Optimization of Constrained Frequent Set Queries with 2-variable Constraints

Presented by Joyce Chen

Why?

- Supporting various kinds of rules including associations, constraint-based, human-centered, etc;
- Optimizing the computation of constrained frequent sets is critical to the success of a system that supports ad-hoc mining;
- Maintaining the constrained frequent set queries (CFQs), and effective pruning optimizations for CFQs with 1-variable constraints

Why?

- sum(S.Prince)<=100 and avg (T. Price)>=200;
- In real world, many natural examples of CFQs need to constrain both antecedent and consequent;
- sum(S.Prince) <= avg (T.Price)

Contributions

- Introduce a notion of ***quasi-succinctness****;*
- Characterize the class of 2-variable constraints that are *quasi-succinct;*
- Develop heuristic techniques for **non-*quasi-succinct*** constraints;
- Propose a query optimizer for CFQs: **ccc-optimality**

CFQ

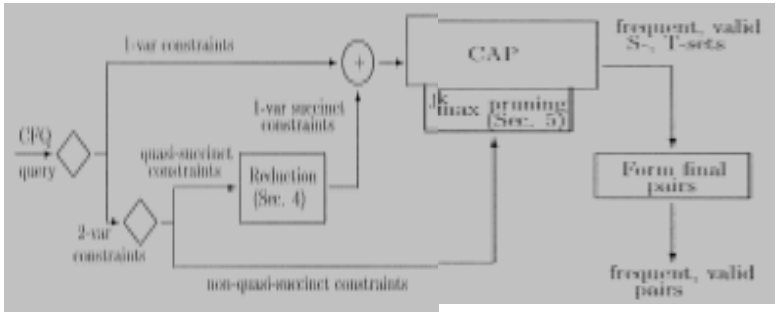
- The primary element in processing constrained mining queries is to compute frequent sets that satisfy the specified constraints;
- Constrained frequent set queries (CFQ);
- {(S, T) | C};
- Its output will be all pairs of frequent sets (S, T) satisfying C

Examples of CFQs

$$\{(S,T) \mid count(S.\text{Type}) = 1 \;\&\; count(T.\text{Type}) = 1 \;\&\; S.\text{Type} \neq T.\text{Type}\}$$

- {(S, T)|S.Type = {Snacks} & T.Type = {Beers} & max(S.Price) <= min(T.Price)};
- {(S, T)|S.Type = {Snacks} & T.Type = {Beers} & sum(S.Price) <= 100 & avg(T.Price) >= 200};

A Schematic Diagram of the CFQ Query Optimizer



1-var anti-monotonicity

- A 1-var constraint C is ***anti-monotone*** iff for any set S: S does not satisfy C => $\forall S' \supseteq S$, S' does not satisfy C;
- For any 1-var constraint C, its solution space **SATc(Item)** is the set consisting of all the subsets of Item that satisfy C;
- Refer to the elements of **SATc(Item)** as **valid sets** w.r.t C;
- For example, $\text{SAT}_{C_3}^{S}$(Item)

Succinctness

- I ⊆ Item is a succinct set if it can be expressed as σ_p(Item) for some selection predicate p;
- SP ⊆ 2^{Item} is a succinct powerset if there is a fixed number of succinct sets $\text{Item}_1, \ldots, \text{Item}_k \subseteq$ Item such that SP can be expressed in terms of the strict powersets of $\text{Item}_1, \ldots, \text{Item}_k$ using union and minus;
- A 1-var constraint C is succinct provided SATc(Item) is a succinct powerset

Anti-monotonicity for 2-var Constraints

- Observation
- The solution space of a 2-var constraint C(S, T) is given by:

  SATc(Item, Dom)={(So, To)|So ⊆ Item & To ⊆ Dom & (So, To) satisfies C};
- We refer (So, To) pairs together satisfying C as the **valid pairs** w.r.t. C.

