CrystalNets.jl: Identification of Crystal Topologies

Lionel Zoubritzky and François-Xavier Coudert*

Chimie ParisTech, PSL University, CNRS, Institut de Recherche de Chimie Paris, Paris, France
* fx.coudert@chimieparistech.psl.eu

May 25, 2022

Abstract

We present here an open-source Julia library for the topological identification of crystalline materials, with algorithmic and computational improvements over the previously available software in the field, resulting in a speed increase of one order of magnitude. This new algorithm and implementation can therefore be used at large scale in high-throughput screening methodologies. We have validated and benchmarked CrystalNets.jl against a diverse set of crystal databases, covering in particular metal–organic frameworks, aluminophosphates, zeolites, and other inorganic compounds.

Contents

1 Introduction

In order to describe the average structure of crystalline materials at the nano-scale, reticular chemistry [1, 2] proposes to classify crystals, and in particular framework materials, in light of their topology. This approach consists in abstractly regrouping the atoms of a crystal into clusters, called *vertices*, bonded together by an abstract kind of linkers, called *edges* — the choice of grouping being informed by chemical sense. In simple compounds, such as dense oxides, the mapping can be trivial, with each atom as a separate vertex and each chemical bond as an edge. For zeolites and aluminosilicates, the tetrahedral atoms T are considered the vertices (T = Al, Si, P, ...) and the T–O–T linkages are the edges. In the case of metal–organic frameworks (MOFs), the vertices would be chosen among the *secondary building units* (SBUs) [3] with more than 2 points of extension, and the edges would correspond to bonds between those as well as ditopic SBUs, for example. Any crystal can then be simplified into a topological structure, called a *net*, consisting of the sets of vertices and edges (see Figure 1).

Like the atomic structure it represents, the net is 3-dimensional and periodic. It contains all the information on the connectivity of the vertices, *i.e.* which pairs of vertices, called *neighbours*, are linked by an edge. Thus, it provides a convenient mathematical abstraction to classify crystal structures. However, it is important to note that this representation is dependent on the choice of vertices and therefore a crystal may be represented by different nets [8]. Moreover, reconstructing the full crystal from the net necessitates additional data, such as the unit cell parameters and the nature and exact geometric placement of the vertices and the edges [5]. A crystal may also be composed of multiple interpenetrating nets [9], which is common in ultraporous framework materials.

Several existing projects, such as the RCSR [7] or Epinet [10, 11], aim at collecting the different nets underlying the structure of real or hypothetical crystals. This is key to allowing researchers to create new materials with a target topology, which is useful to obtain specific physical properties, given the strong link between materials topology and properties for a given chemical composition [12]. However, in order to check the accuracy of the collected nets with regard to existing crystals, one needs a way to determine the net from a given crystal structure.

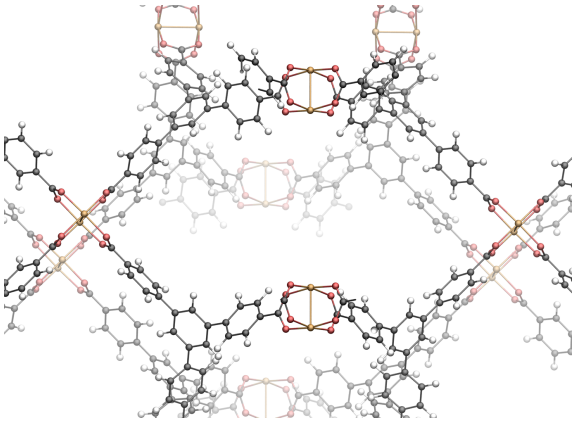
There are two different approaches to solve this problem, which correspond to the two main softwares used in the field: ToposPro [13, 14] and Systre [15]. ToposPro is a crystallographic tool that allows to retrieve an extensive set of topological properties from crystals. It is a proprietary software, but its topology detection tool can be accessed for free online [16][1] for crystallographic files in CIF format [17]. Its strategy to identify the topology of a net consists in computing three specific properties for each vertex u:

- The coordination sequence up to distance 10. A vertex v is at distance k of u is there is a chain of $k+1$ vertices $u = u_0, u_1, \ldots, u_k = v$ such that for all i in $[\![1, k]\!]$ (where $[\![a, b]\!]$ designates the set of integers between a and b), u_{i-1} and u_i are neighbours, and there is no smaller such chain. The k-th term of the coordination sequence of u is then defined as the number of vertices at distance k of u.
- The point symbol.
- The vertex symbol.

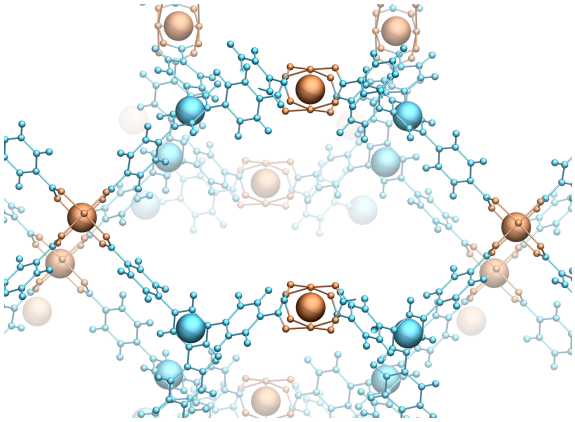
The latter two account for the shortest cycles and rings starting at vertex u [18, 19]. It is found empirically that the combination of these three properties for all vertices forms an almost unique identifier of the topology. However, this approach is not completely sound as there can be different nets that do share these same three properties.

Systre is an open-source software[2]; its inputs are files representing the graph underlying the crystal structure, rather than the crystallographic files directly. It implements an algorithm

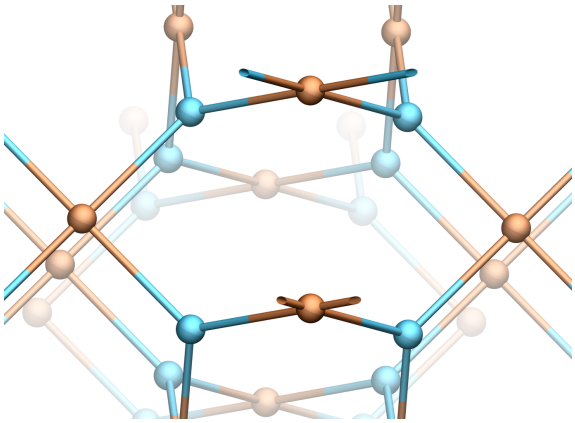
[1] available online at https://topcryst.com/
[2] available online at http://gavrog.org/



(a)



(b)



(c)

Figure 1: Decomposition of a crystalline framework into its underlying net:
(a) Crystalline structure of MOF-143 [4].
(b) Identification of the SBUs: Cu-based paddle-wheel [5] and benzene-tribenzoate (BTB) [6].
(c) The underlying net, called **pto** in the database of the RCSR [7].

that yields a so-called *topological genome* [20] of the net, which is a finite series of numbers. This genome is provably unique for each net, only depends on the net itself and not its representation, and can be computed in polynomial time of the size of the net. To do so, Systre starts by computing an *equilibrium placement* [21] of the vertices, which attributes to each vertex a position which is the barycentre of that of its neighbours. The net is then minimised, which amounts to finding a unit cell. Afterwards, Systre collects a number of *candidates*, which are frames of reference that do not depend on the current representation of the net. For each candidate, the vertices of the net are ordered according to their position in the frame of reference, and a key is determined which uniquely identifies the net with this ordering in this frame. The genome is then chosen as the lexicographically smallest key. Since the candidates do not depend on the initial representation of the graph, the keys do not either, hence the genome is indeed uniquely determined and only depends on the net itself. In graph theory, the problem Systre solves is called *graph canonization*. It incidentally allows determining whether two input graphs are actually two representations of the same graph (the *graph isomorphism* problem) by comparing their canonical forms (here, the genome), but it can also be used to give a canonical name to graphs for example, by accessing a database of names with the canonical form of the graphs.

Apart from ToposPro and Systre, a few alternative tools have been developed to simplify structures into topological descriptors, but none are suitable for topology identification of chemical structures at large. Indeed, they either only work on specific structures (usually zeolites) like TOTOPOL [22], or rely on a non-unique representation of the structure (such as a tiling [23]) like the T-ring representation [24], or because their output is not unique for each input structure, like the Single Repeating Unit representation [25] for example. For finite molecules, the two most common representations are the SMILES [26] file format, which is proprietary and comes with several conflicting implementations, or the IUPAC's International Chemical Identifier (InChi) [27] key, which is open-source and more standardized, but less human-readable. Both include additional chemical information as well.

In this context, we propose a new tool to efficiently determine the topology of crystal structures: the Julia package CrystalNets.jl. Julia [28] is a modern general-purpose and scientific programming language that offers both great performance and ease of programmability [29]. This, as well as carefully chosen optimisations, allows CrystalNets.jl to outperform Systre while leveraging a very similar core algorithm to ensure soundness. Moreover, the package uses the Julia port of chemfiles[3] which allows it to read common chemical file formats like CIF, making it easy to use for the analysis of single cases as well as large databases of crystal structures.

Delgado-Friedrichs and O'Keeffe [15] explained the overall architecture of the Systre algorithm which CrystalNets.jl largely reuses: we will follow its structure, only briefly summarising the key concepts that are already explained elsewhere along the way, while focusing on the implementation choices and some proofs of correctness that do not appear in other publications. We will then focus on a series of new optimisations of the algorithm that allow to greatly improve its performance for most nets. Finally, we will demonstrate the applicability of the method by analysing several existing crystalline databases using CrystalNets.jl.

2 Algorithms and implementation

2.1 Periodic graph structure

Crystal nets are special instances of 3-periodic graphs. In general, a *graph* is a mathematical construct that contains objects, called *vertices*, some of which are linked together by *edges*. Two

[3] available online at https://chemfiles.org/

linked vertices are said to be *neighbours*. In a crystal, the pattern representing the atoms and their chemical bonds periodically repeats itself across three dimensions: for this reason, the graph representing a crystal is called 3-periodic.

More generally an N-periodic graph (with $N \in \mathbb{N}^*$) can be represented using the vector model [30]. This model conceptually follows the description of a crystal as a lattice and a pattern. Any point in the lattice is denoted by an offset $o \in \mathbb{Z}^N$, whose i-th coordinate is the offset along the i-th axis of the lattice compared to a fixed origin. An N-periodic graph is then defined by a pair (V, E) where V is the set of vertex identifiers and $E \subset V \times V \times \mathbb{Z}^N$ is the set of edges. Any vertex in space is uniquely identified by a pair $(v, o) \in V \times \mathbb{Z}^N$ where $v \in V$ is its vertex identifier and $o \in \mathbb{Z}^N$ is the offset of the cell that contains the vertex. Let's note $n = |V|$ the number of vertices per cell and identify V with the integer interval $[\![1, n]\!]$ so that each vertex identifier can be considered to be a number in $[\![1, n]\!]$. When working in a fixed cell, the vertices will simply be referred to by their vertex identifier. Finally, for most of the rest of this article N will be equal to 3 since nets are 3-dimensional, but the algorithms exposed in the rest of this section actually work for any dimension and CrystalNets.jl can identify 2-periodic and 1-periodic graphs as well.

An edge $e = (u, v, o)$ can be broken down into a source $u \in V$, and a destination $(v, o) \in V \times \mathbb{Z}^3$. The source can be seen as the vertex of identifier u and positioned in the cell at the origin of the lattice, and the destination is the vertex to which it is linked. When the source and the destination are in the same cell, the offset o is zero and noted $0_{\mathbb{Z}^3} = (0, 0, 0)$. Edges of the form $(u, u, 0_{\mathbb{Z}^3})$, called *loops*, are absent in nets because they do not convey any structural information.

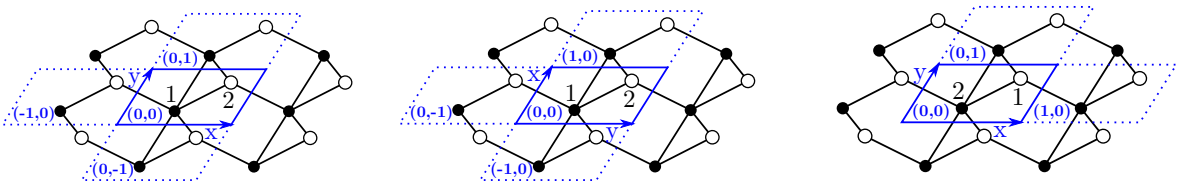
For any edge $e = (u, v, o) \in E$, the reverse edge $(v, u, -o)$ is also in E: this property makes the graph *undirected*. All graphs used to represent crystals are undirected because chemical bonds are symmetric: if atom A is bonded to atom B, then atom B is bonded to atom A, and likewise for clusters of atoms.

As opposed to periodic graphs, *simple* graphs do not have a concept of offset: an edge of a simple graph is only a pair of vertices. Numerous computational representations of simple graphs exist in the literature, but they cannot be directly transposed to periodic graphs. In order to find a computational representation for crystals relevant to topological analyses, a useful observation is that nets share two distinctive features that come from chemistry:

- The number of edges is at most proportional to the number of vertices, because of the limited valence of atoms. Such graphs are generally called *sparse*. This remains true when vertices represent clusters instead of single atoms, although in that case the number of neighbours can be higher than the maximum valence of an atom — for instance, SBUs with up to 66 points of extension have been built. [3].
- The set of edges is small to medium-sized because the number of vertices per unit cell is, in practice, between 1 and 10,000 in most real-world crystalline structures. This is small compared to the trillion of edges of some graphs in social media modelling for instance [31].

Finally, the most important operation on graphs that will subsequently be used is finding the set of neighbours of a given vertex. Driven by this, we chose to computationally represent a periodic graph by a list of size n whose u-th element is the set of neighbours of u: $N_u = \left\{(v, o) \in V \times \mathbb{Z}^3 \,\middle|\, (u, v, o) \in E\right\}$. N_u is stored as a lexicographically sorted list.

Because of the previously mentioned undirectedness of the graph, if $(v, o) \in N_u$, then $(u, -o) \in N_v$. However, it is often convenient to be able to enumerate the edges of the graph without double counting an edge and its reverse. For this reason, the representation comprises another list of size n whose u-th element is the first index i in N_u such that $N_u[i] = (v, o)$ with either $v > u$ or $\left[\ v = u \text{ and } o > 0_{\mathbb{Z}^3}\ \right]$. Such an edge (u, v, o) will be called a *direct* edge.



(a) Reference unit cell

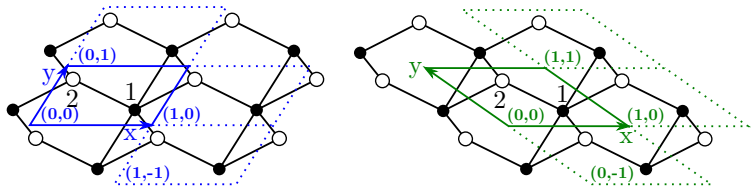
1	1	0	1
1	2	-1	0
1	2	0	-1
1	2	0	0

(b) Inversion of axes

1	1	1	0
1	2	-1	0
1	2	0	-1
1	2	0	0

(c) Renumbering of vertices

1	2	0	0
1	2	0	1
1	2	1	0
2	2	0	1

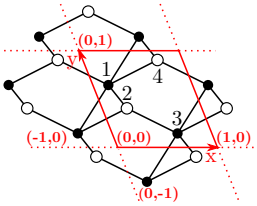


(d) Change of origin

1	1	0	1
1	2	0	0
1	2	1	-1
1	2	1	0

(e) Other unit cell

1	1	1	1
1	2	0	-1
1	2	0	0
1	2	1	0



(f) Non-unit cell

1	2	0	0
1	3	-1	0
1	3	0	1
1	4	-1	0
1	4	0	0
2	3	-1	0
2	3	0	0
3	4	0	-1

Figure 2: Six different representations of the same 2-periodic graph. The first two columns are the source and destination vertices, the last two are the edge offset.

Since loops are forbidden, all edges are either a direct edge, or the reverse of a direct edge. Thus, since each list of neighbours is lexicographically ordered, reading the list of neighbours starting at the given indices i is guaranteed to yield all direct edges of the graph, hence each edge exactly once without its reverse. The lexicographically ordered sequence of direct edges is a unique identifier of a representation (V, E) of a periodic graph in the vector model: such a sequence will be called the *key* associated with the representation.

We implemented the graph structure detailed above in a companion Julia package, PeriodicGraphs.jl, which can be used independently of CrystalNets.jl.

Figure 2 shows several representations of the same 2-periodic graph. The graph is made of two kinds of vertices (represented by open and closed circles), but the choice of the vertex identifiers and of the cell influences the representation. Each choice of cell is characterised by its origin and its axes x and y; some repeated cells are represented as well, along with their offset in the bottom-left corner. For each representation, the key is written below, each line referring to a direct edge. Recall that the Systre algorithm works by choosing the lexicographically smallest key from a set of representations, extracted from candidates that do not depend on the representation of the graph. Hence, if the set of representations was those shown in the figure, the topological genome of the graph would be the smallest key, *i.e.* that under representation (a).

Having a working representation for nets, let's now discuss our reimplementation of the Systre algorithm [15] to compute a unique identifier for each net topology.

2.2 Equilibrium placement

2.2.1 Definition

Given a periodic graph, the first step is to compute, for each vertex $(u, o') \in V \times \mathbb{Z}^3$, a position $p(u, o') \in \mathbb{R}^3$ such that

$$\forall u \in V, o' \in \mathbb{Z}^3,\ p(u, o') = \frac{1}{|N_u|} \sum_{(v,o)\in N_u} p(v, o' + o) \tag{1}$$

The set of positions p, called an *equilibrium placement*, is such that each vertex is at the barycentre of all its neighbours. p must also satisfy the additional constraint that

$$\forall u \in V,\ o \in \mathbb{Z}^3,\ p(u, o) = p(u, 0_{\mathbb{Z}^3}) + o \tag{2}$$

to enforce the periodicity of the equilibrium placement.

The equilibrium placement has been proven to be unique modulo isometries, because it is the solution of a full-rank linear problem [21]. Equation 1 can be rewritten with Equation 2 in matrix form, $A \times P = O$, where $A \in \mathcal{M}_{n,n}(\mathbb{Z})$, $O \in \mathcal{M}_{n,3}(\mathbb{Z})$ and unknown $P \in \mathcal{M}_{n,3}(\mathbb{R})$ such that:

- $\forall (u, v) \in V^2,\ A_{u,v} = \begin{cases} |N_u| & \text{if } u = v \\ -1 & \text{if } u \text{ and } v \text{ are neighbours} \\ 0 & \text{otherwise} \end{cases}$
- $\forall u \in V,\ O_u = \sum_{(v,o)\in N_u} o$

and from which we determine for all $u \in V$, $p\,(u, 0_{\mathbb{Z}^3}) = P_u$.

Since the coefficient of A and O are integers, the computed solution P is always in $M_{n,3}(\mathbb{Q})$ with the additional constraint that one of the vertices u be placed at the origin. The exact solution with coefficients in $\mathbb{Q}$ is actually necessary for the subsequent computations and not an approximation, so the resolution of the system must be exact. This turns out to be a very costly operation: it requires storing integers of arbitrary size, which are computationally far more expensive than fixed-width machine integers.

2.2.2 Computational aspects

The GNU Multiple Precision Arithmetic Library (GMP) [32] provides access to heavily optimised structures that allow to efficiently perform exact integer and rational computations. Julia has native support for the arbitrarily-sized integer structure from GMP, through the `BigInt` type, but does not provide a full interface for GMP's rational numbers (although it is internally used for Julia's native `Rational{BigInt}` type). As part of this work, we provided this special support in a separate package, BigRationals.jl, which exports the `BigRational` type wrapping GMP's rational type.

In addition to the data structure used, a choice has to be made about the algorithm for solving this linear system. The most straightforward idea would be a general linear algebra technique such as LU decomposition followed by forward and backward substitution. Yet, matrix A is structured since it is both symmetric and sparse, calling for the use of more efficient algorithms.

Several algorithms have been discussed in the literature for the exact resolution of sparse and integer linear systems [33–36]. Following the analysis by Eberly *et al.* [33], we decided to implement Dixon's algorithm [35] which relies on a modulo-prime inversion of the matrix. In accordance with their conclusion, this algorithm largely outperforms a sparse LU decomposition algorithm (which we implemented as a reference and for benchmarking), both algorithms

leveraging GMP instructions specific to the `BigRational` type for improved performance. Yet, Dixon's algorithm can fail if matrix A happens not to be invertible modulo the chosen prime: hence, we chose to successively invert it modulo three hard-coded large primes and stop at the first succeeding one, and finally resort to LU decomposition if all three fail. This implementation is up to orders of magnitude faster than that used in the original Systre program written in Java. This extra performance is crucial for the use of the topology detection algorithm as part of large-scale screening methodologies.

The equilibrium placement naturally embeds the graph in a Euclidean space, namely $\mathbb{Q}^3$. The first vertex of the graph is placed in position $(0,0,0) = 0_{\mathbb{Q}^3}$, which is allowed because any translation of the equilibrium placement still results in an equilibrium placement. Moreover, because of Equation 2, for any vertex identifier $u \in V$ there is an offset o such that $p(u,o) \in [0,1[^3$, where $[a,b[$ designates the set of reals between a (included) and b (excluded). In order to simplify the representation, this offset is set to $0_{\mathbb{Z}^3}$ by adjusting the offsets of the edges of the graph accordingly. In this new representation, equations 1 and 2 still hold, as well as the new property:

$$\forall u \in V,\ p(u) \in \left(\mathbb{Q} \cap [0,1[\right)^3 \quad \text{where } p(u) = p(u, 0_{\mathbb{Z}^3})$$

In order to accelerate the subsequent algorithms, the vertices of the graph are lexicographically sorted according to their new position. The first vertex, being at $0_{\mathbb{Q}^3}$, remains first.

Nets where at least two vertices have the same position are called *unstable* and, similarly to Systre, are generally not handled by our implementation. This is usually not problematic since their probability of occurrence is low [37]. Our implementation can however handle unstable nets where each collision site comprises at most four vertices if there is no edge between two different collision sites and no collision site is bonded to two representatives of the same vertex. This accounts for roughly half of the encountered unstable nets in the databases we explored. We will not detail the case of these unstable nets here but focus on stable ones. We can thus assume that no two vertices have the same position.

Figure 3 represents the equilibrium placement for the net of diamond (called **dia** in the RCSR). Note that the natural embedding in $\mathbb{Q}^3$ imposes a cubic unit cell for all nets, which is not representative of the actual unit cell of diamond; however, the geometry of the unit cell is irrelevant here because only the topology of the net matters. As a consequence, the bond lengths are not representative of that of diamond either, which explains why not all have the same lengths in this representation. Nonetheless, each vertex is indeed at the barycentre of the positions of its neighbours.

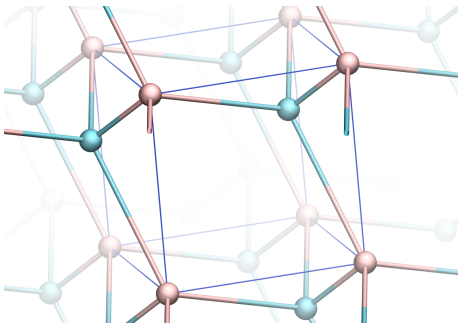


Figure 3: Visualisation of the natural Euclidean embedding of the equilibrium placement for the **dia** net.

2.3 Minimisation of the graph

Once the equilibrium placement is found, the next step of the algorithm is to find a unit cell. In general, a *cell* is a parallelepiped whose three axes $(\vec{a}, \vec{b}, \vec{c})$ are linearly independent translations that constitute symmetries of the graph. Such translations will be called *valid translations*. Another representation for this cell is the matrix A whose columns are respectively $\vec{a}$, $\vec{b}$ and $\vec{c}$. Its volume is $\det(A) = (\vec{a} \wedge \vec{b}) \cdot \vec{c}$. It is a *unit cell* if it contains the smallest possible number of vertices, which is equivalent to having a minimal volume.

At this point, it is important to recall that the Euclidean space in which the graph is embedded through the equilibrium placement depends on its initial representation, so there is no simple way of finding a unique unit cell independently of the original representation. In particular, common cell reduction techniques such as Delaunay [38] or Niggli [39,40], which yield a uniquely defined cell, do not solve this problem. Unfortunately, existing libraries implementing these techniques (like spglib [41] or cctbx [42]) cannot be used either for the simpler problem of finding any unit cell because they do not handle arbitrary precision.

First, note that, by construction of the equilibrium placement, the initial cell is the canonical basis of $\mathbb{Q}^3$, so its matrix is the identity I. Let's now assume that I is not a unit cell and there is a unit cell $A \in \mathbb{Q}^{3\times3}$ of volume $\det(A) \in]0,1[$.

