Noise Resistant Multidimensional Data Fusion via Quasi-Cliques on Hypergraphs

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

16-12-2021 / 17-12-2021

CITATION

Alvarez-Ayllon, Alejandro; Palomo-duarte, Manuel; Dodero, Juan Manuel (2021): Noise Resistant Multidimensional Data Fusion via Quasi-Cliques on Hypergraphs. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.17212079.v1>

DOI

[10.36227/techrxiv.17212079.v1](https://doi.org/10.36227/techrxiv.17212079.v1)

Noise Resistant Multidimensional Data Fusion 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, given the different nature of the data, which can be 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 three public datasets, showing promising results both in its capacity to find multidimensional equally-distributed sets of attributes and in run-time.

Index Terms—Inclusion Dependencies, Equally-Distributed Dependencies, Schema Inference, Quasi-clique, Hypergraph

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 for the following reasons: They come from different sources, they were produced without proper guidelines or a combination of both [1], [2], [3].

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) [4]. For more details, including a survey of existing solutions, we refer the reader to an exhaustive systematic mapping of the literature previously published [5]. 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 [6].

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 [7]:

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 [2] is an example of a proposed solution aimed in this direction. It is based on machine learning techniques, such as Support Vector Machines [8], 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. For instance, in figure 1, A and C attributes are identical, and so are B and D . However, it is obvious that we can not use the coordinates A, B to cross-match with C, D .

Fig. 1. $A = C$ and $B = D$. However, $A, B \neq C, D$

Similarly, 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

- 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

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 [9]

The definition of IND is based on set theory, which is not directly applicable to scientific data where measures are in the real domain (i.e. 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* usually refers to the cardinality of the sets of attributes X and Y . For instance, if $|X| = 1$, we talk about unary INDs; if $|X| = 2$, binary or 2-INDs; and, in general, for $|X| = n$, n -ary INDs. For the rest of the paper, we will use this terminology.

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 INDs discovery. In section 3, we introduce the background for our research. In section 4, we propose a novel algorithm based on quasi-cliques that can be used 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 [10]. 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.

Fig. 2. Example structure of the search space as a lattice for an initial set of 4 unary INDs. As an illustration, if the 2-IND 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.

We used a published experimental evaluation [11] 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.

We describe the unary problem and propose our algorithm, tailored to scientific 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 a set U of valid unary inclusion dependencies, the search space for higher-arity candidates is defined by its power set and a partial order relation called specialization [12]:

Definition 1. Let $I_1 = R[X] \stackrel{d}{=} S[Y]$ and $I_2 = R'[X'] \stackrel{d}{=} S'[Y']$. 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. $(R[AB] \stackrel{d}{=} S[EF]) \prec (R[ABC] \stackrel{d}{=} S[CFG])$. However, $R[AB] \stackrel{d}{=} S[DE] \not\prec (R[ACD] \stackrel{d}{=} S[DFG])$

This partial order enables us to structure the search space as a lattice, as exemplified in figure 2. 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 [12] It 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 [13] It 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 [14] It is based on the equivalence between finding n -INDs and finding cliques on n -uniform hypergraphs (a generalization of the concept of graph where each edge connects n nodes). Each unary IND corresponds to a node, and a n -IND corresponds to an edge on a n -uniform hypergraph. Once such a graph is built, maximal IND will have a corresponding clique, although not all cliques will correspond necessarily to a valid IND. 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.

The other three solutions that were evaluated are more dependent on the data domain, however, and not suitable to our purposes. We are more interested in continuous data (i.e. length, size, mass, flux), which can be stored with arbitrary precision, in floating-point representation, and can very possibly have an associated uncertainty. To be thorough, we briefly mention them here:

BINDER [15] It is capable of finding both unary and n -ary INDs using the same method without assuming that the datasets fit into memory. Yet, it relies on hashes, however, which are not well suited for uncertain numerical data.

FAIDA [16] It is an approximate INDs discovery algorithm that guarantees that all true INDs will be found in a performant way, at the risk of obtaining false positives. Similar to MIND and BINDER, it uses a bottom-up traversal approach. Thus, their performance comes from the operation over hashes rather than tuples, which again poses problems to our use cases.

MIND2 [11] It does not perform any candidate generation. It only needs to access the data once when building what they call *unary IND coordinates*: for a given unary IND $R[A] \subseteq S[B]$, its coordinates are a set of index pairs (i, j) such that $r_i[A] = s_j[B]$. An ϵ term could be used for comparison within a range of tolerance, but it would be hard to infer an acceptable ϵ value without prior knowledge of the domain.

In any case, the reference benchmark shows that MIND, FIND2 and ZIGZAG have a comparable run-time, sometimes even faster, than BINDER and MIND2. While FAIDA is the fastest alternative, its validation strategy requires computing hashes over the attributes and their combinations, which is, again, inapplicable in our case.

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.

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

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 exists if every combination of values in X appears in Y [9], [12].

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 [17].

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 INDs 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

$$H_0 : P(R[X]) = P(S[Y]) \quad (1)$$

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

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

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].

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 of MIND, ZIGZAG and FIND2 for INDs search is a property that allows the pruning of the search space as illustrated in figure 2.

Let $I = R[X] \stackrel{d}{=} S[Y]$. A dataset d satisfies I iff a statistical test fails to reject $H_0 : P(R[X]) = P(S[Y])$. 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})¹
- 2) If $d \models I_2$, then $d \not\models I_1$ with a probability α (Rejecting H_{0_1} does not imply the rejection of H_{0_2})

This property is similar to the one proposed for INDs [12], 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 [14].

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 need to 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:

$$|E'| \geq \gamma \cdot \binom{|V'|}{k} \quad (2)$$

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

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

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

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, which is sensitive to both position and shape [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 $A \stackrel{d}{=} B$. 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).

To avoid quadratic complexity comparing all attributes from a relation R with all attributes of relation S , we propose an interval tree, meaning that only overlapping distributions are compared. 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, unsurprisingly, 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 solutions are not strongly dependent on discrete types: MIND, ZIGZAG and FIND2.

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.

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

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.

However, the search algorithm of FIND2, however, is capable of finding maximal EDDs with fewer tests. This is thanks to its search based on clique finding: the set of valid n -EDDs are represented as a n -uniform hypergraph where nodes are valid unary EDDs, and the edges join the unary EDDs that form a valid n -EDD. In such graphs, cliques are potential higher-arity EDD.

As an input, FIND2 needs a graph generated by another algorithm, such as MIND. For example, figure 3 shows a 2-hypergraph containing a set of *validated* (H_0 was not rejected) 2-EDDs between two relations. This example shows that, given the statistical nature of data and checks, there are quite a few false positives (bound by the statistical power of the test), and a handful of false negatives, or missing edges (bound by the significance level).

Precisely this combination of false positives and false negatives 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.

Fig. 3. When distributions look similar, graphs will probably include false positive spurious nodes and edges (light grey, dotted) and false negatives (missing edges between dark grey nodes). This is a real example coming from one of the datasets used in our experiments (Keel Mortgage/Treasury).

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

To solve this issue, we propose an algorithm based on quasi-cliques, composed of two stages:

Finding quasi-cliques seeds Some initial experiments with FIND2 showed that as-is, the algorithm was able to find relatively high arity EDDs regardless of the missing edges. It makes sense since, generally, a quasi-clique contains smaller but denser sub-graphs [23] and a clique is denser than a quasi-clique.

Furthermore, for each edge e , FIND2 generates a clique candidate by searching all nodes connected to all nodes in e . This candidate is either a clique or a union of all cliques that contain e , which may be a quasi-clique.

