FreSCo: Mining Frequent Patterns in Simplicial Complexes

Giulia Preti
ISI Foundation, Italy
Turin, Italy
giulia.preti@isi.it

Gianmarco De Francisci Morales
ISI Foundation
Turin, Italy
gdfm@acm.org

Francesco Bonchi
ISI Foundation, Italy
Eurecat, Spain
francesco.bonchi@isi.it

ABSTRACT
Simplicial complexes are a generalization of graphs that model higher-order relations. In this paper, we introduce simplicial patterns—that we call simplicets—and generalize the task of frequent pattern mining from the realm of graphs to that of simplicial complexes. Our task is particularly challenging due to the enormous search space and the need for higher-order isomorphism. We show that finding the occurrences of simplicets in a complex can be reduced to a bipartite graph isomorphism problem, in linear time and at most quadratic space. We then propose an anti-monotonic frequency measure that allows us to start the exploration from small simplicets and stop expanding a simplet as soon as its frequency falls below the minimum frequency threshold. Equipped with these ideas and a clever data structure, we develop a memory-conscious algorithm that, by carefully exploiting the relationships among the simplices in the complex and among the simplicets, achieves efficiency and scalability for our complex mining task. Our algorithm, FreSCo, comes in two flavors: it can compute the exact frequency of the simplicets or, more quickly, it can determine whether a simplet is frequent, without having to compute the exact frequency. Experimental results prove the ability of FreSCo to mine frequent simplicets in complexes of various size and dimension, and the significance of the simplicets with respect to the traditional graph patterns.

CCS CONCEPTS
• Mathematics of computing → Hypergraphs; Graph algorithms;
• Information systems → Data mining.

KEYWORDS
Frequent pattern mining, simplicial complex, higher order patterns.

ACM Reference Format:
Giulia Preti, Gianmarco De Francisci Morales, and Francesco Bonchi. FreSCo: Mining Frequent Patterns in Simplicial Complexes. https://doi.org/10.1145/3485447.3512191

1 INTRODUCTION
Frequent pattern mining in graph-structured data aims at finding structures that occur frequently in a given (set of) graph, under the assumption that frequency indicates relevance. This fundamental primitive has gained considerable attention thanks to the numerous applications such as fraud detection [37], network intrusion detection [22], trend discovery [44], Web usage mining [16, 23], link prediction for people-recommeder systems, community discovery, or predicting group activity on social networks [11, 13, 34]. As the search space of the mining task is intrinsically exponential, the measure of frequency must be selected carefully. Typically, anti-monotone measures—such that a pattern cannot be more frequent that any of its sub-patterns [14, 20]—are the choice, as they allow to exploit the so-called apriori property to prune the search space substantially, thus enabling efficient algorithms.

In this paper, we generalize the task of frequent pattern mining from graphs—the case in which entities are involved in pairwise interactions—to the case of higher-order relations. In fact, many interactions in the real-world occur among more than two entities at once [15, 50]. For example, multiple Web pages can be visited in the same session, multiple entities co-occur in the same Wikipedia page, scientific papers are usually written by teams of researchers, and molecules interact in groups. Simple graphs are not expressive enough to model such higher-order relations, because they cannot distinguish the case of, e.g., three entities co-occurring in pairs in three different Wikipedia pages, from the case where all three entities co-occur in the same page [10].

Hypergraphs [17] and simplicial complexes [7] are higher-order generalizations of simple graphs that allow such a distinction [48]. Hypergraphs generalize graphs by allowing hyperedges to connect more than two vertices. Conversely, a simplicial complex is a collection of polytopes such as triangles and tetrahedra, which are called simplices. Both these structures can be used to represent any higher-order relation [47] and their distinction is thin: simplicial complexes require the downward closure property, i.e., every substructure (also known as face) of a simplex contained in a complex \( K \) is also in \( K \). Using the example of three entities co-occurring in a Wikipedia page, the downward closure property models the fact that any pair of the three entities are also co-occurring in the page. Which paradigm is more appropriate depends on the application domain and the semantics we want to associate to a higher-order relation. In domains characterized by interactions that are "maximal" [51], e.g., in scientific collaborations (the authors of a paper are all co-authors) or gene activation pathways (largest group of collectively activated genes), simplicial complexes are more appropriate. Indeed, simplicial complexes have proven to be useful in analyzing the organization of the brain [31], characterizing the scientific collaborations [39], predicting new links between entities [9], and studying the mechanisms of social contagion [26].

Challenges and contributions. Given a simplicial complex, we define the problem of extracting all frequent simplicial patterns where a simplicial pattern is a subcomplex that we dub simplet (in analogy to the graphlet). The frequency measure we adopt is inspired by the MNI support measure [20] to ensure anti-monotonicity and efficiency in the computation. Interestingly, when the input complex encodes only pairwise interactions (i.e., when
it is a graph), our problem formulation reduces to the traditional frequent graph pattern mining. Frequent simplicial mining can find applications in complex summarization and classification, in characterizing the interactions in brains of patients suffering different diseases, in finding relevant functional dependencies in ecological systems, in determining the coactivation patterns in resting-state functional magnetic resonance imaging signals [30], in developing recommending systems [25], and in community detection [12].

Frequent pattern mining in simplicial complexes presents numerous challenges. First, the search space is extremely large, due to all the possible combinations of higher-order simplices that can constitute the simplex. Second, finding the occurrences of simplices in a complex entails determining if two simplices are isomorphic. While there exist many libraries for solving the graph isomorphism problem, little effort has been devoted to complex isomorphism [5]. We show that finding the occurrences of simplices in a complex can be reduced to a bipartite graph isomorphism problem, in linear time and at most quadratic space. We then propose an anti-monotonic downward closure property. The seminal paper on frequent sub-hypergraph mining [25] proves that the problem is NP-hard, but can be solved in incremental polynomial time for node and edge injective hypergraphs. This work, however, finds frequent sub-hypergraphs in hypergraph databases, and hence cannot mine frequent structures in single hypergraphs, nor in single complexes. The reason is that, in the database setting, the frequency of a sub-hypergraph is defined as the number of hypergraphs in the database that contain a given sub-hypergraph, and hence, in a single hypergraph each sub-hypergraph would have frequency equal to either 1 or 0.

Frequent hypergraph mining. Frequent pattern mining has been studied also in the context of hypergraphs, which are generalizations of graphs related to simplicial complexes. We recall that a complex is a special kind of hypergraph that satisfies the downward closure property. The seminal paper on frequent sub-hypergraph mining [25] proves that the problem is NP-hard, but can be solved in incremental polynomial time for node and edge injective hypergraphs. This work, however, finds frequent sub-hypergraphs in hypergraph databases, and hence cannot mine frequent structures in single hypergraphs, nor in single complexes. The reason is that, in the database setting, the frequency of a sub-hypergraph is defined as the number of hypergraphs in the database that contain a given sub-hypergraph, and hence, in a single hypergraph each sub-hypergraph would have frequency equal to either 1 or 0.

Simplicial complexes analysis. Simplicial complexes have been adopted to model higher-order relations and solve several interesting problems [8, 45]. By observing that the interactions between subsets of a group of users in a social network increase the likelihood that the members of the group will be pairwise connected in the future, Benson et al. [9] tackled link prediction via simplicial closure. Similarly, Eswaran et al. [19] addressed the semi-supervised learning task of label propagation in partially-labeled graphs. Iaconini et al. [26] adopted simplicial complexes to describe social contagion and diffusion phenomena, Horak et al. [24] to characterize networks by means of their topological features, Serrano and Gómez [46] to define centrality measures, Preti et al. [41] addressed the problem of truss decomposition of simplicial complexes. Finally, substantial work has been done by using simplicial complexes to study the brain’s functional and structural organization [31, 40].

3 PROBLEM DEFINITION

We next introduce the concepts and the notation needed for our problem statement. After each definition, we use an example referring to Figure 1 to clarify the introduced concept. Let \( V \) be a set of vertices. Simplices are higher-order generalizations of edges.

