

Generating Geometric Graphs Using Automorphisms

Gregory McColm

Department of Mathematics & Statistics
University of South Florida

Abstract

We outline an algorithm for generating nets, a “net” being a geometric graph embedded in a space on which some group acts transitively. We prove that any connected net in such a space can be generated by this process, which can also be regarded as a covering of the net by copies of a connected “transversal” subnet. This paper outlines some of the underlying theory supporting a crystal prediction and design program currently under development. The paper begins with some background in geometric group theory.

Submitted: October 2010	Reviewed: January 2012	Revised: March 2012	Reviewed: May 2012	Revised: July 2012
	Accepted: July 2012	Final: August 2012	Published: August 2012	
	Article type: Regular Paper	Communicated by: G. Liotta		

1 Introduction

The problem of generating a highly symmetric graph embedded in some geometric space is not only of theoretical interest, but has applications to the computer generation of “skeleton graphs” of geometric complexes. If we modelled these embedded graphs as geometric complexes, we can use the algebraic machinery of graph theory, of discrete or computational geometry, or even of combinatorial or geometric group theory, to develop algorithms for computer generation of the complexes outlined by these embedded graphs.

These embedded graphs are often called *nets*. Our primary tools for manipulating these nets are their *symmetries*, i.e., their automorphisms that are induced by symmetries of the underlying geometric space. Our primary interest is in nets embedded in some metric space, and a symmetry of such a net would be an automorphism induced by an isometry of the underlying metric space. For any such net, the symmetries form a group, which we can call its *symmetry group*.

The specific goal is to generate an entire net, given its symmetry group and a (small) connected subnet – and we would like the vertices to have labels that would allow us to reconstruct the generation of this net for subsequent analysis. We use the symmetry group of the net to generate copies of the subnet that collectively cover the net; from the computer’s point of view, starting with the original subnet, one generates the entire net. We would need the initial subnet to be a transversal in the sense that it intersects each orbit of the net’s symmetry group exactly once. We then apply the symmetry group to the subnet to generate sufficiently many copies of that subnet to cover the original net. The procedure is a generalization of a process for generating “uninodal nets” as described in [10], and can be regarded as a generalization of the process of generating “objective structures” as described in [18].

We have under development a program (described in [23] and discussed in the last section of this article), which takes a *possible* transversal subnet and generates subsequent copies of fragments of that transversal, one vertex at a time. This program is based on a variant of the process employed in combinatorial and geometric group theory for traversing a vertex transitive graph using words encoding paths as in [29], [16], and [31]; see [12] and [24]. The program assigns to each edge $\{x, y\}$ a pair of symmetries $\Gamma(x, y)$ and $\Gamma(y, x)$ of the underlying space in which the net is embedded – and not necessarily (automorphisms) of the net itself, where $\Gamma(x, y)(x) = y$ and $\Gamma(y, x)(y) = x$. In a sense, we treat these symmetries as trains, with vertices as stations, and we can imagine a traveler taking symmetries from vertex to vertex, thus traversing the net. Starting with a putative transversal, with associated symmetries, as an initial condition, we can use algebraic conjugacies to generate an entire net, together with the space symmetries $\Gamma(x, y)$ for traversing that entire net. One eventually or at the end of time obtains the net and a “traversal function” Γ for the entire net.

As in group or semigroup theory, we obtain words encoding walks, and construct an “assignment function” Γ^* so that if the word \mathbf{w} encodes a walk from a vertex x to a vertex y , then $\Gamma^*(\mathbf{w})$ is a symmetry of the underlying space

(but not necessarily of the net itself) so that $\Gamma^*(\mathbf{w})(x) = y$. We will define the traversal function Γ in terms of this assignment function Γ^* . Again, we hope to carry out subsequent computations on the net – or more precisely, on the (words encoding the) sequences of iterations used to generate the net – we would like the vertices to be labeled so that we can tell from the label how the vertex was generated. The primary issues in this article are:

1. We would like to know that if we started out with a transversal subnet within a net, with space symmetries assigned in a fashion consistent with the symmetry group of that net, the procedure would indeed generate the net we started with.
2. We would like Γ (and Γ^*) to reflect the symmetry of the net insofar that if \mathbf{w} encodes a walk from x to y , where x and y are in the same orbit of vertices (under the symmetry group of the net), we have $\Gamma^*(\mathbf{w})$ being a symmetry of the net itself, not just of the underlying space.
3. In addition, if there is a symmetry of the net from one edge, $\{x_1, y_1\}$, to another, $\{x_2, y_2\}$, via a symmetry sending $x_1 \mapsto x_2$ and $y_1 \mapsto y_2$, we would like $\Gamma(x_1, y_1)$ to be something like a conjugate of $\Gamma(x_2, y_2)$.

The purpose of this article is to deal these desiderata:

1. We will confirm that using the assignment function we construct, this procedure generates the net itself.
2. We will confirm that if \mathbf{w} is a word encoding a walk from x to y , where x and y are in the same orbit, then the “traversal function” Γ we construct will have $\Gamma^*(\mathbf{w})$ being a symmetry of the net.
3. We will only obtain a partial result for the third desideratum, showing that there exist symmetries ϕ and ρ of the net such that $\rho\Gamma(x_1, y_1) = \phi\Gamma(x_2, y_2)$. We leave as an open question when the ρ can be dispensed with.

We conclude this article with a discussion of extant and planned implementations of this algorithm, which we call the *Crystal Turtlebug* as the computer programs based on this article employ a three dimensional analogue of the “turtle geometry” of [1]. In the last section we will discuss implementing this algorithm, from both a broad and theoretical point of view, (including a cursory discussion of its computational complexity), and from the point of view of the *Crystal Turtlebug* program.

1.1 State of the Art

Before going into the algorithm, we briefly describe the context: this algorithm was developed as part of a project to design crystals for synthesis in the laboratory. So while this algorithm is applicable to a wide range of finite and infinite nets ([19] and [20]), and in theory could be used to construct any net with finitely many orbits of vertices of finite degree, in practice this algorithm is used

primarily for generating highly symmetric nets. The current implementation has been a series of *Crystal Turtlebug* programs intended to facilitate crystal design, and we will describe this implementation in the last section.

One of the major efforts in materials science is the development of synthetic crystals, most notably zeolites (porous silicon-oxygen crystals - imagine porous quartz - with impurities [3]) and metal organic materials (primarily metal ion molecular building blocks connected to each other by organic ligands [22]). One of the major goals is to design novel crystals (at the molecular or even atomic level) prior to synthesis [37].

Now, the traditional (albeit not current¹) definition of a crystal is of a material composed of molecular units that repeat in three linearly independent axial directions. Each of these units is a *unit cell*. A crystal can be represented - and increasingly, is represented - by a 3-periodic (in the sense of Theorem 2.2) net composed of unit cells, and such a graph may be precisely specified by a unit cell. So a typical crystal design program (usually called a “crystal prediction” or “crystal enumeration” program) will produce unit cells.

To our knowledge, in the growing crystal design and enumeration community, there are four groups working on crystal generating programs mathematically similar to ours in that they generate a small fragment of a net or a net-like object and somehow expand it into an entire net. Delgado-Friedrichs et al [13] describes an enumeration of three dimensional tilings and how to generate nets from these. Treacy et al [32] and [33] describes an enumeration starting with “fundamental regions” and uses Coxeter groups to generate unit cells. The Geometrically Restrained Inorganic Structure Prediction (GRINSP) program, described in [4], constructs fragments of the net and then uses various stochastic programs to massage them into a net. Recently, Wilmer et al [36] developed a program for generating metal-organic frameworks from a library of molecular building blocks. In addition, Klee [21] proposed a heuristic for enumerating nets based on the “vector method” as applied to the “quotient graphs” introduced in [9]; this vector method played a major part in inspiring our approach. This list does not include other quite different heuristics being employed; see [7].

2 Groundwork

In this article, we do not assume much specialized knowledge, but we do assume familiarity with graphs and groups. For an accessible introduction to group theory, see [17]. In this section, after fixing the nomenclature, we will review some geometric group theory in order to set the main topic of this paper in context; for an accessible introduction to geometric group theory, see [24]. We will then outline the nomenclature popular in the crystal design literature.

This is a new field in between extant fields, and thus not only is there limited consensus over nomenclature, there are clashes of nomenclatures. (For

¹In 1992, the International Union of Crystallography announced that “... by ‘crystal’ we mean any solid having an essentially discrete diffraction diagram” [25, Ad Interim Commission on Aperiodic Crystals].

a “Graph Theory”-vs.-“Chemical/Crystal Net” Dictionary, see [15].) So some standard nomenclature will have to be adapted from various sources to deal with novel circumstances. Because of these clashes and ambiguities, we will be unusually pedantic in defining our terms. Here is some extant nomenclature that we will use without comment:

- Let $[n] = \{1, 2, \dots, n\}$ for any positive integer n , let \mathbb{Z} be the set of integers, and let \mathbb{Z}_n be the set (or group) of integers modulo n . Let $\mathbb{Z}^{\geq 0}$ be the set of nonnegative integers. Let \mathbb{R} be the set of real numbers. Given a (finite) set S , let $|S|$ be the cardinality of S and let $S^{(2)} = \{\{x, y\} \subseteq S: x \neq y\}$.
- If $\mathbf{x} = x_1x_2 \cdots x_n$ is a string, then its *length* is $|\mathbf{x}| = n$.
- If \mathbb{G} is a group and $\gamma, \phi \in \mathbb{G}$, the (left) *conjugate* of ϕ under γ is ${}^\gamma\phi = \gamma\phi\gamma^{-1}$. The identity of \mathbb{G} is denoted $\text{id}_{\mathbb{G}}$, which we will often write as id if \mathbb{G} is understood.
- If \mathbb{H} is a subgroup of a group \mathbb{G} , write $\mathbb{H} \leq \mathbb{G}$.
- If X is a set, and γ is a permutation of X , and $V \subseteq X$, then the restriction of γ to V is γ_V . Let $\gamma[V] = \{\gamma(v): v \in V\}$, and we say that V is γ -invariant if $\gamma[V] = V$.
- If \mathbb{G} is a group of permutations of X , we denote composition by concatenation: $\phi(\psi(x)) = \phi\psi(x)$. If $V \subseteq X$, let

$$\mathbb{G}|_V = \{\gamma_V: \gamma \in \mathbb{G} \text{ \& } V \text{ is } \gamma\text{-invariant}\}.$$

(For example, for any integer $z \in \mathbb{Z}$, if $[-z, z] = \{-z, -z + 1, \dots, z\}$ and \mathbb{G} is the group of bijections on \mathbb{Z} , then for $a(x) = -x$ and $b(x) = x + 1$ for all $x \in \mathbb{Z}$, $a \in \mathbb{G}|_{[-z, z]}$ but $b \in \mathbb{G} - \mathbb{G}|_{[-z, z]}$, where “ $-$ ” is set difference.)

- If \mathbb{G} is a group of permutations of a set X , and $x, y \in X$, write $x \sim_{\mathbb{G}} y$ if there exists $\eta \in \mathbb{G}$ such that $\eta(x) = y$. Write “ $x \sim y$ ” if \mathbb{G} is understood; the equivalence classes of \sim are the *orbits* of \mathbb{G} . If a set $V \subseteq X$ is a single orbit of (a subgroup of) \mathbb{G} , say that \mathbb{G} *acts transitively* on V . (Thus if \mathbb{G} is the set of automorphisms of a graph, and for any two vertices of that graph there is an automorphism that maps one of the vertices to the other, then the graph is *vertex transitive*.)
- For any set S , let $S^{(2)} = \{\{x, y\}: x, y \in S \text{ \& } x \neq y\}$. If \mathcal{G} is a graph, then $\text{Aut}(\mathcal{G})$ is the automorphism group of \mathcal{G} . Denote $\mathcal{G} = \langle V, E \rangle$, where V is the set of vertices of \mathcal{G} and $E \subseteq V^{(2)}$ the edges (we will identify edges as the unordered pair of their endpoints). If $S \subseteq V$, let $\mathcal{G}[S] = \langle S, E \cap S^{(2)} \rangle$ and

$$\partial S = \{y \in V - S: \exists x \in S \text{ s.t. } \{x, y\} \in E\};$$

thus for any $x \in V$, $\partial\{x\} \cup \{x\}$ is the *neighborhood* of x .

2.1 Some Geometric Group Theory

Geometric group theory, like its progenitor combinatorial group theory, appears motivated by a desire to develop graph-theoretic representations of groups. Our interest is in the opposite direction: to develop useful algebraic tools for building and analyzing graphs. However, in this subsection, we shall follow the mainstream of geometric group theory and put graphs at the service of groups.

One of the standard graphical representations of a group is the Cayley digraph, in which the vertices represent group elements and the arcs represent group generators.

Definition 2.1 *Given a group \mathbb{G} and a set $X \subseteq \mathbb{G}$ of group elements, the subgroup generated by X is the intersection of all subgroups of \mathbb{G} containing all elements of X . This is itself a subgroup and is denoted $\langle X \rangle$.*

Given a group \mathbb{G} and a set X of group elements, X is a set of generators of \mathbb{G} if $\langle X \rangle = \mathbb{G}$.

And then:

Definition 2.2 *Given a group \mathbb{G} and a set of generators X , the Cayley digraph of \mathbb{G} with respect to X is the digraph $\langle \mathbb{G}, \mathbb{G} \times (X \cup X^{-1}) \rangle$, where:*

- *The elements of \mathbb{G} are the vertices.*
- *Letting $X^{-1} = \{x^{-1} : x \in X\}$, $\mathbb{G} \times (X \cup X^{-1})$ is the set of arcs, where (g, x) is the arc of initial vertex g and terminal vertex xg .*

The corresponding Cayley graph is the graph whose set of vertices is \mathbb{G} and whose edges are the unordered pairs $\{\gamma, \eta\gamma\}$, $\gamma \in \mathbb{G}$ and $\eta \in X$.

For example, if the abelian group $\mathbb{Z}_2 \oplus \mathbb{Z}_3$ is generated by an element a of characteristic² 2 and another element b of characteristic 3, then its Cayley digraph is shown in Figure 1.

