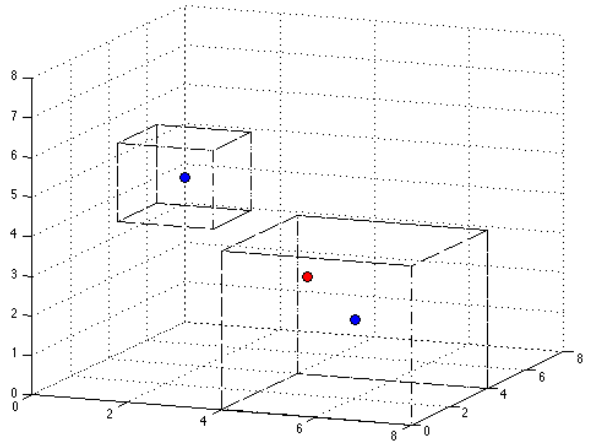


Match **No match**

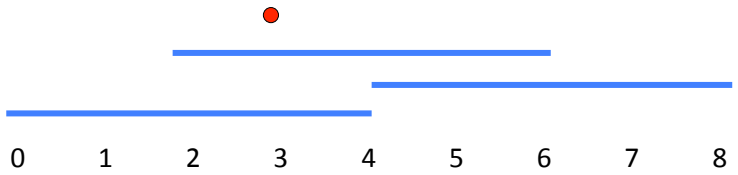
Formally, $a=\{a_1,\ldots,a_m\}$ matches the word $b=\{b_1,\ldots,b_m\}$, denoted $a \approx b$, if and only if:
For any i, $a_i=b_i$ or a_i is * or b_i is *

Encoding Function

- Goal:
  Admissible encoding function *tcode*:
  for points and cubes:
  tcode(p) ≈ *tcode*(C)
  if and only if p is in C

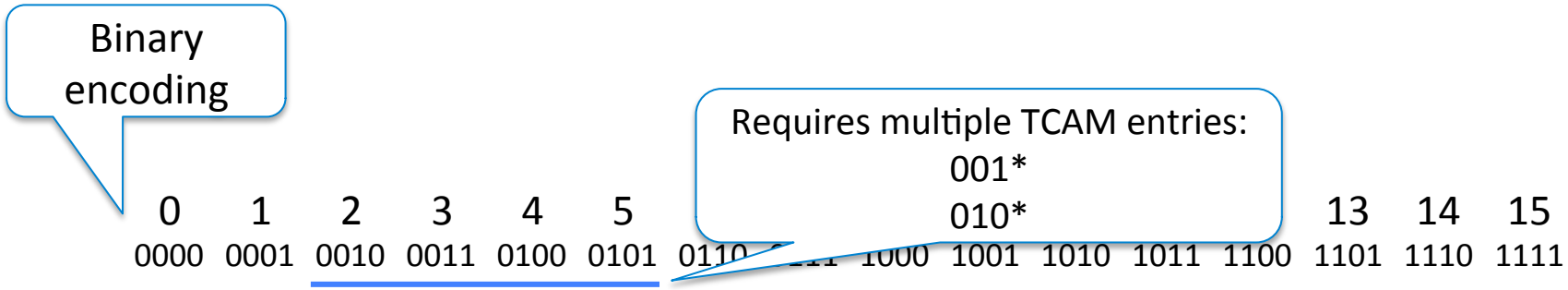


- First step:
  Solution in a single dimensional space:



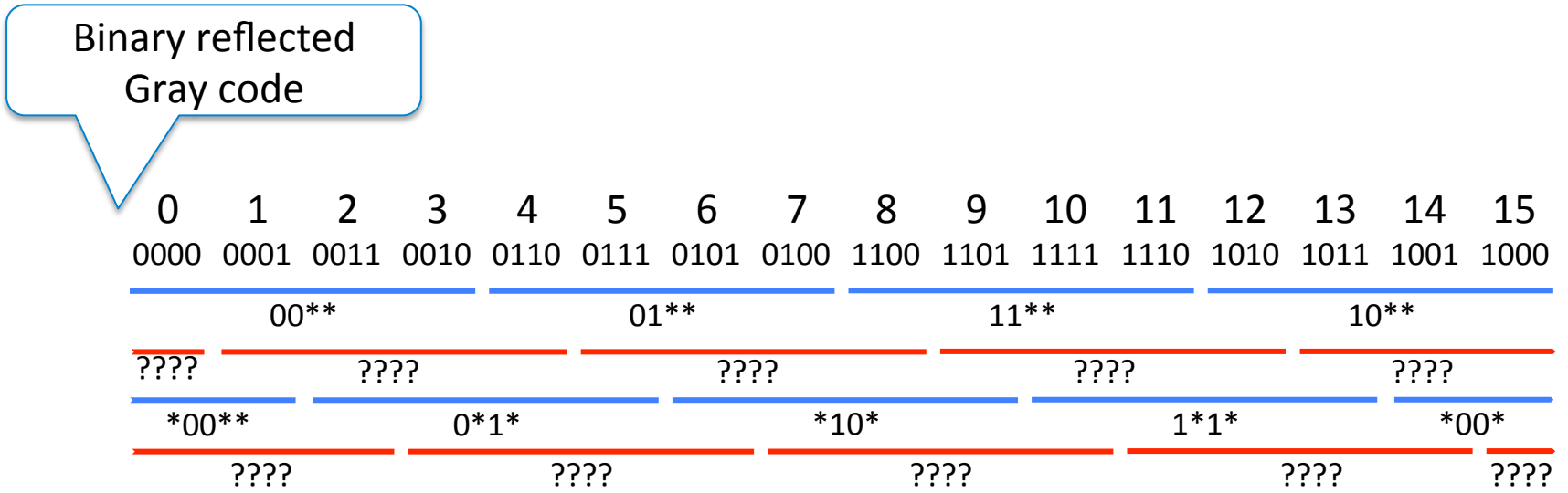
Encoding Function for Intervals

Ternary encoding function *tcode* for points and intervals:



Encoding Function for Intervals

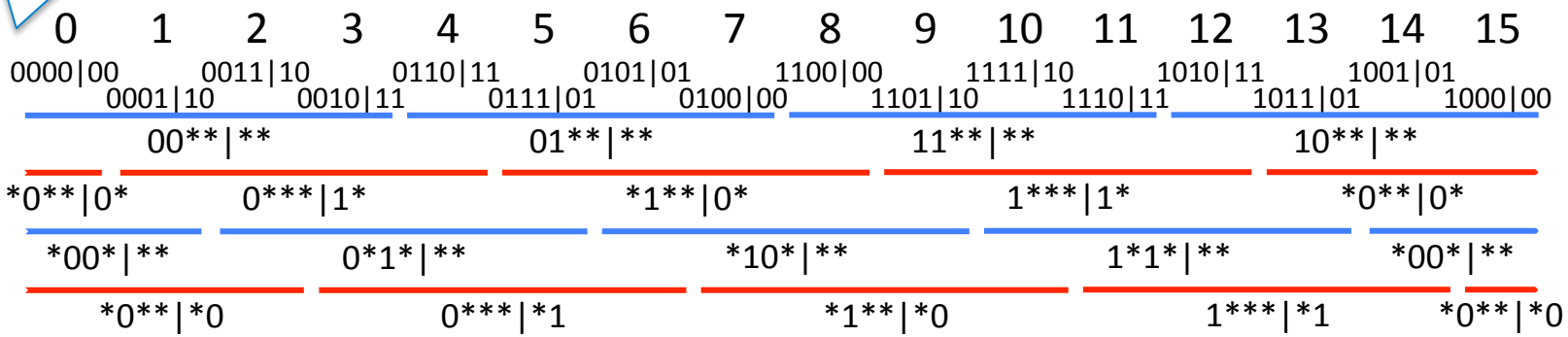
Ternary encoding function *tcode* for points and intervals:



Encoding Function for Intervals

Ternary encoding function *tcode* for points and intervals:

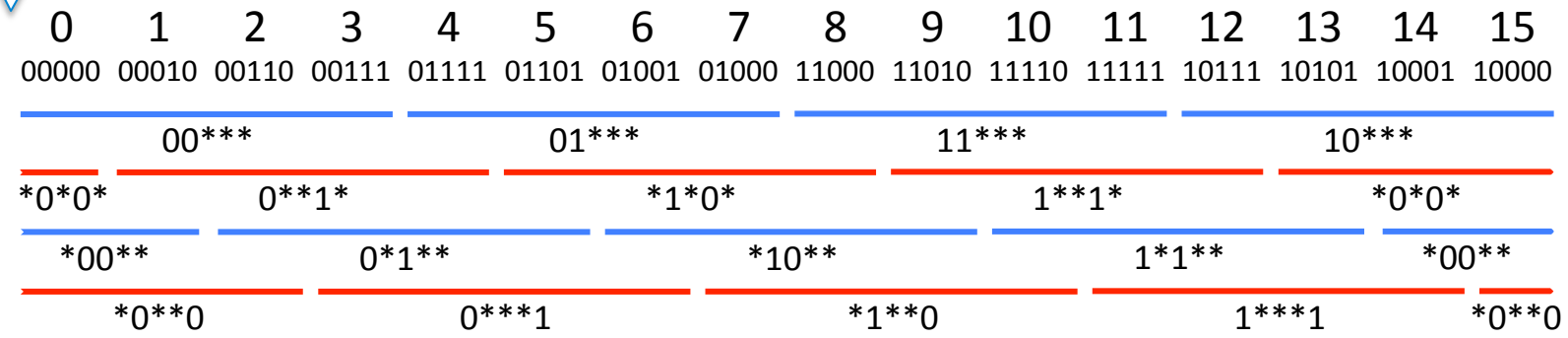
Binary reflected Gray code
+
Extra bits



Encoding Function for Intervals

Ternary encoding function *tcode* for points and intervals:

Final code
After eliminating
redundant bits



Create code for intervals of length h_{max} ➔ Encode all intervals of length ≤ h_{max}
(By intersecting two intervals of length h_{max})