Definition 1 (\( q \)-simplex, face, joist). A \( q \)-dimensional simplex \( \sigma^q \) is a set of \( q + 1 \) interconnected elements of \( V \), i.e., \( \sigma^q = \{v_0, \ldots, v_q\} \subset V \). A face of \( \sigma^q \) of dimension \( r \) is a \( r \)-simplex \( \sigma^r \subset \sigma^q \). The joist of \( \sigma^q \) is the set \( J_{\sigma^q} \) of all its \( q + 1 \) cofaces (faces of dimension \( q - 1 \)) [41].

Example 1. The dark triangle \([2, 3, 7]\) in Figure 1 (left) is a 2-simplex, all its proper subsets of dimension 0 or 1 are its faces, while
the set of 1-simplices \([2,3], [2,7], [3,7]\) constitutes its joist. A 0-simplic is a relation involving a single vertex, and hence is equivalent to a vertex in a graph, while a 1-simplic is comparable to an edge, as they both relate two vertices. In contrast, a q-simplic with \(q > 1\) is a polyadic relation that cannot be expressed in graph terminology.

**Definition 2 (Complex).** A simplicial complex \(K\) is a set of simplices such that all the faces of the simplices in \(K\) are also in \(K\). The dimension of \(K\) is the dimension of its largest simplex, while its vertex set \(V_K\) is the set of distinct vertices in its simplices. The n-skeleton of \(K\) is the complex composed by the subset of simplices of dimensions \(q \leq n\).

**Example 2.** The complex in Figure 1 (left) has dimension 3, given by its 3-simplex \([1,2,3,7]\) and the order 2 simplices \([1,2,3], [2,3,7], [1,2,7]\). The 3-skeleton of the complex \(K\) corresponds to the undirected simple graph induced by the 1-simplices of \(K\). When the dimension of \(K\) is 1, the complex is a graph.

A subcomplex \(S_K \subset K\) is a downward-closed improper subset of simplices in \(K\) (the skeleton is also a subcomplex). We say that two simplices are connected if they share a face, and we call a connected subcomplex a simplic.

**Definition 3 (Simplet).** A simplet \(P\) is a tuple \((V_P, C_P)\), where \(C_P\) is a set of simplices such that (i) \(\sigma \land \varnothing \in C_P\) only if all its faces are in \(C_P\), and (ii) for each pair of vertices \(u, v \in V_P = \bigcup_{\sigma \in C_P} \{v \mid v \in \sigma\}\), there exists a sequence \([u, u_0, u_1, \ldots, u_q, v]\) of 1-simplics that connects \(u\) and \(v\). The dimension of the simplic \(\text{dim}(P)\) is the dimension of the largest simplex in \(C_P\).

**Example 3.** The structure in Figure 1 (right) is a simplet with vertex set \([A, B, C, D, E]\) and simplet set \([\{A, B, C\}, \{C, D\}, \{C, E\}, \{D, E\}\].

**Definition 4 (Isomorphism and Automorphism).** We say that a complex \(K_1\) is isomorphic to another complex \(K_2\), denoted as \(K_1 \cong K_2\), if there exists a bijection \(\phi: V_{K_1} \to V_{K_2}\) that preserves the relations between the simplices, i.e., for each \(\sigma = [v_0, \ldots, v_q] \in K_1\) it holds that \(\sigma' = [\phi(v_0), \ldots, \phi(v_q)] \in K_2\). A bijection from a complex \(K\) to itself is called an automorphism, and the set of all the automorphisms of \(K\) is denoted as \(\text{Auto}(K)\). When the dimension of \(K\) is 1, these definitions are consistent with graph isomorphism and automorphism.

**Definition 5 (Occurrence).** The subcomplexes of \(K\) isomorphic to \(P\) are called occurrences of \(P\) in \(K\) (a.k.a. instances or embeddings).

**Example 4.** The table in Figure 1 reports the occurrences of the simplic above the table, with the corresponding complex-vertex-to-simplic-vertex assignments. This simplic is characterized by one open and one closed triangle connected by a vertex. For example, the subcomplex \(\phi_0 = ([1, 2, 3, 4, 5], [1, 2, 3], [2, 4], [2, 5], [4, 5])\) is mapped to the simplic by the function \(\phi(1) = A, \phi(2) = C, \phi(3) = B, \phi(4) = D, \phi(5) = E\), while no subcomplex including the edges of the triangle \([2, 6, 7]\) can be isomorphic to the simplic, because \([2, 6, 7]\) is an empty triangle that share an edge with any closed triangle in the complex.

We are interested in finding simplices that occur frequently in the complex, with the assumption that frequency captures an important characteristic of the data-generating process. However, finding the occurrences of a simplic in a complex is a challenging task that requires the evaluation of isomorphisms between candidate subcomplexes and the simplic. Simplicial complexes are special kinds of hypergraphs, and the **SUB-HYPERGRAPH ISOMORPHISM problem** (SHI) is NP-hard by reduction from **SUBGRAPH ISOMORPHISM** (SGI) [32]. Given the hardness of the task, it is pivotal to design algorithms able to significantly prune the search space. A pruning strategy extensively used in graph pattern mining is based on the **anti-monotonicity** property, which states that the frequency of a super-pattern is smaller or equal to the frequency of its sub-patterns. Thanks to this property, the mining algorithm can follow a pattern-growth approach that starts from small structures and expands only the frequent ones. However, defining the support as the number of occurrences does not ensure anti-monotonicity [14]. Therefore, alternative support measures have been introduced in the literature, with MINI (Minimum Node-based Image) being the most used one, as it is more efficient to compute than the others [20].

To ensure anti-monotonicity and backward compatibility, we define a support measure \(SUP\) that is a proper generalization of the MINI measure to simplicial complexes:

**Definition 6 (SUP).** Let the set of support \(\mu_P(v)\) of \(v \in V_P\) of a simplic \(P\) be the set of vertices in \(K\) that are mapped to \(v\) by some isomorphism \(\phi\), i.e., \(\mu_P(v) = \{u \in K \mid \exists \phi \text{ s.t. } \phi(u) = v\}\). Then, \(SUP(P, K) = \min_{v \in V_P} \mu_P(v)\).

**Property 1.** SUP is anti-monotone.

The proof of anti-monotonicity of SUP is in Appendix A.2.

We say that a simplic is frequent in \(K\) if \(SUP(P, K) \geq r\), where \(r\) is a user-given minimum frequency threshold. Then, the problem addressed in this paper, i.e., frequent simplic mining in simplicial complexes, can be formalized as follows:

**Problem 1 (Frequent Simplic Mining).** Given a simplicial complex \(K\), a maximum size \(s^*\), a minimum dimension \(d^*\), and minimum frequency threshold \(\tau\), find all the simplics \(P\) such that \(|V_P| \leq s^*\), \(\text{dim}(P) \geq d^*\), and \(SUP(P, K) \geq \tau\).

Problem 1 is a decision problem: determining which simplics are frequent in a complex. The candidate simplics are all the possible simplics up to a given size and with minimum dimension. The minimum dimension leads to more compact but informative results, allowing the user to focus only on the most interesting higher-order structures in the complex. The maximum size limits the search space and hence controls the complexity of the mining task. This design choice is usually adopted also in frequent graph pattern mining [27], given the hardness of the task.

**Observation 1.** Problem 1 is a proper generalization of traditional graph pattern mining. That is, when the simplicial complex \(K\) is a graph, Problem 1 with \(d^* = 1\) corresponds exactly to graph pattern mining. In fact, a 1-dimensional simplic can be mapped, without loss, to a graph pattern, by mapping each 1-simplic to an edge and each 0-simplic to a vertex.

**4 Generation and Canonicality**

Before delving into our algorithm, we give an overview of how it generates the simplics to examine, and how it avoids examining the same simplic twice.

The simplic search space is a lattice, partially ordered by complex inclusion, denoted as \(\leq_C\), and defined as follows:
which has the same dimension as the wedge (it contains only $A$ directed edge connects a node

1-1

different sequences of

different pairs of simplices

$\text{dim}(P') := (V_p \cup \{u\}, C_p \cup \{\{u, v\}\})$, and hence may increase the dimension of $P$.1

Figure 2 illustrates a portion of the lattice. The applications of weaken are denoted by green solid arrows, while the applications of inflate are denoted by blue dotted arrows. At every use of weaken, we add a vertex to the simplex; while at every use of inflate, we strengthen the relations between the simplet vertices. In the latter case, the dimension of the simplet does not necessarily increase. For example, the first use of inflate generates the open triangle, which has the same dimension as the wedge (it contains only 1-simplices), while the second use of inflate generates the closed triangle, which, instead, has dimension 2.

The following theorem proves that the recursive application of the two expansion rules generates all the possible simplets. The proof can be found in Appendix A.2.

Theorem 4.1 (Completeness). Starting from a 0-simplex, the recursive application of weaken and inflate can generate any simplet.

Several nodes may precede the same node $A$, due to the existence of different pairs of simplices ($\sigma, \sigma'$) such that $\sigma \subset \sigma'$. In this case, node $A$ can be reached from the root via different paths. This situation corresponds to the existence of different sequences of applications of the two expansion rules that result in the same simplet. To avoid generating (and examining) the same simplet multiple times, we need to detect if a simplet has already been generated, i.e., we need to determine if two simplets are equal. In

1Note that a pair of vertices $(u, v)$ in $P$ that are not faces of a common $q$-simplex in $C_p$ is the open joint $J_{uv} = \{\{u\}, \{v\}\}$ of the 1-simplex $\sigma = [u, v]$.

Definition 7 (Complex Inclusion). Given two distinct simplets $P_1$ and $P_2$, we say that $P_1 \leq C P_2$ iff for each simplex $\sigma \in C_p$, there exists $\sigma' \in C_{p'}$ such that $\sigma \subseteq \sigma'$.

Each node in the lattice corresponds to a simplet, with the root being the simplet that consists of a 0-simplex $\{r\}$, i.e., $O := (r, \{r\})$. A directed edge connects a node $A$ to a node $B$ iff $A \leq C B$, and so we say that $A$ precedes $B$ in the lattice. The lattice can be generated by recursively applying the following expansion rules to a simplet $P$, initially set equal to the root:

- **weaken**: insert a new 1-simplex $[u, v]$ in $C_p$, such that $v \in V_p$ and $u \notin V_p$. This rule creates the simplet $P' := (V_p \cup \{u\}, C_p \cup \{\{u, v\}\})$, and hence increases the size of the vertex set $V_p$;

- **inflate**: insert a new $q$-simplex $\sigma(q)$ in $C_p$, such that its joint $J_{\sigma(q)}$ is already contained in $C_p$. This rule creates the simplet $P' := (V_p \cup \{u\}, C_p \cup \{\sigma(q)\})$, and hence may increase the dimension of $P$.1

Equations 1 state that the simplets must have the same dimension and number of simplices; Equations 2 that the dimensions of the simplices in the simplets must be the same, and that the vertices of the simplets must be part of the same number of simplices; and, Equation 3 that the simplets must have the same 1-skeleton.

If all the conditions hold, we need to compare the canonical forms of the two simplets. Our solution to find those canonical forms is to reduce the problem to a graph canonization problem. We construct a polynomial-time reduction from the simplet to a bipartite graph, such that the left-hand side vertices are the vertices in the simplet, the right-hand side vertices are the simplices in the simplet, and an edge indicates the membership of a simplet vertex to a simplex. While for hypergraphs this reduction can exponentially increase the input size (the number of possible edges in a hypergraph with $n$ vertices is $2^n$), a reduction from a complex is far more feasible, thanks to the downward closure property of the complexes. In fact, for a complex with $n$ vertices, the number of vertices of the bipartite graph is upper-bounded by $n(n+1)/2$, and the number of edges is upper-bounded by $n(n-1)$. Therefore, we find the canonical form of a complex by first mapping the complex to a bipartite graph, and then finding the canonical form of the bipartite graph via a state-of-the-art library for graph isomorphism [29].

2Note that the two vertex degree sequences are sorted in increasing order.
5 ALGORITHM

FresCo (Frequent Simplets in a Complex, Algorithm 1) extracts the frequent simplets in a simplicial complex, given a maximum size $s'$ and a minimum dimension $d^*$. The algorithm exploits several heuristics to guide the search of frequent simplets. It starts from $P \leftarrow \{(v), \{(v, [u])\}\}$, and recursively expands each frequent simplet until it becomes infrequent (Procedure EXPAND).

For each candidate frequent simplet, FresCo searches for subcomplexes in the complex that are isomorphic to the simplet, and stores the corresponding mapping from complex vertices to simplet vertices (Procedure EXAMINE-DP). Once it has found enough matches, it computes the support of the simplet, and if it is larger than the minimum frequency threshold $\tau$, the simplet is further expanded. If needed, FresCo finds the canonical form of the expanded simplets to determine if they are redundant and should be discarded.

The algorithm exploits several heuristics to guide the search of the occurrences and hence reduce the time required to assess the simplet’s frequency. We next describe the algorithm that answers the decision problem introduced in Section 3. A variant of FresCo that finds the exact frequency of the simplets, at the cost of higher running time is detailed in Appendix A.1.

Simplet expansion. The computation starts from a simplet $O$ consisting of a single vertex $v$, whose support set $I_O(v)$ is the vertex set of the complex. The set of frequent simplets $F$ is initialized as the empty set. The main body of Algorithm 1 lies in Procedure EXPAND, which is called recursively to expand simplets $P$ proven to be frequent. If the number of vertices in $P$ is below the maximum size $s'$, the simplet is first expanded by adding a new vertex $u$ and connecting it to a vertex in $VP$ (widen). The algorithm applies inflate in the for loop at lines 12-15. For each joist $f^{(q)}$ of a simplet $\sigma^{(q)}$ not already in the simplet $P$, a new simplet $P'$ is instantiated by adding $\sigma^{(q)}$ to $CP$. For each expansion $P'$, the image sets of the vertices in $VP$ are initialized with $VP$. To identify the joists in the simplet we adopt a technique similar to the one in [41]: a dynamic inverted index that stores, for each face, all the simplices in the simplicial complex that share that face. Since all the simplices in a joist must share a face by definition, this index allows us to speed up the search of the joists. In particular, for each $q$-simplex $\sigma$ in the simplicial complex, we search for a subset $S$ of other simplices associated to the same face that share a vertex $w$ not in $\sigma$. If $|S| = q + 1$ and the simplex $\sigma \cup \{w\}$ does not exist in the simplet, $S \cup \{\sigma\}$ is a joist.

The same simplet $P'$ can be generated by applying different sequences of widen and inflate. Therefore, before adding $P'$ to $E$ (lines 11 and 15), we check if $P'$ has already been generated. We use a dictionary with key $(scp,scp)$ (i.e., simplicial dimension and simplicial degree sequence) to store the distinct simplets $P$ generated so far. Then, we perform the checks in Equations 1-3 with respect to each $P$ in the dictionary with the same key as $P'$. If all the checks are satisfied for some $P$, we find the canonical form of $P'$ and compare it to the canonical form of $P$. To speed up the computation of the canonical form, we employ a compact simplicial representation such that $CP$ includes only maximal simplices.

Once the algorithm has created all the expansions $P'$ of $P$, the image sets are computed by Procedure EXAMINE-DP in Algorithm 2 (Appendix A.1). Then, if the minimum size of the image sets of $P'$ is above the threshold $\tau$, $P'$ is further expanded. The simplet is added to the output set $F$ only if its dimension $dim(P')$ is greater than the minimum threshold $d^*$. Retaining only the simplets of higher dimension leads to a more compact output set, and thus simplifies its examination by the end user. The full set of frequent simplets can still be obtained by setting $d^* = 1$.

Frequency computation. In the following, we say that two vertices are neighbors if they belong to a common simplex, and denote with $\Gamma(u)$ the set of neighbors of a vertex $u$. Algorithm 2, whose pseudocode is in Appendix A.1, searches for assignments from vertices in the complex to vertices in $P$, such that the membership relations to the simplices in $CP$ are preserved.

The algorithm takes in input a parameter $\tau^*$ for the maximum amount of time that the algorithm can spend trying to find a complete valid assignment for a candidate $u$, and the parent simplicet of $P$, denoted with $SP$. The parent is the simplicet that we expanded to generate $P$. The threshold $\tau^*$ is needed to avoid spending too much time on problematic candidates. The algorithm looks for valid assignments among the vertices in the sets $UI$, initialized with the vertex set of $K$. We initialize the set of valid assignments $IPA(u)$ with the set of all the vertices of simplices having the same or more vertices than $P$, as all the simplices with $d$ vertices are sub-simplicets of simplices with at least $d$ vertices. The sets of non candidate complex vertices $SPNC$ are used to initialize the non-candidate sets $NC$ of $P$, as those vertices cannot be valid assignments for $P$ either.

The algorithm iterates over the vertices of $P$ examining only those for which it has not found enough assignments yet. When examining a vertex $v$, the set of candidates $Cands$ is initialized as the set $Vg$ minus $NC(u) \cup IPA(v)$, as these vertices cannot increase the size of $IPA(v)$, and hence do not help in determining if $P$ is frequent. To speed up the search of the valid assignments, $Cands$ is sorted so that we examine first the candidates that were valid assignments for the vertex $v$ of $SP$. For each candidate $u$ that has at least the same number of neighbors of $v$, we initialize a dictionary $M$ with $M[v] = u$. The dictionary, which stores the partial vertex assignment, is expanded by Procedure FINDMATCH, which searches for valid assignments for all the other vertices in the simplicet. If a complete assignment cannot be found, the vertex $u$ is added

---

Algorithm 1 FresCo

Require: Complex $K$, Frequency Threshold $\tau$, Max Size $s'$, Min Dim $d^*$
Ensure: Set of simplets $F$ with $|\I(P, K)| \geq \tau$
1. $O \leftarrow \{(v), \{(v, [u])\}\}$, $I_O(v) \leftarrow VP$
2. $F \leftarrow$ EXPAND($K, O, \tau, s', d^*$)
3. return $F$

4. function EXPAND($K, P, \tau, s', d^*$)
5. $E \leftarrow \emptyset$, $F \leftarrow \emptyset$
6. while $|VP| < s'$ do
7. for $v \in VP$ do
8. $u \leftarrow$ VERTEX()
9. $P' \leftarrow (VP \cup \{u\}, CP \cup \{[u, v]\})$
10. $IPA(u) \leftarrow VP$ for $w \in VP$
11. if $P'$ not yet generated then $E \leftarrow E \cup \{(P', IPA)\}$
12. for $f^{(q)}$ such that $\sigma^{(q)} \notin CP$ do
13. $P' \leftarrow (VP, CP \cup \{\sigma^{(q)}\})$
14. $IPA(u) \leftarrow VP$ for $w \in VP$
15. if $P'$ not yet generated then $E \leftarrow E \cup \{(P', IPA)\}$
16. for $P, UI \in E$ do
17. $IPA(u) \leftarrow$ EXAMINE($K, P, UI, \tau$), $sup \leftarrow$ min$\{ IPA(v) \}$
18. if $\sup \geq \tau$ then
19. if $dim(P) \geq d^*$ then $F \leftarrow F \cup \{P\}$
20. $F \leftarrow F \cup$ EXPAND($K, P, \tau, s', d^*$)
21. return $F$
to $NC(v)$. Moreover, two conditions allow us to early-stop the examination of $v$: when the remaining candidates are not enough to assess that the simplet is frequent, and when $|IP(o)| \geq \tau$.

Procedure FINDMATCH recursively visits all the not-yet-explored vertices, according to an order obtained by a d.f.s. starting from $v$, so that the recursive call visits neighbor vertices most of the times. When visiting vertex $x$, the algorithm first computes a restricted candidate set $Cands$, and then tries to find a valid assignment for $x$ among the complex vertices in $Cands$. To create the restricted candidate set, we first compute the intersection among the neighbors of the complex vertices associated to simplet vertices that are neighbors of $x$. Thus, we can avoid examining complex vertices that would not preserve the simplex membership relation. A vertex assignment is valid if Procedure SATISFIES CONSTRAINTS returns true. If a candidate $n$ is a valid vertex assignment, the assignment is added to $M$, and Procedure FINDMATCH is recursively called. For each simplex $\sigma$ in the simplex containing the vertex $x$, Procedure SATISFIES CONSTRAINTS checks whether the complex vertices already assigned to the vertices in $\sigma$ form, together with the candidate assignment $n$ of $x$, a simplex in the complex. If this is the case for each $\sigma$, the procedure returns true.

If a complete assignment starting from $u$ cannot be found in $t^*$ time, Procedure FINDMATCH returns the empty set. This way, Algorithm 2 knows that it must store $u$ in the set $resume$, which includes vertices that are reexamined later, if $|IP(o)| < \tau$.

At the end of Procedure FINDMATCH, $M$ contains an assignment found starting with $M[u] = n$. If the assignment is complete, the image sets $IP$ are updated with $M$, by exploiting the orbit information. Given a simplet $P$ and a vertex $v \in V_P$, the orbit $\Omega(o, P)$ of $v$ in $P$ is the subset of $V_P$ mapped to $v$ by exploiting any automorphism of $P$, i.e., $\Omega(o, P) = \{u \in V_P | \exists \phi \in \text{Auto}(P), \phi(u) = v\}$. By definition, vertices in the same orbit have the same image set. Therefore, we propagate the valid assignments in $M$ to vertices in the same orbit.

6 EXPERIMENTAL EVALUATION

FresScO code, implemented in Java 1.8, is available at https://github.com/lady-bluecopper/FresScO together with the 6 simplicial complexes used in the experiments (Table 2 in Appendix A.3). Experiments are run on a 24-Core (1.90 GHz) Intel(R) Xeon(R) E5-2420 with 126GB of RAM, limiting the memory to 110GB, and using all the cores. We set $s^* = 5$ and $d^* = 1$ for all the experiments.

Scalability. We next evaluate the scalability of FresScO (decision version). Figure 3 shows the running time of FresScO for 5 increasing frequency thresholds. For each threshold, the algorithm finds roughly the same number of simplets in each dataset. The running time follows a concave curve: at higher frequencies there are few candidate frequent simplets, and therefore the algorithm terminates earlier; while at lower frequencies, even though the number of candidate frequent simplets is much larger, the presence of higher-order simplices in the complex simplifies the task of searching for occurrences. Moreover, by comparing the running times in DBLP with those in Enron, we can see that a $30\times$ increase in dataset size leads to a time increase of only one order of magnitude.

Comparison with graph pattern mining. To assess the importance of mining higher-order structures when the input is a simplicial complex, Figure 4 presents an overview of the frequent simplets found at different frequency levels in Zebra (left) and DBLP (middle). These charts illustrate the number of frequent simplets for each dimension. If we modeled the datasets as simple graphs, we would find only the simplets denoted in dark green, corresponding to 1-dimensional structures which can be mapped to graph edges without loss. By looking for frequent simplets in the datasets modeled as simplicial complexes, we are able to detect a large number of higher-order structures. In particular, in orange we can see the number of frequent simplets consisting of simplices with dimension at most 2 (i.e., closed triangles), while the light blue bar indicates the number of simplets including simplices with dimension at most 3 (i.e., closed tetrahedra). Some of the simplets of dimension 3 persist over multiple frequency thresholds, which indicates that these higher-order structures are not trivially frequent. All the simplets with dimension larger than 1 can be neither modeled nor mined by adopting traditional graph-based approaches. In particular, these algorithms cannot discriminate between the i-skeleton with $i \leq q$ of a simplicial complex, meaning that the occurrences of a closed triangles are treated as occurrences of an open triangle.

Finally, we observe that in Zebra the simplets are concentrated in the small frequency range $[4200, 4800]$, which corresponds to $[41\%, 47\%]$ of the complex size; while in DBLP, they appear in the larger frequency range $[870k, 1.5M]$, which corresponds to $[45\%, 78\%]$ of the complex size. The fact that frequent simplets exist at larger frequencies indicates that the entities in DBLP are more interconnected than in Zebra, as there are several researchers that collaborate in large groups. The differences in the distributions of frequent simplets highlight that the our proposal captures different characteristics of the underlying simplicial complexes.