Notice that in Figure 1, we could imagine a traveler who could move from vertex to vertex along arcs, following an itinerary consisting of a word in $\{a, b\}^*$, reading from left to right. For example, given the word $bab^{-1}a$ from the vertex at upper right, the first b moves the traveler to upper center, the next a moves the traveler to lower center, the next b^{-1} moves the traveler to lower right, and the final a moves the traveler back to upper right.

Remark 2.1 *The Cayley graph of a group might not be unique. For example, as $\mathbb{Z}_2 \oplus \mathbb{Z}_3$ is isomorphic to \mathbb{Z}_6 , which is generated by a single element of order six, another Cayley graph for this group is the 6-cycle.*

²Recall that a group element g is of characteristic $n > 0$ if $g^n = \text{id}$ and for all m , $0 < m < n \implies g^m \neq \text{id}$.

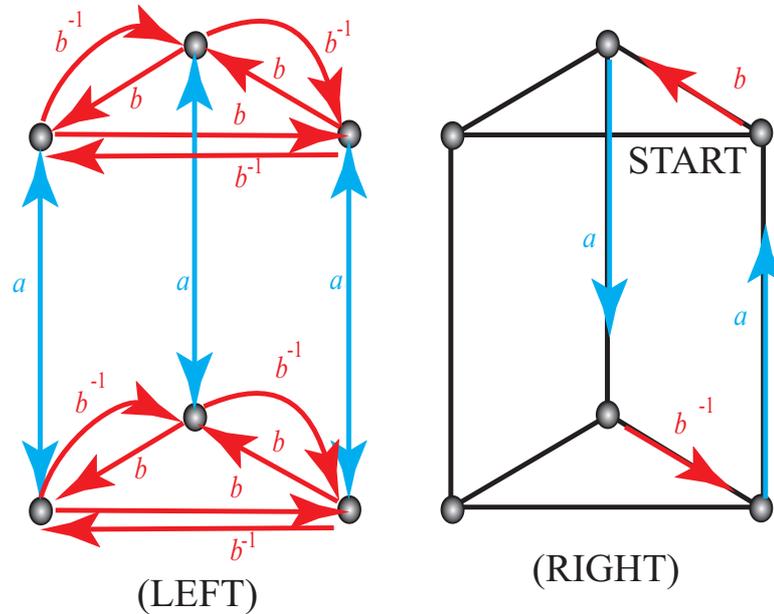


Figure 1: At left, the Cayley digraph for the abelian group $\mathbb{Z}^2 \oplus \mathbb{Z}^3$ as generated by two elements, a and b , where $a^2 = b^3 = \text{id}$. Each vertex represents an element of the group, and each arc represents one of the generators or an inverse of a generator. At right, the corresponding Cayley graph - with unlabeled unoriented edges replacing pairs of arcs - which could be traversed as shown a traveler using an itinerary consisting of a single word, e.g., from the upper right vertex using the word $bab^{-1}a$ as shown.

But there are representations other than Cayley graphs. In Figure 2, we could start with a vertex transitive graph and look at a set of generators for a group of automorphisms. Here, the vertices are the integer vectors of \mathbb{Z}^2 , and our generators are the rotation a by 90° around the origin and the reflection b across the line $x = 1/2$.

In the case of Figure 2, the alphabet represents applications of graph automorphisms, so that a traveler given a word, say ab , employs the named automorphisms to move around. The idea is that the traveler would start at a vertex, take automorphism a to move to a second vertex, and then take automorphism b to move to a third vertex. However, the traversal is undertaken so that the itinerary can be followed in the same way as on the first vertex, so to speak, no matter what vertex the traveler is in. For example, if the traveler started at $(1, 0)$, and if b represents moving from $(1, 0)$ to the $(0, 0)$ by reflecting across the line between the two vertices, then whenever the traveler encounters b in the itinerary, the traveler will want to use a reflection to move to vertex immediately to the traveler's left. Notice that this in this case, the traveler will

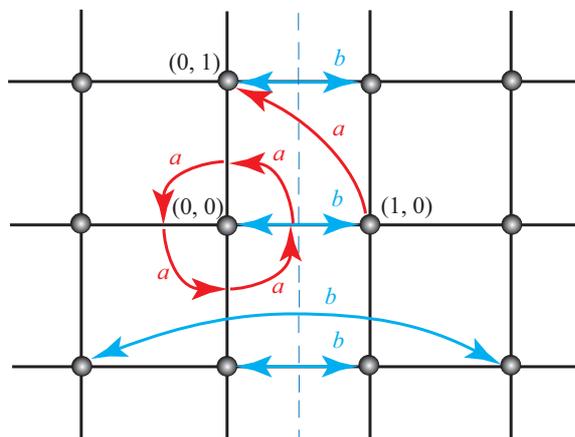


Figure 2: A piece of an infinite graph $\langle \mathbb{Z}^2, \{(i_1, j_1), (i_2, j_2)\} : |i_1 - i_2| + |j_1 - j_2| = 1 \}$. A group of automorphisms is generated by $a(i, j) = (-j, i)$ and $b(i, j) = (-i + 1, j)$.

want to move to the traveler’s left, *not* to the vertex an outside observer with a global view (such as the reader looking at Figure 2) sees as left of the traveler.

Remark 2.2 *This is actually how turtle geometry works [1]: a robot – called a “turtle” – moves around on the floor, following a list of instructions of the sort “rotate clockwise 30°” or “move forward two feet.” It carries out each instruction as is, regardless of its current location or orientation.*

This turtle point of view is accomplished by using conjugates; recall that the *left conjugate* of a bijection α under a bijection β is ${}^\beta\alpha = \beta\alpha\beta^{-1}$. When the traveler is given the word ab , starting at $(1, 0)$, we imagine the traveler in some kind of “standard orientation” standing at a starting point $(1, 0)$. The traveler will always imagine himself to be in standard orientation on the starting point $(1, 0)$, even when he’s not, and act accordingly. Then as in the right hand figure of Figure 3, he will use automorphism a to move from $(1, 0)$ to $a(1, 0) = (0, 1)$, rotating 90° so that now he imagines that $(0, 0)$ is to his left. So as the next word letter is b , he employs the conjugate ${}^a b = aba^{-1}$ to reflect to his left to $(0, 0)$ (the traveler’s left; the reader looking at Figure 3 (Right) sees the traveler reflected *down* one vertex). The result is that the traveler has been moved by two successive automorphisms, a and then ${}^a b$ to go from $(1, 0)$ to $({}^a b \circ a)(1, 0) = aba^{-1}a(1, 0) = ab(1, 0) = (0, 0)$. Notice that the composition actually employed to implement the itinerary ab winds up being the composition ab .

In general, if a traveler wishes to use an itinerary $abcd$ to use the conjugates of a , b , c , and d in succession, each time acting as if the traveler was at the initial vertex v in the initial “orientation”, the destination is computed as follows.

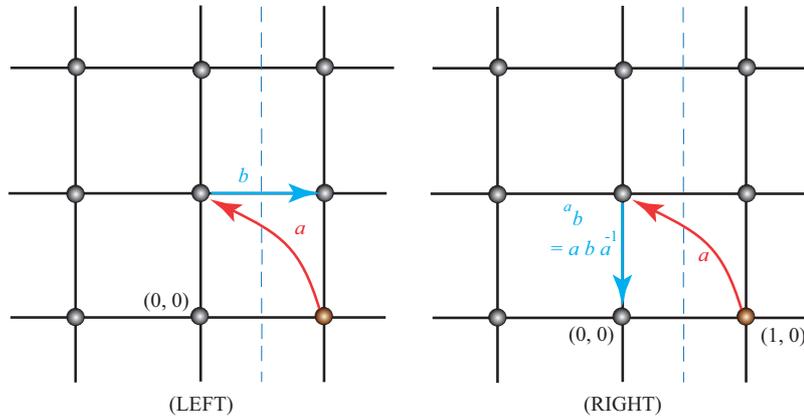


Figure 3: The path at the left represents the composition ba . The path at the right represents the composition ${}^a ba = ab$.

Compute ${}^a b = aba^{-1}$ and ${}^a ba = aba^{-1}a = ab$, ${}^{ab} c = abc(ab)^{-1}$ and ${}^{ab} cab = abc$, and ${}^{abc} d = abcd(abc)^{-1}$ and ${}^{abc} dabc = abcd$:

Fact 2.1 Let a_1, \dots, a_n be bijections on a set X , and for each k , let w_k be the string $a_1 \cdots a_k$ as a string of k letters. For each k , let φ_k be the automorphism corresponding to w_k as follows: $\varphi_1 = a_1$ and $\varphi_{k+1} = {}^{\varphi_k} a_{k+1} \varphi_k$. Then φ_n is the composition $a_1 \cdots a_n$.

This machinery was developed largely to develop graphical representations of groups, e.g., given a group \mathbb{G} , one desires a graph whose automorphism group contains a subgroup isomorphic to \mathbb{G} . We will take the opposite approach, and use this algebraic machinery to analyze - and synthesize - graphs.

2.2 Traversing a Net Using its Symmetry Group

The precise definition of the word “net” seems to vary, but the broad, shallow stream spanning the distance from [11] to [26] seems to lead to something like the following:

Definition 2.3 Let X be a set and let \mathbb{G} be a group of permutations of X . A net over \mathbb{G} is a graph $\mathcal{N} = \langle V, E \rangle$ where $V \subseteq X$, and \mathcal{N} 's symmetry group with respect to \mathbb{G} is $\text{Sym}_{\mathbb{G}}(\mathcal{N}) = \{\gamma \in \mathbb{G}: \gamma_V \in \text{Aut}(\mathcal{N})\}$.

This nomenclature is unconventional, as the symmetry group thus consists of permutations of the underlying space, not of the net itself. In this article, we will deal primarily with permutations of the underlying space, and then say that a symmetry of the net is a permutation of the underlying space that induces an automorphism of the net. And we will usually take \mathbb{G} as given and write

$\text{Sym}(\mathcal{N}) = \text{Sym}_{\mathbb{G}}(\mathcal{N})$. Let's look at two popular examples. Let's start with a Cayley graph.

Example 2.1 Let \mathbb{G} be a group generated by a set $H \subseteq \mathbb{G}$, where $\text{id} \notin H$. We obtain a net by setting $X = V = \mathbb{G}$ and $E = \{\{\gamma, \eta\gamma\}: \gamma \in \mathbb{G} \ \& \ \eta \in H \cup H^{-1}\} = \{\{\gamma, \eta\gamma\}: \gamma \in \mathbb{G} \ \& \ \eta \in H\}$, where $H^{-1} = \{\eta^{-1}: \eta \in H\}$. Here, $\langle V, E \rangle$ is the Cayley graph of \mathbb{G} that generates H .

Returning to Figure 1, we had the commutative group $\mathbb{Z}_2 \oplus \mathbb{Z}_3$ generated by a set $H = \{a, b\}$, and our net thus consisted of vertices $V = \{\text{id}, a, b, ab, b^2, ab^2\}$ (note that $a^{-1} = a$ and $b^{-1} = b^2$) and edges $E = \{\{\text{id}, a\}, \{\text{id}, b\}, \{\text{id}, b^2\}, \{a, ab\}, \{a, ab^2\}, \{b, ab\}, \{b, b^2\}, \{ab, ab^2\}, \{b^2, ab^2\}\}$.

But the work in this paper was motivated by a quite different source, and the following example of [35] (see also [14] on periodic nets and [38] on isometries on \mathbb{R}^n) is what interests crystallographers.

Example 2.2 Suppose that $X = \mathbb{R}^n$ for some n , and that $\mathbb{G} = \mathbb{I}_n$, the group of isometries of \mathbb{R}^n . Let \mathbb{T}_n be the group of translations on \mathbb{R}^n . Suppose that $\mathcal{N} = \langle V, E \rangle$ is a net in this space such that:

- The set V is partitioned into finitely many orbits by $\text{Sym}_{\mathbb{I}_n}(\mathcal{N})$, and
- There is a compact subset of \mathbb{R}^n that intersects every orbit of $\text{Sym}_{\mathbb{I}_n}(\mathcal{N})$, and
- V is uniformly discrete³ in \mathbb{R}^n .

Then by the Schoenflies-Bieberbach Theorem (see, e.g., [8], [30] or [34]), \mathcal{N} is n -periodic in the sense that:

- There is a basis $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^n$, whose associated translations τ_1, \dots, τ_n are symmetries of \mathcal{N} , and hence
- V is partitioned into finitely many orbits by $(\mathbb{T}_n \cap \text{Sym}_{\mathbb{I}_n}(\mathcal{N}))|_V$.

For example, in Figure 4, there is an infinite net whose vertices consist of points on the plane, and whose edges consist of line segments joining those vertices. The group of automorphisms of the net that are induced by plane isometries has finitely many orbits of vertices; there is a compact subset of the plane (in light brown) that intersects each orbit of vertices, and there is a minimum distance between pairs of distinct vertices. By the Schoenflies-Bieberbach Theorem, there are two vectors associated with translations that are symmetries of the graph in the sense that they induce automorphisms of the net.

In Example 2.1, if we wanted to imagine a traveler traversing the Cayley graphs by taking automorphisms from vertex to vertex as a tourist takes trains

³A set $O \subseteq X$ is *uniformly discrete* for a given metric δ on X if there exists $\varepsilon > 0$ such that for each $x, y \in O$, $x \neq y \implies \delta(x, y) \geq \varepsilon$.

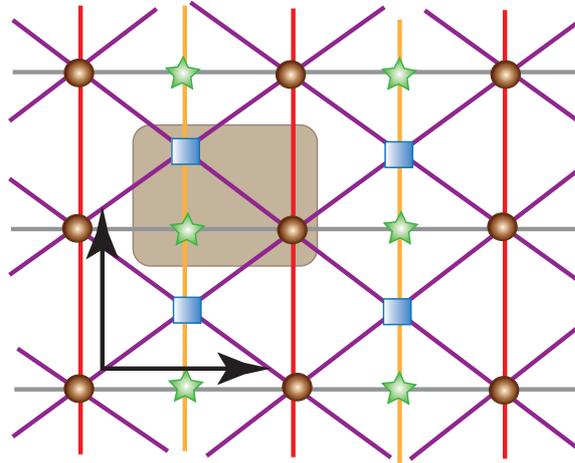


Figure 4: There are three orbits of vertices (brass circles, blue squares and green stars) and four orbits of edges (gray, orange, red, and purple). The compact set of light brown points intersects each orbit of vertices, and hence there is a basis of two black vectors whose associated translations preserve the net.

from station to station, then we would imagine the traveler carrying an itinerary enumerating a schedule for traversing a path in the graph. In Example 2.2, vertices can be in different orbits of the symmetry group of the net, so our traveler has to take isometries of the underlying space from vertex to adjacent vertex *even if some of these isometries are not in the symmetry group of the net*.

