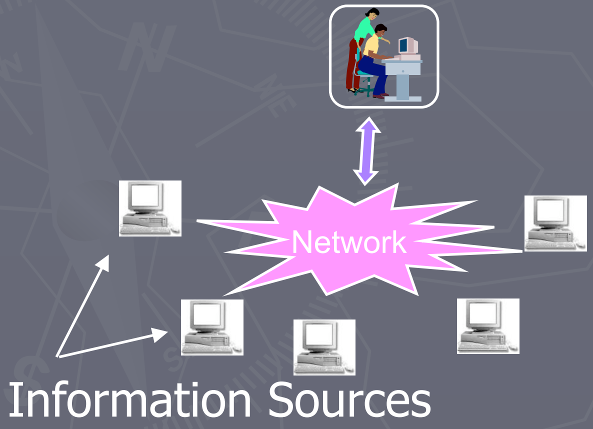
Exploiting Correlated Attributes in Acquisitional Query Processing

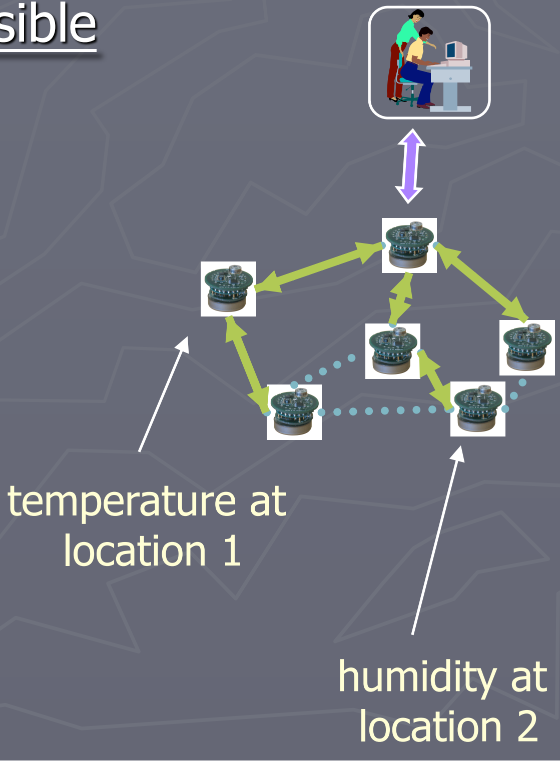
Amol Deshpande
University of Maryland

Joint work with Carlos Guestrin@CMU, Sam Madden@MIT,
Wei Hong@Intel Research

Motivation

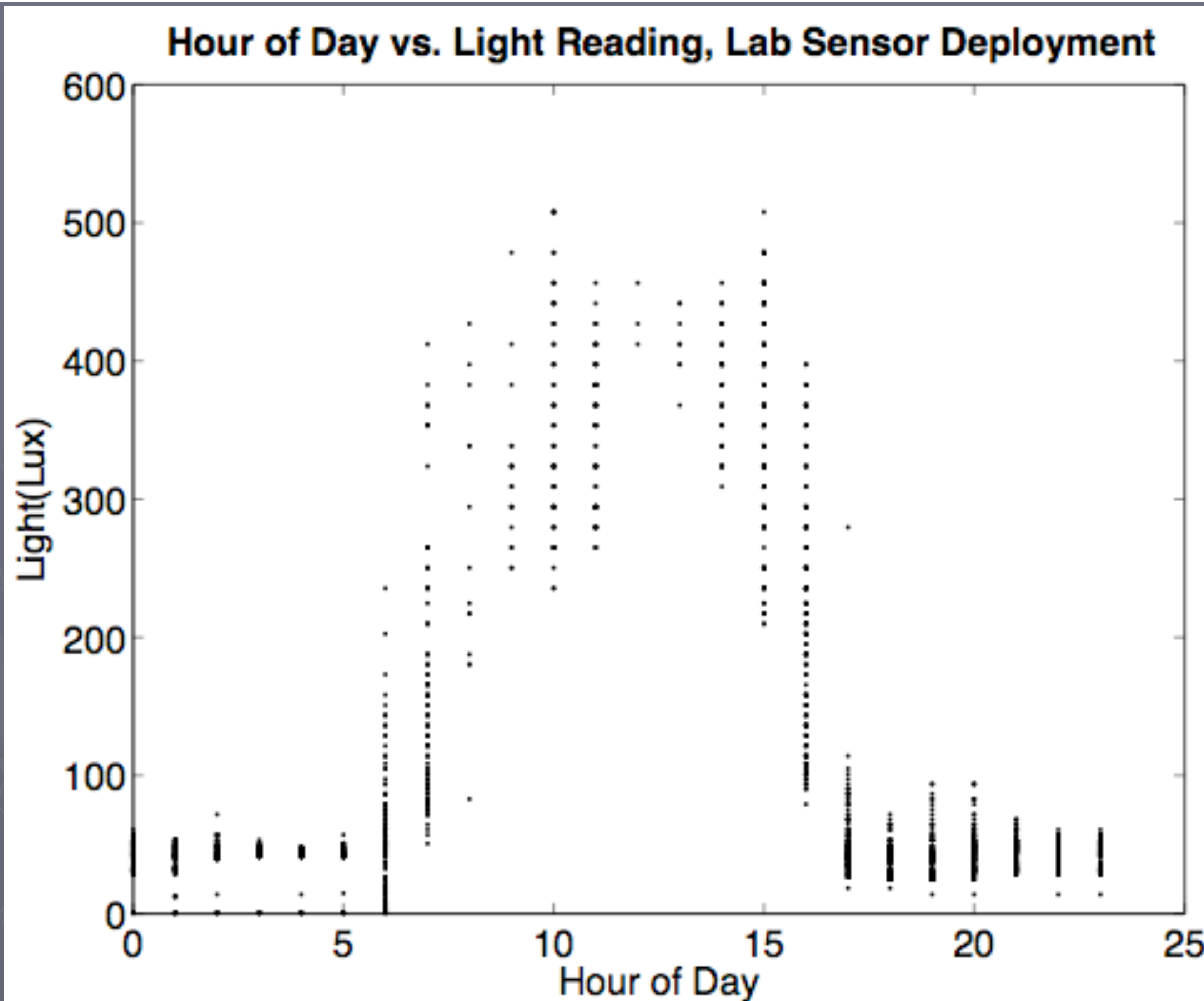
- Emergence of large-scale distributed information systems
  - Web, Grid, Sensor Networks etc..
- Characterized by high data *acquisition* costs
  - Data doesn't reside on the local disk; must acquire it
  - <u>Goal: reduce data acquisition as much as possible</u>