Exact frequency values. Finding all the occurrences of a simplet in a complex is an expensive task (see Section 5). As an example, finding the exact frequencies in ETFs takes $160\times$ more than the decision version of the problem; while in Zebra it takes $152\times$ more. Trading exactness for shorter runtime might be meaningful in many applications, however, when computing the exact frequency values is feasible, we can use them to find the top-k simplets. This filtering allow the user to reduce the size of the output, and get only the most relevant structures. We order all the simplets found in ETFs and Zebra, and compare their top-5 simplets to determine if they are characterized by different structures. Zebra is characterized by $P_1 : \{(0, 1, 2, 3), (0, 1, 3), (0, 1, 2)\}$, i.e., two closed triangles sharing one edge. ETFs is characterized by $P_2 : \{(0, 1, 2, 3, 4), (0, 3) - (0, 1, 2) - (1, 4)\}$, i.e., a closed triangle with two tails. In the literature, frequent patterns have been
successfully used to learn vector representations of graphs for classification purposes [4]. Similarly, frequent simplets can be used to learn finer-grained vectors for complexes.

Figure 4 (right) compares the distributions of frequent simplets in the two complexes. For each simplet of each dimension, we report the relative frequency with respect to the largest frequency in the complex. From this chart, we can better understand how close are the frequent simplets in Zebra, with all the relative frequencies being higher than 0.78. In contrast, in ETFs most of the simplets have less than 0.70 of the frequency of the top simplet, and hence the latter can be a good discriminator for this complex.

Face-to-face interactions. Studying face-to-face interactions in public places can help understanding how, e.g., transmissible diseases propagate. Each simplet encodes a different type of social relation, and frequent simplets indicate relations that happen frequently over time. We can focus on a specific simplet, and study how its frequency changes over time. Persistent simplets of higher dimension indicate patterns of communication with higher risk, as they represent an interaction among more parties [26] that took place frequently over time. These patterns cannot be effectively modeled via graph patterns: an open triangle in a simplicial complex indicates people that have pairwise contacts, while a closed triangle indicates three people that interact simultaneously. The two situations are not distinguishable in simple graphs, although the latter represents a situation of higher risk.

Figure 5 reports the number of frequent (\(\tau = 40\)) simplets found in the High (left) and the Primary (middle) temporal complexes (datasets description in Appendix A.3), with different colors indicating simplets of different dimension. Primary-school students interact more consistently over time, with a higher spike characterized by stronger contacts during the lunch break. In contrast, high-school students have fewer interactions in class (we observe only frequent simplets of dimension 1), and engage with other students mainly during lunch and before class.

Figure 6 shows the normalized entropy of the genders and classes in the occurrences of all the frequent simplets of dimension > 1, found in High (top) and Primary (bottom). Gender (Class) entropy indicates the tendency of people to gather with people of different gender (class). Therefore, smaller values of gender entropy indicate more homophilic interactions; while larger values denote heterophilic interactions. Primary-school students frequently interact with both female and male students, but during lunch, they tend to group together with children of the same gender. The heterophilic interactions take place more often during class, as classes consist of children of different genders. The dimensionality of these interactions is lower, and may be due to children sitting in adjacent desks. Similarly, high-school students interact more frequently with students of the same gender before class and during lunch, and hence the gender entropy is lower around 9am and 12am. By looking at the number of frequent simplets in the two charts, we note that primary-school students have stronger interactions during lunch. The values of class entropy tell us that students frequently mix with students of other classes, whenever they have the chance (e.g., during lunch and before/after class).

Finally, Figure 5 (right) reports the persistence of the frequent simplets of dimension 2 and 3, across all the snapshots of High and Primary. The x-axis is sorted by dimensionality of the simplet, i.e., the 3-dim simplets are on the right. In High, the simplets are less persistent, which indicates that high-school students communicate in large groups less frequently. The difference with Primary is more evident in the simplets of higher dimension. Few higher-order interactions persist among high-school students, whereas primary-school children meet with other children more consistently.

Simplicial closure prediction. Link prediction has the goal of predicting whether two nodes not connected in a network will be linked in the future, and is important in many applications like recommender systems [3]. Benson et al. [9] are the first to study the problem of link prediction in higher-order networks. In their seminal work, they extend the theory of triadic closure in social networks by defining the concept of simplicial closure, i.e., the tendency of groups of nodes with strong past interactions to co-appear in higher-order structures in the future. They focus on a specific case of simplicial closure, i.e., given a 3-node joint, predict whether the corresponding simplex will appear in the future. This problem is ignored in traditional link prediction, as the triangle would be already considered part of the graph. They evaluate the prediction performance of several models as the ratio between the area under the precision-recall curve (AUC-PR) and a random score, calculated as the proportion of open triangles in the training set that close in the test set. The models considered are harmonic (hm), geometric (gm), and arithmetic (am) means of the edge weights in the open triangle, Adamic-Adar coefficient (aa), preferential attachment (pa), personalized PageRank similarity (ppr), Katz similarity (ka), and logistic regression on 6 types of features (lr). The first 7 models provide a score for each triangle, while the latter is a supervised algorithm. Experimental results prove that the task is challenging, and that tie strength is important in predicting simplicial closure.
In our experiment, we consider the two temporal networks High and Primary, and focus on the task of predicting which open triangle at time \( t \) will close at time \( t + 1 \). We generate our ground truth considering all the open triangles at time \( t \), and assigning them 0 if they are open at time \( t + 1 \), and 1 otherwise. For each pair of vertices \( e = (u, v) \) and each timestamp \( t \), we create a binary vector \( b(e, t) \) such that \( b(e, t)[i] = 1 \) if \( v \) appears in some occurrence of the \( i \)-th frequent simplet, and 0 otherwise. Then, we generate a feature vector for each triangle \( T \) and each timestamp \( t \), by taking the harmonic mean of the vectors \( b(e, t) \) of its edges.

We split the data into train and test set \((\frac{2}{9}, \frac{7}{9})\) randomly, and train a Gradient Boosting Classifier (GBC). Since the data is highly imbalanced towards class \( 0 \), we use class weights. We evaluate the model in terms of the normalized AUC-PR, averaged over 10 runs, and compare with the models considered by Benson [9]. Due to space limitations, we refer to [9] and the corresponding Supplementary Material\(^2\) for their detailed description. Table 1 reports the normalized AUC-PR scores for all the models in High (H) and Primary (P).

We observe that all the models achieve better performance than random guessing, with some models achieving one order of magnitude improvement. The fact that the mean-based models perform better than the other baselines indicate that the presence of many higher-order structures around an edge is a good predictor. The logistic regressor, which uses properties of the nodes in the projected graph and information about the simplices containing the nodes, overtakes all the baselines, as it exploits the training data to learn a better model. Our approach, GBC-FresCo, achieves the best performance in Primary, and ranks third in High, proving that encoding the neighborhood of the edges in terms of the frequent simplets is a good predictor of triangle closure. The frequency threshold \( \tau \) used to mine the simplets in the complex, considerably affects the performance of GBC-FresCo, and hence must be properly tuned for the dataset at hand. We plan to further investigate the ability of frequent simplets to predict simplicial closure, by considering complexes of different nature and different frequency thresholds.

Table 1: Normalized AUC-PR scores in the triangle closure task in Primary (P) and High (H), for different models.

| \( K \) | \( \text{hm} \) | \( \text{gm} \) | \( \text{am} \) | \( \text{aa} \) | \( \text{pa} \) | \( \text{ka} \) | \( \text{ppr} \) | \( \text{lr} \) | \( \text{GBC} \) |
|---|---|---|---|---|---|---|---|---|---|
| P | 31.26 | 31.73 | 26.03 | 5.03 | 2.03 | 1.78 | 2.29 | 33.04 | 37.46 |
| H | 47.92 | 48.44 | 38.32 | 9.90 | 4.28 | 7.79 | 3.83 | 60.85 | 48.22 |

7 CONCLUSIONS

We formulated the problem of frequent simplet mining in simplicial complexes, which extends the traditional graph pattern mining to higher dimensions. We showed how to measure the frequency of a simplet in a complex, and how to generate all the candidate simplets up to a max size. We presented our algorithm, FresCo, that exploits the relations between the simplices in the complex and between the simplets to speed up the search and prune the search space. In addition, FresCo adopts a delayed canonization procedure, to reduce the amount of isomorphism checks between simplets. This way, it can scale with the size of the simplets. FresCo can both answer the question "is a simplet \( P \) frequent?", and find the exact frequency of the simplets. Our experimental evaluation showed that FresCo can efficiently mine simplets in various complexes, and that simplets allow finer-grained characterizations of the complexes. Finally, we studied how relations change in two temporal complexes, and discussed the persistence and entropy of those relations.

