APPENDIX A
PROPERTIES OF THE LIFT MEASURE

We describe some properties of the lift measure that underlie the pruning strategies of our algorithm. As described in the next section, FBIMINER performs a depth-first search for forbidden itemsets. To make this search efficient, pruning strategies should be in place that discard all supersets of a particular itemset, i.e., prune the entire search tree underneath the current node. For this purpose, we derive properties that must hold for all subsets of a τ-forbidden itemset. If such a property does not hold for an itemset, then clearly none of its supersets may be τ-forbidden.

The development of pruning strategies for the lift measure is quite challenging. Typically, pattern mining algorithms perform pruning by leveraging monotonicity properties of interesting measures. The lift measure, however, is neither monotonic nor anti-monotonic. Nevertheless, since a low lift itemset requires that an itemset occurs *much less often* than its subsets, we can use the relation between the support of a τ-forbidden itemset and the support of its subsets for pruning.

The challenge is therefore to obtain bounds on the support of an itemset. In general, given a dataset $\mathcal{D}$ and an itemset I occurring in $\mathcal{D}$, it trivially holds that $1 \leq \mathsf{supp}(I,\mathcal{D}) \leq |\mathcal{D}|$. However, when discovering itemsets in a specific dataset, we can obtain a tighter upper bound on the support of itemsets, assuming that the support of individual items is known. Indeed, $\mathsf{supp}(I,\mathcal{D}) \leq \mathsf{supp}(\{i\},\mathcal{D})$ for any $i \in J$ such that $I \subseteq J$. Let σ_I^{max}, denoting the highest support of an item $\{i\}$ in I or any of I's supersets J. That is,

$$\sigma_I^{max} := \max\{\mathsf{supp}(\{i\},\mathcal{D}) \mid i \in J, I \subseteq J\}.$$

In other words, σ_I^{max} is the highest support of a single item in I's branch of the search tree. Clearly, it holds that $\mathsf{supp}(I,\mathcal{D}) \leq \sigma_J^{max}$, and $\sigma_J^{max} \leq \sigma_I^{max}$ for all $I \subset J$.

We first derive a lower bound on the support of subsets of J, when J is a τ-forbidden itemset:

Proposition 6. *For itemsets I and J such that $I \subset J$, if J is a τ-forbidden itemset then* $\mathsf{supp}(I,\mathcal{D}) \geq \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\sigma_I^{max} \times \tau}$. □

Proof: We show this by contradiction. Let J be a τ-forbidden itemset for $\tau < 1$ and assume for the sake of contradiction that there exists a $I \subset J$ with $\mathsf{supp}(I,\mathcal{D}) < \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\sigma_I^{max} \times \tau}$. Then observe the following:

$$\tau \geq \mathsf{lift}(J,\mathcal{D}) \geq \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\mathsf{supp}(J \setminus I,\mathcal{D}) \times \mathsf{supp}(I,\mathcal{D})}.$$

Using our assumption that $\mathsf{supp}(I,\mathcal{D}) < \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\sigma_I^{max} \times \tau}$, we get

$$\begin{aligned}\tau &> \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\mathsf{supp}(J \setminus I,\mathcal{D}) \times \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\sigma_I^{max} \times \tau}} \\ &= \frac{\cancel{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})} \times (\sigma_I^{max} \times \tau)}{\mathsf{supp}(J \setminus I,\mathcal{D}) \times \cancel{(|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D}))}} \\ &= \frac{\sigma_I^{max} \times \tau}{\mathsf{supp}(J \setminus I,\mathcal{D})}.\end{aligned}$$

Since $\mathsf{supp}(J \setminus I,\mathcal{D}) \leq \sigma_J^{max} \leq \sigma_I^{max}$, we thus have that $\tau > \tau$, which is clearly impossible. Hence, every strict subset I of a τ-forbidden itemset J must satisfy $\mathsf{supp}(I,\mathcal{D}) \geq \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D})}{\sigma_I^{max} \times \tau}$. □

Furthermore, for any τ-forbidden itemset J in the dataset, it trivially holds that $\mathsf{supp}(J,\mathcal{D}) \geq 1$ and thus any itemset $I \subset J$ must have $\mathsf{supp}(I,\mathcal{D}) \geq \frac{|\mathcal{D}|}{\sigma_I^{max} \times \tau}$. This implies that in the depth-first search, it suffices to expand only itemsets I for which $\mathsf{supp}(I,\mathcal{D}) \geq \frac{|\mathcal{D}|}{\sigma_I^{max} \times \tau}$.

Proposition 6 can also be leveraged to show that a minimum reduction in support between subsets of a τ-forbidden itemset is required. The following proposition allows us to prune the search tree when we encounter an itemset J having a subset whose support is too close to J's support.

Proposition 7. *For any three itemsets I, J and K such that $I \subset J \subseteq K$ holds, if K is a τ-forbidden itemset, then we have that* $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}) \geq \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$. □

Proof: We show this by contradiction. Let K be a τ-forbidden itemset for $\tau < 1$ and assume for the sake of contradiction that there exist subsets $I \subset J \subseteq K$ with $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}) < \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$. To simplify notation, we denote $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D})$ by δ. We may then rewrite $\mathsf{supp}(K,\mathcal{D})$ as

$$\mathsf{supp}(K,\mathcal{D}) = \mathsf{supp}(K,\mathcal{D}) + \mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}) - \delta$$

Let L denote the subset $K \setminus (J \setminus I)$ of K. We have that $\mathsf{supp}(L,\mathcal{D}) = \mathsf{supp}(K,\mathcal{D}) - (\mathsf{supp}(J,\mathcal{D}) - \mathsf{supp}(I,\mathcal{D}))$, or equivalently that

$$\mathsf{supp}(L,\mathcal{D}) = \mathsf{supp}(K,\mathcal{D}) + \mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}).$$

Hence, $\mathsf{supp}(K,\mathcal{D}) = \mathsf{supp}(L,\mathcal{D}) - \delta$.

Furthermore, since $\mathsf{lift}(K,\mathcal{D}) \leq \tau$, Prop. 6 tells us that $L \subset K$ must satisfy $\mathsf{supp}(L,\mathcal{D}) \geq \frac{|\mathcal{D}|}{\sigma_I^{max} \times \tau}$. Knowing that $|\mathcal{D}| \geq \sigma_I^{max}$, we simplify the inequality to $\mathsf{supp}(L,\mathcal{D}) \geq \frac{1}{\tau}$. We show that this leads to a contradiction. Indeed, by definition we have that

$$\tau \geq \mathsf{lift}(K,\mathcal{D}) \geq \frac{|\mathcal{D}| \times \mathsf{supp}(K,\mathcal{D})}{\mathsf{supp}(K \setminus L,\mathcal{D}) \times \mathsf{supp}(L,\mathcal{D})}.$$

Since we can replace $\mathsf{supp}(K,\mathcal{D})$ by $\mathsf{supp}(L,\mathcal{D}) - \delta$:

$$\begin{aligned}\tau &\geq \frac{|\mathcal{D}|}{\mathsf{supp}(K \setminus L,\mathcal{D})} \times \frac{\mathsf{supp}(L,\mathcal{D}) - \delta}{\mathsf{supp}(L,\mathcal{D})} \\ &= \frac{|\mathcal{D}|}{\mathsf{supp}(K \setminus L,\mathcal{D})} \times \left(1 - \frac{\delta}{\mathsf{supp}(L,\mathcal{D})}\right),\end{aligned}$$

and by using that $\mathsf{supp}(L,\mathcal{D}) \geq \frac{1}{\tau}$ we have

$$\tau \geq \frac{|\mathcal{D}|}{\mathsf{supp}(K \setminus L,\mathcal{D})} \times \left(1 - \frac{\delta}{1/\tau}\right).$$

Since $\mathsf{supp}(K \setminus L,\mathcal{D}) \leq \sigma_K^{max} \leq \sigma_I^{max}$, we obtain that

$$\tau \geq \frac{|\mathcal{D}|}{\sigma_I^{max}} \times \left(1 - \frac{\delta}{1/\tau}\right).$$

From this, we can infer that $\tau \times \sigma_I^{max} \geq |\mathcal{D}| - (|\mathcal{D}| \times \delta \times \tau)$, and hence, $|\mathcal{D}| \times \delta \times \tau \geq |\mathcal{D}| - (\tau \times \sigma_I^{max})$. A further rearrangement of terms leads to

$$\delta \geq \frac{|\mathcal{D}|}{|\mathcal{D}| \times \tau} - \frac{\tau \times \sigma_I^{max}}{|\mathcal{D}| \times \tau} = \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}.$$

Which gives a contradiction of our assumption that $\delta < \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$. As a consequence, the proposition holds. □

In particular, when applied to $I \subset J \subseteq K = J$, Prop 7 indicates that for J to be τ-forbidden, we must have that $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}) \geq \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$ holds for any of its subsets I. During the depth-first search, when expanding I to J, if this condition is not satisfied, then J and all of its supersets K can be pruned.

Furthermore, it holds that $\frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|} > 0$, since $\sigma_I^{max} \leq |\mathcal{D}|$ and $\tau < 1$. Consequently, the proposition implies that τ-forbidden itemsets must have a support different from the support of all their subsets, i.e., if J is τ-forbidden then $\mathsf{supp}(I,\mathcal{D}) > \mathsf{supp}(J,\mathcal{D})$ for any $I \subset J$. Itemsets J that satisfy this condition are called generators [1] or free itemsets [2]. A known property of generators is that all their subsets are generators as well; hence every subset of a τ-forbidden itemset is required to be a generator. If a non-generator is encountered during the search, the entire subtree can therefore be pruned.

Our next pruning method uses the lift of an itemset to bound the support of its supersets. Indeed, we will show that the *denominator* of the lift measure is in fact anti-monotonic.

Proposition 8. *For any two itemsets I and J such that $I \subset J$, it holds that*

$$\mathsf{lift}(J,\mathcal{D}) \geq \frac{\mathsf{supp}(J,\mathcal{D}) \times |\mathcal{D}|}{\min_{S \subset I}\{\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})\}}. \qquad \square$$

Proof: We show the following: For any two itemsets I and J such that $I \subset J$, it holds that:

$$\min_{S \subset I}\{\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})\} \geq \min_{S \subset J}\{\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(J \setminus S,\mathcal{D})\}.$$

Clearly, this implies the proposition. We next show that the inequality holds. Indeed, this follows from the fact that any expression $\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})$ can be rewritten as

$$\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}((J \setminus S) \setminus X,\mathcal{D}),$$

where $X = J \setminus I$. Since

$$\mathsf{supp}((J \setminus S) \setminus X,\mathcal{D}) \geq \mathsf{supp}(J \setminus S,\mathcal{D}),$$

we obtain that the following inequality holds for any $S \subset I$,

$$\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D}) \geq \mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(J \setminus S,\mathcal{D}),$$

as desired. $\square$

Clearly, this lower bound can be used to prune itemsets J that cannot lead to τ-forbidden itemsets. When considering an itemset $I \subset J$ during our depth-first traversal, we already have the supports of subsets of I at our disposal. Hence, although $\mathsf{supp}(J,\mathcal{D})$ is not yet know, the value of the denominator is known when I is considered. This implies that we need a lower bound on $\mathsf{supp}(J,\mathcal{D})$ to lower bound $\mathsf{lift}(J,\mathcal{D})$ using Prop. 8. We again use the trivial lower bound $\mathsf{supp}(J,\mathcal{D}) \geq 1$ and prune

Finally, since itemsets with low lift are obtained when they occur much less often than their subsets, it is expected that such forbidden itemsets will have a low support. In fact, one can precisely characterize the maximal frequency of a τ-forbidden itemset.

Proposition 9. *If I is a τ-forbidden itemset then its frequency is bounded by $\mathsf{freq}(I,\mathcal{D}) \leqslant \frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$. Furthermore, for small τ-values this bound converges (from above) to $\frac{\tau}{4}$.* $\square$

Proof: Consider an itemset I which is τ-forbidden for a given threshold $\tau < 1$. We first lower bound $\mathsf{lift}(I,\mathcal{D})$ by considering the maximal value that $\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})$ can take for $\emptyset \subset S \subset I$. Observe that $\mathsf{supp}(S,\mathcal{D}) + \mathsf{supp}(I \setminus S,\mathcal{D}) \leq |\mathcal{D}| + \mathsf{supp}(I,\mathcal{D})$. It can now be easily verified that the maximal value of the expression $\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})$ is obtained when $\mathsf{supp}(S,\mathcal{D}) = \frac{|\mathcal{D}| + \mathsf{supp}(I,\mathcal{D})}{2}$.

Since $\mathsf{lift}(I,\mathcal{D}) \leqslant \tau$ we then have that

$$\frac{\mathsf{supp}(I,\mathcal{D}) \times |\mathcal{D}|}{\left(\frac{|\mathcal{D}| + \mathsf{supp}(I,\mathcal{D})}{2}\right) \times \left(\frac{|\mathcal{D}| + \mathsf{supp}(I,\mathcal{D})}{2}\right)} \leqslant \mathsf{lift}(I,\mathcal{D}) \leqslant \tau.$$

We can use this to upper bound $\mathsf{supp}(I,\mathcal{D})$, as follows:

$$\mathsf{supp}(I,\mathcal{D}) \leq \frac{\tau}{4|\mathcal{D}|} \times \left(|\mathcal{D}|^2 + 2|\mathcal{D}| \times \mathsf{supp}(I,\mathcal{D}) + (\mathsf{supp}(I,\mathcal{D}))^2\right).$$

It is now a routine exercise to find the maximal value of $\mathsf{supp}(I,\mathcal{D})$. Indeed, one simply needs to solve the equation

$$\frac{\tau}{4|\mathcal{D}|} \times \left(|\mathcal{D}|^2 + 2|\mathcal{D}| \times \mathsf{supp}(I,\mathcal{D}) + (\mathsf{supp}(I,\mathcal{D}))^2\right) - \mathsf{supp}(I,\mathcal{D}) = 0.$$

We obtain the following two solutions:

$$\mathsf{supp}(I,\mathcal{D})^{\pm} = \frac{2|\mathcal{D}|}{\tau} - |\mathcal{D}| \pm 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}}.$$

Since $\frac{2|\mathcal{D}|}{\tau} - |\mathcal{D}| > |\mathcal{D}|$ for any $\tau < 1$, we see that $\mathsf{supp}(I,\mathcal{D})^+ = \frac{2|\mathcal{D}|}{\tau} - |\mathcal{D}| + 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} > |\mathcal{D}|$. This is impossible since no itemset can have a support strictly greater than $|\mathcal{D}|$. Hence, we are left with the other root $\mathsf{supp}(I,\mathcal{D})^- = \frac{2|\mathcal{D}|}{\tau} - |\mathcal{D}| - 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}}$. To obtain the upper bound on the frequency we divide by $|\mathcal{D}|$. This results in that the maximal frequency of a τ-forbidden itemset is $\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$.

To show the second statement in the proposition, we consider the Taylor expansion of $\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ given by

$$\frac{\tau}{4} + \frac{\tau^2}{8} + \frac{5\tau^3}{64} + \frac{7\tau^4}{128} + O(\tau^5).$$

Hence, for decreasing values of τ, the maximal frequency of a τ-forbidden itemset converges from above to $\tau/4$. $\square$

The proposition tells that for small values of τ, τ-forbidden itemsets are at most approximately $\tau/4$-frequent, and that itemsets whose *frequency* exceeds $\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ cannot be τ-forbidden.

This result can also be used to obtain an initial estimate for τ. Clearly, $\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ should be greater than 0, or no itemsets will be forbidden. When picking a higher τ, the bound indicates how frequent forbidden itemsets can get. If this possible frequency gets substantially high, the chosen value τ is most probably too high. The actual value of $\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ then gives a useful indication of how frequent the discovered forbidden itemsets will be.

APPENDIX B
MINING FORBIDDEN ITEMSETS

We present the algorithm FBIMINER, for mining τ-forbidden itemsets in a dataset $\mathcal{D}$. That is, the algorithm computes $\mathsf{dirty}(\mathcal{D},\mathcal{L},q)$ for the language and predicate defined earlier. The algorithm is based on the well-known Eclat algorithm for frequent itemset mining [3]. Here, we only describe the main differences with Eclat. The pseudo-code of FBIMINER is shown in Alg. 3.

The algorithm takes as input a projected dataset in vertical layout $\mathcal{D}\downarrow[P]$, consisting of all objects which contain a prefix itemset P, and the lift threshold τ. The initial call uses the entire dataset $\mathcal{D}$ in vertical data layout, and an empty prefix itemset $P = \emptyset$. Just like Eclat, FBIMINER employs a depth-first search of the itemset lattice (**for** loop line 3 – 25, and recursive call on

Algorithm 3 An Eclat-based algorithm for mining low lift τ-forbidden itemsets

1: **procedure** FBIMINER($\mathcal{D}\downarrow[P], P \subseteq \mathcal{I}, \tau$)
2: $\mathsf{FBI} \leftarrow \emptyset$
3: **for all** $i \in \mathcal{I}$ occurring in $\mathcal{D}\downarrow[P]$ in reverse order **do**
4: $I \leftarrow P \cup \{i\}$
5: **if** not ISGENERATOR(I) **then**
6: **continue**
7: STOREGENERATOR(I)
8: **if** $|I| > 1$ & $\mathsf{freq}(I,\mathcal{D}) \leq \frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ **then**
9: **if** a subset of I has been pruned **then**
10: **continue**
11: **if** $\mathsf{lift}(I,\mathcal{D}) \leqslant \tau$ **then**
12: $\mathsf{FBI} \leftarrow \mathsf{FBI} \cup \{I\}$
13: **if** $\frac{|\mathcal{D}|}{\tau} > \min_{S \subset I}\{\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})\}$ **then**
14: **continue**
15: **if** $\mathsf{supp}(I,\mathcal{D}) < \frac{|\mathcal{D}|}{\sigma_I^{max} \times \tau}$ **then**
16: **continue**
17: $\mathcal{D}\downarrow[I] \leftarrow \emptyset$
18: **for all** $j \in \mathcal{I}$ in $\mathcal{D}$ such that $j > i$ **do**
19: $C \leftarrow \mathsf{cov}(\{i\},\mathcal{D}) \cap \mathsf{cov}(\{j\},\mathcal{D})$
20: $J \leftarrow I \cup \{j\}$
21: $\mathsf{supp}(J,\mathcal{D}) \leftarrow |C|$
22: **if** $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(J,\mathcal{D}) \geq \frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$ **then**
23: **if** $\mathsf{supp}(J,\mathcal{D}) > 0$ **then**
24: $\mathcal{D}\downarrow[I] \leftarrow \mathcal{D}\downarrow[I] \cup \{(j,C)\}$
25: $\mathsf{FBI} \leftarrow \mathsf{FBI} \cup \mathrm{FBIMINER}(\mathcal{D}\downarrow[I], I, \tau)$
26: **return** FBI

line 25). When expanding an itemset I in the search tree (line 4), it is processed (line 5 – 16), before the next layer of the search tree is generated (line 17 – 24). New itemsets are generated by extending I with all items in the dataset that occur in the objects in I's cover (lines 18 – 24). To efficiently compute the support of the new itemsets, Eclat uses set intersections of the covers of the items (line 19). If the size if the newly computed cover C is higher than zero, i.e., the new itemset J is supported, it is added to $\mathcal{D}\downarrow[I]$, the dataset $\mathcal{D}$ in vertical layout projected on I. When generating new itemsets, the candidate itemset j are added according to a total ordering on the items, i.e., items are only added when they come after each item in I (line 18). Items are ordered by ascending support, as this is known to improve efficiency. More details about efficient implementation strategies for Eclat are discussed in [4].

A first challenge is to tweak the Eclat algorithm such that the lift of itemsets can be computed. Observe that the lift of an itemset is dependent on the support of *all* of its subsets. For this purpose, we use the same depth-first traversal as Eclat, but traverse it in reverse order (line 3). Indeed, such a reverse pre-order traversal of the search space visits all subsets of an itemset I before visiting I itself [5]. This is exactly what is required to compute the lift measure, provided that the support of each processed itemset is stored. However, Eclat generates a candidate itemset based on only two of its subsets [3]. Hence, the supports of all subsets of an itemset are not immediately available in the search tree. To remedy this, we store the support of the processed itemsets using a prefix tree, for time- and memory-efficient lookup during lift computation.

Having integrated lift computation in the algorithm, we next turn to our pruning and optimization strategies. We deploy four pruning strategies (lines 9, 13, 15 and 22). The first strategy (line 9) applies to itemsets J for which the lift cannot be computed, which happens when some of its subsets are pruned away in an earlier step. The absence of subsets is detected when the lift computation requests the support of a subset that is not stored. Recall that we derived pruning properties of the lift measure, that must hold for all subsets of a forbidden itemset. If a subset of I was pruned because of these properties, the subset did not satisfy the required property. This implies that I (and all its supersets J) cannot be τ-forbidden and thus can be pruned (line 10).

The second pruning strategy (line 13) applies to supersets of itemsets I for which we have been able to compute their lift, and leverages Prop. 8. Indeed, if we have that $\frac{|\mathcal{D}|}{\tau} > \min_{S \subset I}\{\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I \setminus S,\mathcal{D})\}$ then Prop. 8 tells that I cannot be a subset of a τ-forbidden itemset. Hence, all itemsets in the tree below I are pruned (line 14).

By contrast, the third strategy (line 15) leverages Prop. 6 and skips supersets of itemsets I, regardless of whether their lift was computed. Indeed, if $\mathsf{supp}(I,\mathcal{D}) < \frac{|\mathcal{D}|}{\sigma_I^{max} \times \tau}$ holds, then I cannot be part of a τ-forbidden itemset, resulting in a further pruning of the search space.

The fourth strategy employs Prop. 7 to prune extensions of I that do not cause a sufficient reduction in support. When generating direct supersets J of I, we have immediate access to their supports. If the reduction in support is smaller than $\frac{1}{\tau} - \frac{\sigma_I^{max}}{|\mathcal{D}|}$, the proposition implies that J (and its supersets) cannot be τ-forbidden, hence we do not need to add it to the next layer of the search tree. This check is performed prior to the recursive call (line 22).

Finally, we also implement an optimization that avoids certain lift computations (line 8). Only when the algorithm encounters an itemset I with at least two items and a frequency lower than the bound from Prop. 9, the lift of I is computed. All other itemsets cannot be τ-forbidden by Prop. 9. Note that this only eliminates the need for checking the lift of certain itemsets but by itself does not lead to a pruning of its supersets.

A careful reader may have spotted the optimized pruning of non-generators on lines 5-7. Recall that as a direct consequence of Prop. 7, any τ-forbidden itemset must be a generator, i.e., have a support which is strictly lower than that of all of its subsets. The Talky-G algorithm [6] implements specific optimizations for mining such generators, using a hash-based method that was introduced in the Charm algorithm for closed itemset mining [7]. We use the same technique in FBIMINER to efficiently prune non-generators.

More specifically, during the mining process, all encountered generators are stored in a hashmap (procedure STOREGENERATOR on line 7). As hash value we use, just like the Charm algorithm, the sum of the oid's of all objects in which an itemset occurs. If an itemset has the same support as one of its subsets, it is clear that this itemset must occur in exactly the same objects, and will map to the same hash value. Moreover, the probability of unrelated itemsets having the same *sum of oids* is lower than the probability of them having the same support. Therefore this sum is taken as hash value instead of the support of itemsets.

Procedure ISGENERATOR on line 5 checks all stored itemsets with the same hash value as I. If any of these itemsets are a subset of I with the same support as I, then I is discarded as a non-

generator. Furthermore, since all supersets of a non-generators are also non-generators, the entire subtree can be pruned. If no subset with identical support is discovered for an itemset I, then I is either a generator, or a subset with identical support has previously been pruned, in which case I will eventually be pruned on line 9.

We conclude this section by verifying the correctness of FBIMINER.

Proposition 10. *Let $\mathcal{D}$ be a dataset and τ a maximum lift threshold. Algorithm* FBIMINER$(\mathcal{D}\downarrow[\emptyset],\emptyset,\tau)$ *returns all τ-forbidden itemsets in $\mathcal{D}$.*

Proof: Consider FBIMINER from which all pruning steps are removed. In this case, the algorithm will perform an exhaustive depth-first search traversal of all itemsets and only (and all) τ-forbidden itemsets will be added to FBI (lines 11 and 12). In other words, the algorithm correctly computes all τ-forbidden itemsets in $\mathcal{D}$.

With pruning and optimizations enabled, suppose that for the sake of contradiction, there is a τ-forbidden itemsets I that is not returned by FBIMINER. This implies that I was not added to FBI (lines 11 and 12). There are two possible causes for this: (i) the condition $\mathsf{freq}(I,\mathcal{D}) \leqslant \frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ (line 8) is not satisfied; or (ii) a subset J of I has been pruned. Clearly, case (i) cannot happen. Indeed Prop. 9 implies that $\mathsf{lift}(I,\mathcal{D}) > \tau$, a contradiction. Similarly, case (ii) leads to a contradiction. When a subset J of I is pruned, Propositions 6, 7 and 8 imply that all supersets of J, including I, have a lift greater than τ. We may thus conclude that FBIMINER is indeed correct. □

APPENDIX C
PROOF OF PROPOSITION 2

We formally prove Proposition 2 from Section 5.2.

Proposition 2. *Let $\mathcal{D}$ be a dataset, and I and J itemsets such that* $\mathsf{supp}(J,\mathcal{D}) \leq \mathsf{supp}(I\setminus J,\mathcal{D})$.
▷ *If* $\mathsf{supp}(I,\mathcal{D}) > 1$ *and* $\mathsf{supp}(I\setminus J,\mathcal{D}) > 1$, *then*

$$(\ddagger) = \min\{\frac{|\mathcal{D}| \times (\mathsf{supp}(I,\mathcal{D}) - 1)}{\mathsf{supp}(J,\mathcal{D}) \times (\mathsf{supp}(I\setminus J,\mathcal{D}) - 1)}, \frac{|\mathcal{D}| \times \mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D}) + 1) \times \mathsf{supp}(I\setminus J,\mathcal{D})}\}.$$

▷ *Otherwise, if* $\mathsf{supp}(I,\mathcal{D}) = 1$, *or* $\mathsf{supp}(I\setminus J,\mathcal{D}) = 1$, *then*

$$(\ddagger) = \frac{|\mathcal{D}| \times \mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D}) + 1) \times \mathsf{supp}(I\setminus J,\mathcal{D})}.$$

Proof: The idea behind the proof is to show that for all $(\Delta_I,\Delta_J,\Delta_{I\setminus J})$,

$$\frac{|\mathcal{D}| \times (\mathsf{supp}(I,\mathcal{D}) + \Delta_I)}{(\mathsf{supp}(J,\mathcal{D}) + \Delta_J) \times (\mathsf{supp}(I\setminus J,\mathcal{D}) + \Delta_{I\setminus J})}$$

is either larger than $\frac{|\mathcal{D}|\times(\mathsf{supp}(I,\mathcal{D})-1)}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(I\setminus J,\mathcal{D})-1)}$ (corresponding to the triple $(-1,0,-1)$) or larger than $\frac{|\mathcal{D}|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})+1)\times\mathsf{supp}(I\setminus J,\mathcal{D})}$ (corresponding to the triple $(0,+1,0)$). This is verified by a case analysis.

Let $(\Delta_I,\Delta_J,\Delta_{I\setminus J})$ be such that Δ_I is either -1 (decrement), 0 (no change), or $+1$ (increment), and similarly for Δ_J and $\Delta_{I\setminus J}$. We need to show that for all such triples $(\Delta_I,\Delta_J,\Delta_{I\setminus J})$,

$$\frac{|\mathcal{D}| \times (\mathsf{supp}(I,\mathcal{D}) + \Delta_I)}{(\mathsf{supp}(J,\mathcal{D}) + \Delta_J) \times (\mathsf{supp}(I\setminus J,\mathcal{D}) + \Delta_{I\setminus J})}$$

is either larger than $\frac{|\mathcal{D}|\times(\mathsf{supp}(I,\mathcal{D})-1)}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(I\setminus J,\mathcal{D})-1)}$ (corresponding to the triple $(-1,0,-1)$) or larger than $\frac{|\mathcal{D}|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})+1)\times\mathsf{supp}(I\setminus J,\mathcal{D})}$ (corresponding to the triple $(0,+1,0)$).

Although there are $3^3 = 27$ different triples $(\Delta_I,\Delta_J,\Delta_{I\setminus J})$ we can immediately rule out certain combinations which are impossible in practice.

Firstly, if $\Delta_I =$"$+1$", then neither Δ_J nor $\Delta_{I\setminus J}$ can be "-1". Clearly, if a single modification reduces the support of J or $I\setminus J$, the modified object no longer contains these itemsets, and hence cannot contain a new instance of their superset I. Moreover, at least one of Δ_J or $\Delta_{I\setminus J}$ must be "$+1$" for the support of I to increase. This rules out 6 cases: $(+1,0,0)$, $(+1,0,-1)$, $(+1,-1,0)$, $(+1,+1,-1)$, $(+1,-1,+1)$ and $(+1,-1,-1)$.

A similar argument can be made when $\Delta_I =$"-1", in which case neither Δ_J nor $\Delta_{I\setminus J}$ can be "$+1$", and at least one of Δ_J or $\Delta_{I\setminus J}$ must be "-1" for the support of I to decrease. This again rules out 6 cases: $(-1,0,0)$, $(-1,0,+1)$, $(-1,+1,0)$, $(-1,-1,+1)$, $(-1,+1,-1)$ and $(-1,+1,+1)$.