Con't

- **Valid S-sets**: For a given 2-var constraint C, the set of all valid S-sets w.r.t. C is: SAT^S_C (Item) = {$S_0 \mid \exists T_0 : f_{eq}(T_0)$ & $(S_0, T_0) \in \text{SAT}_C(\text{Item}, \text{Dom})$};
- The set of all valid T-sets w.r.t C can be defined similarly;

> **Definition 4 (2-var anti-monotonicity)** A 2-var constraint $C(S,T)$ is *anti-monotone* with respect to S iff for any S-set S_0 such that for some integer j, the pair (S_0,T) violates C, for all frequent T-sets T of size $\leq j$, it is the case that for all supersets S' of S_0, the pair (S',T') violates C, for all frequent T-sets T' of any size, i.e. $S_0 \notin \text{SAT}^S_{C,j}(\text{Item}) \implies \forall S' \supseteq S_0, S' \notin \text{SAT}^S_C(\text{Item})$.

- Anti-monotonicity w.r.t T can be similarly defined

Con't

- For example, S.A ∩ T.B = ∅ is an anti-monotone 2-var constraint w.r.t. both S and T;
- Anti-monotone 2-var constraints can lead to effective pruning;
- Problem: very few 2-var constraints are anti-monotone;
- Only 2 constraints inside the table are anti-monotone 2-var constraint

Figure1: Characterization of 2-var Constraints: Anti-Monotonicity and Quasi-Succinctness

2-var Constraint	Anti-Monotone	Quasi-Succinct
$S.A \cap T.B = \emptyset$	yes	yes
$S.A \cap T.B \neq \emptyset$	no	yes
$S.A \subseteq T.B$	no	yes
$S.A \not\subseteq T.B$	no	yes
$S.A = T.B$	no	yes
$max(S.A) \leq min(T.B)$	yes	yes
$min(S.A) \leq min(T.B)$	no	yes
$max(S.A) \leq max(T.B)$	no	yes
$min(S.A) \leq max(T.B)$	no	yes
$sum(S.A) \leq max(T.B)$	no	no
$sum(S.A) \leq sum(T.B)$	no	no
$avg(S.A) \leq avg(T.B)$	no	no

Quasi-succinctness

- Prune for a variable in a 2-var constraint, and have the other variable fixed – this forms the basis for the concept of quasi-succinctness;

L_j^S to denote the set of all elements contained in any frequent S-set of size j, i.e. $L_j^S = \{e \mid \exists S : S \subset \texttt{Item} \ \&\ freq(S) \ \&\ |S| = j \ \&\ e \in S\}$

Let CS be a candidate S-set, i.e. $CS \subset \texttt{Item}$.
Then: $\exists$ a frequent T-set T such that $CS.A \cap T.B = \emptyset \implies CS.A \not\supseteq L_1^T.B$

Figure 2: Quasi-succinctness: Reduction of 2-var Domain Constraints

2-var constraint C	sound & tight $C_1(S)$	sound & tight $C_2(T)$
$S.A \cap T.B = \emptyset$	$CS.A \not\supseteq L_1^T.B$	$CT.B \not\supseteq L_1^S.A$
$S.A \cap T.B \neq \emptyset$	$CS.A \cap L_1^T.B \neq \emptyset$	$CT.B \cap L_1^S.A \neq \emptyset$
$S.A \subset T.B$	$CS.A \subseteq L_1^T.B$	$L_1^S.A \cap CT.B \neq \emptyset$
$S.A \not\subseteq T.B$	$(CS \neq \emptyset)$	$L_1^S.A \not\subseteq CT.B$
$S.A = T.B$	$CS.A \subseteq L_1^T.B$	$CT.B \subseteq L_1^S.A$

Con't

- Each 1-var constraint on the second and third column of the table can be regarded as a pruning condition for candidate S-sets;
- Each of them gives a sound and tight pruning condition;
- A pruning condition C1 is sound w.r.t. the original 2-var constraint C, if it does not prune away any valid S-set;
- A pruning condition C1 is tight w.r.t. the original 2-var constraint C, if it prunes away every S-set that is not valid

