This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

PRESQ: Discovery of Multidimensional Equally-Distributed Dependencies via Quasi-Cliques on Hypergraphs

Alejandro Álvarez-Ayllón, Manuel Palomo-Duarte, Juan-Manuel Dodero

Abstract—Cross-matching data stored on separate files is an everyday activity in the scientific domain. However sometimes the relation between attributes may not be obvious. The discovery of foreign keys on relational databases is a similar problem. Thus techniques devised for this problem can be adapted. Nonetheless, when the data is numeric and subject to uncertainty, this adaptation is not trivial. This paper firstly introduces the concept of *Equally-Distributed Dependencies*, which is similar to the *Inclusion Dependencies* from the relational domain. We describe a correspondence in order to bridge existing ideas. We then propose PRESQ: a new algorithm based on the search of maximal quasi-cliques on hyper-graphs to make it more robust to the nature of uncertain numerical data. This algorithm has been tested on seven public datasets, showing promising results both in its capacity to find multidimensional equally-distributed sets of attributes and in run-time.

Index Terms—Data mapping, Probabilistic algorithms, Hypergraphs, Inclusion Dependencies, Equally-Distributed Dependencies, Quasi-clique

1 INTRODUCTION

Nowadays, it is not uncommon for many types of users — from proficient data scientists to enthusiasts without formal training— to dive into overwhelming sets of data looking for any relevant pattern they can find. This data may consist of raw files that have not yet been ingested into a database system and for which the schema may be unfamiliar and not adequately documented. Furthermore, the entire data set may be composed of multiple files with heterogeneous schemes [1], [2].

For the *in-situ* interactive exploration, there are many proposals at different levels: database (indexes, physical layout), middleware (pre-fetching, query approximation) and user interface (visualization, assisted exploration) [3]. For more details, including a survey of existing solutions, we refer the reader to an exhaustive systematic mapping of the literature previously published [4]. As a result of this survey, we realized that most of the solutions treat files separately, leaving it to the end-user to work out how they are related. This is an observation shared by other authors [5].

Therefore, our goal is to assist users to understand how multiple raw files are related; to identify shared sets of attributes, and to facilitate relationship-based mining between different files with heterogeneous schemes. To illustrate this, we show three possible scenarios for this kind of exploration of associations. These are focused mainly on astronomy but they can be extrapolated to other areas [6]:

- *A. Álvarez-Ayllón works in the Department of Astronomy, University of Geneva — Chemin d'Ecogia 16, 1290 Versoix, Switzerland*
  E-mail: alejandro.alvarezayllon@unige.ch
- *M. Palomo-Duarte and J.M. Dodero are with the Department of Computer Science and Engineering, University of Cadiz — 10 Avenida de la Universidad, 11519 Puerto Real, Spain*

- *Spatial*: Identify objects in the same location.
- *Temporal*: Identify events occurring within the same time period.
- *Coincidence*: In general, apply clustering techniques to identify objects that are co-located within a multidimensional space.

REDISCOVER [1] is an example of a proposed solution aimed in this direction. It is based on machine learning techniques, such as Support Vector Machines, to identify matching columns between scientific tabular data. Yet, this system focuses mainly on the correspondence between individual columns. This is insufficient for spatial and coincidence associations, as they are multidimensional.

Our general research question is: Can we use the actual *data* to automatically guide the user to cross-match different files or to use them together as a single source, taking multidimensionality into account? To bridge this gap, we propose the concept of *Equally-Distributed Dependencies* (EDDs), which is inspired by the idea of *Inclusion Dependencies* (INDs) from the relational algebra:

> *An inclusion dependency between column A of relation R and column B of relation S, written $R.A \subseteq S.B$, or $A \subseteq B$ when the relations are clear from the context, asserts that each value of A appears in B. Similarly, for two sets of columns X and Y , we write $R.X \subseteq S.Y$, or $X \subseteq Y$, when each distinct combination of values in X appears in Y [7]*

The definition of IND is based on set theory, which is not directly applicable to numeric data where measures are in the real domain (e.g. spatial coordinates) and usually have an associated uncertainty that may or may not be explicitly stored.

However, this definition can be naturally reformulated

in terms of equality of distribution $X \stackrel{d}{=} Y : F_X(x) = F_Y(x)\ \forall x$, where F_X and F_Y are the cumulative distribution functions of X and Y, respectively:

> *An equally-distributed dependency between a set of columns X from of relation R and a set of columns Y of relation S, written $R.X \stackrel{d}{=} S.Y$ or $X \stackrel{d}{=} Y$, asserts that the values of X and Y follow the same probability distribution.*

The term *arity* refers to the cardinality of the sets of attributes X and Y. For instance, if $|X| = 1$, we talk about unary EDDs; if $|X| = 2$, binary or 2-EDDs; and, in general, for $|X| = n$, n-ary EDDs.

Contribution: This paper develops the basis for equally-distributed dependencies and proposes a statistically robust algorithm for finding them. Different experiments show that our proposal successfully finds dependencies in a reasonable amount of time. In addition, it shows how different parametrizations balance performance (run-time), efficacy (capability of finding high-arity EDDs), and efficiency (avoidance of redundant results).

Paper organization: In section 2, we briefly discuss existing work done on IND discovery. In section 3, we introduce the background for our research. In section 4, we propose a novel algorithm based on quasi-cliques to infer common equally-distributed multidimensional attributes. In section 5, we show experimental results, and in section 6, we discuss our findings. We list the threats to the validity of this study in section 7. Finally, in section 8, we compile the conclusions and propose areas for further work.

2 Related Works

Finding high arity INDs is a NP-hard problem [8]. For instance, for two sets of n attributes in R and S, there are $n!$ different possible permutations to check. In comparison, finding unary INDs seems a relatively simple problem, as the worst case has complexity $O(n^2)$. Nonetheless, testing over real files may require expensive input/output operations. Furthermore, as we will see later, false positives at this stage can quickly make finding high arity INDs unfeasible. This is because the search space tends to grow exponentially with the number of one-attribute matches, making unary INDs search time much less important than reducing the number of false positives.

We used a published experimental evaluation [9] as a starting point for assessing how adequate existing solutions are for our problem. The authors carried out a set of experiments with thirteen IND algorithms, of which seven are for unary INDs, four for n-ary INDs, and two for both types. A more recent survey confirms that this work contains the current state-of-the-art for Inclusion Dependencies [10].

We describe the unary problem and propose our algorithm, tailored to numeric data, in section 4.1. We will now describe briefly the n-ary finding algorithms evaluated by the authors and discuss their suitability for our needs.

2.1 n-INDs finding algorithms

Given two relations R and S, with attributes A and B respectively, a unary Inclusion Dependency (uIND) exists if $R.A \subseteq S.B$. More generally, for two sets of attributes X and Y, both of cardinality n, an n-ary Inclusion Dependency (nIND) exists if every combination of values in X appears in Y [7], [11].

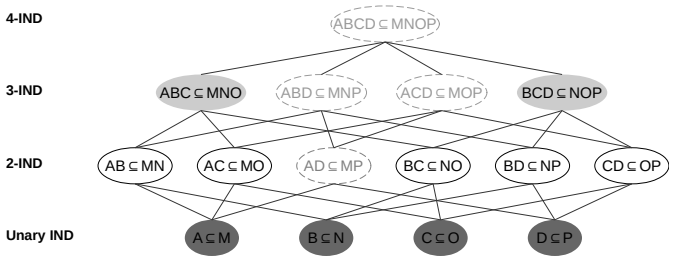

Fig. 1. Example structure of the search space as a lattice for an initial set of 4 unary INDs. As an illustration, if the 2-INDs surrounded by a solid line were valid, a bottom-up traversal would only need checking the validity of the 3-INDs with a grey background since the others could not be valid.

Given a set U of valid uINDs, the search space for higher-arity candidates is defined by its power set and a partial order relation called specialization [11]:

Definition 1. *Let $I_1 = R[X] \subseteq S[Y]$ and $I_2 = R'[X'] \subseteq S'[Y']$. I_1 **specializes** I_2 (denoted $I_1 \prec I_2$) iff*

1) *$R = R'$ and $S = S'$.*
2) *X and Y are sub-sequences of X' and Y', respectively.*

*Equivalently, we can also say that I_2 **generalizes** I_1.*

Example 1. *$(R[AB] \subseteq S[EF]) \prec (R[ABC] \subseteq S[EFG])$. However, $R[AB] \subseteq S[DE]) \not\prec (R[ACD] \subseteq S[DFG])$*

This partial order enables us to structure the search space as a lattice, as exemplified in figure 1. Most solutions leverage this property to explore the search space bottom-up —from level k to k+1— or top-down —from level k to k–1— order.

MIND [11] is a bottom-up approach: it starts from a set of known, satisfied unary INDs and builds higher arity candidates combining them. These new candidates are then validated against the database and those satisfied are used for computing the next level candidates until no more candidates are available.

ZIGZAG [12] starts with a MIND bottom-up approach up to a given arity $n \geq 2$. Then, it uses all satisfied INDs to initialize a *positive border* and the non-satisfied to initialize a *negative border*. The set of satisfied INDs is used to generate the set of candidates with the highest arity possible, called *optimistic border*, which is then validated against the database. This is the bottom-up part of the search. Valid candidates are directly added to the positive border. Invalid candidates are treated depending on how many tuples are different between relations. Those above a given threshold (too many different tuples) are added to the negative border. Those below are top-down traversed, from level n to n–1, validated, and then added to the positive border if they are satisfied. The algorithm then iterates, building a new optimistic border until it is not possible to generate new INDs. The optimistic approach can prune the search space very aggressively when there are high-arity INDs, but when most arities are low, MIND may perform better.

FIND2 [13] is based on the equivalence between finding n-INDs and finding cliques on n-uniform hypergraphs

(a generalization of the concept of a graph where each edge connects n nodes). Each unary IND corresponds to a node, and an n-IND corresponds to an edge on an n-uniform hypergraph. Once such a graph is built, each IND corresponds to a clique and maximal INDs correspond to maximal cliques. They present the HYPERCLIQUE algorithm, capable of finding maximal cliques performantly, which can be mapped back to candidate maximal INDs. These are finally validated using database queries. As ZIGZAG, FIND2 starts with a bottom-up approach to look for maximal cliques (i.e., potential maximal INDs). The invalid ones are used to generate a new (n+1) uniform graph. This is a stage that corresponds to the top-down traversal.

While these three algorithms were evaluated on INDs between relational datasets and with attributes that can be directly compared (i.e. from discrete domains), their traversal of the search space and their validation steps are well decoupled. They can be easily be adapted to the equality-of-distribution statistical tests.