Fig. 4. Example of a possible spurious candidate generated when the only constraint is the edge ratio ($\gamma = 0.74$). The quasi-clique contains 301 edges out of 406 possible ones, but several vertices have a low degree. This is a real example, with 29 nodes even though there are only 20 true unary EDDs. It will be discarded—duplicated attributes are present—and it will trigger the generation of $\binom{29}{3} = 3,654$ 3-EDDs candidates.

Therefore, we use a modified version of FIND2 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. For instance, figure 4 shows a real example when only γ is used.

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 $G[V \setminus v]$ 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. Note that 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 $K' \setminus K = v^*(K')$, 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 oriented towards γ -quasi-cliques exclusively, 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 entire 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, due to the combinatorial nature of the search space, each mid-size quasi-clique can trigger the detection of hundreds of others. The overall run-time may suffer depending on the number of spurious edges, as we will show in section 5. Therefore, we consider this stage as optional.

4.3.1 Parameters

The expected number of missing edges can be directly derived from the significance level used for the tests. Or simply:

$$\gamma = 1 - \alpha$$

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. We refer again to figure 4 for a particularly bad case for $\lambda = 0$.

To solve this dilemma, we propose to use an adaptive value for λ based on the quasi-clique being checked: if we assume the quasi-clique is, indeed, the *true* clique with missing edges, 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 5 shows some examples of this distribution for a quasi-clique with 16 nodes and the critical value for a one-tail test with a significance level of 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 instead that the set of edges connecting this particular node are probably 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.

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

Fig. 5. 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 bar corresponds to the one-tail test with $\alpha = 0.05$. If the degree of a node were lower than this value, we would consider the node to be a spurious inclusion.

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

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 when the number of spurious edges is high.

5 EXPERIMENTS

We have implemented in Python the original algorithm FIND2 and the improved version proposed in this paper. 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.

5.1 Experimental design

We have performed two different sets of experiments: one exclusively benchmarks the search algorithms, 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

TABLE 1

Set of measurements taken for the quasi-clique finding problem

<i>Recovery ratio</i>	For each quasi-clique Q' found, we compute the Jaccard index [25] 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.
<i>Time</i>	Wall-clock time
<i>Timeouts</i>	How many runs exceeded the timeout

TABLE 2

Combination of parameters for the quasi-clique find problem. We chose the time limit based on the measured run-time from early test runs.

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

- 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. The statistical power—which influences the number of false positives—could improve. 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:

TABLE 3

Set of measurements taken from individual runs

<i>Time</i>	Wall-clock time, without accounting for the initialization stage, as this is shared
<i>Number of tests</i>	Time spent looking for quasi-cliques and validating the candidates. Tests can be potentially expensive, so we measure how many statistical tests are necessary.
<i>EDD count</i>	Without removing non-maximal EDDs
<i>Maximal EDD count</i>	Removing non-maximal EDDs
<i>Timeouts</i>	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.
<i>Highest arity</i>	The maximum EDD arity found

- 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 [16].
- 3) A random sample is taken from both relations (it defaults to 200)
- 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 mainly consider:

- The higher the match ratio, the better: the highest arity EDD is potentially the most interesting and

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

<i>Match ratio</i>	It is a ratio between the maximum arity of the maximal quasi-clique found and the <i>true</i> maximum EDD possible to find on each separate run (since some true unary EDDs may be falsely rejected). This <i>truth</i> is solely based on attribute names, so the algorithms can, and do, find higher arity EDDs when the values are taken into account. We consider this a proof of success: the metadata would not have sufficed to capture this trait.
<i>Accuracy</i>	Measured as the number of total returned EDDs, divided by the number of statistical tests executed. A 1 ratio (best) would imply that every candidate quasi-clique was accepted by the statistical test, while a 0 ratio (worst) would imply that all candidate quasi-cliques were rejected. This value can also be affected by the power of the statistical test as a function of dimensionality: i.e. since kNN has relatively low power with only two dimensions, the initial graph will be very dense, and many quasi-cliques found. However, kNN will easily reject equality of distribution at higher dimensions.