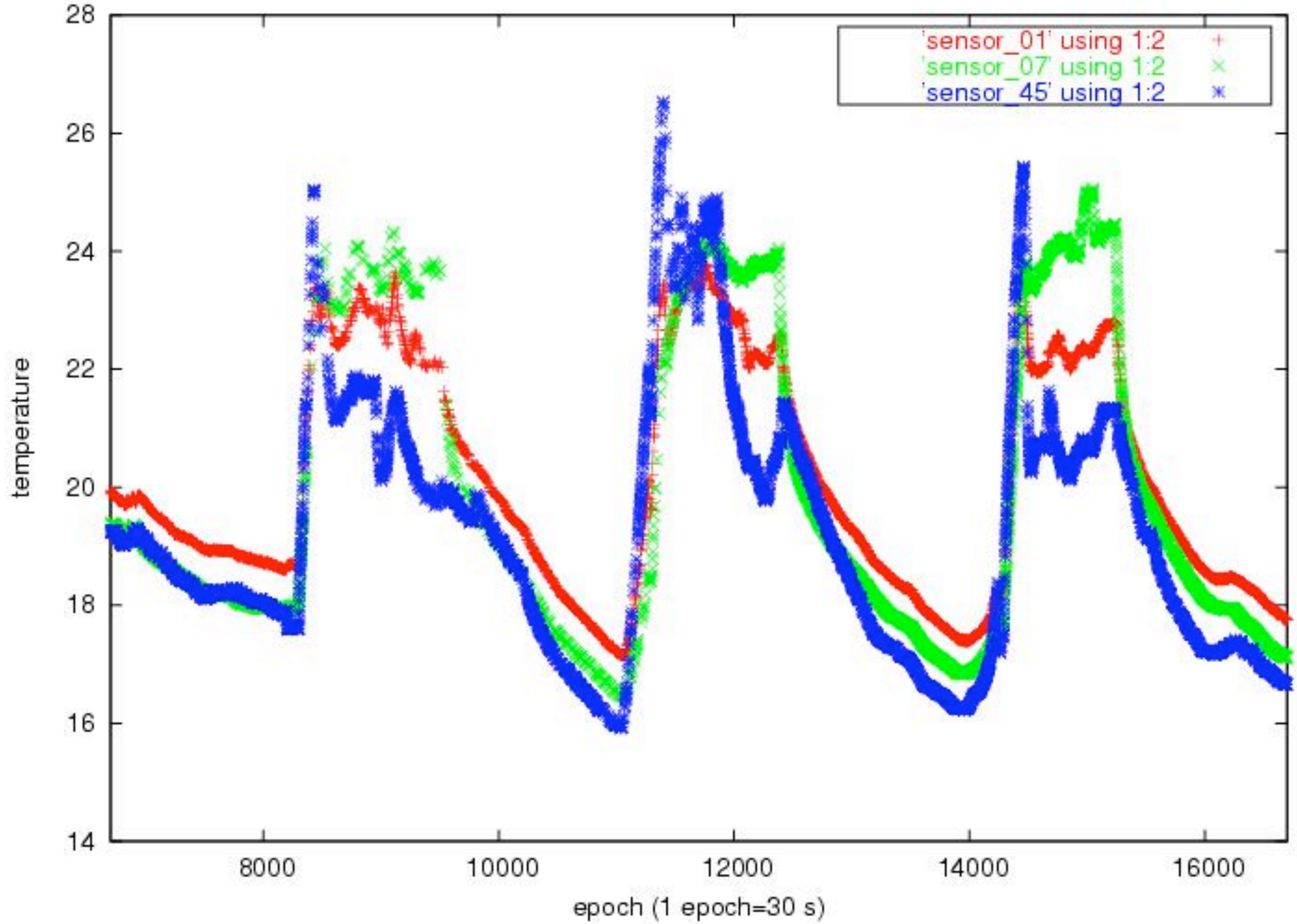




Motivation

- Many of these systems exhibit strong data correlations
  - E.g. in sensor networks…
    - Temperature and voltage
    - Temperature and light
    - Temperature and humidity
    - Temperature and time of day
    - etc.


Hour of Day vs. Light Reading, Lab Sensor Deployment
Light(Lux)
Hour of Day
0
100
200
300
400
500
600
0
5
10
15
20
25



'sensor_01' using 1:2
'sensor_07' using 1:2
'sensor_45' using 1:2
temperature
epoch (1 epoch=30 s)
28
26
24
22
20
18
16
14
8000
10000
12000
14000
16000

Motivation

- Many of these systems exhibit strong data correlations
  - E.g. in sensor networks…
    - Temperature and voltage
    - Temperature and light
    - Temperature and humidity
    - Temperature and time of day
    - etc.
  - Observing one attribute $\Rightarrow$ information about other attributes

Motivation

- Database systems typically ignore correlations
- Our agenda:
  - Exploit the naturally occuring *spatial* and *temporal* correlations in novel ways to reduce data acquisition
- Model-driven Acquisition in Sensor Networks [VLDB'04]
  - A new way to look at data management
- This talk:
  - A more traditional query optimization question
  - Signficant benefits possible even here
    - Even in traditional domains

Conjunctive Queries

- Queries of the form:

  SELECT X1, X2, X3
  WHERE *pred1*(X1)
  AND *pred2*(X2)
  AND *pred3*(X3)

- Common in acquisitional environments
  - Sensor networks:
    - Find sensors reporting temperature between 10^{o}C and 20^{o}C and light less than 100 Lux
    - Are there any sensors not within above bounds ?
  - Web data sources:
    - What fraction of people who donated to Gore in 2000 also have patents ?