Furthermore, if $\Delta_I =$"0", then Δ_J and $\Delta_{I\setminus J}$ cannot both be "$+1$" or "-1". Indeed, if a modification were to introduce (resp. remove) an occurence of both J and $I\setminus J$, then clearly it must also introduce (resp. remove) an occurrence of their union, $J\cup(I\setminus J) = I$, which contradicts the fact that $\Delta_I = 0$". This rules out 2 more cases, $(0,+1,+1)$ and $(0,-1,-1)$.

Finally, the case when $(\Delta_I,\Delta_J,\Delta_{I\setminus J}) = (0,0,0)$ is not very interesting since it does not change the lift, and will always end up being higher than any modification that reduces the lift. We can therefore ignore this in our case analysis.

This leaves us with the following 12 cases:

(1)	$(0,0,+1)$	(2)	$(0,+1,0)$
(3)	$(0,0,-1)$	(4)	$(0,-1,0)$
(5)	$(0,-1,+1)$	(6)	$(0,+1,-1)$
(7)	$(+1,0,+1)$	(8)	$(+1,+1,0)$
(9)	$(+1,+1,+1)$	(10)	$(-1,0,-1)$
(11)	$(-1,-1,1)$	(12)	$(-1,-1,-1)$

We will show that cases (2) and (10) lead to a maximum possible decrease in lift. Furthermore (2) and (10) are incomparable, i.e., depending on the instance and itemsets under consideration, (2) may lead more decrease in lift than (10), or vice versa, after one modification is made to $\mathcal{D}$. In fact, we will show that cases (1)-(12) form two distinct partial orderings. Let $(i) \leq (j)$ denote that (i) leads to more (or equal) decrease in lift than (j):

$$\begin{array}{ll} (2) & \leq (1) \quad \leq (9) \quad \leq (8) \leq (7) \\ & \leq (6) \quad \leq (5) \\ (10) & \leq (11) \leq (12) \leq (3) \leq (4) \end{array} \qquad (\clubsuit)$$

We will now prove the inequalities that underlie the orderings of the cases. Recall that we assumed that $\mathsf{supp}(I\setminus J,\mathcal{D}) = \mathsf{supp}(J,\mathcal{D}) + a$ for some constant $a \geq 0$.
▷ Consider cases (2) and (1). To show that $(2) \leq (1)$, we verify that

$$\frac{|\mathcal{D}'| \times \mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D}) + 1) \times (\mathsf{supp}(J,\mathcal{D}) + a)} \leq \frac{|\mathcal{D}'| \times \mathsf{supp}(I,\mathcal{D})}{\mathsf{supp}(J,\mathcal{D}) \times (\mathsf{supp}(J,\mathcal{D}) + a + 1)}$$

holds. It is a straightforward exercise to check that this inequality always holds. Indeed, it boils down to verifying that

$$(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a) \geq \mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a+1),$$

which in turn is equivalent to checking that

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a+1)\mathsf{supp}(J,\mathcal{D})+a \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a+1)\mathsf{supp}(J,\mathcal{D}).$$

This always holds since $a\geq 0$. Hence, it holds that $(2)\leq(1)$.

▷ Cases (1) and (9): To show that $(1)\leq(9)$ we verify that

$$\frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a+1)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})+1)}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a+1)}$$

holds. This is equivalent to checking whether

$$\frac{\mathsf{supp}(I,\mathcal{D})}{\mathsf{supp}(J,\mathcal{D})}\leq\frac{\mathsf{supp}(I,\mathcal{D})+1}{\mathsf{supp}(J,\mathcal{D})+1},$$

or

$$\mathsf{supp}(I,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+1)\leq\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(I,\mathcal{D})+1)$$

is true. This holds since $\mathsf{supp}(I,\mathcal{D})\leq\mathsf{supp}(J,\mathcal{D})$.

▷ Cases (9) and (8): To show that $(9)\leq(8)$ we verify that

$$\frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})+1)}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a+1)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})+1)}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a)}$$

holds. This follows immediately from the fact that

$$\frac{1}{\mathsf{supp}(J,\mathcal{D})+a+1}\leq\frac{1}{\mathsf{supp}(J,\mathcal{D})+a}.$$

▷ Cases (8) and (7): To show that $(9)\leq(8)$ we verify that

$$\frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})+1)}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})+1)}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a+1)},$$

holds. This pours down to verifying that

$$(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a) \geq (\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a+1),$$

which in turn is equivalent to checking that

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a+1)\mathsf{supp}(J,\mathcal{D})+a \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a+1)\mathsf{supp}(J,\mathcal{D})$$

holds. This follows again from the fact that $a\geq 0$.

▷ Cases (2) and (6): To show that $(2)\leq(6)$ we verify that

$$\frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a)} \leq \frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a-1)}$$

holds. This is equivalent to checking

$$(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a) \geq (\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a-1),$$

or, in other words, whether

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a+1)\mathsf{supp}(J,\mathcal{D})+a \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+a\,\mathsf{supp}(J,\mathcal{D})+a-1,$$

holds. Clearly, $a+1>a$ and $a>a-1$ and hence this inequality is trivially satisfied.

▷ Cases (6) and (5): To show that $(6)\leq(5)$ we verify that

$$\frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a-1)} \leq \frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a+1)}$$

holds. This pours down to checking

$$(\mathsf{supp}(J,\mathcal{D})+1)\times(\mathsf{supp}(J,\mathcal{D})+a-1) \geq (\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a+1),$$

or equivalently, whether

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+a\,\mathsf{supp}(J,\mathcal{D})+a-1 \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+a\,\mathsf{supp}(J,\mathcal{D})-a-1$$

holds. This inequality is satisfied because $a\geq 0$ and hence $a\geq -a$.

▷ Cases (10) and (11): To show that $(10)\leq(11)$ we verify that

$$\frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})-1)}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a-1)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})-1)}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a)}$$

holds. This is equivalent to checking

$$\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a-1) \geq (\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a)$$

or, whether

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-1)\mathsf{supp}(J,\mathcal{D}) \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-1)\mathsf{supp}(J,\mathcal{D})-a$$

is true. Again, this follows from $a\geq 0$ and hence $0\geq -a$.

▷ Cases (11) and (12): To show that $(11)\leq(12)$ we verify that

$$\frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})-1)}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})-1)}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a-1)}$$

holds. We can equivalently check

$$(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a). \geq (\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a-1),$$

or whether

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-1)\mathsf{supp}(J,\mathcal{D}) \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-2)\mathsf{supp}(J,\mathcal{D})-a+1$$

is true. This inequality holds since $\mathsf{supp}(J,\mathcal{D})\geq 1$ and $a\geq 0$. Indeed, from these we have that $(a-2)\mathsf{supp}(J,\mathcal{D})-a+1\leq(a-1)\mathsf{supp}(J,\mathcal{D})$, from which the inequality follows.

▷ Cases (12) and (3): To show that $(12)\leq(3)$ we verify that

$$\frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D})-1)}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a-1)} \leq \frac{|\mathcal{D}'|\times(\mathsf{supp}(I,\mathcal{D}))}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a-1)}$$

holds. It suffices to show that

$$\frac{\mathsf{supp}(I,\mathcal{D})-1}{\mathsf{supp}(J,\mathcal{D})-1} \leq \frac{\mathsf{supp}(I,\mathcal{D})}{\mathsf{supp}(J,\mathcal{D})}$$

which holds since $\mathsf{supp}(I,\mathcal{D}) \leq \mathsf{supp}(J,\mathcal{D})$.

▷ Cases (3) and (4): To show that (3) $\leq$ (4) we verify that

$$\frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a-1)} \leq \frac{|\mathcal{D}'|\times\mathsf{supp}(I,\mathcal{D})}{(\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a)}$$

holds. This is equivalent to checking

$$\mathsf{supp}(J,\mathcal{D})\times(\mathsf{supp}(J,\mathcal{D})+a-1) \geq (\mathsf{supp}(J,\mathcal{D})-1)\times(\mathsf{supp}(J,\mathcal{D})+a)$$

or

$$\big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-1)\mathsf{supp}(J,\mathcal{D}) \geq \big(\mathsf{supp}(J,\mathcal{D})\big)^2+(a-1)\mathsf{supp}(J,\mathcal{D})-a$$

holds. This follows immediately from $a \geq 0$ and hence $0 \geq -a$.

The previous case comparisons prove the partial orderings (♣). It remains to show that (2) and (10) are not comparable in general. We show this by providing counter examples. Assume first that (2) $\geq$ (10). However, when choosing $\mathsf{supp}(I,\mathcal{D})=2$, $\mathsf{supp}(J,\mathcal{D})=2$ and $a=1$, we have that:

$$\frac{|\mathcal{D}'|\times 2}{3\times 3} < \frac{|\mathcal{D}'|\times 1}{2\times 2}$$

Which contradicts the assumption. Hence, (2) $\not\geq$ (10).

Assume next that (2) $<$ (10). However, when choosing $\mathsf{supp}(I,\mathcal{D})=2$, $\mathsf{supp}(J,\mathcal{D})=2$ and $a=3$, we have that:

$$\frac{|\mathcal{D}'|\times 2}{3\times 5} > \frac{|\mathcal{D}'|\times 1}{2\times 4}$$

Which again contradicts the assumption. Hence, (2) $\not<$ (10), which proves that (2) and (10) are not comparable. This concludes our proof that (2) and (10) are indeed the two cases that cause a maximum reduction in lift. □

APPENDIX D
REPAIR-OBLIVIOUS PRUNING PROPERTIES

As explained in Section 5.2, we have developed an algorithm called A-FBIMINER, for discovering the set $\mathbb{A}_\tau$ of almost forbidden itemsets (or more precisely a subset of $\mathbb{A}_\tau$ with the same properties, as will be explained shortly). This algorithm is similar in spirit to FBIMINER, using mplift rather than lift. However, mplift is too lenient as the basis for an efficient discovery algorithm, as it is based on worst-case scenarios. After all, it accommodates for all possible repairs obtained by at most k modifications. As a consequence, $\mathbb{A}_\tau$ may contain too many itemsets. Since the A-FBIMINER algorithm also deploys a depth-first search strategy, we next explore how $\mathbb{A}_\tau$ can be reduced by pruning itemsets that cannot be τ-forbidden in any repair $\mathcal{D}'$. We will design the A-FBIMINER algorithm based on such repair-oblivious pruning properties and hence, A-FBIMINER will return a subset $\mathbb{A}$ of $\mathbb{A}_\tau$, yet without violating property (†). That is, it still holds that $\mathsf{FBI}(\mathcal{D}',\tau)\subseteq\mathbb{A}$ for any $\mathcal{D}'$.

Recall that algorithm A-FBIMINER is to mine almost forbidden itemsets without looking at specific modifications, i.e., only $\mathcal{D}$ and an upper bound k on the number of modified objects is available. Clearly k is at most $|\mathcal{D}_{dirty}|$. Additionally, the set $\mathcal{D}_{dirty}$ is known up front, and may be used to refine the mplift measure. We next adapt the pruning strategies from FBIMINER, by revising the underlying properties to take possible modifications into account. Since clean objects are never modified, a tight bound on the support of an itemset in any $\mathcal{D}'$, obtained from $\mathcal{D}$ by at most k modifications, can be obtained. Indeed, we observe that for any itemset I, the following holds:

$$\mathsf{supp}(I,\mathcal{D}_{clean}) \leq \mathsf{supp}(I,\mathcal{D}') \leq \mathsf{supp}(I,\mathcal{D}_{clean})+k. \qquad (\S)$$

An immediate consequence is that A-FBIMINER must also consider itemsets I with $\mathsf{supp}(I,\mathcal{D}_{clean})=0$ as these can become supported in repairs. Furthermore, we can now modify Propositions 6 and 7. In the following, $\mathcal{D}'$ denotes a dataset obtained from $\mathcal{D}$ by at most k modifications to objects in $\mathcal{D}_{dirty}$.

Proposition 3. *For any two itemsets I and J such that $I\subset J$, if J is a τ-forbidden itemset in $\mathcal{D}'$, then we have that* $\mathsf{supp}(I,\mathcal{D}_{clean}) \geq \frac{|\mathcal{D}|\times\mathsf{supp}(J,\mathcal{D}')}{\sigma^{max}_{I,\mathcal{D}'}\times\tau}-k.$ □

Proof: We show this by contradiction. Let J be a τ-forbidden itemset in $\mathcal{D}'$ for $\tau<1$ and assume for the sake of contradiction that there exists a $I\subset J$ with $\mathsf{supp}(I,\mathcal{D}_{clean}) < \frac{|\mathcal{D}|\times\mathsf{supp}(J,\mathcal{D}')}{\sigma^{max}_{I,\mathcal{D}'}\times\tau}-k$. From the inequalities (§) it follows that:

$$\mathsf{supp}(I,\mathcal{D}') \leq \frac{|\mathcal{D}|\times\mathsf{supp}(J,\mathcal{D}')}{\sigma^{max}_{I,\mathcal{D}'}\times\tau}-k+k$$

And hence $\mathsf{supp}(I,\mathcal{D}') \leq \frac{|\mathcal{D}|\times\mathsf{supp}(J,\mathcal{D}')}{\sigma^{max}_{I,\mathcal{D}'}\times\tau}$. According to Prop. 6, this implies that J cannot be a τ-forbidden itemset in $\mathcal{D}'$, which contradicts our initial assumption. Hence, every subset I of a τ-forbidden itemset J in $\mathcal{D}'$ must satisfy $\mathsf{supp}(I,\mathcal{D}_{clean}) \geq \frac{|\mathcal{D}|\times\mathsf{supp}(J,\mathcal{D}')}{\sigma^{max}_{I,\mathcal{D}'}\times\tau}-k$. □

Proposition 4. *For any three itemsets I, J and K such that $I\subset J\subseteq K$, if K is a τ-forbidden itemset in $\mathcal{D}'$, then* $\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) \geq \frac{1}{\tau}-\frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|}-k.$ □

Proof: We show this by contradiction. Let K be a τ-forbidden itemset in $\mathcal{D}'$ for $\tau<1$ and assume for the sake of contradiction that there exist subsets $I\subset J\subseteq K$ with $\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) < \frac{1}{\tau}-\frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|}-k$. From the inequalities (§) it follows that

$$k \leq \mathsf{supp}(I,\mathcal{D}') - \mathsf{supp}(I,\mathcal{D}_{clean})$$

By substituting this inequality for k, we obtain:

$$\begin{aligned}\cancel{\mathsf{supp}(I,\mathcal{D}_{clean})} - \mathsf{supp}(J,\mathcal{D}_{clean}) &< \frac{1}{\tau}-\frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|} - \mathsf{supp}(I,\mathcal{D}') + \cancel{\mathsf{supp}(I,\mathcal{D}_{clean})}\\ \mathsf{supp}(I,\mathcal{D}') - \mathsf{supp}(J,\mathcal{D}_{clean}) &< \frac{1}{\tau}-\frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|}\end{aligned}$$

Since $\mathsf{supp}(J,\mathcal{D}_{clean}) \leq \mathsf{supp}(J,\mathcal{D}')$, it follows that

$$\mathsf{supp}(I,\mathcal{D}') - \mathsf{supp}(J,\mathcal{D}_{clean}) \leq \mathsf{supp}(I,\mathcal{D}') - \mathsf{supp}(J,\mathcal{D}')$$

Hence, we may conclude that

$$\mathsf{supp}(I,\mathcal{D}') - \mathsf{supp}(J,\mathcal{D}') < \frac{1}{\tau} - \frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|}$$

This violates Prop. 7, implying that no superset of J can be τ-forbidden in $\mathcal{D}'$. This contradicts our initial assumption that $K \supseteq J$ is τ-forbidden in $\mathcal{D}'$. □

In order to use these propositions without relying on knowledge about supports in $\mathcal{D}'$ (recall that we do not know which $\mathcal{D}'$ we are considering), we need to replace $\mathsf{supp}(J,\mathcal{D}')$ and $\sigma^{max}_{I,\mathcal{D}'}$ by quantities derived from $\mathcal{D}$ alone.

For Prop. 3, similarly to the pruning strategies for FBIMiner, we again use the trivial lower bound $\mathsf{supp}(J,\mathcal{D}') \geq 1$. Also, note that $\sigma^{max}_{I,\mathcal{D}_{clean}}$ can be computed, and it holds that $\sigma^{max}_{I,\mathcal{D}'} \leqslant \sigma^{max}_{I,\mathcal{D}_{clean}} + k$. Hence, Prop. 3 is used to prune supersets of I whenever

$$\mathsf{supp}(I,\mathcal{D}_{clean}) < \frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}} + k) \times \tau} - k. \qquad \text{(P1)}$$

Similarly, Prop. 4 prunes itemsets J if there is a subset I of J such that

$$\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) < \frac{1}{\tau} - \frac{\sigma^{max}_{I,\mathcal{D}'}}{|\mathcal{D}|} - k.$$

Since it holds that $\sigma^{max}_{I,\mathcal{D}'} \leq |\mathcal{D}|$, we may avoid the approximation of $\sigma^{max}_{I,\mathcal{D}'}$ altogether, and instead simplify the expression to:

$$\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) < \frac{1}{\tau} - 1 - k. \qquad \text{(P2)}$$

Proposition 8 also needs to be modified to account for possible modifications. Recall that this proposition was based on the anti-monotonicity of the lift denominator. In order to preserve this property under modifications, we need to compute the worst case increase in the denominator of the lift measure. Since this denominator consists of a product of supports, we apply upper bounds on both supports:

Proposition 5. *For any two itemsets I and J such that $I \subset J$, it holds that* $\mathsf{lift}(J,\mathcal{D}') \geq$

$$\frac{\mathsf{supp}(J,\mathcal{D}') \times |\mathcal{D}|}{\min\limits_{S \subset I}\big\{(\mathsf{supp}(S,\mathcal{D}_{clean}) + k) \times (\mathsf{supp}(I \setminus S,\mathcal{D}_{clean}) + k)\big\}}.$$ □

Proof: This is a straightforward extension of Prop. 8. Indeed, observe that

$$\mathsf{lift}(J,\mathcal{D}') \geq \frac{\mathsf{supp}(J,\mathcal{D}') \times |\mathcal{D}|}{\min\limits_{\emptyset \subset S \subset I}\big\{\mathsf{supp}(S,\mathcal{D}') \times \mathsf{supp}(I \setminus S,\mathcal{D}')\big\}}.$$

By applying the inequalities $\mathsf{supp}(S,\mathcal{D}') \leq \mathsf{supp}(S,\mathcal{D}_{clean}) + k$ and $\mathsf{supp}(I \setminus S,\mathcal{D}') \leq \mathsf{supp}(I \setminus S,\mathcal{D}_{clean}) + k$, if follows that $\mathsf{lift}(J,\mathcal{D}') \geq$

$$\frac{\mathsf{supp}(J,\mathcal{D}') \times |\mathcal{D}|}{\min\limits_{S \subset I}\big\{(\mathsf{supp}(S,\mathcal{D}_{clean}) + k) \times (\mathsf{supp}(I \setminus S,\mathcal{D}_{clean}) + k)\big\}}.$$

□

To make use of this proposition for pruning, without knowing $\mathcal{D}'$, we set $\mathsf{supp}(J,\mathcal{D}') = 1$ and hence prune I's supersets whenever

$$\frac{|\mathcal{D}|}{\min\limits_{S \subset I}\big\{(\mathsf{supp}(S,\mathcal{D}_{clean}) + k) \times (\mathsf{supp}(I \setminus S,\mathcal{D}_{clean}) + k)\big\}} \geq \tau \qquad \text{(P3)}$$

Finally, we update Prop. 9 to obtain the maximal support in $\mathcal{D}_{clean}$ of a τ-forbidden itemset in any $\mathcal{D}'$ obtained by modifying $\mathcal{D}$. Recall that this is not really a pruning strategy, but provides a way of avoiding certain unnecessary lift computations.

Proposition 6. *For any itemset I that is τ-forbidden in $\mathcal{D}'$, we have* $\mathsf{supp}(I,\mathcal{D}_{clean}) \leqslant |\mathcal{D}| \times (\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1)$. □

Proof: Prop. 9 states that, for I to be τ-forbidden in $\mathcal{D}'$, it must hold that:

$$\mathsf{freq}(I,\mathcal{D}') \leqslant \frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$$

In terms of support, this implies that

$$\mathsf{supp}(I,\mathcal{D}') \leqslant |\mathcal{D}| \times (\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1)$$

From the inequalities in (§), we use the fact that $\mathsf{supp}(I,\mathcal{D}') \geqslant \mathsf{supp}(I,\mathcal{D}_{clean})$ to obtain that:

$$\mathsf{supp}(I,\mathcal{D}_{clean}) \leqslant |\mathcal{D}| \times (\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1).$$

□

The three pruning strategies (P1), (P2) and (P3) allow substantial pruning when mining almost forbidden itemsets, similarly as their counterparts for FBIMINER. We will denote by $\mathbb{A}$ the set of almost forbidden itemsets returned by A-FBIMINER with these pruning strategies enabled. However, observe that k impacts the pruning power of (P1) and (P2). Indeed, when k is "chosen" such that the upper bounds in (P1) and (P2) do not impose any constraints on the supports of the itemsets involved, these pruning strategies are of no use. For example, suppose that k takes the maximal value, i.e., $k = |\mathcal{D}_{dirty}|$. Then, it is likely that (P1) and (P2) do not allow for any pruning. Worse still, the minimal possible lift, which also depends on k, will declare many itemsets to be almost forbidden, since the accuracy of this lower bound wanes as k increases. Although A-FBIMINER will need to be run only once to obtain the set $\mathbb{A}$ and all dirty objects can be repaired based on $\mathbb{A}$, this set will be big and inefficient to compute (due to lack of pruning). On the other hand, when $k = 1$, strong pruning will be possible and $\mathbb{A}$ will be of reasonable size. However, to clean all dirty objects, we need to deal with them one-at-a-time, re-running A-FBIMINER for $k = 1$ after a single dirty object is repaired. To attain maximal efficiency for algorithm A-FBIMINER we therefore propose, in the following subsection, to repair dirty objects in batches of limited size k.

APPENDIX E
REPAIRING IN BATCHES

To improve the repair performance, we have suggested repairing dirty objects in batches of a specific size. Instead of repairing all k objects at once, we now consider a partitioning of $\mathcal{D}_{dirty}$ into blocks of a size r. We want to optimize the trade-off between the runtime of A-FBIMINER and the number of runs of A-FBIMINER. The question, of course, is what block size to select. We already described block sizes $r = 1$ and $r = |\mathcal{D}_{dirty}|$, i.e., one-by-one and all-at-once. The question we next wish to answer is, how do the intermediate block sizes influence our pruning capabilities, and hence our runtime? We therefore analyze Prop. 3 and Prop. 4, replacing k with r, in order to identify the ranges for r in which pruning will still be possible.

We first turn our attention to Prop. 3. Let I be the itemset under consideration. Pruning strategy (P1) states that I's supersets can be

pruned if $\mathsf{supp}(I,\mathcal{D}_{clean}) < \frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau} - r$. This is useful only if $\frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau} - r > 0$, since it trivially holds that $\mathsf{supp}(I,\mathcal{D}_{clean}) \geq 0$. In order to perform any pruning with strategy (P1), we thus need to choose $r < \frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau}$.

From Prop. 4, we derived pruning strategy (P2), stating that I's supersets may be pruned if $\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) < \frac{1}{\tau} - 1 - r$. Again, $\mathsf{supp}(I,\mathcal{D}_{clean}) - \mathsf{supp}(J,\mathcal{D}_{clean}) \geq 0$ holds trivially, and hence this strategy is only useful if $\frac{1}{\tau} - 1 - r > 0$. We thus require a block size $r < \frac{1}{\tau} - 1$ to perform any pruning using strategy (P2).

Moreover, it holds that $\frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)} \geq 1$, since $\sigma^{max}_{I,\mathcal{D}_{clean}} \leq |\mathcal{D}_{clean}|$, $|\mathcal{D}_{clean}| + |\mathcal{D}_{\mathsf{dirty}}| = |\mathcal{D}|$, and $r \leq |\mathcal{D}_{\mathsf{dirty}}|$. It is then easy to see that $\frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau} > \frac{1}{\tau} - 1$. This implies that pruning with (P1) is always possible if pruning with (P2) is possible, but not vice versa. To have *any* pruning with these strategies, a block size $r < \frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau}$ is thus required.

The pruning strategy (P3) makes use of the number of dirty objects r in computing the values $(\mathsf{supp}(S,\mathcal{D}_{clean})+r)$ and $(\mathsf{supp}(I\setminus S,\mathcal{D}_{clean})+r)$, i.e., in bounding the support of I's subsets. Since the applicability of (P3) depends on the actual values of $\mathsf{supp}(S,\mathcal{D}_{clean})$ and $\mathsf{supp}(I\setminus S,\mathcal{D}_{clean})$, relative to each other, it is hard to gauge the impact of the block size r on this pruning strategy, and we cannot derive a maximal block size for which this strategy is useful. Of course, the general idea that a lower r improves pruning power holds for this strategy as well.

As a consequence, there is no universally optimal block size r. The specifics of the data and even the choice of which objects to include in a partition of r objects all impact the pruning power. It is also important to note that the block sizes derived above guarantee that the associated propositions are *applicable*, but they will still offer *reduced pruning power* in comparison to smaller block sizes. In the experimental section, we show that $r = \frac{1}{2\tau}$, half of the maximal size for which (P2) is applicable, provides a sensible default value of r. On the other hands, using $r = \frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}}+r)\times\tau} - 1$, the maximal size for any pruning with (P1), improves the runtime on datasets with a small number of attributes.

APPENDIX F
MINING ALMOST FORBIDDEN ITEMSETS

The algorithm A-FBIMINER for mining almost τ-forbidden itemsets is identical in structure to FBIMINER, as shown in Alg. 4. Itemsets are again discovered in a depth-first way, using a reverse pre-order traversal. The algorithm takes as input arguments a projected dataset in vertical layout $\mathcal{D}{\downarrow}[P]$, consisting of all objects which contain a prefix itemset P, and the lift threshold τ. Additionally, we pass the number k of dirty objects under consideration as a fourth argument. We next discuss the difference between the algorithms in terms of lift computation and pruning, and conclude this section with a proof of A-FBIMINER's correctness.

The most important difference between A-FBIMINER and FBIMINER is that, for every processed itemset I, we now compute the mplift measure instead of regular lift. Just like the lift measure, the mplift measure requires access to all subsets of I, which we store during the traversal using a prefix tree. The procedure for computing mplift is invoked on line 12. If the itemset I under consideration cannot be τ-forbidden in any $\mathcal{D}'$ according to Prop. 6, because it is too frequent in $\mathcal{D}_{clean}$, we avoid computing its mplift on line 9.

The procedure for mplift first computes the actual lift of the itemset I, in order to identify a subset $S \subset I$ such that $\mathsf{lift}(I,\mathcal{D}) = |D| \times \mathsf{supp}(I,\mathcal{D})/(\mathsf{supp}(S,\mathcal{D}) \times \mathsf{supp}(I\setminus S,\mathcal{D}))$. In other words, we first find the partition that currently maximizes the lift, as explained earlier (just after Definition 6). Using the supports of I, S and $I\setminus S$, the function f from Fig. 5 in Sec. 5.2 computes the worst case configuration of these supports after k modifications. This configuration then gives us $\mathsf{mplift}_k(I)$. If mplift is lower than τ, then I is added to the set $\mathbb{A}$ consisting of almost FBIs (lines 12-13).

The pruning strategies employed by A-FBIMINER are equivalent to those in FBIMINER, except that the revised pruning strategies presented in Sec. D are used. First, Prop. 4 implies that non-generator itemsets can be pruned only when $k \leq \frac{1}{\tau} - 1$ (line 5). If this condition is satisfied, then procedures ISGENERATOR and STOREGENERATOR are used for pruning, exactly as before.

Similarly to the pruning strategies in FBIMINER, the pruning conditions for A-FBIMINER are used to prune all supersets of the current itemset I. On line 10, we detect whether a subset of I has previously been pruned. If this is the case, then I and all of its supersets may be pruned as well. If no subsets have been pruned, we are able to compute the mplift and lift measures of I. Using the denominators considered during lift computation, we can invoke pruning (line 15) based on the anti-monotonicity of the lift denominator as stated in Prop. 5. The minimum support in $\mathcal{D}_{clean}$ of subsets of itemsets which are τ-forbidden in $\mathcal{D}'$ (Prop. 3) is verified on line 17, after processing I itself. If $\mathsf{supp}(I,\mathcal{D}_{clean})$ does not satisfy this condition, then all subsets of J are pruned from the search tree. Finally, extensions J of I that do not satisfy the minimum reduction in lift, based on Prop. 4, are discarded on line 25, when computing their supports, similar to FBIMINER.

The computation of supports in A-FBIMINER again makes use of set intersections of the covers of the items. These supports are computed in $\mathcal{D}$. Note, however, that A-FBIMINER also needs to consider itemsets I for which $\mathsf{supp}(I,\mathcal{D}) = 0$. Indeed, the check $\mathsf{supp}(J,\mathcal{D}) > 0$ on line 23 of FBIMINER has disappeared. For our pruning strategies, we also require the support of the itemsets in $\mathcal{D}_{clean}$. We compute these by mining the supports of the itemsets in $\mathcal{D}_{dirty}$, since this set is typically much smaller than $\mathcal{D}_{clean}$. Supports in $\mathcal{D}_{dirty}$ are also computed using set intersections of the covers of the items. The value $\mathsf{supp}(I,\mathcal{D}_{clean})$ is eventually obtained as $\mathsf{supp}(I,\mathcal{D}) - \mathsf{supp}(I,\mathcal{D}_{dirty})$ (line 24).