Let's define $f : \mathbb{Q} \to \mathbb{Q} \cap [0,1[\,, \frac{p}{q} \mapsto \frac{p\%q}{q}$ where $p\%q$ is the remainder of the euclidean division of p by q (hence $p\%q \in [\![0, q-1]\!]$). Let $B = \left(f(A_{i,j})\right)_{1\le i,j\le 3} \in \mathbb{Q}^{3\times3}$. By definition of f, the columns of B are axes within the initial cell; moreover, since $\forall x \in \mathbb{Q},\ x - f(x) \in \mathbb{Z}$, each axis of B can be seen as the corresponding axis of A plus an offset $o \in \mathbb{Z}^3$. But since I is a repeating unit, such an offset is a valid translation, so each axis of B is a valid translation. Finally, since $\det(A) \notin \mathbb{Z}$, not all coefficients of A can be integers, so A has at least one coefficient x such that $f(x) \neq 0$, so at least one axis of B is not $0_{\mathbb{Q}^3}$.

To summarise, the previous observations show that if the initial cell was not a unit cell, then there exists a non-zero valid translation $t \in \left(\mathbb{Q} \cap [0,1[\right)^3 \setminus \{0_{\mathbb{Q}^3}\}$.

Let's use this fact to find a unit cell by enumerating all such translations. Symmetries map vertices to vertices, so the first vertex (lexicographically sorted by position) is mapped to another. But since the first vertex was put in position $0_{\mathbb{Q}^3}$, the translation is directly given by the position of the other vertex. Hence, only $n-1$ translations need to be checked, and if none is valid then the initial cell is a unit cell.

Checking whether a translation is valid can be done with $O(n \log n)$ time complexity. Vertices are already sorted by position, which allows to efficiently check that all the vertices are mapped to vertices. Doing so first allows to collect the map from initial vertices to their new counterparts and return early if the translation is invalid. Afterwards, it remains to be checked that all edges are also mapped to edges, which can also be done efficiently since the lists of neighbours N_u where $u \in V$ are also lexicographically sorted.

If I is not a unit cell, then a valid translation $t = (t_1, t_2, t_3) \in \left(\mathbb{Q} \cap [0,1[\right)^3 \setminus \{0_{\mathbb{Q}^3}\}$ must have been found. Let $k \in [\![1,3]\!]$ be such that t_k is the smallest non-zero coefficient of t. Then a new smaller cell is given by matrix $A^{(t)}$ where

$$A^{(t)}{}_{i,j} = \begin{cases} t_i & \text{if } j = k \\ 1 & \text{if } j \neq k \text{ and } i = j \\ 0 & \text{otherwise} \end{cases}$$

The volume of $A^{(t)}$ is $\det(A^{(t)}) = t_k \in]0,1[$ and by constructions its axes are all valid translations, so A is indeed a new smaller cell. This process can then be iterated in the new Euclidean

space created by mapping each point $x \in \mathbb{Q}^3$ to $(A^{(t)})^{-1}x$, until no valid translation is left.

There are several ways to improve the computational complexity of this procedure. The first one consists in observing that if $t \in \mathbb{Q}^3$ is a valid translation, then so is $k \times t + o$ where $k \in \mathbb{Z}$ and $o \in \mathbb{Z}^3$. This allows to minimise the volume of the new cell $A^{(t)}$ thanks to algebraic considerations. Let's illustrate this point in two dimensions for simplicity, although it can be generalised to any number of dimensions.

Let $t = \left(\frac{p_1}{q_1}, \frac{p_2}{q_2}\right)$ be a valid translation. Let's assume that $p_1 \neq 0$ and $q_1 \geq q_2$ (the opposite reasonning can be done in the other case). Let $k \in \mathbb{Z}$ and $k' \in \mathbb{Z}$ be Bézout coefficients for p_1 and q_1, *i.e.* such that $kp_1 + k'q_1 = 1$, and let $o = (k', 0) \in \mathbb{Z}^2$. Then $k \times t + o = \left(\frac{1}{q_1}, \frac{kp_2}{q_2}\right)$.

Hence, matrix $A^{(kt+o)}$ can replace $A^{(t)}$ to reduce the cell to a volume $\frac{1}{q_1}$ instead of $\frac{p_1}{q_1}$.

Another potential improvement consists in using all the valid translations instead of stopping at the first one encountered. This way, the translation that minimises the volume of the new cell can be chosen instead of any valid translation. However the cost of collecting all valid translations can be up to n times higher than finding the first valid translation.

Furthermore, $A^{(t)}$ is just a special case of the set of possible new cells. In general, any invertible matrix whose columns are valid translations (including columns of the identity matrix) is the matrix of a valid cell. The best new cell is the one with the smallest volume, but combinatorial exploration of all possible such matrices that lead to smaller cells has a worst-case complexity of $O(n^3)$. In practice, many candidate matrices can be eliminated before computing their determinant so using this method is actually quite efficient.

2.4 Search space exploration

At this stage in the workflow, the graph has a unit cell and a corresponding equilibrium placement. Let's now look for the topological genome. The strategy follows that implemented in Systre, which consists in taking the lexicographically smallest key among those computed for a number of candidates. Both the computation of the key from a given candidate and the selection of the candidates should not depend on the current representation of the graph.

A candidate is defined as a triplet of linearly independent edges, and represented by a pair {vertex, basis of $\mathbb{Q}^3$} where the vertex is the source of the first edge and the columns of the basis are the translations induced by the edges. The process of selecting candidates is explained in section 3. This subsection focuses on the computation of a key from a given candidate.

This precise algorithm is explained in [15] and analysed as Algorithm 5 in [21]. It is detailed in Figure 4. The main point is that the algorithm creates a new representation of the graph which is independent of the previous one by redefining both the unit cell and the numbering of vertices thanks to a unique euclidean space derived from the candidate. The first element of the candidate, the vertex, is the origin of the new space and the columns of the basis are its three axes.

Step **(b)** is a simple affine transformation of the positions and steps **(c)** to **(h)** are straightforward to implement, they only require keeping track of the positions of the numbered vertices. Step **(j)** is also straightforward, it consists in multiplying each translation by the inverse of the matrix of the new unit cell to obtain an integer offset. This allows to retrieve the set of edges E of the graph, and since $V = [\![1, n]\!]$, this directly yields the representation of a periodic graph in the vector model.

Step **(i)** deserves special consideration. It consists in building a new unit cell from the set of positive translations t of the edges, positive in the sense of being lexicographically greater than $0_{\mathbb{Q}^3}$. Negative translations are negated to only keep positive ones. By construction, this set

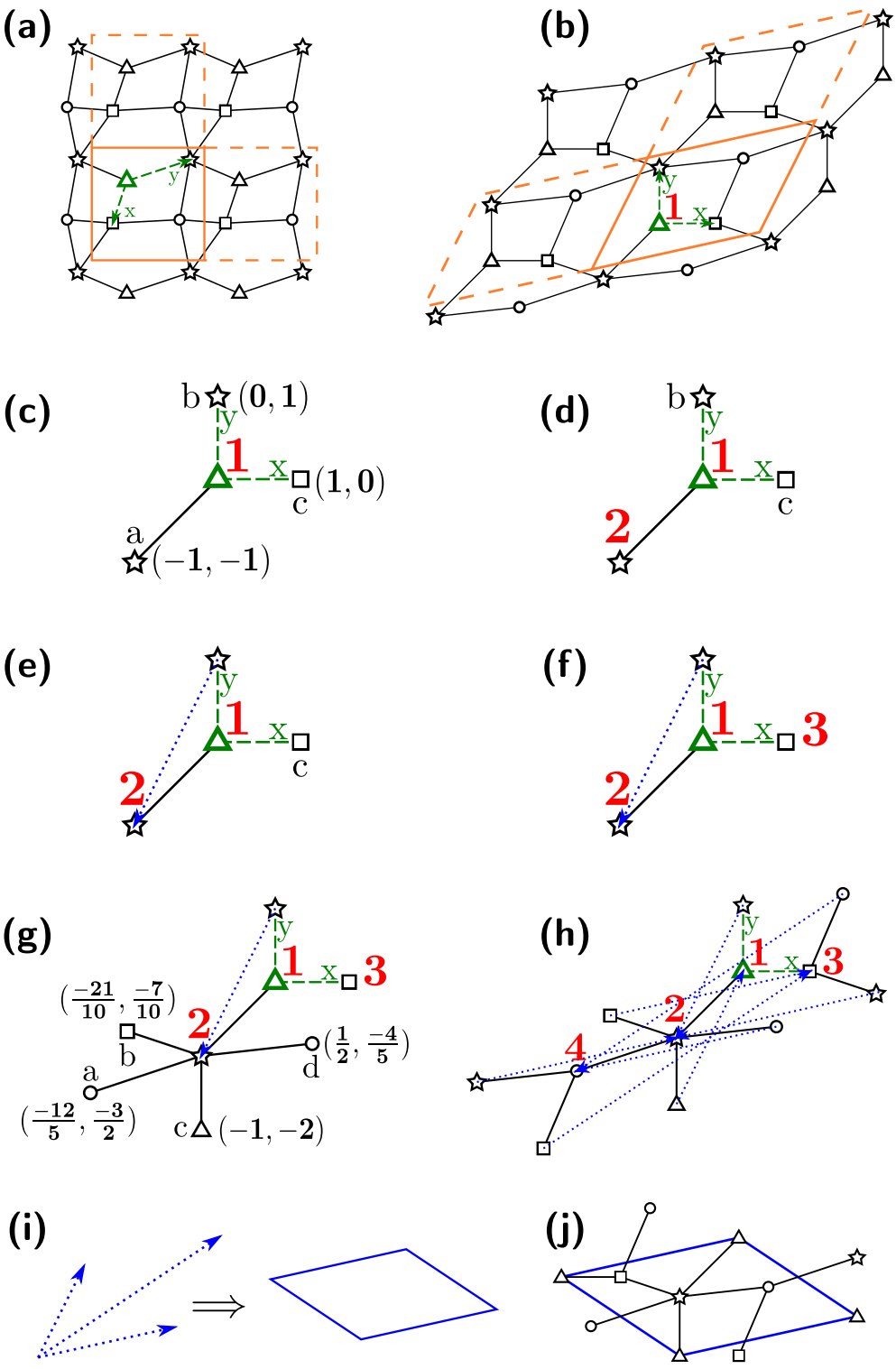


Figure 4: Illustration of the algorithm used to extract a key from a candidate on a 2-dimensional graph

(a) Start from a candidate (in green), whose origin is Δ and axes are x and y. Initialise an empty queue of vertices.

(b) Represent the graph in the basis given by the candidate (the former unit cell and two of its neighbours are represented in orange). Note that axes x and y are now orthogonal. Assign number 1 to the origin Δ and push it to the queue.

(c) Take the first vertex u of the queue and sort its neighbours according to the lexicographical order of their position. Here, $u = 1$ and the sorted neighbours are a, b and c.

(d) For each neighbour v', in order, if it is not the representative of an already encountered vertex, assign it the first available number between 1 and n and push it to the queue. In this case, neighbour a is assigned number 2.

(e) Otherwise, store the translation between the newly encountered representative v' and that which has been recorded v (the blue arrow). Here, v' is neighbour b and v is vertex 2: both are representatives of the same vertex.

(f) Continue until all the neighbours have been processed. In this case, only neighbour c remains and it is assigned number 3.

(g) Repeat the steps from **(c)** until the queue is empty. Here, the neighbours of 2 are being processed.

(h) When the queue is empty, all numbers between 1 and n have been assigned and a set of translations has been recorded.

(i) Triangulate the set of positive translations to obtain a new unit cell (see section 2.4).

(j) Convert the translations to integer offsets using the unit cell to find a new representation of the graph, and extract the key.

Figure 4: *Continued:* Illustration of the algorithm used to extract a key from a candidate on a 2-dimensional graph

does not depend on the previous representation of the graph, it only depends on the candidate. As a consequence, any deterministic algorithm that transforms a set of translations into a unit cell can be used for this purpose. Delgado-Friedrichs and O'Keeffe [15] mention this point and suggest triangulating the translations, without detailing their algorithm. Before presenting the one used here, let's sketch the general idea and prove a lemma.

Given their origin, translations of the set should correspond to integer offsets between vertices and their representatives in the new unit cell to be found. Thus, the matrix $B \in \mathbb{Q}^{3\times3}$ representing the unit cell must be such that each positive translation $t \in \mathbb{Q}^3$ of the set verifies:

$$B^{-1}t = o \in \mathbb{Z}^3 \tag{3}$$

Not all such basis B will do, since it should also represent a unit cell, hence the columns of B must be valid translations. Since all positive translations of the set map some vertex to one of its representatives, they are all valid translations, so it is tempting to look among the integer combinations of these translations to find the columns of B. However, it remains to be proven that such combinations indeed span the entire group of valid translations.