Conjunctive Queries

- Queries of the form:

  SELECT X1, X2, X3
  WHERE X1 = x1
  AND X2 = x2
  AND X3 = x3

May involve:
turning a sensor
on and sampling, *or*
querying a web index
over the network

- Current approach: *Sequential Plans*
  - Choose a single order of attributes to observe, for all tuples

Observe *X1*

If *X1 = x1*, then Observe *X2*

If *X1 = x1* and *X2 = x2*, then Observe *X3*

Finding Sequential Plans

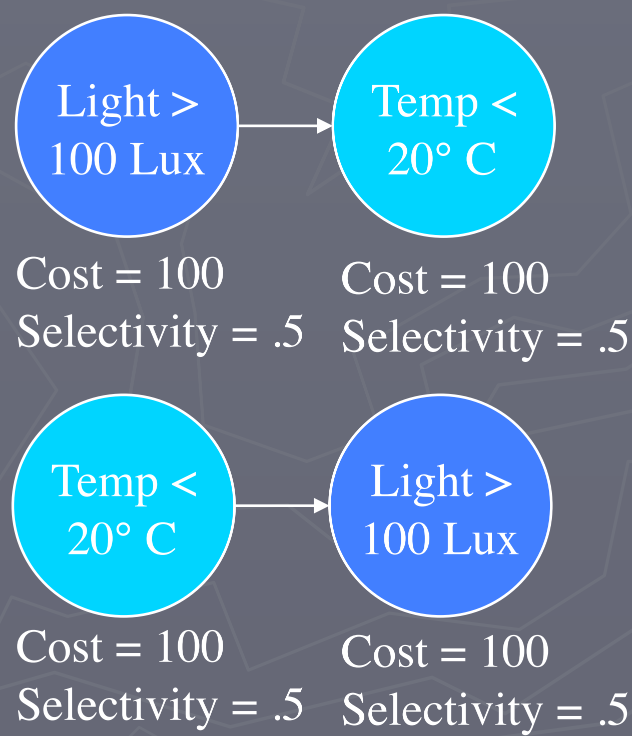
- Naïve:
  - Order predicates by *cost/(1 – selectivity)*
    - *selectivity* = fraction of tuples that pass the predicate
  - Ignores correlations
  - Used by most current query optimizers
- Optimal:
  - Find the optimal order considering correlations
  - NP-Hard
- A 4-approximate solution:
  - Greedily choose the attribute that maximizes the return [Munagala, Babu, Motwani, Widom, ICDT 05]

Observation

- Cheap, correlated attributes can improve planning
  - By improving estimates of selectivities
  - Extreme Case - Perfect Correlation: *X4* → *X1*
  - Observing *X4* sufficient to decide whether *X1 = x1* is true
    - Would be a better plan if *X4* cheaper to acquire
      - E.g. *temperature* and *voltage* in SensorNets
    - Unfortunately, perfect correlations rarely exist → False +ve's and –ve's
      - Approach taken by Shivakumar et al [VLDB 1998]
- Our Approach: Choose different plans based on observed attribute values
  - Query always evaluated correctly
  - Sometimes, observe attributes not involved in query
  - Sounds adaptive, but we generate complete plans *a priori*
  - Applicable in both exact and approximate case

