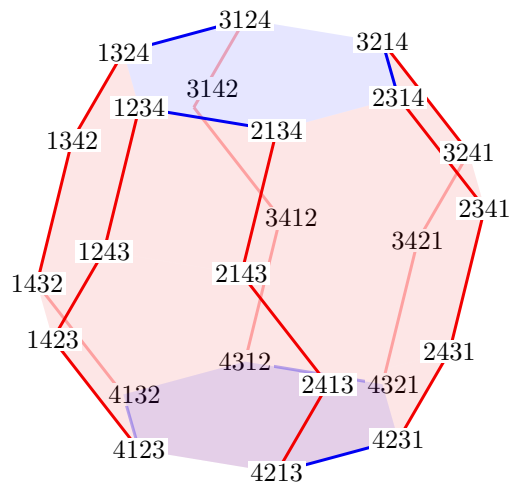


Open problems booklet

Combinatorics, Algorithms, and Geometry Workshop 2026 Kassel



This workshop is funded by the DFG Priority Programme SPP 2458 'Combinatorial Synergies', and the DFG Heisenberg Grant 522790373 'Principles of Combinatorial Algorithms'.

1: A warm-up mystery

(suggested by Aaron Williams)

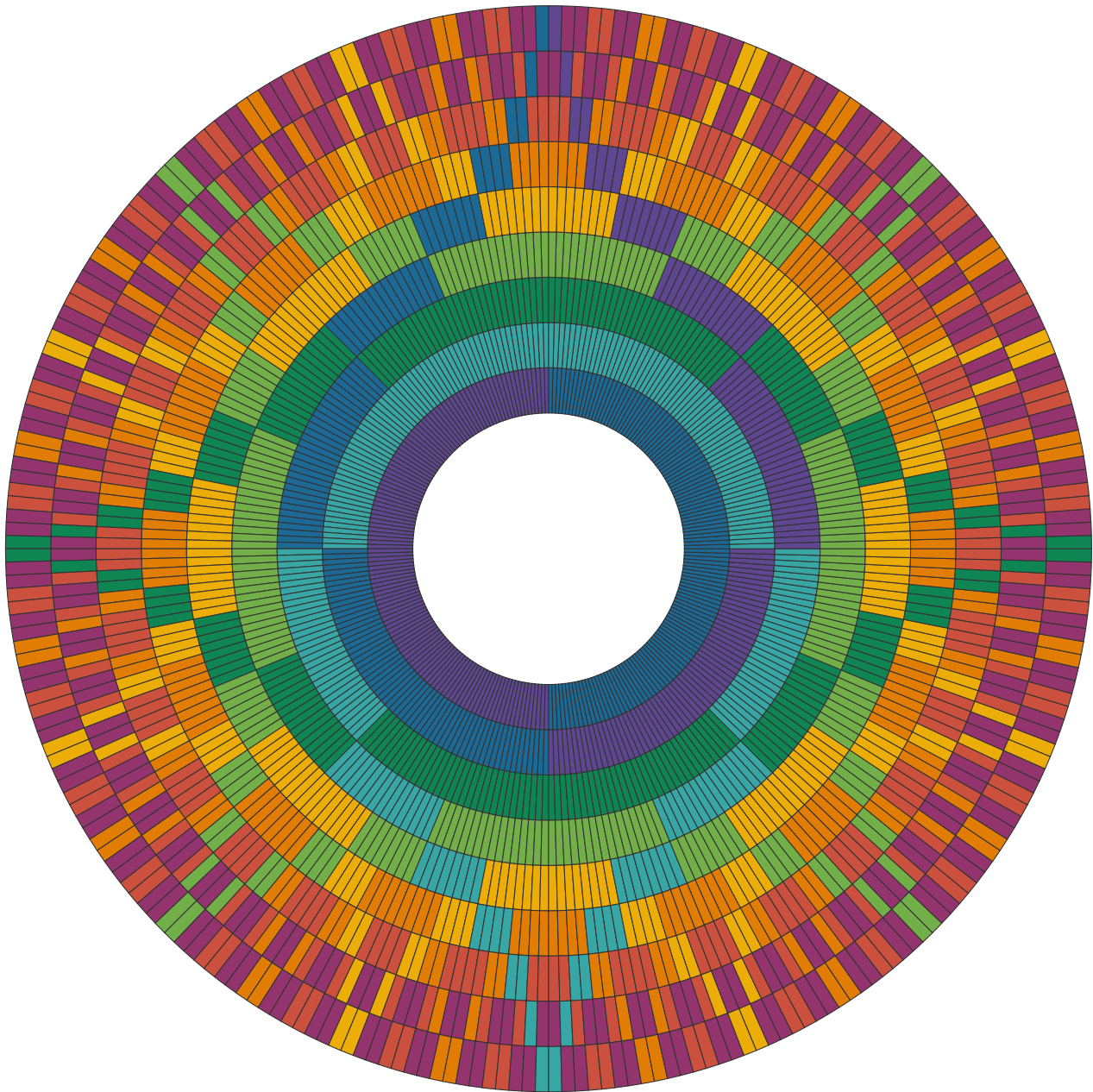


Figure 1: *Mysterious figure.*

Question: What does Figure 1 show?

2: Constant factor approximations for the flip distance

(suggested by Oswin Aichholzer, Joseph Dorfer, Peter Kramer, and Christian Rieck)

Consider a triangulation of a convex polygon with n vertices. A *flip* in a triangulation is an operation that removes one edge from the triangulation and adds another such that the resulting structure is again a triangulation. Such a flip is performed by taking two triangles that share an edge, removing the shared edge and adding the other diagonal of the resulting convex quadrilateral. For an example, see Figure 2. A *flip sequence* between two triangulations T and T' is a sequence $T = T_0, T_1, \dots, T_{k-1}, T_k = T'$ in which two consecutive triangulations only differ by a single valid flip. The index k denotes the *length* of a flip sequence. The *flip distance* of two triangulations T and T' is the minimum length of a flip sequence from T to T' .

In 1986 Sleator, Tarjan, and Thurston [5] proved that there always exists a flip sequence from any triangulation of a convex n -vertex polygon to any other triangulation of the same polygon of length at most $2n - 10$ for $n > 12$. The proof produces a flip sequence that flips to a fan-shaped intermediate configuration as depicted in Figure 2. There are examples for which this bound on the flip distance is tight [4].

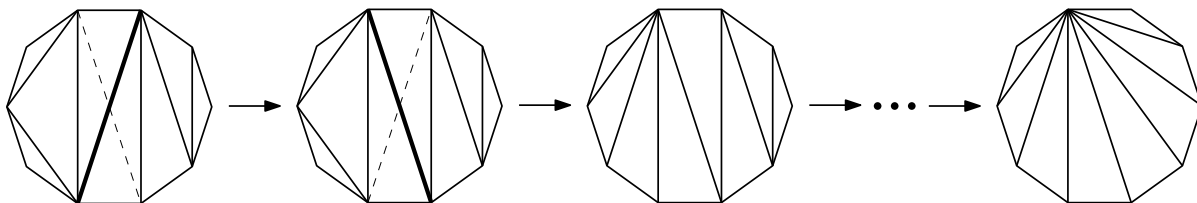


Figure 2: A flip sequence to a fan triangulation. One flip is especially highlighted at the start.

Until now, this approach yields the best possible approximation algorithm for the flip distance problem by providing a 2-factor approximation by the following modifications: Observe that $|T \setminus T'|$ gives a trivial lower bound on the flip distance between T and T' . If an interior edge e is contained in both T and T' , then we split both triangulations along e obtaining two pairs of smaller triangulations. The flip distance from T to T' is then upper bounded by the sum of the flip distances between the smaller triangulations. (In fact, the flip distance is even equal to the sum of flip distances between the smaller triangulations as proven in a lemma in [5]). We repeat this procedure until no common interior edges are left. In the approach from [5], every interior edge is flipped at most twice. Therefore, the flip distance is at least $|T \setminus T'|$ and at most $2 \cdot |T \setminus T'|$.

Question 1: Can we provide a better polynomial time computable constant factor approximation with approximation ratio $2 - \epsilon$ for some $\epsilon > 0$? How large can we make ϵ ?

There are some known partial results in that direction. These include (1) providing an upper bound on the flip distance in the number of crossings of $T \cup T'$ [2]; (2) heuristics that efficiently compute upper and lower bounds on the flip distance, but without any approximation guarantee, see [1] and the related work thereof; and (3) algorithms with a better approximation ratio in case T and T' are of a special form [3].

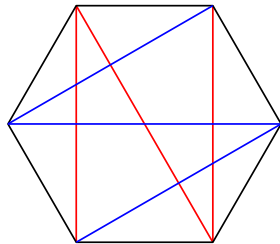


Figure 3: A pair of triangulations in which the flip distance is strictly larger than $|T \setminus T'|$.

Some further thoughts: In Figure 3, we provide an example in which the flip distance is 4 while $|T \setminus T'| = 3$. The reason is that we cannot directly add an edge of one triangulation to the other is that every edge in one triangulation is crossed twice by edges from the other triangulation.

Can we argue that we can always either make many “good” flips that exchange many edges from the initial triangulation with edges from the target triangulation or find a “bad” substructure that allows us to argue that we cannot do better? If we can always do either of the two, this would allow us to make arguments towards an improved approximation ratio.

We further throw in a different viewpoint on the problem:

Question 2: Is there an approximation scheme of any form to approximate the flip distance?

The idea behind Question 2 is that the approximation factor might improve if we are allowed to brute force increasingly larger “small” subinstances of triangulations. For a fixed size of instances that we brute force, the runtime is polynomial. However, the runtime of the brute force approach may grow exponentially with the size of the instances.