Furthermore, the reference benchmark shows that MIND, FIND2 and ZIGZAG have a comparable run-time, sometimes even faster, than the alternatives. While FAIDA is generally faster, its validation strategy requires computing hashes over the attributes and their combinations, which is inapplicable for continuous data that can very possibly have an associated uncertainty.

From the three suitable candidates, MIND's bottom-up approach can be performant enough for relatively low arity IND relations. However, it has one substantial disadvantage: it requires an exponential number of tests, prohibitive for higher arity INDs. Both ZIGZAG and FIND2 overcome this limitation by alternating between *optimistic* (top-down) and *pessimistic* (bottom-up) traversals. Finally, FIND2 maps the search of INDs to the search of maximal cliques. We know that using statistical tests will introduce unavoidable false negatives, which would translate into missing edges. A clique with missing edges is a quasi-clique, and finding quasi-cliques, while at least as hard as finding cliques, is doable. This has influenced our approach.

2.2 Foreign Key Discovery

Foreign key discovery is outside the scope of this paper. We briefly survey this area, however, since we consider it complementary to IND discovery. A foreign key (FK) constrain on an attribute A over a primary key (PK) B implies that all values present on A must also be present on B. Therefore, there exists an inclusion dependency between A and B. However, the reverse is not necessarily true. For instance, two auto-increment attributes from two different relations may have an accidental Inclusion Dependency with no semantic meaning.

To distinguish between accidental and meaningful INDs, Rostin *et al*. [14] propose to train machine learning models over a set of features extracted from positive PK/FK relations and negative, non-meaningful INDs. However, their proposal is limited to unary INDs.

Zhang *et al*. [15] present an algorithm capable of handling multi-column PK/FK relations. They define the concept of *Randomness Test*, which assumes that an FK is a representative sample of a PK and, therefore, should follow a similar distribution. They use an approximation of the Earth-Mover Distance (EMD) —the cost of transforming one distribution into another— to measure the similarity between PK and FK. Their algorithm ranks PK/FK candidates by distance —closest first— and selects the top $X\%$, where X must be chosen to balance precision and recall.

More recently, Jian *et al*. [16] introduced an approach that identifies both PK and FK holistically. They validate Zhang's concept of *Randomness* and propose a simplified estimator that treats each attribute separately. They do not need the PKs to be known but require a list of INDs as input.

It is worth noting that even though the latter two publications use the idea of the FK being a random sample of the PK, their methods use the *distance* between distributions for ranking candidates [15] or as a feature [16]. Our method is based, however, on statistical hypothesis testing[1].

In the next section, we will provide the necessary background for describing our proposal, described in section 4.

3 Background

3.1 Equally Distributed Dependencies

An Inclusion Dependency exists if all combinations of values from a given set of attributes in one relation are contained within a set of attributes from another. However, in the real domain, we will hardly ever find a strict subset relation between two attributes. Measurements may have associated uncertainty, and even floating-point representation may vary (i.e. 32 vs 64 bits). In general, it is a flawed idea to compare floating-point numbers with strict equality.

Instead, we can use $R.X \stackrel{d}{=} S.Y$ as an approximation, meaning that the two sets of attributes are equally distributed. This relation is, unlike the subset relation, symmetrical.

Following the parallelism with IND finding, we say that the dataset d satisfies the relation defined by equality of distribution $\stackrel{d}{=}$ when a statistical test fails to reject the null hypothesis

$$H_0 : P(R[X]) = P(S[Y]) \tag{1}$$

Three inference rules can be used to derive some additional INDs from an already known set of INDs. They are defined using sets and subsets [18], but they translate to the equality of distribution:

Reflexivity

$R[X] \stackrel{d}{=} R[X]$

Permutation and projection

If $R[A_1, \ldots, A_n] \stackrel{d}{=} S[B_1, \ldots, B_n]$ then $R[A_{i_1}, \ldots, A_{i_m}] \stackrel{d}{=} S[B_{i_1}, \ldots, B_{i_m}]$ for each sequence $i_1, \ldots, i_m$ of distinct integers from $\{1, \ldots, n\}$

Transitivity

$R[X] \stackrel{d}{=} S[Y] \wedge S[Y] \stackrel{d}{=} T[Z] \implies R[X] \stackrel{d}{=} T[Z]$

The reflexivity, permutation and transitivity rules are well known to hold for $\stackrel{d}{=}$ [19]. We have proven that the projection rule also holds, as is logical [20].

1. While the EMD could be used as a test statistic, it would be computationally expensive [17].

Thanks to the validity of these rules, particularly the permutation and projection, we can use the specialization relation seen in definition 1 when dealing with distributions.

With these rules we have defined the search space similar to the one from IND discovery. The last requirement is a property that allows the pruning of the search space as illustrated in figure 1.

Let $I = R[X] \overset{d}{=} S[Y]$. A dataset d *satisfies* I iff a statistical test fails to reject $H_0 : P(R[X]) = P(S[Y])$ given a significance level α. This is denoted as $d \models I$.

Property 1. *Given* $I_1 \prec I_2$*:*

1) *If* $d \models I_2$*, then* $d \models I_1$ *(Accepting* H_{0_2} *implies accepting* H_{0_1}[2]*)*
2) $d \not\models I_1$ *with a probability* α *when* $d \models I_2$ *(Rejecting* H_{0_1} does not *imply the rejection of* H_{0_2}*)*

This property is similar to that proposed for INDs [11], with the exception that even if $d \models I_2$, there is a probability to falsely reject I_1 bound by the significance level α.

Example 2. *If we have two sets of* 10 *attributes that are equally distributed, the number of 3-dimensional projections (specializations) that must be equally distributed will be* $\binom{10}{3} = 120$ *. If we have a significance level of* $\alpha = 0.1$*, the expected number of falsely rejected 3-dimensional equalities is then* 12*.*

3.2 Uniform n-Hypergraphs and quasi-cliques

A hypergraph is a generalization of a graph where the edges may connect any number of nodes. It is defined as a pair $H = (V, E)$, with V the set of nodes and E the set of edges. An edge $e \in E$ is a set of distinct elements from V.

Definition 2. *Given the hypergraph* $H = (V, E)$*,* H *is a* n-hypergraph *iff all of its edges have size* n*.*

A clique or hyper-clique on a n-hypergraph $H = (V, E)$ is a set of nodes $V' \subseteq V$ such that every edge defined by the permutations of distinct n nodes from V' exists in E [13].

A quasi-clique or hyper-quasiclique (sometimes named pseudo-clique) is a generalization of a clique where a given number of edges can be missing. The exact definition can be based on the ratio of missing edges or based on the node degrees. Another option is to combine both measures [21], which is our preferred method.

We generalize the definition of quasi-cliques to k-uniform hypergraphs :

Definition 3. *Given a k-uniform hypergraph* (V, E)*, and two parameters* $\lambda, \gamma \in [0, 1]$ *the sub-graph* $H' = (V', E')$ *induced by a subset* $V' \subseteq V$ *is a* $(\lambda - \gamma)$ *quasi-clique iff:*

$$|E'| \geq \gamma \cdot \binom{|V'|}{k} \tag{2}$$

$$\forall v \in V' : deg_{V'}(v) \geq \lambda \cdot \binom{|V'| - 1}{k - 1} \tag{3}$$

Where $deg_{V'}(v)$ *represents the degree of* v*, and* E' *is a subset of* E *such that* $\forall e \in E' : e \subseteq V'$

2. Strictly speaking, *not rejecting* H_{0_2} implies that we can not reject H_{0_1}.

In other words, condition 2 allows for some edges to be missing, while condition 3 enforces a lower bound on the degree of each node. Intuitively, the latter is essential to avoid quasi-cliques where most nodes are densely connected and a handful of nodes are connected only to a few.

The hyper-clique problem is a particular case when either $\lambda = 1$ or $\gamma = 1$.

4 Inferring Common Multidimensional Data

The first required step to find multidimensional EDDs is to find a set of unary EDDs, for which a naive approach would mean quadratic complexity. To reduce the complexity, we propose an algorithm based on interval trees in section 4.1. In section 4.2, we discuss the difficulties of the existing adaptable algorithms when dealing with uncertainties. Finally, in section 4.3, we propose a novel algorithm, based on quasi-cliques, which is more resilient to both false positives and false negatives.

4.1 Uni-dimensional EDDs

The first required step for any of the three algorithms is to find a set of valid unary EDDs on the datasets. i.e., attribute pairs that follow the same distribution. It can be done with the non-parametric Kolmogorov-Smirnov (KS) two-sample test [22]. More formally, for a possible pair of attributes A and B from two different relations, the null hypothesis H_0 for the KS test is $A \overset{d}{=} B$. As for any statistical test, this null hypothesis is accepted or rejected with a significance level $\alpha \in [0, 1]$, which is the probability of falsely rejecting H_0 (false negative).

Consider a dataset containing the relations $R_1, R_2, \ldots, R_n$ with a total number of attributes $N = \sum_{i=1}^{n} |R_i|$. A naive approach to finding unary EDDs requires $N - 1$ statistical tests for each attribute. Since the EDD relation is symmetric ($A \overset{d}{=} B \iff B \overset{d}{=} A$) half of the tests can be avoided, bounding the total number of tests by the quadratic expression $(N \times (N - 1))/2$.

We propose using an interval tree built over the complete dataset to reduce the number of tests. The building of the tree can be performed in $O(N \log(N))$ time, and each query done in $O(\log(N) + m)$, where m is the number of overlapping intervals for a given attribute. When $m \ll N$, which we expect to be generally the case, the number of operations can be thus reduced to $O(N \log(N) + M)$, where M is the total number of overlapping pair of attributes. Note that $M \leq (N \times (N - 1))/2$, so the worst-case remains quadratic.

However, the cost of the tests themselves is almost negligible when compared to the cost of finding n-ary EDDs, which is exponential with the number of unary EDDs. Therefore, a low significance level α for finding unary EDDs will considerably increase the cost at later stages.

4.2 Multidimensional EDDs

Once we have a set of unary matches, we need to find which, if any, higher dimensional sets of attributes are shared between each pair of relations. As discussed in section 2, only three of the existing IND finding solutions are not

strongly dependent on discrete types: MIND, ZIGZAG and FIND2. However, replacing the *inclusion* tests with statistical tests affects their behavior.

MIND traverses the search space bottom-up. Thus, for two relations with a single multidimensional EDD with n attributes, every combination of k nodes from $k = 2$ to $k = n$ must be tested, as shown in equation 4.

$$\sum_{k=0}^{n} \binom{n}{k} = \sum_{k=0}^{n} \frac{n!}{n!(n-k)!} \tag{4}$$

Since statistical tests are not exact, the chances of having at least one false rejection in the validation chain increases with the maximal EDD arity, introducing discontinuities in the search space. This makes its traversal more difficult.

The search algorithm of FIND2 is capable of finding maximal INDs with fewer tests. As an input, it requires a set of valid unary and n-ary IND relations. A k-uniform hypergraph $G(V, E)$ is then constructed, where the set of accepted unary INDs are mapped to the set of vertices V, and the set of accepted k-INDs to the set of edges E. Given this initial hypergraph, the authors of FIND2 prove that finding higher arity INDs can be mapped to the problem of finding cliques, since all the generalized k-ary INDs *must* appear as edges.

However, this is not always true for the EDD finding problem. The statistical test will yield some false positives and some false negatives, a combination that makes it difficult for FIND2 to find the true relations. Cliques will likely be *broken* due to the false rejections, and there will be spurious edges due to false positives. Higher arity EDDs do not appear as cliques in this scenario.

Finally, ZIGZAG can not recover well from missing EDDs. Any rejected EDD is added to the negative border and will not be considered any further. Additionally, some early experiments with ZIGZAG indicated that the combination of false positives and false negatives makes the algorithm run close to its worst-case complexity (factorial).

4.3 PRESQ algorithm

As we have discussed, EDD finding does not map well to the clique-finding problem due to missing and spurious edges caused by the statistical tests. We propose instead an algorithm based on quasi-cliques as described in definition 3. This approach is better suited to the uncertainties associated to hypothesis testing.

Finding quasi-cliques *seeds*: Some initial experiments with FIND2 showed that, by only modifying the validation strategy to use statistical tests, the algorithm was able to find relatively high arity EDDs regardless of the missing edges.

The modified FIND2 maps the initial set of EDDs into a graph and lets the HYPERCLIQUE algorithm find the set of maximal cliques, then maps them back to EDDs and validates the inferred EDDs. Therefore, if FIND2 finds high arity EDDs is because HYPERCLIQUE finds maximal cliques close to the maximal quasi-clique. This makes sense since, generally, a quasi-clique contains smaller but denser subgraphs [23] and a clique is denser than a quasi-clique.

Therefore, we use a modified version of HYPERCLIQUE to search for quasi-clique *seeds*, accepting a candidate if it is a quasi-clique, as per the joint definition of equations 2 and 3. We combine both definitions since limiting only the number of missing edges tends to accept quasi-cliques with too many vertices.

Growing the quasi-clique *seeds*: This is similar to KERNELQC's idea [23], but based on a quasi-clique enumeration algorithm. Given a quasi-clique *seed* from the first stage, candidates are *grown* following a tree-shaped, depth-first traversal [24].

Let v be a node on a graph $G[V]$ with a degree lower or equal to the average degree. The density (i.e. γ) of $G[V \backslash v]$ is no less than the density of $G[V]$. In other words, if we remove from a γ-quasi-clique a node v with a degree lower than the average degree, the resulting graph is still *at least* a γ-quasi-clique. This is consistent with the observation that a quasi-clique contains denser sub-graphs [23].

Consequently, removing the vertex with the lowest degree means that the resulting quasi-clique is still a γ-quasi-clique. In the case of a tie, we can choose the vertex by its index (or name). This node is named $v^*(V)$.

Finally, a quasi-clique K' is considered a child of another quasi-clique K if and only if $K' \backslash K = v^*(K')$, i.e a quasi-clique K' is a child of K if it has one additional node that is the first node when sorted in ascending order by degree and index. This defines a strict parent-to-child relationship between quasi-cliques, which can be modelled and traversed like a tree.

The original algorithm [24] is exclusively oriented towards γ-quasi-cliques, and this traversal would include many candidates that are not λ-quasi-cliques. To prune the search space and avoid branches that will not yield any valid quasi-clique, at each recursion step, we compute the degree that the nodes on K' should have, so that K' is a λ-quasi-clique. When adding a node, the expected minimum degree may increase. By knowing this value, we can ignore all nodes with a degree lower than the threshold *in the input graph*, as no matter how many more nodes we were to add afterwards, no child candidate would satisfy the λ threshold.

This step successfully increases the number of quasi-cliques found. However, the number of maximal cliques is bound in general by an exponential expression of the form $\Omega(a^{|V|/b})$, where a, b are two constants that depend on the rank of the hypergraph [25]. Since cliques are a particular case of quasi-cliques, we can expect the lower bound for the maximum number of quasi-cliques also to be exponential. Even if enumerating quasi-cliques can be done in polynomial time *per quasi-clique* [24], the total runtime has a worst-case exponential complexity for dense hypergraphs. Therefore, it would be advisable to disable this stage for datasets with attributes hard to differentiate at low dimensionality or restrict it to the top-k seeds found.

4.3.1 Parameters

Before explaining how to tailor the parameterization of the quasi-clique finding for the purpose of searching EDDs, we need to remind that, given two sets of attributes $R[X]$ and $S[Y]$, our algorithm builds on the null hypothesis $H_0 : P(R[X]) = P(S[Y])$. In other words, it is based on the *assumption* that any EDD candidate is valid.

Let α be the significance level chosen by the user before running the algorithm. Let G be the initial k-uniform

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

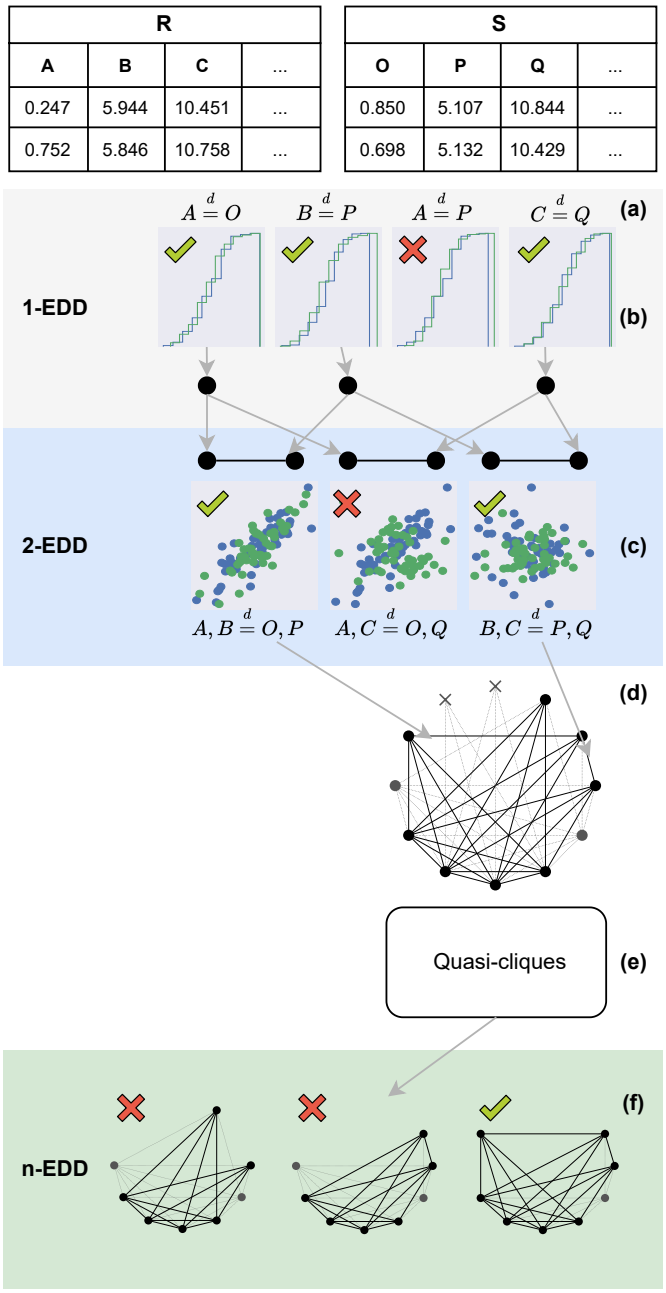

Fig. 2. Simplified schematic of the algorithm. Given datasets R and S, (a) Candidate 1-EDDs are found applying the interval-tree as described in section 4.1.
(b) Those for which the Kolmogorov-Smirnov finds a significant difference are discarded, and the rest are mapped to nodes.
(c) All pairwise combinations are tested, and those equally distributed are (d) mapped to edges on a 2-hypergraph. The algorithm works with hypergraphs of any rank (e.g., triplets mapped to edges on a 3-hypergraph).
(e) PRESQ searches for quasi-cliques as described in section 4.3.
(f) A quasi-clique of cardinality n corresponds to an n-EDD, which is then validated by a statistical test. Those rejected are decomposed to generate the edges for a 3-hypergraph, which are verified (c), used to build a 3-hypergraph (d) and finally passed as input back to (e).
The graph above displays spurious nodes and edges (light grey, dotted) and false negatives (missing edges between dark nodes) based on attribute names. The full graph is not a valid quasi-clique because the hypergeometric test on the node degree (eq. 6) prunes the two nodes shown with crosses. Three candidates of arity 8 are generated given the constrain on the number of edges (eq. 2). Two of them are rejected by the n-dimensional statistical test and used to compute the edges of the 3-hypergraph.

hypergraph and let K be a quasi-clique candidate. Under H_0, K represents a $|K|$-ary EDD, and by the projection rule, all possible edges between the nodes in K are also valid k-ary EDDs. If we run null hypothesis tests over these k-ary specialized EDDs, by the definition of type-I error, we can expect as many as $\alpha \times \binom{|K|}{k}$ false rejections. In other words, under H_0, we can expect a ratio of α missing edges. This is equivalent to setting the threshold for equation 2 as:

$$\gamma = 1 - \alpha$$

Adjusting λ is less straightforward: a high threshold will reject good candidates. A low one will accept spurious ones, triggering unnecessary tests. Even worse, the spurious quasi-cliques tend to have a high cardinality. Once rejected, they will cascade and cause an increase on lower-arity EDDs to be tested as much as $\binom{n}{k+1}$, where n is the arity of the EDD candidate, and k is the current level of the bottom-up exploration.

To solve this dilemma, we propose to use an adaptive value for λ based on the quasi-clique being checked: under H_0, there is no reason to think that any particular subset of the edges from the clique has a higher probability of having missing members. In other words, if a given node has an unexpected low degree, it is most likely connected by spurious edges.

Let N be the number of edges and n the maximum degree of the nodes on a clique with $|V'|$ nodes. Under this null hypothesis, the degree of the nodes should roughly follow a hypergeometric distribution:

$$\Pr(\text{Degree}(v) = d) = \frac{\binom{|E'|}{d}\binom{N-|E'|}{n-d}}{\binom{N}{n}}, \text{ for } v \in V' \quad (5)$$

This fact allows us to perform a statistical test and accept or reject our quasi-clique candidate with a given significance level. Figure 3 shows some examples of this distribution for a quasi-clique with 29 nodes and the critical value for a one-tail test with $\alpha = 0.05$. In other words, if the degree of a node within a quasi-clique candidate is less than the critical value, we can reject the null hypothesis and accept that the set of edges connecting the node are spurious.

In summary, as a constant number of missing edges could be considered too restrictive [21], we consider a fixed ratio to be limiting as well, and harder to make sense of —i.e., why choose $\lambda = 0.6$ and not $\lambda = 0.7$?. We propose that instead, replacing equation 3 with equation 6 could be a more intuitive and flexible approach.

$$\forall v \in V' : \text{CDF}(\text{Degree}(v)) \geq \Lambda \quad (6)$$

Where $0 \leq \Lambda \leq 1$. As with γ and λ, a value of 1 would only accept regular cliques.

The proposed parameterization for γ and Λ are internally consistent since they are both constructed under H_0.

Figure 2 visually summarizes the stages of PRESQ algorithm, and the effects of the parameters γ and Λ on the quasi-clique finding stage.

In the following section, we will show that adapting FIND2 clique validation with ours is enough to improve its performance in run-time and results. The *growing* step improves the efficacy (i.e. more maximal EDDs found) at the cost of a higher run-time.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

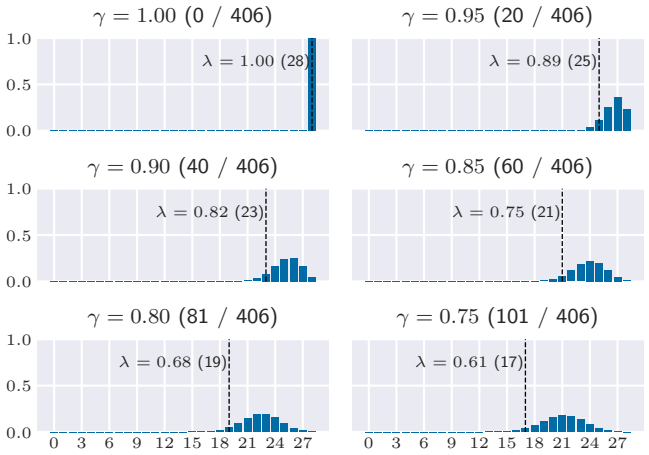

Fig. 3. Distribution of the degree of the nodes under the null hypothesis that the missing edges on the quasi-clique are due to the expected false negative rate of the statistical test. The vertical line corresponds to the one-tail test with $\alpha = 0.05$.

TABLE 1
Set of measurements taken for the quasi-clique finding problem.

Recovery ratio	For each quasi-clique Q' found, we compute the Jaccard index for each found quasi-clique, $J(Q, Q') = \|Q \cap Q'\| \div \|Q \cup Q'\|$. From all the obtained values, we report the maximum. A value of 1 signals a perfect match.
Time	Wall-clock time
Timeouts	How many runs exceeded the timeout

TABLE 2
Combination of parameters for the quasi-clique find problem.

Rank	α	β	Timeout (s)
2	$[0.05, 0.30]$, step 0.05 0.1	0.0 $[0.0 - 0.8]$ step 0.2	240
3	$[0.05, 0.30]$, step 0.05 0.1	0.0 $[0.0 - 0.8]$ step 0.2	300 1200
4	$[0.05, 0.30]$, step 0.05 0.1	0.0 $[0.0 - 0.8]$ step 0.2	1200 3000

5 Experiments

We have implemented in Python a version of FIND2 that validates candidates with statistical tests, and the proposed PRESQ. Both share most of the code, including initialization and statistical tests. Any difference in run-time is only because the modified version searches for quasi-cliques instead of full cliques.

We focus on comparing these algorithms for two main reasons: 1) To prove that quasi-clique finding can outperform clique finding both in run-time and results when the data is noisy, an advantage not necessarily exclusive to EDD finding; 2) While INDs are targeted towards inferring foreign-key relationships and generally of low arity, we expect EDDs to be of high arity —*co-located within a multidimensional space*—, and FIND2 performs well when the arity is high [9].

5.1 Experimental design

We have performed two different sets of experiments: one exclusively benchmarks the quasi-clique search, while the other runs over real-world datasets.

5.1.1 (Quasi-) clique search

This experiment decouples the testing of the quasi-clique search from the uncertainty associated with the data. The test accepts as parameters the rank for the hyper-graph k, the cardinality for the clique n, the number of additional nodes N, the fraction of missing edges α and the fraction of *spurious* edges β. With these parameters, the test performs the following initialization procedure:

1) Create n nodes belonging to the clique
2) Create N additional nodes
3) Create the set E of $\binom{n+N}{k}$ edges connecting *all nodes*
4) Create the set Q of $\binom{n}{k}$ edges belonging to the clique
5) Obtain the set of all edges not belonging to the clique $C = E \setminus Q$

With these sets, and to obtain an estimation of the distribution of the target measurement, it then repeatedly generates noisy versions of the original clique through the following steps:

6) Remove $\alpha \times |Q|$ random edges from the original full clique Q
7) Add $\beta \times |C|$ random edges from C
8) Run FIND2 and PRESQ over the resulting graph

The parameters α and β simulate the effect of type I and type II errors respectively.

PRESQ is configured with $\gamma = 1 - \alpha$ and $\Lambda = 0.05$. The number of additional nodes is fixed to half the number of nodes in the clique: $N = \frac{n}{2}$.

This experiment measures, in a controlled manner, the capability of the algorithms to find the *true* clique and how their run-time is affected by the number of missing and spurious edges. Since the inputs are randomized, some will unavoidably run with exponential complexity, the worst case for all the algorithms. To avoid spending too much time on these extreme cases, the test also accepts a timeout parameter. We describe the measurements we have taken in table 1, and the different parametrizations in table 2.

5.1.2 Real-world datasets

For the statistical tests, we use a non-parametric multivariate test based on k-Nearest Neighbors (kNN) [26], [27], but any other multivariate test could be used. However, regardless of the chosen test, there will always be a number of false negatives bound by the significance level. In any case, the techniques here discussed remain relevant.

The initialization stage of the test is as follows:

1) We load two separate datasets.
2) The constant columns, where every tuple has the same value —including *null*— or only a handful of different values, are dropped. FAIDA authors followed a similar procedure to reduce the number of columns to check [28].
3) A random sample is taken from both relations (it defaults to 200)

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

TABLE 3
Set of measurements taken from individual runs.

Time	Wall-clock time, without accounting for the initialization stage, as this is shared
Number of tests	Time spent looking for quasi-cliques and validating the candidates. Tests can be potentially expensive, so we measure how many statistical tests are necessary.
EDD count	Without removing non-maximal EDDs
Maximal EDD count	Removing non-maximal EDDs
Timeouts	The execution time has a time limit of 3000 seconds. We report the percentage of runs that could not finish within the allocated time window.
Highest arity	The maximum EDD arity found

TABLE 4
Set of measurements derived over the complete set of runs.

Match ratio	It is a ratio between the maximum arity of the maximal quasi-clique found and the *true* maximum EDD possible to find on each separate run. This *truth* is solely based on attribute names. The algorithms can find higher arity EDDs when the values are taken into account. This is a proof of success: the metadata would not have sufficed to capture this trait.
Accuracy	Measured as the number of total returned EDDs, divided by the number of statistical tests executed. A ratio of 1 (best) means that every candidate was accepted by the statistical test, while a ratio of 0 (worst) means that all candidate quasi-cliques were rejected. This value is also affected by the power of the statistical test as a function of dimensionality.

4) The algorithm described in section 4.1 is used to find a set of valid unary EDDs.
5) *All* possible n-EDDs (for $n \in \{2, 3\}$) are generated and validated. The tests begin at different arities in order to compare the resiliency of FIND2 and PRESQ for different initial conditions.
6) Valid n-EDDs are used to create the initial graph passed as input to PRESQ.

The fifth step is performed at different significance levels of $\alpha \in \{0.05, 0.10, 0.15\}$ to verify how the number of missing and spurious edges affects the search algorithms. Typically, MIND would generate the graph (i.e. 3-EDDs are generated from valid 2-EDDs). Nonetheless, we start with all possible n-EDDs for simplicity: it is easier to model and understand how many missing edges are expected as a function of α.

The input for both search algorithms is, thus, identical at every run. However, since there is an unavoidable effect of the randomization of the sampling in step 3 and the N-dimensional permutation tests, we have repeated the experiment. As a result, we are confident that the difference is significant and not due to chance.

While FIND2 has no parameters beyond the initial set of EDDs, PRESQ requires a value for both γ and Λ. As we mentioned earlier, it makes sense to bind γ to the expected number of missing edges (false negatives): $\gamma = 1 - \alpha$. For Λ, we have tested with the values 0.05 and 0.1 since lower values yield too many accidental quasi-cliques, while higher values defeat the tolerance introduced by γ.

To measure the efficacy (EDDs finding) and efficiency (run-time) of the algorithms, we took the measurements summarized in tables 3 and 4.

Given the variability and the number of dimensions, it can be hard to assess the quality of the results. As a general guideline, we consider:

- The higher the match ratio, the better: the highest arity EDD is potentially the most interesting and selective candidate for cross-matching.
- For a similar match ratio, the lower the run-time, the better.

For a similar match ratio, a higher number of maximal EDDs is desirable. Arguably not for the IND discovery — after all, a few good candidates may suffice—, but it proves the capacity of finding maximal quasi-cliques.

It is important to note that some of these measures are interdependent. For instance, if a maximal EDD with a higher arity is found, the number of EDDs should generally decrease. Conversely, if a true, high arity candidate is rejected, multiple generalizations will be considered and possibly accepted, increasing the number of unique EDDs. Similarly, finding more maximal EDDs implies running more statistical tests, so the run-time will be worse. Ultimately, it is up to the user to decide what is more important and parameterize the algorithm accordingly.

We have run the tests disabling the limitation on the degree ($\Lambda = 0$) and the limitation on the total number of edges ($\gamma = 0$). In this manner, we can evaluate if there is any difference when using one, the other, or both.

TABLE 5
Summary of the datasets used for validation.

Dataset	Tables	Rows	Columns	1-EDD
Mortgage/Treasury	2	1k + 1k	16 + 16	26
Ailerons/Elevators	2	14k + 17k	41 + 19	44
DC2	2	198k + 193k	39 + 33	279
AFDS	4	172×4	8×4	63
Waveform	2	5k + 5k	22 + 41	145
KEEL	43	43 — 41k	444	972
ChEMBLDB	79	5 — 19M	418	599

Datasets: To test the algorithms, we have run them over two pairs of relations from the KEEL regression datasets [29], the training and test catalog from the *Euclid photometric-redshift challenge* [30], and a set of sensor measurements from an aircraft fuel distribution system [31]. For the scalability tests, we have used the full KEEL regression dataset, two variants from the Waveform Database Generator [32], [33], and versions 29 and 30 of the ChEMBL database [34].

Some statistics about these datasets are summarized in table 5.

Mortgage / Treasury contain the same data, permuted by rows and by columns. These datasets are an example of data de-duplication.

Ailerons /Elevators share their origin (control of an F16 aircraft) but have different sets of attributes. These datasets are an example of data fusion.

DC2 comes from a single catalog of astronomical objects, split based on the sky coordinates. The authors masked some of the attributes of the training set (i.e., coordinates and the target attributes: red-shift). Therefore, both catalogs share some of the attributes but for different sources. A naive one-to-one schema matching will easily mistake these

attributes for small sample sizes. In contrast, for bigger samples some true correspondences will be falsely rejected. These datasets require some more resilient methods capable of working on a multidimensional space. These datasets are an example of schema inference/matching and automatic feature discovery.

Aircraft Fuel Distribution System (AFDS) comprises five different files, all sharing the same schema but containing sensor measurement values for different scenarios: one nominal, and four abnormal. Our implementations of FIND2 and PRESQ can process the five files at the same time.

Waveform Database Generator We use version 1, with 21 attributes, and version 2, which shares the same 21 attributes and adds 19 extra features that are just Gaussian noise. This 21-ary EDD between the datasets goes beyond the maximum 7-ary evaluated in previous works [9]. Additionally, the number of attributes and their distribution similarity generates many false positives at low dimensionality, stressing the capability of processing noisy, dense, graphs.

ChEMBL Database We use versions 29 and 30 of the ChEMBL database, each of size 20GiB. They are stored on BEEGFS, a clustered filesystem. We evaluate the scalability with respect to the number of columns, adding tables progressively. In this scenario, the overhead introduced by the sampling becomes significant.

The two pairs from KEEL (i.e. Mortgage/Treasury and Ailerons/Elevators) were found running over the whole KEEL dataset initial versions of the algorithms described in this paper, proving their capabilities. We report the performance of this exercise, together with the other two scalability tests, in section 5.3.3.

5.2 Environment

The tests were run on a cluster, where each node is fitted with an Intel(R) Xeon(R) Gold 6240 CPU at 2.60GHz with 36 virtual cores, running on a standard CentOS Linux 7.9. The default memory allocation per core was 3 GB.

For the (quasi-)clique search, we submitted one job with as many tasks as parameter combinations described in table 2 and 1 CPU per task, for cliques of size $10, 20$ and 30. We chose the time limit based on the measured run-time from early test runs.

For the real dataset tests, we submitted jobs with 8 tasks and 1 CPU per task, limited to 24 hours. The objective of concurrent runs was to increase the number of data points since the code has not been parallelized.

Finally, we executed ten randomized runs for each increment on the number of columns for the scalability tests.

5.3 Results

In this section, we will summarize the results from our test setup. We will discuss our interpretation in section 6.

5.3.1 (Quasi-) clique search

We summarize the *wall-time* and *recovery ratio* metrics by estimating their distribution mean and its associated standard error following the Bootstrap method. The *timeout* is measured by counting how many runs fail to find a quasi-clique within the allocated time window.

While the *wall-time* distribution is far from Gaussian, we consider that randomizing the input, pruning the long-running cases, and averaging the results of a few short-running iterations is a valid usage of the algorithms. This makes comparing the means a reasonable assessment.

Influence of spurious edges: We show in figure 4 the performance of the algorithms for 3-hypergraphs and different ratio of spurious edges. The exponential worst-case complexity becomes more apparent the more connected nodes there are. FIND2 is the most affected, but at some point, PRESQ performance also degrades significantly and eventually also fails to finish on time. These results confirm that spurious edges influence the *run-time* of these algorithms very negatively [35].

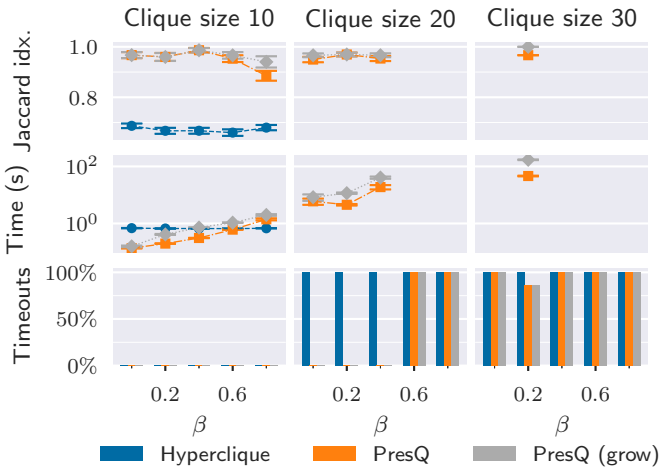

Fig. 4. Recovery ratio and run-times for cliques on uniform 3-hypergraphs for different ratios of spurious edges (β). Each data-point corresponds to 15 runs.

Influence of missing edges: Figure 5 shows that our proposal generalizes for hypergraphs. PRESQ with the growing stage enabled, oscillates very close to the original clique even when 30% of the edges are missing. However, the number of timeouts increases given that the algorithm needs to traverse more levels from the seed to the maximal quasi-clique. Interestingly, there is an inverse correlation between the number of missing edges and run-time.

Influence of correlated ratios: In a more realistic scenario — i.e., when using statistical tests — as the number of missing edges increases, the number of spurious edges should decrease. We have run tests with the growing stage enabled for different parametrization on the node degree threshold. This includes a regular λ parameter with a value of 0.8 chosen based on good empirical results we obtained during early iterations of this work. The correlation between β and α is based on the empirical statistical power of the kNN test as a location test on k dimensions and a sample size of 100. In all cases, $\gamma = 1 - \alpha$.

Figure 6 summarizes the results. A hand-picked parameter of $\lambda = 0.8$ can perform well for some hypergraphs but quickly underperforms as the hypergraphs become noisier. To the contrary, our proposal based on the hypergeometric distribution remains stable. However, disabling the degree limitation performs better for this particular setup. This

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

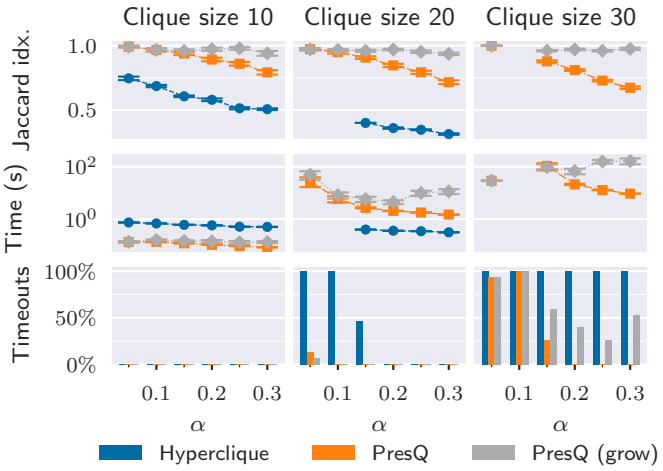

Fig. 5. Recovery ratio and run-times for cliques on uniform 3-hypergraphs for different ratios of missing edges (α). Each data-point corresponds to 15 runs.

makes sense since there is no correlation between missing edges.

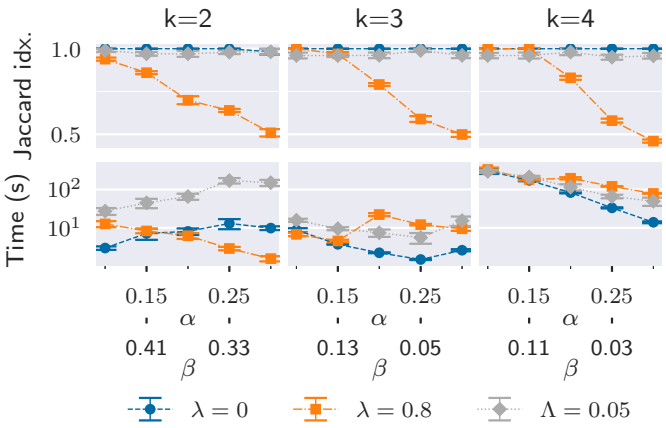

Fig. 6. Recovery ratio and run-times for cliques of size 20 on uniform $(2, 3, 4)$-hypergraphs. In this setup there were no timeouts. Each data-point corresponds to 5 runs.

5.3.2 Real-world datasets

The initial randomized state heavily influences the proposed performance measurements. Their distribution can not be assumed normal. Purely comparing their means is not enough to assess the validity of our proposal, and we also need an estimation of variability.

The metric of choice used to compare our measurements is the *percent difference* between sample means, being its sample estimator [36]:

$$\hat{\phi} = \frac{\hat{\mu}_{\text{PRESQ}} - \hat{\mu}_{\text{FIND2}}}{\hat{\mu}_{\text{FIND2}}} \tag{7}$$

The distribution of $\hat{\phi}$ can be estimated using bootstrapping. In this manner, we obtain the estimated population mean and standard deviation. Finally, we compute the 95% confidence interval $\hat{\mu_\phi} \pm 1.96\hat{\sigma_\phi}$

Figure 7 shows this confidence interval for match ratio, unique EDDs, number of tests and wall time (columns) for a significance level of 0.10, against the different datasets (rows).

The DC2 case is particularly interesting. The attributes of the datasets are relatively numerous —compared to the others— and very similar in their distributions. A low initial significance level will generate very dense graphs, with a few missing edges, and many spurious, which impacts the performance considerably. This is a known issue of FIND2 [35]. Increasing the significance level reduces the number of spurious edges, at the cost of missing true ones. Consequently, the efficiency is improved at the cost of the efficacy. PRESQ allows us to increase the significance level without sacrificing much efficacy.

For the AFDS dataset, when comparing the maximum EDD arity found per pair of files, scenarios two and three are the most similar. We can obtain this insight without even knowing what the schema nor the content of the files are. After seeing this result, we checked the original paper from where the dataset was obtained, verifying that, indeed, they are "two closely related scenarios" [31]. We consider this another proof of the utility of the proposed techniques.

Table 6 summarizes the overall results when we execute our tests over the datasets *Mortage vs Treasury*, *Ailerons vs Elevators* and *DC2* for different values of γ and Λ — note that FIND2 is equivalent to either one of the two parameters set to 1.. For run-time, match ratio, and the number of unique EDDs, we provide the first and third quartiles. The *precision* column shows a measure of how many candidates are accepted by the statistical test. A value of 1 means that all candidates were valid EDDs.