Example

SELECT * FROM sensors
WHERE light < 100 Lux *and* temp > 20° C



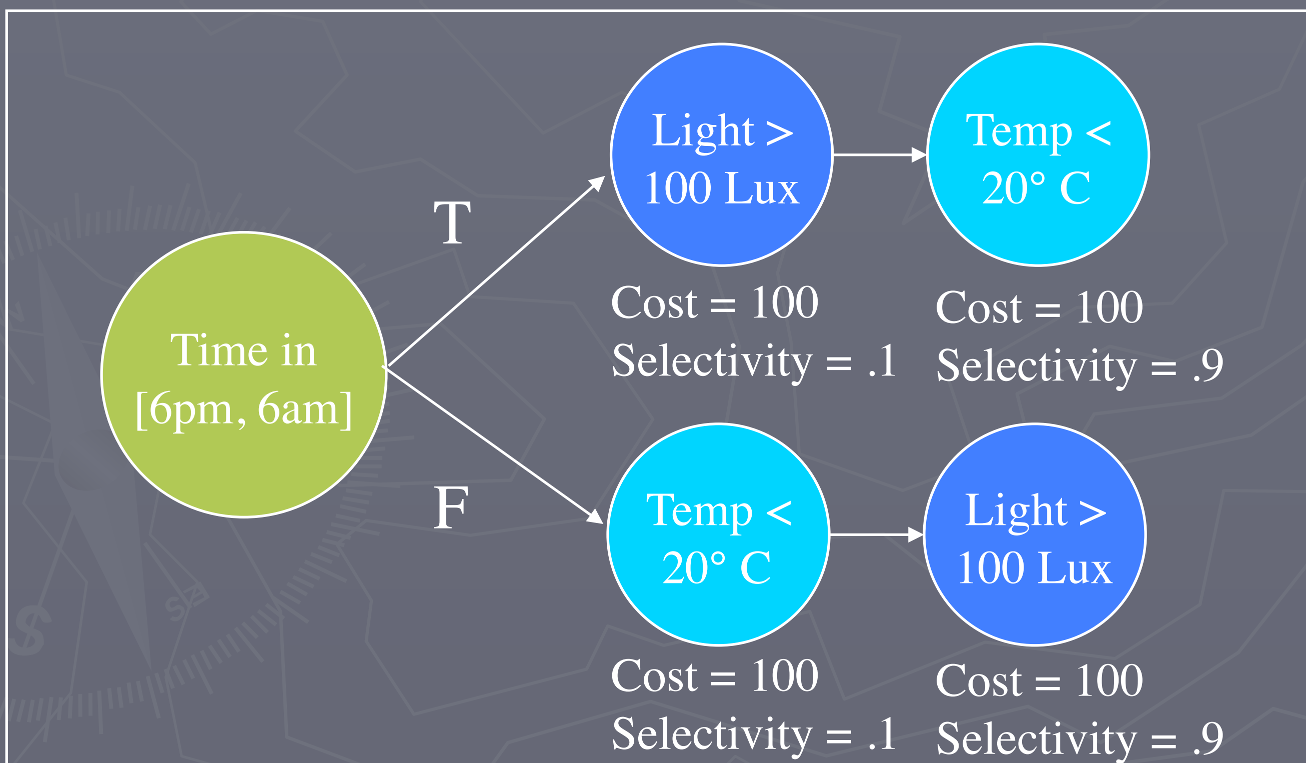
Expected Cost = 150

Expected Cost = 150

Example

A Conditional Plan

SELECT * FROM sensors
WHERE light < 100 Lux *and* temp > 20° C



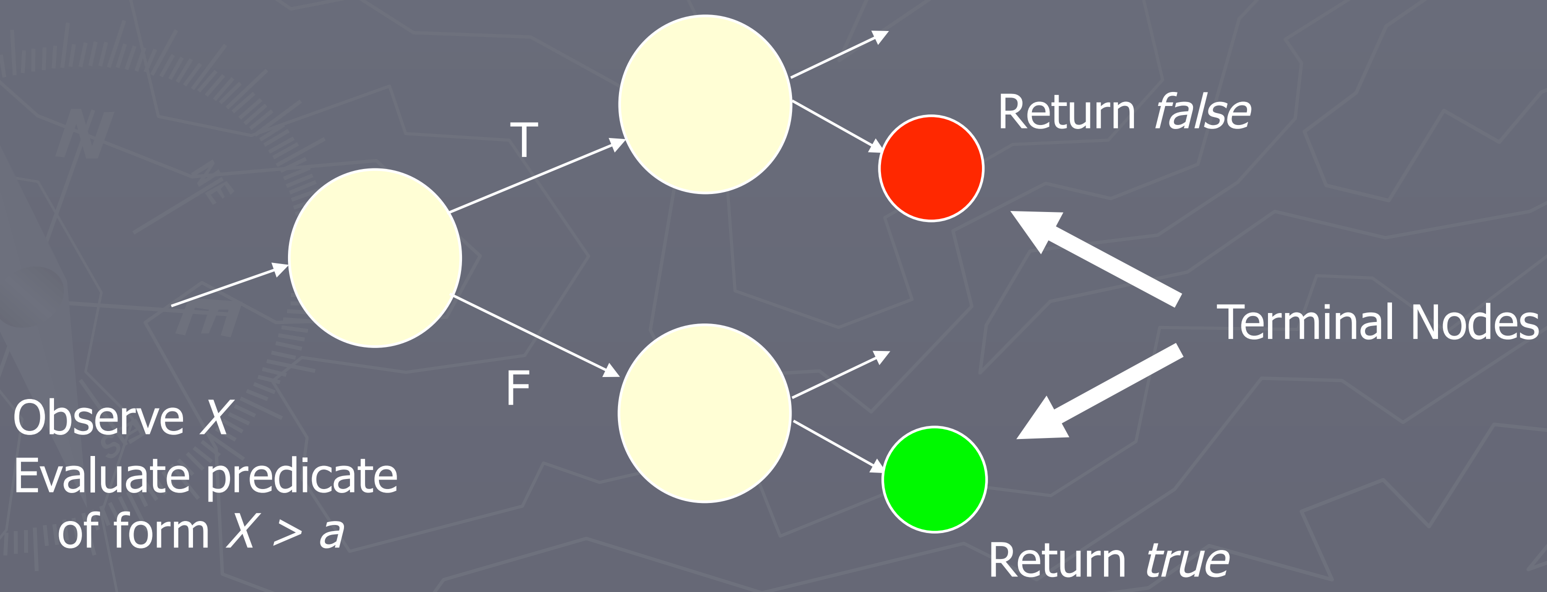
Expected Cost
= 110

Problem Statement

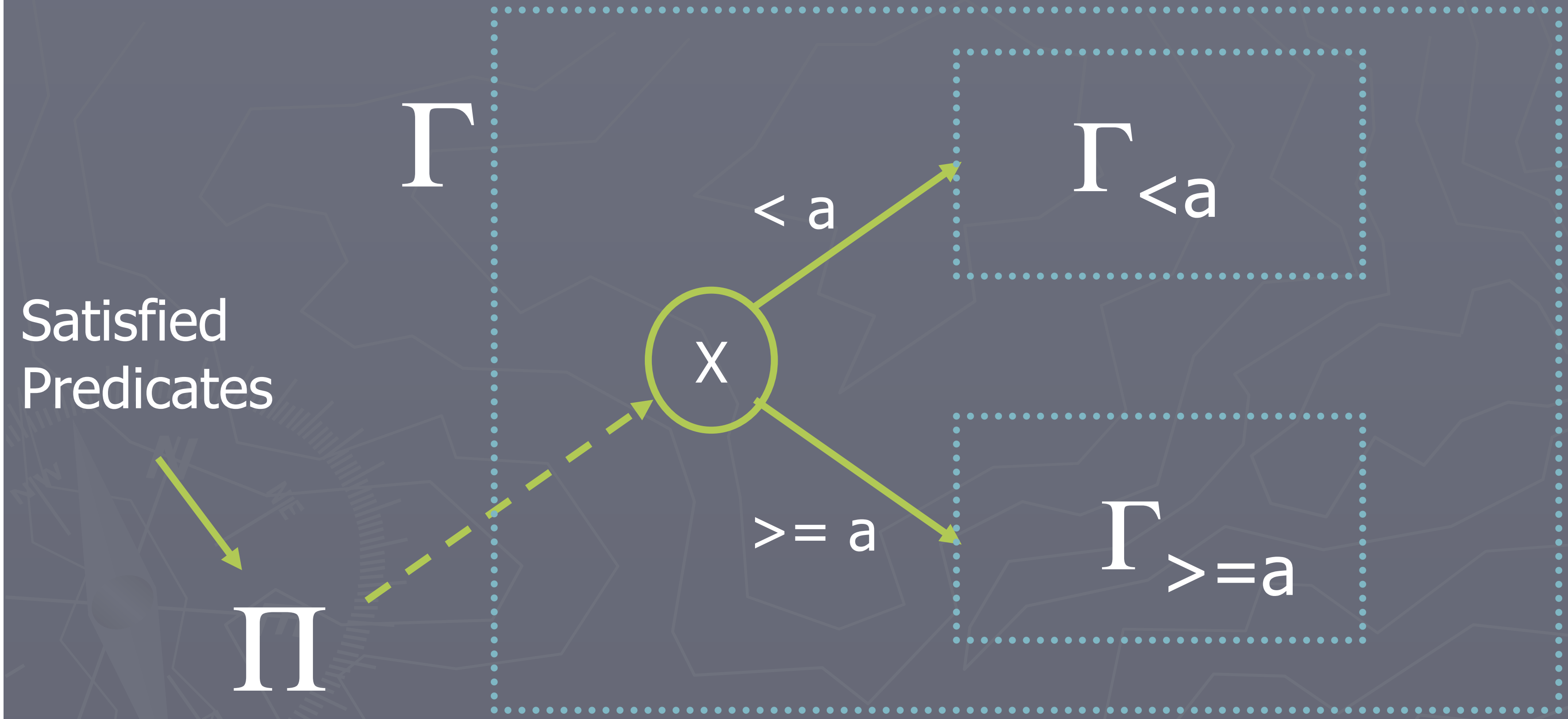
- Given a conjunctive query of the form

  $X_1 = x_1$ *and* $X_2 = x_2$ *and* … *and* $X_m = x_m$

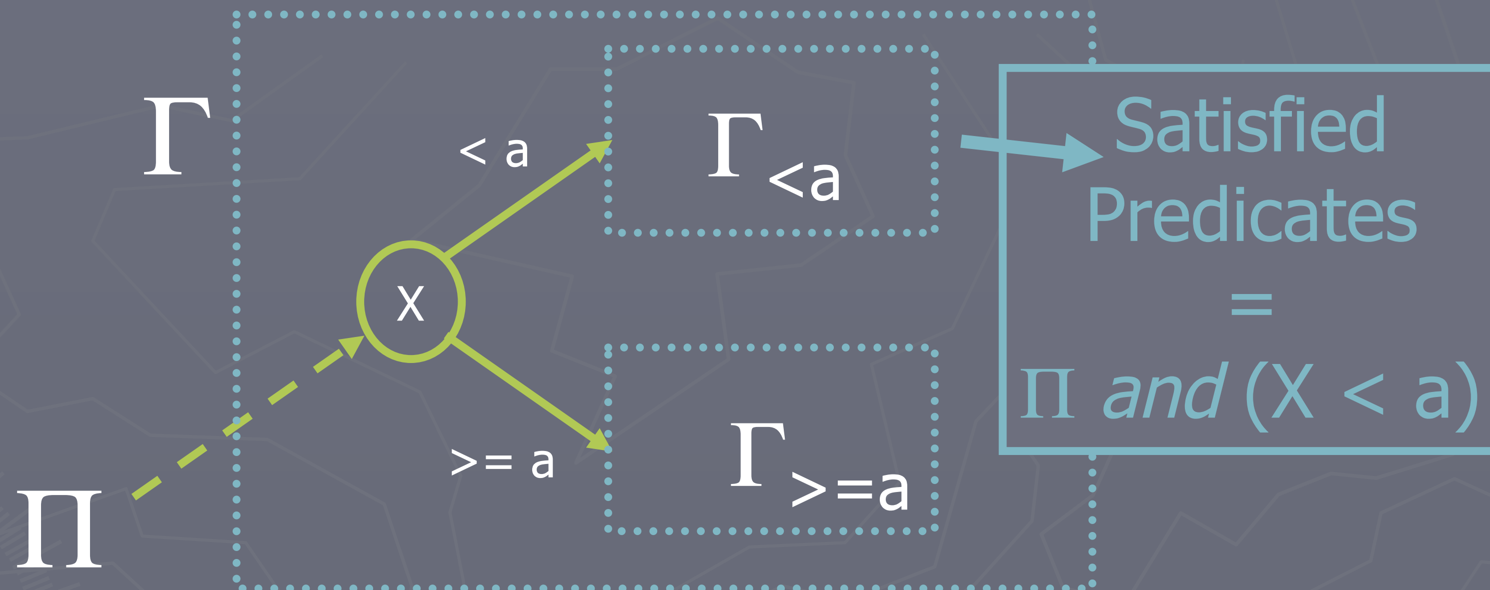
  and additional attributes X_{m+1}, …, X_n not referenced in the query, find the optimal conditional plan
  - We will restrict ourselves to *binary* conditional plans



Costing a conditional plan



Costing a conditional plan



$$\text{Cost}(\Gamma) = \boxed{C(X \mid \Pi)} + P(X < a \mid \Pi)\ \text{Cost}(\Gamma_{<a}) + P(X >= a \mid \Pi)\ \text{Cost}(\Gamma_{>=a})$$

Complexity

- Given an oracle that can compute any conditional probabilities in $O(1)$ time, deciding whether a plan with expected cost < K exists is #P-hard