References

- [1] BARIL, J.-L., AND PALLO, J.-M. Efficient lower and upper bounds of the diagonal-flip distance between triangulations. *Information Processing Letters* 100, 4 (2006), 131–136.
- [2] HANKE, S., OTTMANN, T., AND SCHUIERER, S. The edge-flipping distance of triangulations. *Journal of Universal Computer Science* 2, 8 (1996), 570–579.
- [3] LI, M., AND ZHANG, L. Better approximation of diagonal-flip transformation and rotation transformation. In *International Computing and Combinatorics Conference* (1998), Springer, pp. 85–94.
- [4] POURNIN, L. The diameter of associahedra. *Advances in Mathematics* 259 (2014), 13–42.
- [5] SLEATOR, D. D., TARJAN, R. E., AND THURSTON, W. P. Rotation distance, triangulations, and hyperbolic geometry. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing* (1986), pp. 122–135.

3: Properties of the flip graph of odd matchings

(suggested by Oswin Aichholzer, Joseph Dorfer, Christian Rieck, and Francesco Verciani)

We study the reconfiguration of *odd matchings*, that is, matchings in a graph that leave exactly one vertex unmatched. In a reconfiguration step, called a *flip*, one edge of the matching is modified. Specifically, the unmatched vertex is matched, and consequently exactly one previously matched vertex becomes unmatched.

There are two distinct settings as depicted in Figure 4. In the *geometric setting*, vertices are points in the plane and all odd matchings are crossing-free. In the *combinatorial setting*, we study odd matchings in arbitrary graphs without geometric restrictions.

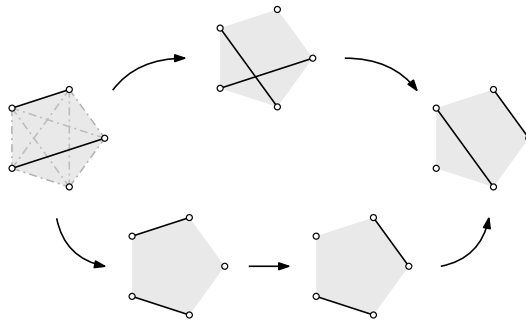


Figure 4: *Flips of odd matchings in the geometric and combinatorial setting in counterclockwise and clockwise order, respectively.*

Given a collection of configurations together with a specified set of permitted reconfiguration steps, each transforming one configuration into another, we naturally obtain the *flip graph* of the configurations, a graph whose vertices represent configurations and whose edges correspond to the allowed moves.

While the flip graph of odd matchings is connected in the geometric setting [2], this property does not hold in general in the purely combinatorial setting. Nevertheless, a simple characterization of the class of graphs whose flip graphs are connected has been established [1].

In case of connected flip graphs, one may be interested in shortest paths between any two configurations. From a computational complexity perspective, deciding whether there exists a flip sequence transforming two odd matchings into one another with at most k flips is NP-hard in both settings. We can, however, make statements about the diameter of the flip graph. In the combinatorial setting, we obtain a diameter of $\Theta(n)$. In the geometric setting, we currently only have a lower bound of $\Omega(n)$ and an upper bound of $\mathcal{O}(n^2)$.

Question 1: Can we improve the lower or upper bound on the diameter of the flip graph of odd matchings in the geometric setting?

Moreover, reconfiguration is deeply connected to *Gray codes* [3], which aim to enumerate all feasible configurations in a sequence where consecutive configurations differ by exactly one flip. In other words, a Gray code is a Hamiltonian cycle in the respective flip graph. This perspective emphasizes not only reachability between solutions, but also the structure of the solution space itself.

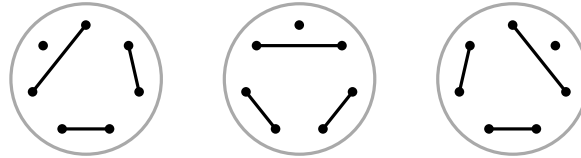


Figure 5: Configurations of this type, namely those defined by a single chord that separates one vertex from the remaining vertices, have exactly two adjacent configurations in the flip graph. For a convex point set of odd size $n \geq 7$, these n configurations collectively induce a cycle of length $2n$, which does not span all vertices of the flip graph.

In case of the geometric setting, the flip graph of odd matchings on point sets in convex position does not contain a Hamiltonian cycle, as depicted in Figure 5.

The same is much less clear in the combinatorial setting. For example, the flip graph of odd matchings for K_5 contains a Hamiltonian cycle, as visualized in Figure 6.

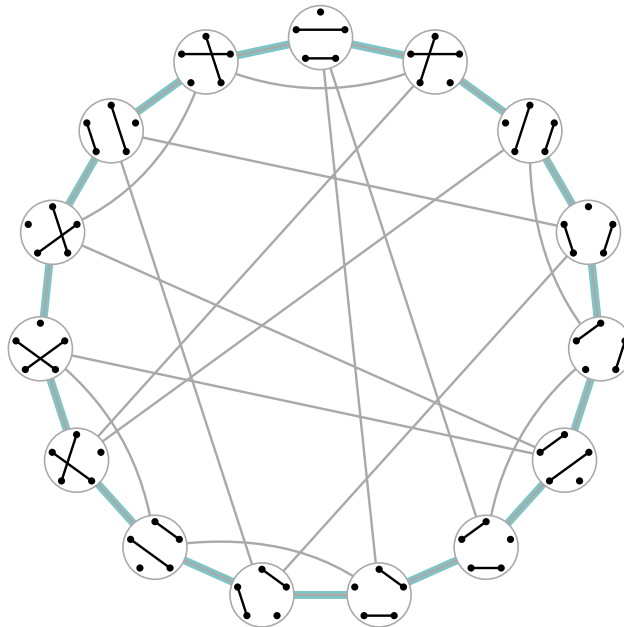


Figure 6: The flip graph of odd matchings in K_5 , together with a Hamiltonian cycle.

Accordingly, we pose the following question:

Question 2: Is the flip graph of odd matchings in complete graphs Hamiltonian?

Note that this flip graph is vertex-transitive.

References

[1] AICHHOLZER, O., BRENNER, S., DORFER, J., HOANG, H. P., PERZ, D., RIECK, C., AND VERCIANI, F. Flipping odd matchings in geometric and combinatorial settings. In *International Symposium on Graph Drawing and Network Visualization (GD)* (2025), pp. 12:1–12:18.

- [2] AICHHOLZER, O., BRÖTZNER, A., PERZ, D., AND SCHNIDER, P. Flips in odd matchings. *Comput. Geom.* 129 (2025), 102184.
- [3] MÜTZE, T. Combinatorial Gray codes—an updated survey. *Electron. J. Combin.* DS26, Dynamic Surveys (2023), 93 pp.

4: Packing edge-disjoint plane graphs on point sets

(suggested by Linda Kleist)

Let S be a set of n points in the plane. How many edge-disjoint crossing-free straight-line spanning paths does S allow? For points in convex position, there always exist $\lfloor n/2 \rfloor$ such paths, for an illustration consider Figure 7. As the complete graph has $n/2(n - 1)$ edges, the bound is best possible. For general point sets, Kindermann, Kratochvíl, Liotta, and Valtr [5] show that there always exists 3 edge-disjoint paths and that there exist point sets that allow for at most $\lfloor n/3 \rfloor$ edge-disjoint paths. The question at hand is to improve these bounds.

Question 1: How many edge-disjoint spanning paths can we guarantee for each set of n points in the plane?

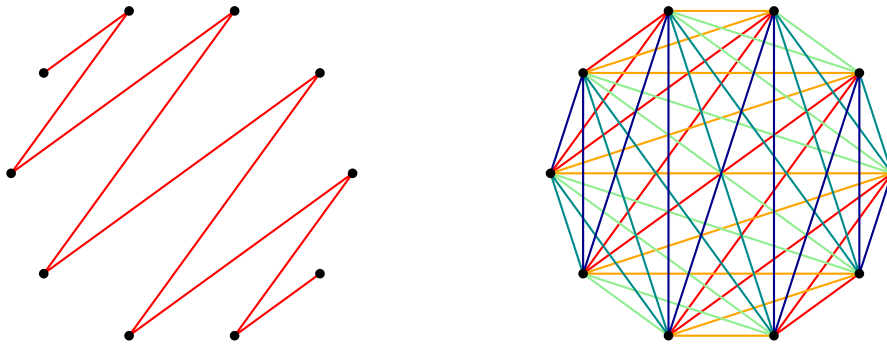


Figure 7: An n -point set in convex position with $n/2$ edge-disjoint crossing-free straight-line spanning paths.

This problem can be understood as a special case of geometric graph packing where edge-disjoint copies of a plane graph are to be packed in the complete geometric graphs defined by n points in the plane. Previous work considers the case of perfect matchings [3] and spanning trees [1, 2, 4]. Clearly, one may wonder about other interesting families, e.g., spanning trees with bounded degree.

Question 2: How many edge disjoint spanning trees with degree at most 3 does there exist on every n -point set in the plane?

Concerning Hamilton cycles, it is easy to see that some point sets only allow for one crossing-free cycle. For instance, in convex position, there is a unique crossing-free Hamilton cycle. The last question concerns the problem of better understanding what point sets allow for edge-disjoint Hamilton cycles.

Question 3: Which point sets allow for two edge-disjoint crossing-free Hamilton cycles?

References

- [1] AICHHOLZER, O., HACKL, T., KORMAN, M., PILZ, A., VAN RENNSSEN, A., ROELOFFZEN, M., ROTE, G., AND VOGTENHUBER, B. Packing plane spanning graphs with short edges in complete geometric graphs. *Computational Geometry 82* (2019), 1–15.
- [2] AICHHOLZER, O., OBENAU, J., ORTHABER, J., PAUL, R., SCHNIDER, P., STEINER, R., TAUBNER, T., AND VOGTENHUBER, B. Edge partitions of complete geometric graphs. In *38th International Symposium on Computational Geometry*, vol. 224 of *LIPICs. Leibniz Int. Proc. Inform.* Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022, pp. Art. No. 6, 16 pp.
- [3] BINIAZ, A., BOSE, P., MAHESHWARI, A., AND SMID, M. Packing plane perfect matchings into a point set. *Discrete Math. Theor. Comput. Sci.* 17, 2 (2015), 119–142.
- [4] BINIAZ, A., AND GARCÍA, A. Packing plane spanning trees into a point set. *Computational Geometry 90* (2020), 101653.
- [5] KINDERMANN, P., KRATOCHVÍL, J., LIOTTA, G., AND VALTR, P. Three edge-disjoint plane spanning paths in a point set. *Discrete Mathematics* 349, 3 (2026), 114780.

5: Growth rate of pattern-avoiding pseudoline arrangements

(suggested by Justin Dallant)

A (Euclidean) *pseudoline arrangement* is a collection of n simple curves in the Euclidean plane (called pseudolines) extending to infinity in both directions such that every two curves intersect at exactly one point, where they cross. It is *simple* if no three pseudolines intersect at a common point. Two pseudoline arrangements are *isomorphic* if one can be mapped to the other by an orientation-preserving homeomorphism of the plane. Given two pseudoline arrangements A and A' , A' is a subarrangement of A if there is a subset of the pseudolines in A isomorphic to A' .

It is known that the number B_n of simple non-isomorphic arrangements of n pseudolines is $2^{\Theta(n^2)}$ [2, Chapt. 9], although the leading hidden constant in the exponent is not known.

The Stanley-Wilf conjecture (proven by Marcus and Tardos [3]) states that for every finite permutation π , there is a constant c_π such that the number of permutations of length n which avoid π is at most c_π^n (as n grows to infinity, this a vanishingly small proportion of the $n!$ permutations of length n in total).

Goaoc and Welzl [1] raised the following question about an analogue of this result for simple pseudoline arrangements.

Question 1: Is it true that for any fixed simple pseudoline arrangement A , the proportion of simple pseudoline arrangements of size n that do not have A as a subarrangement tends to 0 as $n \rightarrow \infty$?

A stronger version of this question might be easier to answer in the negative:

Question 2: Is it true that for any fixed simple pseudoline arrangement A , the number $B_n(A)$ of simple pseudoline arrangements of size n that do not have A as a subarrangement is at most $2^{o(n^2)}$?

It is also natural to ask similar questions while focusing on *stretchable* pseudoline arrangements, which are those isomorphic to an arrangement of straight lines, either by counting only those or by restricting the question to the case where A is stretchable (the latter being potentially easier to answer in the affirmative).

References

- [1] GOAOC, X., AND WELZL, E. Convex hulls of random order types. *Journal of the ACM* 70, 1 (2023), 1–47.
- [2] KNUTH, D. E. *Axioms and hulls*. Lecture Notes in Computer Science. Springer, 1992.
- [3] MARCUS, A., AND TARDOS, G. Excluded permutation matrices and the Stanley–Wilf conjecture. *Journal of Combinatorial Theory, Series A* 107, 1 (2004), 153–160.

6: Factored connected hypersubgraph arrangements

(suggested by Leonie Mühlherr)

A *hyperplane arrangement* \mathcal{A} in a vector space V of dimension n is a finite set of $(n-1)$ -dimensional subspaces. Call $L(\mathcal{A})$ the *lattice of intersections* of the arrangement and for $X \in L(\mathcal{A})$ define the *localization* of \mathcal{A} at X as

$$\mathcal{A}_X = \{H \in \mathcal{A} \mid X \subseteq H\}.$$

In [1], Cuntz and Kühne defined the following:

Definition 1. Let $G = (N, E)$ be a simple graph with vertex set $N = [n] = \{1, 2, \dots, n\}$ and edge set $E \subseteq \binom{N}{2}$. The *connected subgraph arrangement* \mathcal{A}_G in $V = \mathbb{Q}^n$ is defined as

$$\mathcal{A}_G := \{H_I \mid \emptyset \subsetneq I \subseteq N, G[I] \text{ is connected}\},$$

where H_I is the hyperplane $H_I = \ker \sum_{i \in I} x_i$.

From this definition, interesting arrangements can be recovered as connected subgraph arrangements for special graphs such as the complete graph or the path graph. Cuntz and Kühne analyzed different properties of these arrangements, notably the property of factoredness.

Definition 2 ([3], Definition 2.66). Let $\pi = (\pi_1, \dots, \pi_s)$ be a partition of \mathcal{A} . The partition π is called *independent* if for any choice $H_i \in \pi_i$ for $1 \leq i \leq s$, the resulting s hyperplanes are linearly independent.

Let $X \in L(\mathcal{A})$. The *induced partition* π_X of \mathcal{A}_X is given by the nonempty blocks of the form $\pi_i \cap \mathcal{A}_X$. The partition π is a *factorization* of \mathcal{A} if

1. π is independent
2. For each $X \in L(\mathcal{A}) \setminus V$, the induced partition π_X admits a block which is a singleton.

We call \mathcal{A} *factored* if it admits a factorization.

For the connected subgraph arrangements, there is a characterization of factoredness:

Theorem 3 ([1], Theorem 8.10). Let G be a connected graph. The connected subgraph arrangement \mathcal{A}_G is factored if and only if G is a path graph P_n or a path-with-triangle graph $\Delta_{n,1}$.

See Figure 8 for an illustration of these graphs.

One can replace the graphs in Definition 1 by hypergraphs. Each of these arrangements is associated to a unique *Boolean building set* and each building set defines such an arrangement, thus, we call these arrangements *building set arrangements*.

Definition 4 ([2], Lemma 3.9). A family \mathcal{F} of subsets of $[n]$ is a *Boolean building set* if and only if \mathcal{F} contains all singletons $\{i\}, i \in [n]$, and the following condition holds: if $F, F' \in \mathcal{F}$ and $F \cap F' \neq \emptyset$, then $F \cup F' \in \mathcal{F}$.



Figure 8: Examples of graphs P_6 and $\Delta_{6,1}$ corresponding to factored connected subgraph arrangements.

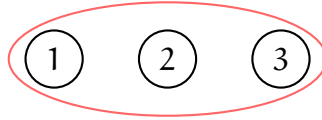


Figure 9: Example hypergraph whose connected hypergraph arrangement is not a connected subgraph arrangement

Every connected hypersubgraph arrangement \mathcal{A}_G is a subarrangement of the connected subgraph arrangement \mathcal{A}_{K_n} , where K_n is the complete graph with n vertices. Nevertheless, not every connected hypersubgraph arrangement is a connected subgraph arrangement, see Figure 9.

Thus, we can ask:

Question 1: Is there a characterization of factored building set arrangements by hypergraph or building set properties?

A follow-up question concerns itself with algebraic implications of the factoredness of an arrangement.

Proposition 5 ([4], Corollary 1.2). *Let \mathcal{A} be a factored arrangement and let $\pi = (\pi_1, \dots, \pi_s)$ be a factorization of \mathcal{A} . Then the Poincaré polynomial of \mathcal{A} factors as*

$$\text{Poin}(\mathcal{A}, t) = \prod_{i=1}^s (1 + |\pi_i| \cdot t).$$

Question 2: Is there a characterization of the numbers $|\pi_i|$ by hypergraph or building set properties?

References

- [1] CUNTZ, M., AND KÜHNE, L. On arrangements of hyperplanes from connected subgraphs, 2022. <https://arxiv.org/abs/2208.09251>.
- [2] FEICHTNER, E. M., AND STURMFELS, B. Matroid polytopes, nested sets and Bergman fans. *Port. Math. (N.S.)* 62, 4 (2005), 437–468.
- [3] ORLIK, P., AND TERAQ, H. *Arrangements of hyperplanes*, vol. 300 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1992.
- [4] TERAQ, H. Factorizations of the Orlik-Solomon algebras. *Advances in Mathematics* 91, 1 (1992), 45–53.

7: Hamiltonicity of 2-way transportation polytopes

(suggested by Hung Hoang and Arturo Merino)

A *2-way transportation polytope* $T(u, v)$ parameterised by two vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ is the set of all $m \times n$ non-negative matrices whose row sums equal to u and whose column sums equal to v . See [1] for a survey on this object, including the following facts and open question.

The polytope $T(u, v)$ is non-empty if and only if $\sum_{i=1}^m u_i = \sum_{i=1}^n v_i$. For every point $x = (x_{ij})_{i \in [m], j \in [n]}$ of the polytope, its *support graph* $B(x)$ is the following subgraph of the complete bipartite graph $K_{m,n}$: The vertices of $B(x)$ are partitioned into two sets $\{\sigma_1, \dots, \sigma_m\}$ and $\{\delta_1, \dots, \delta_n\}$. For $i \in [m], j \in [n]$, $\sigma_i \delta_j$ is an edge of $B(x)$ if and only if $x_{ij} > 0$. In other words, $B(x)$ encodes all the (strictly) positive entries of x .

A point x is a vertex of $T(u, v)$, if and only if $B(x)$ is a spanning tree. Two points x and y are adjacent in the graph of $T(u, v)$, if and only if $B(x) \cup B(y)$ contains exactly one cycle.

Question: Is a 2-way transportation polytope always Hamiltonian?

References

- [1] DE LOERA, J. A., AND KIM, E. D. Combinatorics and geometry of transportation polytopes: an update. In *Discrete geometry and algebraic combinatorics*, vol. 625 of *Contemp. Math.* Amer. Math. Soc., Providence, RI, 2014, pp. 37–76.

8: Kalai's conjectures and 2-level polytopes

(suggested by Martin Winter)

A polytope $P \subset \mathbb{R}^d$ is *centrally-symmetric* (or *origin-symmetric*) if $-P = P$. Despite much progress in understanding face numbers of polytopes, the following elementary question remains widely open:

Conjecture 1 (Kalai's 3^d conjecture, 1989, [2]). *Every centrally-symmetric d -polytope has at least 3^d non-empty faces.*

Here, the polytope itself is considered as a face, but the empty set is excluded. Observe that the conjectured lower bound is attained for the d -dimensional cube as well as the d -dimensional crosspolytope (the dual of the cube). There are further known polytopes that attain this bound, the *Hanner polytopes*, and it is conjectured that these are the only minimizers. The Hanner polytopes are constructed recursively: a Hanner polytope of dimension one is a line segment. Higher-dimensional Hanner polytopes are obtained as Cartesian products of Hanner polytopes of lower dimensions and of their duals. The number of Hanner polytopes in dimension $d = 1, 2, 3, \dots$ is $1, 1, 2, 4, 8, 18, 40, 94, 224, 548, \dots$

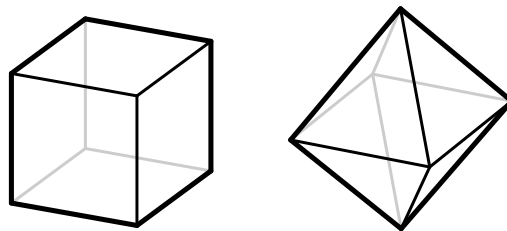


Figure 10: *The 3-cube and octahedron (the 3-dimensional crosspolytope). Both are Hanner polytopes and 2-level polytopes.*

A *flag* of a polytope is a maximal chain $\emptyset = F_{-1} \subset F_0 \subset \dots \subset F_{d-1} \subset F_d = P$ of faces of P , where F_i is i -dimensional. Kalai also posed the following related conjecture:

Conjecture 2 (Kalai's flag conjecture, 1989, [2]). *Every centrally-symmetric d -polytope has at least $d!2^d$ flags.*

Again, the Hanner polytopes are conjectured to be the only minimizers.

Some special cases have been resolved. For example, both conjectures are known to hold for simplicial polytopes and on the class of unconditional (aka coordinate-symmetric) polytopes [5, 1]. Both conjectures are true in dimension $d \leq 3$, and the 3^d conjecture is also verified in dimension $d = 4$ [4].

We propose another polytope class on which these conjectures should be studied. A centrally-symmetric polytope is *2-level* if each facet contains precisely half of the vertices of the polytope, or equivalently, if each facet together with its opposite facet contain all vertices of the polytope. All 2-level polytopes are 0/1-polytopes, closed under duality and include the Hanner polytopes. They are therefore a natural class to study both of Kalai's conjectures. In fact, they also contain all known near-misses to the 3^d conjecture, such as polytopes with $3^d + 16$ non-empty faces. The 2-level polytopes also encompass many

well-known polytope classes, such as order polytopes of posets, stable set polytopes of perfect graphs, Hansen polytopes, spanning tree polytopes of series-parallel graphs, Birkhoff polytopes and certain matroid base polytopes. In each case, Kalai's conjectures express non-trivial properties of the underlying combinatorial objects. Some interesting combinatorial properties of their face numbers are known, e.g., that they cannot simultaneously have both many vertices and facets: $f_0 \cdot f_{d-1} \leq d2^{d-1}$ [3]. Lastly, it is known that minimizers to face and flag numbers must be *linearly compact*, that is, the space of centrally-symmetric realizations of the same combinatorial type must be compact (modulo linear transformations). 2-level polytopes are in fact *linearly unique*, i.e., they have only a single realization up to linear transformation.

2-level polytopes have been enumerated up to dimension $d = 8$. Their number in dimension $d = 1, 2, 3, \dots$ is $1, 1, 2, 4, 13, 45, 238, 1790, \dots$. The first non-Hanner 2-level polytope appears in dimension five. We verified Kalai's conjectures computationally on all 2-level polytopes up to dimension $d = 8$.

Finally, we also propose the following strengthening of Kalai's 3^d conjecture:

Conjecture 3. *For each centrally-symmetric polytope P there exists a central hyperplane (i.e., a hyperplane that passes through the origin) that intersects the relative interior of at most a third of the faces of P .*

References

- [1] CHOR, A. Kalai's flag conjecture for locally anti-blocking polytopes. [https://arXiv:2507.22284](https://arxiv.org/abs/2507.22284) (2025).
- [2] KALAI, G. The number of faces of centrally-symmetric polytopes. *Graphs and Combinatorics* 5, 1 (1989), 389–391.
- [3] KUPAVSKII, A., AND WELTGE, S. Binary scalar products. *Journal of Combinatorial Theory, Series B* 156 (2022), 18–30.
- [4] SANYAL, R., WERNER, A., AND ZIEGLER, G. M. On Kalai's conjectures concerning centrally symmetric polytopes. *Discrete & Computational Geometry* 41, 2 (2009), 183–198.
- [5] SANYAL, R., AND WINTER, M. Kalai's 3^d conjecture for unconditional and locally anti-blocking polytopes. *Proceedings of the American Mathematical Society* 153, 01 (2025), 279–290.

9: Kotzig’s conjecture and paths in intersecting cycles

(suggested by Martin Winter)

A finite simple graph G is a P_k -graph for some integer $k \geq 1$ if for any two distinct vertices of the graph there exists *precisely one* path of length k between them. Kotzig proposed the following conjecture:

Conjecture 1 (Kotzig’s conjecture, 1974, [1]). *For $k \geq 3$ there are no P_k -graphs with two or more vertices.*

Evidently, P_1 -graphs are complete graphs. The P_2 -graphs are classified by the friendship theorem proven by Erdős, Rényi, and Sós [2]: they are precisely the windmill graphs, i.e., graphs that consist of finitely many triangles that are joined at a common vertex (see Figure 11). It is known that there are no P_k -graphs for $3 \leq k \leq 20$ [3].

Let vw be an edge of a P_k -graph for a $k \geq 2$. The edge vw together with the k -path between v and w form a $(k + 1)$ -cycle. One can show that every P_k -graph decomposes into an edge-disjoint union of such $(k + 1)$ -cycles. One can furthermore show that any two such $(k + 1)$ -cycles must intersect. It turns out that already in the subgraph consisting of two such intersecting cycles it is hard (perhaps impossible) to avoid a *double- k -path*, i.e., two k -paths (potentially intersecting many times) with the same end vertices.

This gives rise to another conjecture. A *$(k + 1)$ -cycle intersection* is a graph that can be written as the edge-disjoint union of two $(k + 1)$ -cycles.

Conjecture 2 (cycle intersection conjecture). *Every $(k + 1)$ -cycle intersection contains two k -paths with the same end vertices.*

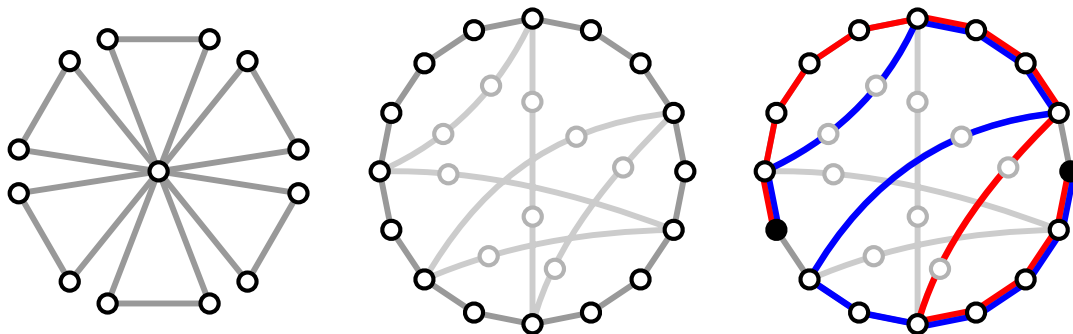


Figure 11: *Left: a P_2 -graph (i.e., a windmill graph). Middle: a 16-cycle intersection. The edges of the two cycles are shown in different colors. Right: a double 15-path in the cycle intersection between the two black vertices.*

One can show that in a potential counterexample to Conjecture 2 the cycles must have at least six, and at most $k - 2$ vertices in common.

Evidently, an affirmative answer to Conjecture 2 would also resolve Conjecture 1. The advantage of the cycle intersection conjecture is that it provides more structure to work with as compared to Kotzig’s original conjecture. It is known that a P_k -graph does contain at least three $(k + 1)$ -cycles. One could therefore also formulate and investigate a conjecture quite analogous to Conjecture 2, based on a $(k + 1)$ -cycle triple-intersection, which would provide even more structure to work with.

A central tool for studying cycle intersections are Hamiltonian decompositions. A cycle intersection can be seen as a subdivision of a 4-regular (multi-)graph H . The two cycles provide a *Hamiltonian decomposition* of H , i.e., two Hamilton cycles that together cover all edges of the graph. It is known that a 4-regular graph with a Hamiltonian decomposition has at least two further such decompositions [4]. Since Hamiltonian decompositions provide potential for finding a double- k -path, counterexamples to Conjecture 2 are most likely to be constructed through subdividing 4-regular graphs with few (but non-zero) Hamiltonian decompositions.

References

- [1] BONDY, J. Kotzig's conjecture on generalized friendship graphs—a survey. In *North-Holland Mathematics Studies*, vol. 115. Elsevier, 1985, pp. 351–366.
- [2] ERDŐS, P., RÉNYI, A., AND T SÓS, V. On a problem of graph theory. *Studia Scientiarum Mathematicarum Hungarica 1* (1966), 215–235.
- [3] KOSTOCHKA, A. The nonexistence of certain generalized friendship graphs. In *Combinatorics (Eger, 1987)*. North-Holland, Amsterdam, 1988, pp. 341–356.
- [4] THOMASON, A. G. Hamiltonian cycles and uniquely edge colourable graphs. In *Annals of Discrete Mathematics*, vol. 3. Elsevier, 1978, pp. 259–268.

10: Complexity of determining extension complexity

(suggested by Christoph Hertrich)

The *extension complexity* $xc(P)$ of a polytope P is the minimum number of inequalities required to represent P as the feasible region of a linear program, potentially allowing extra variables. An equivalent definition is the minimal number of facets of any polytope Q that projects onto P . Such a polytope Q is called an *extended formulation* of P . For example, in Figure 12, we see a three-dimensional polytope with six facets that projects to the octagon, proving that the extension complexity of the octagon is at most six. In some cases, for example for the spanning tree polytope, $xc(P)$ is exponentially smaller than the number of facets [2], showing that efficient LP formulations are possible for the associated optimization problem. In contrast, Rothvoß [3] proved that the matching polytope has exponential extension complexity even though an optimal matching can be found in polynomial time. This shows that the matching problem cannot be solved by a single polynomial-size linear program. Furthermore, Fiorini et al. [1] proved that the traveling salesperson polytope and other related polytopes associated with NP-hard optimization problems have exponential extension complexity. The authors of [1] and [3] received the 2023 Gödel Prize for these results.

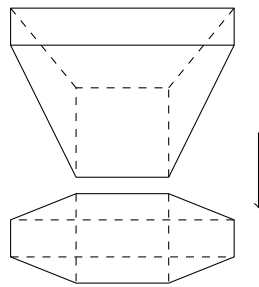


Figure 12: *Extended formulation of the octagon.*

While these strong results concern specific families of polytopes, the following general question remains open.

Question: What is the computational complexity of determining $xc(P)$ for a given polytope P ?

For this question, we assume that P is given in double-description form, i.e., we are given both the facets and the vertices of P .

A crucial ingredient for computing $xc(P)$ is Yannakakis’ theorem [6], stating that $xc(P)$ equals the nonnegative rank of the slack matrix of P . If P has f facets and v vertices, the *slack matrix* $S \in \mathbb{R}_{\geq 0}^{f \times v}$ of P is the matrix that records, for each vertex–facet pair, the slack of the vertex in the corresponding facet-defining inequality. The *nonnegative rank* of a matrix $S \in \mathbb{R}_{\geq 0}^{f \times v}$ is the smallest $r \in \mathbb{N}$ such that S can be written as UV for two nonnegative matrices $U \in \mathbb{R}_{\geq 0}^{f \times r}$ and $V \in \mathbb{R}_{\geq 0}^{r \times v}$.

Shitov [5] showed that computing the nonnegative rank of a matrix S is complete for the *existential theory of the reals* ($\exists\mathbb{R}$), a complexity class that is conjectured to be a strict superset of NP. However, Shitov’s reduction does not produce slack matrices

of polytopes. Hence, it remains open whether the same hardness holds when the input matrix is restricted to slack matrices of polytopes and, therefore, by Yannakakis' theorem, for determining $\text{xc}(P)$. In particular, not even NP-hardness is known for slack matrices of polytopes. On the other hand, all known algorithms to compute $\text{xc}(P)$ use general nonnegative rank computations. Therefore, the most promising way towards resolving the open question is to find an alternative reduction to the one by Shitov that produces slack matrices. A useful reference on $\exists\mathbb{R}$ -completeness is this compendium [4].

References

- [1] FIORINI, S., MASSAR, S., POKUTTA, S., TIWARY, H. R., AND DE WOLF, R. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM (JACM)* 62, 2 (2015), 1–23.
- [2] MARTIN, R. K. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters* 10, 3 (1991), 119–128.
- [3] ROTHVOSS, T. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)* 64, 6 (2017), 1–19.
- [4] SCHAEFER, M., CARDINAL, J., AND MILTZOW, T. The existential theory of the reals as a complexity class: A compendium. [https://arxiv:2407.18006](https://arxiv.org/abs/2407.18006) (2024).
- [5] SHITOV, Y. A universality theorem for nonnegative matrix factorizations. [https://arxiv:1606.09068](https://arxiv.org/abs/1606.09068) (2016).
- [6] YANNAKAKIS, M. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences* 43, 3 (1991), 441–466.

11: Firefighting is hard

(suggested by Torben Schürenberg)

Consider the following pursuit-evasion game introduced in [2]. The game is played on a simple undirected graph G , and at the start of the game, we imagine all nodes of the graph G to be on fire. A fixed number of firefighters are trying to extinguish the fire. Each time step consists of two phases. In the first phase, every firefighter can extinguish one freely chosen node (without any restrictions like moving along edges), but must then leave to gather more water. In the second phase, with the firefighters absent, the fire spreads: Each node with a burning neighbor catches fire again. In particular, this can include nodes that have been extinguished in the first phase. The smallest number of firefighters for which it is possible to extinguish the fire entirely is called the *firefighter number* $\text{ffn}(G)$. The decision problem **FIREFIGHTING** asks whether $\text{ffn}(G) \leq m$ for a given graph G and a given integer m .

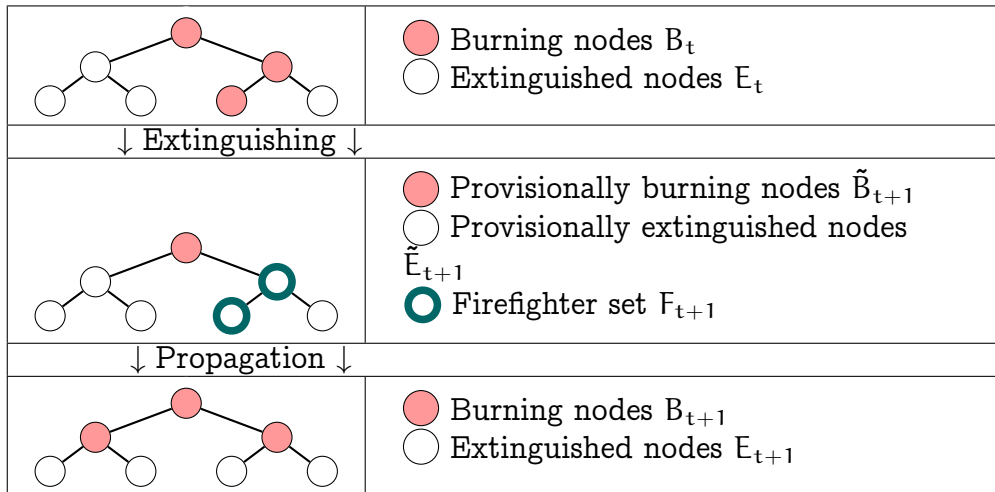


Figure 13: *Two firefighters try to extinguish a partially burning tree. Visualization of the two phases, namely extinguishing and propagation phase, of the fire.*

Recently, we have shown [1] that **FIREFIGHTING** is NP-hard. My hope is that we can use this fact to show NP-hardness when we change the setting to different fire spreading rules:

- A node catches fire if at least k of its neighbors are burning.
- A node catches fire if at last a p -fraction of its neighbors are burning.

Question: Is **FIREFIGHTING** still NP-hard for the modified fire spreading rules mentioned above?

References

[1] ALTHOETMAR, J., SCHADE, J., AND SCHÜRENBERG, T. Complexity of firefighting on graphs, 2025. <https://arxiv.org/abs/2505.11082>.

[2] BERNSHTEYN, A., AND LEE, E. Searching for an intruder on graphs and their subdivisions. *Electron. J. Combin.* 29, 3 (2022), Paper No. 3.9, 46 pp.

12: Quiver mutations

(suggested by Jean Cardinal)

A *quiver* is a directed multigraph without any loop or 2-cycle. Arcs of a quiver are usually referred to as *arrows*. It plays a fundamental role in various areas of mathematics, in particular in *cluster algebras*, which can encode Coxeter associahedra and triangulations of surfaces. We refer to [5] and [2] for nice introductions to cluster algebras¹.

A particular operation known as a *mutation* transforms a quiver into another one with the same vertex set, and is instrumental in the definition of exchanges in cluster algebras. A mutation occurs at some vertex k , and consists of the following three steps:

1. for each pair of arrows $i \rightarrow k$ and $k \rightarrow j$, add an arrow $i \rightarrow j$,
2. reverse all arrows incident to k ,
3. remove all arrows in a maximal set of pairwise disjoint directed 2-cycles.

An illustration is given in Figure 14.

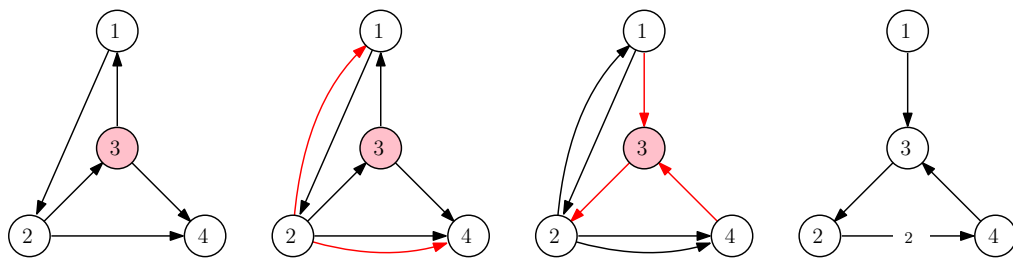


Figure 14: *The three steps of a quiver mutation. Here the mutation takes place at vertex 3 (from [1]). Edge labels indicate multiplicity.*

It can be seen that we retrieve flips in triangulations when the quiver is constructed from a triangulation, as illustrated in Figure 15.

There are many open questions about mutations of quivers. In particular, the following problem, posed by Fomin², is open:

Question: Consider the computational problem of deciding, given two quivers on the same vertex set, whether there exists a sequence of mutations that transforms one into the other. Is this problem decidable?

This question is probably difficult. It might be possible, though, to give hardness results for simple cases. A first step in this direction was made by Soukup [4], who proved that a variant of the problem for so-called *ice quivers*, in which edges between “frozen” vertices are not ignored, is NP-hard. Maybe one could start by considering those reductions and

¹See also the Cluster Algebra portal: <https://websites.umich.edu/~fomin/cluster.html>.

²See the talk “Quiver mutations”, at the Open Problems in Algebraic Combinatorics conference, May 2022, on YouTube: <https://www.youtube.com/watch?v=bIRnb00Fv1M>.

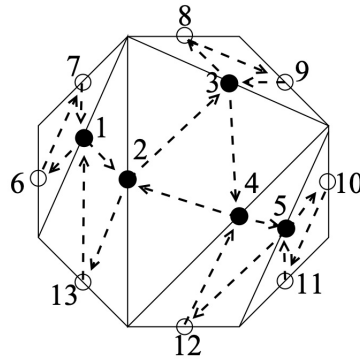


Figure 15: A quiver constructed from a triangulation (from [5]). Mutations can only occur on black vertices; the others are “frozen”.

prove NP-hardness, or maybe even PSPACE-hardness, of the original problem. It was also shown recently that mutation graphs can have long cycles [3].

I believe that there are many nice and fundamental open questions about these objects that are worth exploring and have not yet been tackled by the algorithms community.

References

- [1] AKITAYA, H., CARDINAL, J., FELSNER, S., KLEIST, L., AND LAUFF, R. Facet-Hamiltonicity. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2025), SIAM, Philadelphia, PA, pp. 5051–5064.
- [2] FOMIN, S. Total positivity and cluster algebras. In *Proceedings of the International Congress of Mathematicians. Volume II* (2010), Hindustan Book Agency, New Delhi, pp. 125–145.
- [3] FOMIN, S., AND NEVILLE, S. Long mutation cycles. *Selecta Math. (N.S.)* 31, 5 (2025), Paper No. 103, 44 pp.
- [4] SOUKUP, D. Complexity of ice quiver mutation equivalence. *Ann. Comb.* 29, 1 (2025), 91–100.
- [5] WILLIAMS, L. K. Cluster algebras: an introduction. *Bull. Amer. Math. Soc. (N.S.)* 51, 1 (2014), 1–26.

13: Reachability in STRIPS₁¹ planning problems

(suggested by Petr Gregor)

A *STRIPS₁¹ planning instance*³ (V, A, s_I, s_G) consists of the set V of n propositional variables, a set A of possible actions on the literals of V , and the initial and the goal states s_I and s_G . In STRIPS₁¹ each *action* is a pair of literals $a = (l_1, l_2)$ meaning that we can set l_2 to be true if l_1 is true. The literals l_1 and l_2 are called the *precondition* and the *effect* of a , respectively. A *state* is a (total) truth assignment on V , i.e. a consistent set of n literals. By symmetry, we may assume w.l.o.g. that s_I is the set of all negative literals. The task is to find a shortest *plan*, i.e. a sequence of actions that lead from s_I to s_G .

This can be conveniently represented by a directed graph on all $2n$ literals with a marked goal state, see the example below. Initially, we have n tokens on the negative literals (the bottom vertices). An arc (l_1, l_2) represents an action that allows us to move the token from \bar{l}_2 to l_2 if there is a token on l_1 .

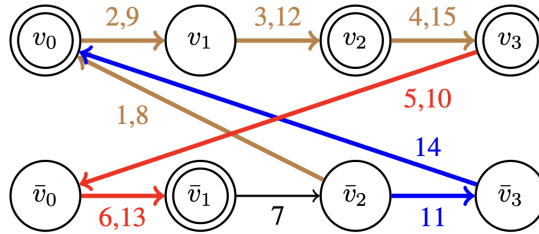


Figure 16: An example instance of STRIPS₁¹ for $n = 4$. The initial state is w.l.o.g. $s_I = (\bar{v}_0, \bar{v}_1, \bar{v}_2, \bar{v}_3)$, the goal state here is $s_G = (v_0, \bar{v}_1, v_2, v_3)$. The numbers along the arcs indicate steps in a shortest plan from s_I to s_G . Here, it requires 15 steps, which means that all states must be visited.

Question 1: Is it true that s_G is always reachable from s_I in $\text{poly}(n)$ steps or not reachable at all?

The study of STRIPS₁¹ was initiated by Bylander [1] who showed that determining whether some plan exist is NP-complete for the variant STRIPS₁¹⁺ where the effects of all actions are positive literals. This implies that determining whether some plan exists is NP-hard for STRIPS₁¹. A positive answer to Question 1 would imply that STRIPS₁¹ is also in NP.

Edelkamp and Jonsson [3] found instances for every $n \geq 6$ that require $\Omega(n^2)$ steps in the shortest plan and they conjectured that Question 1 is true. Recently, we have found an instance for $n = 5$ that requires 29 actions and we have verified by SAT solvers that 29 actions always suffice for $n = 5$ [2]. Similarly, for $n = 6$ we have an instance requiring 39 actions, which is also optimal. For $n = 7$ we have an instance requiring 50 actions but we do not know if it is optimal.

On the other hand, we have shown [2] that any shortest plan in a good instance omits at least 2^{n-2} states for $n \geq 16$, and at least 2^{n-3} states for $12 \leq n < 16$. An instance is *good* if for every variable v there are two actions whose effects are v and \bar{v} and whose preconditions are consistent.

³STRIPS stands for the Stanford Research Institute Planning System (Fikes and Nilsson, 1977).

References

- [1] BYLANDER, T. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69, 1-2 (1994), 165–204.
- [2] EDELKAMP, S., FINK, J., GREGOR, P., JONSSON, A., AND NEBEL, B. Intermediate results on the complexity of STRIPS₁, 2026. <https://arXiv:2602.08708>.
- [3] EDELKAMP, S., AND JONSSON, A. On the complexity of STRIPS₁. In *SIG Planning/Scheduling/Configuration Workshop (PUK) presented at German Conference on Artificial Intelligence (KI)* (2024). http://www.puk-workshop.de/rc_images/complexitystrips1_32_.pdf.

14: Circuits for the tropical permanent

(suggested by Christoph Hertrich)

Two closely related one-sentence formulations of this open problem:

- Can polynomial-size ReLU neural networks solve minimum-weight perfect matching in bipartite graphs (a.k.a. the assignment problem)?
- Are there polynomial-size $(\min, +, -)$ -circuits to compute the tropical permanent?

Given a matrix $X = (x_{ij}) \in \mathbb{R}^{n \times n}$, the *tropical permanent* is the expression

$$\text{tperm}(X) = \min_{\sigma \in S_n} \sum_{i=1}^n x_{i, \sigma(i)},$$

where S_n denotes the set of permutations of $\{1, \dots, n\}$. Equivalently, $\text{tperm}(X)$ is the weight of a minimum-weight perfect matching in a bipartite graph with edge weight matrix X . The tropical permanent is the tropicalized version of the classical permanent, that is, the polynomial

$$\text{perm}(X) = \sum_{\sigma \in S_n} \prod_{i=1}^n x_{i, \sigma(i)}.$$

Computing the classical permanent is $\#P$ -complete. In algebraic complexity theory, the permanent is the canonical VNP-complete family. It is widely believed but open that the permanent does not admit polynomial-size arithmetic circuits; a proof would separate VP from VNP.

In contrast, computing the tropical permanent is possible in polynomial time via the Hungarian algorithm. This motivates the question whether there also exist polynomial-size circuits to represent tperm . Such circuits would naturally operate in tropical algebra, using the operations \min , $+$, and $-$. However, polynomial-time computability does not imply the existence of polynomial-size circuits. The Hungarian algorithm crucially relies on adaptive, data-dependent branching based on comparisons of intermediate quantities. Circuits, in contrast, must compute the output by a fixed composition of operations without conditional control flow.

Question: (see [4, Section 6.5, Problem 3]) Are there polynomial-size $(\min, +, -)$ -circuits to compute the tropical permanent?

It is known that allowing subtraction is essential: there is an exponential lower bound for $(\min, +)$ -circuits [4].

Beyond circuit complexity, a crucial motivation for this problem comes from neural network expressivity. Every $(\min, +, -)$ -circuit can easily be simulated by a ReLU feedforward neural network, one of the most standard models in machine learning. While there exists a lot of work about the approximation capabilities of ReLU networks, it is largely unknown what classes of functions they can exactly represent with polynomial size. Recent upper bounds

were proven for maximum flows [3], minimum spanning trees [3], and regular matroids [2]. The existence of polynomial-size $(\min, +, -)$ -circuits to compute tperm would add min-weight bipartite perfect matching to that list.

Similarly to [3], a promising approach could be to find a specialized algorithm for min-weight perfect matching in bipartite graphs that only uses a very restricted set of operations: namely affine transformations and maxima or minima, but no data-dependent branching. Such algorithms are called *straight-line programs*. To this end, one could try to modify existing versions of the Hungarian method such that it avoids branching. It is to be expected that the dual variables will play an important role in this computation, as they depend in a continuous and piecewise linear manner on the weights, and can hence potentially be modeled directly in the circuit. As an inspiration, numerous algorithms for the assignment problem can be found in this book [1].

References

- [1] BURKARD, R., DELL'AMICO, M., AND MARTELLO, S. *Assignment problems: revised reprint*. SIAM, 2012.
- [2] HERTRICH, C., KOBER, S., AND LOHO, G. Arithmetic circuits and neural networks for regular matroids. In *International Conference on Integer Programming and Combinatorial Optimization* (2026), Springer.
- [3] HERTRICH, C., AND SERING, L. ReLU neural networks of polynomial size for exact maximum flow computation. *Mathematical Programming* 210, 1 (2025), 377–406.
- [4] JUKNA, S. *Tropical Circuit Complexity: Limits of Pure Dynamic Programming*, 1 ed. SpringerBriefs in Mathematics. Springer Cham, Cham, Switzerland, 2023.

15: The search for simple symmetric Venn diagrams

(suggested by Torsten Mütze)

An *n-Venn diagram* is a collection of n simple closed curves in the plane that intersect only finitely many times, thus creating exactly 2^n regions, one for every possible combination of being inside/outside of each of the curves. It is well-known and easy to show that n -Venn diagrams exist for every $n \geq 1$. For example, Figure 17 (a) below shows a 5-Venn diagram. This diagram has the additional feature of rotational symmetry. Formally, in a *symmetric* n -Venn diagram, each of the n curves is obtained from a single curve through rotation by $2\pi/n \cdot i$ for $i = 1, \dots, n$. An easy necessary condition for a symmetric n -Venn diagram to exist is that n is a prime number [3]. The question arises whether this condition is sufficient, i.e., is there a symmetric n -Venn diagram for every prime n ? This question has been open for a long time, until in a 2004 breakthrough, Griggs, Killian and Savage [2] answered it affirmatively.

Another additional feature of the 5-Venn diagram in Figure 17 is that it is *simple*, i.e., at most two curves cross in any point. The symmetric diagrams from [2] are not simple, unfortunately. Consequently, Ruskey, Savage and Weston [6] raised the following question.

Question 1: Is there a simple symmetric n -Venn diagram for every prime $n \geq 3$?

In tackling this question, it might be easier to restrict the search to monotone diagrams. A *k-region* in an n -Venn diagram is a region that is inside of exactly k of the curves, and outside the remaining $n - k$ curves. A Venn diagram is *monotone* if for every $0 < k < n$, every k -region is adjacent to both a $(k - 1)$ -region and a $(k + 1)$ -region. Geometrically, monotone diagrams are precisely those that can be drawn by convex curves [1].

Simple symmetric (and monotone) n -Venn diagrams are known only for $n = 3, 5, 7, 11, 13$ [5], but not for any larger primes. It is known that for any prime n , there is a symmetric (and monotone) n -Venn diagram in which at least 50% of all crossings are simple [4].

The dual graph of a monotone simple Venn diagram is a *rhombic strip*. This is a planar spanning subgraph of the cover graph of the Boolean lattice of subsets of $[n] = \{1, \dots, n\}$ in which every face is a diamond, i.e., a 4-face spanning three consecutive levels. Restricting to monotone diagrams, Question 1 can thus be restated as follows:

Question 2: Does the Boolean lattice of subsets of $[n]$ admit a rhombic strip that is invariant under cyclic shifts for every prime $n \geq 3$?

The paper [2] constructs symmetric (non-simple and monotone) Venn diagrams by first finding a *symmetric chain partition* in the *necklace poset*, which is obtained from the Boolean lattice by identifying sets that differ by cyclic shifts, and then connecting those chains to a planar spanning subgraph of the Boolean lattice that is invariant under cyclic shifts. However, not all faces of this subgraph are 4-faces, but longer, leading to non-simple crossings in the corresponding Venn diagram.

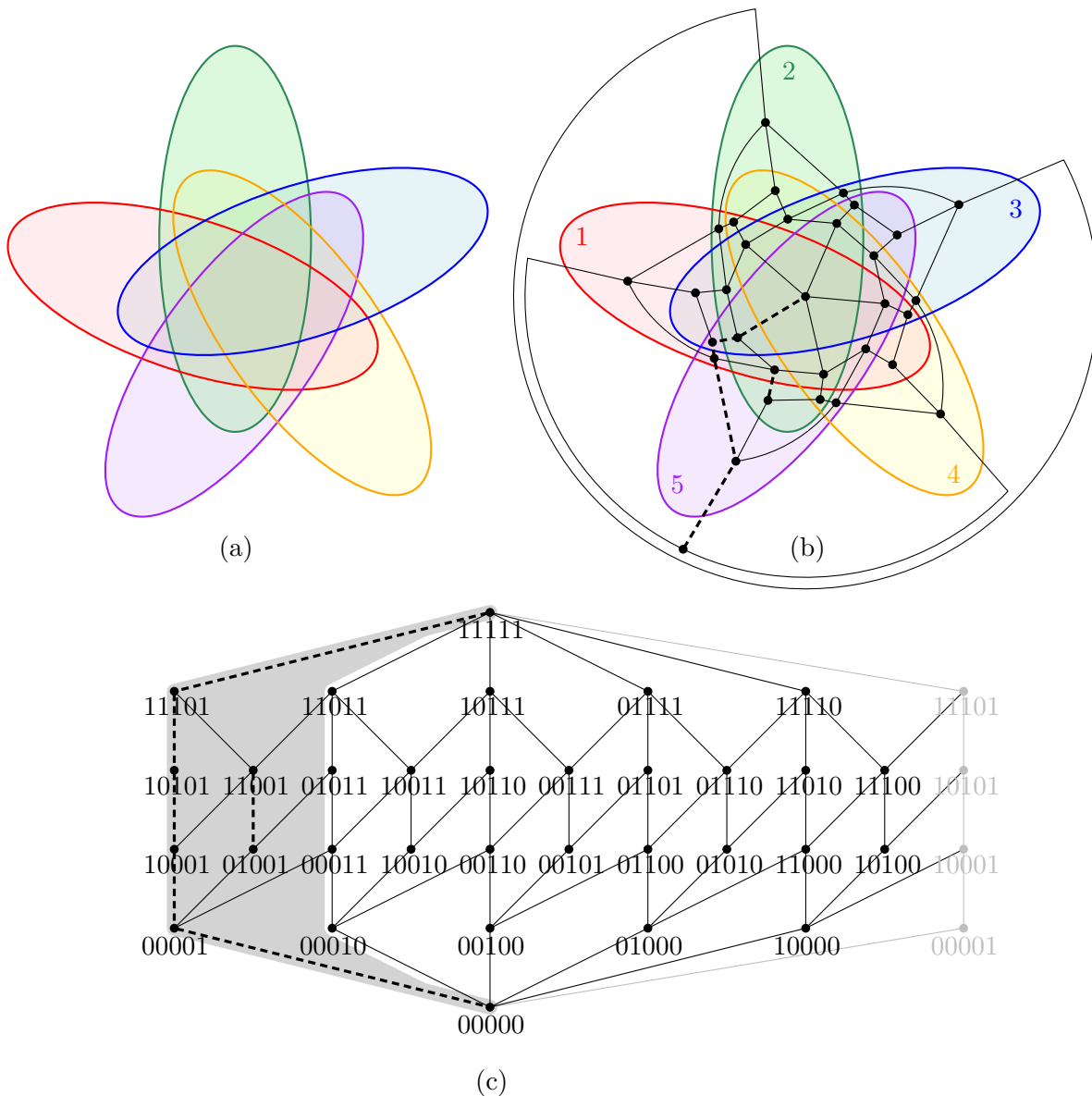


Figure 17: (a) A 5-Venn diagram that is both simple and monotone. (b) The dual graph of the Venn diagram. (c) The dual graph represented as a rhombic strip that is invariant under cyclic shifts, with one set of representatives highlighted by the gray polygon. A symmetric chain partition is emphasized by dashed lines. The diagram wraps around at the boundaries, and the left chain is duplicated on the right (in gray).

References

[1] BULTENA, B., GRÜNBAUM, B., AND RUSKEY, F. Convex drawings of intersecting families of simple closed curves. In *Proceedings of the 11th Canadian Conference on Computational Geometry, UBC, Vancouver, British Columbia, Canada, August 15-18, 1999* (1999).

[2] GRIGGS, J., KILLIAN, C. E., AND SAVAGE, C. D. Venn diagrams and symmetric chain decompositions in the Boolean lattice. *Electron. J. Combin.* 11, 1 (2004), Research Paper 2, 30 pp.

- [3] HENDERSON, D. W. Classroom Notes: Venn Diagrams for More than Four Classes. *Amer. Math. Monthly* 70, 4 (1963), 424–426.
- [4] KILLIAN, C. E., RUSKEY, F., SAVAGE, C. D., AND WESTON, M. Half-simple symmetric Venn diagrams. *Electron. J. Combin.* 11, 1 (2004), Research Paper 86, 22 pp.
- [5] MAMAKANI, K., AND RUSKEY, F. New roses: simple symmetric Venn diagrams with 11 and 13 curves. *Discrete Comput. Geom.* 52, 1 (2014), 71–87.
- [6] RUSKEY, F., SAVAGE, C. D., AND WAGON, S. The search for simple symmetric Venn diagrams. *Notices Amer. Math. Soc.* 53, 11 (2006), 1304–1312.

16: Rhombic strips for ideals of graphs, existence and non-existence

(suggested by Stefan Felsner and Robert Lauff)

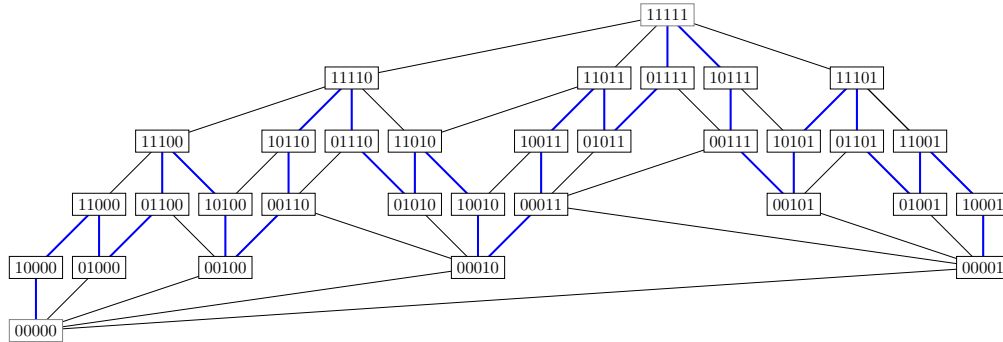


Figure 18: A rhombic strip in the boolean lattice of subsets of $\{1, \dots, 5\}$. In blue, the binary reflected Gray code.

Given a graded lattice L , a *rhombic strip* of L is a subdrawing R of the diagram of L with the following three properties:

- R is the diagram of a lattice and spans all elements of L ,
- R is plane (non-crossing),
- All bounded faces of R have degree 4 (resemble rhombic shape).

Drawings of rhombic strips are particularly nice and readable if all elements of rank k of L are placed on a horizontal line, for example on $y = k$. The unbounded face of R is adjacent to 2 maximal chains of L connecting the maximum and the minimum of L . They are called *left and right boundaries of R* , respectively.

A rhombic strip R is *Hamiltonian* if it contains a Hamilton path that, for any rank k , visits the elements of rank k in left-to-right order. A rhombic strip is *cyclic* if L restricted to the elements from the left and right boundary of R admits a rhombic strip using no edges from R . A more visual definition would be: A rhombic strip is *cyclic* if it has a plane embedding on the sphere such that all faces have degree 4. Hamiltonicity for cyclic rhombic strips is defined in the obvious way.

Admitting a rhombic strip, and especially a Hamiltonian rhombic strip, is a very strong property for a lattice and has some remarkable consequences. As an example, the boolean lattice of subsets \mathcal{B}_n (with subsets replaced by binary vectors) admits a Hamiltonian cyclic rhombic strip such that the binary reflected Gray code is an χ -monotone Hamilton cycle in it [2, 1]. The rank function of \mathcal{B}_n counts the number of 1s in the vector, which implies the following: When restricting the binary reflected Gray code to those binary vectors containing a fixed number k of 1s, we obtain a Gray code for the k -combinations of $[n]$ in which two adjacent combinations have a symmetric difference of size 2. In general, a rhombic strip provides simultaneous Gray codes for all ranks of the lattice.

Rhombic strips are related to facet-Hamiltonian cycles on polytopes and Venn diagrams. Details can be found in [1].

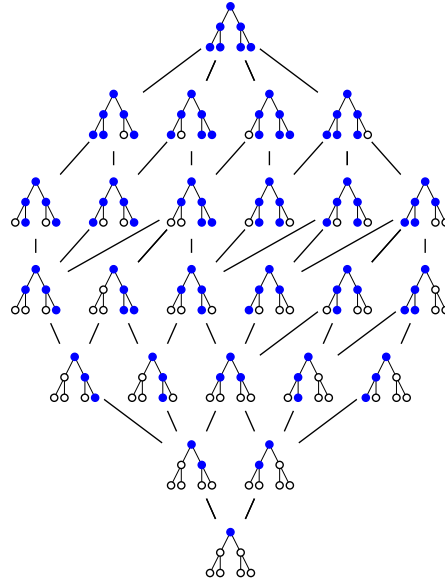


Figure 19: A rhombic strip in the lattice of subtrees of a complete binary tree.

We propose two problems concerning rhombic strips:

Problem 1. Consider the lattice $L(G, r)$ of connected subgraphs of a graph G which contain a fixed vertex $r \in V(G)$, with the inclusion order. The rank function counts the number of vertices in a subgraph. We can show that $L(G, r)$ admits a rhombic strip if G is a tree.

Question 1: Under which conditions on G and r can we guarantee that $L(G, r)$ admits a rhombic strip?

Problem 2. Some lattices do not admit rhombic strips. For example, the Young lattice, and the set of ideals of the poset C_3 (known as the cycle poset on 6 elements) do not admit rhombic strips. The product of chains $[3] \times [3] \times [3]$ admits a rhombic strips but no cyclic rhombic strip. We conjecture more generally, that no product of odd chains admits a cyclic rhombic strip.

Question 2: Can we identify necessary conditions for a lattice to admit a (cyclic) rhombic strip?

References

- [1] AKITAYA, H., CARDINAL, J., FELSNER, S., KLEIST, L., AND LAUFF, R. Facet-Hamiltonicity. *Proceedings of SODA 2025*, 5051–5064.
- [2] BEHROOZNI, N., BRENNER, S., MERINO, A., MÜTZE, T., RIECK, C., AND VERCIANI, F. Listing faces of polytopes. *Proceedings of SODA 2026*, 6212–6222.

17: Finite universal posets

(suggested by Sofia Brenner)

Let \mathcal{P} be a family of finite posets. A poset P is *universal* for \mathcal{P} if P contains all posets in \mathcal{P} as subposets. Universal objects have been studied for various classes of combinatorial objects (for instance, for graphs and directed graphs), and so a natural question is the following:

Question 1: What is the number of elements of the smallest poset P that is universal for all posets on n elements (for fixed n)?

An easy argument (see [1, Proposition 2.2]) shows that the Boolean lattice, that is, the poset $(2^{[n]}, \subseteq)$, is universal for the set of all n -element posets. This provides an upper bound of 2^n for the number of elements of the smallest universal poset. Concerning lower bounds, one can show that a universal poset must have at least $2^{(1+o(1))n/4}$ elements.

Clearly, one approach to close this gap is to focus on suitable subposets of the Boolean lattice. Using this idea, Bastide, Groenland, and Nenadov recently proved the following:

Theorem 1 ([1]). *There is a constant $C > 0$ such that for any $n \in \mathbb{N}$, there is a subposet P_n of $(2^{[n]}, \subseteq)$ on at most $2^{2n/3+C\sqrt{n}}$ elements that contains all n -element posets.*

For the proof of this theorem, they essentially partition the set of posets on n elements into two parts, depending on whether the poset contains an antichain of size at least α for some fixed $\alpha \in \mathbb{N}$. The posets containing such an antichain are embedded into a smaller Boolean lattice, the remaining ones are treated by a special construction.

It remains to close the gap between the two bounds. As a special case, one might want to use the same strategy and consider only subposets of the Boolean lattice:

Question 2: What is the minimum order of a subposet of the Boolean lattice $(2^{[n]}, \subseteq)$ that contains all posets on n elements?

In general, however, it would be very interesting to come up with different constructions for universal posets, i.e., to construct universal posets that are not subposets of the Boolean lattice.

References

- [1] BASTIDE, P., GROENLAND, C., AND NENADOV, R. Smaller universal posets. <https://arxiv.org/abs/2509.17820>, 2025.

18: Universal tori for permutations

(suggested by Aaron Williams with Liz Hartung and Joe Sawada)

De Bruijn sequences

A *de Bruijn sequence* of order n is a circular binary string of length 2^n that contains every n -bit binary string exactly once as a substring. For example, 00010111 suffices for $n = 3$ as its substrings of length 3 are 000, 001, 010, 101, 011, 111, 110, 100 where the last two wrap-around. Similarly, 001021122 is a k -ary de Bruijn sequence for $k = 3$ and $n = 2$.

These linear sequences are well understood. They are the labeled Eulerian circuits of the de Bruijn graph (i.e., one node per word of length $n - 1$ with edges of the form $x\alpha \rightarrow \alpha y$ with label y for all $x, y \in [k]$) which leads to their famous counting formulae [2, 23]. Specific sequences can also be generated in various ways. In particular, Knuth refers to the lexicographically least sequences by the name *grand-daddy* [16] and they were among the first combinatorial objects to be generated greedily [17]. Alternatively, they are created by reducing necklaces in lexicographic order to their aperiodic prefixes (e.g., 00010111 = $0 \cdot 001 \cdot 011 \cdot 1$ is obtained from the necklaces 000,001,011,111) [8, 9] and similar approaches work using co-lexicographic [5, 4] and cool-lex order [21, 22]. These constructions and others can be visualized as trees [20] and created efficiently with successor rules [10, 11]. For additional information see <http://debruijnsequence.org> [19]. As with the binary reflected Gray code, de Bruijn sequences are an example of Stigler's Law [3].

De Bruijn tori

One way to generalize de Bruijn sequences is in their shape. In particular, the two-dimensional analogues are known as *de Bruijn tori*. These objects have five parameters:

- R -by- C is the shape of the torus.
- r -by- c is the shape of the window.
- k is the number of symbols or values (e.g., $k = 2$ for binary).

At the *Workshop on Generalizations of de Bruijn cycles and Gray Codes* (BIRS 2004) Ron Graham asked if these tori always exist up to simple constraints. More specifically, Problem 480 [14] asks if a k -ary (R, C, r, c) -*de Bruijn torus* exists whenever $k > 1$, $RC = k^{rc}$, $R > r$ and $C > c$. Despite the various ways of understanding and generating de Bruijn sequences, this question is still wide open. However, some special cases have been constructed — including the square-square cases (i.e., $R = C$ and $r = c$) for binary [7] and k -ary [13] — often by layering, rotating, and concatenating linear de Bruijn sequences.

Universal cycles for permutations

Another way to generalize de Bruijn sequences is in the objects they contain. The term *universal cycle* is typically used in this context [1].

It can be easily seen that permutations written in one-line notation do not have universal cycles when $n \geq 3$. If such a cycle were to exist for $n = 3$, then it must contain 123 and so it could be rotated to have the form $123\square\square\square$. However, $23\square$ must be a permutation, so the first \square is 1. