Haar Wavelets

- Wavelets: Used to study “frequency components” of a signal
- Also proposed in databases for data reduction or approximate query answering
Haar Wavelets

- Wavelets: Used to study “frequency components” of a signal
- Also proposed in databases for data reduction or approximate query answering
Haar Wavelets

- Keep only top “k” coefficients
  - Need to normalize first (not discussed in paper in detail)
  - Minimizes the sum-squared error
- Allows for multi-resolution query processing
Haar Wavelets

- Keep only top “k” coefficients
  - Need to normalize first (not discussed in paper in detail)
  - Minimizes the sum-squared error

- Allows for multi-resolution query processing

- Ordering of data elements?
  - In signal processing, the numbers form a sequence, so they are ordered
  - In some cases, you might be able to reorder
  - Most likely NP-Hard to find the best ordering

- In sensor networks, ordering may need to be preserved
  - Depends on the application
TAG interface can be used to implement this *merging function*:

- Take the two PSRs from children, and merge them
- Send top “k” coefficients up
- Two problems:
  - May have more than 2 children
  - The children may be of different sizes
TAG interface can be used to implement this merging function:

- Take the two PSRs from children, and merge them
- Send top “k” coefficients up
- Two problems:
  - May have more than 2 children
  - The children may be of different sizes
- Must “zero-pad”
  - Can have too many zeros
  - This introduces larger errors

Consider a Haar wavelet on the sequences:
{100, 100, 0, 0, 0, 0, 0, 0} vs {100, 100}
Can we choose the communication tree so that zero-padding is not required?

What is the optimal tree?

- For Haar wavelets, it is a \textit{binomial} tree
Can we choose the communication tree so that zero-padding is not required?

What is the optimal tree?

For Haar wavelets, it is a *binomial* tree.

Why? An alternate explanation:

- Consider a node that has “k” children: \( c_1, \ldots, c_k \)
- It also has its own value, \( v \) (singleton)
- One of the children must be of size 1 (so it can be merged with \( v \) without zero-padding)
- Another child must be of size 2
- The k’th child must be of size \( 2^{k-1} \)
Haar Wavelets in Sensor Networks

The binomial tree comes from the data structures literature: the binomial tree \([4]\) (right hand side of Figure 4).
Can we enforce such a communication tree in the sensor network?

- Open problem
Generalizing the Ideas

- What about other types of aggregates?
- May not be able to map the support graph directly onto communication graph

Basic ideas also extend to peer-to-peer networks

Use overlay networks
Nodes forward messages without applying merging functions
Essentially assume a complete communication graph
Hard to optimize (need to use weighted graphs)

Very interesting problems in this line of work
Generalizing the Ideas

- What about other types of aggregates?
- May not be able to map the support graph directly onto communication graph
  - Use overlay networks
  - Nodes forward messages without applying merging functions
    - Essentially assume a complete communication graph
    - Hard to optimize (need to use weighted graphs)
- Very interesting problems in this line of work
  - Basic ideas also extend to peer-to-peer networks
Followup work after TAG

“Acquisitional” query processing

- We can control when/where to observe data (within certain limits)
- In traditional databases, the data is given
- How does this change query processing?
Followup work after TAG

“Acquisitional” query processing
- We can control when/where to observe data (within certain limits)
- In traditional databases, the data is given
- How does this change query processing?

Later work
- “Model-driven data acquisition in sensor networks”; VLDB 2004
- Raises “value of information” problems
- E.g. if I can only sample one sensor among all, which one should it be?
- Energy consumption
- Little difference in receiving and transmitting costs

![Time v. Current Draw In Different Phases of Query Processing](image)

**Figure 2: Phases of Power Consumption In TinyDB**
Basic query:

```sql
SELECT nodeid, light, temp
FROM sensors
SAMPLE INTERVAL 1s FOR 10s
```

Creating storage points (persistent views):

```sql
CREATE
STORAGE POINT recentlight SIZE 8
AS (SELECT nodeid, light FROM sensors
SAMPLE INTERVAL 10s)
```
Basic query:

```
SELECT nodeid, light, temp
FROM sensors
SAMPLE INTERVAL 1s FOR 10s
```

Creating storage points (persistent views):

```
CREATE
STORAGE POINT recentlight SIZE 8
AS (SELECT nodeid, light FROM sensors
SAMPLE INTERVAL 10s)
```

Allows querying over sliding windows

```
SELECT COUNT(*)
FROM sensors AS s, recentLight AS rl
WHERE rl.nodeid = s.nodeid
AND s.light < rl.light
SAMPLE INTERVAL 10s
```
Event-based queries: Much superior to polling

```
ON EVENT bird-detect(loc):
    SELECT AVG(light), AVG(temp), event.loc
    FROM sensors AS s
    WHERE dist(s.loc, event.loc) < 10m
    SAMPLE INTERVAL 2 s FOR 30 s
```

Lifetime-based queries:
- Can specify a lifetime for the query
- Decide the sampling rate based on that
  - Tricky to estimate lifetimes accurately
  - Also, the sampling rate is usually dependent on the application
Query optimization

- Cost-based (as with traditional relational databases)
- Maintain a catalog with sensing costs etc..
  - Also aggregates (whether they are monotonic, distributive etc.)
- When to take samples?
  - Should interleave taking samples and evaluating predicates
Query optimization

- Cost-based (as with traditional relational databases)
- Maintain a catalog with sensing costs etc..
  - Also aggregates (whether they are monotonic, distributive etc.)
- When to take samples?
  - Should interleave taking samples and evaluating predicates
  - Query: WHERE light < 100 Lux and temp > 20C
Query optimization

- Cost-based (as with traditional relational databases)
- Maintain a catalog with sensing costs etc..
  - Also aggregates (whether they are monotonic, distributive etc.)
- When to take samples?
  - Should interleave taking samples and evaluating predicates
  - Query: WHERE light < 100 Lux and temp > 20°C
  - If we know the “selectivities”, can solve optimally
- Similar optimizations for MAX
- Also, use “event batching” for event-based queries
  - Some tricky semantics issues
ICDE 2005

“Exploiting correlated attributes in acquisitional query processing”

Use attribute correlations to optimize this further

May sample “time” first and then decide between “light” and “temp”
ICDE 2005

“Exploiting correlated attributes in acquisitional query processing”

Use attribute correlations to optimize this further

May sample “time” first and then decide between “light” and “temp”
Semantic Routing Trees
- A form of “index” on the sensor network attributes
- Built on the “constant” attributes
  - Each node stores the range of attribute values over its descendants
  - If a query doesn’t overlap the range, no need to disseminate
Semantic Routing Trees
- A form of “index” on the sensor network attributes
- Built on the “constant” attributes
  - Each node stores the range of attribute values over its descendants
  - If a query doesn’t overlap the range, no need to disseminate

Sleep cycling?
- All nodes awake at about the same time

Data prioritization
- What if we run out of queue space?
- Should keep those which have the most information