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. As an example, if we fail to validate a 12-EDD, we may still find 10 out of the 12 possible 11-EDDs that generalize it. 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.

Datasets

TABLE 5

Summary of the datasets used for validation. The two values under *By Value* represent the total number of positives accepted by the statistical test (from all possible paired permutations) on the right and the number of unique columns from the first dataset with at least one match on the left.

Dataset	Rows	Columns	Attribute matches	
			Name	Value
Mortgage/Treasury	1k + 1k	16 + 16	16	26 / 16
Ailerons/Elevators	14k + 17k	41 + 19	10	44 / 35
DC2	198k + 193k	39 + 33	39	35 / 22
AFDS	172 × 4	8 × 4	8	n/a

To test the algorithms, we have run them over two pairs of relations from the KEEL regression datasets [28], the training and test catalog from the *Euclid photometric-redshift challenge* [29], and a set of sensor measurements from an

aircraft fuel distribution system [30]. Some statistics about these datasets are summarized in table 5.

Mortgage / Treasury

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

Ailerons /Elevators

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

DC2

The datasets from this challenge come 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 attributes but for different sources. The features are flux and shape measurements. A naive one-to-one schema matching will easily mistake these attributes for small sample sizes. In contrast, the statistical tests will quickly reject their similarity for bigger samples due to small fluctuations (i.e., different positions on the sensor or the sky). 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)

This dataset 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, proving that our solution can perform well even with multiple inputs.

The two pairs from Keel (i.e. Mortgage/Treasury and Ailerons/Elevators) were found empirically running over the whole Keel dataset initial versions of the algorithms described in this paper, which, by itself, proves their capabilities.

5.2 Environment

The tests were run on a Slurm [31] 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 Slurm 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.

For the real dataset tests, we submitted Slurm 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.

Fig. 6. Recovery ratio and run-times for cliques on uniform 2-hypergraphs for different ratios of spurious edges (β). We show the estimated means with error bars corresponding to their estimated standard deviation. We display the timeout with a bar plot.

5.3 Results

In this section, we will summarize the results from our experimental setup. We will go on to 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 of means a reasonable assessment.

Influence of spurious edges

We show in figure 6 the performance of the algorithms for 2-hypergraphs and different ratio of spurious edges. The exponential worst-case complexity becomes more apparent the more connected nodes there are. This is either due to the clique size or the number of spurious edges. FIND2 is the most affected, but at some point, PRESQ performance also degrades significantly.

For 3-hypergraphs (figure 7) we can see how increasing the rank k of the graph worsens the performance given that the number of edges is combinatorial over N choose k . PRESQ manages to find quasi-cliques within the allocated time-window for higher values of β and N , but eventually also fails to finish on time.

For 4-hypergraphs (figure 8), the performance degrades so quickly with the number of edges that all the algorithms failed to find any clique of $n = 30$. For $n = 20$ only PRESQ manages to find in time for $\beta \leq 0.6$.

These results confirm that spurious edges influence the *run-time* of these algorithms very negatively [32].

Influence of missing edges

Fig. 7. Recovery ratio and run-times for cliques on uniform 3-hypergraphs for different ratios of spurious edges (β).

Fig. 8. Recovery ratio and run-times for cliques on uniform 4-hypergraphs for different ratios of spurious edges (β).

For 2-hypergraphs (figure 9), FIND2 only manages to start giving results for bigger cliques if enough edges are missing. Otherwise, it runs with its worst complexity. PRESQ seed stage begins close to the original clique, while the growing stage pushes the result closer. For $k = 2$, our proposal is similar to existing quasi-clique finding algorithms [23], [24], so these results are to be expected.

For 3-hypergraphs (figure 10), our tests show 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.