Con't

- The same method will apply to T-set;
- Formal definition of Quasi-succinctness:

  Constraint C(S, T) is quasi-succinct if it can be reduced to two 1-var constraints C1(S), C2(T) such that:

  - C1, involving only the variable S, is succinct, and is a sound and tight pruning condition for candidate S-sets; and

  - C2, involving only T, is succinct, and is a sound and tight pruning condition for candidate T-sets

Con't

- Succinct 1-var constraints can operate in a generate-only environment, not in a generate-and-test environment. Therefore, efficiency will be maximized;
- Note: $L_1^T.B$ and $L_1^S.A$ are the constants in the constraints of the Quasi-succinctness Table;
- Each sound and tight pruning condition does not necessarily have the same pruning power;
- Aggregation constraints involve only min() and max()

Figure3: Quasi-succinctness: Reduction of min() and max() Constraints

2-var constraint C	sound & tight $C_1(S)$	sound & tight $C_2(T)$
$min(S.A) \leq min(T.B)$	$min(CS.A) \leq max(L_1^T.B)$	$min(CT.B) \geq min(L_1^S.A)$
$min(S.A) \leq max(T.B)$	$min(CS.A) \leq max(L_1^T.B)$	$max(CT.B) \geq min(L_1^S.A)$
$max(S.A) \leq min(T.B)$	$max(CS.A) \leq max(L_1^T.B)$	$min(CT.B) \geq min(L_1^S.A)$
$max(S.A) \leq max(T.B)$	$max(CS.A) \leq max(L_1^T.B)$	$max(CT.B) \geq min(L_1^S.A)$

Figure1: Characterization of 2-var Constraints: Anti-Monotonicity and Quasi-Succinctness

2-var Constraint	Anti-Monotone	Quasi-Succinct
$S.A \cap T.B = \emptyset$	yes	yes
$S.A \cap T.B \neq \emptyset$	no	yes
$S.A \subset T.B$	no	yes
$S.A \not\subseteq T.B$	no	yes
$S.A = T.B$	no	yes
$max(S.A) \leq min(T.B)$	yes	yes
$min(S.A) \leq min(T.B)$	no	yes
$max(S.A) \leq max(T.B)$	no	yes
$min(S.A) \leq max(T.B)$	no	yes
$sum(S.A) \leq max(T.B)$	no	no
$sum(S.A) \leq sum(T.B)$	no	no
$avg(S.A) \leq avg(T.B)$	no	no

Optimizing 2-var Constraints Involving sum() and avg()

- The constraints involving sum() and avg() are called non-quasi-succinct constraints;
- 2 approaches to optimize non-quasi-succinct constraints;
- induce weaker 2-var constraints that are quasi-succinct;
- develop an iterative heuristic pruning algorithm

Inducing Weaker Quasi-succinct Constraints

- C≡sum(S.A) <= max(T.B) implies the weaker constraint:

  C'≡max(S.A) <= max(T.B)

  - CS $\in SAT_{C'}^{S}$(Item)=>CS $\in SAT_{C'}^{S}$(Item)
  - i.e. CS violating C', CS must also violate C (sound);
  - However, these are only sound pruning conditions for candidate S-sets and T-sets w.r.t. C since the pruning is not tight (by def. of tight). Therefore, an additional verification against C must be performed

Con't

- C≡avg() <= agg() induce C'≡min()<=agg();
- C≡sum() <= agg() induce C≡max()<=agg();
- C≡agg() <= avg() induce C≡agg()<=max();
- Based on this idea, following table shows the sound pruning conditions C1(S) and C2(T)

Figure 4: Induced Weaker Constraints for Constraints involving sum() and/or avg()