To show this, let's take the Euclidean space of step **(b)** and consider a path between the origin and any of its representatives. This path consists in a series of edges $(u, v, t) \in V \times V \times \mathbb{Q}^3$ where t is the translation between vertex v and the vertex v' which is the actual neighbour of u. After any number of steps, let's note $(u, \tau) \in V \times \mathbb{Q}^3$ the pair where u is identifier of the vertex on which the path has just landed and τ is the sum of the translations t encountered so far. The initial pair thus is $(1, 0_{\mathbb{Q}^3})$, since 1 is the number of the origin. An induction on the number of steps shows that the position of the exact vertex on which the path lands when reaching pair (u, τ) is that of vertex u translated by τ. As a consequence, when reaching the final vertex of the path, the pair is $(u_{\text{end}}, \tau_{\text{end}}) = (1, \tau_{\text{end}})$. Thus, by construction τ_{end} is the sum of translations t stored in the edges during the algorithm, so it is an integer combination of the positive translations of the set. But since this reasoning was valid for the translation between the origin and any of its representatives, this proves that any valid translation can be written as an integer combination of positive translations of the set.

Let's now proceed to the algorithm that converts the set of positive translations into a matrix B satisfying Equation 3.

First, note that most positive offsets present in the usual representations of periodic graphs are one of either $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$. As a consequence, in the most common cases, there are only three elements in the set of positive translations, which can simply be mapped to the three canonical offsets. This fact serves as a heuristic optimisation for the more general algorithm, by representing all the positive translations in the basis $A \in \mathbb{Q}^{3\times3}$ whose columns are the three first linearly independent positive translations. The reverse transformation will be applied at the end of the algorithm. Overall, this operation is not crucial for the algorithm to work and the input remains a set of translations, but it tends to reduce the denominators of the coefficients of the translations, which is computationally helpful.

Given a set of translations, the goal now is to find three combinations such that they form the columns of a matrix B satisfying Equation 3. In other words, these combinations must span the group of valid translations. This problem on rationals is lifted to integers by storing the least common multiple of the denominators of the coefficients of the translations columnwise, and pointwise multiplying the translations by this triplet of integers in order to obtain a set of integer translations.

From that point, several algorithms exist in order to solve the problem on this integer form [33–36]. The standard procedure is to build the rectangular matrix whose lines are the triplets representing integer translations and put it in Hermite normal form [43], which is

precisely such that it is built from an integer combination of the initial triplets and still spans the same group as that of the triplets.

Determining the exact Hermite normal form is useless in this case however, since there is no need for the complexity brought by its other properties, and the existing solvers are specialised to have their best performance on big square matrices, not thin rectangular matrices one of its dimensions being only 3. Thus, we implemented a custom solver for this particular problem which performs the following steps, for each column j in a fixed order:

- Find Bézout coefficients of the j-th element of the integer translations, as well as their greatest common divisor $d \in \mathbb{N}^*$. To do so, we implemented an efficient algorithm from Havas *et al.* [43].
- Set the j-th column of B as the sum of the integer translations weighted by their Bézout coefficient.
- For each integer translation $t \in \mathbb{Z}^3$, update it to zero its j-th element by subtracting the new column of B times the quotient of the j-th element of t by d. By construction, after this step all translations have at most $(j-1)$ non-zero elements.

Finally, columnwise divide B by the triplet of least common multiples that was found before to get back to rationals, and left-multiply it by A to go back to the Euclidean space specified by the candidate.

The resulting matrix B has all the required properties for being a unit cell. Thus, each translation $t \in \mathbb{Q}^3$ of an edge now corresponds to an offset $o = B^{-1}t \in \mathbb{Z}^3$ with respect to the unit cell, so the lexicographically ordered set of edges $(u, v, o) \in V \times V \times \mathbb{Z}^3$ is a representation of the initial graph that does not depend on its initial representation, but only on the candidate.

Let's note that this method transforms a set of translations into a unit cell, and this is precisely the problem encountered when minimising the cell from the set of all valid translations in subsection 2.3. However, in the current case, the procedure cannot rely on a starting cell to minimise, which is why it is more complex.

To conclude, let's observe that this representation might be impossible to embed, in the sense it may be impossible to put all the vertices of the graph, as they were determined by the algorithm, within the same unit cell given the constraints on their edges. This can be circumvented by taking appropriate representatives for each vertex given a fixed unit cell, but even that is unnecessary since the goal is only to extract a key, and it does not matter whether the key itself can be embedded.

3 Reduction of the search space

The only aspect of the algorithm that remains to be explored is the selection of the candidates, which is a crucial point for the overall complexity of the algorithm. After explaining how the implementation of Systre does this, we will present a series of novel reductions of the search space, *i.e.* of the number of candidates, that yield a notable performance gain.

3.1 The initial search space

Let's recall that a candidate is defined as a triplet of edges, which is used to create a new representation for the graph as explained in Figure 4. The topological genome is chosen as the lexicographically smallest key extracted from the candidates, so the set of candidates must be chosen in a fashion that is independent of the initial representation of the graph, lest the selection biases the search space. As a consequence, Delgado-Friedrichs and O'Keeffe [15] proposed the following selection pattern:

- If there exists at least one vertex such that its outgoing edges are not all coplanar, then the set of candidates is the set of triplets of linearly independent edges that all start at the same vertex.
- Otherwise, the candidates are all triplets of linearly independent edges such that the two first start from a common vertex and the third starts from another vertex.

In both cases, the set of valid candidates is not empty (otherwise it would mean that the graph is either not connected or not 3-periodic) and independent of the initial representation of the graph. It can be further reduced by removing duplicate candidates that yield the same key, using symmetry considerations as will be shown in subsection 3.3.

For many graphs, this search space remains rather large compared to the set of all possible triplets of edges. Our goal in this project was for our software to be able to tackle very big structures very efficiently, for two different types of applications: high-throughput screening of hypothetical materials, where the topological detection is performed on hundreds of thousands (and sometimes several millions) of structures [44], and application to very large periodic structures, such as models of amorphous framework materials [45]. Therefore, reducing this search space is crucial for performance since it directly tackles one of the bottlenecks in the original implementation of Systre.

3.2 Categorisation of vertices and edges

Since the search space must remain independent of the representation of the graph, it can only depend on some properties that are common to all its representations. Such a property is called a *topological invariant*. The only invariant used so far is whether there exists a vertex whose outgoing edges are not all coplanar, but many other invariants exist, such as the three used by ToposPro [13]. These properties can be used to considerably reduce the search space by separating vertices according to them.

The core idea of vertex categorisation consists in trying to separate vertices into as many categories as possible, each of these categories being identifiable independently of the representation of the graph. For instance, in this work, vertices are separated according to their coordination sequence (the number of vertices at distance k from the vertex of interest with $k = 1, 2, ...$, see Introduction), so that two vertices are in the same category if and only if they share the same coordination sequence up to distance 10.

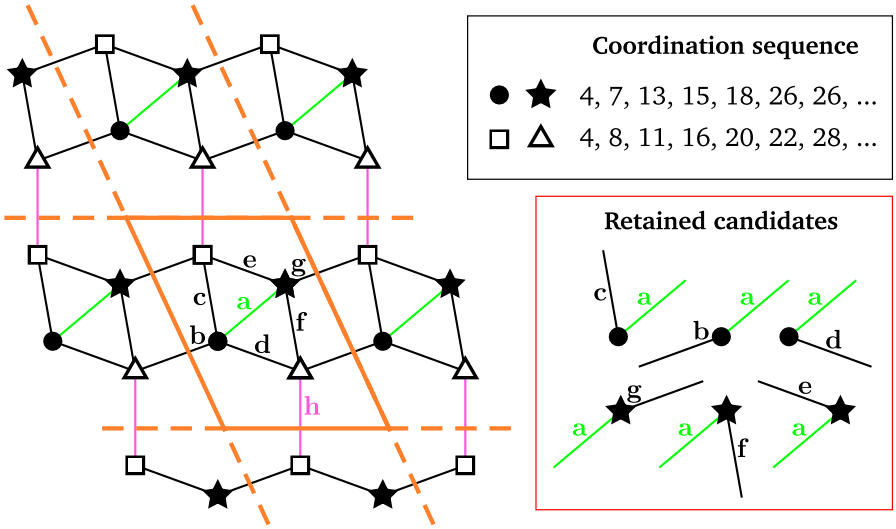
Vertex categories are then sorted by the total number of outgoing edges from vertices in that category and, in case of equality, by the invariant they were defined from. For example, in our case, ties are broken by lexicographically sorting according to the coordination sequences.

With these sorted categories in mind, let's look for candidates and try to minimise their number. The following strategy is used in our improved algorithm:

- Let c be the first vertex category such that there is a triplet of non-coplanar edges starting from the same vertex in c (if no such category exists, skip to the alternative below). Then, for all such triplets, let c_1, c_2 and c_3 be the sorted categories of the vertices to which the initial vertex is bonded with these edges. There are four subsets of such triplets, characterised by whether they satisfy:

  1. $c_1 = c_2 = c_3$.
  2. $c_1 < c_2 = c_3$.
  3. $c_1 = c_2 < c_3$.
  4. $c_1 < c_2 < c_3$.

  Then the candidates are all the triplets of edges whose categories are in the last non-empty such subset with (c_1, c_2, c_3) lexicographically minimal.
  This strategy illustrated for the 2D case in Figure 5.



1. The coordination sequence of all vertices is computed. In this particular case, there are two vertex categories, one represented with filled symbols (circle and star) and the other with empty symbols (square and triangle). Edges are colored in green when they connect two filled symbols, in magenta for two empty symbols and black otherwise.
2. Both categories have the same number of outgoing edges, so they are sorted in lexicographical order of the coordination sequences: the "filled" category comes first.
3. There exists a filled vertex with two linearly independent outgoing edges (here, all vertices actually match the criterion because no two neighboring edges are aligned). Hence, c is the "filled" category.
4. Consider all ordered pairs of linearly independent edges starting from a filled symbol u. There are 24 such pre-candidates, represented as the two first lines of Figure 6. Among those, only 6 bond u to vertices of category c_1 and c_2 respectively, such that $c_1 < c_2$. Keep those as candidates.

   If there was no such pre-candidate, the candidates would have been all the pre-candidates where $c_1 = c_2$ with c_1 minimal, *i.e.* "filled" if possible, "empty" otherwise.

Figure 5: Illustration of the candidate selection on the **cqe** net.

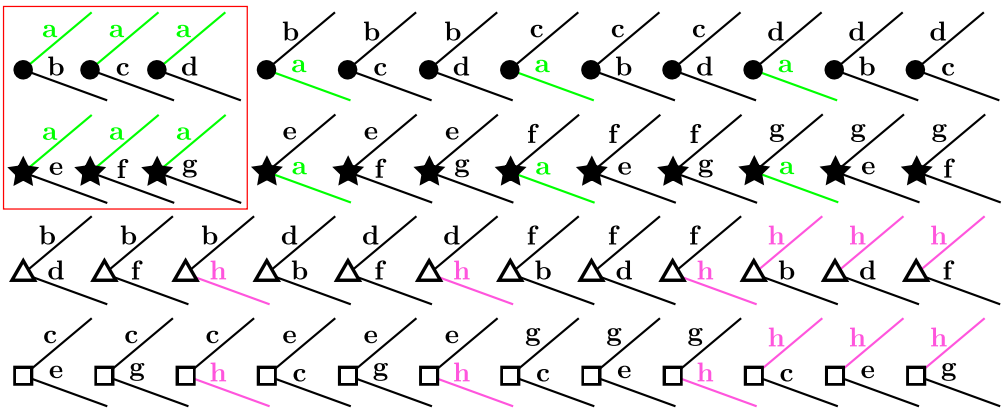


Figure 6: The 6 candidates of the **cqe** net selected by CrystalNets.jl, among the 48 possible candidates.

- Alternatively, if all vertices have only coplanar outgoing edges, let c be the first category such that there exists another category c' such that there is a non-coplanar triplet of edges, the two first of which start from a vertex of c and the last from c'. c' is chosen as the first such category. Let c_1, and c_2 be the sorted categories of the destinations of the two first vertices, and c_3 be the destination of the third. Then the candidates are all the triplets of edges, the two first of which start in c and the last from c' such that (c_1, c_2, c_3) is lexicographically minimal.

The main difference with the strategy of Systre is that the categorisation of vertices bounds the number of candidates with respect to the maximal size of a category, hence the goal of having as many small categories as possible. However, the two strategies are exactly the same in the worst-case scenario where there is only one category. Note that, although rather complex, the constraints on the retained candidates described above can actually be implemented in an efficient way, which is necessary to prevent the search space reduction from taking more time than the actual search space exploration.