For 4-hypergraphs (figure 11), the inverse correlation between the number of missing edges and run-time is more visible. The found quasi-cliques are close to the original clique when the growing stage is enabled.

Influence of correlated ratios

Fig. 9. Recovery ratio and run-times for cliques on uniform 2-hypergraphs for different ratios of missing edges (α).

Fig. 10. Recovery ratio and run-times for cliques on uniform 3-hypergraphs for different ratios of missing edges (α).

Fig. 11. Recovery ratio and run-times for cliques on uniform 4-hypergraphs for different ratios of missing edges (α).

Fig. 12. Recovery ratio and run-times for cliques of size 20 on uniform (2, 3, 4)-hypergraphs.

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 beta and alpha 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 12 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 makes sense since there is no correlation between missing edges.

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.

We adopt instead the following method [33]:

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

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

In other words, simply the percent difference of the sample means. The distribution of $\hat{\phi}$ can be finally estimated using bootstrapping [34]. In this manner, we obtain the estimated population mean and standard deviation. Finally, we compute the 95% confidence interval $\hat{\mu}_{\hat{\phi}} \pm 1.96\hat{\sigma}_{\hat{\phi}}$

Figures 13, 14 and 15 show this confidence interval for match ratio, unique EDD, number of tests and wall time (columns) against significance levels 0.05, 0.10 and 0.15 (rows).

The DC2 case (figure 15) 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 — and many spurious — edges, which impacts the performance considerably. This is a known issue of FIND2 [32]. 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, and reducing considerably the run-time. In this case the *growing* stage can compete in terms of run-time with FIND2, even improving both the arity of the EDDs found and their number.

Figures 16 and 17 show the same metrics when an 3-hypergraph is given as input.

Figure 18 shows the matrix with the pairwise max arity found on the AFDS dataset. When comparing the maximum EDD arity found per pair of files, it is visible that 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” [30]. We consider this another proof of the utility of the proposed techniques.

Fig. 13. Results from the runs over the Keel Mortgage vs Treasury datasets. On average, PRESQ can recover more EDD with similar, or better, run-times. Increasing Δ can reduce the computation needs. The growing stage allows finding $> 6x$ the number of EDDs, at the cost of higher run-times.

Tables 6 and 7 summarize the overall results for a case where the initial significance level is 0.1 since this value seems to give the best efficacy/efficiency ratio. Note that, for time, match ratio and number of unique values, we provide the first and third quartiles instead of a mean, because their distributions are far from normal. PRESQ(G) identifies PRESQ with the optional growing stage.

6 DISCUSSION

Identifying shared attributes between multiple scientific datasets is an interesting problem. It combines the challenging nature of algorithms devised to find Inclusion Dependencies, an NP-hard problem [10], with the unavoidable

Fig. 14. Results from the runs over the Keel Elevators vs Ailerons datasets. In this case, the gains in terms of quasi-clique (and EDD) recovery of PRESQ is more marked than for the Mortgage test case. Total run-time (without the growing stage) is consistently lower. The *growing* stage is similarly able to find more valid EDDs.

Fig. 15. Results from the runs over the DC2 Train vs Test datasets.

Fig. 16. Results from the runs over the Keel Mortgage vs Treasury datasets, starting with a 3-uniform hypergraph. Run-times are consistently lower, and the number of tests are greatly reduced.

Fig. 17. Results from the runs over the Keel Elevators vs Ailerons datasets, starting with a 3-uniform hypergraph. PRESQ shows a reduction of run-time and number of tests for $\alpha = 0.05$.

TABLE 6

Summary of run-time, matching ratio (based on name), and number of maximal quasi-cliques found. The initial significance level is $\alpha = 0.1$, and the initial arity $k = 2$.