\(^2\)https://www.pnas.org/content/suppl/2018/11/08/1800683115.DCSupplemental
REFERENCES

[1] Ehab Abdelhamid, Ibrahim Abdelaziz, Panos Kalnis, Zbahir Khayyat, and Fuad Jamour. 2016. ScaleMine: Scalable parallel frequent subgraph mining in a single large graph. In SC'16: 716–727.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In VLDB’94: 489–499.

[3] Luca Maria Arieo, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. 2012. Friendship prediction and homophily in social media. TWEB 6, 2 (2011), 1–33.

[4] Md Alam, Chowdhury Farhan Ahmad, Md Samiullah, Carson K Leung, et al. 2021. Discriminating frequent pattern based supervised graph embedding for classification. In PAKDD: 16–28.

[5] Vikraman Arvind, Bireswar Das, Johannes Köbler, and Seinosuke Toda. 2010. Colored hypergraph isomorphism is fixed parameter tractable. In FSTTCS: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[6] Cigdem Aslay, Muhammad Anis Uddin Nasir, Gianmarco De Francisci Morales, and Aristides Gionis. 2018. Mining Frequent Patterns in Evolving Graphs. In CIKM: 923–932.

[7] Ron Arkin. 1974. Mathematical structure in human affairs. Heinemann Educational Publishers.

[8] Austin Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. Science 353, 6295 (2016), 163–166.

[9] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jalabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. PNAS 115, 48 (2018), E11211–E11210.

[10] Austin R. Benson, David F. Gleich, and Desmond J. Higham. 2021. Higher-order Network Analysis Takes Off, Fuelled by Classical Ideas and New Data. SIAM News (2021).

[11] Michele Berlingerio, Fabio Pinelli, and Francesco Calabrese. 2013. ABCACUS: frequent pAttern mining-BAsed Community discovery in multidimensional network. Data Min. Knowl. Discov. 27, 3 (2013), 294–320.

[12] Jacob Charles Wright Billings, Mirko Hu, Giulia Lerda, Alexey N Medvedev, Francesco Mottes, Adrian Onicas, Andrea Santoro, and Giovanni Petri. 2019. Simplex2vec embeddings for community detection in simplicial complexes. arXiv preprint arXiv:1906.09988 (2019).

[13] Björn Bringmann, Michele Berlingerio, Francesco Bonchi, and Aristides Gionis. 2010. Learning and Predicting the Evolution of Social Networks. IEEE Intell. Syst. 25, 4 (2010), 26–35.

[14] Björn Bringmann and Siegfried Nijssen. 2008. What is frequent in a single graph? In PAKDD: 858–863.

[15] Yuri Dabaghian, Facundo Mémoli, Lorenzo Frank, and Gunnar Carlsson. 2012. A topological paradigm for haptic spatial map formation using persistent homology. PloS Computational Biology 8, 8 (2012).

[16] Prasanna Desikan and Jauderp Srivastava. 2006. Mining Temporally Changing Web Usage Graphs. In Advances in Web Mining and Web Usage Analysis, Bamdad Mohamer, Olfa Nasraoui, Bing Liu, and Brian Massand (Eds.). Springer Berlin Heidelberg, 1–17.

[17] W Dörfler and DA Waller. 1980. A category-theoretical approach to hypergraphs. Archiv der Mathematik 34, 1 (1980), 185–192.

[18] M. Eidesy, E. Abdelhamid, S. Skadopoulos, and P. Kalnis. 2014. Grami: Efficient sampling from the simplicial configuration model. In SIAM Data Mining.

[19] Prasanna Desikan and Bamshad Bamshad. 2019. Why, how, and when of representations for complex systems. arXiv preprint arXiv:1912.03149 (2019).

[20] David I Spivak. 2009. Higher-dimensional models of networks. Journal of Statistical Mechanics: Theory and Experiment 2009, 03 (2009), P03014.

[21] Tamas Horvath, Björn Bringmann, and Luc De Raedt. 2006. Frequent hypergraph mining. In International Conference on In deductive Logic Programming: 244–259.

[22] Jacopo Iacopini, Giovanni Petri, Alain Barrat, and Vito Latora. 2019. Simplicial models of social contagion. Nature Communications 10, 1 (2019), 1–9.

[23] Kastra Jamshidi, Rakesh Madahasa, and Kevai Vora. 2020. Peregine: a pattern-aware graph mining system. arXiv preprint arXiv:2006.09068.

[24] Ruoming Jin and G. Agrawal. 2005. An algorithm for in-core frequent itemset mining on streaming data. In ICDM: 8–pp.

[25] Xiao Liu, Nanyin Zhang, Cate Chang, and Jeff H Duny. 2018. Co-evolution patterns in resting-state fMRI signals. Neuroimage 180 (2018), 485–494.

[26] Louis-David Lord, Paul Expert, Henriette M. Fernandes, Giovanni Petri, Tim J. Van Harteveld, Francisco Vaccarino, Gustavo Deco, Federico Turkoheimer, and Morten L. Kringelbach. 2016. Insights into Brain Architectures from the Holomorphic Scaffolds of Functional Connectivity Networks. Frontiers in Systems Neuroscience 10 (2016), 85.

[27] Eugene M. Luks. 1989. Hypergraph isomorphism and structural equivalence of boolean functions. In ACM Symposium on Theory of Computing: 652–658.

[28] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. 2015. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. PloS one 10, 9 (2015), e0136497.

[29] Changpeng Meng, S Chandra Mouli, Bruno Ribeiro, and Jennifer Neville. 2018. Subgraph pattern neural networks for high-order graph evolution prediction. In ACh Conference on Artificial Intelligence: Vol. 32.

[30] S. Moens, E. Aksehur, and B. Goethals. 2013. Frequent Itemset Mining for Big Data. In BigData: 111–118.

[31] Muhammad Anas Uddin Nasir, Cigdem Aslay, Gianmarco De Francisci Morales, and Matteo Riondato. 2020. TipTap: Approximate mining of frequent k-Subgraph patterns in evolving graphs. TKDD 15, 3 (2021), 1–35.

[32] C. C. Noble and D. J. Cook. 2003. Graph-based Anomaly Detection. In SIGKDD: 631–636.

[33] Takeshi Obayashi, Yuki Kagaya, Yuichi Aoki, Shu Tadaka, and Kengo Kominoha. 2018. COXPRESS: A gene coexpression database for 11 animal species supported by 23 coexpression processes for technical evaluation and evolutionary inference. Nucleic Acids Research 47, D1 (2018), D55–D62.

[34] Alice Patania, Giovanni Petri, and Francesco Vaccarino. 2017. The shape of collaborations. EFJ Data Science 6, 1 (2017), 1–16.

[35] Giovanni Petri, Paul Expert, Federico Turkoheimer, Robin Carhart-Harris, David Nutt, Peter J Hellyer, and Francesco Vaccarino. 2014. Homological scaffolds of brain functional networks. Journal of The Royal Society Interface 11, 101 (2014), 20140873.

[36] Giulia Preti, Gianmarco De Francisci Morales, and Francesco Bonchi. 2021. TRuD-Trius Decomposition of Simplicial Complexes. In Proceedings of The Web Conference 2021: 3408–3418.

[37] Giulia Preti, Gianmarco De Francisci Morales, and Matteo Riondato. 2021. MaN-ACS: Approximate mining of frequent subgraph patterns through sampling. In KDD: 1348–1358.

[38] Giulia Preti, Matteo Lissandrini, Davide Mottole, and Yannis Velegrakis. 2018. Beyond Frequencies: Graph Pattern Mining in Multi-weighted Graphs. In IJDBT: Vol. 18: 169–180.