3.3 Reduction through symmetry

In addition to the individual properties of vertices which are used to separate them into categories, symmetry considerations on the equilibrium placement can be useful in reducing the search space. Indeed, if two candidates are symmetry-related, then the representations extracted from both will be exactly identical. This optimisation is already used in Systre, and in the current work, it is useful to perform a symmetry analysis on the equilibrium placement of the unit cell before even categorising vertices.

Unfortunately, we do not know of any library that can find symmetries for an input given in arbitrary precision; all existing libraries take approximate positions, and thus return only approximate symmetries. We use the *spglib* library [41] to this purpose. *spglib* is a C library, which makes it easy to interface with Julia without sacrificing speed. Since all its inputs and outputs are approximate, the returned symmetries need to be checked against the exact equilibrium placement of the graph to rule out false positives. Because of the imprecision of the input, there might also be some false negatives – that is, some symmetries may be missing – but they do not matter because symmetries can only help to eliminate candidates that are symmetric images of other ones already encountered. Hence, the use of symmetries is purely an optimisation, but it has no influence on the result of the algorithm.

Symmetries are useful to avoid needlessly recomputing coordination sequences when dividing the vertices into categories. Indeed, the search space is reduced by singling out one vertex per symmetry orbit, and using the convention that only these can serve as the source of the first edge of a candidate. No such constraint can be placed on the destinations of the edges unfortunately, as that would be unsound.

On the particular example of the **cqe** net shown in Figure 5, there is only one symmetry and it maps each symbol of a category on the other of the same category. As a consequence, only one of the two first lines and one of the two last lines of candidates shown in Figure 6 need to be kept. Systre thus examines 24 candidates, and only 3 for CrystalNets.jl.

3.4 Ordering of keys

Finally, the algorithm can be made faster by changing the mode of comparison of the candidates. In Systre, the topological genome was defined as the smallest key extracted from a candidate. However, we observe that, arrived at step **(h)** of the algorithm in Figure 4, both the embedding of the graph as well as the ordering of its vertices only depend on the candidate. Hence, we can define at that point a pseudo-key consisting in the sorted list of triplets

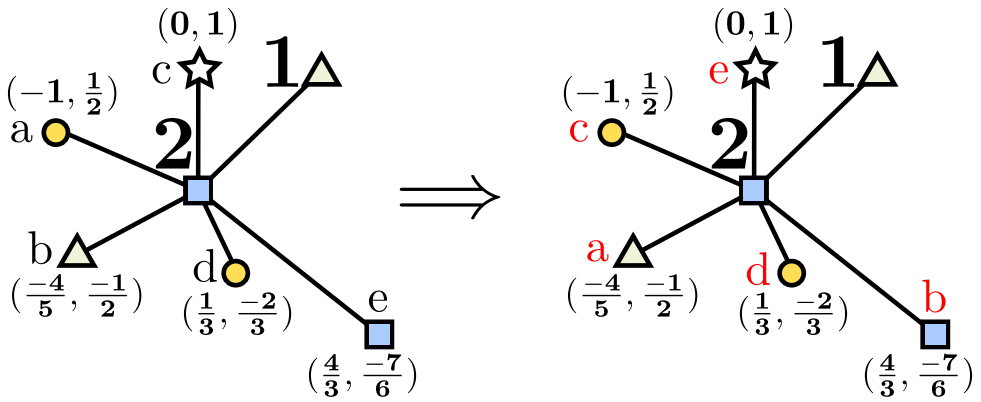


Figure 7: Difference between previous and new ordering of neighbours at step **(c)** on an example. Representatives of the same vertex have the same symbol.

$(s, d, t) \in V \times V \times \mathbb{Q}^3$ if there is an edge between the vertex defined by its number s and the representative of vertex number d which is translated from the latter by t. This pseudo-key only depends on the intermediate representation of the graph obtained in step **(h)**, which means it is exclusively determined by the candidate, so we can define the genome as the key extracted from the candidate with the smallest pseudo-key.

Instead of comparing the keys extracted after a costly lifting of the positive translations to a proper representation of the graph, it is thus sufficient to compare the pseudo-keys. Moreover, this pseudo-key can be computed incrementally during the main loop of the algorithm. To do so, at step **(c)**, the neighbours of vertex u should be sorted in a different manner (represented on Figure 7): all representatives of the same vertex should be contiguous and lexicographically sorted according to their position within each category; the categories corresponding to vertices which have a number (assigned in step **(d)**) are placed first and sorted according to this number, while the others are placed afterwards and sorted according to the position of the first vertex in the category. This makes it so that each edge of the graph is added to the pseudo-key in the order in which it is studied (in steps **(d)** – **(f)**), while preserving the invariant that the pseudo-key is lexicographically sorted.

Since the pseudo-key is constructed on-the-fly, it can thus be compared at each step with the previous smallest pseudo-key. If a triplet of the current pseudo-key is lexicographically greater than the triplet at the same position in the previous best pseudo-key, the computation for the candidate is aborted; if it is lower then this comparison can be skipped for the rest of the computation and the pseudo-key will be the new smallest. Finally, once all the candidates have been checked, the minimal pseudo-key is converted to an actual key through steps **(i)** and **(j)**, yielding the topological genome of the graph.

4 Application to materials databases

CrystalNets.jl is a Julia package which can be intuitively used from an interactive Julia shell or as an executable to determine the structure of single crystallographic files in the variety of file formats recognised by chemfiles[4]. Since chemical bonds are not systematically present in such files, it uses a custom bond detection algorithm inspired from that of chemfiles, and can export both the input with the detected bonds as well as the extracted underlying net to allow visually checking the correspondence with the input. The bonding algorithm reuses available information on the structure to accurately account for the larger bonds surrounding metals in

[4] full list available at https://chemfiles.org/chemfiles/latest/formats.html#list-of-supported-formats

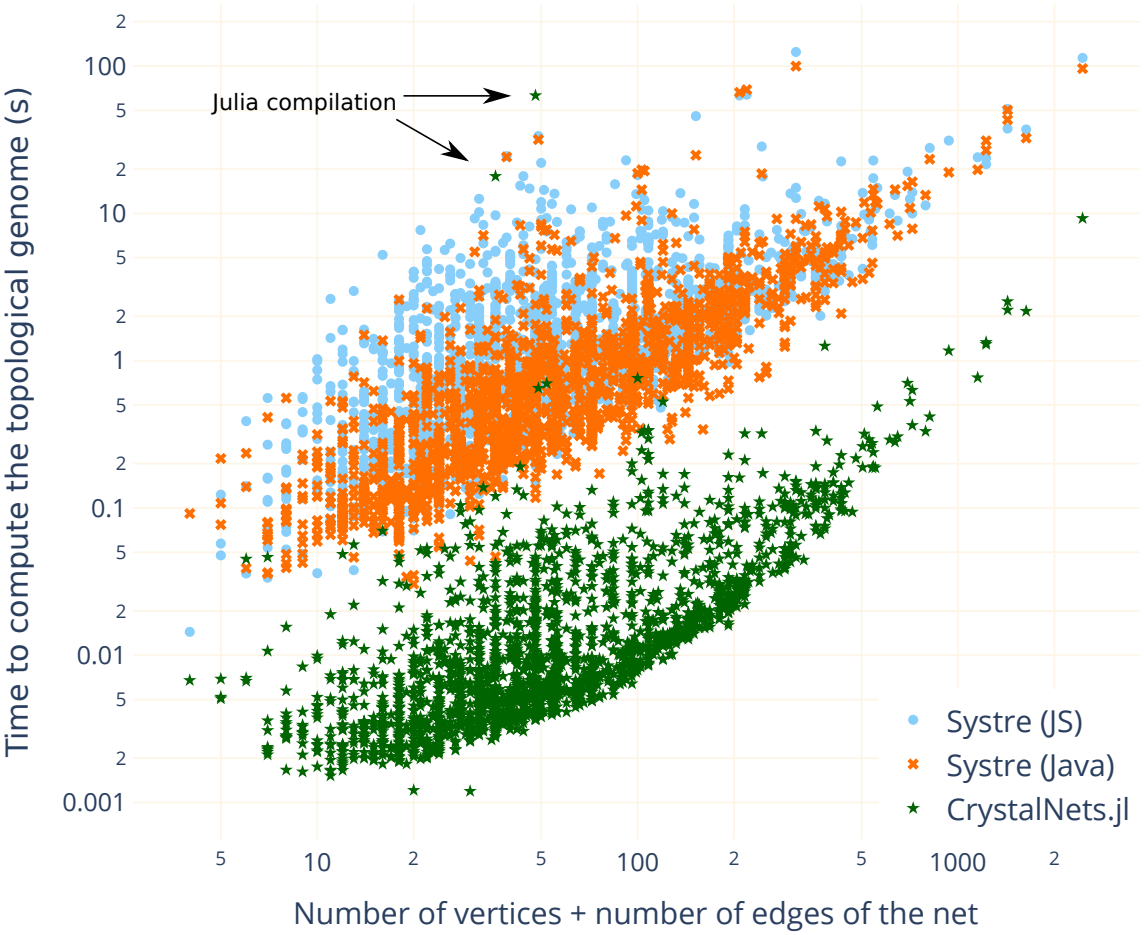


Figure 8: Comparison of processing times for 2,053 nets with Systre and CrystalNets.jl.

MOFs for instance, but the user can also override the cutoff distance for bonding if necessary.

By combining the bond detection heuristic with topology determination, CrystalNets.jl can also be used to survey large databases of crystalline structures, without the need for user input or relying on other chemical information (such as choice of SBUs) on the structures themselves. In this section, we demonstrate the code by exploring a series of available materials databases.

4.1 RCSR, EPINET and IZA-SC database

The Reticular Chemistry Structure Resource (RCSR) [7] [5] is a repository of 3,132 structures with 2 or 3-periodicity (excluding weaving and 0- and 1-periodic structures), each given a canonical name. It offers a Systre archive containing 2,928 nets that we reprocessed to obtain the 2,928 topological keys corresponding to each structure. These keys are not the same as those of the archive because of our algorithmic differences explained before.

We evaluated the performance of CrystalNets.jl by comparing its time to compute the topological keys of 2,053 structures of the RCSR chosen at random, compared to both Java and JavaScript (JS) implementations of Systre[6]. All three programs were executed 6 times with one input and 6 times with another, each input being a textual file containing a different representation of the same 2,053 nets. All ran on a single thread with Intel Xeon Silver 4210R

[5] available online at http://rcsr.anu.edu.au/nets (3D) and http://rcsr.anu.edu.au/layers (2D)
[6] available online at github.com/odf/gavrog and github.com/odf/systreKey respectively

CPUs (2.4 GHz). The average timing over the 12 executions is represented on Figure 8. On average, CrystalNets.jl is 56 times faster than the Java version of Systre, and 117 times faster than the its stripped-down JavaScript implementation. Two outliers are marked with a black arrow: these correspond to nets **sas** and **dhe**, the two first nets in the input file of dimension respectively 3 and 2. Since Julia is compiled Just-In-Time [29], it must compile the entire algorithm the first time the program computes the topological genome of a net, hence a significant overhead. Rerunning the computation on the same two nets after compilation yields an average timing of 3.9 ms for **sas** and 5.9 ms for **dhe**, compared to 63 s and 18 s respectively when including compilation, as represented on the figure.

This performance is consistent across the nets. On the **tep** net, which is the largest of the RCSR (920 vertices and 1840 edges per unit cell), CrystalNets.jl computes a topological genome in 8.6 s, compared to 229 s for the Java implementation of Systre, and 193 s for the JavaScript version. For the smallest net **pcu**, CrystalNets.jl takes 6.5 ms, which is twice as fast as the JavaScript implementation of Systre, and 16 times faster than the Java version. The maximum observed speedup is for the **tsl** net where CrystalNets.jl takes 7.1 ms on average compared to 5.2 s for the Java version of Systre and 16 s for the JavaScript version.

The RCSR partially overlaps with the larger EPINET database [10] [7] which contains, among other data, 14,532 stable 3D nets, which we processed similarly. Finally, both databases partially overlap that of Structure Commission of the International Zeolite Association (IZA-SC) [46], which provides names for 255 zeolitic framework structures and corresponding CIF files.[8] These structures have unambiguous vertices, defined by the tetrahedral atoms, and edges between two such atoms bonded by an oxygen, thus a uniquely-defined underlying net.

The results for all three databases are stored within CrystalNets.jl, so that they may be used to recognise the topology of structures processed by the code. In the following, we call *unknown* nets those that are not part of any of these databases — although it should be noted that other complementary databases of nets do exist (in particular the proprietary databases used by ToposPro).