Mortgage vs Treasury												
		Λ	γ	Time (s)		Match		Unique		Prec.	Timeouts	
				Q1	Q3	Q1	Q3	Q1	Q3			
		FIND2		0.44	0.64	0.75	1.00	12	21	0.99	0.0%	
		PRESQ	0.00	0.9	136.43	510.87	1.00	1.00	11	15	0.03	0.0%
		PRESQ	0.05	0.0	0.84	9.65	0.89	0.94	12	18	0.52	0.0%
		PRESQ	0.05	0.9	0.76	10.61	0.88	1.00	11	17	0.57	0.0%
		PRESQ	0.10	0.9	0.73	1.99	0.84	1.00	11	17	0.75	0.0%
		PRESQ(G)	0.05	0.9	47.10	247.39	0.88	1.00	125	262	0.22	0.0%
Ailerons vs Elevators												
		FIND2			5.63	48.41	0.78	1.00	142	291	0.98	0.0%
		PRESQ	0.00	0.9	11.35	37.13	1.00	1.22	91	184	0.24	0.0%
		PRESQ	0.05	0.0	18.54	61.79	1.08	1.15	66	220	0.13	0.0%
		PRESQ	0.05	0.9	7.51	20.12	1.00	1.11	86	198	0.35	0.0%
		PRESQ	0.10	0.9	6.83	22.40	1.00	1.11	89	205	0.41	0.0%
		PRESQ(G)	0.05	0.9	57.86	674.26	1.11	1.25	321	1062	0.22	0.0%
DC2												
		FIND2			74.94	805.71	0.60	0.71	73	150	0.90	34.0%
		PRESQ	0.00	0.9	254.18	254.18	0.70	0.70	53	53	0.01	85.7%
		PRESQ	0.05	0.0	33.47	209.96	0.81	0.94	43	103	0.11	9.1%
		PRESQ	0.05	0.9	25.57	214.27	0.76	0.89	46	113	0.14	13.2%
		PRESQ	0.10	0.9	18.61	144.98	0.76	0.87	42	98	0.18	23.1%
		PRESQ(G)	0.00	0.9	1819.06	1819.10	0.78	0.78	109	109	0.005	83.3%
		PRESQ(G)	0.05	0.0								100.0%
		PRESQ(G)	0.05	0.9	458.26	1881.02	0.81	0.93	518	798	0.23	50.0%
		PRESQ(G)	0.10	0.9	285.50	1320.01	0.79	0.89	372	613	0.21	57.1%

Fig. 18. Pairwise max arity found on the AFDS dataset

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 [11]. 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 [32]. 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.

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

TABLE 7

Summary of run-time, matching ratio (based on name), and number of maximal quasi-cliques found. The initial significance level is $\alpha = 0.1$, and the initial arity is $k = 3$.

<i>Mortgage vs Treasury</i>									
	Λ	γ	Time (s)		Match		Unique		Prec.
			Q1	Q3	Q1	Q3	Q1	Q3	
FIND2			10.45	19.28	0.81	1.00	72	147	0.99
PRESQ	0.05	0.9	9.60	19.21	0.81	1.00	73	133	0.98
PRESQ	0.10	0.9	9.58	20.47	0.81	1.00	73	137	0.99
PRESQ(G)	0.05	0.9	42.88	369.47	0.81	1.00	202	403	0.98

<i>Ailerons vs Elevators</i>									
FIND2			51.27	1129.27	0.89	1.00	610	820	1.00
PRESQ	0.05	0.9	27.27	74.81	1.00	1.13	393	556	0.97
PRESQ	0.10	0.9	27.73	81.45	1.00	1.13	412	571	0.97
PRESQ(G)	0.05	0.9	160.69	524.19	1.11	1.21	742	1181	0.93
PRESQ(G)	0.10	0.9	174.15	514.59	1.11	1.17	753	1184	0.95

<i>DC2</i>									
FIND2									100.0%
PRESQ	0.05	0.9	41.25	139.87	0.73	0.82	207	244	0.80
PRESQ	0.10	0.9	265.76	866.40	0.70	0.77	247	301	0.72
PRESQ(G)	0.05	0.9	201.96	540.78	0.72	0.81	360	570	0.82
PRESQ(G)	0.10	0.9	449.15	912.88	0.70	0.76	393	609	0.84

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 in the 2-graph.