* Full details and correction proofs are in the paper

Multiple Dimensions

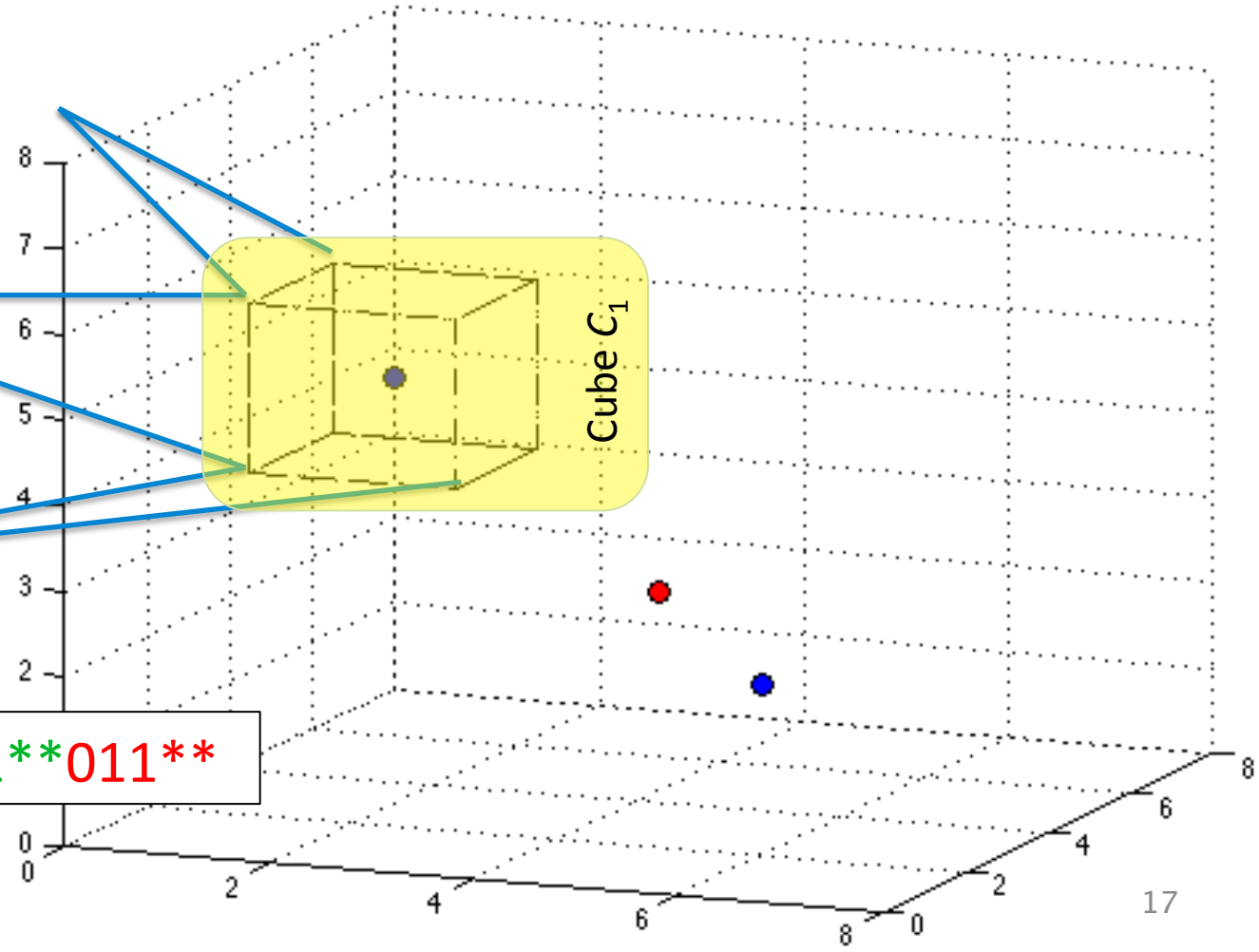
Trivial expansion to *d* dimensions:

Y=[2,4)
tcode(Y)=001**

Z=[4,6)
tcode(Z)=011**

X=[1,3)
tcode(X)=00*10

tcode(C_1)=00*10001**011**



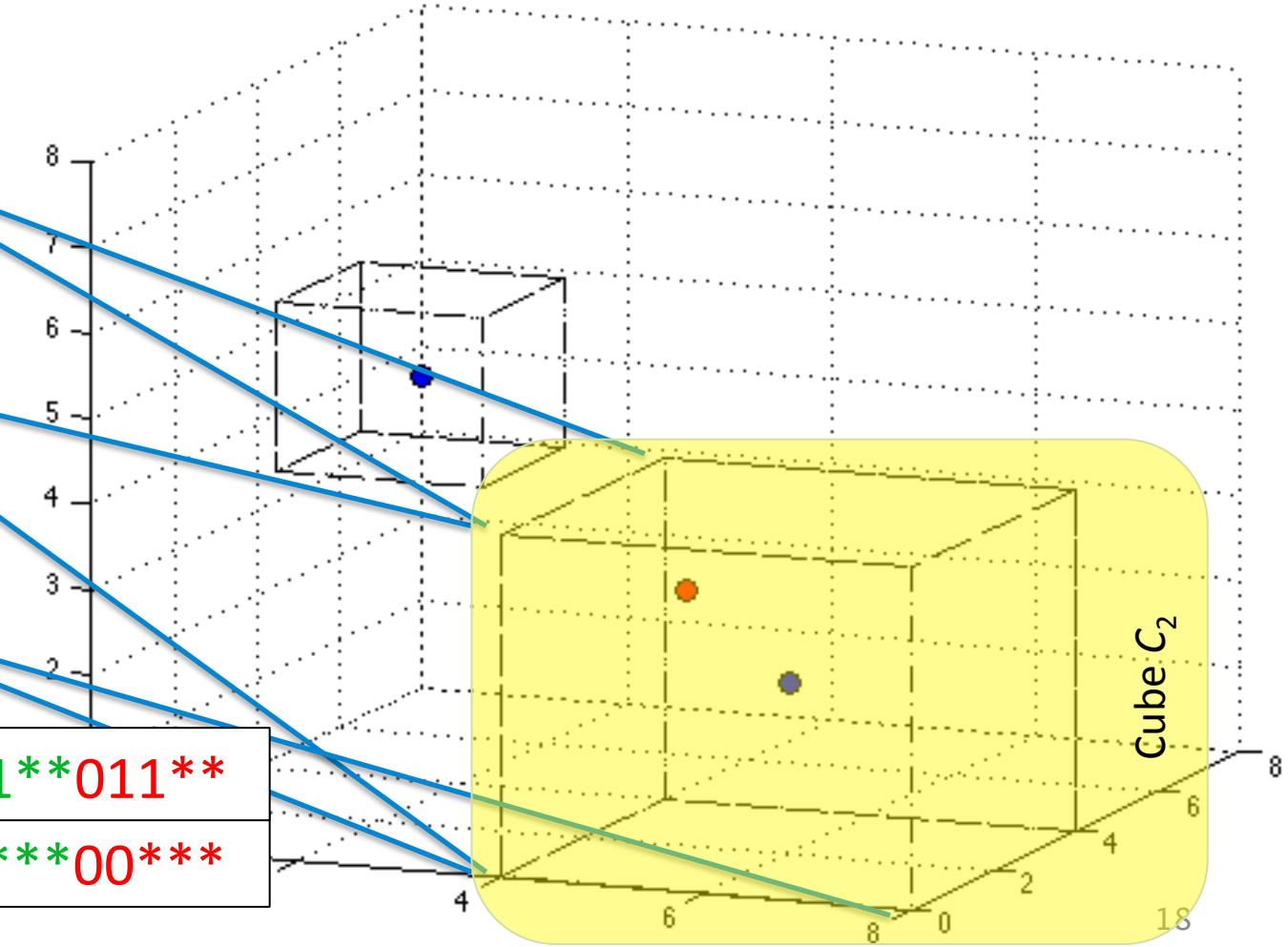
Multiple Dimensions

Trivial expansion to *d* dimensions:

Y=[0,4)
tcode(Y)=00***

Z=[0,4)
tcode(Z)=00***

X=[4,8)
tcode(X)=01***



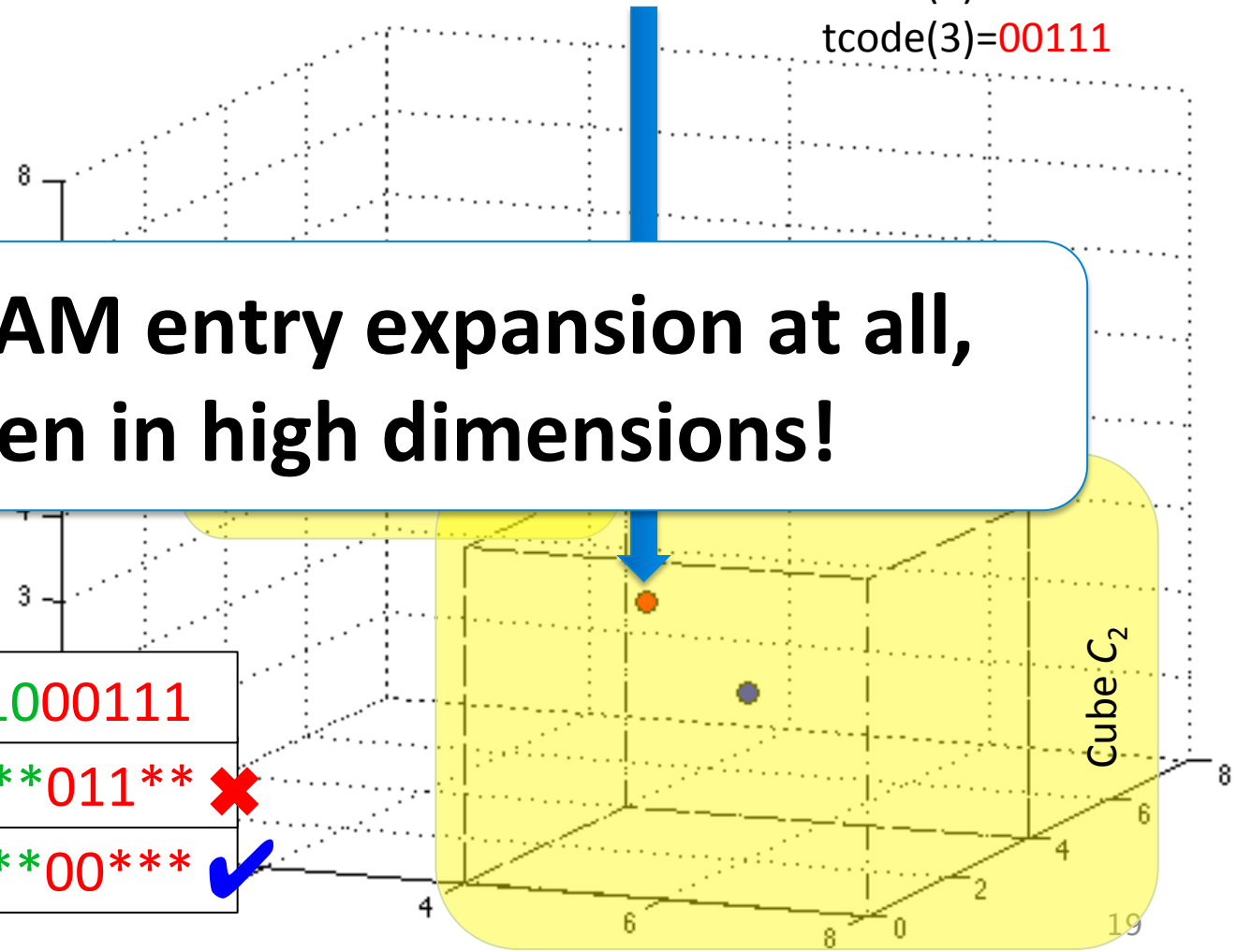
tcode(C_1)=00*10001**011**
tcode(C_2)=01***00***00***

Multiple Dimensions

Trivial expansion to *d* dimensions:

Query q=(5,2,3)

tcode(5)=01101
tcode(2)=00110
tcode(3)=00111



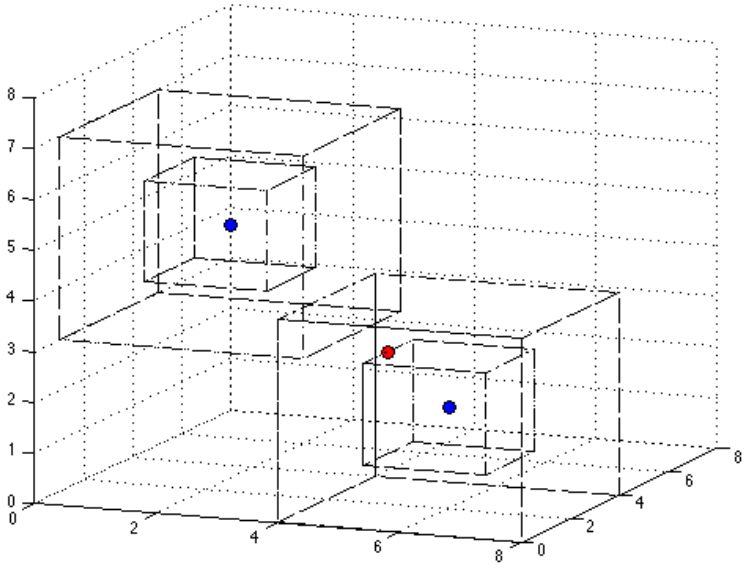
No TCAM entry expansion at all, even in high dimensions!

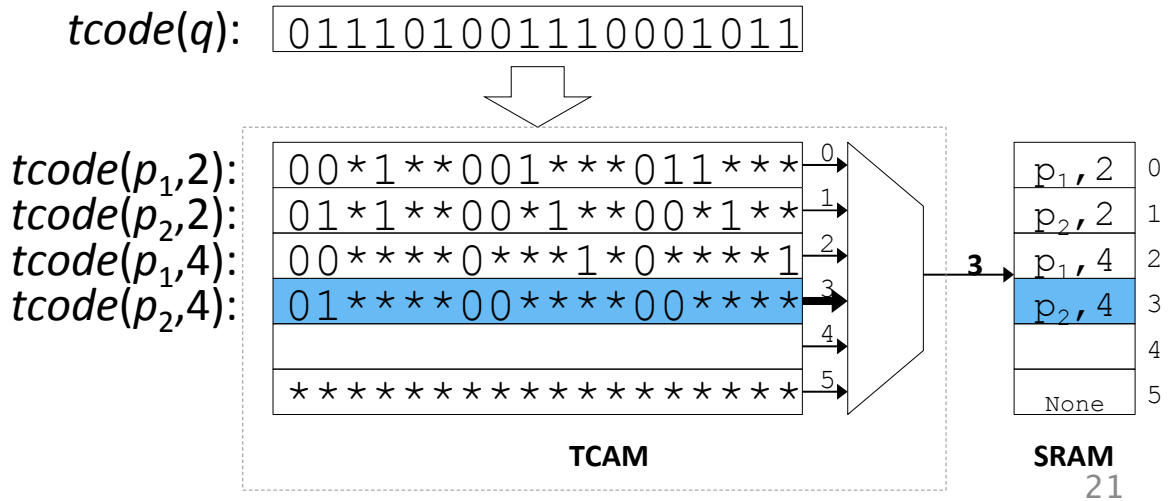
tcode(*p*)=0110100110000111
tcode(C_1)=00*10001**011** ✖
tcode(C_2)=01***00***00*** ✔

Solving Nearest Neighbor Search on TCAM

Nearest Neighbor Search on TCAM

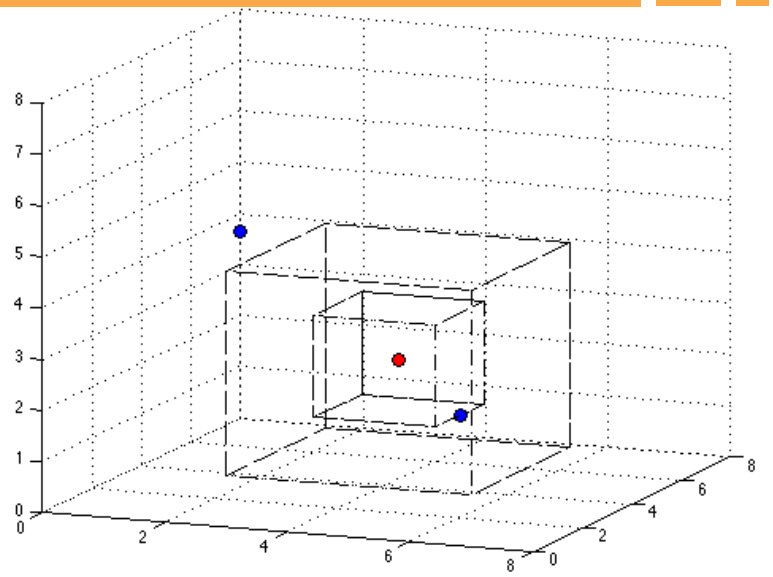
- <u>Time efficient method:</u>
  TCAM entries are cubes around data points
  - Sort cubes by size – smaller first
  - Solve in a single TCAM lookup!