Notice the local / global dichotomy.

- From the traveler’s point of view, a permutation on X is merely a vehicle that may be available for traversing an edge. Or, for an entire journey, a permutation on X could be a composition of such traversing permutations.
- From a global perspective, each permutation on X permutes the entire space, and some of these permutations are symmetries of the net that the traveler is touring.

2.3 Traversals on Fundamental Transversals

As we use these permutations to direct a traveler to traverse the net, it would be helpful to have a function that would tell what permutation to use in traversing any given edge from a given incident vertex to the other incident vertex; viz., *if* there was an edge joining vertices v, w , then it would be nice to have a function Γ that would tell us which permutation the traveler is to use to move *from* v to w .

Definition 2.4 Let \mathcal{N} be a net as described in Definition 2.3, and let

$$A_{\mathcal{N}} = \{(x, y) \in V^2: \{x, y\} \in E\}$$

(think of these as trips the traveler could take along the edges). A traversal function for \mathcal{N} is a function $\Gamma: A_{\mathcal{N}} \rightarrow \mathbb{G}$ such that for each $(x, y) \in A_{\mathcal{N}}$, $\Gamma(x, y)(x) = y$.

It is not particularly difficult to verify that if \mathbb{G} acts transitively on X then \mathcal{N} admits a traversal function: just enumerate $A_{\mathcal{N}} = \{(x_{\xi}, y_{\xi}): \xi \in \Xi\}$ for some index set Ξ , and by a (possibly transfinite) induction assign to each arc (x_{ξ}, y_{ξ}) a permutation $\gamma_{\xi} \in \mathbb{G}$ such that $\gamma_{\xi}(x_{\xi}) = y_{\xi}$. But if we consider Γ to be a book of instructions, we would like Γ to be “small” enough for a computer to be able to readily handle it, even if \mathcal{N} is “large” (or even infinite); that way, we would not need much information to move around in the net.

Definition 2.5 Given a net \mathcal{N} as described in Definition 2.3, a traversal function Γ on \mathcal{N} is strict if, for each $(x_1, y_1), (x_2, y_2) \in A_{\mathcal{N}}$, if there exists a symmetry of \mathcal{N} mapping $x_1 \mapsto x_2$ and $y_1 \mapsto y_2$, then for some $\phi \in \text{Sym}(\mathcal{N})$ satisfying $\phi(x_1) = x_2$,

$$\phi\Gamma(x_1, y_1) = \Gamma(x_2, y_2).$$

A traversal function is semi-strict if, for each $(x_1, y_1), (x_2, y_2) \in A_{\mathcal{N}}$ such that there is a symmetry of \mathcal{N} sending $x_1 \mapsto x_2$ and $y_1 \mapsto y_2$, there exist $\phi, \rho \in \text{Sym}(\mathcal{N})$ such that $\phi(x_1) = x_2$ and

$$\phi\Gamma(x_1, y_1) = \rho\Gamma(x_2, y_2).$$

In other words, if the movement from x_1 to y_1 is symmetric (with respect to the net) to the movement from x_2 to y_2 , then the instruction that the traveler follows is also (in a sense) symmetric.

We present the notion of a “strict” traversal since it seems the more natural notion, but we have only been able to develop the theory for semi-strict traversals. We will describe how to construct a semi-strict traversal function where the permutations used are generated by some finitary algorithm. To do this, we will need to treat the net as if it was composed of many small identical pieces (which we will call “fundamental transversals”) and setting up this treatment will require setting up a system for representing paths through the net.

We need a fact distilled in [16, Prop. I.2.6].

Proposition 2.1 Given a connected net $\mathcal{N} = \langle V, E \rangle$ with a group \mathbb{G} of automorphisms, there exists a set $S \subseteq V$ such that:

- The set S intersects each orbit of \mathbb{G} in V , with that intersection consisting of one vertex.
- The induced subgraph $\mathcal{N}[S]$ is connected.

Their proof was for graphs alone, but the proof in [24] is readily adapted.⁴

Let us call such a set S a *fundamental \mathbb{G} -transversal*, or just a *fundamental transversal* when \mathbb{G} is clear or implied. Observe also that the number of orbits – and the fundamental \mathbb{G} -transversal itself – depends on the original group \mathbb{G} (see Figure 5). For different groups \mathbb{G} , one obtains different fundamental \mathbb{G} -transversals.

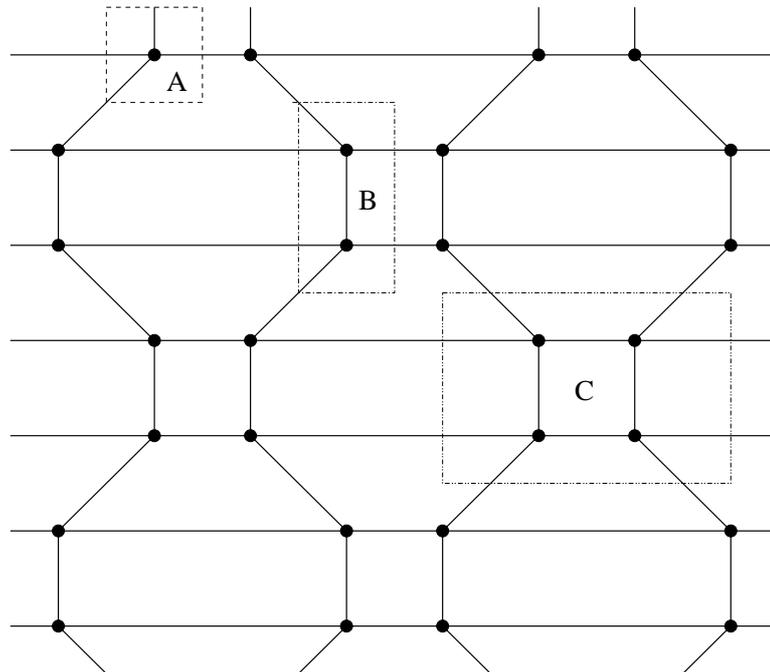


Figure 5: Here is a net \mathcal{N} on the plane. If \mathbb{G}_1 was the group of plane isometries, a fundamental \mathbb{G}_1 -transversal would be a single vertex (A). If \mathbb{G}_2 was the group of plane isometries generated by the translations and the reflection across the vertical axis – i.e., by the reflection $(x, y) \mapsto (-x, y)$ – we would need two vertices for a fundamental \mathbb{G}_2 -transversal, as in (B). If \mathbb{G}_3 was just the group of plane translations, we would need four vertices for a fundamental \mathbb{G}_3 -transversal, as in (C).

Our plan is to start with a single fundamental transversal and cover the net with copies of it. More precisely, we will start with a single fundamental transversal and cover the net with copies of corresponding vertices from that fundamental transversal. In order to do this, we will encode paths through

⁴ Start with any vertex v_0 and let $S_0 = \{v_0\}$. For any n , if there are any vertices in V not in an orbit intersecting S_n , choose a vertex v in an unrepresented orbit and a path from a vertex in S_n to v , and let v' be the first vertex on that path in an orbit unrepresented in S_{n+1} . There must be a vertex $v_{n+1} \in \partial S_{n+1}$ such that $v_{n+1} \sim v'$, so let $S_{n+1} = S_n \cup \{v_{n+1}\}$. Continue until all orbits are represented.

the net. Keeping the goal of semi-strictness in mind, we will use a fundamental transversal for this encoding. To get started from a given fundamental transversal, we need the function that can take any vertex from the net and produce the equivalent node in the fundamental transversal.

Definition 2.6 *Given a fundamental \mathbb{G} -transversal S for a net \mathcal{N} (embedded in a space X on which a group \mathbb{G} acts transitively), we define a function $\kappa: V \rightarrow S$ as follows. For any $x \in V$, let $\kappa(x)$ be the unique $y \in S$ such that $x \sim_{\text{Sym}_{\mathbb{G}}(\mathcal{N})} y$.*

For the rest of this paper, we will presume \mathbb{G} and \mathcal{N} and write “ $x \sim y$.”

3 Traversing the Net

We outline a method for generating a traversal function given a fundamental transversal, and for generating words (“itineraries”) defining paths through the net, employing appropriate traversal functions.

3.1 Itineraries

We develop a formal system for encoding paths using words. In order to get the alphabet, and to obtain a representation of the symmetry that moves the traveler *from one vector to an adjacent vector*, we employ a gadget built upon the notion of a fundamental transversal.

Definition 3.1 *A transversal diagram of a net $\mathcal{N} = \langle V, E \rangle$ embedded in a space X on which a group \mathbb{G} of permutations act transitively (and where \mathbb{G} induce a corresponding group of automorphisms of \mathcal{N}) is a tuple $\mathcal{S} = \langle S, \partial S, \Sigma \rangle$ where S is a fundamental \mathbb{G} -transversal and $\Sigma = \{(u, v) \in S \times (S \cup \partial S) : \{u, v\} \in E\}$.*

We use the set Σ to generate words for traversing the net by starting at the appropriate vertex in the (given) fundamental transversal and follow the symbols of the word as if they enumerate a list of instructions. We need to characterize those words that the traveler may use for travelling, i.e., those words that the traveler can actually use as itineraries.

Definition 3.2 *Fix a transversal diagram $\langle S, \partial S, \Sigma \rangle$. The itineraries of Σ are the strings from Σ defined by the following recursion.*

A itinerary \mathbf{w} will have an initial vertex $\text{init}(\mathbf{w}) \in S$, a terminal vertex $\text{term}(\mathbf{w}) \in S$, and be generated as follows.

- *For each $v \in S$, the blank for v , denoted ϵ_v , is an itinerary with $\text{init}(\epsilon_v) = \text{term}(\epsilon_v) = v$. (The word ϵ_v means “you are on the vertex v , and you stay there.”)*
- *Σ consists of individual instructions (which we will call “steps”), so that for any $(u, v) \in \Sigma$, we have $\text{init}((u, v)) = u$ and $\text{term}((u, v)) = v$. (The word (u, v) means that “you are on the vertex u and you move to the adjacent vertex v along the sole connecting edge using the scheduled permutation for (u, v) .”)*

- If \mathbf{v} and \mathbf{w} are itineraries, and if $\text{term}(\mathbf{v}) = \text{init}(\mathbf{w})$, then \mathbf{vw} is an itinerary with $\text{init}(\mathbf{vw}) = \text{init}(\mathbf{v})$ and $\text{term}(\mathbf{vw}) = \text{term}(\mathbf{w})$. In particular, if \mathbf{v} is an itinerary and $(x, y) \in \Sigma$, then $\mathbf{v}(x, y)$ is a itinerary if and only if $\text{term}(\mathbf{v}) = x$. (The itinerary \mathbf{vw} means “carry out the permutation encoded by \mathbf{v} and then the permutation encoded by \mathbf{w} .”)

Denote the set of itineraries by $\mathbb{W} = \mathbb{W}_{\mathcal{N}, \mathbb{G}}$.

Notice that if $\mathbb{G}|_V$ does not act transitively on the vertices of \mathcal{N} , then this set of strings generated from Σ is not a semigroup as it is not closed under concatenation, but if $\mathbb{G}|_V$ does act transitively on the vertices of \mathcal{N} , it is a monoid.

Using itineraries as (lists of) instructions, if some nice railroad engineer has assigned appropriate permutations to each of the instructions, the traveler can move around the net using something like a state transition function of automata theory. The idea is that if a traveler started at some $r \in S$ and followed an itinerary \mathbf{v} (where $\text{init}(\mathbf{v}) = r$), it would reach some $z \in V$. If provided with a next instruction (x, y) , where $\kappa(z) = x$, the traveler would move to an adjacent vertex w where the appropriate symmetry maps $\{x, y\}$ to $\{z, w\}$. These “appropriate symmetries” will be compositions of the symmetries of \mathcal{N} .

Definition 3.3 Fix a transversal diagram $\langle S, \partial S, \Sigma \rangle$. Call each pair $(x, y) \in \Sigma \cap S^2$ a local step, and each pair $(x, y) \in \Sigma \cap (S \times \partial S)$ a jump step. Given an itinerary \mathbf{v} , let $\|\mathbf{v}\|$ be the number of jump steps in \mathbf{v} . If $\|\mathbf{v}\| = 0$, call \mathbf{v} jump-free.

A local step instructs a traveler to move from a vertex within a fundamental transversal to another vertex within that same fundamental transversal, while a jump step instructs a traveler to move from one fundamental transversal to another. See Figure 6.

3.2 The Algorithm: Assigning Actions to Itineraries

The heart of the algorithm is the assignment of permutations to itineraries.

We first assign permutations to individual steps, and then to entire itineraries. We first need a convention to allow us to traverse itineraries of local steps in reverse order: this is not quite the usual string reversal operation, for we are also “reversing” the individual symbols – the individual local steps – in the string as well.

Definition 3.4 Given a transversal diagram $\langle S, \partial S, \Sigma \rangle$ and a local step (x, y) , let $(x, y)^R = (y, x)$. Given two jump-free strings \mathbf{u} and \mathbf{v} , with $\text{term}(\mathbf{u}) = \text{init}(\mathbf{v})$, let $(\mathbf{uv})^R = \mathbf{v}^R \mathbf{u}^R$.

Notice that this makes sense as $\text{init}(\mathbf{u}^R) = \text{term}(\mathbf{u}) = \text{init}(\mathbf{v}) = \text{term}(\mathbf{v}^R)$. (We will not attempt to reverse strings having jump steps.) We now return to making assignments.