2-var constraint C	induced weaker constraint C'	sound $C_1(S)$
$avg(S.A) \leq min(T.B)$	$min(S.A) \leq min(T.B)$	$min(CS.A) \leq max(L_1^T.B)$
$sum(S.A) \leq max(T.B)$	$max(S.A) \leq max(T.B)$	$max(CS.A) \leq max(L_1^T.B)$
$avg(S.A) \leq avg(T.B)$	$min(S.A) \leq max(T.B)$	$min(CS.A) \leq max(L_1^T.B)$

sound $C_2(T)$
$min(CT.B) \geq min(L_1^S.A)$
$max(CT.B) \geq min(L_1^S.A)$
$max(CT.B) \geq min(L_1^S.A)$

Heuristic Iterative Pruning with J^k_{max}

- Problem with induced weaker constraints;
- For given all the frequent T-sets of size k for some k>=2, what is an upper bound on the size of the largest frequent T-set?
- The procedure of computing an Upper Bound on Largest Frequent T-set, J^k_{max} :

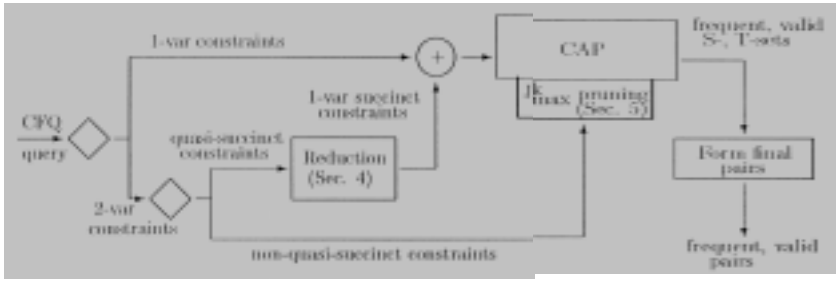
1. Set N_i^k to be the number of frequent sets of size k containing t_i.
2. Set J_i^k to be the largest j with: $N_i^k \geq \binom{k+j-1}{k-1}$ (1)
3. Set J^k_{max} to be $max\{J_i^k \mid 1 \leq i \leq m\}$.

Con't

- To compute iterative pruning for the constraint sum(S.A)<=sum(T.B), by producing a series of 1-var constraints sum(S.A)<= V^i, 1<=i=k, where the upper bounds get tighter with i increases;

1. For an arbitrary element $t_i \in L_k^T$, among all frequent T-sets of size k containing t_i, let the set T_i^k be the one with the maximum value of $sum(T.B)$. Let that sum be Sum_i^k.
2. Let E_i^k be the set of all elements of L_k^T that are not in T_i^k but co-occurring with t_i in some frequent set of size k. Let $e_1, \ldots, e_w$ be an enumeration of all the elements in E_i^k in descending order of their B-values, i.e. $e_1.B \geq \ldots \geq e_w.B$.
3. Set $MaxSum_i^k$ to be $Sum_i^k + \sum_{u=1}^{J^k_{max}} e_u.B$.
4. Set V^k to be $max\{MaxSum_i^k \mid 1 \leq i \leq m\}$.

A Schematic Diagram of the CFQ Query Optimizer



A CFQ Query Optimizer

- A schematic diagram of the optimizer;
- To measure performance – consider 2 cost components;
- What "ccc" stands for?

Definition 6 (ccc-optimality) A computation strategy is *ccc-optimal* for a class of constraints provided for every set $\mathcal{C}$ of constraints from that class, it satisfies the following conditions:

(1) the strategy counts for the support of a candidate set CS iff: all subsets of CS are frequent, and CS is valid;
(2) the strategy invokes the constraint checking operation on a candidate S-set CS, only if $|CS| = 1$.