  - Reduction from #3-SAT (counting version of 3-SAT)

- Given a dataset D, finding the optimal conditional plan for that dataset is NP-hard
  - Reduction from complexity of finding binary decision trees

Solution Steps

- Plan Costing
  - Need method to estimate conditional probabilities
- Plan Enumeration
  - Exhaustive vs. heuristic

Conditional Probability Estimation

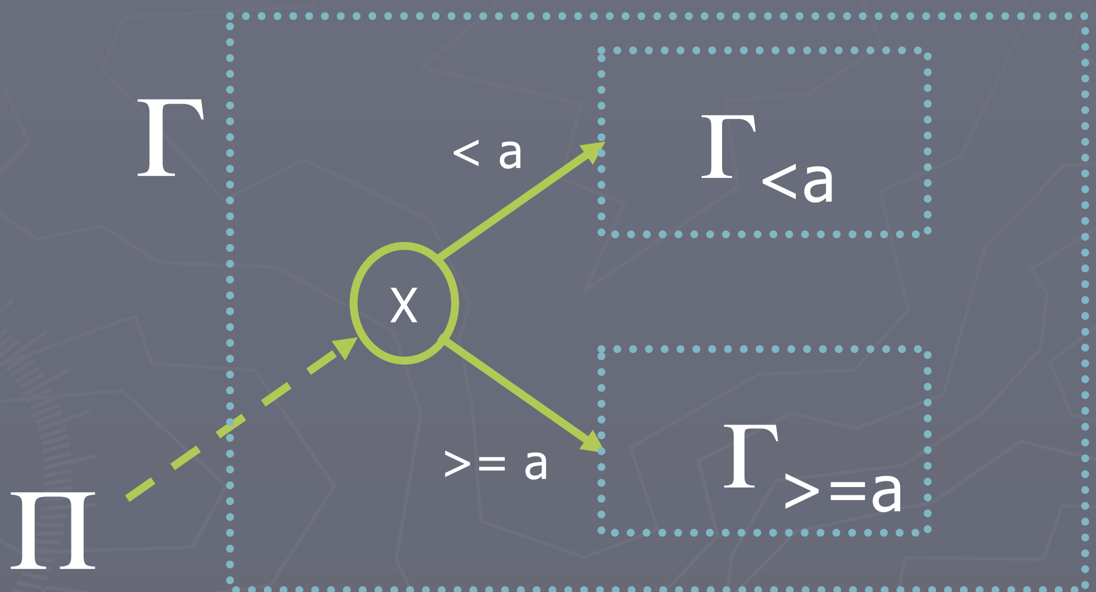
- We estimate conditional probabilities using observations over historical data
- Options:
  - Build a complete multidimensional distribution over attributes
    - \+ Can read off probabilities
    - \- Very large memory requirements
  - Scan historical data as estimates are needed
    - \+ Minimal memory requirements
    - \+ Can use random samples if datasets too large
  - Build a model that allows quick estimation of probabilities
    - E.g., graphical models
    - Allows reasoning about unobserved events
    - Avoids overfitting