[39] Saif Ur Rehman and Sohail Agha. 2020. Online social network trend discovery using frequent subgraph mining. Social Network Analysis and Mining 10, 1 (2020), 1–13.

[40] Vaerolod Salnikov, Daniele Casseau, and Renaud Lambiotte. 2018. Simplicial complexes and complex systems. European Journal of Physics 40, 1 (2018), 1–16.

[41] Daniel Hernández Serrano and Darío Sánchez Gómez. 2019. Centrality measures for hypergraphs. Computation 7, 1 (2019), 1–13.

[42] Toby Rocklin, Arndt Seger, and C. C. Noble. 2017. Simplicial closure using frequent subgraph mining. In Proceedings of the 2nd SDM Workshop on Mining Networks and Graphs: A Big Data Analytic Challenge.

[43] Mehdi Heydari, Raed Ali Helal, and Khairel Imran Ghiath. 2009. A graph-based web usage mining method considering client side data. In 2009 International Conference on Electrical Engineering and Informatics, Vol. 1. IEEE: 147–153.

[44] Danijela Horak, Slobodan Maletić, and Milan Rajković. 2009. Persistent homology of complex networks. Journal of Statistical Mechanics: Theory and Experiment 2009, 03 (2009), P03014.
A SUPPLEMENTARY MATERIAL

In the following, we provide the pseudocode of the procedures used by FreScO, the proofs of the main theorems, and characteristics and description of the datasets used in the experimental analysis.

Algorithm 2 EXAMINE (DP)

Require: Complex $\mathcal{K}$, Simplicial $P$, Parent simplicial $SP$

Ensure: Image sets $IP$ of the vertices in $VP$

1: $NC \leftarrow SP.NC$
2: $H \leftarrow \{v \in VP \mid 3\sigma \in \mathcal{K} \land v \in \sigma \land |V_\sigma| \geq |V_P|\}$
3: $IP(\omega) \leftarrow H$ for $\omega \in VP$
4: for $v \in VP$ do
5: if $IP(v) < r$ then
6: $Cands \leftarrow VP \setminus (IP(v) \cup NC(\omega))$
7: sort Cands placing first the vertices in $IP(v)$
8: if $|Cands \cup IP(v)| < r$ then return $\emptyset$
9: $c \leftarrow 0$, resume $\emptyset$
10: for $u \in Cands$ do
11: $c \leftarrow c + 1$
12: if $|\Gamma(u)| < |\Gamma(\omega)|$ then continue
13: $M \leftarrow 0$, $M[u] \leftarrow u$
14: $M \leftarrow \text{findMatch}(\mathcal{K}, P, M, time()$, $t^*)$
15: if $|M| = |VP|$ then
16: update and propagate $IP$ with $M$
17: else if $M = 0$ then
18: resume $\leftarrow \text{resume} \cup \{u\}$, $c \leftarrow c - 1$
19: else NC(\omega) $\leftarrow NC(\omega) \cup \{u\}$
20: if $\{Cands \cup IP(v) - c < r$ then return $\emptyset$
21: if $|IP(v)| \geq r$ then break
22: if $|IP(v)| < r$ then do loop at 10-21 for $c$ in resume
23: $P.NC \leftarrow NC$
24: return $IP$

25: function $\text{findMatch}(\mathcal{K}, P, M, t^*)$
26: if $time() - t^* > r$ then return $\emptyset$
27: if $|M| = |VP|$ then return $M$
28: $x \leftarrow$ next vertex to match
29: $Cands \leftarrow \left\{(\text{inc} \wedge \text{weak}(\mathcal{K}, \mathcal{K}) \Gamma(M[w])) \right\}$
30: for $c$ in Cands do
31: if $c$ satisfiesConstraints($\mathcal{K}, P, M, x, n$) then
32: $M[x] \leftarrow n$, $M \leftarrow \text{findMatch}(\mathcal{K}, P, M, t^*)$
33: else if $|M| = |VP|$, $M = 0$ then return $M$
34: else if time() $- t^* > r$ then return $\emptyset$
35: return $M$

36: function satisfiesConstraints($\mathcal{K}, P, M, x, n$)
37: for $\sigma$ in $CP$ such that $x \in \sigma$ do
38: if $M \mid \sigma \cap n \notin \mathcal{K}$ then return false
39: return true

A.1 Pseudocode

We report the pseudocode of Procedure EXAMINE invoked at line 17 of Algorithm 1 in its two versions: Algorithm 2 determines if a simplicial vertex is frequent without computing the exact frequency, and Algorithm 3 finds the exact frequency of all the frequent simplices. We also provide a brief description of Algorithm 3.

Computing exact frequencies. Algorithm 3 provides the procedure EXAMINE to find the exact frequencies. To save time, the algorithm looks for valid assignments for a vertex $v$ in the expansion $P$, only among vertices that were valid assignments for the corresponding vertex in the parent simplicial $SP$, i.e., the set of candidates $U(v)$ is initialized with $IP(v)$. However, if $v$ is a newly added vertex, $U(v) \leftarrow V_P$. This initialization leverages that only valid occurrences of $SP$ can be expanded into valid occurrences of its expansions $P$. The algorithm sorts the vertices $v$ in the simplet by increasing size of $U(v)$, with the hope to detect earlier if the simplet is not frequent, and hence stop the examination. Then, it iteratively visits each vertex $v$ to compute the image set $IP(v)$. Given that an assignment $u$ may have already been added to $IP(v)$ when visiting another simplicial vertex, we initialize the set of candidates as the set difference between $U(v)$ and $IP(v)$. Similarly to Algorithm 2, once assigned a candidate $u$ to $v$, Procedure $\text{findMatch}$ finds an assignment for all the other vertices in the simplicial. Procedure $\text{findMatch}$ examines each candidate match $n$ for each other vertex $x$ of $P$, and differs from Procedure $\text{findMatch}$ in Algorithm 2, in two parts. First, it creates the restricted set $\text{Cands}$ via intersection with the upper bound set $U(x)$ (line 18). Secondly, it has no time constraints, and hence searches for a valid assignment starting from $u$ until it either finds it or it has explored all the possibilities.

Algorithm 3 EXAMINE (EF)

Require: Complex $\mathcal{K}$, Simplicial $P$, Sets of Candidates $U1$

Ensure: Image sets $IP$ of the vertices in $VP$

1: $H \leftarrow \{v \in VP \mid 3\sigma \in \mathcal{K} \land v \in \sigma \land |V_\sigma| \geq |V_P|\}$
2: $IP(\omega) \leftarrow H$ for $\omega \in VP$
3: sort $v \in VP$ by size of $U(v)$
4: for $v \in VP$ do
5: $\text{Cands} \leftarrow U1(\omega) \setminus IP(\omega)$, $c \leftarrow 0$
6: for $u \in \text{Cands}$ do
7: $c \leftarrow c + 1$
8: if $|\Gamma(u)| < |\Gamma(\omega)|$ then continue
9: $M \leftarrow 0$, $M[u] \leftarrow u$
10: $\text{M} \leftarrow \text{findMatch}(\mathcal{K}, P, M, U1, M)$
11: if $|M| = |VP|$ then
12: update and propagate $IP$ with $M$
13: if $\{\text{Cands} \cup IP(v) - c < r$ then return $\emptyset$
14: return $IP$

15: function $\text{findMatch}(\mathcal{K}, P, U1, M)$
16: if $|M| = |VP|$ then return $M$
17: $x \leftarrow$ next vertex to match
18: $\text{Cands} \leftarrow \left\{(\text{inc} \wedge \text{weak}(\mathcal{K}, \mathcal{K}) \Gamma(M[w])) \cap U1(x) \right\}$
19: for $n \in \text{Cands}$ do
20: if $\text{satisfiesConstraints}(\mathcal{K}, P, M, x, n)$ then
21: $M[x] \leftarrow n$, $M \leftarrow \text{findMatch}(\mathcal{K}, P, U1, M)$
22: if $|M| = |VP|$ then return $M$
23: return $M$

A.2 Proofs

Proof of Theorem 4.1.