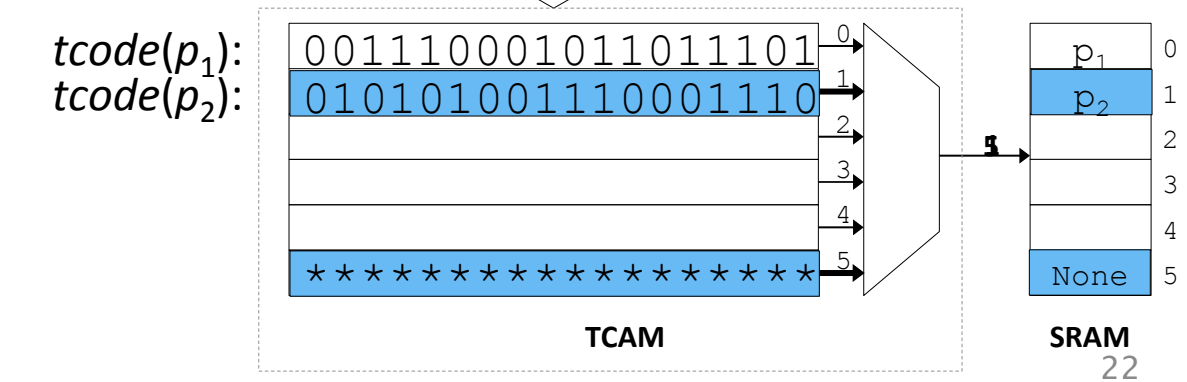


Nearest Neighbor Search on TCAM

- Space efficient method:
  TCAM entries are the data points
  Query is split to a set of cubes
  - Solve in a several TCAM lookups
  - Requires less TCAM space
  - Binary search on cube size

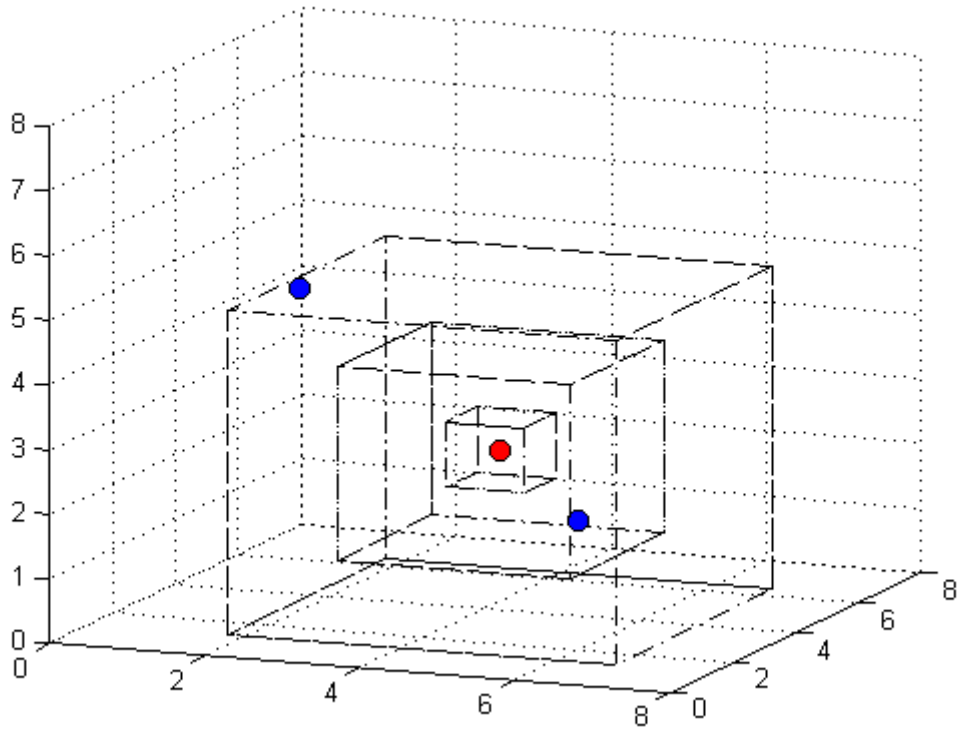


tcode(*q*,2): 011***00**10001***
tcode(*q*,4): 0****100****0***1*



Optimal Results in ℓ_∞

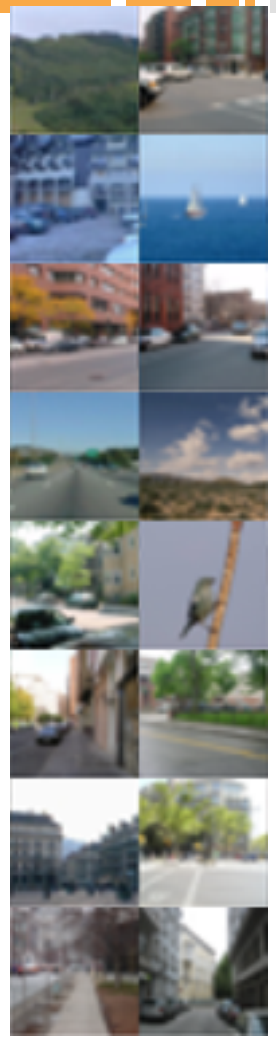
- Space is discrete
- Encoding all cubes of odd edge lengths 1,3,5,… provides an exact solution in ℓ_∞
  - 1=The point itself
  - 3=Distance 1 in ℓ_∞
  - 5=Distance 2 in ℓ_∞
  - …
- Less cubes: c-approximation in ℓ_∞
- In ℓ_p: $c \cdot \sqrt[p]{d}$-approximation ($c \geq 1$)