Solution Steps

- Plan Costing
  - Need Method to Estimate Conditional Probabilities
- Plan Enumeration
  - Exhaustive vs. heuristic

Exhaustive Search

- Dynamic programming applicable



Subproblem defined by conditioning predicates (Π)
→ Can solve independently

Exhaustive Search

- Dynamic programming applicable
- Complexity: $O(K^{2n})$
- Prohibitive in most cases !!
  - Even if we use branch-and-bound techniques

Greedy Binary Split Heuristic

- Uses optimal sequential plans as base case
- Chooses locally optimal splits to improve greedily

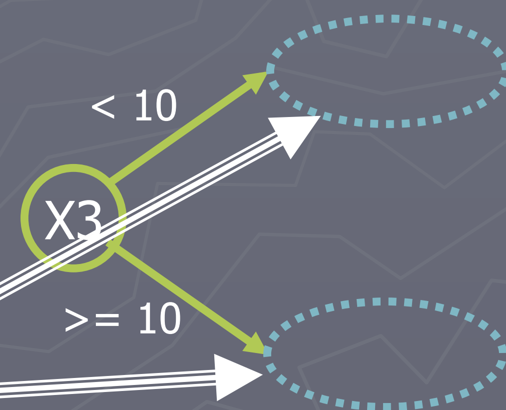
Example Query:

X1 = 1 *and* X2 = 1

1. Optimal Sequential Plan

2. Check all possible splits:

Eg:



Use optimal sequential plans to solve the (smaller) subproblems

Greedy Binary Split Heuristic

- Uses optimal sequential plans as base case
- Chooses locally optimal splits to improve greedily

Example Query:

X1 = 1 *and* X2 = 1

1. Optimal Sequential Plan

2. Check all possible splits:

Eg:

3. Choose locally optimal split

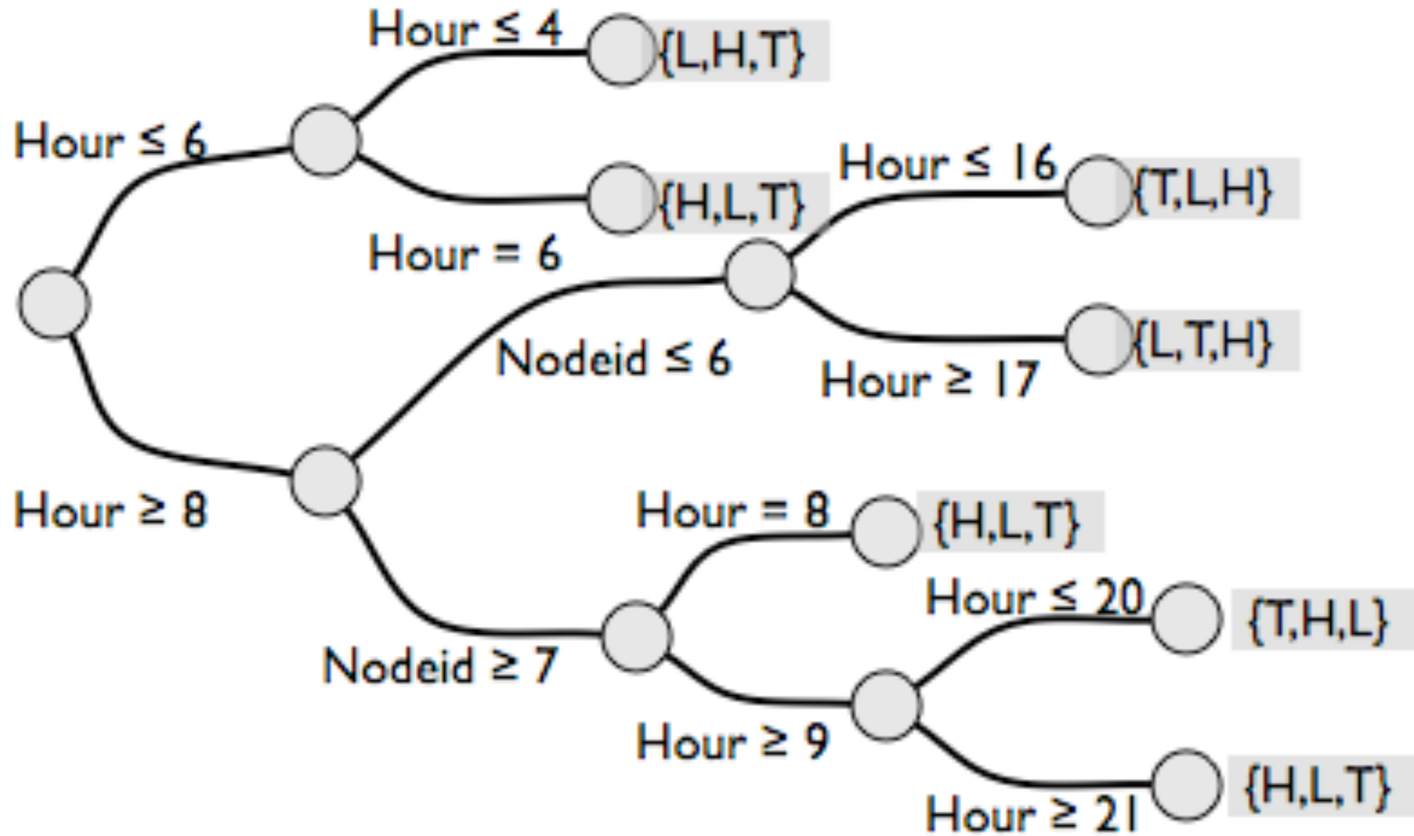
4. Recurse

Evaluation

- Datasets from real deployments
  - Lab
    - 45 motes deployed in Intel Berkeley Lab
    - 400,000 readings
    - Total 6 attributes; 3-predicate queries
  - Garden-11:
    - 11 motes deployed in a forest
    - 3 attributes per mote, temperature, voltage, and humidity.
    - Total 34 attributes; 33-predicate queries
    - Queries are issued against the sensor network as a whole
  - Also experiemented with Garden-5, and synthetic datasets
- Separated *test* from *training*
- Randomly generated range queries for a given set of query variables
- Java implementation, simulated execution based on cost model
  - Costs represent data acquisition only