Finally, we note that this processing of the databases exposed a few mismatches between the names of the structures given in the three databases, which were all reported. Namely, the RCSR nets **ucn**, **rad**, **jst**, **fuv** and **fvb** have as IZA-SC names SBN, ATS, JST, NPT and PTT, which did not appear on the RCSR. Additionally, we noticed that 137 structures from EPINET were missing their RCSR name, while 6 RCSR structures mentioned an incorrect EPINET counterpart. This highlights the need for systematic and automated checks of material and topology databases for consistency.

4.2 MOF structures

In order to test CrystalNets.jl on a realistic workload, we surveyed the CoRE MOF 2019 [47] database of metal–organic frameworks (version 1.1.3). To do so, we implemented a custom clustering algorithm to regroup atoms into SBUs. Since the definition of MOF SBUs is not universal [3] and IUPAC recommends providing complementary topological descriptions when necessary [48], we implemented several clustering options to account for different strategies used in the literature.

We provide five main clustering options: “all nodes”, “single nodes”, “standard”, “PE” and “PEM”. Except for the “standard” representation which is uniquely defined by its implementation in ToposPro [49, 50], the other clustering options are only described but not uniquely

[7] available online at http://epinet.anu.edu.au
[8] available online at http://www.iza-structure.org/databases/

determined by an explicit algorithm [8,48,49,51–53]. We also chose not to follow the original implementation of the "standard" representation [50], primarily because of the required computational cost of determining the minimal ring around each bond. Instead we use a custom algorithm whose main steps are as follows:

- First, organic cycles are replaced by a new virtual carbon atom. Afterwards, all metallic atoms are assigned to the "inorganic" class and all carbon atoms to the "organic" class. Then, in a loop, each remaining atom that is bonded to an atom in the "inorganic" class is itself classified as "inorganic", until no such atom remains. All remaining unassigned atoms are then classified as "organic". P and S atoms are treated differently and considered organic if directly bonded to a carbon, metallic if directly bonded to a metal, and part of a third class otherwise.

- Afterwards, SBUs are formed by regrouping together all adjacent atoms belonging to the same class. Paddle-wheel patterns are identified separately and regrouped into a single inorganic SBU even when the two opposite metal centres are not bonded together. The *points of extension*, which are organic atoms bonded to an inorganic atom, form their own SBU each. The algorithm stops here if there is no periodic SBU, *i.e.* if there is no infinite set of adjacent atoms belonging to the same class.

  Otherwise, if there are such periodic SBUs, they are split by virtually removing the atoms with the highest connectivity and regrouping the other atoms of the periodic SBU into the remaining connected components. A few other heuristics are used to ensure that all final SBUs are finite, and to tackle the various possible configurations encountered in framework structures.

This algorithm yields our "all nodes" representation [48]. It can be converted to "PE" (standing for "points of extension") [54,55] by removing all metallic atoms and bonding their corresponding points of extension to describe the coordination polyhedra around the removed metals. The "all nodes" representation is also convertible to "single nodes" [48] simply by merging the organic points of extension with their neighbouring organic SBU. It can also be transformed to yield the "PEM" (for "points of extension and metals") representation [53,54] by splitting inorganic clusters to keep only one metal atom per SBU. Combining the algorithm for "PEM" and "single nodes" yields our "standard" representation [49]. For simple cases, we also provide an "each vertex" clustering option which bypasses the entire clustering step and keeps each atom as its own SBU, as well as an "input" option which determines clusters directly from the `Residues` of the chemfiles-parsed input file, so that the user may provide their own SBUs.

Once the SBUs are determined, they are converted to vertices and bonded together according to the atomic bonds. Finally, as with all nets in CrystalNets.jl, if a vertex has one or zero neighbour, it is removed, and if it has exactly two neighbours it is converted into an edge between its neighbours, and this iteratively until convergence. Once vertices and edges are defined, interpenetrating nets are separated, each of them accounting for a single structure.

To simplify the presentation, we will only discuss the results when using the "single nodes" approach, as well as the "MOF" structure option which widens metallic bond radii. CrystalNets.jl recognises 61% of all 21,692 resulting structures of CoRE MOF 2019 as one of the nets in either the RCSR, the IZA-SC or the EPINET databases (only 33% structures are recognised with "PEM" and between 45% and 55% with the other clustering options). Among the unrecognised structures, less than 2% correspond to unstable nets which CrystalNets.jl does not handle. These proportions are similar for each of the four subcategories of the database, divided whether only free solvent was removed (FSR) or all solvent molecules (ASR), and within

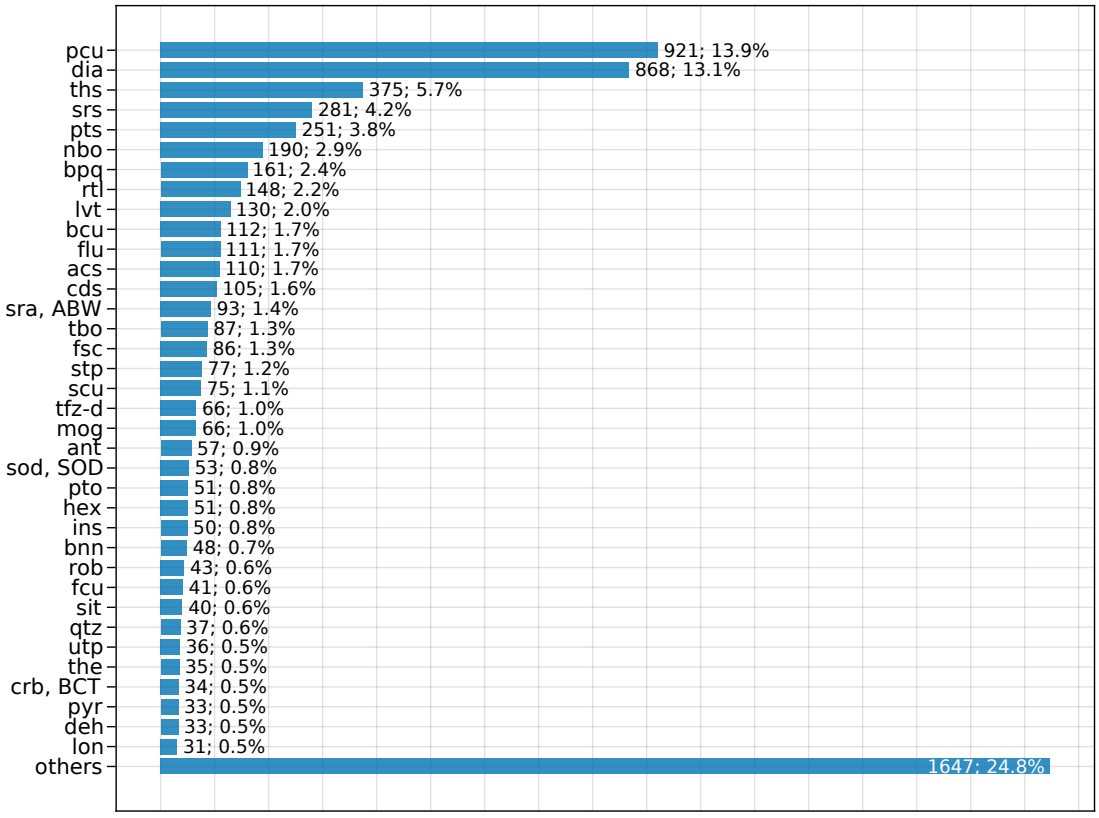


Figure 9: Number of structures per known net in the ordered structures of ASR [47].

each, whether the structure is disordered or not. The closeness of these numbers is expected since most structures from FSR also belong to ASR with only molecules of solvent as difference, which should not affect the topology. This highlights the robustness of the algorithm.

Next, we focus on the 11,190 ordered structures of the ASR subset to avoid duplicates. By studying the distribution of the number of structures per corresponding net, we observe that this distribution drops faster for unknown nets than for known ones. This is not surprising since we expect known nets to correspond to the most common occurring in nature, whereas unknown nets should only correspond to at most a few corresponding structures. Indeed, among the latter, 1,488 unknown nets only correspond to one MOF structure each, and 302 to two structures. Yet, 5 of them still correspond to thirty MOF structures or more each: we manually checked through the TopCryst interface [16] that these 5 nets were part of the ToposPro database, but we could not do so for all the unknown nets since these databases are not open. In comparison, for known nets, 110 of them only correspond to a single MOF structure, 54 to two and 38 to thirty or more each. These most represented nets are detailed in Figure 9: **pcu** (primitive cubic lattice), **dia** (diamond) are by far the most common, together accounting for 27% of know nets, followed by **ths**, **srs** and **pts**. For zeolitic nets, ABW, SOD (sodalite) and BCT are the most commonly encountered.

The high number of nets with known topologies is a good indicator of the quality of our clustering algorithm. Yet, since it partly relies on heuristics, it should not be considered fail-proof. To assess its validity, CrystalNets.jl exports both the input and the detected net in VTF format, which can be superposed in a molecular visualisation software to check the correspondence between the vertices of the nets and the expected clusters. We manually checked this correspondence for hundreds of structures with both known and unknown nets, to assert the validity of our implementation. Comparison with TopCryst on many instances comforts our

Name (CSD identifier)	CrystalNets.jl			ToposPro	
	Single/All Nodes	Standard	PE	Single/All Nodes	Standard (MOF)
HKUST-1 (LUDLED)	**tbo**	**tbo**	□	**tbo**	5,6T2
JXUST-1 (SAXVEW)	**pcu**	□	□	**pcu**	3,5,5,5T11
MIL-53 (MINVUA02)	**bpq/rna**	**bpq**	**sra**, ABW	∗	**bpq**
MIL-100 (BUSPIP)	**moo**	□	□	*Error: timeout*	
MIL-101 (OCUNAC)	**mtn-e**	□	**mtn-e-a**	*Error: timeout*	
MOF-5 (FIQCEN)	**tbo**	**tbo**	□	**tbo**	5,6T2
MOF-14 (QOWRAV)	**pto**	**pto**	sqc11259	**pto**	5,6T46
MOF-177 (BABRII)	□	□	□	3,3,5,6T36	4,4,4,4,4,6,8,8T1
MOF-801 (BOHKAM)	**fcu**	**xbi**	**ubt**	**fcu**	3,4,8T15
PCN-700 (RUBLAD)	**bcu**	□	**pcb**, ACO	**bcu**	3,3,4,6,8T9
UiO-66 (SURHEU)	**fcu**	**xbi**	**ubt**	**fcu**	3,4,8T15
ZIF-8 (FAWCEN)	**sod**, SOD	**sod**, SOD	**sod-e**	**sod**, SOD	**sod**, SOD
ZIF-67 (GITTOT02)	**sod**, SOD	**sod**, SOD	**sod-e**	**sod**, SOD	**sod**, SOD

Figure 10: Comparison of CrystalNets.jl and ToposPro [13] (through TopCryst [16]) for the identification of topologies on a selection of MOFs.
Topologies in bold come from the RCSR [7], in capitals from IZA-SC [46], sqc11259 comes from EPINET [11] and the rest are part of ToposPro databases. □ indicates unknown nets.
∗: ToposPro does not provide Single/All Nodes results for MIL-53 but returns **rna** as "Standard representation of covalent and ionic compounds" in addition to **bpq** as "Standard representation of coordination compounds and valence-bonded MOFs".

belief that the vast majority of unknown nets encountered in CoRE MOF are not the result of a wrong clustering by CrystalNets.jl, but simply missing from the databases we use, probably because they occur on only a handful of actual structures. On the other hand, the proprietary databases of ToposPro appear to include many such nets.

We compared CrystalNets.jl and ToposPro, accessed through its online TopCryst interface, on a selection of MOFs common in the literature. The result, presented on Figure 10, shows that both usually return the same topology if it is in the RCSR when using the "single nodes" or the 'all nodes' clustering options. On the other hand, the "standard" clustering leads to different results in several cases. This comes from the difference in clustering heuristics: our implementation regroups each paddle-wheel into a single SBU and removes SBUs made of a lone oxygen (while bonding their neighbors together) while ToposPro does not systematically do so. These two heuristics can be individually toggled off in CrystalNets.jl by the user if necessary. In practise, we observe that keeping these heuristics usually leads to less clusters without loss of chemical meaning, which can in turn help identify relevant underlying topologies, like the **xbi** net for MOF-801 and UiO-66 which contains the information on connectivity between individual metal atoms and with organic linkers. Having access to other clustering options, like "PE" also allows considering alternative topological descriptions of the topology.