Experiments and Evaluation

Dataset

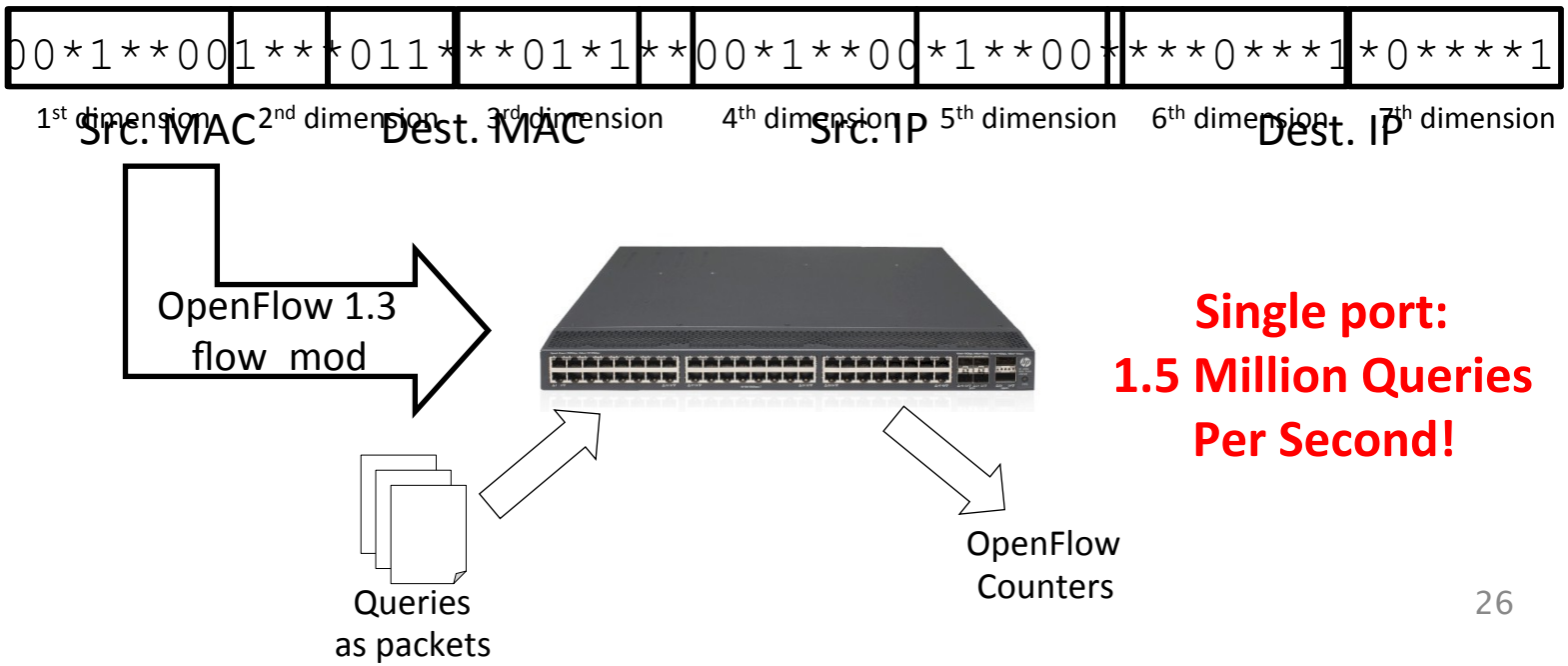
- Use case: Image similarity search (LabelMe 22K)
  - Using GIST descriptors (vectors in R^{512})
  - Downsampled to R^{40} using PCA
  - Quantified uniformly to $\{0,\ldots,255\}^{40}$
    - TCAM Entry width: 40 × 12 bits = 480 bits
      (contemporary TCAMs support up to 640 bits per entry)
    - Maximal cube edge length: 8
- 22,019 images
  - Data set: up to 21,019 images
    - Total size on TCAM: 9.6Mbit to 80Mbit
      (contemporary TCAMs have 40Mbit-80Mbit space)
  - Queries: 1000 images



Experiment on a Real TCAM

- Currently - no evaluation board for TCAM
- Instead, we used a commodity network switch that contains a TCAM
  - Switch has 48 ports of 1Gbps each (1.5 Million packet per second)
  - TCAM is 192 bits wide

d-Dimensional Cube Representation:



Single port:
1.5 Million Queries Per Second!

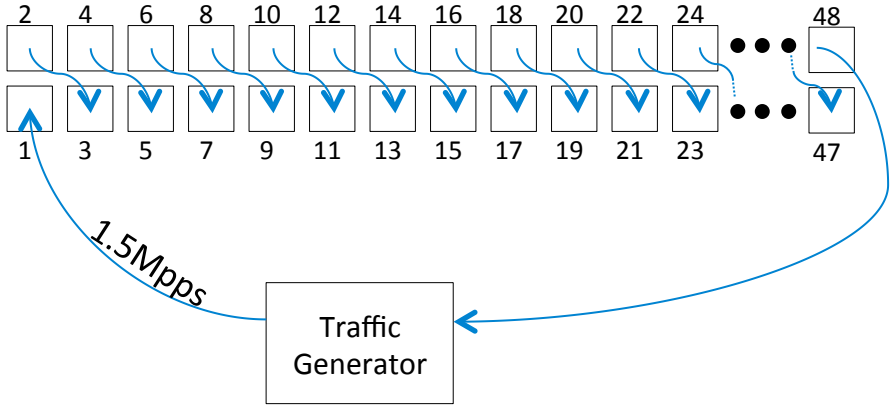
Experiment on a Real TCAM

- Can we do more?
  - Yes!