Con't

- The first condition of ccc-optimality ensures its subset is frequent. It satisfies all 1-var constraints in C. There exists a frequent T-set , such that (CS, CT) is a valid pair for the 2-var constraint;
- The second condition checks the constraint;
- In the most cases, Apriori$^+$ is not ccc-optimal;
- However, if C consists of only 2-var quasi-succinct constraints where the variables S and T effectively point to the same lattice computation, then it is ccc-optimal

Con't

- Algorithm CAP is ccc-optimal for the class of 1-var succinct constraints;
- The general idea is for any set of 1-var succinct constraints, including non-anti-monotone, there is a corresponding function, called the member generating function (MGF), that can generate exactly those sets that satisfy the constraints. This satisfies the second condition of ccc-optimality. Together with the MGF, CAP guarantees the first condition of ccc-optimality is also met;
- The strategy generated by the CFQ query optimizer is ccc-optimal for the class of constraints consisting of 1-var succinct and 2-var quasi-succinct constraints;
- The strategy generated by the optimizer is not ccc-optimal for non-quasi-succinct constraints

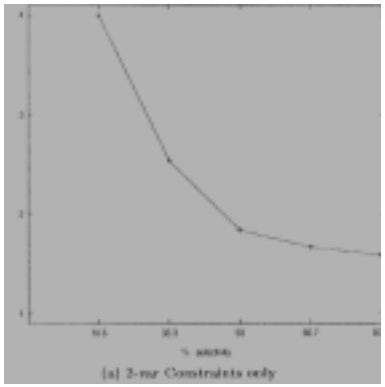
Experimental Evaluation

- For Quasi-succinctness: 2-var constraints only;
- Columns of table – sizes of the frequent sets;
- Each entry is a/b, where a is number of frequent sets satisfying the 1-var succinct constraint, and b is total number of frequent sets (by Apriori$^+$)

	L_1	L_2	L_3	L_4	L_5	L_6
for S	425/425	153/372	54/179	21/122	6/48	1/8
for T	402/402	112/414	8/181	0/123	0/48	0/8

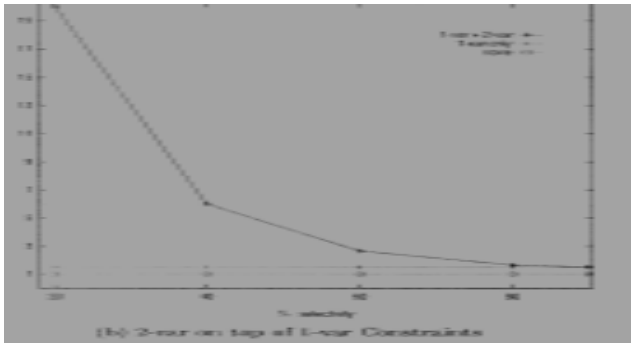
Con't (2-var constraints only)

- The speedup is about 4 times with 16.6% overlap;
- The speedup is over 1.5 times with 83.4% overlap;



Con't (2-var constraints together with 1-var constraints)

- Comparison of Apriori^+ , CAP algorithm (only optimizes the first two 1-var constraints), and the strategy of exploiting the quasi-succinct 2-var constraints as well as 1-var;



Con't (2-var constraints together with 1-var constraints)

- X-axis is the percentage overlap between T and S, and y-axis gives the speedup relative to the Apriori^+ algorithm;
- CAP only checks 2-var constraint at the end. Thus, it also gives a horizontal line;
- Optimizing 1-var constraints alone gives a speedup of 1.5 times. If quasi-succinctness is applied to 2-var constraint, there is a significant speedup

Con't (Optimizing sum() and avg() Constraints with J^k_{max})

- Experiment is done on iterative pruning with J^k_{max};
- Following table shows the speedup with different mean T.Price

Mean of T.Price	Speedup with J^k_{max}
400	3.14 times
600	1.91 times
800	1.36 times
1000	1.11 times

Conclusions

- Developing pruning optimizations for 2-var constraints;
- Discovering the truth that there are very few 2-var constraints are anti-monotone;
- Introducing the notion of quasi-succinctness and completely characterize the class of all such constraints;
- Reducing the quasi-succinctness into two 1-var constraints;
- Developing weaker quasi-succinct constraints and iterative pruning strategy for non-quasi-succinct;
- Proposing a query optimizer for CFQs

THANK YOU