ToposPro may fail for big nets because of the 3-minutes timeout of TopCryst: this happens for MIL-100 and MIL-101 among the tested structures. By comparison, CrystalNets.jl takes less than 10 s to handle them, but these timings are not comparable because we do not know the performance of the server on which the TopCryst webservice depends.

Since our analysis relies solely on CrystalNets.jl, which is independent from the tools used to build and clean the CoRE MOF database, it allowed us to identify 28 improper structures in the database, which were reported.

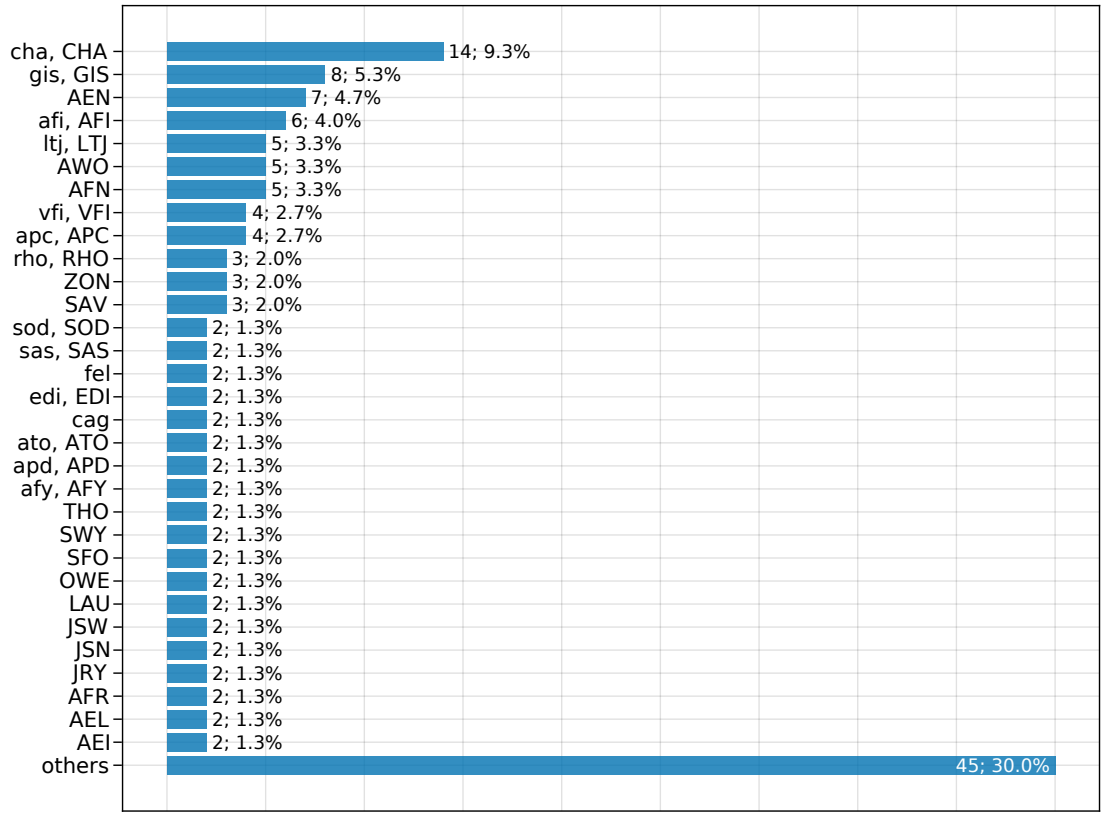


Figure 11: Number of structures per known net in the ALPO database [56].

4.3 Database of aluminophosphates

Additionally, we surveyed the topologies of Zheng, Li and Yu's database of aluminophosphates [56]. These 385 experimental structures are given as CIF files with little to no cleaning from the publications they stem from. In particular, solvent is included and the structures can contain multiple overlapping atoms with partial occupancy, while atoms of the main structure are often partially substituted. CrystalNets.jl automatically removes 0-dimensional residues like solvent and keeps for each site only the atom with the highest occupancy. Using the "Zeolite" structure option of CrystalNets.jl, we recognise 64% of the 205 3D nets in the database. This number rises to 73% by adding a heuristic to remove P–O–P and Al–O–Al linkages, which are usually not accounted for when computing the topologies of AlPOs.

The 31 known nets represented on more than one structure in the database are represented on Figure 11. With the exception of **fel** (originating from feldspars) and **cag** (occuring in variscite), they are all referenced in the database of zeolite frameworks of IZA-SC. This is expected since AlPOs are structurally close to zeolites, with the replacement of Si atoms by Al and P. Conversely, it shows that **fel** and **cag** may be nets underlying the structure of unreported zeolites, especially since both only have 4-coordinates vertices, corresponding to the T-atoms, and can be embedded to exhibit pores.

Most known structures have a net represented only once (45 nets *i.e.* 30%) or twice (19 nets *i.e.* 25%). In comparison, among the unknown nets, 1 occurs on three structures, 5 on two and the 32 others appear only once each. Both distribution tails and comparison of known and unknown nets follow a pattern similar to the CoRE MOF database.

For the structures for which the underlying net has been recognised as one of the IZA-

SC, there is little doubt on the validity of the detected net. For the others however, there is no systematic way of determining whether the detected net is indeed that representing the structure or not except from visually checking the adequate superposition of the detected net and the initial framework for each file. This limitation comes from the lack of cleaning of the input CIF files, which can lead CrystalNets.jl to interpret solvent residues as part of the framework for instance, or duplicate atoms as two different atoms instead of one. This issue can be circumvented by either cleaning the CIF inputs, as in the CoRE MOF database, or by manually checking the correspondence if working on individual files. Obviously, this limitation is inherent to the quality of the input files and affects all topology determination codes.

4.4 Hypothetical zeolites

The study of framework topology has historically been extensively applied to zeolites [57]. One reason is that the structure of zeolites, made only from tetrahedral (T = Si, Al, ...) atoms bridged by T–O–T linkages, is simple enough to show an almost one-to-one correspondence between each framework and its underlying net. Using this fact, large databases of hypothetical zeolites have been devised: the core idea consists in embedding 4-connected nets in real space, either by enumeration of the nets [58] or by Monte-Carlo sampling of vertex positions [59]. The vertices correspond to Si atoms while O atoms are placed along the edges. Each obtained structure is subsequently refined by minimizing its energy through molecular dynamics and with simulated annealing.

Since these structures are built from their underlying net, we tried to perform the opposite operation on the Predicted Crystallography Open Database (PCOD) [59] to assess our implementation. This database contains 331,172 CIF files representing zeolites with energies within 30 kJ/mol (per SiO_2 unit) of α-quartz, which were analyzed by CrystalNets.jl in 45 minutes using 20 cores (Intel Xeon Silver 4210R CPUs, 2.4 GHz). Bond guessing forced O atoms to have at most 2 neighbours, which is the default heuristic for O atoms not bound to metals: this incidentally ensures that all vertices of the net correspond to Si atoms.

As the SiO_4 tetrahedra are somewhat distorted during the refinement steps, the bonds guessed from the resulting structure do not always correspond to those from which it was built. As a result, we found 11 frameworks whose T-atoms were not all 4-connected, and 75 other frameworks whose coordination sequences did not match that recorded in the database. Two structures were found to have the same underlying net: 8013876 and 8035701, the latter having different detected coordination sequences than indicated. Barring these 86 mismatched structures, 18 nets only have a 2D periodicity and 31 are unstable nets not handled by CrystalNets.jl. All those "problematic" nets represent less than 0.04% of all nets in the database. For the rest, no two structures had the same underlying net.

Indeed, by construction of the database, structures whose topology is already part of the database are removed. The invariants used to identify the topology are the coordination sequences and the ring sizes: this strategy does ensure that no two structures with the same net are present, but it is too restrictive since two distinct nets could share these invariants. In the future, the use of a unique topological identifier through CrystalNets.jl could thus potentially lead to the generation of more structures.

5 Conclusion and perspectives

We have presented CrystalNets.jl, a Julia library implementing topology identification on crystalline materials. The core algorithm for topology detection is strongly inspired from that used in Systre [15] which computes a topological genome, that is a sequence of numbers uniquely

identifying the net underlying the given structure. The algorithmic and implementation improvements of CrystalNets.jl still preserve the soundness and polynomial time complexity of the initial algorithm, while offering a much better performance across all kinds of nets: it is on average 56 times faster than Systre and up to thousands of times faster in some cases. Moreover, our software directly takes crystallographic files as input and can automatically detect bonds if they are absent; it also includes a general clustering algorithm with different target options for framework materials, and heuristics adapted to metal–organic frameworks specifically.

Overall, CrystalNets.jl was successfully tested against a large diversity of crystalline structures: MOFs, aluminophosphates, zeolites and many other crystals. Even on large databases, our implementation shows great performance and can reliably recognise any structure in the Systre archive of the RCSR, in the IZA-SC database of zeolite structures and in EPINET. It successfully attributes known topologies to 60% of the structures in CoRE MOF after automatically clustering the vertices according to the "single nodes" policy. It was also able to recognise 73% aluminophosphate structures from Zheng *et al.*'s database [56], automatically removing solvent residues and colliding atoms when possible.

We believe CrystalNets.jl can be used as a faster and more convenient alternative to Systre and ToposPro for the identification of nets underlying crystalline structures. Its programmatic interface can be integrated to any computational pipeline written in Julia, as well as some other languages through bindings. Looking forward, we plan on providing a website interface similar to TopCryst [16], which could be useful to compare any structure to one of those surveyed in the previously exposed databases. We also plan on including results from the topological survey of the COD [60] and CSD [61] databases. The latter could then serve as comparison for our bonding and clustering algorithms against those of ToposPro [50].

Acknowledgments

We acknowledge financial support from the Agence Nationale de la Recherche under project "MATAREB" (ANR-18-CE29-0009-01) and from ANRT CIFRE project 2021/1151 in association with Air Liquide.

References

[1] N. W. Ockwig, O. Delgado-Friedrichs, M. O'Keeffe and O. M. Yaghi, *Reticular chemistry: Occurrence and taxonomy of nets and grammar for the design of frameworks*, ChemInform **36**(24) (2005), doi:10.1002/chin.200524292.

[2] O. M. Yaghi, M. O'Keeffe, N. W. Ockwig, H. K. Chae, M. Eddaoudi and J. Kim, *Reticular synthesis and the design of new materials*, Nature **423**(6941), 705 (2003), doi:10.1038/nature01650.

[3] D. J. Tranchemontagne, J. L. Mendoza-Cortés, M. O'Keeffe and O. M. Yaghi, *Secondary building units, nets and bonding in the chemistry of Metal–Organic frameworks*, Chem. Soc. Rev. **38**(5), 1257 (2009), doi:10.1039/b817735j.

[4] H. Furukawa, Y. B. Go, N. Ko, Y. K. Park, F. J. Uribe-Romo, J. Kim, M. O'Keeffe and O. M. Yaghi, *Isoreticular expansion of Metal–Organic frameworks with triangular and square building units and the lowest calculated density for porous crystals*, Inorg. Chem. **50**(18), 9147 (2011), doi:10.1021/ic201376t.

[5] S. Bureekaew, V. Balwani, S. Amirjalayer and R. Schmid, *Isoreticular isomerism in 4,4-connected paddle-wheel Metal–Organic frameworks: Structural prediction by the reverse topological approach*, CrystEngComm **17**(2), 344 (2015), doi:10.1039/c4ce01574f.

[6] J. Kim, B. Chen, T. M. Reineke, H. Li, M. Eddaoudi, D. B. Moler, M. O'Keeffe and O. M. Yaghi, *Assembly of Metal–Organic frameworks from large organic and inorganic secondary building units: New examples and simplifying principles for complex structures*, J. Am. Chem. Soc. **123**(34), 8239 (2001), doi:10.1021/ja010825o.

[7] M. O'Keeffe, M. A. Peskov, S. J. Ramsden and O. M. Yaghi, *The reticular chemistry structure resource (RCSR) database of, and symbols for, crystal nets*, Acc. Chem. Res. **41**(12), 1782 (2008), doi:10.1021/ar800124u.

[8] M. O'Keeffe and O. M. Yaghi, *Deconstructing the crystal structures of Metal–Organic frameworks and related materials into their underlying nets*, Chem. Rev. **112**(2), 675 (2012), doi:10.1021/cr200205j.

[9] S. R. Batten, *Topology and interpenetration*, In *Metal-Organic Frameworks: Design and Application*, pp. 91–130. John Wiley & Sons, MacGillivray, L. R. (2010).

[10] S. Hyde, O. D. Delgado-Friedrichs, S. Ramsden and V. Robins, *Towards enumeration of crystalline frameworks: The 2D hyperbolic approach*, Solid State Sci. **8**(7), 740 (2006), doi:10.1016/j.solidstatesciences.2006.04.001.

[11] S. Ramsden, V. Robins and S. Hyde, *Three-dimensional Euclidean nets from two-dimensional hyperbolic tilings: Kaleidoscopic examples*, Acta Cryst. A **65**(2), 81 (2009), doi:10.1107/s0108767308040592.

[12] J. C. Tan, T. D. Bennett and A. K. Cheetham, *Chemical structure, network topology, and porosity effects on the mechanical properties of Zeolitic Imidazolate Frameworks*, Proc. Nat. Acad. Sci. **107**(22), 9938 (2010), doi:10.1073/pnas.1003205107.

[13] V. A. Blatov, A. P. Shevchenko and D. M. Proserpio, *Applied topological analysis of crystal structures with the program package ToposPro*, Cryst. Growth Des. **14**(7), 3576 (2014), doi:10.1021/cg500498k.

[14] V. A. Blatov, *Multipurpose crystallochemical analysis with the program package TOPOS*, IUCr CompComm Newsletter **7**(4), 4 (2006).

[15] O. Delgado-Friedrichs and M. O'Keeffe, *Identification of and symmetry computation for crystal nets*, Acta Cryst. A **59**(4), 351 (2003), doi:10.1107/s0108767303012017.

[16] E. V. Alexandrov, A. P. Shevchenko and V. A. Blatov, *Topological databases: Why do we need them for design of coordination polymers?*, Cryst. Growth Des. **19**(5), 2604 (2019), doi:10.1021/acs.cgd.8b01721.

[17] S. R. Hall, F. H. Allen and I. D. Brown, *The crystallographic information file (CIF): A new standard archive file for crystallography*, Acta Cryst. A **47**(6), 655 (1991), doi:10.1107/s010876739101067x.

[18] O. Delgado-Friedrichs and M. O'Keeffe, *Crystal nets as graphs: Terminology and definitions*, J. Sol. State Chem. **178**(8), 2480 (2005), doi:10.1016/j.jssc.2005.06.011.

[19] V. Blatov, M. O'keeffe and D. Proserpio, *Vertex-, face-, point-, Schläfli-, and Delaney-symbols in nets, polyhedra and tilings: Recommended terminology*, CrystEngComm **12**(1), 44 (2010), doi:10.1039/b910671e.

[20] O. Delgado-Friedrichs, S. T. Hyde, M. O'Keeffe and O. M. Yaghi, *Crystal structures as periodic graphs: The topological genome and graph databases*, Struct. Chem. **28**(1), 39 (2017), doi:10.1007/s11224-016-0853-3.

[21] O. Delgado-Friedrichs, *Barycentric drawings of periodic graphs*, In *International Symposium on Graph Drawing*, pp. 178–189. Springer (2003).

[22] M. Treacy, M. Foster and K. Randall, *An efficient method for determining zeolite vertex symbols*, Micropor. Mesopor. Mater. **87**(3), 255 (2006), doi:10.1016/j.micromeso.2005.08.024.

[23] V. A. Blatov, O. Delgado-Friedrichs, M. O'Keeffe and D. M. Proserpio, *Three-periodic nets and tilings: Natural tilings for nets*, Acta Cryst. A **63**(5), 418 (2007), doi:10.1107/s0108767307038287.

[24] K. Theisen, B. Smit and M. Haranczyk, *Chemical hieroglyphs: Abstract depiction of complex void space topology of nanoporous materials*, Journal of chemical information and modeling **50**(4), 461 (2010), doi:10.1021/ci900451v.

[25] A. Gandhi and M. F. Hasan, *A graph theoretic representation and analysis of zeolite frameworks*, Comput. Chem. Eng. **155**, 107548 (2021), doi:10.1016/j.compchemeng.2021.107548.

[26] D. Weininger, *SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules*, J. Chem. Inf. Comput. Sci. **28**(1), 31 (1988), doi:10.1021/ci00057a005.

[27] S. R. Heller, A. McNaught, I. Pletnev, S. Stein and D. Tchekhovskoi, *InChI, the IUPAC international chemical identifier*, J. Cheminformatics **7**(1), 1 (2015), doi:10.1186/s13321-015-0068-4.

[28] J. Bezanson, A. Edelman, S. Karpinski and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM review **59**(1), 65 (2017), doi:10.1137/141000671.

[29] J. Bezanson, J. Chen, B. Chung, S. Karpinski, V. B. Shah, J. Vitek and L. Zoubritzky, *Julia: Dynamism and performance reconciled by design*, Proceedings of the ACM on Programming Languages **2**(OOPSLA), 1 (2018), doi:10.1145/3276490.

[30] S. J. Chung, T. Hahn and W. Klee, *Nomenclature and generation of three-periodic nets: The vector method*, Acta Cryst. A **40**(1), 42 (1984), doi:10.1107/s0108767384000088.

[31] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis and S. Muthukrishnan, *One trillion edges: Graph processing at Facebook-scale*, Proceedings of the VLDB Endowment **8**(12), 1804 (2015), doi:10.14778/2824032.2824077.

[32] T. Granlund et. al., *The GNU Multiple-Precision (GMP) arithmetic library*, https://gmplib.org/.

[33] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann and G. Villard, *Solving sparse rational linear systems*, In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pp. 63–70, doi:10.1145/1145768.1145785 (2006).

[34] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann and G. Villard, *Faster inversion and other black box matrix computations using efficient block projections*, In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, pp. 143–150, doi:10.1145/1277548.1277569 (2007).

[35] J. D. Dixon, *Exact solution of linear equations using P-adic expansions*, Numer. Math. **40**(1), 137 (1982), doi:10.1007/bf01459082.

[36] E. Kaltofen and B. D. Saunders, *On Wiedemann's method of solving sparse linear systems*, In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pp. 29–38. Springer, doi:10.1007/3-540-54522-0_93 (1991).

[37] O. Delgado-Friedrichs, S. T. Hyde, S.-W. Mun, M. O'Keeffe and D. M. Proserpio, *Nets with collisions (unstable nets) and crystal chemistry*, Acta Cryst. A **69**(6), 535 (2013), doi:10.1107/s0108767313020655.

[38] B. Delaunay, *Neue Darstellung der geometrischen Kristallographie*, Z. Kristallogr. Cryst. Mater. **84**(1-6), 109 (1933), doi:10.1524/zkri.1933.84.1.109.

[39] P. Niggli, W. Wien and F. Harms, *Handbuch Der Experimentalphysik*, Akademische Verlagsgesellschaft, Leipzig (1928).

[40] A. Santoro and A. Mighell, *Determination of reduced cells*, Acta Cryst. A **26**(1), 124 (1970), doi:10.1107/s0567739470000177.

[41] A. Togo and I. Tanaka, *Spglib: A software library for crystal symmetry search*, arXiv preprint arXiv:1808.01590 (2018), 1808.01590.

[42] R. W. Grosse-Kunstleve, N. K. Sauter, N. W. Moriarty and P. D. Adams, *The Computational Crystallography Toolbox: Crystallographic algorithms in a reusable software framework*, J. Appl. Cryst. **35**(1), 126 (2002), doi:10.1107/s0021889801017824.

[43] G. Havas, B. S. Majewski and K. R. Matthews, *Extended gcd and Hermite normal form algorithms via lattice basis reduction*, Exp. Math. **7**(2), 125 (1998), doi:10.1080/10586458.1998.10504362.

[44] E. Ren, P. Guilbaud and F.-X. Coudert, *High-throughput computational screening of nanoporous materials in targeted applications*, arXiv preprint arXiv:2202.09886 (2022), 2202.09886.

[45] N. Castel and F.-X. Coudert, *Atomistic Models of Amorphous Metal–Organic Frameworks*, J. Phys. Chem. C (2022), doi:10.1021/acs.jpcc.2c01091.

[46] C. Baerlocher and L. B. McCusker, *Database of Zeolite Structures*, https://europe.iza-structure.org/IZA-SC/ftc_table.php.

[47] Y. G. Chung, E. Haldoupis, B. J. Bucior, M. Haranczyk, S. Lee, H. Zhang, K. D. Vogiatzis, M. Milisavljevic, S. Ling, J. S. Camp, B. Slater, J. I. Siepmann *et al.*, *Advances, Updates, and Analytics for the Computation-Ready, Experimental Metal–Organic Framework Database: CoRE MOF 2019*, J. Chem. Eng. Data **64**(12), 5985 (2019), doi:10.1021/acs.jced.9b00835.

[48] C. Bonneau, M. O'Keeffe, D. M. Proserpio, V. A. Blatov, S. R. Batten, S. A. Bourne, M. S. Lah, J.-G. Eon, S. T. Hyde and S. B. Wiggin, *Deconstruction of crystalline networks into underlying nets: Relevance for terminology guidelines and crystallographic databases*, Cryst. Growth Des. **18**(6), 3411 (2018), doi:10.1021/acs.cgd.8b00126.

[49] A. P. Shevchenko and V. A. Blatov, *Simplify to understand: How to elucidate crystal structures?*, Struct. Chem. **32**(2), 507 (2021), doi:10.1007/s11224-020-01724-4.

[50] E. Alexandrov, V. Blatov, A. Kochetkov and D. Proserpio, *Underlying nets in three-periodic coordination polymers: Topology, taxonomy and prediction from a computer-aided analysis of the Cambridge Structural Database*, CrystEngComm **13**(12), 3947 (2011), doi:10.1039/c0ce00636j.

[51] M. Li, D. Li, M. O'Keeffe and O. M. Yaghi, *Topological analysis of Metal–Organic frameworks with polytopic linkers and/or multiple building units and the minimal transitivity principle*, Chem. Rev. **114**(2), 1343 (2014), doi:10.1021/cr400392k.

[52] B. J. Bucior, A. S. Rosen, M. Haranczyk, Z. Yao, M. E. Ziebel, O. K. Farha, J. T. Hupp, J. I. Siepmann, A. Aspuru-Guzik and R. Q. Snurr, *Identification schemes for Metal–Organic frameworks to enable rapid search and cheminformatics analysis*, Cryst. Growth Des. **19**(11), 6682 (2019), doi:10.1021/acs.cgd.9b01050.

[53] L. S. Xie, E. V. Alexandrov, G. Skorupskii, D. M. Proserpio and M. Dincă, *Diverse π–π stacking motifs modulate electrical conductivity in tetrathiafulvalene-based metal–organic frameworks*, Chem. Sci. **10**(37), 8558 (2019), doi:10.1039/c9sc03348c.

[54] F. M. A. Noa, M. Abrahamsson, E. Ahlberg, O. Cheung, C. R. Göb, C. J. McKenzie and L. Öhrström, *A unified topology approach to dot-, rod-, and sheet-MOFs*, Chem **7**(9), 2491 (2021), doi:10.1016/j.chempr.2021.07.006.

[55] A. Schoedel, M. Li, D. Li, M. O'Keeffe and O. M. Yaghi, *Structures of metal–organic frameworks with rod secondary building units*, Chem. Rev. **116**(19), 12466 (2016), doi:10.1021/acs.chemrev.6b00346.

[56] C. Zheng, Y. Li and J. Yu, *Database of open-framework aluminophosphate structures*, Sci. Data **7**(1) (2020), doi:10.1038/s41597-020-0452-4.

[57] J. V. Smith, *Topochemistry of zeolites and related materials. 1. Topology and geometry*, Chem. Rev. **88**(1), 149 (1988), doi:10.1021/cr00083a008.

[58] M. Treacy, I. Rivin, E. Balkovsky, K. Randall and M. Foster, *Enumeration of periodic tetrahedral frameworks. II. Polynodal graphs*, Micropor. Mesopor. Mater. **74**(1-3), 121 (2004), doi:10.1016/j.micromeso.2004.06.013.

[59] R. Pophale, P. A. Cheeseman and M. W. Deem, *A database of new zeolite-like materials*, Phys. Chem. Chem. Phys. **13**(27), 12407 (2011), doi:10.1039/c0cp02255a.

[60] S. Gražulis, D. Chateigner, R. T. Downs, A. F. T. Yokochi, M. Quirós, L. Lutterotti, E. Manakova, J. Butkus, P. Moeck and A. Le Bail, *Crystallography Open Database – an open-access collection of crystal structures*, J. Appl. Cryst. **42**(4), 726 (2009), doi:10.1107/s0021889809016690.

[61] C. R. Groom, I. J. Bruno, M. P. Lightfoot and S. C. Ward, *The Cambridge structural database*, Acta Cryst. B **72**(2), 171 (2016), doi:10.1107/s2052520616003954.