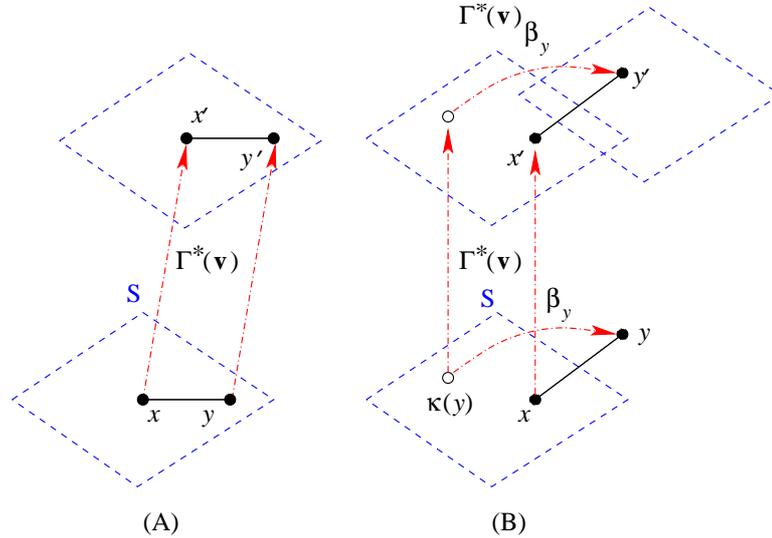


Figure 6: If the traveler is at a vertex x' , its itinerary will tell it how to move as if it was at x . So it will move using a conjugate permutation (the conjugacy function being a symmetry of \mathcal{N}). In (A), the traveler moves within the fundamental transversal, while in (B) it jumps out to another fundamental transversal that is a copy of the first (copied by the net symmetry $\Gamma^{*(\mathbf{v})}\beta_y$).

Definition 3.5 Given a fundamental \mathbb{G} -transversal S of a net \mathcal{N} and a group \mathbb{G} of permutations on X , an assignment function is a function $\Gamma^*: \mathbb{W} \rightarrow \mathbb{G}$ such that for any $\mathbf{w} \in \mathbb{W}$, $\kappa(\Gamma^*(\mathbf{w})(\text{init}(\mathbf{w}))) = \text{term}(\mathbf{w})$.

So we encode traversals across individual edges of the transversal diagram. But first, a piece of nomenclature.

Definition 3.6 Suppose we are given a transversal diagram $\langle S, \partial S, \Sigma \rangle$ for a net \mathcal{N} . In addition, suppose that we are given a spanning tree \mathcal{T} of the graph $\langle S, \{\{x, y\}: (x, y) \in \Sigma \cap S^2\} \rangle$ (\mathcal{T} exists as the fundamental \mathbb{G} -transversal is connected). Then for each $x, y \in S$, let $\mathbf{p}_{x,y}$ be the itinerary representing the unique path from x to y in \mathcal{T} . In particular, for each x , let $\mathbf{p}_{x,x} = \epsilon_x$, the empty string of initial and terminal vertex x .

We construct Γ^* by recursion on the itineraries \mathbf{w} . We start by considering itineraries of local steps. Order the vertices of the transversal diagram as follows: choose $z_1 \in S$ and order the vertices $z_1, \dots, z_{|S|}$ of S so that for each j , if $S_j = \{z_1, z_2, \dots, z_j\}$, then the induced subgraph of \mathcal{T} of vertices S_j is connected. Thus each S_j is the set of vertices of a subtree of \mathcal{T} .

Next, order the local steps as follows:

$$\Sigma \cap S^2 = \{(x_1, y_1), (x_2, y_2), \dots, (x_{2c}, y_{2c})\}, \tag{1}$$

noting that a pair (x, y) is a local step (i.e., an orientation of an edge of \mathcal{N} within the fundamental transversal) iff (y, x) is, so that the number of local steps is even. (Notice that each x_i is some z_j , just as each y_l is some z_l .) For this ordering, we require that if (z_i, z_j) precedes (z_k, z_l) in the ordering of local steps, then either $\max\{i, j\} < \max\{k, l\}$, or $\max\{i, j\} = \max\{k, l\}$ and $\min\{i, j\} < \min\{k, l\}$, or $i = l < j = k$. Note that typically $x_{2i} = y_{2i-1}$ and $y_{2i} = x_{2i-1}$ for $i \leq c$.

Remark 3.1 For each $z_j \in S$, let j be the rank of z_j . Observe that for any $x, y \in S$, $\mathbf{p}_{x,y}$ represents a path in \mathcal{T} , and by the ordering (1), either x or y must be the highest ranked vertex in $\mathbf{p}_{x,y}$. Observe also that for each $j > 1$, exactly one of z_j 's neighbors in \mathcal{T} is of rank lower than j .

Finally, for each j , let \mathbb{W}_j be the set of itineraries from S_j , so that $\bigcup_j \mathbb{W}_j$ is the set of all jump-free itineraries. Let $\mathbb{W}_0 = \emptyset$.

0. Γ^* on jump-free sets. We define Γ^* on the languages \mathbb{W}_j by recursion on j . As $\mathbb{W}_0 = \emptyset$, Γ^* is vacuously defined on \mathbb{W}_0 . Now, suppose that we have defined Γ^* on \mathbb{W}_j so that if $x, y \in S_j$, then $\Gamma^*(\mathbf{p}_{x,y})(x) = y$. Define Γ^* on $\mathbb{W}_{j+1} - \mathbb{W}_j$ as follows.

- i. **The basis of the recursion I.** Let $\Gamma^*(\epsilon_{z_{j+1}}) = \text{id}$, the identity symmetry of \mathbb{G} , and observe that $\Gamma^*(\epsilon_{z_{j+1}})(\text{init}(\epsilon_{z_{j+1}})) = \Gamma^*(\epsilon_{z_{j+1}})(z_{j+1}) = z_{j+1} = \text{term}(\epsilon_{z_{j+1}})$.
- ii. **The basis of the recursion II.** Choose the unique $(x_l, y_l) \in \Sigma$ such that $y_l = z_{j+1}$, x_l is of lower rank than y_l , and $\mathbf{p}_{x_l, y_l} = (x_l, y_l)$, i.e., (x_l, y_l) is on \mathcal{T} . Choose $\Gamma^*((x_l, y_l)) \in \mathbb{G}$ such that $\Gamma^*((x_l, y_l))(x_l) = y_l$, and noting that $x_{i+1} = y_l$ and $y_{i+1} = x_l$, let $\Gamma^*((y_l, x_l)) = \Gamma^*((x_l, y_l))^{-1}$. This gives $\Gamma^*((x_l, y_l))(\text{init}((x_l, y_l))) = \Gamma^*((x_l, y_l))(x_l) = y_l = \text{term}((x_l, y_l))$, and similarly $\Gamma^*((y_l, x_l))(\text{init}((y_l, x_l))) = \text{term}((y_l, x_l))$.
- iii. **Recursive step I.** Let (x_l, y_l) be the unique step in \mathcal{T} from $x_l \in S_j$ to $y_l = z_{j+1}$: by Remark 3.1, $\mathbf{p}_{x_i, x_l} \in \mathbb{W}_j$. For each $(x_i, y_i) \in \Sigma$ such that $x_i \in S_j$ and $y_i = z_{j+1}$, let $\Gamma^*((x_i, y_i)) = \Gamma^*((x_l, y_l))\Gamma^*(\mathbf{p}_{x_i, x_l})$. Then $\Gamma^*((x_i, y_i))(x_i) = \Gamma^*((x_l, y_l))\Gamma^*(\mathbf{p}_{x_i, x_l})(x_i) = \Gamma^*((x_l, y_l))(x_l) = y_l = y_i$ by 0(ii). Let $\Gamma^*((y_i, x_i)) = \Gamma^*((x_i, y_i))^{-1}$.
- iv. **Recursive step II.** For any $\mathbf{w}, (x, y) \in \mathbb{W}_{j+1}$ such that $\text{term}(\mathbf{w}) = x$, let $\Gamma^*(\mathbf{w}(x, y)) = \Gamma^*((x, y))\Gamma^*(\mathbf{w})$. Observe that $\Gamma^*(\mathbf{w}(x, y))(\text{init}(\mathbf{w})) = \Gamma^*((x, y))\Gamma^*(\mathbf{w})(\text{init}(\mathbf{w})) = \Gamma^*((x, y))(x) = y$.

Repeat until Γ^* is defined on \mathbb{W}_{j+1} . Repeat for all $j \leq |S|$.

We should make four useful observations.

Lemma 3.1 For any jump-free \mathbf{u}, \mathbf{v} such that $\text{term}(\mathbf{u}) = \text{init}(\mathbf{v})$, $\Gamma^*(\mathbf{u}\mathbf{v}) = \Gamma^*(\mathbf{v})\Gamma^*(\mathbf{u})$. For any jump-free \mathbf{w} , $\Gamma^*(\mathbf{w}^R) = \Gamma^*(\mathbf{w})^{-1}$. For any jump-free \mathbf{w} , $\Gamma^*(\mathbf{w}) = \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{w}), \text{term}(\mathbf{w})})$. Finally, if \mathbf{w} is jump-free and $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w})$, then $\Gamma^*(\mathbf{w}) = \text{id}$.

Proof. For the first claim, if $\mathbf{u} = (x_{s(1)}, y_{s(1)}) \cdots (x_{s(m)}, y_{s(m)})$ and $\mathbf{v} = (x_{t(1)}, y_{t(1)}) \cdots (x_{t(n)}, y_{t(n)})$ and $y_{s(m)} = x_{t(1)}$, then by (0)(iv) in the above recursion, $\Gamma^*(\mathbf{uv})$ is:

$$\begin{aligned}
&= \Gamma^*((x_{s(1)}, y_{s(1)}) \cdots (x_{s(m)}, y_{s(m)})(x_{t(1)}, y_{t(1)}) \cdots (x_{t(n)}, y_{t(n)})) \\
&= \Gamma^*((x_{s(2)}, y_{s(2)}) \cdots (x_{s(m)}, y_{s(m)})(x_{t(1)}, y_{t(1)}) \cdots (x_{t(n)}, y_{t(n)})) \\
&\qquad\qquad\qquad \Gamma^*((x_{s(1)}, y_{s(1)})) \\
&= \dots \\
&= \Gamma^*((x_{t(1)}, y_{t(1)}) \cdots (x_{t(n)}, y_{t(n)})) \Gamma^*((x_{s(m)}, y_{s(m)})) \cdots \Gamma^*((x_{s(1)}, y_{s(1)})) \\
&= \dots \\
&= \Gamma^*((x_{t(1)}, y_{t(1)}) \cdots (x_{t(n)}, y_{t(n)})) \Gamma^*((x_{s(1)}, y_{s(1)}) \cdots (x_{s(m)}, y_{s(m)})) \\
&= \Gamma^*(\mathbf{v}) \Gamma^*(\mathbf{u}).
\end{aligned}$$

To prove the second claim using the first, observe that if $\mathbf{u} = (x, x')(x', x'') \cdots (x''', x''')$, then

$$\begin{aligned}
\Gamma^*(\mathbf{u}) &= \Gamma^*((x''', x''') \cdots (x', x)) \\
&= \Gamma^*((x', x)) \cdots \Gamma^*((x''', x''')) \\
&= (\Gamma^*((x''', x'''))^{-1} \cdots \Gamma^*((x', x))^{-1})^{-1} \\
&= (\Gamma^*((x''', x''')) \cdots \Gamma^*((x, x')))^{-1} \\
&= \Gamma^*((x, x') \cdots (x''', x'''))^{-1} = \Gamma^*(\mathbf{u})^{-1}.
\end{aligned}$$

We prove the third and fourth claims by simultaneous induction. We observe that they hold vacuously for \mathbb{W}_0 and as our Inductive Hypothesis, we assume that both claims hold for \mathbb{W}_j : we claim that they both hold for \mathbb{W}_{j+1} .

We first prove the third claim for itineraries of the form $\mathbf{u}(x, z_{j+1})$, where $\mathbf{u} \in \mathbb{W}_j$ and $\text{term}(\mathbf{u}) = x$. Using the first claim and the Induction Hypothesis, if x_l is the unique vertex of rank at most j adjacent to z_{j+1} in \mathcal{T} , then

$$\begin{aligned}
\Gamma^*(\mathbf{u}(x, z_{j+1})) &= \Gamma^*((x, z_{j+1})) \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x}) \\
&= \Gamma^*((x_l, z_{j+1})) \Gamma^*(\mathbf{p}_{x, x_l}) \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x}) \\
&= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x} \mathbf{p}_{x, x_l}(x_l, z_{j+1})).
\end{aligned}$$

Recalling that our designated paths are in the tree \mathcal{T} , let x' be the vertex on the path of $\mathbf{p}_{\text{init}(\mathbf{u}), x_l}$ such that $\mathbf{p}_{x', x}$ intersects $\mathbf{p}_{\text{init}(\mathbf{u}), x_l}$ at x' : see Figure 7.

Note that $\mathbf{p}_{x, x'}, \mathbf{p}_{x', x} \in \mathbb{W}_j$, so by the Induction Hypothesis $\Gamma^*(\mathbf{p}_{x', x} \mathbf{p}_{x, x'}) = \text{id}$. Thus

$$\begin{aligned}
\Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x} \mathbf{p}_{x, x_l}) &= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x'} \mathbf{p}_{x', x} \mathbf{p}_{x, x'} \mathbf{p}_{x', x_l}) \\
&= \Gamma^*(\mathbf{p}_{x', x_l}) \Gamma^*(\mathbf{p}_{x', x} \mathbf{p}_{x, x'}) \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x'}) \\
&= \Gamma^*(\mathbf{p}_{x', x_l}) \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x'}) \\
&= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x'} \mathbf{p}_{x', x_l}) \\
&= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x_l})
\end{aligned}$$

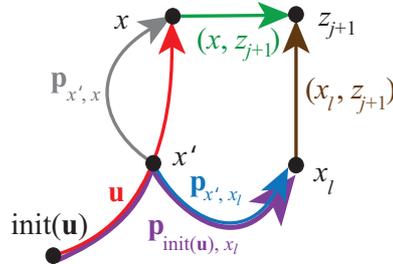


Figure 7: The purple and gray paths $\mathbf{p}_{\text{init}(\mathbf{u}), x_l}$ and $\mathbf{p}_{x', x}$ lie in \mathcal{T} , as does the step (x_l, z_{j+1}) . As \mathcal{T} is a tree, the leftover piece of \mathbf{u} and the link (x, z_{j+1}) do not lie in \mathcal{T} .