Proof of concept experiment:

Forwarding Rules:

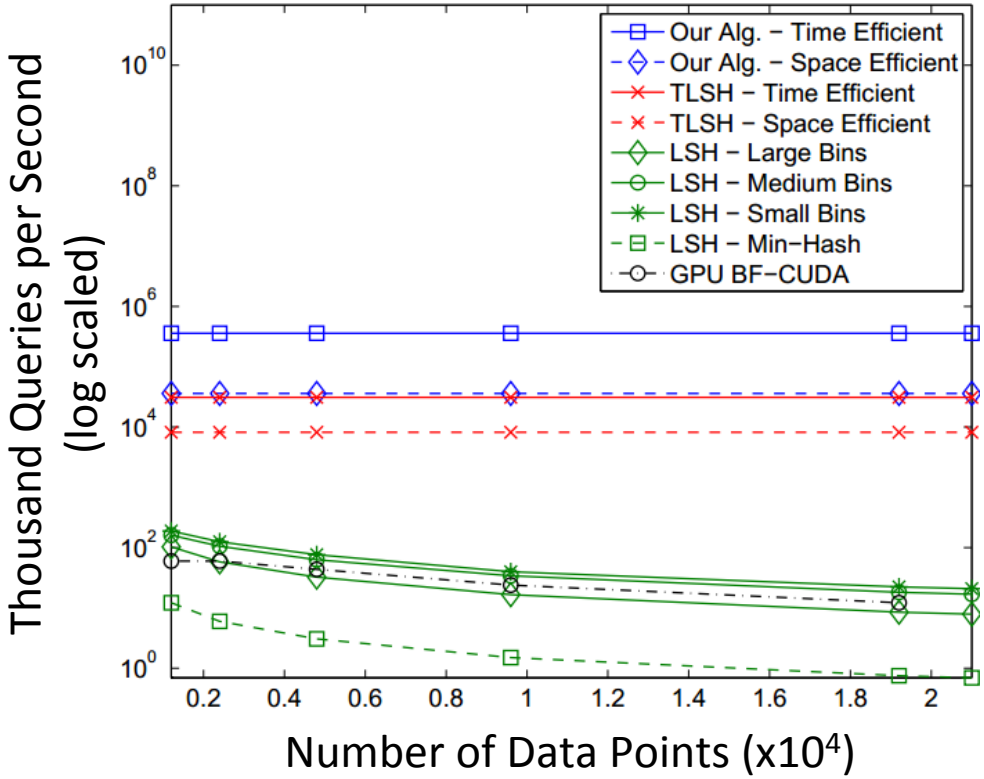
- From port 1 to port 2
- From port 3 to port 4
- From port 5 to port 6
- …
- From port 47 to port 48



24 ports: 35.69 Million Queries Per Second!
(almost 24*1.5M)

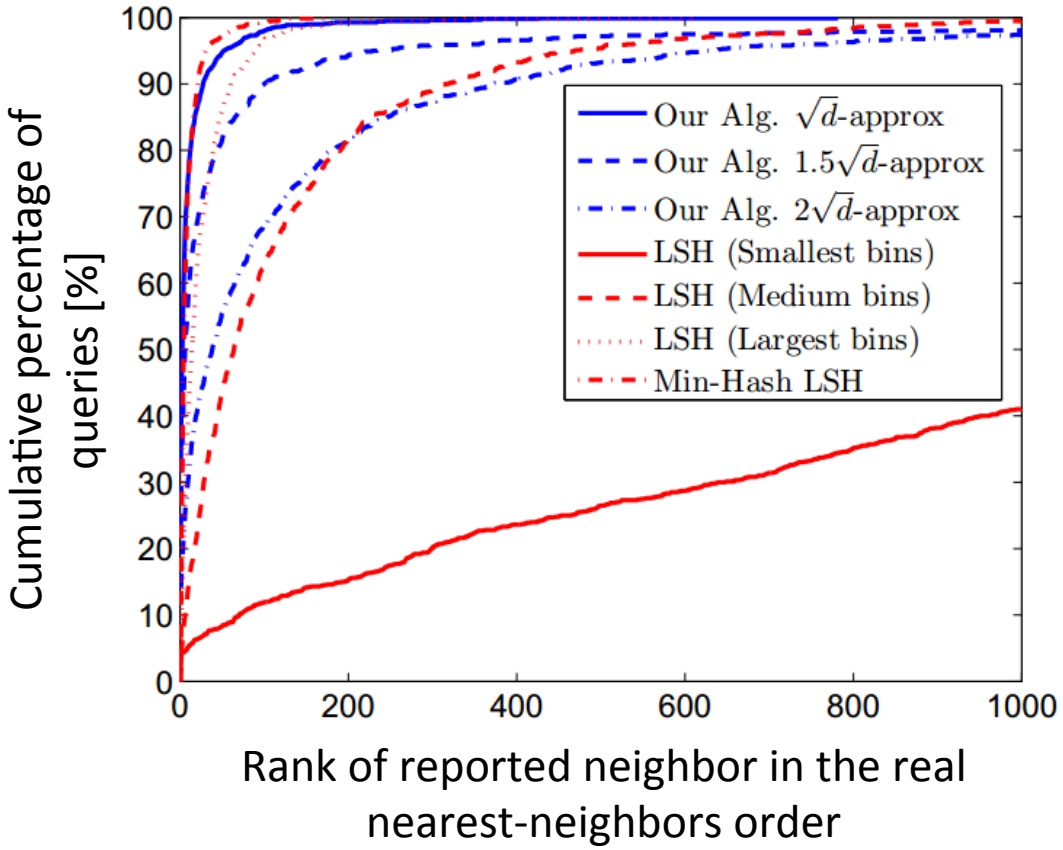
Performance Analysis & Simulation

- Contemporary TCAMs provide rates of **360 million** to more than **1 billion** queries per second
- We were *limited by the number of* ***ports***. TCAM could do more



Quality of Results

- In ℓ_∞: optimal results
- In ℓ_2: approximated results:



Conclusions

- TCAM provides high parallelism
- Our encoding scheme and algorithms solve the Nearest Neighbor problem in high dimensional spaces using a single (or a few) TCAM lookups
  - Very high search rate (many millions per second)
- We believe TCAM can be useful for more purposes, also outside the networking field
  - Use TCAM as a coprocessor / search accelerator

Questions?

Thank you.