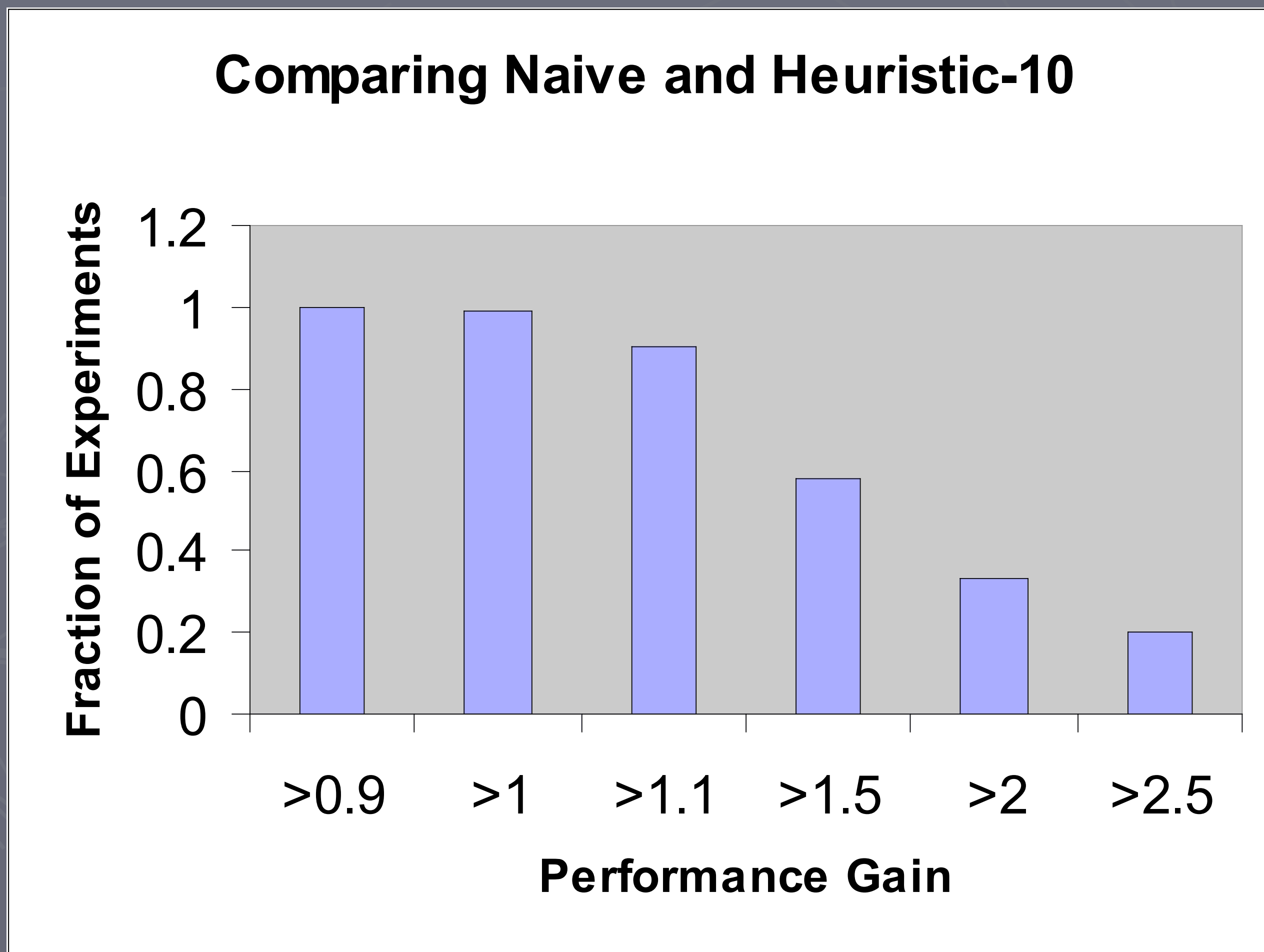
Example Plan: Lab Dataset

Query: SELECT * FROM sensors WHERE humidity in [36.5%, 42.5%]
AND temp in [17.8°C, 19.8 °C]
AND light in [593 lux, 2093 lux]