When the search algorithm looks for cliques (first entry for each dataset), the precision is high since almost all candidates were accepted. However, these candidates are, on average, of lower arity. This is visible on the *Match* columns. As the potential maximal arity becomes higher — e.g., *DC2*— the chance of having missing edges increases, thus making the search more resource intensive.

On the other hand, in a too permissive setup where only γ constrains the quasi-cliques (second entry), the algorithm is too eager and accepts EDD candidates later rejected either by the statistical test or by the limitation of not accepting duplicated columns. The precision is low, and the search time increases as well.

Our proposed Λ parameter, based on the *expectation* on the number of missing edges, is more effective at constraining the set of candidates even when used alone (third entry). The precision increases and the run-time is reduced. When combined with γ (fourth and fifth entries), the precision increases and fewer tests are required.

As an illustration of this balance, let us examine in more detail the consequences of the different Λ parameterization following the process shown in figure 2 when running over the DC2 dataset. The first four stages are unaffected by this parameter:

(a) As described in section 4.1, an interval tree is built over the attributes from both relations. Only overlapping ranges are compared, reducing by 27% the number of tests required.

(b) 810 KS tests need to be done. 49 pairwise combinations are considered equally distributed (uEDDs).
(c) $(n \times (n-1)) \div 2 = 1176$ edges are build combining all uEDDs and validated using the kNN test. 612 edges are considered valid.
(d) The initial graph has half as many edges as the complete graph. Since we know the ground truth, we can extract the sub-graph induced by the set of true uEDD and find the number of missing edges to be ≈ 0.10 on average, as we expected.

The following table exemplifies the consequences of different values of Λ — see equation 6 — on the count and size of the found quasi-cliques (e) and the number validated by the kNN test (f). Those invalid are 'decomposed' into candidate 3-EDDs, validated, and used to build a 3-hypergraph (d) feedback to the stage (e) for the next iteration.

	Quasicliques	Valid	Median size
$\Lambda = 0.00$	2385	292	19
$\Lambda = 0.05$	107	64	12
$\Lambda = 1.00$[3]	53053	52291	6

For $\Lambda = 0$, the search algorithm is too greedy and accepts quasi-cliques that are poor candidates. Too many are invalid and need to be feedback to the algorithm, increasing run-time. For $\Lambda = 1$, the search algorithm is too restrictive. Its precision is high, but it spends much time enumerating small cliques. $\Lambda = 0.05$ provides the right balance, improving the result and performance.

Finally, the growing stage increases the number of candidates of all arities. This requires a more exhaustive traversal of the search space and the execution of more tests, increasing the total run-time. While we run the growing stage over *all* found seeds this stage could be restricted only to a subset of the most interesting *seeds* —e.g., highest cardinality.

5.3.3 Scalability tests

For measuring the scalability of our algorithm, we executed the algorithm over the KEEL, Waveform, and ChEMBL datasets, progressively adding columns, measuring run-time; the number of 1, 2, and n-EDDs; and the number of unary tests saved by the interval tree. In all cases the chosen parameterization is $\alpha = 0.1$, $\Lambda = 0.05$, $\gamma = 1 - \alpha$ and 200 samples. We set the run-time limit at 3000 seconds. The relations and their attributes are consistently added in alphanumeric order. Figure 8 summarizes our results.

The accepted 1-EDDs are used to compute all the possible 2-EDDs, while the accepted 2-EDDs define the initial edges for the n-EDD finding.

The *KEEL* dataset contains 43 different relations. The interval tree saves around 45% of the tests since many columns do not overlap. The number of EDDs increments in 'bursts' when a relation that matches a previous one enters the pool. This is due to the existence of high arity EDDs (16, 12, 7 and 6). The high number of 2-EDDs makes the growing stage eventually impractical.

For the *ChEMBL* databases, we have used a naive random sampling that only requires a single pass over the entire database. Even then, the reading time is small with respect to the rest of the EDD finding algorithm. The number of 1-EDDs increments steadily as relations are added, but 2 and n-EDDs remain relatively stable — the corresponding error bars overlap—, and so does the run-time. The arities are lower than those from KEEL, the maximum being 6 for the tables `molecule_dictionary` and `compound_properties`. The interval tree saves 57.9% tests for 1-EDDs.

Finally, while the *Waveform Generator* datasets are the smallest, it is the case for which the algorithm shows the worst performance. This is due to the high arity possible (up to 20), and because the attributes are hard to distinguish —the interval tree can not save even one test. The number of 2-EDDs grows super-linearly with respect to the number of attributes, resulting in a very dense and noisy initial graph with many possible quasi-cliques.

From these experiments, we can conclude that the algorithm scales well in relation with the number of relations and columns and that the sampling has a low impact even for big datasets. However, when the statistical test has low power for the input data, the run-time degrades significantly even for moderate input sizes since the search space is combinatorial and little pruning is possible. Then the user can choose a different test or increase the sample size to increase the power. Figure 9 exemplifies the effect the sample size has on the result set and, therefore, run-time for the complete Waveform dataset. The power of the test is low for low dimensionality, and the initial graph becomes very dense.

6 Discussion

Identifying shared attributes between multiple numerical datasets is an interesting problem. It combines the challenging nature of algorithms devised to find Inclusion Dependencies, an NP-hard problem [8], with the unavoidable uncertainty of statistical methods. This uncertainty reflects as falsely rejected EDDs and falsely accepted EDDs.

FIND2 is an algorithm that maps inclusion dependencies to hyper-cliques, which generally performs at least as well as the alternatives [9]. It is not strongly coupled to the discrete nature of the underlying data. However, its ability —and of most, if not all, of the existing algorithms— to find high arity EDDs will be impaired by the level of false rejections.

A lower rejection threshold could compensate for this. Yet, it increases the number of false detections, which is a known factor that degrades its performance significantly as well as other hypergraph-based methods' performance [35]. We have experimentally confirmed this problem in section 5.3.1.

We propose a new algorithm based on quasi-cliques, where a candidate is accepted even if some edges are missing. This algorithm has three parameters:

- The ratio of missing edges (γ).
- The tolerance on the number of missing edges connecting a node from the quasi-clique (Λ).
- Whether to use the found quasi-clique as seeds.

We provide a generalization of this parameterization from regular 2-graphs [21] to uniform n-graphs in equations 2 and 3.

3. Equivalent to clique finding

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

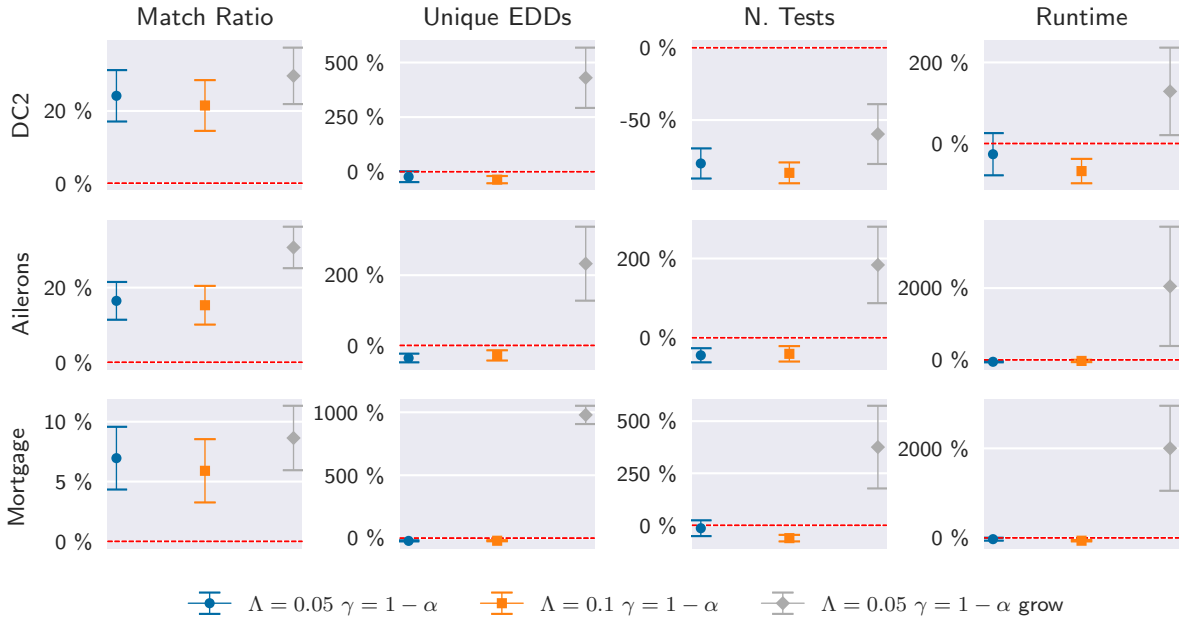

Fig. 7. 95% confidence intervals for the percent difference (equation 7) between FIND2 and three parameterizations of PRESQ for the DC2, Ailerons vs. Elevator, and Mortgage vs. Treasury datasets. Intervals that do not intersect the horizontal dashed line at 0% show a statistically significant result. For ratio, higher is better. For tests and run-time, lower is better. Unique is harder to assess since the results also depend on the statistical power of the chosen test. Since the growing stage can generate many candidates, a low-powered test will accept many false EDDs.

The results shown in the quasi-clique test set (section 5.3.1) demonstrate that the seed stage of PRESQ provides results close to the original cliques on uniform n-hypergraphs. The growing stage can recover them even for a high number of missing edges (up to 30%), at the expense of a higher run-time. These results also prove that the degree threshold based on the hypergeometric distribution offers comparable performance to a hand-picked ratio λ while being more stable and predictable.

For real datasets, the ratio of missing edges can be intuitive to configure (simply $\gamma = 1 - \alpha$, where α is the test significance level), but λ can be harder to interpret. We propose instead an intuitive and statistically interpretable method to adapt the threshold to the degree dynamically, which is expected to follow a hypergeometric distribution and can be adjusted based on the quasi-clique itself, as shown in equation 6.

While our tests on artificial hypergraphs seem to point to the redundancy of the parameter Λ, the results shown in the real-world test set (section 5.3.2) prove that for real noisy graphs, the combination of both performs consistently better than either of them separately. The γ parameter enables recovery from missing edges and, at the same time, Λ avoids too many false positives due to the existence of spurious edges. Thanks to them, the efficacy can be kept even while maintaining, or even increasing, the significance level of the tests. This reduces the risk of decreased performance since the density of the graphs can be kept under control.

If a more exhaustive listing of maximal quasi-cliques is required, the initial set of quasi-cliques can be used as *seeds* to grow other quasi-cliques by adding suitable vertices. The results shown in section 5 demonstrate that this method is capable of finding considerable more maximal quasi-cliques (not contained in any other found quasi-cliques) at the expense of a higher run-time. This is due both to the traversal of the search space and the validation of the EDDs represented by the quasi-cliques.

The loss of accuracy introduced by this growing stage is minor when starting at $n = 3$, which means that the statistical test could not reject most candidates. However, for an initial $n = 2$, most candidates were rejected. We consider that this is mostly due to the lack of power of the kNN test for low dimensions, which introduces many spurious edges.

The overall run-time of the EDD finding algorithms is heavily influenced by the chosen parameter values. A strict parametrization will reject most seed candidates, and the quasi-clique search will fall into exponential complexity. Conversely, a flexible one will be faster at finding quasi-clique candidates. Yet, the statistical test will likely reject them, causing their decomposition into an exponential number of newer, smaller candidates. In general, a balanced parametrization based on both γ and Λ is more predictable.

As a final remark, the set of accepted EDDs may contain several false positives depending on the power of the multivariate statistical test. A second pass adapted from the foreign key literature can prove helpful. For instance, we can envision a ranking based on the previously discussed *randomness* [15] in order to decide which set of EDDs is more suitable for cross-matching the datasets.

7 THREATS TO VALIDITY

Internal validity: The results shown in the experiments described in section 5 could risk being just a fluctuation, not due to an underlying algorithmic improvement. However, the experimental design described in section 5.1 significantly reduces this possibility thanks to the randomization of the initial conditions, and the number of measurements.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

TABLE 6
Summary of run-time, matching ratio (based on name), and number of maximal quasi-cliques found accepted by the statistical test. The significance level is $\alpha = 0.1$. N corresponds to the number of randomized runs. PRESQ(G) identifies PRESQ with the growing stage.

	Λ	γ	Time (s)		Match		Unique		Prec.	N	Timeouts
			Q1	*Q3*	*Q1*	*Q3*	*Q1*	*Q3*			
Mortgage vs Treasury											
FIND2			0.44	0.64	0.75	1.00	12	21	0.99	527	0.0%
PRESQ	**0.00**	0.9	44.86	459.04	0.94	1.00	11	15	0.06	212	0.0%
PRESQ	0.05	**0.0**	0.71	11.08	0.88	1.00	11	17	0.49	535	0.0%
PRESQ	0.05	0.9	0.76	10.61	0.88	1.00	11	17	0.57	507	0.0%
PRESQ	0.10	0.9	0.73	1.99	0.84	1.00	11	17	0.75	503	0.0%
PRESQ(G)	0.05	0.9	47.10	247.39	0.88	1.00	125	262	0.22	503	0.0%
Ailerons vs Elevators											
FIND2			5.63	48.41	0.78	1.00	142	291	0.98	93	0.0%
PRESQ	**0.00**	0.9	8.82	36.75	1.00	1.22	88	174	0.24	128	0.0%
PRESQ	0.05	**0.0**	22.68	52.87	1.00	1.22	113	239	0.16	126	0.0%
PRESQ	0.05	0.9	7.51	20.12	1.00	1.11	86	198	0.35	60	0.0%
PRESQ	0.10	0.9	6.83	22.40	1.00	1.11	89	205	0.41	60	0.0%
PRESQ(G)	0.05	0.9	57.86	674.26	1.11	1.25	321	1062	0.22	60	0.0%
DC2											
FIND2			74.94	805.71	0.60	0.71	73	150	0.90	53	34.0%
PRESQ	**0.00**	0.9	681.51	1536.19	0.68	0.69	102	200	0.01	16	87.5%
PRESQ	0.05	**0.0**	40.07	189.45	0.80	0.93	46	115	0.10	21	47.6%
PRESQ	0.05	0.9	25.57	214.27	0.76	0.89	46	113	0.14	53	13.2%
PRESQ	0.10	0.9	18.61	144.98	0.76	0.87	42	98	0.18	52	23.1%
PRESQ(G)	0.05	0.9	458.26	1881.02	0.81	0.93	518	798	0.23	52	50.0%

Looking at the results summarized in table 6, it is evident that, on average, the quasi-clique-based searching algorithm consistently performs better both in terms of run-time and ability to find the maximal EDD. It has enough runs as to make the difference significant. It is worth mentioning that there are proposed heuristics [13] to find higher arity EDDs, even when edges are missing, by merging found lower-arity EDDs and testing them instead. Nonetheless, we consider that the run-time differences are significant enough to make the quasi-clique-based search a better approach in those cases. Even so, that heuristic can be applied to the output of our proposed algorithm as well.

We have implemented FIND2 and PRESQ from scratch, with both sharing many parts of the code — i.e., data structures, statistical tests, etc. While there is room for optimizations, both would benefit. Since the relative differences would remain similar, we are confident that the gains come from the algorithm rather than its implementation.

External validity: The experiments have been run over four different datasets of diverse nature and from three separate sources. The chosen statistical tests for uni- and multi- dimensional distributions have not been customized to any of them. However, a better statistical test can be used if the underlying data distribution is more or less known (or simply suspected), which may reduce, or even remove, the advantage of the quasi-clique approach. Although it is also unlikely that the performance would be any worse, since an entire clique is still a quasi-clique, and our algorithm can identify all of them, as well as the original FIND2 algorithm.

One significant caveat of our approach is that it may not find any dependencies if prior filtering has been applied to only one of the two relations (i.e., signal to noise filtering). This is a limitation of the validation step. This issue was also recognized on the original FIND2 proposal [13].

All the necessary code to reproduce our tests, our measurements, and figures, are publicly available[4].

8 CONCLUSIONS AND FUTURE WORK

Finding sets of equally-distributed dependencies between numerical datasets is a similar problem to that of finding Inclusion Dependencies between tables in a relational model. However, the statistical nature of tests, with their potential uncertainties, can make more difficult their finding and considerably degrade the performance of existing algorithms. This problem can be mapped to finding quasi-cliques, as the IND problem can be mapped to finding full cliques.

In this paper, we have introduced the concept of EDD, similar to the IND from the relational domain. We have proposed PRESQ, a new algorithm based on the search of maximal quasi-cliques on hyper-graphs. We have proven that by limiting the quasi-cliques by the number of missing edges, and the degree of the nodes, our algorithm can successfully identify shared sets of attributes.

In general, comprehensive approaches will be needed to find very high arity EDDs, given the complexity of the IND/EDD discovery problem. For further work, we can envision three main routes:

Improving the finding of quasi-cliques in hypergraphs Via novel algorithms or by generalizing some of the many existing techniques [37].

Data-aware algorithms For instance, the correlation matrix on both sides of the EDD is likely to be similar. Perhaps this kind of information can be used to augment the algorithms, or inform the traversal.

4. https://doi.org/10.5281/zenodo.6865856

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

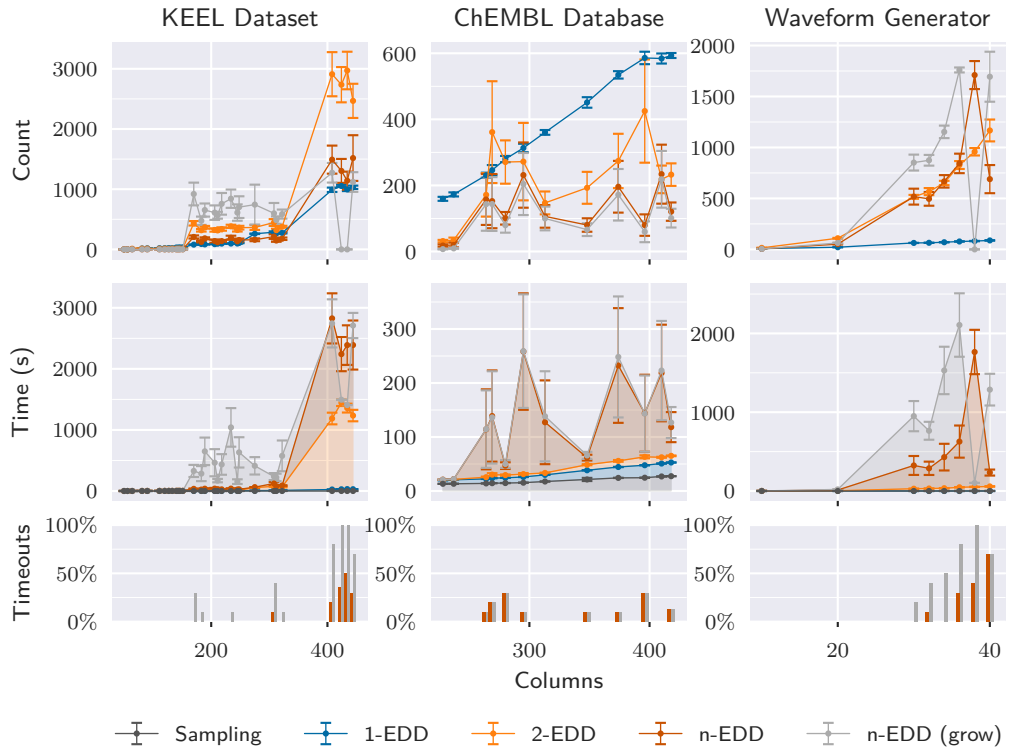

Fig. 8. Scalability of PRESQ with respect to the number of columns. The top row corresponds to the number of EDDs with arities 1, 2, and $n \geq 3$. The middle row shows the time spent on each stage: sampling, searching, and testing for the different arities. Note that the two n-EDDs variants are stacked over the previous stages, displaying the total run-time. The last row shows the percentage of runs timed out at 50 minutes. Each data-point summarizes between 10 and 13 randomized runs.

Dimensionality reduction Searching for quasi-cliques involves exponential time complexity on the number of nodes. Thus, applying a dimensionality reduction beforehand would reduce the total run-time and also decrease the noise. Nonetheless, a complication arises from the premise that we do not know which attributes are shared.

ACKNOWLEDGMENTS

This research was funded by the Spanish AEI (DOI:10.13039/501100011033) through the project CRÊPES (ref. PID2020-115844RB-I00) with ERDF funds.

REFERENCES

[1] A. Alawini, "Identifying relationships between scientific datasets," Ph.D. dissertation, Portland State University, 2016. [Online]. Available: https://pdxscholar.library.pdx.edu/open_access_etds/2922/

[2] Y. Zhang and Y. Zhao, "Astronomy in the big data era," *Data Science Journal*, vol. 14, 2015. [Online]. Available: https://datascience.codata.org/articles/10.5334/dsj-2015-011/

[3] S. Idreos, O. Papaemmanouil, and S. Chaudhuri, "Overview of Data Exploration Techniques," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15*, 2015, pp. 277–281. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2723372.2731084

[4] A. Alvarez-Ayllon, M. Palomo-Duarte, and J. M. Dodero, "Interactive Data Exploration of Distributed Raw Files: A Systematic Mapping Study," *IEEE Access*, vol. 7, pp. 10 691–10 717, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8540356/

[5] V. Silva, D. de Oliveira, P. Valduriez, and M. Mattoso, "Analyzing related raw data files through dataflows," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 8, pp. 2528–2545, 2016. [Online]. Available: http://dx.doi.org/10.1002/cpe.3616

[6] K. D. Borne, "Data Mining in Astronomical Databases," in *Mining the Sky*. Springer, 2000, pp. 671–673. [Online]. Available: http://dx.doi.org/10.1007/10849171_88

[7] Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: a survey," *The VLDB Journal*, vol. 24, pp. 557–581, 2015. [Online]. Available: https://link.springer.com/article/10.1007%2Fs00778-015-0389-y

[8] M. Kantola, H. Mannila, K.-J. Räihä, and H. Siirtola, "Discovering functional and inclusion dependencies in relational databases," *International journal of intelligent systems*, vol. 7, pp. 591–607, 1992. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/int.4550070703

[9] F. Dürsch, A. Stebner, F. Windheuser, M. Fischer, T. Friedrich, N. Strelow, T. Bleifuß, H. Harmouch, L. Jiang, T. Papenbrock, and F. Naumann, "Inclusion dependency discovery: An experimental evaluation of thirteen algorithms," in *Proceedings of the 28th ACM international conference on information and knowledge management*. Association for Computing Machinery, 2019, pp. 219–228. [Online]. Available: https://doi.org/10.1145/3357384.3357916

[10] J. Kossmann, T. Papenbrock, and F. Naumann, "Data dependencies for query optimization: A survey," *The VLDB Journal*, vol. 31, no. 1, pp. 1–22, 2022.

[11] F. De Marchi, S. Lopes, and J.-M. Petit, "Efficient algorithms for mining inclusion dependencies," in *International conference on extending database technology*. Springer, 2002, pp. 464–476. [Online]. Available: https://link.springer.com/chapter/10.1007%2F3-540-45876-X_30

[12] F. De Marchi and J.-M. Petit, "Zigzag: a new algorithm for mining large inclusion dependencies in databases," in *Third IEEE international conference on data mining*. IEEE, 2003, pp. 27–34. [Online]. Available: https://ieeexplore.ieee.org/document/1250899

[13] A. Koeller and E. A. Rundensteiner, "Discovery of high-dimensional inclusion dependencies," in *Proceedings 19th international conference on data engineering)*. IEEE, 2003, pp. 683–685. [Online]. Available: https://ieeexplore.ieee.org/document/1260834

[14] A. Rostin, O. Albrecht, J. Bauckmann, F. Naumann, and U. Leser, "A machine learning approach to foreign key discovery." in

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

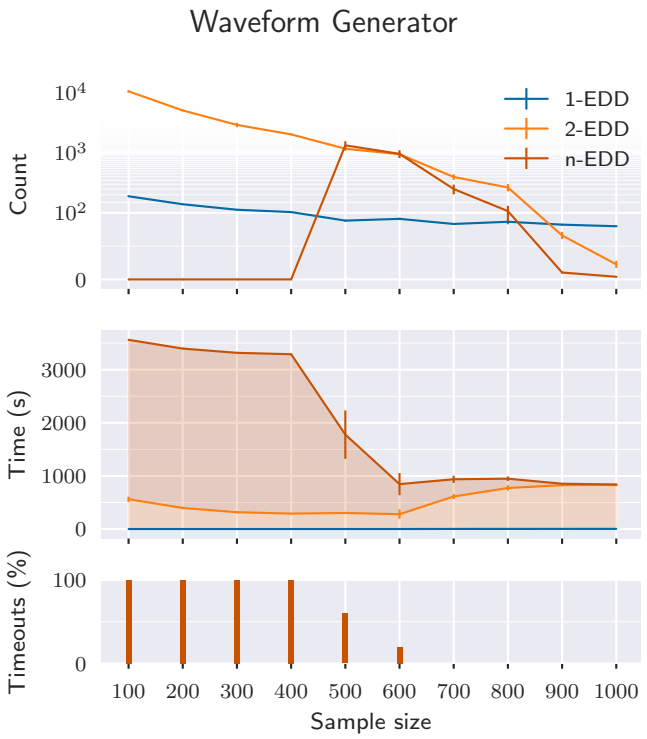

Fig. 9. Scalability of PRESQ with respect to the number of samples for the Waveform datasets. Note that for the top row the y axis is linear between 0 and 100, and logarithmic afterwards. Each data-point summarizes 10 randomized runs.

WebDB, 2009.

[15] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava, "On multi-column foreign key discovery," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 805–814, 2010.

[16] L. Jiang and F. Naumann, "Holistic primary key and foreign key detection," *Journal of Intelligent Information Systems*, vol. 54, no. 3, pp. 439–461, 2020.

[17] M. Hallin, G. Mordant, and J. Segers, "Multivariate goodness-of-fit tests based on Wasserstein distance," *Electronic Journal of Statistics*, vol. 15, no. 1, pp. 1328–1371, 2021.

[18] M. A. Casanova, R. Fagin, and C. H. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies," *Journal of Computer and System Sciences*, vol. 28, pp. 29 – 59, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022000084900758

[19] R. H. Randles and D. A. Wolfe, "Introduction to the theory of nonparametric statistics," John Wiley, Tech. Rep., 1979.

[20] A. Alvarez-Ayllon, M. Palomo-Duarte, and J.-M. Dodero, "Inference of common multidimensional equally-distributed attributes," 2021. [Online]. Available: https://arxiv.org/abs/2104.09809

[21] M. Brunato, H. H. Hoos, and R. Battiti, "On effectively finding maximal quasi-cliques in graphs," in *International conference on learning and intelligent optimization*. Springer, 2007, pp. 41–55. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-540-92695-5_4

[22] J. L. Hodges, "The significance probability of the smirnov two-sample test," *Arkiv för Matematik*, vol. 3, no. 5, pp. 469–486, 1958.

[23] S.-V. Sanei-Mehri, A. Das, and S. Tirthapura, "Enumerating top-k quasi-cliques," in *2018 IEEE international conference on big data (big data)*, 2018, pp. 1107–1112. [Online]. Available: https://ieeexplore.ieee.org/document/8622352

[24] T. Uno, "An efficient algorithm for solving pseudo clique enumeration problem," *Algorithmica*, vol. 56, pp. 3–16, 2010. [Online]. Available: https://link.springer.com/article/10.1007/s00453-008-9238-3

[25] I. Tomescu, "Le nombre maximum de cliques et de recouvrements par cliques des hypergraphes chromatiques complets," *Discrete Mathematics*, vol. 37, no. 2, pp. 263–277, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X81902259

[26] N. Henze, "A multivariate two-sample test based on the number of nearest neighbor type coincidences," *The Annals of Statistics*, vol. 16, no. 2, pp. 772–783, 1988. [Online]. Available: http://www.jstor.org/stable/2241756

[27] M. F. Schilling, "Multivariate two-sample tests based on nearest neighbors," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 799–806, 1986. [Online]. Available: https://www.jstor.org/stable/2241756

[28] S. Kruse, T. Papenbrock, C. Dullweber, M. Finke, M. Hegner, M. Zabel, C. Zöllner, and F. Naumann, "Fast approximate discovery of inclusion dependencies," in *Datenbanksysteme für business, technologie und web (BTW 2017)*. Gesellschaft für Informatik, Bonn, 2017, pp. 207–226. [Online]. Available: https://dl.gi.de/handle/20.500.12116/629

[29] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[30] G. Euclid Collaboration: Desprez, S. Paltani, J. Coupon, and et al., "Euclid preparation. X. The Euclid photometric-redshift challenge," *Astronomy & Astrophysics*, vol. 644, p. A31, Dec. 2020. [Online]. Available: https://www.aanda.org/articles/aa/abs/2020/12/aa39403-20/aa39403-20.html

[31] Y. Gheraibia, S. Kabir, K. Aslansefat, I. Sorokos, and Y. Papadopoulos, "Safety+AI: a novel approach to update safety models using artificial intelligence," *IEEE Access*, vol. 7, pp. 135 855–135 869, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8839038

[32] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[33] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification And Regression Trees*. Routledge, 1984. [Online]. Available: https://doi.org/10.1201/9781315139470

[34] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. P. Magariños, J. P. Overington, G. Papadatos, I. Smit, and A. R. Leach, "The ChEMBL database in 2017," *Nucleic Acids Research*, vol. 45, no. D1, pp. D945–D954, 11 2016. [Online]. Available: https://doi.org/10.1093/nar/gkw1074

[35] A. Koeller and E. A. Rundensteiner, "Heuristic strategies for the discovery of inclusion dependencies and other patterns," in *Journal on Data Semantics V*. Springer, 2006, pp. 185–210. [Online]. Available: https://link.springer.com/chapter/10.1007%2F11617808_7

[36] F. Campelo and F. Takahashi, "Sample Size Estimation for Power and Accuracy in the Experimental Comparison of Algorithms," *Journal of Heuristics*, vol. 25, no. 2, pp. 305–338, Apr. 2019. [Online]. Available: https://link.springer.com/article/10.1007%2Fs10732-018-9396-7

[37] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems," *European Journal of Operational Research*, vol. 242, no. 3, pp. 693–709, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221714008030

Alejandro Alvarez-Ayllon is a PhD student at the University of Cádiz, where he received the Master degree on Computer Science Engineering in 2010. He is currently combining his studies with a full time position as a Software Engineer at the Astronomy Department of the University of Geneva since the beginning of 2018. He formerly worked at the European Organization for Nuclear Research (CERN) between 2009 and 2018 on different Data Management components for the LHC Computing Grid.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2022.3198252

Manuel Palomo-Duarte is an Associate Professor of Computer Systems & Languages at the University of Cadiz. He has a Computer Science MSc from the University of Seville and a Computer Engineering PhD from University of Cadiz. His main research interests are creative computing, serious games and collaboration, fields in which he has published different contributions in indexed peer-reviewed journals and research conference proceedings.

Juan-Manuel Dodero is a Full Professor of Computer Systems & Languages at the University of Cádiz. He has a Computer Science BSc and MSc from the Polytechnic University of Madrid and a Computer Engineering PhD from Carlos III University. He has worked before as a Lecturer for other Spanish universities and as an R&D consultant for ICT companies.