Proposition 7. *Let $\mathcal{D}$ be a dataset and τ a maximum lift threshold. Algorithm* A-FBIMINER$(\mathcal{D}{\downarrow}[\emptyset],\emptyset,k,\tau)$ *returns a set $\mathbb{A}$ such that, if $\mathsf{lift}(J,\mathcal{D}') \leq \tau$ for some $\mathcal{D}'$ obtained from $\mathcal{D}$ by at most k modifications, then $J \in \mathbb{A}$.*

Proof: Consider the core of algorithm A-FBIMINER in which all pruning is disabled. Then, A-FBIMINER mines all itemsets and returns the set $\mathbb{A}_\tau$ consisting of all J with $|J| > 1$ and $\mathsf{mplift}_k(J,\mathcal{D}) \leq \tau$. Since $\mathsf{FBI}(\mathcal{D}',\tau) \subseteq \mathbb{A}_\tau$ the proposition follows.

We next argue that the property holds, even when pruning is enabled. Assume, for the sake of contradiction, that there exists an itemset I with $|I| > 1$ and $\mathsf{lift}(I,\mathcal{D}') \leq \tau$, where $\mathcal{D}'$ is a dataset obtained from $\mathcal{D}$ by at most k modifications, yet I is not returned by A-FBIMINER$(\mathcal{D}{\downarrow}[\emptyset],\emptyset,k,\tau)$. There are only two possibilities why I could not have been returned: Either (i) $\mathsf{mplift}_k(I,\mathcal{D}) > \tau$ or (ii) one of I's subsets was pruned.

However, (i) is impossibly since $\mathsf{lift}(I,\mathcal{D}') \leq \tau$ implies $\mathsf{mplift}_k(I,\mathcal{D}) \leq \tau$. Similarly, (ii) is impossible. Indeed, when a subset of I is pruned, Propositions 3–5 imply that I and all

Algorithm 4 Mining Almost Forbidden Itemsets

procedure A-FBIMINER($\mathcal{D}\downarrow[P], P \subseteq \mathcal{I}, \tau, k$)
  $\mathbb{A} \leftarrow \emptyset$
  for all $i \in \mathcal{I}$ occuring in $\mathcal{D}\downarrow[P]$ in reverse order **do**
    $I \leftarrow P \cup \{i\}$
    if $k \leq \frac{1}{\tau} - 1$ **then**
      if not ISGENERATOR(I) **then**
        continue
      STOREGENERATOR(I)
    if $|I| > 1$ & $\mathsf{freq}(I, \mathcal{D}_{clean}) \leq \frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1$ **then**
      if a subset of I has been pruned **then**
        continue
      if $\mathsf{mplift}_k(I, \mathcal{D}) < \tau$ **then**
        $\mathbb{A} = \mathbb{A} \cup I$
      if $|\mathcal{D}|/\tau > \min_{S \subset I} \{(\mathsf{supp}(S, \mathcal{D}_{clean}) + |\mathcal{D}_{dirty}|) \times$
          $(\mathsf{supp}(I \setminus S, \mathcal{D}_{clean}) + |\mathcal{D}_{dirty}|)\}$ **then**
        continue
    if $\mathsf{supp}(I, \mathcal{D}_{clean}) < \frac{|\mathcal{D}|}{(\sigma^{max}_{I, \mathcal{D}_{clean}} + k) \times \tau} - k$ **then**
      continue
    $\mathcal{D}\downarrow[I] \leftarrow \emptyset$
    for all $j \in \mathcal{I}$ in $\mathcal{D}$ such that $j > i$ **do**
      $C \leftarrow \mathsf{cov}(\{i\}, \mathcal{D}) \cap \mathsf{cov}(\{j\}, \mathcal{D})$
      $J \leftarrow I \cup \{j\}$
      $\mathsf{supp}(J, \mathcal{D}) \leftarrow |C|$
      $\mathsf{supp}(J, \mathcal{D}_{clean}) \leftarrow \mathsf{supp}(J, \mathcal{D}) - \mathsf{supp}(J, \mathcal{D}_{dirty})$
      if $\mathsf{supp}(I, \mathcal{D}) - \mathsf{supp}(J, \mathcal{D}) \geq \frac{1}{\tau} - 1 - k$ **then**
        $\mathcal{D}\downarrow[I] \leftarrow \mathcal{D}\downarrow[I] \cup \{(j, C)\}$
    $\mathbb{A} \leftarrow \mathbb{A} \cup$ A-FBIMINER($\mathcal{D}\downarrow[I], I, \tau, k$)
  return $\mathbb{A}$

its supersets have $\mathsf{lift}(I, \mathcal{D}') > \tau$, again contradicting the initial assumption that I was τ-forbidden.

We may thus conclude that A-FBIMINER returns (at least) all I such that $|I| > 1$ and $\mathsf{lift}(I, \mathcal{D}') \leq \tau$ for all $\mathcal{D}'$ obtained from $\mathcal{D}$ by at most k modification. In other words, the set $\mathbb{A}$ returned by A-FBIMINER satisfies property (†) stated in Prop. 1 when L is taken to be the lift measure. □

APPENDIX G
TIME COMPLEXITY

We first derive the time complexity of the algorithms FBIMINER and A-FBIMINER. In terms of worst-case complexity, both algorithms are the same. The difference in actual runtime is due to pruning in practice.

We first derive the size of the itemset lattice. Let $|\mathcal{A}|$ denote the number of attributes, with d the highest domain size of any attribute. At level i in the lattice, the itemsets can contain items from $\binom{|\mathcal{A}|}{i}$ attribute combinations. For each attribute, there are at most d different items, meaning there are $\binom{|\mathcal{A}|}{i} d^i$ different itemsets at level i. The total number of itemsets in the lattice is then:

$$\sum_{i=1}^{|\mathcal{A}|} \binom{|\mathcal{A}|}{i} (d)^i.$$

For each itemset, the algorithm then performs two main computations. The lift computation requires all partitionings of an itemset $|I|$ into two distinct subsets. For an itemset of length i, there are 2^{i-1} such partitionings. For each processed itemset, the Eclat algorithm then performs intersections with all itemsets that differ in 1 attribute. There are $|\mathcal{A}| - i$ such attribute combinations, meaning there are $(|\mathcal{A}| - i)d^i$ intersections. If we denote by s the support of the most frequent item in the data, and hence the maximal length of any tidlist to be intersected, it follows that $\mathcal{O}(2^{i-1} + ((|\mathcal{A}| - i) \times d^i \times s))$ operations are required for each itemset in the search lattice. Assuming $d \geq 2$, we can simplify this to $\mathcal{O}((|\mathcal{A}| - i) \times d^i \times s)$. The total number of operations for the algorithms, in the worst case, is thus:

$$\mathcal{O}(s \sum_{i=1}^{|\mathcal{A}|} \binom{|\mathcal{A}|}{i} (d)^i \times (|\mathcal{A}| - i) \times d^i).$$

To obtain the overall complexity, we first simplify $\mathcal{O}(\sum_{i=1}^{|\mathcal{A}|}(|\mathcal{A}| - i) = \mathcal{O}(|\mathcal{A}|)$. By rewriting $\sum_{i=1}^{|\mathcal{A}|} \binom{|\mathcal{A}|}{i}(d)^i \times d^i)$ as $\sum_{i=1}^{|\mathcal{A}|} \binom{|\mathcal{A}|}{i}(d^i)^2 \times 1^{|\mathcal{A}|-i})$, we can compute this to be equal to $(d^2+1)^{|\mathcal{A}|}$. We then arrive at the total complexity of the algorithms:

Proposition 8. *The time complexity of algorithms* FBIMINER *and* A-FBIMINER *is* $\mathcal{O}(s \times |\mathcal{A}| \times (d^2+1)^{|\mathcal{A}|})$.

The complexity of repairing, Algorithm 1, can be split into two parts. Firstly, for every dirty object $o \in \mathcal{D}_{dirty}$, we compute the similarity with each clean object $o_c \in \mathcal{D}_{clean}$, having a complexity of $\mathcal{O}(|\mathcal{D}_{dirty}| \times |\mathcal{D}_{clean}|)$. Secondly, for each block of dirty objects $\mathcal{R}_i$ in BLOCKS($\mathcal{D}_{dirty}, r$), algorithm A-FBIMINER has to be run, i.e., a total of $\frac{|\mathcal{D}|}{r}$ times. This gives us the overal time complexity of repairing:

Proposition 9. *The time complexity of algorithm* REPAIR *is* $\mathcal{O}(|\mathcal{D}_{dirty}| \times |\mathcal{D}_{clean}| + \frac{|\mathcal{D}|}{r} \times s \times |\mathcal{A}| \times (d^2+1)^{|\mathcal{A}|})$.

APPENDIX H
PROOF OF PROPOSITION 3

We formally prove Proposition 3, presented in Section 5.5.

Proposition 3. *Consider object* $o_k = \langle \mathsf{oid}, I \rangle$ *in* $\mathcal{D}'_{k-1}$ *and its candidate modification* $o_{\text{rep},k} = \langle \mathsf{oid}, I' \rangle$ *in* $\mathcal{D}'_k$. *Let* $\tau < 3/4$. *If the following conditions are satisfied*

(a) $\mathsf{covitems}(I, \mathbb{P}_i) \subseteq \mathsf{covitems}(I', \mathbb{P}_i)$; *and*
(b) for each $a \in I' \setminus I$,

$$\mathsf{itemscov}(\{a\}, \mathbb{P}_i) \subseteq \mathsf{covitems}(I', \mathbb{P}_i),$$

then for each $J \in \mathbb{P}_i$, *we have that* $\mathsf{lift}(J, \mathcal{D}'_{k-1}) > \tau$ *implies* $\mathsf{lift}(J, \mathcal{D}'_k) > \tau$.

Proof: We only have to consider the case when $\mathsf{lift}(J, \mathcal{D}'_{k-1}) > \mathsf{lift}(J, \mathcal{D}'_k)$. By assumption and by the definition of lift, we have

$$\tau < \mathsf{lift}(J, \mathcal{D}'_{k-1}) = \frac{|\mathcal{D}| \times \mathsf{supp}(J, \mathcal{D}'_{k-1})}{\mathsf{supp}(S, \mathcal{D}'_{k-1}) \times \mathsf{supp}(J \setminus S, \mathcal{D}'_{k-1})}$$

for some $S \subset J$. If J is supported by neither o_i (in $\mathcal{D}'_{k-1}$) nor $o_{\text{rep},i}$ (in $\mathcal{D}'_k$), then $\mathsf{supp}(J, \mathcal{D}'_{k-1}) = \mathsf{supp}(J, \mathcal{D}'_k)$. Otherwise, condition (a) implies that if J is supported by o_i, and hence $J \in \mathsf{covitems}(I, \mathbb{P}_i)$, then also $J \in \mathsf{covitems}(I', \mathbb{P}_i)$. Hence, J is also supported by $o_{\text{rep},i}$. From this we may conclude that $\mathsf{supp}(J, \mathcal{D}'_{k-1}) \leq \mathsf{supp}(J, \mathcal{D}'_k)$.

Assume that $\mathsf{lift}(J, \mathcal{D}'_{k-1}) > \mathsf{lift}(J, \mathcal{D}'_k)$. Given that $\mathsf{supp}(J, \mathcal{D}'_{k-1}) \leq \mathsf{supp}(J, \mathcal{D}'_k)$, it must be the case that

$\mathsf{supp}(S,\mathcal{D}'_k) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_k) > \mathsf{supp}(S,\mathcal{D}'_{k-1}) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})$. Since only one modification is considered at a time, the maximal value of $\mathsf{supp}(S,\mathcal{D}'_k) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_k)$ is given by $(\mathsf{supp}(S,\mathcal{D}'_{k-1})+1) \times (\mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})+1)$.

This in turn can only occur when S and $J \setminus S$ contain items in $I' \setminus I$ (otherwise, their supports would be the same as in $\mathcal{D}'_{k-1}$). For example, let $a \in I' \setminus I$ such that $a \in S$. Then, $a \in J$ and hence $J \in \mathsf{itemscov}(\{a\},\mathbb{P}_i)$. Hence, condition (b) implies that J is supported by $o_{\text{rep},i}$. In addition, since S and $J \setminus S$ contain items in $I' \setminus I$, o_i cannot support S and $J \setminus S$. Hence, o_i also did not support J. This implies that $\mathsf{supp}(J,\mathcal{D}'_k) = \mathsf{supp}(J,\mathcal{D}'_{k-1})+1$.

To conclude the proof, we now derive the following implication: For $\tau < 3/4$,

$$\tau < \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D}'_{k-1})}{\mathsf{supp}(S,\mathcal{D}'_{k-1}) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})}$$

implies that

$$\tau < \frac{|D| \times (\mathsf{supp}(J,\mathcal{D}'_{k-1})+1)}{(\mathsf{supp}(S,\mathcal{D}'_{k-1})+1) \times (\mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})+1)},$$

which in turn is $\leq \mathsf{lift}(J,\mathcal{D}'_k)$.

We start from the fact that

$$\tau < \frac{|\mathcal{D}| \times \mathsf{supp}(J,\mathcal{D}'_{k-1})}{\mathsf{supp}(S,\mathcal{D}'_{k-1}) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})} \Rightarrow$$
$$\tau < \frac{|D| \times \mathsf{supp}(J,\mathcal{D}'_{k-1})+1}{(\mathsf{supp}(S,\mathcal{D}'_{k-1})+1) \times (\mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})+1)},$$

holds for all $\tau < 3/4$.

The implication is trivially satisfied if

$$\frac{|\mathcal{D}| \times (\mathsf{supp}(J,\mathcal{D}'_{k-1})+1)}{(\mathsf{supp}(S,\mathcal{D}'_{k-1})+1) \times (\mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})+1)} \geq \frac{|D| \times \mathsf{supp}(J,\mathcal{D}'_{k-1})}{\mathsf{supp}(S,\mathcal{D}'_{k-1}) \times \mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})}.$$

We therefore consider the case when the previous inequality is not satisfied. In other words, we only need to consider the case where the lift drops after a modification. To simplify notation, we let $\sigma_J = \mathsf{supp}(J,\mathcal{D}'_{k-1})$, $\sigma_S = \mathsf{supp}(S,\mathcal{D}'_{k-1})$ and $\sigma_{J\setminus S} = \mathsf{supp}(J \setminus S,\mathcal{D}'_{k-1})$. We therefore consider the case when

$$\frac{|\mathcal{D}|(\sigma_J+1)}{(\sigma_S+1) \times (\sigma_{J\setminus S}+1)} < \frac{|D| \times \sigma_J}{\sigma_S \times \sigma_{J\setminus S}} \tag{1}$$

holds. We now proof the proposition by contradiction. That is, suppose that there exists a $\tau < 3/4$ such that $\tau < \frac{|D|\times\sigma_J}{\sigma_S\times\sigma_{J\setminus S}}$ but $\frac{|\mathcal{D}|(\sigma_J+1)}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} \leq \tau$.