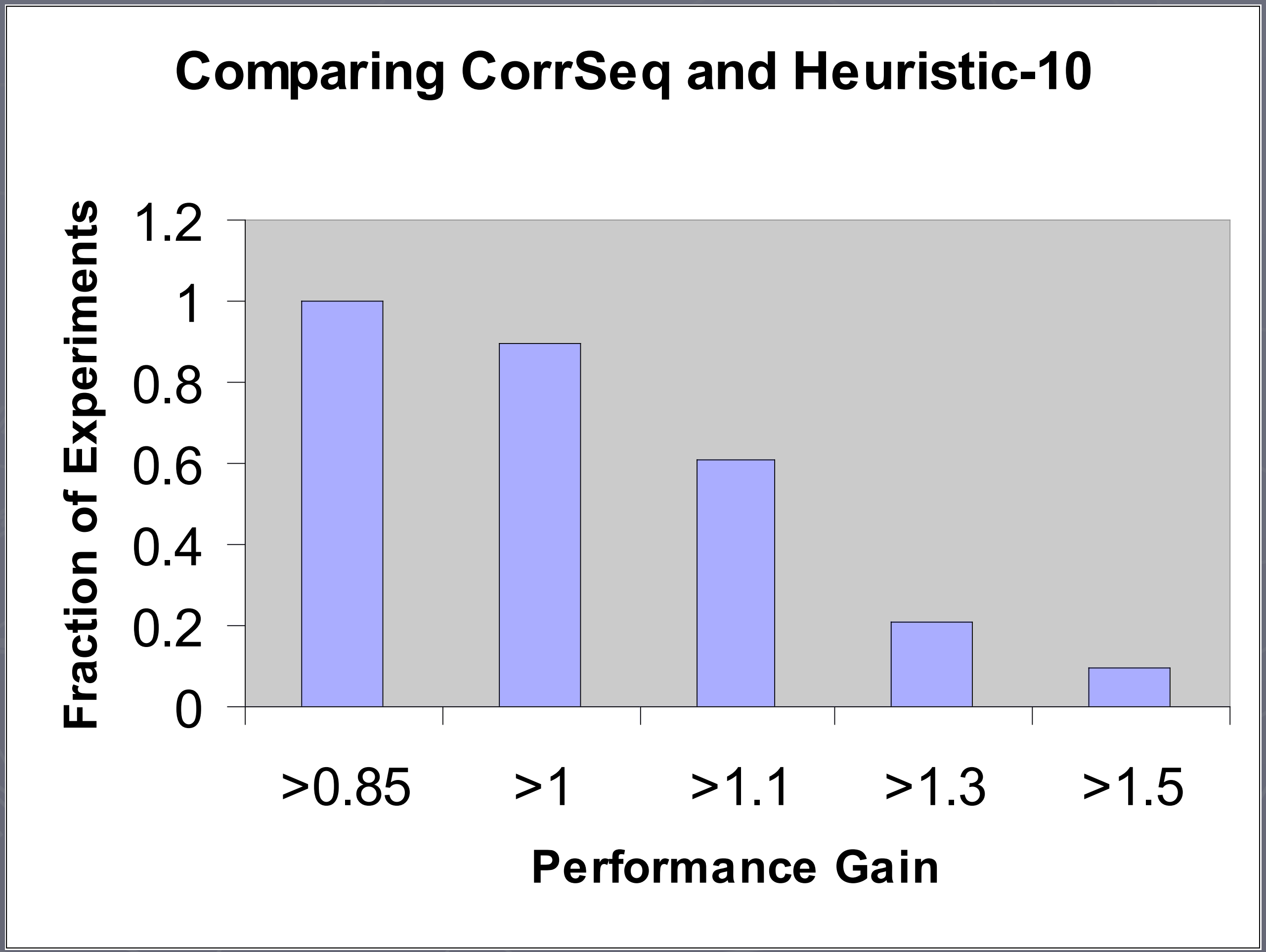


{H,L,T} denotes a sequential plan that samples Humidity, then Light, then Temperature

Garden-11 (1)


Comparing Naive and Heuristic-10
Fraction of Experiments
1.2
1
0.8
0.6
0.4
0.2
0
>0.9
>1
>1.1
>1.5
>2
>2.5
Performance Gain

Garden-11 (2)

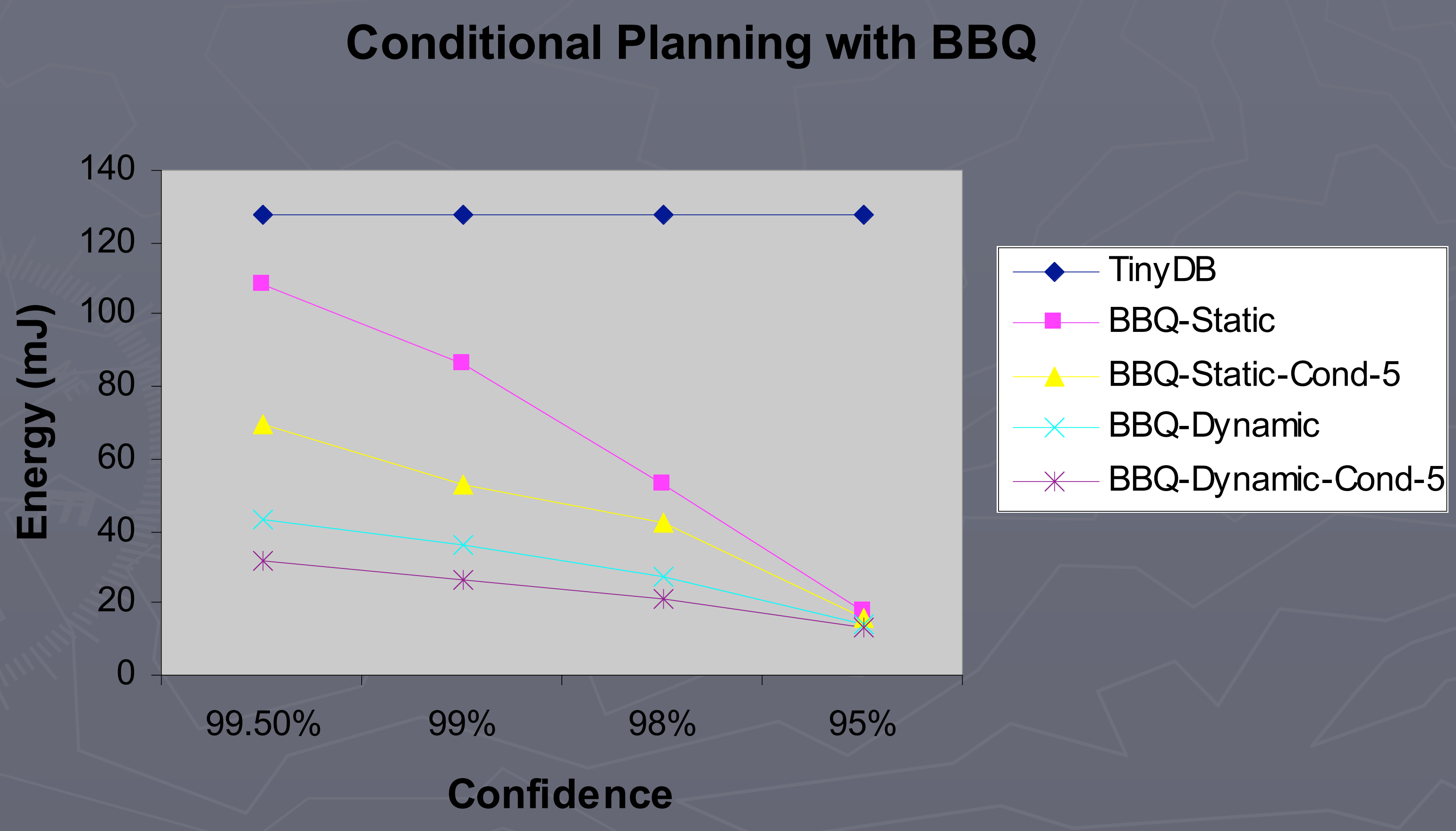

Comparing CorrSeq and Heuristic-10
Fraction of Experiments
1.2
1
0.8
0.6
0.4
0.2
0
>0.85
>1
>1.1
>1.3
>1.5
Performance Gain

Extensions

- Probabilistic queries with confidences
- General queries
  - E.g. Disjunctive queries
- A large class of Join queries
  - E.g. "Star" queries with K-FK join predicates
- Existential queries
  - E.g., "tell me k answers to this query"
    - Can order the observations so that tuples most likely to satisfy are observed first
- Adaptive conditional planning
  - With *eddies*

Conditional Planning for Probabilistic Queries

- Basic idea similar
  - Cost computation is different; typically requires numerical integration



Conclusions

- Large-scale distributed information systems
  - Must acquire data carefully
    - High disparate acquisition costs
  - Exhibit strong temporal and spatial correlations
- Conditional planning
  - Change plans based on observed attribute values
  - Significant benefits even for traditional tasks
- Many other opportunities to exploit such correlations…

Questions ?