IT in Industry, vol. 6, no.3, 2018 Published online 11-Sep-2018

The D-basis Algorithm for Association Rules of High Confidence

Oren Segal, Justin Cabot-Miller, Kira Adaricheva and J.B.Nation

Abstract—We develop a new approach for distributed computing of the association rules of high confidence on the attributes/columns of a binary table. It is derived from the D-basis algorithm developed by K.Adaricheva and J.B.Nation (Theoretical Computer Science, 2017), which runs multiple times on sub-tables of a given binary table, obtained by removing one or more rows. The sets of rules retrieved at these runs are then aggregated. This allows us to obtain a basis of association rules of high confidence, which can be used for ranking all attributes of the table with respect to a given fixed attribute. This paper focuses on some algorithmic details and the technical implementation of the new algorithm. Results are given for tests performed on random, synthetic and real data.

Keywords—Binary table, association rules, implications, the D-basis algorithm, parallel computing, the relevance

I. Introduction

In data mining, the retrieval and sorting of association rules is a research problem of considerable interest. Association rules uncover the relationships between the attributes of a set of objects recorded in a binary table. This, for example, can be a transaction table, where the objects are sale transactions and the attributes are groups of products: position (r, c) in the table, in row r and column c, is marked by 1 if the transaction r includes a product from group c, otherwise, it is marked by 0. An association rule $X \to b$ in the table means that the entire set of transactions shows the tendency that whenever a transaction includes products from all groups in set X (i.e., there are 1s in all columns from X), then some product in group b will appear as well. The confidence of such a rule is measured by the portion of transactions that include b among all transactions that have every product from X.

In the world of transaction data, a rule $X \to b$ with confidence of 0.1 might demonstrate that b *sells together* with all products in group X.

There is an immense effort in the data mining community to develop reliable tools for the discovery of meaningful association rules. However, the hurdles encountered while developing such solutions are numerous. The benchmark algorithms, such as *Apriori* in Agrawal et al. [6], have time complexity that is exponential in the size of the input table. Moreover, the number of association rules is staggering, and thus analyzing them requires further tools to obtain a short subset of rules that are significant. There are no strong mathematical results confirming a particular choice of such short subsets, and numerous approaches to the filtering process are described in various publications devoted to the topic. See, for example, Kryszkiewicz [17] and Balcázar [8]. Recent approaches include constraint-based patterns, preference learning, instant mining and pattern space sampling, which are often interactive methods targeting the user's implicit preferences, see [7], [22], [23].

One particular subset of association rules, the *implications*, or rules of full confidence, merit particular attention in data mining. They are also at the center of ongoing theoretical research. From a practical point of view, implications are the strongest rules available in a given table because they hold true for any row of the table.

Some types of data present cases where the implications uncovered in the table contain non-essential information because the support of those implications might be very low. The support of an association rule $X \to b$ is the number of rows where ones appear in all columns from $X \cup b$. For example, the transaction data might have only a few implications with small sets X, whose support could be a single-digit percentage of all transactions. Mining of transaction data tends to uncover rules of lower confidence but relatively large support, rather than those implications which hold everywhere.

The Apriori algorithm and the concept of generating rules of lower confidence are relevant when considering transaction data, but not so for medical data. The attributes of a table that represent genetic and clinical data of patients (rows) may have a tighter connection than relationships in transaction data, in which case the confidence of relevant association rules could be expected to be high and closer to 1. Implications would serve as the imperfect representation of the laws of nature in this data.

At the same time, every data set may contain errors, missing entries and miscalculations. Additionally, some patients may have extraneous conditions affecting the value in the target column (e.g., co-morbidity with an untracked illness). Even if only one row contains such deviations, it may prevent us from discovering important implications that would otherwise hold in the table. This is why extracting implications from sub-tables that omit only a few rows may uncover important rules.

In this paper, we expand the approach developed in Adaricheva et al [2] for the extraction of implications and ranking of attributes with respect to a target attribute.

Our goal is not to uncover particular rules and rank them with respect to some measurement. Rather, we want to gener-

O. Segal is with the department of Computer Science, Hofstra University, Hempstead, NY, 11549 USA e-mail: Oren.Segal@hofstra.edu

J. Cabot-Miller is CS and mathematics major, Hofstra University e-mail: JCabotMiller1@pride.hofstra.edu

K. Adaricheva is with the department of Mathematics, Hofstra University, Hempstead, NY, 11549 USA e-mail: Kira.Adaricheva@hofstra.edu

J.B. Nation is with the department of Mathematics, University of Hawaii, Honolulu HI 11823 USA e-mail: JB@math.hawaii.edu

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

ate a basis Σ of association rules which satisfactorily describes dependencies among attributes. We could then use Σ to rank the importance of attributes with respect to a target attribute, b. In medical data, b may describe high survival probability of a patient after particular treatment, while other attributes may record physical parameters in the patients. Having large Σ is not necessarily a bad feature; on the other hand, an optimally small set is desirable. Consider the case where $X \to b$, $X \cup c \to b$ and $X \cup d \to b$ are in the basis and have high confidence. We may want to keep $X \to b$ in Σ and remove the other rules, which may unnecessarily inflate the importance of attributes c and d for b. This is because attributes c, d could be completely unrelated to b, although they appear because the actual "law" $X \to b$ is *blocked* in several rows.

Since the initial version of the paper was accepted for the Proceedings of DTMN-2018 conference [21], the D-basis code implementation was expanded to multiple-thread computation version, which is presented in detail in section 7. We also conducted more testing of synthetic data in section 6.2 featuring a prominent rule and a high relevance of a few attributes to one fixed attribute, then applied various levels of noise to the input to test whether the multiple row reduction runs of the algorithm would reconstruct original higher ranking of prominent attributes.

As far as a structure of the paper is concerned, we combine both algorithmic and practical issues of the algorithm and report on various testing approaches. We give a short description of the proposed algorithm in section 2, then in section 3, we explain how the association rules are used to compute the relevance parameter of one attribute with respect to the other.

Sections 4 and 5 are devoted to algorithmic aspects of the D-basis program that can improve performance of the code for the one-or-several-rows deletion process. Then in section 6.1 we discuss the levels of the relevance parameter in random matrices. It shows that the high relevance may occur in random data with the high density of ones.

Finally, in section 8 we outline the further direction of the work, either with additional experimentation on relevance computation and MRD process, or application in various data sets.

II. General flow of the algorithm

Herein we describe a several-stage approach that allows us to compute, beyond just implications, the potentially most valuable association rules, those whose confidence is rather high, say, > 0.9.

At the core of our approach is the connection between a binary table, its implications, and the closure operator defined on the set of columns and associated lattice of closed sets, known as the concept lattice or Galois lattice, see [14].

An algorithm in Adaricheva and Nation [1] works to extract the basis of implications of the table, and it is known as the *D-basis*, which was introduced in Adaricheva, Nation, and Rand [3]. The advantage of this basis is in the possibility to use algorithms for dualization of an associated hypergraph that are known to be sub-exponential in their complexity, see Fredman and Khachiyan [12]. The algorithm in [1] avoids generating the Galois lattice from the table and only uses the arrow relations, which can be computed in polynomial time, to produce a hypergraph for each requested attribute. In that way, the existing code for hypergraph dualization, such as in Murakami and Uno [18], can be borrowed for execution.

The main idea of the current algorithm follows from the observation that the association rules of high confidence may be computed by removing one or more rows (objects) from the table and computing implications (on attributes) of a shorter table. Upon the program's execution and output, one can record only new implications that were not present in the original data set.

A new rule may be derived from the shorter table given that one of the removed rows fails it. If new rules are present in this table and exhibit high support, or if numerous new rules are found with average support, then the row(s) temporarily removed to form this shorter table are called *blockers* due to their tendency to block the rules that were found with their removal.

We can choose various strategies to identify the set of blockers. Together with several straightforward statistics on new rules found on a shorter table, we are considering several heuristics and ranking systems. The goal is to identify a set S of rows/objects that are potential blockers.

In the next stage of the suggested procedure, we choose $n \leq |S|$ which is a number of rows from S that will be deleted from original table to form a shorter table. With S and n fixed, we can run algorithm k times, where k is specified by a user, and limited by $C(|S|, n) = \frac{n!}{|S|!(|S|-n)!}$, which is a number of combinations to choose n rows from set S. If $k < C(|S|, n)$, then the choice can be done randomly, otherwise, the rows can by systematically removed.

This process of removing sets of rows and re-running the program can be organized in parallel, and all the outputs are combined and aggregated following the same procedure as used to retrieve the D-basis of implications in the original D-basis algorithm.

The final set of rules is guaranteed to have the probability of at least $\frac{N-n}{N}$ that each rule holds in a table, where N is the total number of rows (objects) in the table, and n is the number of deleted rows in each run of the algorithm. If s is the support of some implication $A \to b$ in a shortened table with $N - n$ rows, then on average the support of A in n deleted rows will be $s \cdot \frac{n}{N-n}$.

In the worst case scenario, i.e. when $b = 0$ in all deleted rows, the support of $A \to b$ on n deleted rows will be 0. This gives a lower estimate for the confidence of association rule $A \to b$ in the full table as $\dfrac{s}{s + s \cdot \frac{n}{N-n}} = \dfrac{N-n}{N}$.

For example, given an original table of 90 rows, the confidence of a rule found as an implication in a sub-table of 80 rows, i.e., after deleting 10 rows, will be, on average, around $\frac{80}{90} = 0.88$.

III. Ranking the attributes of the table with respect to a given fixed attribute

The algorithm described in the previous section, when we try to identify the *blockers* among the objects/rows of the data,

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

can be interpreted as *unsupervised* learning about the data set. These blockers are then interpreted as outliers.

In this section we will take a look at *supervised* exploration of the data set, when one of the parameters is a target column/attribute, and we try to discover other attributes that might be essential for describing the behavior of the target parameter.

The algorithm in [1] allows us to retrieve only those implications $X \to b$ in the D-basis that have a fixed attribute b as a conclusion. This is called a *b-sector* of the basis. It is important to notice that one does not need to obtain the full basis in order to get particular b-sector of the basis.

In our current approach, instead of a set of implications, we obtained a set Δ of association rules, where we have traded the confidence for higher support of the rules. We choose a number of shorter tables, as described in algorithm of section 2, and compute b-sectors of implications. A final part of the algorithm then performs a special trimming of the rules, called an *aggregation*, leaving only the *strongest* rules with respect to the *binary part* of Δ, which consists of rules of the form $a \to d$. Thus, in order to obtain the b-sector of basis Δ, one needs only the binary part of Δ and combination of b-sectors of implications of shorter tables. The resulting b-sector of Δ after its aggregation will be denoted $\Delta(b)$. Note that one can generate multiple sets $\Delta(b)$, so that the following computations will depend on the particular instance of $\Delta(b)$. For example, a user may decide to include into $\Delta(b)$ only the rules whose support exceeds some given threshold $minsup$.

Similarly, we could form $\Delta(\neg b)$, where the original attribute is replaced by its complement $\neg b$.

Having fixed $\Delta(b)$ and $\Delta(\neg b)$, we then used the approach described in [2] to rank the attributes by the *relevance* parameter.

For any attribute a, the relevance $rel_b(a)$ of a to b is computed based on frequency of a appearing in the antecedents of implications/association rules related to b in b-sectors $\Delta(b)$ and $\Delta(\neg b)$. From this definition one can see some relation between the relevance parameter and the *conviction*, see for example [20]. The computation of this parameter takes into account the support of each individual implication in the basis where a appears. Since this time we have association rules of different confidence, we include the confidence into computation of the relevance as well. For a rule $\alpha = (X \to b)$, $conf(\alpha) = \frac{sup(X \cup b)}{sup(X)}$.

We believe that, for each attribute $a \in A \setminus b$, the important parameter of relevance of this attribute to $b \in A$ is a parameter of *total support*, computed with respect to set of rules Δ:

$$tsup_b(a) = \Sigma\{\frac{|sup(X)|}{|X|} \cdot conf(X \to b) : a \in X, (X \to b) \in \Delta(b)\}$$

Thus $tsup_b(a)$ shows the frequency of parameter a appearing together with some other attributes in implications $X \to b$ of the set $\Delta(b)$. The contribution of each implication $X \to b$, where $a \in X$, into the computation of total support of a is higher when the support of X is higher, i.e., column a is marked by 1 in more rows of the table, together with other attributes from X, but also when X has fewer other attributes besides a.

While the frequent appearance of a particular attribute a in implications $X \to b$ might indicate the relevance of a to b, the same attribute may appear in implications $X \to \neg b$.

Replacing $\Delta(b)$ by $\Delta(\neg b)$ in above formula, we can also compute the *total support* of $\neg b$, for each $a \in A \setminus b$:

$$tsup_{\neg b}(a) = \Sigma\{\frac{|sup(X)|}{|X|} \cdot conf(X \to \neg b) : a \in X, (X \to \neg b) \in \Delta(\neg b)\}$$

Define now the relevance of parameter $a \in A \setminus b$ to parameter b, with respect to bases $\Delta(b)$ and $\Delta(\neg b)$:

$$rel_b(a) = \frac{tsup_b(a)}{tsup_{\neg b}(a) + 1}.$$

The highest relevance of a is achieved by a combination of high total support of a in rules $X \to b$ and low total support in rules $X \to \neg b$. This parameter provides the ranking of all parameters $a \in A \setminus b$.

As indicated above, the computation of the relevance can be done not only with implications but with any set of association rules Δ. We believe that association rules of high confidence may provide a better set for computation of the relevance.

Observation with the data shows, and theoretical results confirm [4], [5], that a rule $X \to b$ that fails in one or a few rows of table may appear through the set of implications $X \cup d \to b$, with multiple attributes d, which may inflate $tsup_b(a)$ for element $a \in X$. When one or a few rows failing the rule $X \to b$ are deleted, then $X \to b$ will be discovered, and the process of the aggregation will eliminate all the rules $X \cup d \to b$ from the final set of rules used for computation of the relevance.

IV. Algorithmic aspects one-or-several-object deletion in binary tables

In this section we consider some aspects of the algorithm of the D-basis retrieval, when there is a modification to the original binary table $T = (U, A, R)$, in particular, one or more elements in set U are deleted. Here U is a set of objects/rows, A is the set of attributes/columns, and $R \subseteq U \times A$ is the subset of table entries marked as 1. We refer to the description of the D-basis algorithm given in [1]. Skipping this and the following sections will not affect the understanding of the rest of the paper.

The first procedure of the D-basis algorithm is the reduction of sets A and U.

Our goal is to show that, when we need to run the algorithm on the table with one or several elements of U deleted, one can skip the step of row reduction in the algorithm and re-use the set U' from the process run on the original table.

Recall that relation R can be reduced to a relation $R' \subseteq U' \times A'$, where $U' \subseteq U$, $A' \subseteq A$ and $R' = R|_{U' \times A'}$, so that $|U'|$ and $|A'|$ are minimal with respect to property $L'_R \simeq L_R$, where L_R is the Galois lattice associated with table $T = (U, A, R)$, similarly, for L'_R. Check the discussion of the reduction procedures for the general closure systems, for example, section 2 of [3]. In other words, one may leave only essential objects and attributes in the relation and remove the others. The elements of U' are in bijection with the set of meet-irreducible elements of Galois lattice L_R, and elements

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

of A' are in bijection with the set of join-irreducible elements of L_R. Moreover, one has 1 in the table T' in row u and column a iff $a \leq u$ in the lattice L_R.

Let us discuss in more detail how the reduction from T to T' occurs. Element $a \in A$ may be reduced (leaving L_R unchanged) if there exists a subset $X_a \subseteq A \setminus \{a\}$ such that $S_U S_A(\{a\}) = S_U S_A(X_a) = \bigcap_{x \in X_a} S_A(x)$. Recall that S_A is a standard mapping from A to U defined as $S_A(Y) = \{u \in U : (u, y) \in R, \text{ for all } y \in Y\}$, for $Y \subseteq A$. Symmetrically, $S_U(Z) = \{a \in A : (z, a) \in R, \text{ for all } z \in Z\}$, for $Z \subseteq U$.

When $X_a \neq \emptyset$, this is equivalent to the fact that double implication $a \leftrightarrow X_a$ holds in closure system $(A, S_U \circ S_A)$. Such an implication is easy to check in a binary table: $(u, a) \in R$ iff $(u, x) \in R$ for all $x \in X_a$.

There is also a special case when column a has only 1 entries, which corresponds to $X_a = \emptyset$ and the set of implications $a' \to a$ for all $a' \in A$. Another special case is when column a has only 0 entries, which corresponds to the set of implications $a \to a'$ for all $a' \in A$, but removal of a in this case may potentially remove the top element of L_R.

Similarly, an element $u \in U$ may be reduced (leaving L_R unchanged) if there exists a subset $X_u \subseteq U$ such that $S_A S_U(\{u\}) = S_A S_U(X_u) = \bigcap_{x \in X_u} S_U(x)$. Special cases are also with all 0 or all 1 in row $u \in U$, similar to those for $a \in A$.

A table $T' = (U', A', R')$ is called *reduced*, if rows and columns are non-trivial and $S_U S_A(a) \neq S_U S_A(X_a)$ for any $a \in A$, $X_a \subseteq A' \setminus \{a\}$, and, similarly, $S_A S_U(u) \neq S_A S_U(X_u)$ for any $u \in U$, $X_u \subseteq U' \setminus \{u\}$.

The following observation is the basis for the reduction process.

Lemma 1. *Given a binary table $T = (U, A, R)$, suppose $S_U S_A(\{a\}) = S_U S_A(X_a) =$ $= \bigcap_{x \in X_a} S_U S_A(x)$, for some $a \in A, X_a \subseteq A$. Consider the table $T^* = (U, A \setminus \{a\}, R)$. Then an element $u \in U$ may be reduced in $T^* = (U, A \setminus \{a\}, R)$ iff it may be reduced in the original table $T = (U, A, R)$.*

Proof: Denote $[u, a] = 1$, if $(u, a) \in R$, and 0 otherwise.

If u can be reduced in $T = (U, A, R)$, then there exists (assume non-empty) $X_u \subseteq U$ with $S_U(\{u\}) = S_U(X_u) = \bigcap_{x \in X_u} S_U(x)$. This is equivalent to $[u, a'] = \Pi_{u' \in X_u}[u', a']$, for any $a' \in A$. If one ignores $a \in A$ that is getting reduced, then equality still holds for all $a' \in A \setminus \{a\}$. Therefore, the same u may be reduced in $(U, A \setminus a, R)$.

Vice versa, consider a (non-trivial) u that may be reduced in $T^* = (U, A \setminus a, R)$. We need to check that $[u, a] = \Pi_{u' \in X_u}[u', a]$.

If $[u, a] = 1$, then due to the assumption about a, we have $[u, a] = \Pi_{a' \in X_a}[u, a']$, therefore, $[u, a'] = 1$ for all $a' \in X_a$. Now $X_a \subseteq A \setminus \{a\}$, so we may apply the assumption about reducibility of u in $(U, A \setminus a, R)$: for each fixed $a' \in X_a$, we have $1 = [u, a'] = \Pi_{u' \in X_u}[u', a']$, therefore, $[u', a'] = 1$ for all $u' \in X_u, a' \in X_a$. If we now fix $u' \in X_u$, then we can compute $[u', a] = \Pi_{a' \in X_a}[u', a'] = 1$. Therefore, every factor on the right of the formal equality $[u, a] = \Pi_{u' \in X_u}[u', a]$ is equal to 1, turning it into true equality.

If $[u, a] = 0$, then due to the assumption about a, we have $[u, a] = \Pi_{a' \in X_a}[u, a']$, so that $[u, a'] = 0$ for at least one $a' \in X_a$. This implies, due to the reducibility of u in $(U, A \setminus a, R)$ that $[u', a'] = 0$ for at least one $u' \in X_u$. Again, due to assumption about a, we have $[u', a] = \Pi_{a'' \in X_a}[u', a''] = 0$. Hence, the right side of formal equality $[u, a] = \Pi_{u' \in X_u}[u', a]$ turns into 0, thus, making it true equality.

It is easy to verify the statement in trivial cases when a or u has all 0 or all 1 entries. ■

It follows from the Lemma that one may reduce element $u \in U$ before or after reducing element $a \in A$. Also, the dual statement can be done switching the roles of U and A in Lemma 1.

The following statement looks into a typical setting of the row deletion process that is central in this paper.

Proposition 1. *Suppose table $T = (U, A, R)$ has a reduced form $T' = (U', A', R')$, with $U' \subseteq U, A' \subseteq A$. If $u^* \in U'$ is deleted, then $T^* = (U \setminus u^*, A, R)$ can we reduced to $T'' = (U' \setminus u^*, A'', R'')$.*

Proof: First, check that both tables $(U' \setminus u^*, A, R)$ and $(U' \setminus u^*, A', R')$ are reduced with respect to their set of objects. Indeed, if $u' \in U' \setminus \{u^*\}$ could be reduced, then, for some $X_u \subseteq U' \setminus \{u^*\}$, $[u', a] = \Pi_{u'' \in X_u}[u'', a]$, for any $a \in A$. This would mean that u' can be reduced in table T', which contradicts the assumption.

On the other hand, table $(U' \setminus u^*, A', R')$ may become reducible with respect to set A'. This will occur if, for some choice of $a \in A', X_a \subseteq A' \setminus \{a\}$, the equality $[u, a] = \Pi_{a' \in X_a}[u, a']$ holds for all elements $u \in U'$ except u^*. Then the element a becomes reducible after u^* is deleted.

So assume that we reduced the set A to $A'' \subseteq A'$. Then due to Lemma 1, after some element $a \in A'$ is reduced, the set $U' \setminus \{u^*\}$ still cannot be reduced. ■

Apparently, consecutive application of Proposition 1 brings to the similar statement for a multiple row deletion case. This proposition can be used to reduce overhead when running multiple processes on the table after deleting one or several rows.

V. Arrow relations in the original table and after one-object reduction

The second step of the D-basis algorithm in [1] deals with establishing arrow relations between objects and attributes of the reduced table. In this section we look into some overhead reduction for the computation of arrow relations, when one of the objects/rows is deleted.

We may assume that we are working with a reduced table $T' = (U', A', R')$. The binary table allows one to quickly recover additional information on L'_R ($Mi(L'_R)$ and $Ji(L'_R)$ stand for meet-irreducibles and join-irreducibles of the Galois lattice L'_R, respectively):

(1) Establishing a partial order $(U', \leq) = (Mi(L'_R), \leq)$;
(2) Establishing a partial order $(A', \leq) = (Ji(L'_R), \leq)$;
(3) Establishing arrow relations $\uparrow$, $\downarrow$, and $\updownarrow$.

Recall that $u_1 \leq u_2$, for $u_1, u_2 \in U'$, iff $(u_1, a) \in R'$ implies $(u_2, a) \in R'$ for every $a \in A'$. Symmetrically, $a_1 \leq a_2$, for $a_1, a_2 \in A'$, iff $(u, a_2) \in R'$ implies $(u, a_1) \in R'$, for every $u \in U'$.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Now for $a \in A'$ and $u \in U'$, $a \uparrow u$ is defined to hold iff u is a $\leq$-maximal element among elements of U' that are not greater than a.

Dually, $a \downarrow u$ iff a is a $\leq$-minimal element among elements of A' that are not less than u.

Finally, $a \updownarrow u$, if both $a \uparrow u$ and $a \downarrow u$ hold.

It is clear that algorithmically, the reconstruction of arrow relations is equivalent to finding maximal or minimal elements in a particular partially ordered set, which are sub-posets of either $(U', \leq)$ or $(A', \leq)$.

Now suppose that the element $u^* \in U'$ is deleted in table T', so that $T^* = (U' \setminus \{u^*\}, A', R')$. According to the discussion in the previous section, the table may be reduced to $T'' = (U' \setminus \{u^*\}, A'', R'')$, where $A'' \subseteq A'$, i.e. only the set A' might be reduced. Then

(1) Partial order on $(U' \setminus \{u^*\})$ induced by table T'' is the extension of $\leq$ defined on elements of U';
(2) Partial order on $(A'', \leq)$ induced by table T'' is the extension of the sub-poset induced by $(A', \leq)$ on A'';

This implies the following statement that reduces the computation of $\uparrow$, $\downarrow$ and $\updownarrow$ in T''.

Lemma 2. *Suppose $T'' = (U' \setminus \{u^*\}, A'', R'')$ is a reduced sub-table of another reduced table $T' = (U', A', R')$ after deletion of item u^*.*

- *If $u_1 \leq u_2$ or $a_1 \leq a_2$ in T', then the same is true in T''.*
- *If $a \uparrow u$ in T'', then $a \uparrow u$ in T'. For every $a \uparrow u_1$ in T' which does not hold in T'', there exists another $a \uparrow u_2$ in T' such that $u_1 \leq u_2$ in T''.*
- *If $a \downarrow u$ in T'', then $a \downarrow u$ in T'. For every $a_1 \downarrow u$ in T' which does not hold in T'', there exists another $a_2 \downarrow u$ in T' such that $a_2 \leq a_1$ in T''.*

The check is straightforward and left to the reader.

VI. The relevance computation in random matrices and synthetic data

A. Random matrices

In this section we look into computation of the relevance parameter in random matrices. As was discussed in section III, the relevance parameter $rel_b(a)$ measures the frequency of attribute a appearing in the rules $X \to b$ in comparison with the rules $X \to \neg b$. We note that the computation depends on the set of association rules Δ that is chosen for the measurement.

According to the formula for the relevance, only attributes a with $rel_b(a) > 1$ are of interest as possible "influencers" for behavior of b. On average, one would expect that values of relevance values in random matrices would tend to be close to 1.

It turns out, and the probabilistic argument as well as testing confirm, that random matrices with high density of ones may generate high values of the relevance of one attribute to the other. We ran a large number of test experiments showing that

(1) the average value of the relevance of an attribute in the table to one fixed attribute remains close to 1.5 for the densities of the table between 0.3 and 0.6.
(2) In the tables with densities of ones higher than 0.7, the relevance exhibits considerable grows.
(3) In the tables with densities below 0.3 the average values of relevance stay below 1.

This suggests that relevance could be reliable tool to rank the "importance" of attributes with respect to one fixed attribute in the range of densities of ones in a table up to 0.6.

We will assume that the randomness of the matrix is specified by parameter $q = 1-p$, which is a probability of any entry being 1, so that the probability of 0 is p. The matrix can be obtained through the process of generating its entries, every time choosing a real number r randomly in interval [0,1], and placing 1, when $r \geq p$, and 0 otherwise. Then the density of ones in resulting matrix should be expected close to q.

In general, it can be computed that the probability of fixed implication $X \to k$, $|X| = x$, having support t in random matrix with m rows and density q is given by formula

$$P(tsup(X \to k) = t) = C(t, m)q^{(x+1)t}(1 - q^x)^{m-t}.$$

The formula indicates that the probability could be relatively large when q is high and t is high, but diminishes for smaller t. For example, in a matrix with number of rows $m = 10$, $t = 5$ and $q = 0.8$, for any choice of distinct attributes i, j, k:

$$P(tsup(ij \to k) = 5) = C(5, 10)(0.8)^{15}(0.2(1.8))^5 \approx 0.05361.$$

This is considerably higher than the probability of such an implication having support $t = 2$:

$$P(tsup(ij \to k) = 2) = C(2, 10)(0.8)^6(0.2(1.8))^8 \approx 0.0033,$$

In the D-basis produced by algorithm on random matrix $10\times$ 22 with $q = 0.8$, we have 8 implications $i \to k$, 13 implications $ij \to k$ and one implication $ijs \to k$. All implications $i \to k$ have support 7 and 8, except for one with support 6, and all implications $ij \to k$ have support between 4 and 7, most with support 5 and 6.

When the relevance is computed for any attribute x with respect to attribute k, the column of attribute k is inverted to $\neg k$, and thus the density of that column turns to $1 - q$. Therefore,

$$P(tsup(X \to \neg k) = t) = C(t, m)q^{xt}(1 - q)^t(1 - q^x)^{m-t}.$$

For $q = 0.8$ we get $p = 0.2$, thus

$$P(tsup(ij \to \neg k) = 5) = C(5, 10)(0.8)^{10}(0.2)^5(0.2(1.8))^5 \approx 0.05E - 3.$$

This explains why one could expect relatively high possible values for relevance parameter $rel_k(i)$, when q is high.

When switching to relatively low values of q, there is higher probability of implications of low support. For example, with $q = 0.3$

$$P(tsup(ij \to k) = 2) = C(2, 10)(0.3)^6(0.7(1.3))^8 \approx 0.0154,$$

$$P(tsup(i \to k) = 7) = C(7, 10)(0.3)^{21}(0.7(1.3))^3 \approx 9.45E{-}10,$$

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

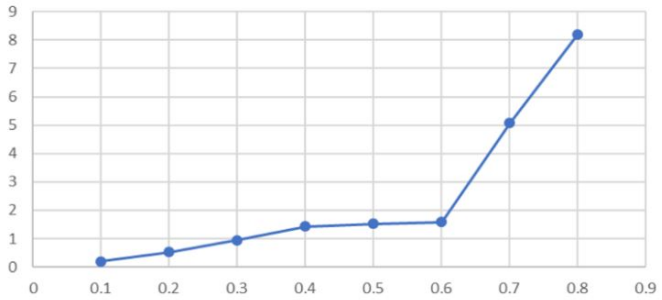

Fig. 1: The average value of the relevance (y-axis) depending on the density of a table (x-axis)

This time, when switching to $xy \to \neg k$, we may see probability increased. For example,

$$P(tsup(ij \to \neg k) = 2) = C(2,10)(0.3)^4(0.7)^2(0.7(1.3))^8$$
$$\approx 0.0839,$$

As a result, lower values of $rel_k(i)$ are expected for low values of q.

We tested a large series of random matrices of the same size 20×32, computing the relevance of attributes $1, \dots 31$ to $k = 32$, while varying $q = 0.1$ to 0.9. The average values of the relevance values are summarized in the Figure 1.

Several thousand random binary tables of fixed size 20×32 and two different densities were analyzed, and their relevance characteristics are described in *Table 1*.

TABLE I: Example of the relevance at two densities

Density	Max	Average	50th percentile	75th percentile	90th percentile
.3	19.75	.951	.520	1.182	2.212
.5	121.833	1.523	.972	1.476	2.718

When the relevance is computed in real data, the values should be adjusted to the average expected for particular density. For example, when analyzing data with $q = 0.4$, one would expect the average relevance to be 1.5. Therefore, if $rel_k(i) = 1.3$ for some parameters i, k, this indicates that i is not particularly important for k, even though $rel_k(i) > 1$.

B. Synthetic data with a prominent rule

In this section we describe the test with synthetic data: a binary table of size 50×50, which was initially generated as a random table of density $d = 0.3$, but then 20 rows were modified to enforce the rule $1, 2, 3, 4 \to 50$ that had support in the first 16 rows, and the next 4 rows were modified to fail one of the shorter rules: $1, 2, 3 \to 50$, $1, 2, 4 \to 50$, $1, 3, 4 \to 50$, or $2, 3, 4 \to 50$. This would ensure the optimality of the rule $1, 2, 3, 4 \to 50$, i.e., that no shorter rule $Y \to 50$ with $Y \subseteq \{1, 2, 3, 4\}$ would hold in the table. The rest of the table was checked and slightly modified to guarantee that the rule $1, 2, 3, 4 \to 50$ did not fail in all remaining rows.

Then we imposed various levels of noise: whenever the noise was at level $p \in (0, 1)$, with probability p each entry of the table would switch from 0 to 1 or from 1 to 0.

We used the following approach to estimate the number of rules that need to be removed in order to reveal the rule that fails due to noise.

If we have that the rule $X \to b$ fails in some tabled data with noise, but should hold in the original table without the noise, then a good estimate for the number of rows where the rule may fail due to the noise is

$$N = ndp,$$

where n is the number of rows in the table, d is the density of ones in column b (thus, nd=number of ones in column b), and p is the probability that a 1 in column b is turned to 0 due to noise. Indeed, most failures of the rule will occur in rows of the table which would support the rule without noise, but with a 1 in column b erroneously switched to 0.

Under additional assumptions that the density d in the table does not change much due to noise, nd gives approximately the number of ones in column b in the original table, whence ndp will give the number of rows where 1 switched to 0 in column b.

With this estimation, if one deletes N rows where the rule $1, 2, 3, 4 \to 50$ fails, the rule will hold in all remaining rows of the sub-table, so that it can be uncovered by the D-basis algorithm processed on the sub-table.

One of the options to run the multiple row deletion algorithm based on the D-basis is to choose the number k of row that will be deleted for every run of the algorithm, and then run it $C(n, k) = \frac{n!}{k!(n-k)!}$ times on sub-tables with $n-k$ rows. Due to the fast growth of $C(n, k)$, this exhaustive choice of sub-tables is possible for small values of k only.

The other approach is to run a data-specific number of sub-tables, choosing deleted rows randomly. In both cases, the program will aggregate and reduce the set of rules to approximate a complete and optimal set of rules describing a given table. We mention experiments of this sort with synthetic data later in this section.

A more advanced approach is to use the algorithm in a different mode, when some statistics are computed for new rules generated due to row deletion, and to identify "the blockers", the rows that fail essential rules. See more discussion in section II.

We tested several ranking methods to select the rows should be deleted in order to recover rules lost to noise. One of the tested heuristics is based upon analyzing the inverted table, although the reason for this heuristic's efficacy has not yet been identified.

This direction of research and experiments seems to be worth pursuing, and we plan to conduct more experiments in the future.

Our current experiment was conducted for the table with implication $1, 2, 3, 4 \to 50$ that had support $s = 16$. Columns 5 through 49 were generated randomly with density 0.3. We computed relevances $rel_{50}(a)$ for $a = 1, \dots 49$ on the original table, based on the part of the D-basis with minimal support 5. Values of $rel_{50}(1)$, $rel_{50}(2)$, $rel_{50}(3)$, $rel_{50}(4)$ were the highest in the range of 101–194, while the next highest $rel_{50}(a)$ was 69.5.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

We then imposed 2% noise and recomputed relevances first using the D-basis with min-support=5 only, and then with MRD process that was run $1125 = C(50, 2)$ times, on subtables obtained by removing 2 rows, again using the rules with min-support ≥ 5. Note that there were 226 implications of min support 5 in the D-basis and 368 rules in the set obtained via MRD process.

It is worth noting that the relevances computed on the D-basis did not change much compared to initial table. In particular, the relevances for the first 4 attributes had still the highest rank. With addition of the rules uncovered by MRD, the result was slightly worse: only two out of the first 4 attributes preserved their highest rank among all the attributes.

The next level of the noise $p = 5\%$ brought similar results: the D-basis with min-support ≥ 5 had 177 implications and with MRD about 280 rules. The program computed $rel_{50}(1)$, $rel_{50}(2)$, $rel_{50}(3)$, $rel_{50}(4)$, just on the D-basis, in the range of 46.5–99.8, and three out of these numbers ranked as the first, the second and the fourth. With MRD imposed, three best ranked among attributes 1,2,3,4 were in the first, the fifth and the sixth positions.

At the level of $p = 20\%$, the D-basis produced the rankings of $rel_{50}(1)$, $rel_{50}(2)$, $rel_{50}(3)$, $rel_{50}(4)$ in the range of 65.9–208.3 and three of these values were with the highest rank. For MRD process, based on the estimation $N = npd$, we needed to delete 5 rows at a time, so instead of exhaustive choice we opted for 500 processes, choosing 5 rows randomly. None of the 4 attributes 1,2,3,4 had high rank in the relevance computation.

Finally, at the level $p = 26\%$, we saw only one of $rel_{50}(1)$, $rel_{50}(2)$, $rel_{50}(3)$, $rel_{50}(4)$ as a top ranking, with only D-basis computation, another was at the 5th position, and two others close to 0. With MRD, we ran 1000 random processes removing 5 rows at a time. As before, only the rules with min-support ≥ 5 were included into the computation. The results were comparable with those done with the D-basis only.

This experiments indicates that the MRD process needs finer tuning, and more experimentation should be done with the initial estimation of possible "blockers". Moreover, the massive run of row deletion across all the rows of the table may have an inflation effect on relevance computation for less important attributes.

We would like to highlight the robust effect of the D-basis relevance computation even with the high levels of the noise in this experiment. We also re-computed relevances using the D-basis while changing the level of min-support: to ≥ 8 for the case of 2% noise, and to ≥ 7 for the case of $p = 20\%$ noise. While the absolute values of relevances change, we still obtained highest values for all 4 and 3 out of 4 first attributes, respectively.

VII. Technical implementation

Sequential algorithms are of little practical use when dealing with sufficiently computational complex problems and/or sufficiently large problems [9]. Since the beginning of the demise of Dennard's scaling in the mid 2000's [10], the focus of mainstream computing hardware has moved away from sequential acceleration into parallel acceleration using multiple cores of execution [16]. Since then, algorithms can no longer rely on regular incremental improvements in serial execution speed due to Dennard's scaling. Instead the focus of high performance programming shifted to parallel algorithms. In addition, due to the demands of the age of big data and the decline in custom computing hardware, many of the algorithms in use today must also scale beyond the confines of a single homogeneous computing environment into a distributed heterogeneous execution environment [9].

Finding association rules is a computationally complex problem [1] and our approach is meant to address this issue using a collection of parallelizable algorithms that are scalable in a distributed processing environment.

A. Algorithm Descriptions

At the heart of our new approach for discovering new implications is the parallel execution of a Hypergraph Dualization Algorithm(HDA). Given a set of rows to remove from the original table, our algorithm will run multiple parallel instances of the HDA, dividing the work between all available computer cores.

To facilitate the parallel execution of the HDA we use the Master/Worker parallel computation model [24] running on top of MPI [13], in which a master process orchestrates the parallel execution of the complete job by breaking it into multiple smaller pieces and submitting them to worker processes.

The choice of MPI was made to allow us maximum flexibility and portability in the execution of our algorithms. MPI can scale from a single machine with multiple cores to hundreds of machines with thousands of cores [11] and can run on a variety of computer clusters and hardware configurations.

B. Algorithm Implementations

The Master/Worker model requires two separate processes: a master process and a worker process.

1) Master Process: The master process (seen in algorithm 1) is in-charge of orchestrating the work. It accepts a list of row-groups to remove and the original matrix as parameters. It starts by running the D-Basis algorithm and finding all the implications on the original matrix (line 1). Then, while there are still row-groups left to process, it checks if any idle workers (see worker algorithm 2) are available (line 3) and if so, submits a new job (line 5) with a row-group as a job parameter for the worker.

It then checks if any previously submitted jobs have finished (line 7) and if so calculates the new implications (line 9) by removing any duplicates that were found in the first step (line 1).

The algorithm allows two modes of operation: one aggregates implications and the other does not. In aggregation mode, we collect all the new implications produced by all the row group deletions (line 12) and when aggregation is off, we only report the new implications after each row group deletion (line 10) but we do not aggregate them. In addition, when aggregation mode is on, we reduce the number of final implications following the procedure described in Proposition 17 of [3].

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Algorithm 1 DBasis Master

Require: RCL contains a list of row groups to remove; $OrgMat$ contains original Matrix to work on
1: $allImplications \leftarrow$ FINDDBASIS($OrgMat$) ▷ find D-Basis on original table
2: **while** NOTEMPTY(RCL) **do**
3: **if** HASIDLEWORKERS() **then** ▷ check if we have idle workers that can do some work
4: $rowGroup \leftarrow$ REMOVENEXTGROUP(RCL) ▷ get next group in RCL
5: SUBMITNEWJOB($rowGroup$) ▷ submit row group to parallel processing
6: **end if**
7: **if** ANYFINISHEDJOBSREADY() **then**
8: $newImplications \leftarrow$ GETFINISHEDJOBIMPLICATIONS()
9: $newImplications \leftarrow (newImplications - allImplications)$ ▷ remove duplicates from new implications
10: $Report(newImplications)$
11: **if** $shouldAggregate$ **then**
12: $allImplications \leftarrow allImplications + newImplications$ ▷ aggregate all the implications
13: **end if**
14: **end if**
15: **end while**
16: **if** $shouldAggregate$ **then**
17: $Reduce(allImplications)$ ▷ reduce final inclusive list
18: $Report(allImplications)$ ▷ report final inclusive list
19: **end if**

Algorithm 2 DBasis Worker

Require: $OrgMat$ contains original Matrix to work on
1: **while** MASTERNEEDSME() **do**
2: **if** ANYPENDINGJOBS() **then**
3: $rowGroup \leftarrow$ GETPENDINGJOBROWGROUP()
4: $newImplications \leftarrow$ FINDDBASIS($OrgMat - rowGroup$) ▷ find DBasis on original table - $rowGroup$
5: SUBMITJOBRESULT($newImplications$) ▷ submit row group to parallel processing
6: **end if**
7: **end while**

2) Worker Process: The worker process (seen in algorithm 2) receives the original matrix when it is first spawned. It then waits (line 2) for jobs submitted by the master process (see algorithm 1). If a job is available, it will extract the row-group parameter (line 3) and proceed to call the HDA algorithm (line 4) using the original matrix minus the rows to remove. Once the results are ready, it will submit a reply (line 5) with the new implications to the master process. When the master is done, it will signal the worker to terminate (line 1) and free any system resources it is holding. The number of workers are controlled from the command line on the master node when executing the MPI job.

Algorithm 3 Parallel Reduce

Require: $imps$ list of all implications
1: **procedure** PARALLELREDUCE($imps, resList, startIndex, stepSize$) ▷ parallel reduce entry point
2: **for** $i \leftarrow startIndex; i < sizeOf(imps); i+ = stepSize$ **do** ▷ from startIndex in steps of stepSize
3: **for** $j \leftarrow 0; j < sizeOf(imps); j+ = 1$ **do** ▷ from first to last
4: **if** (Equal(imps[i],impj[j]) Or Cover(imps[i],impj[j]) And NotRemoved(resList[j]) **then**
5: MarkForRemoval(resList[i]) ▷ check if the implication is implied by another and mark for removal if so
6: **end if**
7: **end for**
8: **end for**
9: **end procedure**

3) Parallel Reduction: When running in aggregation mode (see algorithm 1), the last stage of the master process includes reduction, i.e., searching and removing redundant and weaker rules from the final implication list. While conducting experiments we discovered that the computation of this step becomes a bottleneck and increases the overall computation time of our algorithms. To solve this issue, we have devised a parallel reduction algorithm. The parallel reduction algorithm (algorithm 3) allows for multiple hardware threads to search for redundant implications and remove them.

Each instance of the parallel reduction algorithm accepts the original list of implications and which sub list to work on(using the start index and the step size). It then traverses the list of implications (lines 2–3) and tries to independently find if an implication is equal or covered by another stronger implication (line 4). If indeed a stronger implication is found, the current implication is marked for removal. Note that the algorithm makes sure that no implication will be removed twice or that an implication that was already deleted by another parallel thread will be used.

C. Algorithm Performance Analysis

1) Experiment Design: In order to evaluate the performance of the new D-basis parallel algorithm implementation *versus* sequential implementation, we have devised an experiment that involves a typical workload and real-world data from a medical database containing 291 ovarian cancer patients [2].

Each experiment run involves calling D-basis 50 times and each time removing 3 rows with column 81 selected as the implication right hand target and minimum support equal 5. The run results are then aggregated, reduced and reported to the user. We run multiple experiments on different configurations and report the performance results.

The experiments were conducted on a custom made Open-MPI based computer cluster comprised of 4 Virtual Machines(VMs), each with 8 Intel(R) Xeon(R) CPU E5-2660 v2

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

@ 2.20GHz cores and 16GiB of DDR3 RAM, provisioned on Hofstra University's Big Data center; bringing the total number of cores to 32 and total RAM to 64GiB.

2) Experimental Results: We start by using a single VM (8 cores) and running the sequential D-basis algorithm followed by a reduce operation which can either be sequential or parallel.

Figure 2 shows the average run time of the sequential version of D-basis followed by sequential reduce (1) *versus* a sequential D-basis followed by parallel reduce (1+pr).

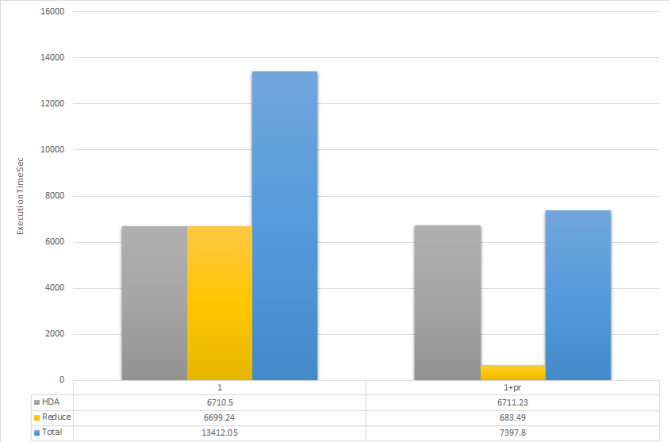

Fig. 2: Sequential vs Parallel Reduce - time in sec of HDA, Reduce and Total run time

On average using parallel reduce improves the performance of the reduce operation (see algorithm 1) by 6.8X. The algorithm is executed on eight parallel hardware threads to achieve this speedup.

Since in this workload the reduction step is almost as expensive as the HDA part, the overall run time is 1.8X faster as well.

3) Scaling Beyond a Single Machine: In the next step we test how the D-basis parallel HDA execution scales from multiple cores on a single machine to multiple machines across the cluster.

Figure 3 shows the average run time of the sequential version of the D-basis algorithm *versus* a version of the algorithm that uses parallel reduce and parallel D-basis.

The graph analyzes three components: HDA time, Reduce time and Total run time on different number of cores (1–32). The X axis specifies the number of cores, from 1 core on the left to 32 cores on the right. Each group of 8 cores represents one VM. When launching a parallel job we specify to MPI how many cores we wish to use and depending on the number of cores available on the VM/CPU it will first try to use a single VM and if it runs out of cores it will spill over to the next VM. in our case since we have 4 VMs each with 8 cores, up to 8 cores would mean a single VM, 8–16 cores would mean 2 VMs etc.

A single run of the HDA algorithm takes an average of 133 seconds to complete. When using the parallel version we can see that the overall time drops as we use more cores. A sequential run will take a minimum of 6650 ($133*50$) seconds to complete on average but as we scale the number of cores we drop the time it takes to run all HDA instances since they are done in parallel. As can be expected the performance improves as we add more cores. The best performance was achieved when we used the maximum number of cores (32). At that point the total time to compute HDA was 22.5X faster.

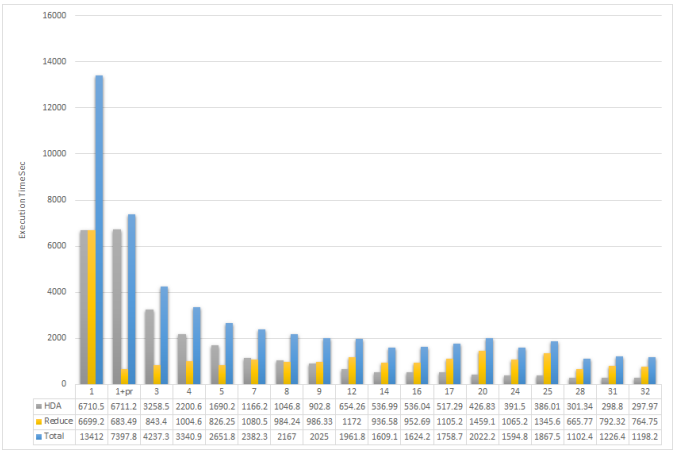

Fig. 3: Sequential vs Parallel Results- time in sec of HDA, Reduce and Total run time

Parallel reduce time fluctuates across experiments between 665.77 and 1459.14 seconds but on average is 6.8X faster than the sequential reduce version.

The total run time fluctuates as well but as can be seen, it scales with the number of cores and the best performance achieved was 12.2X speedup when we used 28 cores running across 4 VMs on the MPI cluster. The top three results, as expected, were achieved when we used the most number of cores (28–32).

D. Comparison to Sequential Performance Results

In previous work [21] we reported the runtime performance of our sequential implementation. We now compare those results with the performance of the new parallel algorithm versions.

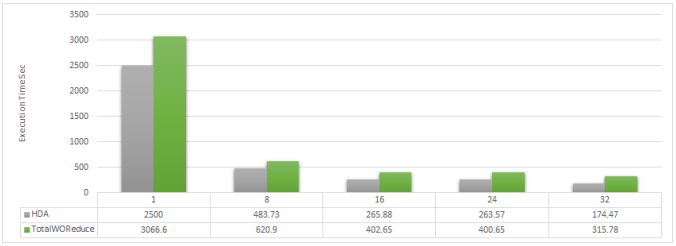

Fig. 4: Old Sequential vs New Parallel Medical Results- time in sec of HDA, and total run time without reduce operation

Figure 4 shows the runtime comparison of the old sequential algorithm *versus* the new parallel algorithm running on the same medical database containing 291 ovarian cancer patients [2] but with a different set of parameters, replicating the original experiment in [21]. The experiment involved calling D-basis 25 times and each time removing 20 rows with

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

column 81 selected as the target and minimum support equal 5. Also note that during the original experiment the reduction step was not used and so we report the same here. As can be seen the total run time without reduce scales with number of cores and reaches a peak of 9.7X speedup when using 32 cores. Total HDA run time scales as well and reaches a peak of 14.3X speedup at 32 cores.

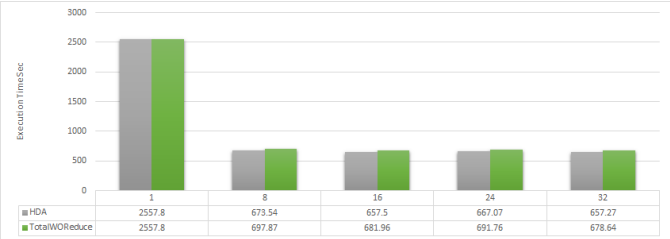

Fig. 5: Old Sequential vs New Parallel Retail Results- time in sec of HDA, and total run time without reduce operation

Figure 5 shows the runtime comparison of the old sequential algorithm *versus* the new parallel algorithm running on the Frequent Itemset Mining Dataset Repository [26], a retail market basket data from an anonymous Belgian retail store with a set of parameters replicating the original experiment in [21] which involved calling D-basis 200 times and each time removing 3 rows with no column selected and minimum support equal 3. Here again, during the original experiment, the reduction step was not used and we do the same. In this workload (type of data), since the matrix has low density (0.0162) the HDA algorithm takes a fraction of a second to execute and so the communication overhead makes it difficult to scale well across multiple cores and to gain a significant advantage over using sequential HDA. Close to maximum performance gain is achieved on a relatively low number of cores and the maximum performance gain is 3.9X. The total run time speedup without reduce is reached at 32 cores but with a mere 3.8X speedup and with very little improvement over using 8 cores, for example.

VIII. Conclusion and Future Work

In this paper we have discussed the development of an algorithm for the retrieval of association rules in a binary table i.e., a table consisting of ones and zeroes as another representation of the data. This is done via dualizing the hypergraph associated with the dataset, then reducing the task of rule generation to traversing this associated hypergraph via a sub-exponential time-complexity algorithm [1].

Several development proceedings were discussed, including the parallelization of the program and a *top-down* method of retrieving rules which hold in all rows of the table except a few deleted rows. Analyzing a slightly smaller sub-table allows to discover rules which have high confidence and may fail in a few rows due to noise in the data. The ability for this code to be parallelized and its low theoretical time complexity make it a powerful tool for data mining.

The *relevance* parameter for determining the importance of one attribute to an outcome was also tested to see how it might be used in analyzing real data. It was seen that even random rules could produce a notable relevance values, summarized in Figure 1. Then, a synthetic rule was constructed and noise was added to the table in order to see what effect noise would have on the relevance of the parameters of the constructed rule, and whether or not the process of retrieving rules of high confidence via row deletion could recover high relevances of important parameters for an outcome. The tests revealed that relevance stayed relatively stable up until approximately 20% noise, when it was computed using the D-basis, and some further tuning of MRD process is required to improve these results. For this, additional analysis of possible "blockers" should be further explored before applying MRD process. This will be a prominent direction of further analysis and experiments.

The further development of the D-basis implementation is underway, with the goal to use parallel computation of minimal hitting sets in algorithm [18], which are now available in public domains. We currently have several data sets in biology, medicine and meteorology which we plan to explore, working in collaboration with Biology Department, Geology, Environment and Sustainability Department of Hofstra University, as well as Donald and Barbara Zucker School of Hofstra-Northwell. We also plan to continue the collaboration with the Cancer Center of University of Hawai'i, and contribute to the exploration of data sets of various cancers, which combines several available methods [19].

Acknowledgements. We used for testing the data set on ovarian cancer available at [25], which was also used in [2] and managed in the lab of Dr. Gordon Okimoto, at University of Hawai'i Cancer Center. We thank the support of Computer Science Department of Hofstra University for providing the virtual machine environment for this project. The second author was supported by Honor's College of Hofstra University's Research Assistant Grant and the Hofstra Advanced Summer Program in Research (ASPiRe).

References

[1] K. Adaricheva, and J.B. Nation, *Discovery of the D-basis in binary tables based on hypergraph dualization*, v.658 (2017), Theoretical Computer Science, Part B, 307–315.

[2] K. Adaricheva, J.B. Nation, G. Okimoto, V. Adarichev, A. Amanbekkyzy, S. Sarkar, A. Sailanbayev, N. Seidalin, and K. Alibek, *Measuring the Implications of the D-basis in Analysis of Data in Biomedical Studies*, Proceedings of ICFCA-15, Nerja, Spain; Springer, 2015, 39–57.

[3] K. Adaricheva, J.B. Nation and R. Rand, *Ordered direct implicational basis of a finite closure system*, Disc. Appl. Math. **161** (2013), 707–723.

[4] K.Adaricheva and T. Ninesling, *Direct and binary-direct bases for one-set updates of a closure system*, manuscript; presented in poster session of ICFCA-2018 http://icfca2017.irisa.fr/program/accepted-papers/

[5] Adaricheva K., Turan G., Sloan R. and Szorenyi B., Horn belief contraction: remainders, envelopes and computational aspect, in Proceedings of KR-2012, Italy, 107–115.

[6] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, *Fast discovery of association rules*, Advances in Knowledge discovery and data mining, AAAI Press, Menlo Park, California (1996), 307–328.

[7] Guillaume Bosc, Marc Plantevit, Jean-Franois Boulicaut, Moustafa Bensafi, and Mehdi Kaytoue, h(odor): Interactive discovery of hypotheses on the structure-odor relationship in neuroscience, in ECML/PKDD 2016 (Demo), 2016.

[8] J.L. Balcázar, *Redundancy, deduction schemes, and minimum-size bases for association rules*, Log. Meth. Comput. Sci. **6** (2010), 2:3, 1–33.

[9] Gordon Bell, Jim Gray, and Alex Szalay. Petascale computational systems. *Computer*, 39(1):110–112, 2006.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

[10] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Communications Surveys & Tutorials*, 13(2):223–244, 2011.

[11] George Bosilca, Thomas Herault, Ala Rezmerita, and Jack Dongarra. On scalability for mpi runtime systems. In *2011 IEEE International Conference on Cluster Computing*, pages 187–195. IEEE, 2011.

[12] M. Fredman and L. Khachiyan, *On the complexity of dualization of monotone disjunctive normal forms*, J. Algorithms **21** (1996), 618–628.

[13] Edgar Gabriel, Graham E Fagg, George Bosilca, Thara Angskun, Jack J Dongarra, Jeffrey M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, et al. Open mpi: Goals, concept, and design of a next generation mpi implementation. In *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*, pages 97–104. Springer, 2004.

[14] B. Ganter, and R.Wille, Formal concept Analysis: Mathematical Foundations, Springer, 1999.

[15] E.L. Kaplan and P. Meier, *Nonparametric estimation from incomplete observations*, J. Amer. Statist. Assn. **53** N282 (1958), 457–481.

[16] Jonathan Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing*, 33(3):46–54, 2011.

[17] M. Kryszkiewicz, *Concise representation of association rules*, Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery, Springer-Verlag, London, UK, 92–109.

[18] K. Murakami and T. Uno, *Efficient algorithms for dualizing large scale hypergraphs*, Disc. Appl. Math. **170** (2014), 83–94.

[19] J. B. Nation, G. Okimoto, T. Wenska, A. Achari, J. Maligro, T. Yoshioka, and E. Zitello, A Comparative analysis of MRNA expression for sixteen different cancers, preprint, $http$: $//www.math.hawaii.edu\ jb/lust\_2017\_615.pdf$

[20] D. Prajapati, S. Garg and N.C.Chanhan *Interesting association rule mining with consistent and inconsistent rule detection from big sales data in distributed environment*, Future Computing and Informatics Journal **2** (2017), 19–30.

[21] O. Segal, J. Cabot-Miller, K. Adaricheva, J.B. Nation, A. Sharafudinov, The Bases of Association Rules of High Confidence, Proceedings of SIGPRO - 2018, Sydney, David C.Wyld et al. (Eds) pp. 39-51, 2018.

[22] Arnaud Soulet and Bruno Crmilleux, *Mining constraint-based patterns using automatic relaxation*, Intell. Data Anal., 13(1):109–133, 2009.

[23] Arnaud Soulet, Chedy Rassi, Marc Plantevit, and Bruno Cremilleux,*Mining dominant patterns in the sky* In IEEE 11th Int. Conf on Data Mining (ICDM 2011), pages 655–664. IEEE, 2011.

[24] T. Rauber and G. Rünger. *Parallel Programming: for Multicore and Cluster Systems*. Springer Berlin Heidelberg, 2013.

[25] The Cancer Genome Atlas Research Network, *The Cancer Genome Atlas Pan-Cancer analysis project*, Nature Genetics **45** (2013), 1113–1120.

[26] Frequent Itemset Mining Dataset Repository, publicly available at http:// fimi. ua. ac.be /data/retail.dat

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731