
HLS: Tunable Mining for Approximate Functional Dependencies

Jeremy Engle and Edward Robertson

Department of Computer Science, Indiana University, Bloomington, IN USA

{jtengle, edrbtsn}@cs.indiana.edu

Outline: Introduction

- Introduction: An AFD Framework
- Definitions and Background
- HLS: An AFD Search Framework
- Empirical Evaluation
- Questions

Related Work

- FDs and Normalization (1970's-present)
- FDs and Query Optimization (1980's-present)
- AFDs and Query Optimization [5,9] (1990's-present)
- AFD Mining Algorithms [1,8,10,11,12] (1995-2004)
- AFD Data Cleaning, Bayes Classifiers, Query Answering [2,15] (2002-present)

Problem Statement

- Previous mining algorithms only addressed a single application
- We address the need for two types of tuning:
 - Efficiency based on characteristics of data
 - Effectiveness based on needs of application
 - Attribute Pruning
 - Result Ordering
- Our new approach is the development an AFD mining framework

A Framework Approach to AFD Mining

- A framework is a common in software engineering, but is new to AFD mining
- A framework uses components
 1. Allows search algorithm to vary by locality
 2. Allows ordering to vary by need of application
 3. Allows dynamic decisions based on the state of the search
- These advances are possible due to two things
 - HLS- A prototype framework which manages how components work together
 - Lozenge Search- A template for a search algorithm
 - Determines what/where decisions can be made

Outline: Definitions

- Introduction: An AFD Framework
- **Definitions and Background**
- HLS: An AFD Search Framework
- Empirical Evaluation
- Questions

Definition: Dependencies

Let R be a relation's schema and r an instance of R

- A Functional Dependency (FD) says:
 $X \rightarrow Y$ where $X, Y \subseteq R$
For all tuples $t_1, t_2 \in r$, $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$
- Real data contains rules which are “close” to being an FD
- Approximate Functional Dependencies (AFD) are a formalization of this closeness

AFD Approximation

- An *approximation measure* is a function from two attribute sets, X and Y , to $[0, 1]$, written $\varphi(X \rightarrow Y)$
- $\varphi(X \rightarrow Y) = 0$ iff the FD $X \rightarrow Y$ holds
- We use a $[0, 1]$ threshold, ϵ , and mine for minimal AFDs which satisfy $\varphi(X \rightarrow Y) \leq \epsilon$
- **Monotonicity Property:** $X \subset W \Rightarrow \varphi(X \rightarrow Y) \geq \varphi(W \rightarrow Y)$

Outline: HLS

- Introduction: An AFD Framework
- Definitions and Background
- **HLS: An AFD Search Framework**
- Empirical Evaluation
- Questions

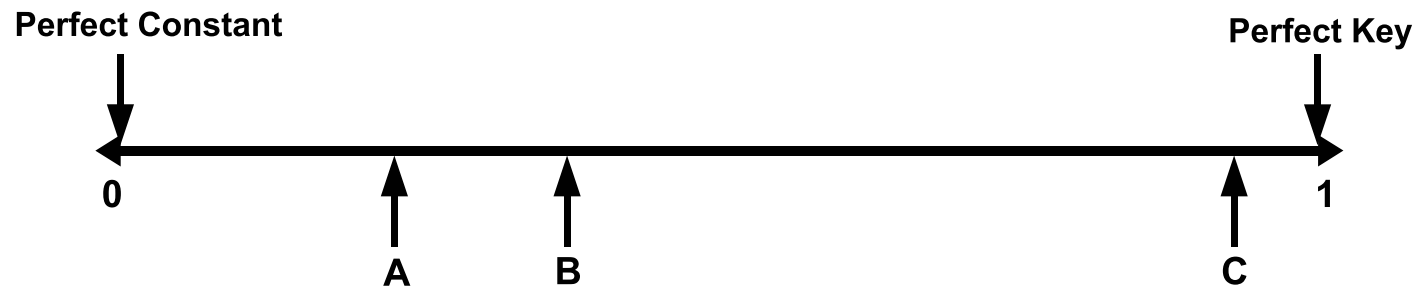
Heuristic Lozenge Search Abilities

- Creates a new strategy based on the space to be searched in each iteration (lozenge)
- A lozenge is the result of adding an attribute to an existing space
- Order of iteration an important decision
- Search within a lozenge can be chosen dynamically

Psuedocode

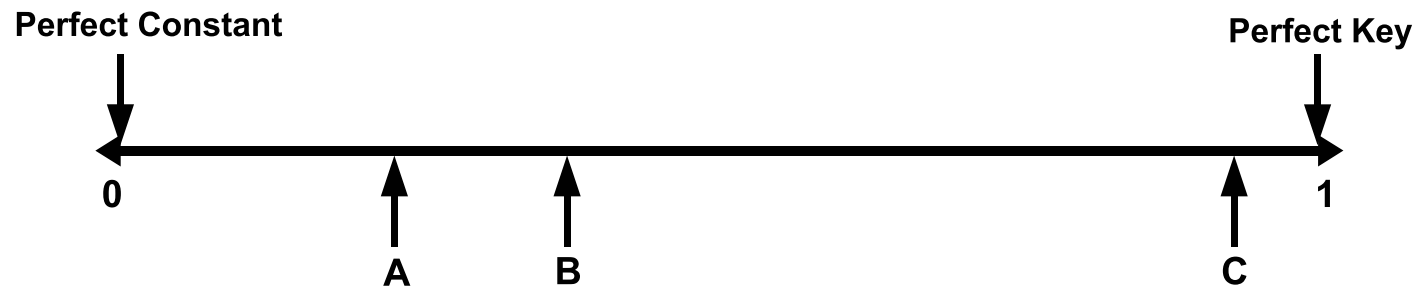
- 1 pick attribute as new for each in R
- 2 for “ $X \rightarrow A$ ” in carryForward
- 3 add “ $X \cup \text{new} \rightarrow A$ ” to queue
- 4 add “act \rightarrow new” to queue
- 5 evaluateLozenge(queue)
- 6 add new to act

Lozenge Ordering with `pick`



- Coreness- Closest to the middle(B,A,C) tunes for an application

Lozenge Ordering with `pick`



- Coreness- Closest to the middle(B,A,C) tunes for an application
- Keyness- Starts from right (C,B,A) tunes for efficiency

Enforcing minimality

- A minimal rule has no attribute which can be removed from LHS and still satisfy the ϵ threshold
- Results of previous lozenges effect minimality in the current lozenge
- `carryForward` list stores required history
- Rules on `carryForward` are used to prune a lozenge of inferred non-minimal pass rules

Pseudocode

1 pick attribute as new for each in R

2 for “ $X \rightarrow A$ ” in carryForward

3 add “ $X \cup \text{new} \rightarrow A$ ” to queue

4 add “act \rightarrow new” to queue

5 evaluateLozenge(queue)

6 add new to act

Searching within a Lozenge

- HLS does not specify how a lozenge is searched (e.g. top-down, bottom-up, DFS,BFS)
- HLS allows local choices thus improving efficiency
- By using a priority queue the priority function allows different search algorithms to be used in a lozenge
- Future Work: How to choose different search algorithms

Pseudocode

- 1 pick attribute as new for each in R
- 2 for “ $X \rightarrow A$ ” in carryForward
- 3 add “ $X \cup \text{new} \rightarrow A$ ” to queue
- 4 add “act \rightarrow new” to queue
- 5 evaluateLozenge(queue)
- 6 add new to act

Outline: Evaluation

- Introduction: An AFD Framework
- Definitions and Background
- HLS: An AFD Search Framework
- **Empirical Evaluation**
- Questions

Data sets and Cost Measures

- Ordering Heuristics Evaluated: Coreness, Keynes, and Random
- Test Data Sets
 - Reed, UWM, Washington, WSU (universities' course listing information)
 - LowInfo (Artificial)
- Machine independent cost measure
 - `numRulesVisited` number of rules in the lattice that are evaluated

Efficiency: HLS vs. Traditional BFS

Dataset		BFS	HLS			HLS Keyness improves BFS
Name	avgLHS		Keyness	Random	Coreness	
Reed	2.51	1178	1260	1858	2770	-7.0%
Washington	2.49	664	704	1275	3901	-6.0%
UWM	2.71	1275	1155	2369	3421	9.4%
LowInfo	5.98	5682	3796	3792	8408	33.2%
WSU	3.6	14074	8457	24890	27581	39.9%

Table 1: numRulesVisited by Algorithm Type

- Ordering of lozenges effects efficiency
- Pruning using `carryForward` allows new levels of efficiency
- Longer rules in the result set increase HLS's efficiency

Need for Local Decisions

Attribute		10		11	
Keyness		.1137		.6662	
Side of rules with <code>new</code>		LHS	RHS	LHS	RHS
<code>numRulesVisited</code>	Top-down	1638	340	3820	66
	Bottom-up	1638	466	3820	1023

Table 2: Costs of `new` by side on LowInfo using HLS Coreness

- The effect of `new` differs whether it is on LHS or RHS
- Dynamically choosing search behaviors can provide new levels of efficiency
- HLS is more likely to use this because lozenges grow in size

Conclusion

- A framework approach allows a family of problems to be solved
- An iterative lozenge based approach with dynamic decisions can improve efficiency and effectiveness
- Need to investigate how different search techniques can offset the cost of less efficient lozenge orderings

Questions?