We start by eliminating σ_J from the latter expression. To this aim, observe that (1) implies that

$$\frac{|\mathcal{D}|}{\sigma_S + \sigma_{J\setminus S}+1} < \frac{|D| \times \sigma_J}{\sigma_S \times \sigma_{J\setminus S}},$$

or, equivalently, that

$$\frac{\sigma_S \times \sigma_{J\setminus S}}{\sigma_S + \sigma_{J\setminus S}+1} < \sigma_J. \tag{2}$$

Now, observe the following:

$$\begin{aligned}
\tau &\geq \frac{|\mathcal{D}|(\sigma_J+1)}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} \\
&> \frac{|\mathcal{D}| \times (\frac{\sigma_S\times\sigma_{J\setminus S}}{\sigma_S+\sigma_{J\setminus S}+1}+1)}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} && \text{(Using (2))} \\
&= \frac{|\mathcal{D}| \times \frac{\sigma_S\times\sigma_{J\setminus S}+\sigma_S+\sigma_{J\setminus S}+1}{\sigma_S+\sigma_{J\setminus S}+1}}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} \\
&= \frac{|\mathcal{D}| \times \frac{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)}{\sigma_S+\sigma_{J\setminus S}+1}}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} \\
&= \frac{|\mathcal{D}|}{\sigma_S+\sigma_{J\setminus S}+1}.
\end{aligned}$$

From this, we may infer that $\frac{|\mathcal{D}|}{\tau} - 1 < \sigma_S + \sigma_{J\setminus S}$. Furthermore, due to the inclusion-exclusion principle, it holds that

$$\sigma_S + \sigma_{J\setminus S} - |\mathcal{D}| \leq \sigma_J$$

And thus, we can lower bound σ_J in terms of τ:

$$\frac{|\mathcal{D}|}{\tau} - 1 - |\mathcal{D}| \leq \sigma_J.$$

On the other hand, Prop. 9 gives us an upper bound on the support of any forbidden itemset. If itemset J becomes τ-forbidden in $\mathcal{D}'_k$, then it must hold that:

$$\sigma_J + 1 \leq |\mathcal{D}| \times (\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1).$$

We can now use both the upper and lower bound on σ_J to derive a lower bound on τ:

$$\begin{aligned}
\frac{|\mathcal{D}|}{\tau} - |\mathcal{D}| - 1 &\leq |\mathcal{D}| \times (\frac{2}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - 1) - 1 \\
&= \frac{2|\mathcal{D}|}{\tau} - 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} - |\mathcal{D}| - 1
\end{aligned}$$

or,

$$\frac{|\mathcal{D}|}{\tau} \leq \frac{2|\mathcal{D}|}{\tau} - 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}},$$

which in turn holds when

$$0 \leq \frac{|\mathcal{D}|}{\tau} - 2|\mathcal{D}|\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} = |\mathcal{D}|(\frac{1}{\tau} - 2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}}).$$

Hence, we may infer that $2\sqrt{\frac{1}{\tau^2} - \frac{1}{\tau}} \leq \frac{1}{\tau}$. Squaring both sides and multiplying by τ^2, we obtain that $4(1-\tau) \leq 1$ and thus $3/4 \leq \tau$. This contradicts our earlier assumption that $\tau < 3/4$. Hence, no J, S and $J \setminus S$ exist such that $\tau < \frac{|D|\times\sigma_J}{\sigma_S\times\sigma_{J\setminus S}}$ but $\frac{|\mathcal{D}|(\sigma_J+1)}{(\sigma_S+1)\times(\sigma_{J\setminus S}+1)} \leq \tau$ hold, for $\tau < 3/4$, as desired.This concludes the proof. □

APPENDIX I
PROOF OF PROPOSITION 4

We formally prove Proposition 4, presented in Section 5.5.

Proposition 4. *Let $\mathcal{D}$ be a dataset, τ a maximum lift threshold,* sim *a similarity function, and let $\mathcal{D}_{\text{dirty}}$ and $\mathcal{D}_{\text{clean}}$ denote the dirty and clean parts of $\mathcal{D}$, respectively. Algorithm* REPAIR($\mathcal{D}_{\text{dirty}}$, $\mathcal{D}_{\text{clean}}$, sim, τ) *returns sets $\mathcal{D}'$ and $\mathcal{D}''$ of repaired and unrepairable objects. If the repair was successful, i.e., $\mathcal{D}'' = \emptyset$, then* $\mathsf{FBI}(\mathcal{D}_{\text{clean}} \cup \mathcal{D}', \tau) = \emptyset$.

$$f(x,y,z) = \begin{cases} g(x,y,z) & y \le z \\ g(x,z,y) & y > z. \end{cases} \qquad g(x,y,z) = \begin{cases} (x-1,y,z-1) & \text{if } \frac{x-1}{y\times(z-1)} \le \frac{x}{(y+1)\times z} \text{ and } x-1>0, z-1>0; \\ (x,y+1,z) & \text{if } \frac{x-1}{y\times(z-1)} > \frac{x}{(y+1)\times z} \text{ and } x-1>0, z-1>0; \\ (x,y+1,z) & \text{if } x=1 \text{ or } z=1; \\ (x+1,y+1,z+1) & \text{if } x=0. \end{cases}$$

Figure 5. Definition the recursive function f that identifies those updates to x, y and z that will lead to the largest reduction in the lift.

Proof: For the sake of contradiction, suppose that there exists a τ-forbidden itemset J in $\mathsf{FBI}(\mathcal{D}_{clean} \cup \mathcal{D}', \tau)$.

Observe that no object in $\mathcal{D}_{clean}$ supports a τ-forbidden itemset in $\mathsf{FBI}(\mathcal{D}, \tau)$. By Prop. 4, no modification made by algorithm REPAIR can create new forbidden itemsets. Since REPAIR does not make modifications to clean objects, no object in $\mathcal{D}_{clean}$ supports a τ-forbidden itemset in $\mathsf{FBI}(\mathcal{D}_{clean} \cup \mathcal{D}', \tau)$ either.

Suppose that J is supported by an object $o_i \in \mathcal{D}'$. Then, Prop. 4 implies that $J \in \mathsf{FBI}(\mathcal{D}'_k, \tau)$. We have just observed that accepted repairs cannot support such forbidden itemsets. Consequently, no τ-forbidden itemset J in $\mathsf{FBI}(\mathcal{D}_{clean} \cup \mathcal{D}', \tau)$ can exist and therefore $\mathcal{D}_{clean} \cup \mathcal{D}'$ is indeed clean. □

APPENDIX J
ANNOTATIONS ON REPAIRING

In Section 5.2, we have assumed that $\mathsf{supp}(I, \mathcal{D}) > 0$. When $\mathsf{supp}(I, \mathcal{D}) = 0$. Since we are only interested to lower bound the lift of I in $\mathcal{D}'$ when $\mathsf{supp}(I, \mathcal{D}') > 0$, this implies that we may assume that $\mathsf{supp}(I, \mathcal{D}') = 1$, and thus $\Delta_I = +1$. Furthermore, this also implies that $\mathsf{supp}(J, \mathcal{D}') \ge \mathsf{supp}(J, \mathcal{D})$ for any $J \subset I$, and $\mathsf{supp}(J, \mathcal{D}') = \mathsf{supp}(J, \mathcal{D}) + 1$ for at least one $J \subset I$. We thus have three possible triples to describe these changes: $(+1, 0, +1)$, $(+1, +1, 0)$ and $(+1, +1, +1)$. Clearly, the latter causes the greatest increase in the denominator, leading to the minimal lift. We thus have that $\mathsf{lift}(I, \mathcal{D}')$ is greater than or equal to

$$\frac{|\mathcal{D}| \times (\mathsf{supp}(I, \mathcal{D}) + 1)}{(\mathsf{supp}(J, \mathcal{D}) + 1) \times (\mathsf{supp}(I \setminus J, \mathcal{D}) + 1)}. \qquad (\dagger\dagger)$$

Using the σ notation introduced in Sect. 5.2, and denoting $\sigma^1_I = \sigma_I + 1$, $\sigma^1_J = \sigma_J + 1$ and $\sigma^1_{I\setminus J} = \sigma_{I\setminus J} + 1$, we have that:

$$\mathsf{lift}(I, \mathcal{D}') \ge \frac{|\mathcal{D}| \times \sigma^1_I}{\sigma^1_J \times \sigma^1_{I\setminus J}}.$$

In Section 5.3, Definition 6, the mplift measure was defined using the updates σ^i_I, σ^i_J, and $\sigma^i_{I\setminus J}$ of σ_I, σ_J and $\sigma_{I\setminus J}$, respectively, for $i = 1, 2, \ldots, k$. We formalize the recursive selection of these updates, by defining a function $f(x,y,z)$ from triples in $\mathbb{N}^3$ to triples in $\mathbb{N}^3$, as shown in Figure 5. Here x, y and z correspond to σ_I, σ_J and $\sigma_{I\setminus J}$, respectively.

First, $f(x,y,z)$ is defined in terms of another function $g(x,y,z)$ which just ensures that the second argument is smaller or equal than the third argument. This is needed to reduce the computation of expression (‡) to the two triples mentioned earlier. Indeed, observe that Prop. 2 requires $\mathsf{supp}(J, \mathcal{D}) \le \mathsf{supp}(I \setminus J, \mathcal{D})$ and hence we either call $g(x,y,z)$ if $y \le z$, or call $g(x,z,y)$ if $y > z$. This corresponds to changing the role of J and $I \setminus J$ in the analysis.

Then, $g(x,y,z)$ captures the case analyses explained previously. For example, when $x = 0$ ($\sigma_I = 0$) we need to update x, y and z according to expression (††). That is, $g(x,y,z) = (x+1, y+1, z+1)$. Similarly, when either $x = 1$ or $z = 1$, letting $g(x,y,z) = (x, y+1, z)$ results in the minimization of (‡) as described in Proposition 2. In all other cases, $g(x,y,z) = (x-1, y, z-1)$ or $g(x,y,z) = (x, y+1, z)$ depending on which of the two updates to the supports minimizes (‡) as described in Proposition 2. The former case corresponds to the triple $(-1, 0, -1)$, the latter to triple $(0, +1, 0)$. It should be clear that $f(x,y,z)$ correctly updates the values x, y and z. That is, it returns updated values that compute (‡) or (††) when one modification is considered. By recursively applying the function f we ensure that $(x^i, y^i, z^i) = f(x^{i-1}, y^{i-1}, z^{i-1})$ correspond to updates that minimize the lift after i modifications.

Table 5
Statistics of the UCI datasets used in the experiments. We report the number of objects, distinct items, and attributes.

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{A}\rvert$
Adult	48842	202	11
CensusIncome	199524	235	12
CreditCard	30000	216	12
Ipums	70187	364	32
LetterRecognition	20000	282	17
Mushroom	8124	119	23

APPENDIX K
ADDITIONAL EXPERIMENTS

We present various addtional experiments, as discussed at the beginning of Section 7. Statistics of the datasets used in the additional experiments are shown in Table 5.

K.1 Lift Measure for Forbidden Itemsets

We begin the additional experiments by given some more evidence supporting our use and definition of lift for forbidden itemsets. In section 3, we have already provided some intuition behind the choice for low lift itemsets as an error detection method. The low lift forbidden itemsets rely on two "design choices". Firstly, we have opted to use lift instead of, for instance, confidence, as commonly used in approximate CFDs or association rules. Secondly, we have defined the lift of an itemset using two partitions, i.e., based on the association rules that can be created by splitting the itemset in two. We next provide examples and results to support our design choices and the general usefulness of forbidden itemsets to detect errors.

We have opted for the lift measure as opposed to confidence, since confidence does not take correlation between items into account. Here, we define confidence of an itemset as the highest confidence of any association rule between two partitions of the itemset, similar to how we defined lift. For example, on the Adult dataset, the itemset (MARITALSTATUS=WIDOWED, COUNTRY=GUATEMALA) can be discovered with a confidence of 0.05. Clearly, widowed people in Guatemala are not erroneous; this itemset is simply a consequence of the low frequency of the item (COUNTRY=GUATEMALA) in this dataset.

Table 6
Five least likely itemsets discovered in Adult dataset, using different likeliness functions

Likeliness	Example Itemsets
Pairwise Lift	Sex=Female, Relation=Husband Sex=Male, Relation=Wife Relation=Not-in-family, Marital=Married Marital-status=Married, Age=<18 Relation=Husband, Age=<18
Full Indep.	Sex=Female, Relation=Husband Sex=Male, Relation=Wife Relation=Not-in-family, Marital=Married, Educ.=College Relation=Not-in-family, Marital=Married, Occup.=Other Relation=Not-in-family, Marital=Married, Age=> 50
Confidence	Sex=Female, Relation=Husband Country=Mexico, Race=Asian-Pac-Islander Age=18-21, Education=Masters Occupation=Protective-serv, Educ.=Prof-school Occupation=Machine-op-inspct, Educ.=Prof-school

In fact, a similar low-confidence itemset can be found with other Country-values, such as Honduras. When using the lift measure, however, the lift of (MARITALSTATUS=WIDOWED, COUNTRY=GUATEMALA) becomes 1.46, indicating that these items in fact have a positive correlation in the Adult dataset.

Our definition of the lift on an itemset uses a bipartitional model. One might also define lift using an arbitrary number of partitions, up to the number of items in the itemset. Such a "full independence" assumption has been used to define lift [8, p. 310]:

$$\mathsf{lift}_1(I,\mathcal{D}) := \frac{\mathsf{freq}(I,\mathcal{D})}{\mathsf{freq}(\{i_1\},\mathcal{D}) \times \cdots \times \mathsf{freq}(\{i_k\},\mathcal{D})}.$$

However, this definition introduces an undesirable bias towards larger itemsets: many items with a slight negative correlation might have a lower lift than two items with a strong negative correlation, as the expected frequency does not take the number of items into account.

We illustrate the outlined issues with confidence and full independence lift in Table 6. This table displays the five least-likely itemsets using the different measures, where "pairwise lift" is the lift measure used throughout the paper for forbidden itemsets. All three measures agree that (SEX=FEMALE, RELATION=HUSBAND) is the most probable error in the dataset. Full independence lift's bias towards larger itemsets shows with respect to (RELATION=NOT-IN-FAMILY, MARITAL=MARRIED): pairwise lift detects this itemset as one of the least likely itemsets. Full independence lift, however, ranks multiple supersets of this problematic itemset as *less likely*. The addition of items, such as Age=≥50, for example, does not add much information, since the core problem is the co-occurence of Relation=Not-in-family and Marital=Married. When considering confidence instead of lift, we see the expected problems with items that are simply uncommon, but not necessarily correlated. For example, (AGE=18-21, EDUCATION=MASTERS) would not be very unlikely according to lift, and is probably not an error, but scores very low in terms of confidence because of the small number of people in the data with a Masters education.

K.2 Discovering Forbidden Itemsets

We continue by validating our algorithm for discovering forbidden itemsets (Section K.2.1) and the power of our pruning strategies (Section K.2.2).

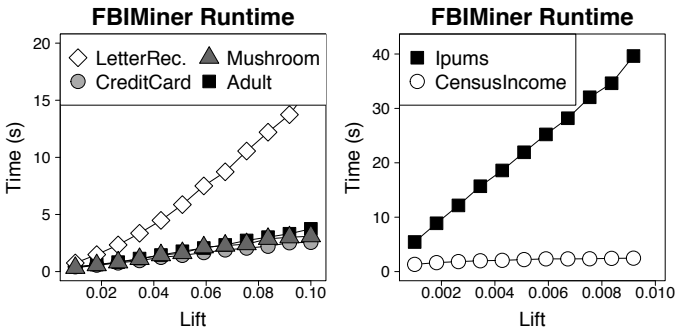


Figure 6. Runtime of FBIMiner in function of maximum lift threshold τ.

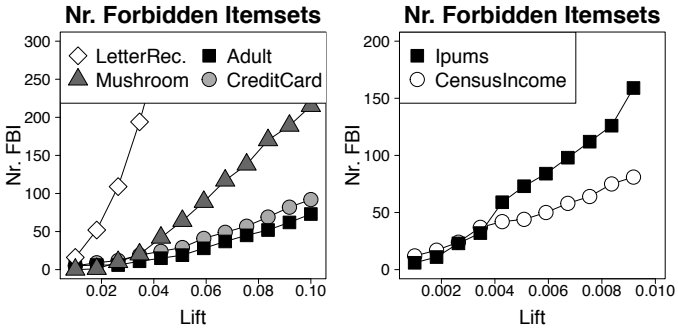


Figure 7. Number of Forbidden Itemsets in function of maximum lift threshold τ.

K.2.1 FBIMiner

We ran the forbidden itemset mining algorithm FBIMINER with full pruning, reporting the total runtime, the number of forbidden itemsets, and the number of objects containing a forbidden itemset, for increasing values of τ. For the larger datasets, Ipums and CensusIncome, a smaller τ range was considered. This prevents an explosion in the number of forbidden itemsets and the associated high runtime. The results are shown in Fig. 6, Fig. 7 and Fig. 8, respectively.

The results show that the runtime of the algorithm (Fig. 6) scales well with τ. As a result of the depth-first search, the runtime is strongly influenced by the number of distinct items. As a consequence, the algorithm runs slowest on the Ipums dataset. The runtime on the LetterRecognition dataset is explained by its relatively high number of items, and the fact that it contains many forbidden itemsets.

The number of forbidden itemsets (Fig. 7) is typically small, although there is a stronger than linear increase as the lift threshold increases, illustrating that τ should indeed be chosen very small. Especially for the LetterRecognition database, the number of forbidden itemsets increases exponentially. This is because the dataset is very noisy, since the contained letters were randomly distorted. In contrast, the less noisy Adult and CensusIncome datasets have relatively few dirty objects. The number of dirty objects (Fig. 8) naturally follows a similar pattern as the number of forbidden itemsets, with an occasionally big increase if a forbidden itemset with a relatively high support is discovered.

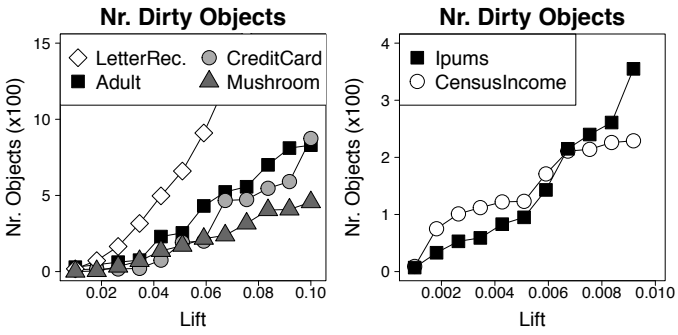


Figure 8. Number of objects containing one or more Forbidden Itemsets in function of maximum lift threshold τ.

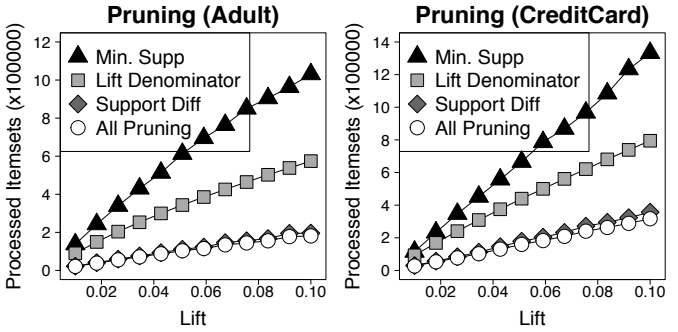
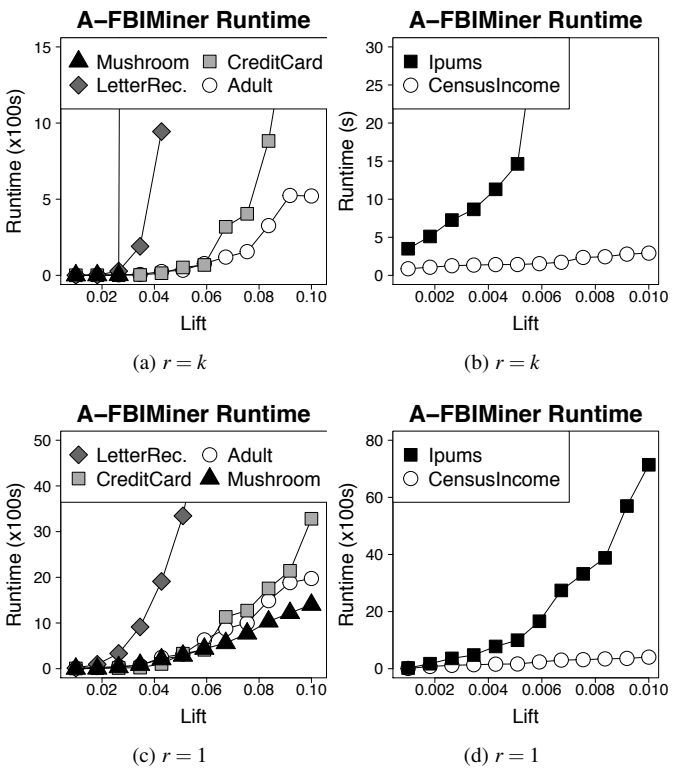


Figure 9. Number of itemsets processed with different pruning strategies in function of maximum lift threshold τ.



(a) $r = k$ (b) $r = k$

(c) $r = 1$ (d) $r = 1$

Figure 10. Runtime of A-FBIMINER in function of maximum lift threshold τ, for block sizes $r = k$ and $r = 1$.

K.2.2 Pruning

In order to evaluate the influence of the pruning strategies, we report the number of itemsets processed with only one type of pruning enabled, and contrast these with the number of itemsets processed when all pruning is enabled. We distinguish between *Min. Supp* pruning, using Prop. 6 on line 15 of Alg. 3; *Lift Denominator* pruning, using Prop. 8 on line 13; and *Support Diff* pruning, using Prop. 7 on line 22.

The results are shown in Fig. 9 for the Adult and CreditCard datasets; results for CensusIncome and LetterRecognition were similar. On the Mushroom and Ipums datasets, which have many attributes, FBIMiner became infeasible for larger values of τ without Support Diff pruning. Clearly, Support Diff pruning is dominant in most cases. Since this strategy also entails non-generator pruning, it is definitely crucial for the runtime of FBIMiner. Especially as τ increases, the other strategies also improve the overall result, indicating that all are beneficial and complementary to each other. We do not show the results when all pruning is disabled since all itemsets are then considered (independently of τ), leading to a high number of processed itemsets and running time.

A separate issue is the maximal frequency of a forbidden itemset, used on line 8 of Alg. 3. Recall that this is not a pruning strategy: using the frequency bound effectively increases the number of itemsets processed, since it disables the pruning strategies that require the lift to be computed first. On the other hand, the frequency bound may reduce runtime by avoiding certain unnecessary lift computations. This bound was disabled for the previous pruning results, to prevent painting a distorted picture of the influence of each pruning strategy. Table 7 shows the percentage influence of the frequency bound on the runtime without this bound, and on the number of visited itemsets. Clearly, these two stats are highly correlated. On the larger datasets, the negative impact on pruning is high, and hence the frequency bound results in a strong increase in runtime. On the other datasets, the impact of the frequency bound on pruning is limited, resulting in improved or unaffected runtimes. The results indicate that the maximal frequency bound is a useful optimization on smaller datasets, especially for smaller τ-values, but should be disabled for larger datasets.

K.3 Discovering Almost Forbidden Itemsets

We next focus on discovering almost forbidden itemsets (Section K.3.1) and the impact of block sizes on the algorithm (Section K.3.2).

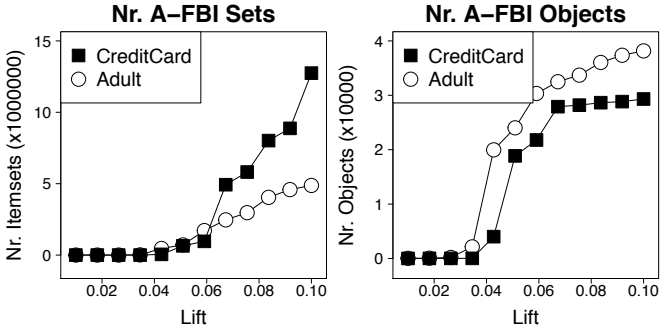
K.3.1 A-FBIMINER

The discovery of almost forbidden itemsets is the most computationally expensive part in our methodology. Recall that the runtime of A-FBIMINER depends both on the lift threshold τ and the number of dirty objects as discovered by FBIMiner. Since a larger τ automatically entails a higher number of dirty objects, clearly scalability in τ is an issue.

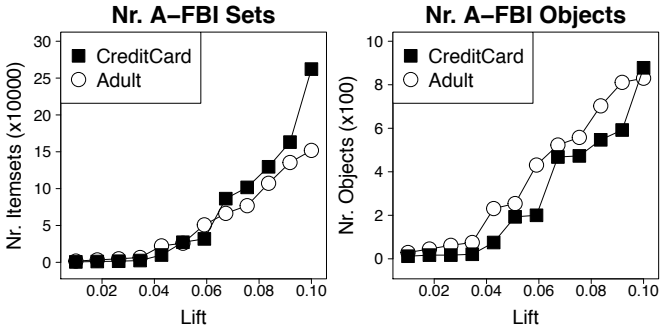
For each dataset and each τ, we first run algorithm FBIMINER to obtain the forbidden itemsets. Let k denote the number of dirty objects found. We then run algorithm A-FBIMINER a number of times with block size r, indicating the number of dirty objects to be repaired at once, until all k objects have been repaired. We first consider only the extreme cases of the block size, i.e., $r = 1$ and $r = k$. Experiments with other blocksizes are discussed in the next section. The obtained runtimes are shown in Fig. 10.

Table 7
Runtime influence of maximal frequency bound in function of τ. We show the percentage decrease/increase in runtine when using the frequency bound, as opposed to the runtime without frequency bound.

		τ-value					
Dataset	**Stat**	0.01	0.026	0.043	0.067	0.084	0.1
Adult	Time	-8%	-18%	-8%	-7%	-2%	-2%
Adult	Visited	+4%	0%	0%	0%	0%	0%
CensusIncome	Time	+16%	+23%	+25%	+28%	+30%	+31%
CensusIncome	Visited	+23%	+33%	+38%	+41%	+38%	+37%
CreditCard	Time	-8%	-18%	-10%	+1%	+1%	+3%
CreditCard	Visited	+5%	0%	0%	0%	0%	0%
Ipums	Time	+37%	+50%	+54%	+58%	+62%	+62%
Ipums	Visited	+38%	+49%	+56%	+61%	+63%	+64%
LetterRecognition	Time	-3%	+1%	0%	-2%	+2%	+1%
LetterRecognition	Visited	+1%	+1%	0%	0%	0%	0%
Mushroom	Time	-15%	-33%	-28%	-14%	-12%	-8%
Mushroom	Visited	+5%	+6%	+1%	0%	0%	0%



(a) $r = k$ (b) $r = k$



(c) $r = 1$ (d) $r = 1$

Figure 11. Number of discovered Almost Forbidden Itemsets and number of objects containing an Almost Forbidden Itemset, for block sizes $r = k$ and $r = 1$.

The difference between both block sizes is clear. For $r = k$, runtimes start out reasonably low, but quickly explode as the algorithm loses its pruning power and computation becomes infeasible. This is the most problematic for Mushroom, which has a larger number of attributes, and LetterRecognition, which has a very high number of dirty objects k. Block size $r = 1$ remains feasible throughout the τ range, but is slower overall.

Similar to FBIMINER, we are also interested in the number of itemsets returned by A-FBIMINER, and the number of objects containing such sets. We perform this analysis, and compare between block sizes $r = 1$ and $r = k$, on the Adult and CensusIncome datasets. These datasets permit us to investigate the entire τ-range, unlike the Mushroom and LetterRecognition datasets on which the mining of almost forbidden itemsets quickly becomes infeasible. Results are shown in Fig. 11.