Finally, 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 its worst possible complexity (exponential). 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, as described in figure 4. In general, a middle ground parametrization based on both γ and Λ is more predictable.

7 THREATS TO VALIDITY

Internal validity At first, 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 (shared) initial conditions and the number of measurements.

Looking at the results summarized in tables 6 and 7, 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 [14] proposes a heuristic 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 from them. Since the relative differences would remain similar, we are confident that the

gains come from the underlying algorithm rather than its implementation.

External validity The experiments have been run over three different datasets of diverse nature and from two 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 statistical test of choice. In any case, this issue was also recognized on the original FIND2 proposal [14], and it is not a problem of the (quasi)clique finding algorithm per se, but of the candidate validation step instead.

For the sake of transparency, all the necessary code to reproduce our results, together with the raw results, and notebooks used to generate the figures and tables, are publicly available².

8 CONCLUSIONS AND FUTURE WORK

Finding sets of equally-distributed dependencies between scientific 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 INDs 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 belonging to the clique, our algorithm can successfully identify these sets of attributes without requiring any knowledge about their metadata.

In general, it would seem that comprehensive approaches will be needed to find very high arity EDDs, given the complexity of the INDs/EDDs 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 [35].
- *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.
- *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] T. Hey, S. Tansley, and K. Tolle, “Jim Gray on eScience: A Transformed Scientific Method,” in *The fourth paradigm: Data-intensive scientific discovery*. Microsoft Research, Oct. 2009, pp. xvii–xxxi. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/>
- [2] 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/
- [3] 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/>
- [4] 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>
- [5] 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/>
- [6] 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>
- [7] 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
- [8] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, Sep. 1995. [Online]. Available: <https://doi.org/10.1007/BF00994018>
- [9] 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>
- [10] 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>
- [11] 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*, ser. CIKM ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 219–228. [Online]. Available: <https://doi.org/10.1145/3357384.3357916>
- [12] 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
- [13] 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>

2. <https://github.com/ayllon/MatchBox/>

- [14] 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>
- [15] T. Papenbrock, S. Kruse, J.-A. Quiané-Ruiz, and F. Naumann, "Divide & conquer-based inclusion dependency discovery," *Proceedings of the VLDB Endowment*, vol. 8, pp. 774–785, 2015. [Online]. Available: <https://dl.acm.org/doi/10.14778/2752939.2752946>
- [16] 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)*, B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, T. Härder, O. Kopp, and M. Wieland, Eds. Gesellschaft für Informatik, Bonn, 2017, pp. 207–226. [Online]. Available: <https://dl.gi.de/handle/20.500.12116/629>
- [17] B. Dawson, "Comparing floating point numbers," Cygnus Software, Tech. Rep., 2008. [Online]. Available: <https://staff.polito.it/claudio.fornaro/CorsoINF/Other%20interesting%20lectures/Comparing%20Floating%20Point%20Numbers.pdf>
- [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/002200084900758>
- [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. Doderó, "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/10.1007/978-3-540-92695-5_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] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912. [Online]. Available: <https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x>
- [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] 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.
- [29] 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>
- [30] 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>
- [31] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple linux utility for resource management," in *Job scheduling strategies for parallel processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60. [Online]. Available: https://link.springer.com/chapter/10.1007/10968987_3
- [32] 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/10.1007/978-3-540-31161-7_7
- [33] 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/10.1007/978-3-319-9396-7_7
- [34] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," in *Breakthroughs in Statistics: Methodology and Distribution*, S. Kotz and N. L. Johnson, Eds. New York, NY: Springer New York, 1992, pp. 569–593. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_41
- [35] 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>