The proof for itineraries of \mathbb{W}_{j+1} in which z_{j+1} appears elsewhere or repeatedly follows by repeated application of the above, and the following observation: if $\text{init}(\mathbf{w}) = z_{j+1} = \text{term}(\mathbf{w})$ with $\mathbf{w} = (z_{j+1}, x)\mathbf{u}(x', z_{j+1})$ for some $\mathbf{u} \in \mathbb{W}_j$, then using the Induction Hypothesis,

$$\begin{aligned} \Gamma^*(\mathbf{w}) &= \Gamma^*((x_l, z_{j+1})\Gamma^*(\mathbf{p}_{x_l, x}\mathbf{p}_{x, x'}\mathbf{p}_{x', x_l})\Gamma^*((z_{j+1}, x_l))) \\ &= \Gamma^*((x_l, z_{j+1}))\Gamma^*((z_{j+1}, x_l)) = \text{id}. \end{aligned}$$

Then

$$\begin{aligned} &\Gamma^*(\mathbf{u}(x, z_{j+1})(z_{j+1}, y)\mathbf{u}' \cdots \mathbf{u}''') \\ &= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x}\mathbf{p}_{x, x_l}(x_l, z_{j+1})(z_{j+1}, x_l)\mathbf{p}_{x_l, y} \cdots (z_{j+1}, x_l)\mathbf{p}_{x_l, \text{term}(\mathbf{u}''')}) \\ &= \cdots = \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \text{term}(\mathbf{u}''')}). \end{aligned}$$

Finally, the Fourth claim follows from the second. If $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w})$ where $\mathbf{w} = \mathbf{u}\mathbf{v}$, then

$$\Gamma^*(\mathbf{w}) = \Gamma^*(\mathbf{u}\mathbf{v}) = \Gamma^*(\mathbf{u})\Gamma^*(\mathbf{v}) = \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \text{term}(\mathbf{u})})\Gamma^*(\mathbf{p}_{\text{init}(\mathbf{v}), \text{term}(\mathbf{v})}).$$

But as $\text{init}(\mathbf{u}) = \text{term}(\mathbf{v})$ and $\text{term}(\mathbf{u}) = \text{init}(\mathbf{v})$, this becomes

$$\begin{aligned} &\Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \text{term}(\mathbf{u})})\Gamma^*(\mathbf{p}_{\text{term}(\mathbf{u}), \text{init}(\mathbf{u})}) \\ &= \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \text{term}(\mathbf{u})})\Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \text{term}(\mathbf{u})})^{-1} = \text{id}. \end{aligned}$$

■

We now have assignments for all itineraries for movement within the fundamental transversal. If we desire to jump out, we need some additional assignments. But first, we need a bit of nomenclature.

Definition 3.7 *Suppose that we are given a transversal diagram $\langle S, \partial S, \Sigma \rangle$ for a net \mathcal{N} . In addition, suppose that one is given, for each $y \in \partial S$, a symmetry $\beta_y \in \text{Sym}(\mathcal{N})$ such that $\beta_y(\kappa(y)) = y$.*

We complete the construction of Γ^* as follows.

1. Γ^* for one jump step. If (x, y) is a jump step, let $\Gamma^*((x, y)) = \beta_y \Gamma^*(\mathbf{p}_{x, \kappa(y)})$. Notice that $\Gamma^*((x, y))(x) = \beta_y \Gamma^*(\mathbf{p}_{x, \kappa(y)})(x) = \beta_y(\kappa(y)) = y$.

2. Γ^* for adding local steps. Suppose that (x, y) is a local step and that $x = \text{term}(\mathbf{u})$. If $\Gamma^*(\mathbf{u}) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x})$ and $\gamma \in \text{Sym}(\mathcal{N})$ (Theorem 3.1 will assure that this can be done), then set $\Gamma(\mathbf{u}(x, y)) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x}(x, y)) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), y})$. We get $\Gamma^*(\mathbf{u}(x, y))(\text{init}(\mathbf{u})) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), y})(\text{init}(\mathbf{u})) = \gamma(y)$ as desired.

3. Γ^* for adding jump steps. Suppose that we are given \mathbf{u} and a jump step (x, y) such that $\text{term}(\mathbf{u}) = x$. Suppose that for some $\gamma \in \text{Sym}(\mathcal{N})$, $\Gamma^*(\mathbf{u}) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x})$. Let $\Gamma(\mathbf{u}(x, y)) = \gamma \beta_y \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), x} \mathbf{p}_{x, \kappa(y)}) = \gamma \beta_y \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \kappa(y)})$. Observe that $\Gamma(\mathbf{u}(x, y))(\text{init}(\mathbf{u})) = \gamma \beta_y \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{u}), \kappa(y)})(\text{init}(\mathbf{u})) = \gamma \beta_y(\kappa(y)) = \gamma(y)$ as desired.

3.3 Proving the Desiderata

Before we prove the three desiderata, we need a generalization of the first claim of Lemma 3.1.

Lemma 3.2 *If $\Gamma^*(\mathbf{u}) = \gamma \Gamma^*(\mathbf{p}_{x, y})$ and $\Gamma^*(\mathbf{v}) = \delta \Gamma^*(\mathbf{p}_{y, z})$, then $\Gamma^*(\mathbf{uv}) = \gamma \delta \Gamma^*(\mathbf{p}_{x, z})$.*

This extends the first claim of Lemma 3.1:

$$\begin{aligned} \Gamma^*(\mathbf{uv}) &= \gamma \delta \Gamma^*(\mathbf{p}_{x, z}) = \gamma \delta \Gamma^*(\mathbf{p}_{x, y} \mathbf{p}_{y, z}) = \gamma \delta \Gamma^*(\mathbf{p}_{y, z}) \Gamma^*(\mathbf{p}_{x, y}) \\ &= \gamma \delta \Gamma^*(\mathbf{p}_{y, z}) \gamma^{-1} \gamma \Gamma^*(\mathbf{p}_{x, y}) = \gamma (\delta \Gamma^*(\mathbf{p}_{y, z})) \gamma \Gamma^*(\mathbf{p}_{x, y}) \\ &= \gamma \Gamma^*(\mathbf{v}) \Gamma^*(\mathbf{u}). \end{aligned}$$

Proof of Lemma 3.2. Proceed by induction on the length of \mathbf{v} . If $\mathbf{v} = \epsilon_y$ for $y = z$, this is immediate. If the Lemma is true of \mathbf{u} and \mathbf{v} , then there are two cases.

First, if (z, u) is a local step so that $\Gamma^*(\mathbf{v}(z, u)) = \delta \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{v}), u})$, then by ((2) Γ^* for adding local steps), $\Gamma^*(\mathbf{uv}(z, u)) = \gamma \delta \Gamma^*(\mathbf{p}_{x, u})$. Second, if (z, u) is a jump step, then by ((3) Γ^* for adding jump steps), so that $\Gamma^*(\mathbf{v}(z, u)) = \delta \beta_y \Gamma^*(\mathbf{p}_{z, \kappa(u)})$, we have $\Gamma^*(\mathbf{uv}(z, \kappa(u))) = \gamma \delta \beta_y \Gamma^*(\mathbf{p}_{x, \kappa(u)})$. Combining these cases, we have the Inductive Step. ■

We now present a normal form for the permutations defined by itineraries. Recall that we fixed, for each $y \in \partial S$, a symmetry $\beta_y \in \text{Sym}(\mathcal{N})$ such that $\beta_y(\kappa(y)) = y$.

Theorem 3.1 *If*

$$\mathbf{w} = \mathbf{v}_0(x_1, y_1) \mathbf{v}_1(x_2, y_2) \mathbf{v}_2 \cdots \mathbf{v}_{m-1}(x_m, y_m) \mathbf{v}_m$$

is an itinerary with \mathbf{v}_i being jump-free for each i , and for (x_i, y_i) being a jump step for each i , then

$$\Gamma^*(\mathbf{w}) = \left(\prod_{i=1}^m \beta_{y_i} \right) \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{v}_0), \text{term}(\mathbf{v}_m)}).$$

Thus if $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w})$, $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$.

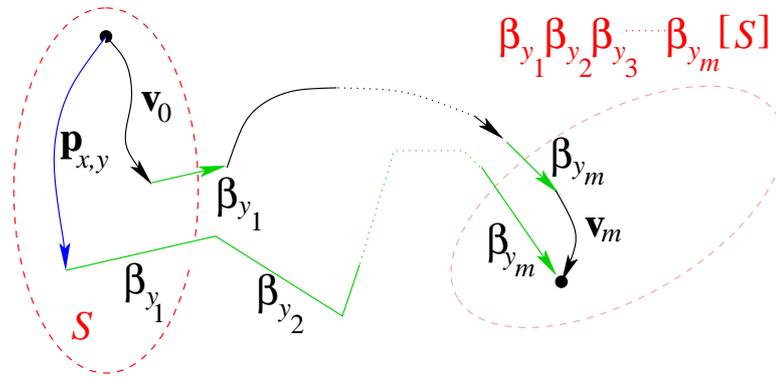


Figure 8: A normal form representation of permutations associated with itineraries.

Proof. The proof is by induction on the number of jump steps $\|\mathbf{w}\|$.

For the basis of the induction, $\|\mathbf{w}\| = 0 \implies \Gamma^*(\mathbf{w}) = \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{w}), \text{term}(\mathbf{w})}) = \text{id}$ by Lemma 3.1.

For the inductive step, if the lemma is true of itineraries of m jumps, and $\|\mathbf{w}\| = m + 1$, suppose $\mathbf{w} = \mathbf{v}(x, y)\mathbf{u}$ where $\|\mathbf{v}\| = m$ and (x, y) is a jump step and \mathbf{u} is jump-free. If the m jumps in \mathbf{v} are to $y_1, y_2, \dots, y_m \in \partial S$, respectively, and if $z = \text{init}(\mathbf{w})$, Lemma 3.2 gives us $\Gamma^*(\mathbf{v}(x, y)) = \gamma \Gamma^*(\mathbf{p}_{z, \kappa(y)})$ for some $\gamma \in \text{Sym}(\mathcal{N})$. Thus $\Gamma^*(\mathbf{v}(x, y)\mathbf{u}) = \gamma \Gamma^*(\mathbf{p}_{z, \text{term}(\mathbf{w})})$. But $\Gamma^*(\mathbf{v}) = (\prod_{i=1}^m \beta_i) \Gamma^*(\mathbf{p}_{z, x})$ by the Induction Hypothesis, and $\Gamma^*((x, y)) = \beta_y \Gamma^*(\mathbf{p}_{x, \kappa(y)})$, so by Lemma 3.2 again, $\Gamma^*(\mathbf{v}(x, y)) = (\prod_{i=1}^m \beta_i) \beta_y \Gamma^*(\mathbf{p}_{z, \kappa(y)})$. So for $y = y_{i+1}$, $\Gamma^*(\mathbf{w}) = \left(\prod_{i=1}^{m+1} \beta_i \right) \Gamma^*(\mathbf{p}_{z, \text{term}(\mathbf{w})})$. ■

We are now ready for the desiderata. We prove the second desideratum first: the composition of steps from a vertex to another vertex in the same $\text{Sym}(\mathcal{N})$ -orbit is itself in $\text{Sym}(\mathcal{N})$.

Theorem 3.2 *If \mathbf{w} is an itinerary such that $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w})$, then $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$.*

Proof. This is an immediate consequence of Theorem 3.1: for some $\gamma \in \text{Sym}(\mathcal{N})$, $\Gamma^*(\mathbf{w}) = \gamma \Gamma^*(\mathbf{p}_{\text{init}(\mathbf{w}), \text{term}(\mathbf{w})}) = \gamma \in \text{Sym}(\mathcal{N})$. ■

The first desideratum is the most basic: the algorithm actually generates the net it is supposed to generate.

Theorem 3.3 *Given the net $\mathcal{N} = \langle V, E \rangle$, the transversal diagram $\langle S, \partial S, \Sigma \rangle$, the itineraries \mathbb{W} , etc., let*

$$V^* = \{\Gamma^*(\mathbf{w})(x) : \mathbf{w} \in \mathbb{W} \ \& \ \text{init}(\mathbf{w}) = x \in S\}$$

and let E^* be the set of all unordered pairs $\{\Gamma^*(\mathbf{w})(x), \Gamma^*(\mathbf{w}(y, z))(x)\}$ such that

$$\mathbf{w} \in \mathbb{W} \ \& \ \text{init}(\mathbf{w}) = x \in S \ \& \ \text{term}(\mathbf{w}) = y \in S \ \& \ (y, z) \in \Sigma.$$

Then $V^* = V$ and $E^* = E$.

Proof. We start by proving that $V^* = V$.

First: $V^* \subseteq V$. Let $\text{init}(\mathbf{w}) = x$ and $\text{term}(\mathbf{w}) = y$; we claim that $\Gamma^*(\mathbf{w})(x) \in V$. By Theorem 3.1, $\Gamma^*(\mathbf{w}) = \gamma \Gamma^*(\mathbf{p}_{x,y})$ for some $\gamma \in \text{Sym}(\mathcal{N})$, so as $x, y \in S \subseteq V$, $\Gamma^*(\mathbf{w})(x) = \gamma(y) \in V$.

Second: $V \subseteq V^*$. If $y \in V$, there exists a path $\kappa(y) = y_0, y_1, y_2, \dots, y_n = y$ from $\kappa(y) \in S$ to $y \in V$; we claim that $y_i \in V^*$ for each i , and hence $y \in V^*$. We prove this for each $y_i \in V^*$ by induction on i , starting with $y_0 = \kappa(y) \in S \subseteq V$, with $y_0 = \Gamma^*(\epsilon_{\kappa(y)}) \in V^*$. If $y_i \in V^*$, then $y_i = \Gamma^*(\mathbf{v})(\kappa(y))$ for some \mathbf{v} , and note that by Theorem 3.2, $\Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v}) \in \text{Sym}(\mathcal{N})$. As y_{i+1} is adjacent to y_i , $y' = \Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v})^{-1}(y_{i+1})$ is adjacent to $\kappa(y_i)$, and if $y' \in S$, then as $\Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v}) \in \text{Sym}(\mathcal{N})$ and $(\kappa(y_i), y')$ is a local step, Theorem 3.1 gives us:

$$\begin{aligned} \Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v})(\kappa(y_i), y')(\kappa(y_i)) &= \Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v}) \Gamma^*((\kappa(y_i), y'))(\kappa(y_i)) \\ &= \Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v})(y') = y_{i+1} \end{aligned}$$

which is thus an element of V^* . Similarly, if $y' \notin S$, then $y' \in \partial S$, and it follows that $\Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v}) \beta_{y'}(\kappa(y_{i+1})) = \Gamma^*(\mathbf{p}_{\kappa(y_i), \kappa(y)} \mathbf{v})(y') = y_{i+1} \in V^*$.

For the rest of the proof, we use the fact that $V = V^*$ without further comment, i.e., V consists of precisely those points y in X such that for some $x \in S$ and some itinerary \mathbf{w} with $\text{init}(\mathbf{w}) = x$, $\Gamma^*(\mathbf{w})(x) = y$.

We now prove that $E^* = E$.

First, $E^* \subseteq E$. If $\{x, y\} \in E^*$, then for some itinerary \mathbf{w} with $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w}) = x$, $\Gamma^*(\mathbf{w})(\kappa(x)) = x$ and some $y' \in S \cup \partial S$ such that $(\kappa(x), y') \in \Sigma$ we have $\Gamma^*(\mathbf{w})(y') = y$. Using Lemma 3.2 and the fact that $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$, we have

$$\Gamma^*(\mathbf{w})(\kappa(x), y')(\kappa(x)) = \Gamma^*(\mathbf{w}) \Gamma^*((\kappa(x), y'))(\kappa(x)) = \Gamma^*(\mathbf{w})(y') = y.$$

As $(\kappa(x), y') \in \Sigma$, $\{\kappa(x), y'\} \in E$. As $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$, sending $\{\kappa(x), y'\}$ to $\{x, y\}$, we have $\{x, y\} \in E$.

And $E \subseteq E^*$. If $\{x, y\} \in E$, choose an itinerary \mathbf{w} such that $\text{init}(\mathbf{w}) = \text{term}(\mathbf{w}) = \kappa(x)$ and $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$, and $\Gamma^*(\mathbf{w})(\kappa(x)) = x$ (we can do this as $V = V^*$). As $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$,

$$\{\Gamma^*(\mathbf{w})^{-1}(x), \Gamma^*(\mathbf{w})^{-1}(y)\} = \{\kappa(x), \Gamma^*(\mathbf{w})^{-1}(y)\} \in E$$

as this is an edge within $S \cup \partial S$. Thus as $(\kappa(x), \Gamma^*(\mathbf{w})^{-1}(y)) \in \Sigma$ and Lemma 3.2 gives us

$$\begin{aligned} \Gamma^*(\mathbf{w}(\kappa(x), \Gamma^*(\mathbf{w})^{-1}(y)))(\kappa(x)) &= \Gamma^*(\mathbf{w})\Gamma^*((\kappa(x), \Gamma^*(\mathbf{w})^{-1}(y)))(\kappa(x)) \\ &= \Gamma^*(\mathbf{w})\Gamma^*(\mathbf{w})^{-1}(y) = y, \end{aligned}$$

and we have $\{x, y\} \in E^*$. ■

Finally, we get the third desideratum. Let's extend Γ^* to a traversal function Γ for the net $\mathcal{N} = \langle V, E \rangle$ as follows. Start with a transversal diagram $\langle S, \partial S, \Sigma \rangle$, and an assignment function Γ^* , and for each $x \in V$ and $\{x, y\} \in E$, choose an itinerary \mathbf{w} such that $\Gamma^*(\mathbf{w})(\kappa(x)) = x$ and let $y' = \Gamma^*(\mathbf{w})^{-1}(y) \in S \cup \partial S$. As $\Gamma^*(\mathbf{w}) \in \text{Sym}(\mathcal{N})$ maps $\kappa(x)$ to x and y' to y , $\{\kappa(x), y'\} \in E$, we have $(\kappa(x), y') \in \Sigma$.

We define the (alleged) traversal function as follows: for each $(x, y) \in V^2$ with $\{x, y\} \in E$, choose an itinerary \mathbf{w} such that $\Gamma^*(\mathbf{w})(\kappa(x)) = x$, and that for some $(\kappa(x), y') \in \Sigma$, $\Gamma^*(\mathbf{w})(y') = y$. Then set $\Gamma(x, y) = \Gamma^*(\mathbf{w})\Gamma^*((\kappa(x), y'))$. (Notice that we have a lot of discretion in choosing the itineraries \mathbf{w} .) Then

$$\begin{aligned} \Gamma(x, y)(x) &= \Gamma^*(\mathbf{w})\Gamma^*((\kappa(x), y'))(x) = \Gamma^*(\mathbf{w})\Gamma^*((\kappa(x), y'))\Gamma^*(\mathbf{w})^{-1}(x) \\ &= \Gamma^*(\mathbf{w})\Gamma^*((\kappa(x), y'))(\kappa(x)) = \Gamma^*(\mathbf{w})(y') = y, \end{aligned}$$

and repeating for all (x, y) , Γ is a traversal function.

Theorem 3.4 *The traversal function Γ is semistrict, i.e., for any $x_1, x_2, y_1, y_2 \in V$ such that $\{x_1, y_1\}, \{x_2, y_2\} \in E$ and for some $\gamma \in \text{Sym}(\mathcal{N})$, $\gamma(x_1) = x_2$ and $\gamma(y_1) = y_2$, there exist $\rho, \phi \in \text{Sym}(\mathcal{N})$ such that $\rho\Gamma(x_2, y_2) = \phi\Gamma(x_1, y_1)$.*

Proof. Suppose that the hypotheses hold and let $x = \kappa(x_1) = \kappa(x_2)$. There exist itineraries $\mathbf{v}_1, \mathbf{v}_2$ such that $\Gamma^*(\mathbf{v}_1)(x) = x_1$ and $\Gamma^*(\mathbf{v}_2)(x) = x_2$; note that as $\text{init}(\mathbf{v}_1) = \text{term}(\mathbf{v}_1)$ and $\text{init}(\mathbf{v}_2) = \text{term}(\mathbf{v}_2)$, we have $\Gamma^*(\mathbf{v}_1), \Gamma^*(\mathbf{v}_2) \in \text{Sym}(\mathcal{N})$. Let $y'_1 = \Gamma^*(\mathbf{v}_1)^{-1}(y_1)$ and $y'_2 = \Gamma^*(\mathbf{v}_2)^{-1}(y_2)$ and let $y = \kappa(y_1) = \kappa(y_2)$. Then we have

$$\Gamma^*(\mathbf{v}_1)\Gamma^*((x, y'_1)) = \Gamma(x_1, y_1) \quad \text{and} \quad \Gamma^*(\mathbf{v}_2)\Gamma^*((x, y'_2)) = \Gamma(x_2, y_2).$$

Letting $\beta_{\kappa(y)} = \text{id}$ (just in case $y'_1 = y$ or $y'_2 = y$), Theorem 3.1 implies $\Gamma^*((x, y'_1)) = \beta_{y'_1}\Gamma^*(\mathbf{p}_{x,y})$ and $\Gamma^*((x, y'_2)) = \beta_{y'_2}\Gamma^*(\mathbf{p}_{x,y})$, and we get $\Gamma^*((x, y'_2)) = \beta_{y'_2}\beta_{y'_1}^{-1}\Gamma^*((x, y'_1))$. Thus

$$\begin{aligned} \Gamma(x_2, y_2) &= \Gamma^*(\mathbf{v}_2)\Gamma^*((x, y'_2)) \\ &= \Gamma^*(\mathbf{v}_2)\left(\beta_{y'_2}\beta_{y'_1}^{-1}\Gamma^*((x, y'_1))\right) \\ &= \Gamma^*(\mathbf{v}_2)\left(\beta_{y'_2}\beta_{y'_1}^{-1}\right)\Gamma^*(\mathbf{v}_2)\Gamma^*(\mathbf{v}_1)^{-1}\Gamma(x_1, y_1), \end{aligned}$$

so setting $\rho = \Gamma^*(\mathbf{v}_2)\left(\beta_{y'_1}\beta_{y'_2}^{-1}\right)$ and $\phi = \Gamma^*(\mathbf{v}_2)\Gamma^*(\mathbf{v}_1)^{-1}$, we have $\rho\Gamma(x_2, y_2) = \phi\Gamma(x_1, y_1)$. ■

4 Implementation

Several programs implementing this algorithm or variants of this algorithm have been composed, primarily for application in crystal design. Of course, crystal design involves infinite nets embedded in Euclidean spaces, and this algorithm can be applied to finite and infinite nets in many kinds of spaces. In this last section, we consider computational complexity – or perhaps more precisely, computational resource – considerations in implementing this algorithm. In this brief account, we will consider finite nets generated in finite spaces, and (infinite) periodic nets generated in Euclidean spaces.

4.1 Finite Nets Generated in Finite Spaces

Suppose we wanted a single very large but finite graph, embedded in a very large but possibly finite space, whose given group of permutations would thus be large. We probably would not want to generate the entire space or the entire group; we would want to generate points in the space and elements of the group as needed. Then an implementation might work as follows. First, we set up the input.

- We assume a space X , on which a group \mathbb{G} of permutations acts transitively, and we can generate elements of X and \mathbb{G} as necessary. We are given $S, \partial S \subseteq X$ such that $S \cap \partial S = \emptyset$; these are intended as a fundamental transversal and its boundary in the net we are constructing. We are given $\Sigma \subseteq S^2 \cup (S \times \partial S)$ such that for each $y \in \partial S$, there exists $\kappa(y) \in S$ with $(\kappa(y), y) \in \Sigma$; this is intended as the set of (oriented) edges of the *initial transversal diagram*. This initial transversal diagram is $\mathcal{S} = \langle S, \partial S, \Sigma \rangle$.
- For simplicity, in this implementation, we are only concerned with the jumps in the itineraries, and not with local steps, so we can work with *jump itineraries*, i.e., itineraries that tell a farsighted traveler (one able to see an entire fundamental transversal) how to move from fundamental transversal to fundamental transversal. For each $y \in \partial S$, we are given $\beta_y \in \mathbb{G}$ such that for some $\kappa(y) \in S$ we have $\beta(\kappa(y)) = y$. Let \mathbb{W}^{jump} be the set of strings from the alphabet $\{\beta_y: y \in \partial S\}$, and call \mathbb{W}^{jump} the set of jump itineraries.

Let $\ell: \mathbb{Z}^{\leq 0} \rightarrow \mathbb{W}^{\text{jump}}$ enumerate jump itineraries; it will turn out that ℓ is the critical consideration for time complexity. Starting with the initial transversal diagram $\mathcal{S}_0 = \mathcal{S}$, we enumerate $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \dots$ while maintaining their unions $\mathcal{D}_n = \langle \bigcup_{i \leq n} (S_i \cup \partial S_i), \bigcup_{i \leq n} \Sigma_i \rangle$, which can be regarded as cumulative digraphs from which the desired net will be derived; note in the following that these unions and cumulative digraphs are not graphs as they have “oriented edges”, i.e., arcs.

Definition 4.1 *Given a digraph (or directed graph) $\langle V, A \rangle$, call it symmetric if, for every $x, y \in V$, either both or neither of (x, y) and (y, x) are elements*

of A . Given a symmetric digraph $\langle V, A \rangle$, say the associated graph is the graph $\langle V, E \rangle$, where $E = \{\{x, y\}: (x, y) \in A\}$.

In this implementation of the algorithm, we will also generate sets of “unused indices”; denote the first such set by $U_0 = \emptyset$. Again, we start with an initial transversal diagram $\mathcal{S}_0 = \langle S_0, \partial S_0, \Sigma_0 \rangle$ and $\mathcal{D}_0 = \langle S_0 \cup \partial S_0, \Sigma_0 \rangle$.

1. For the n th step, where $n > 0$, let $\ell(n) = (\beta_1, \beta_2, \dots, \beta_m)$ and $S_n = \beta_1\beta_2 \cdots \beta_m[S_0]$, $\partial S_n = \beta_1\beta_2 \cdots \beta_m[\partial S_0]$, and

$$\Sigma_n = \{(\beta_1\beta_2 \cdots \beta_m(x), \beta_1\beta_2 \cdots \beta_m(y)): (x, y) \in \Sigma_0\}.$$

and $\mathcal{S}_n = \langle S_n, \partial S_n, \Sigma_n \rangle$.

2. Continuing the n th step, if $S_n \cup \partial S_n \subseteq V_{n-1}$, for $\mathcal{D}_{n-1} = \langle V_{n-1}, A_{n-1} \rangle$, and if $\Sigma_n \subseteq A_{n-1}$, then n is an “unused index” and we set $U_n = U_{n-1} \cup \{n\}$ and $\mathcal{D}_n = \mathcal{D}_{n-1}$. On the other hand, if $S_n \cup \partial S_n \not\subseteq V_{n-1}$ or $\Sigma_n \not\subseteq A_{n-1}$, let $U_n = U_{n-1}$ and $\mathcal{D}_n = \langle V_{n-1} \cup S_n \cup \partial S_n, E_{n-1} \cup \Sigma_n \rangle$.
3. If $\mathcal{D}_n = \langle V_n, A_n \rangle$ is a symmetric digraph then halt, for the graph now consists of copies of the original fundamental transversal with no stray edges. Output \mathcal{D}_n 's associated graph $\mathcal{N} = \langle V, E \rangle$ where $V = V_n$ and $E = \{\{x, y\}: (x, y) \in A_n\}$ (and denote $U = U_n$). Otherwise, return to (1) for the $n + 1$ st step.

Notice that this program will halt iff the net it is generating is finite – provided that ℓ generates all of (or enough of) \mathbb{W}^{jump} .