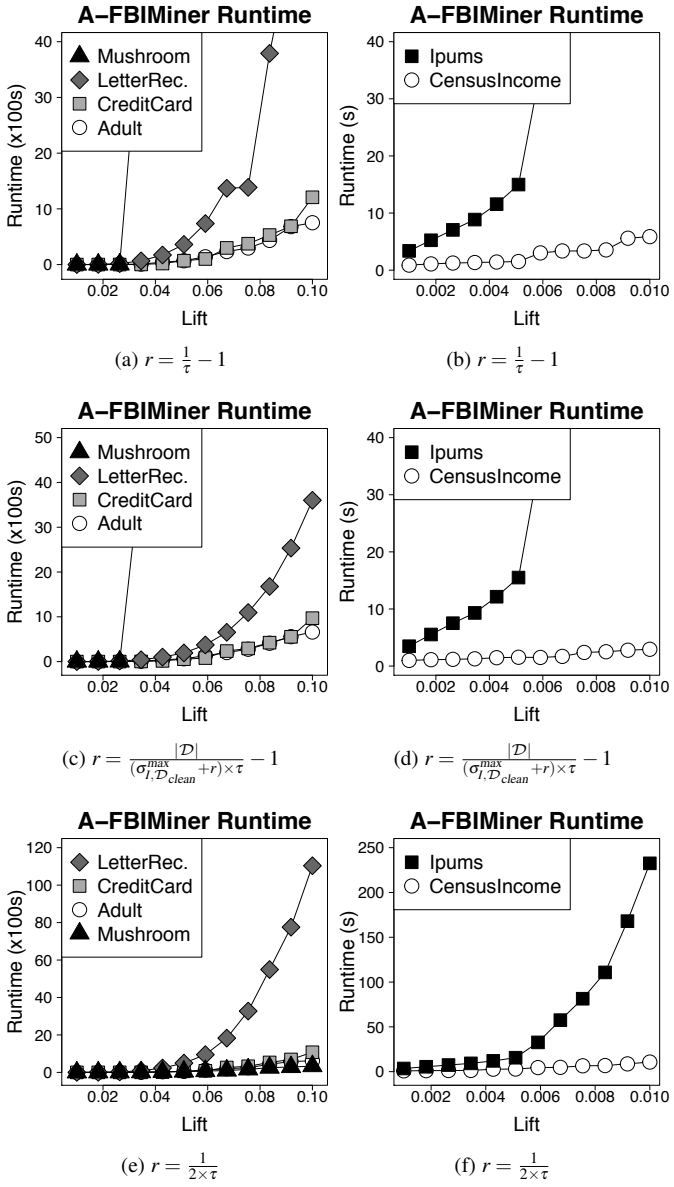
The obtained results show that the block size has a considerable impact on the number of almost forbidden itemsets, which in turn affect repairability. Indeed, when using block size $r = k$, we see that eventually every object in the CreditCard dataset contains at least one almost forbidden itemset, similarly for the Adult dataset. The number of almost forbidden itemsets is, predictably, very large. When using block size $r = 1$ instead, the number of almost forbidden itemsets is reduced by approximately a factor 100, and the number of objects containing an almost forbidden itemset never exceeds a moderate percentage of the total dataset.

K.3.2 Block Size

Next, we focus on the optimal block size r. As outlined in Sect. F, we can identify the quantities $\frac{1}{\tau} - 1$ and $\frac{|\mathcal{D}|}{(\sigma^{max}_{I,\mathcal{D}_{clean}} + r) \times \tau} - 1$ as the maximal block sizes for which Prop. 3 and Prop. 4, respectively, are still applicable.

Fig. 12a–12d displays the obtained runtimes using these block sizes. Note that the chosen values for r are τ-dependent. Consequently, for every τ-value, a different number of dirty objects is obtained and partitioned into blocks of a *different* absolute size. As expected, the runtimes are lower than for $k = 1$, while feasibility is better than for $r = k$, proving that the right block size indeed improves the overall performance of the algorithm. Runtime on the LetterRecognition dataset naturally suffers from the exponential increase in the number of dirty objects on that dataset. However, performance on the Mushroom dataset is still problematic: the number of attributes leads to a deep search tree, and pruning power is too limited; the same holds for Ipums.

As an alternative, we consider the block size $r = \frac{1}{2\tau}$, the halfway point between $r = 1$ and $r = \frac{1}{\tau} - 1$. Figure 12e–12f shows that this block size provides sufficient pruning power for the Mushroom dataset, and indeed outperforms all other considered sizes over the entire τ-range. For higher τ-values, the algorithm still struggles on the Ipums dataset, its high number of items



(a) $r = \frac{1}{\tau} - 1$ (b) $r = \frac{1}{\tau} - 1$

(c) $r = \frac{|\mathcal{D}|}{(\sigma^{max}_{l,\mathcal{D}_{clean}} + r) \times \tau} - 1$ (d) $r = \frac{|\mathcal{D}|}{(\sigma^{max}_{l,\mathcal{D}_{clean}} + r) \times \tau} - 1$

(e) $r = \frac{1}{2 \times \tau}$ (f) $r = \frac{1}{2 \times \tau}$

Figure 12. Runtime of A-FBIMINER in function of maximum lift threshold τ, for various block sizes r.

proving problematic. On the other datasets, A-FBIMINER is fast for low values of τ, and feasible across the considered τ-range.

K.4 Comparison with HoloClean

We have performed a basic comparison with HoloClean [27] as this method was shown to outperform other repairing methods. We ran HoloClean on a small sample from the Soccer dataset with errors from 50 CFDs, and a sample of the Adult dataset, each containing 10% of the original dataset. For each dataset, we first discovered forbidden itemsets, and then inserted them into HoloClean in the form of Denial Constraints. As mentioned before, HoloClean does not remove all forbidden itemsets, and in fact introduces *new* forbidden itemsets (which is not surprising as this is not the goal of their system). In Table 8, we show the number of discovered FBIs on the dataset, the number found on the dataset after repairing, and how many of these are *new* FBIs. From this, we conclude that state-of-the-art methods do not ensure

Table 8
Forbidden Itemsets found in Holoclean Repairs on the Soccer(50) and Adult datasets

Dataset	τ	$\lvert\text{FBI}(\mathcal{D})\rvert$	$\lvert\text{FBI}(\mathcal{D}_{Rep})\rvert$	$\lvert\text{FBI}(\mathcal{D}_{Rep}) \setminus \text{FBI}(\mathcal{D})\rvert$
Soccer	0.05	119	83	9
	0.1	297	232	35
	0.15	510	381	44
Adult	0.05	2	1	0
	0.1	10	90	83
	0.15	25	200	188

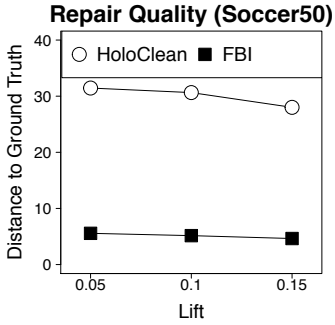


Figure 13. Distance between repaired dataset and ground truth, comparison between FBI and HoloClean, in function of threshold τ. Obtained on a sample of the Soccer dataset with errors generated by 50 cCFDs.

that no forbidden itemsets (and thus errors, in our setting) can be found in their repairs. This once more shows the usefulness of our dynamic notion of data quality.

HoloClean's seemingly bad performance, especially in terms of creating new forbidden itemsets on Adult, stems from a core difference with our approach. HoloClean exploits correlations between *attributes*, which are weak in our datasets, and combined with the way that our constraints and HoloClean's featurizer interact, Holoclean is able to detect errors, but struggles to fix them. We note that HoloClean does, in general, not guarantee that all errors are repaired due to the transformation of constraints to weak constraints. By contrast, our method finds repairs based only on *localized* value similarities between objects and does not suffer from these issues. At the same time, our method takes less global information into account than HoloClean, and will always repair a dirty object if a safe repair exists.

Additionally, in Fig. 13, we compare the distance between obtained repairs and ground truth on the sample of the Soccer dataset. Here, the repair obtained with FBI is much closer to the ground truth than the HoloClean repair. We did not tune HoloClean's parameters, which in combination with the relatively small size of the Soccer dataset, might account for the large difference in performance. From the comparison with HoloClean, we mainly conclude that our localized, high-precision approach based on a single parameter, shows much merit in certain scenarios.

K.5 Repairing Forbidden Itemsets

We conclude the experimental section with a further evaluation of the repair algorithm. For all experiments, the block size $r = \frac{1}{2\tau}$ was chosen, as described in the previous paragraph.

In Figure 14, we report the runtime of algorithm REPAIR on our selection of datasets, performed in parallel. It is clear that the repair algorithm is efficient. The plots are similar to those in

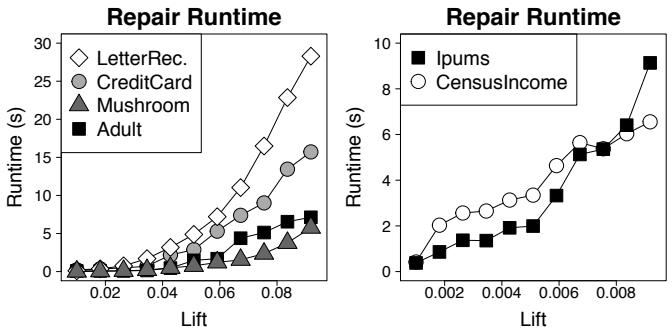


Figure 14. Runtime of the repair algorithm on various datasets, with repairs performed in parallel, in function of τ.

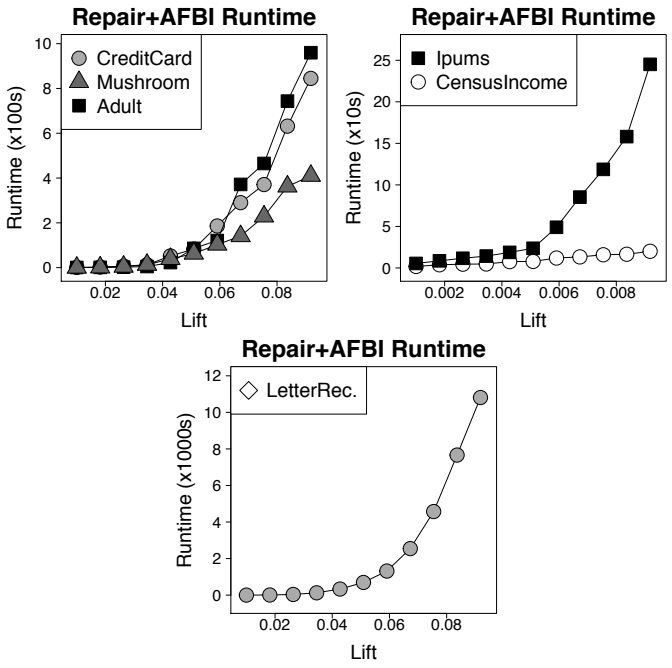


Figure 15. Runtime of repairing, including the mining of almost forbidden itemsets with blocksize $1/2\tau$, on various datasets.

Figure 8, showing the number of dirty objects, since the repair algorithm has to compute all possible modifications for each dirty object separately. Note that the repair algorithm, in itself, is largely independent of the value τ, which only has an indirect impact through the number of forbidden itemsets and dirty objects.

In the previous experiment, the runtime for algorithm REPAIR was shown in isolation. However, in order to perform correct repairs, it is also necessary to compute almost forbidden itemsets for every block of dirty objects. In Figure 15, we report runtimes which include the runtime of A-FBIMINER. While the runtimes increase noticeably, the overall scalability of repairing on the various datasets is still retained. Only on the LetterRecognition dataset, the runtimes of A-FBIMINER rise very strongly for larger values of τ, as already evidenced by the A-FBIMINER runtime experiments.

In Table 9, we report on the quality of the obtained repairs for various values of τ. We report the minimal and maximal similarity between a dirty object and its repair (within the τ range as above), with a similarity value of 1 indicating identical objects. The obtained repairs consistently have a high similarity

Table 9
Average quality of repairs.

Dataset	τ-range	Min-Max Sim.	$\lvert\mathcal{D}''\rvert$
Adult	0.01-0.1	0.94-0.95	1
CensusIncome	0.001-0.01	0.90-0.95	0
CreditCard	0.01-0.1	0.94-0.96	10
Ipums	0.001-0.01	0.95-0.98	94
LetterRecognition	0.01-0.1	0.96-0.98	33
Mushroom	0.01-0.1	0.94-0.99	238

...	Age	Marital Status	Relationship	Sex	...
	<18	Married-Civ.	Husband	Male	
	>50	Married-Civ.	Husband	Male	
	<18	Married-Civ.	Own Child	Female	
	<18	Never Married	Own Child	Female	
	22-30	Married-Civ.	Not in Family	Male	
	22-30	Never Married	Own Child	Male	
	22-30	Married-Civ.	Wife	Male	
	22-30	Married-Civ.	Wife	Female	
	31-50	Married-Civ.	Wife	Male	
	31-50	Married-Civ.	Husband	Male	

Figure 16. Example repairs on Adult dataset.

in the given τ-range. As such, our method obtains a dataset which doesn't contain any forbidden itemsets, without making drastic changes to the dirty objects.

We also report the number of objects that could not be repaired at the highest τ-value, denoted by $|\mathcal{D}''|$. For Adult, CensusIncome, CreditCard and LetterRecognition, only a few objects are unrepairable and this occurs only for high values of τ. A higher number of unrepairable objects is encountered for the Ipums and Mushroom datasets. This seems to suggest that a higher number of attributes causes problems for repairing.

For illustrative purposes, Fig. 16 shows example repairs obtained on the Adult dataset ($\tau = 0.01$).

As a final comment, we detail the similarity measure sim used in the REPAIR algorithm. We made use of the lin-similarity measure [9] which weights both matches and mismatches based on the frequency of the actual values:

$$\mathsf{linsim}(o,o') = \frac{\sum_{A\in\mathcal{A}} S(o[A],o'[A])}{\sum_{A\in\mathcal{A}} \log(\mathsf{freq}(\{(A,o[A])\},\mathcal{D})) + \log(\mathsf{freq}(\{(A,o'[A])\},\mathcal{D}))}$$

where $S(o[A],o'[A])$ is given by

$$\begin{cases} 2\log(\mathsf{freq}(\{(A,o[A])\},\mathcal{D})) & \text{if } o[A]=o'[A]\text{; and} \\ 2\log(\mathsf{freq}(\{(A,o[A])\},\mathcal{D}))+ & \\ \qquad \log(\mathsf{freq}(\{(A,o'[A])\},\mathcal{D})) & \text{otherwise.} \end{cases}$$

For example in the context of census data, a match or mismatch in gender would be more influential than a match or mismatch in the age category. Of course, any other similarity measure could be used instead. As part of future work, we intend to compare the influence of different similarity functions.

REFERENCES

[1] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *ICDT*, 1999, pp. 398–416.
[2] J.-F. Boulicaut, A. Bykowski, and C. Rigotti, "Approximation of frequency queries by means of free-sets," in *PKDD*, 2000, pp. 75–85.

[3] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li *et al.*, "New algorithms for fast discovery of association rules." in *KDD*, 1997, pp. 283–286.
[4] R. Bayardo, Jr., B. Goethals, and M. Zaki, Eds., *FIMI*, ser. CEUR Workshop Proceedings, vol. 126, 2004.
[5] T. Calders and B. Goethals, "Depth-first non-derivable itemset mining," in *SDM*, 2005, pp. 250–261.
[6] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin, "Efficient vertical mining of frequent closures and generators," in *Advances in Intelligent Data Analysis VIII*. Springer, 2009, pp. 393–404.
[7] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining." in *SDM*, 2002, pp. 457–473.
[8] M. J. Zaki and W. Meira Jr, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.
[9] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: A comparative evaluation," in *SDM*, 2008, pp. 243–254.