PROOF. Let $P$ be a simplicial and $\sigma^{(q)}$ be a $q$-simplicial in $CP$. By definition, $\sigma^{(q)}$ consists in a set of $q + 1$ interconnected vertices, W.L.O.G., pick a vertex $v$ in $\sigma^{(q)}$ and start the procedure from $[v]$. By applying $\text{widen}$ $q$ times, we add the remaining vertices in $\sigma^{(q)}$, connecting them to $[v]$ via 1-simplices. This generates a star structure with $[v]$ at the center. Then, by applying $\text{inflate}$ $q(q - 1)/2$ times, we add the remaining 1-simplices between the vertices, and by applying it other $\sum_{i=1}^{q-1} (q-1)^i$ times, we obtain $\sigma^{(q')}$. In particular, for each $i \geq 3$, the $(q-1)^i$ applications add all the $(i-1)$-simplices. By definition of simplicial, all the simplices in $CP$ are connected to at least another simplicial in $CP$. Therefore, there exists at least one $\sigma^{(q')} \in CP$ that shares a face with $\sigma^{(q)}$. By applying $\text{widen}$, we add...
to the simplicial vertex in $\sigma(i)$, and hence, if a valid assignment exists, it can find it. By repeating the last two steps for each other simplex in $P$, we obtain $P = P$. □

**Anti-monotonicity of SUP.** Proof of Property 1.

Proof. We prove the anti-monotone property by contradiction. Let consider a simplex $P$ and its sub-simplicial $SP$, and let $\psi$ be an injective function from $V_{SP}$ to $V_P$ that preserves the relations between the vertices (we can define such $\psi$ because $C_{SP} \subseteq C_P$). Let assume that $P$ has frequency $SUP(P, K)$ greater than $SP$. By definition of $SUP$, this means that for all $\psi \in V_{SP}$ such that $|ISP(\psi)| = SUP(SP, K)$, it holds that $|IP(\psi(\psi(\psi)))| \geq |ISP(\psi)|$. W.l.o.g., let's pick one simplicial vertex $v = \psi(v)$ and a complex vertex $n \in IP(v \setminus ISP(\psi))$. By definition of image set, if $n$ belongs to $IP(v)$, then there exists a bijection $\phi$ from a subset of complex vertices to $V_P$ such that (i) $\phi(n) = \psi(n)$ and (ii) it preserves the relations between the vertices. However, this means that $\theta = \psi^{-1} \circ \phi_{Cdm}(\psi)$, where $Cdm(\psi)$ is the codomain of $\psi$, is a valid bijection from a subset of complex vertices to $V_{SP}$ such that (i) $\theta(n) = v$ and (ii) it preserves the relations between the vertices. We reach a contradiction, because we assumed $n \in IP(v \setminus ISP(\psi))$, but by definition of image set, we must have that $n \notin ISP(\psi)$. □

**Completeness and correctness of Algorithm 1.**

**Theorem A.1 (Completeness and Correctness).** Algorithm 1 returns all and only the simplets $P$ with $SUP(P, K) \geq \tau$.

Proof. Theorem 4.1 proves that the expansion strategy used by Procedure expand generates the complete lattice. Thanks to the anti-monotonicity property of the SUP measure, a simplex can be frequent only if its sub-simplexes are frequent. Therefore, by expanding only the frequent simplexes we do not discard any candidate frequent simplex. This ensures the completeness of the algorithm.

We first show the correctness of Algorithm 2. This algorithm searches for an assignment for a vertex $v$ among all the complex vertices, and hence, if a valid assignment exists, it can find it. Early stops happen in two cases: if it has found at least $\tau$ assignments for all the vertices in $P$, or if the remaining candidates are not enough to prove that $P$ is frequent. In the first case, $SUP(P, K) \geq \tau$ and hence $P$ is correctly added to the output set. In the second case, the simplex cannot be frequent, because $SUP(P, K)$ cannot be greater than $\tau$ even if all the remaining candidates are valid assignments, and hence we do not need to examine them. Therefore, the algorithm ensures a correct answer to the question $SUP(P, K) \geq \tau$.

We now show the correctness of Algorithm 3. This algorithm initializes the sets of candidates for the simplicial vertices of $P$ with the image sets of the parent simplex of $P$. This optimization leads to correct results, because only valid assignments for sub-simplexes of $P$ can be expanded to valid assignments for $P$. The algorithm iteratively assigns each candidate $u$ to each simplicial vertex $v$, and then tries to find an assignment for all the other simplicial vertices via Procedure findMatches. Since findMatches examines each candidate for all the other simplicial vertices, if a valid assignment exists, we surely find it. The validity of the assignment is checked in lines 18 and 38, where we ensure that assignments for vertices that are connected in the simplicial are connected in the complex. Therefore, Algorithm 3 leads to the correct computation of $SUP(P, K)$, and hence ensures a correct answer to the question $SUP(P, K) \geq \tau$. □

**Complexity of FanSCo** Reference [32] proves that the isomorphism problem for sub-hypergraphs is NP-hard. As complexes are special kinds of hypergraphs, the simplicial isomorphism problem is also NP-hard. Let $S_k = \sum_{i=2}^k \binom{n}{i}$ denote the number of $(j - 1)$-simplices with up to $k$ vertices. Then, the number of simplexes with $k$ vertices is upper bounded by $2^k$ (there is no known formula for the exact number of distinct simplexes). To determine the frequency of a simplex with $k$ vertices, we need all its isomorphisms in the complex, which takes up to $O(k^2)$ space. Therefore, examining all the simplexes of size $k$ requires up to $O(n^2k^2|V|^k)$. By summing over $k$ up to $\tau$, we obtain the total running time. Thanks to our heuristics (canonical forms, expansion strategy, candidate pruning, early termination), this time is significantly reduced in practice.

### A.3 Datasets

Table 2 reports the characteristics of the simplicial complexes used.

| Datasets   | Vertices | Edges | Triangles | Maximal Simplices | Max Dim | DBLP | Enron | Zebra | ETFs | High | Primary |
|------------|----------|-------|-----------|-------------------|---------|------|-------|-------|------|------|---------|
| DBLP       | 1914581  | 50224 | 10112     | 2340              | 327     | 242  |       |       |      |      |         |
| Enron      | 7683900  | 330179| 110910    | 5543              | 5818    | 8317 |       |       |      |      |         |
| Zebra      | 1730504  | 75296 | 8146      | 1125              | 4862    | 8010 |       |       |      |      |         |
| ETFs       | 14350197 | 1431200| 1737598   | 7130              | 2370    | 5139 |       |       |      |      |         |
| High       | 17       | 64    | 62        | 19                | 5       | 5    |       |       |      |      |         |
| Primary    | 9        |       |           |                   |         |      |       |       |      |      |         |

**DBLP** (Benson et al. [9]) is a DBLP co-authorship simplicial complex: each simplex represents a publication and its vertices are the corresponding authors.

**Enron** (https://www.cs.cmu.edu/~enron) contains emails sent from employees of the Enron corporation. Vertices are senders and recipients, and simplices denote emails.

**Zebra** is constructed from genetic data of zebrafishes [38]. Starting from the gene correlation table (https://coxpresdb.jp/download), we obtain a simplex from each set of genes with mutual rank value lower than $\frac{1}{2}$th of the average value in its neighborhood. We retain the simplices with size in the 99th percentile.

**ETFs** (Exchange-Traded Funds - shorturl.at/itEK7) contains general assets, portfolio indicators, returns, and financial ratios of 2353 ETFs, scraped from the Yahoo Finance website (https://finance.yahoo.com). Each simplex is a set of ETFs with Kendall correlation of the 99th percentile, where the correlation is computed exploiting the aforementioned features. We retain the simplices with size in the 99.9th percentile.

**High** [33] and **Primary** [21] are temporal simplicial complexes constructed from two SocioPattern datasets (http://sociopatterns.org).

These datasets report active face-to-face contacts measured between students in a high school in Lycée Thiers, Marseilles, France; and children in a primary school in Lyon, France. Simplices are constructed by combining contacts that happen at the same time and location, and aggregate them into 10-minutes snapshots. The two complexes have 246 and 104 snapshots, respectively.