For the rest of this subsection, for any $\beta_1, \dots, \beta_k \in \mathbb{G}$, denote the composition $\beta_1 \cdots \beta_k$ by $\llbracket (\beta_1, \dots, \beta_k) \rrbracket$. Thus for $\ell: \mathbb{Z}^{\geq 0} \rightarrow \mathbb{W}^{\text{jump}}$, if $\ell(k) = (\beta_1, \dots, \beta_k)$, then $\llbracket \ell(k) \rrbracket = \beta_1 \cdots \beta_k$. Observe that even if ℓ enumerates each word of \mathbb{W}^{jump} only once, $\llbracket \ell \rrbracket$ could enumerate the same group element repeatedly.

Perhaps the most unambiguous time measure is the number of transversals generated before the program halts; more refined measures would involve the time to write a group element or point of X , or the time to copy the entire transversal diagram (which should be proportional to its size), but presuming that such refinements will only introduce relatively small factors, we overlook them here.

Definition 4.2 *Suppose we are given a net $\mathcal{N} = \langle V, E \rangle$ on a space X , and a group \mathbb{G} of permutations of X . Suppose we are given a transversal diagram $\mathcal{S} = \langle S, \partial S, \Sigma \rangle$ of \mathcal{N} (with respect to \mathbb{G} and X). And suppose that for each $y \in \partial S$, we are given $\beta_y \in \mathbb{G}$ such that $\beta_y|_V$ is an automorphism of \mathcal{N} and that for some $\kappa(y) \in S$, $\beta_y(\kappa(y)) = y$. Given $\ell: \mathbb{Z}^{\geq 0} \rightarrow \mathbb{W}^{\text{jump}}$, let $\text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, X)$ be the least t such that $V = \bigcup_{n \leq t} \llbracket \ell(n) \rrbracket[S]$ and for each $\{x, y\} \in E$, $(x, y), (y, x) \in \bigcup_{n \leq t} \llbracket \ell(n) \rrbracket[\Sigma]$.*

But counting transversals generated means that the choice of the enumeration function ℓ is critical. For example ...

Perhaps the most naive choice for ℓ is to fix an ordering of $\partial S = \{y_1, y_2, \dots, y_{|\partial S|}\}$, and hence of $\{\beta_{y_i} : i = 1, \dots, |\partial S|\}$, and enumerate \mathbb{W}^{ump} so that $n < n' \implies |\ell(n)| \leq |\ell(n')|$, and use the ordering of ∂S to order each subsequence $\ell(|\partial S|^{m-1}), \dots, \ell(|\partial S|^m - 1)$ lexicographically. We will make use of this naive enumeration, although we will find that we can often greatly reduce time complexity by using an enumeration designed from properties the group elements $\{\beta_y : y \in \partial S\}$. As an extreme example of the advantage of custom-made enumeration, suppose that all the elements of $\{\beta_y : y \in \partial S\}$ commute. Then for any k and any permutation $\pi: [k] \rightarrow [k]$, $\beta_{\pi(1)}\beta_{\pi(2)} \cdots \beta_{\pi(k)}[S_0] = \beta_1\beta_2 \cdots \beta_k[S_0]$, so it would suffice if, for each k , ℓ enumerated all strings of length k of form

$$(\underbrace{\beta_{y_1}, \dots, \beta_{y_1}}_{k_1}, \underbrace{\beta_{y_2}, \dots, \beta_{y_2}}_{k_2}, \beta_{y_3}, \dots, \beta_{y_{|\partial S|-1}}, \underbrace{\beta_{y_{|\partial S|}}, \dots, \beta_{y_{|\partial S|}}}_{k_{|\partial S|}}),$$

for $k_1 + k_2 + \dots + k_{|\partial S|} = k$, lexicographically. While the naive enumeration would list $\sum_{n=0}^m |\partial S|^n$ transversals in enumerating all words of \mathbb{W}^{ump} of length at most m , which is exponentially many (with respect to m), the custom-made enumeration would list only

$$\sum_{n=0}^m \binom{n + |\partial S| - 1}{|\partial S| - 1} < ((|\partial S| - 1)!)^{-1} (m + |\partial S| - 1)^{|\partial S| - 1}$$

transversals, which is polynomially many (if the transversal order and size are fixed).

This blunt instrument gives us some crude estimates.

Proposition 4.1 *Let \mathbb{G} be a group of permutations of a space X , and let $\mathcal{N} = \langle V, E \rangle$ be a net embedded in X . Suppose that $\langle S, \partial S, \Sigma \rangle$ is a transversal diagram of \mathcal{N} (with respect to \mathbb{G}). Then for any appropriate set of $|\partial S|$ generators, using the definitions above,*

$$\lceil |V|/|S| \rceil - 1, \lceil 2|E|/|\Sigma| \rceil - 1 \leq \text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, X) - U \leq (|V| - |S|) + (2|E| - |\Sigma|).$$

Proof. First of all, observe that in the computation, for each $\{x, y\} \in E$, transversal diagrams containing both (x, y) and (y, x) will have to be generated, and thus transversal diagrams containing both x and y as interior vertices will have to be generated. Thus the number of transversal diagrams generated (in addition to the initial one) *and used* must be at least $\lceil |V|/|S| \rceil - 1$ and at most $|V| + 2|E| - |S| - |\Sigma|$.

Secondly, as two ordered pairs must be generated for each edge in E , the number of transversal diagrams generated (in addition to the initial one) must be at least $\lceil 2|E|/|\Sigma| \rceil - 1$ and at most $2|E| + |V| - |S| - |\Sigma|$. ■

We start with two examples exhibiting the dependence of time complexity on the enumeration function ℓ . Fix positive integers m, n , $m \leq n$. Recall an old group theoretic notation: for permutations of $[n]$, $(i_1 i_2 i_3 \cdots i_m)$ denotes the permutation β where $\beta(i_k) = i_{k+1}$ for $k < m$, and $\beta(i_m) = i_1$. Let \mathbb{G} be the group of permutations on $[n]$.

- Generate an m -clique in $[n]$ as follows. For the transversal diagram, choose vertex set $S_0 = \{1\}$, boundary $\partial S_0 = \{i: 1 < i \leq m\}$, and oriented edges $\Sigma_0 = \{(1, i): 1 < i \leq m\}$. For each $i \in [m] - \{1\}$, let $\beta_i = (1\ i)$ be the group action used to map the original fundamental transversal $\{1\}$ to a copy $\{i\}$. Then if $\ell(0)$ is the identity and $\ell(k) = \beta_{k-1}$ for $k < m$, then the first m transversals are used, and after enumerating them, the computation halts: $\text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, [n]) = m - 1$.
- Generate an m -cycle in $[n]$ as follows. For the transversal diagram, choose vertex $\{1\}$, boundary $\{2, m\}$, and oriented edges $\{(1, 2), (1, m)\}$. Let $\beta_+ = (1\ 2\ 3 \cdots m)$ and $\beta_- = \beta_+^{-1} = (1\ m\ m-1 \cdots 2)$. Let's consider two quite different enumerations.
 - If $\ell_1(k) = (\beta_+, \beta_+, \dots, \beta_+)$ is of length k so that $\llbracket \ell_1(k) \rrbracket = \beta_+^k$ for each $k \geq 0$, then after $m - 1$ iterations, the algorithm has generated m transversals making up the entire m -cycle. By Proposition 4.1, this is optimal, as here $U = 0$, the number of vertices is m , the number of vertices in the transversal is 1, and the time is m , so $\lceil |V|/|S| \rceil - 1 \leq \text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, X) - U$ becomes $m/1 - 1 \leq (m-1) - 0$.
 - If ℓ_2 enumerates the words of $\{\beta_+, \beta_-\}$ by length and then lexicographically – so that $\llbracket \ell(0) \rrbracket = \text{id}$, $\llbracket \ell(1) \rrbracket = \beta_+$, $\llbracket \ell(2) \rrbracket = \beta_-$, $\llbracket \ell(3) \rrbracket = \beta_+\beta_+ = \beta_+^2$, $\llbracket \ell(3) \rrbracket = \beta_+\beta_- = \text{id}$, $\llbracket \ell(4) \rrbracket = \beta_-\beta_+ = \text{id}$, $\llbracket \ell(5) \rrbracket = \beta_-\beta_- = \beta_-^2$, etc. – then generating the entire cycle will take $2^{\lceil m/2 \rceil}$ iterations.

Here the choice of the enumeration can make the difference between linear and exponential time.

See Figure 9.

Sometimes the optimal enumeration does not meet the inequalities of Proposition 4.1. Say that a d -tree is a tree (i.e., an acyclic graph from which we select a “root” vertex) whose root vertex is of degree d and whose non-root non-leaf vertices are all of degree $d + 1$. Recall that a tree is *complete* if each branch (path from the root to a leaf) is the same length, that length (counting edges) being the *rank* of the tree.

Choose positive integers r and m , and suppose that $X = [m + 1] \times [r^m]$. We can embed a complete r -tree of rank m in X as follows. Let $f: [r]^m \rightarrow [r^m]$ be a lexicographic ordering of $[r]^m$, and employ the following recursion:

- Let the root of the tree be $(1, 1)$, and its neighbors are $(2, f(1, 1, \dots, 1))$, $(2, f(2, 1, \dots, 1))$, $(2, f(m, 1, \dots, 1))$.
- For each k , $1 < k \leq m$, if $(k, f(i_1, i_2, \dots, i_{k-1}, 1, 1, \dots, 1))$ is a vertex of the tree, add edges to new vertices $(k+1, f(i_1, i_2, \dots, i_{k-1}, ir^{m-k+1}, 1, \dots, 1))$, $i \in [r]$.

A fundamental transversal will then consist of the vertices of some branch. For example, we could choose as our initial transversal diagram the branch



Figure 9: *Left.* A 5-clique \mathcal{N} may be generated on $[6]$ by taking one point 1 as a fundamental transversal, leaving one point, 1 in blue, not in the clique. The transversal diagram then consists of that point, four other points 2, 3, 4, and 5, and edges connecting 1 to each i , $1 < i < 6$. For each such i , let β_i be the permutation $(1\ i)$, and the original transversal diagram \mathcal{S} (in red) together with the additional four copies $\beta_i[\mathcal{S}]$ make up the 5-clique (purple and red). A breadth-first enumeration ℓ satisfies $\text{time}(\mathcal{N}, \mathcal{S}, \ell; \text{Perm}([6]), [6]) = 5$ and $U = 0$. *Right.* A cycle may be generated from the original transversal diagram \mathcal{S} (in red) applying one of the two permutations $\beta_+ = (1\ 2\ 3\ 4\ 5)$ and $\beta_- = \beta_+^{-1} = (1\ 5\ 4\ 3\ 2)$ repeatedly: such a depth first enumeration succeeds after generating five transversals: $\mathcal{S}, \beta_+[\mathcal{S}], \beta_+^2[\mathcal{S}], \beta_+^3[\mathcal{S}], \beta_+^4[\mathcal{S}]$, with $U = 0$. However, a breadth first enumeration (with “+” preceding “-” lexicographically) generates $\mathcal{S}, \beta_+[\mathcal{S}], \beta_-[\mathcal{S}], \beta_+^2[\mathcal{S}], \beta_+\beta_-[\mathcal{S}], \beta_-\beta_+[\mathcal{S}], \beta_-^2[\mathcal{S}]$, with $\text{time}(\mathcal{N}, \mathcal{S}, \ell; \text{Perm}([6]), [6]) = 7$ and $U = 2$.

of vertices $(k, f(1, 1, \dots, 1))$, $k \in [m + 1]$, having the internal oriented edges $((k, f(1, 1, \dots, 1)), (k + 1, f(1, 1, \dots, 1)))$ and $((k + 1, f(1, 1, \dots, 1)), (k, f(1, 1, \dots, 1)))$, and with exterior oriented edges $((k, f(1, 1, \dots, 1)), (k + 1, f(2, 1, \dots, 1)))$ to boundary vertices $(k + 1, f(2, 1, \dots, 1))$. For each k , let β_k be the permutation of X defined by

$$\beta_k(i, n) = \begin{cases} (i, n) & i < k \\ (i, n + r^{m-k+1}) & i \geq k \ \& \ n + r^{m-k+1} \leq r^m \\ (i, n + r^{m-k+1} - r^m) & i \geq k \ \& \ n + r^{m-k+1} > r^m \end{cases}$$

and notice that $\kappa((k, f(1, 1, \dots, 1, 2, 1, \dots, 1))) = (k, f(1, 1, \dots, 1, 1, 1, \dots, 1))$, where the 2 is in the $k + 1$ st argument. See Figure 10.

For definitiveness, suppose that \mathbb{G} is the set of permutations of X generated by elements

$$\gamma_{d_1, d_2, \dots, d_{m+1}}(i, n) = \begin{cases} (i, n + d_i) & n + d_i \leq r^m \\ (i, n + d_i - r^m) & \text{otherwise.} \end{cases}$$

(In addition, to make \mathbb{G} act transitively on X , add as a generator the permutation

$$\gamma(i, n) = \begin{cases} (i + 1, n) & i < m + 1 \\ (1, n) & i = m + 1. \end{cases}$$

Notice that \mathbb{G} is abelian.

To cover this tree with copies of the first transversal, it suffices to enumerate r^m transversal diagrams, including the first one, each having its own leaf. This can be done by enumerating words

$$(\underbrace{\beta_2, \dots, \beta_2}_{e_2}, \underbrace{\beta_3, \dots, \beta_3}_{e_3}, \beta_4, \dots, \beta_m, \underbrace{\beta_{m+1}, \dots, \beta_{m+1}}_{e_{m+1}}) \in \mathbb{W}^{\text{jump}},$$

for if we took S_0 as the first fundamental transversal, then the images

$$\beta_2^{e_2} \beta_3^{e_3} \dots \beta_{m+1}^{e_{m+1}} [S_0]$$

cover the leaves, and hence the entire tree, with r^m transversals: for the tree \mathcal{N} , given the initial branch for the transversal diagram \mathcal{S} , $\text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, X) = r^m$ and $U = 0$.

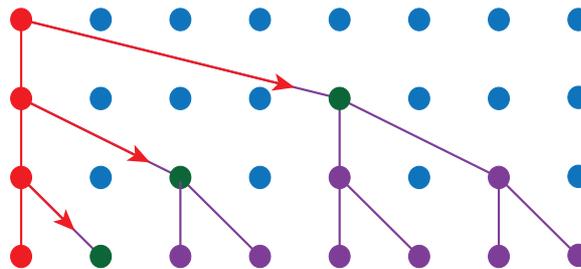


Figure 10: Embed a 2-tree of rank $m = 3$ into $X = [3 + 1] \times [2^3]$; the blue points are those that are not vertices of the tree. The red vertices being a fundamental transversal, and the red edges the resulting alphabet of the transversal diagram, the green vertices being the boundary vertices, and the purple vertices the rest.

Since the tree consists of $(1 + o(1))r^m$ vertices, the time complexity for a complete r -tree – assuming an optimal enumeration of \mathbb{W}^{jump} – is proportional to the order of the tree. (Of course, as the the transversal diagram is of logarithmic size with respect to the size of the entire tree, we are overlooking a logarithmic factor in this rough estimate.) However, in this case the transversal diagrams generated are *not* mutually disjoint, and in fact while the optimal ordering gave $U = 0$, so that the inequalities of Proposition 4.1 became

$$\begin{aligned} (1 + o(1))r^m m^{-1} = [|V|/|S|] - 1 &\leq \text{time}(\mathcal{N}, \mathcal{S}, \ell; \mathbb{G}, X) \\ &\leq |V| + 2|E| - |S| - |\Sigma| = 3(1 + o(1))r^m, \end{aligned}$$

and it is the right inequality that gives the better estimate, albeit off by a factor of 3. And that estimate cannot be improved upon as all automorphisms of the tree require fundamental transversals of $m + 1$ vertices.

We conclude that for naive implementations of this algorithm, for generating very large graphs, the time complexity is highly dependent on the enumeration ℓ of \mathbb{W}^{jump} . The result is a tradeoff: the more ℓ is intelligently designed, the lower

is U . Clearly, for any net \mathcal{N} embedded in any space X with given fundamental transversal and group elements, there is an enumeration ℓ of \mathbb{W}^{jump} such that the corresponding sequence of transversal diagrams covers \mathcal{N} with $U = 0$. On the other hand, the computational complexity of ℓ itself may become an issue. So we conclude this subsection with a question.

Question 4.1 *Given a net \mathcal{N} generated from a fundamental transversal S using words of \mathbb{W}^{jump} from a group \mathbb{G} of permutations on a finite space X , what upper bounds on the time complexity hold for the enumeration function ℓ and the generation of \mathcal{N} from copies of the initial transversal diagram?*

5 The Crystal Turtlebug Programs

The previous subsection involved generating nets we knew we would obtain from given transversals. Suppose that all we wanted was *some* net (with nice properties) that we do not know, which would be generated from putative transversal diagrams, given some space and group. Then the computational resources required depends not only on the complexity of the implemented algorithm (and the enumeration of group elements) but also on the geometry of the underlying space. We concentrate on the motivating application: a program that enumerates blueprints for crystal design. For details on crystallographic groups in the following discussion, see [38].

If all we wanted was a unit cell of a net we knew, then as a unit cell for a three dimensional crystal net requires at most 48 fundamental transversals, such a computation would not be onerous. But what a chemist desires is a not one net consisting of a single unit cell of a known crystal; this is, in fact, the sort of image readily available in such databases as the Reticular Chemistry Structure Resource (RCSR, [27]), TOPOS ([5] and [6]), and the Euclidean Patterns in Non-Euclidean Tilings (EPINET, [28])⁵. In fact, what a chemist, crystallographer, or chemical engineer desires is a catalogue of chemically plausible nets for novel crystals. We typically do not know fundamental transversals and space groups that generate chemically plausible nets, so the Crystal Turtlebug program described in [23] traverses a search space of fundamental transversals (with vertices at integer points, so there are countably many of these) and crystallographic space groups (which may be specified by selecting three vectors (also integer vectors) representing the unit cell, and then which of the 230 conjugacy classes of crystallographic space groups to employ). (Actually, the current version of The Crystal Turtlebug uses a less efficient but more modularly adjustable variant of this algorithm.) The cost of the catalogue arises from the size of the search space of transversal diagrams – and of vectors and parameters defining crystallographic groups.

For example, the first program, described in [10], generates uninodal (i.e., vertex transitive) nets. The fundamental transversal consists of a single node

⁵ These are crystal enumeration databases; the big database used by the crystallographic community is the Cambridge Structural Database, maintained by the Cambridge Crystallographic Data Centre, with 596,810 entries as of 1 January 2012; see, e.g., [2].

(at $(0, 0, 0) \in \mathbb{R}^3$) and e orbits of edges; each orbit of edges can be fixed by a single edge (the rest generated by application of the group elements), which can be fixed by an integer vector. Then one chooses a crystallographic space group, which (in the most naive implementation) consists of selecting three vectors (also integer vectors) representing the unit cell, and then which of the 230 conjugacy classes of crystallographic space groups to employ. Thus one chooses $e + 3$ vectors in \mathbb{Z}^3 , and an element of a set of 230. Assuming that the edge vectors were somewhat smaller than the unit cell vectors, one could imagine selecting the former from a $m \times m \times m$ cube and the latter from an $M \times M \times M$ cube, and the most naive implementation will run the algorithm on $230m^{3e}M^3$ initial conditions. For three orbits of edges, with $m = 3$ and $M = 10$, that's 4.5 billion nets to generate. And that's just for uninodal nets.

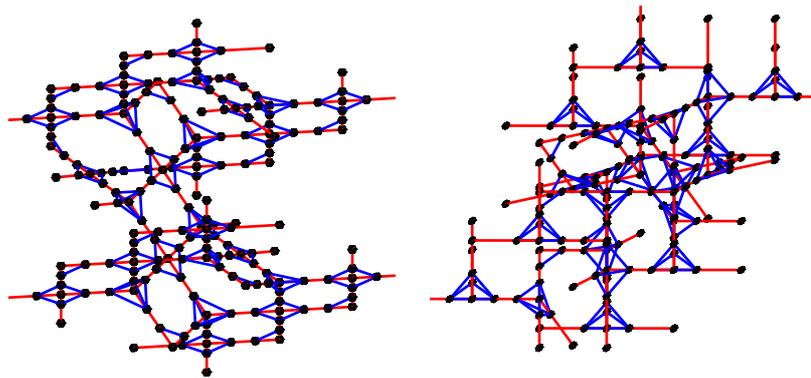


Figure 11: Two isomorphic nets found by a early version of the Crystal Turtlebug program, but were unknown to Systre and the RCSR database. The vertices of the net at left has (non-planar) tetrahedral and (planar) quadrilateral vertex figures, while the net at right has two orbits of vertices with (non-planar) tetrahedral vertex figures. Thus the two nets are not congruent, even though Systre confirmed that they are isomorphic. For more examples, and more extended discussion of them, see [23].

From the mathematician's point of view, the exponential explosion is a problem if one intends to use a program like the Crystal Turtlebug to do experimental mathematics. After all, these nets are geometric objects, and it would be interesting to explore their properties. They seem to resist analysis: after a century of being subjects of great scientific interest, the mathematical theory of these rather difficult objects is still a bit thin. Surveys and studies of these nets could prove helpful in suggesting directions for the theory to be developed, but the exponential explosion suggests that even highly restricted comprehensive surveys are out of the question.

But the demand for crystal designs is likely to grow substantially during the coming decades. And hence, the demand for a better theoretical understanding

of these designs, and of algorithms for their synthesis and analysis, is likely to grow as well.

Acknowledgments

This project began with Edwin Clark and the author, and Clark composed a Maple program that enumerated vertex transitive nets, including a number of novel ones. The underlying rationale for this algorithm, which in turn rests on [29], is described in [10], which can be regarded as a uninodal version of the algorithm heuristic described in this article. I would like to thank Edwin Clark for his assistance, in computer programming, in the development of the theory, and for feedback on this paper. I would like to thank John Conway, Mile Krajcekvski, and David Rabson for useful discussions, and I would like to thank the referees for their useful suggestions. I would also like to acknowledge the support of the University of South Florida, which provided me with a spring sabbatical in 2008.

References

- [1] H. Abelson and A. A. deSessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. MIT Pr., 1980.
- [2] F. H. Allen. The cambridge structural database: A quarter of a million crystal structures and rising. *Acta Cryst.*, B58:380–388, 2002.
- [3] C. Baerlocher, L. B. McCusker, and D. H. Olson. *Atlas of Zeolite Framework Types*. Elsevier, 6th edition, 2007.
- [4] A. L. Bail. Inorganic structure prediction with grinsp. *J. Appl. Cryst.*, 38:389–395, 2005.
- [5] V. A. Blatov. Multi purpose crystallochemical analysis with the program package topos. *IUCr Compcom Newsletter*, 7:4–38, 2006.
- [6] V. A. Blatov, A. P. Shovchenko, and V. N. Seezhkin. Topos-3.2 - a new version of the program package for multipurpose crystal-chemical analysis. *J. Appl. Cryst.*, 33(4):1193, 2000.
- [7] N. R. Champness. Making coordination frameworks. In E. Braga and F. Grepioni, editors, *Making Crystals by Design: Methods, Techniques and Applications*, pages 193–207. Wiley, 2007.
- [8] L. S. Charlap. *Bieberbach Groups and Flat Manifolds*. Springer-Verlag, 1986.
- [9] S. J. Chung, T. Hahn, and W. E. Klee. Nomenclature and generation of three-periodic nets: the vector method. *Acta Cryst.*, A40:42–50, 1984.
- [10] W. E. Clark. Notes on uninodal nets, 2008. <http://shell.cas.usf.edu/~eclark/UninodalNetNotes.doc>.
- [11] H. S. M. Coxeter. *Regular Polytopes*. Dover, corrected 2nd ed. edition, 1973.
- [12] M. Davis. *The Geometry and Topology of Coxeter Groups*. Princeton U. Pr., 2007.
- [13] O. Delgado-Friedrichs, A. W. M. Dress, D. H. Huson, J. Kłowski, and A. L. MacKay. Systematic enumeration of crystalline networks. *Nature*, 400:644–647, 12 Aug. 1999.
- [14] O. Delgado-Friedrichs, M. D. Foster, M. O’Keeffe, D. Proserpio, M. M. J. Treacy, and O. M. Yaghi. What do we know about three-periodic nets? *J. Solid State Chem.*, 178(8):2533–2554, 2005.
- [15] O. Delgado-Friedrichs and M. O’Keeffe. Crystal nets as graphs: Terminology and definitions. *J. Solid State Chem.*, 178:2480–2485, 2005.

- [16] W. Dicks and M. J. Dunwoody. *Groups acting on graphs*. Cambridge U. Pr., 1989.
- [17] I. N. Herstein. *Topics in Algebra*. Wiley, 2nd edition, 1975.
- [18] R. James. Objective structures. *J. Mech. Physics Solids*, 54:2354–2390, 2006.
- [19] N. Jonoska and G. McColm. Flexible versus rigid tile assembly. In C. S. C. et al, editor, *5th International Conference on Unconventional Computation (LNCS 4135)*, pages 421–436. Springer-Verlag, 2006.
- [20] N. Jonoska and G. McColm. Describing self-assembly of nanostructures. In V. Geffert, J. Karhumki, and A. B. et al, editors, *SOFSEM 2008: Theory and Practice of Computer Science (LNCS 4910)*, pages 66–73. Springer-Verlag, 2008.
- [21] W. E. Klee. Crystallographic nets and their quotient graphs. *Cryst. Res. Technol.*, 39(11):959–968, 2004.
- [22] L. MacGillivray. *Metal-Organic Frameworks*. Wiley, 2010.
- [23] G. L. McColm, W. E. Clark, M. Eddaoudi, L. Wojtas, and M. Zaworotko. Crystal engineering using a “turtlebug” algorithm: A de novo approach to the design of binodal metal-organic frameworks. *Crystal Growth & Design*, 11(9):3686–3693, 2011.
- [24] J. Meier. *Groups, Graphs and Trees: An Introduction to the Geometry of Infinite Groups*. Cambridge U. Pr., 2008.
- [25] I. U. of Crystallography. Report of the executive committee. *Acta Cryst.*, A48:922–946, 1992.
- [26] M. O’Keeffe and B. G. Hyde. *Crystal Symmetry I: Patterns and Symmetry*. Mineralogical Society of America, 1996.
- [27] 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):1782–1789, 2008. <http://www.rcsr.anu.edu.au>.
- [28] S. J. Ramsden, V. Robins, S. T. Hyde, and S. Hungerford. Epinet: Euclidean patterns in noneuclidean tilings, 2005. <http://epinet.anu.edu.au>.
- [29] G. Sabidussi. Vertex-transitive graphs. *Monatsh. Math.*, 68:426–438, 1964.
- [30] R. L. E. Schwarzenberger. *N-dimensional crystallography*. Pitman, 1980.
- [31] J.-P. Serre. *Trees*. Springer-Verlag, 1980. trans. J. Stillwell.
- [32] M. M. J. Treacy, K. H. Randall, and S. Rao. The search for new zeolite frameworks. In *Proc. 12th Internat’l Zeolite Conf.*, pages 517–532, 1999.

- [33] M. M. J. Treacy, I. Rivin, E. Balkovsky, K. H. Randall, and M. D. Foster. Enumeration of periodic tetrahedral frameworks. ii. polynodal graphs. *Microporous and Mesoporous Materials*, 74(1-3):121–132, 2004.
- [34] E. B. Vinberg and O. V. Shvartsman. Discrete groups of motions of spaces of constant curvature. In E. B. Vinberg, editor, *Geometry II: Spaces of Constant Curvature*. Springer-Verlag, 1993.
- [35] A. Wells. *Three-dimensional Nets and Polyhedra*. Wiley, 1977.
- [36] C. E. Wilmer, M. Leaf, C. Y. Lee, O. K. Farha, B. G. Hauser, J. T. Hupp, and R. Q. Snurr. Large-scale screening of hypothetical metal-organic frameworks. *Nature Chemistry*, 4:83–89, 2012.
- [37] O. M. Yaghi, M. O’Keeffe, N. Ockwig, H. K. Chae, M. Eddadoudi, and J. Kim. Reticular synthesis and the design of new materials. *Nature*, 423:705–714, 2003.
- [38] P. Yale. *Geometry and Symmetry*. Holden-Day, 1968.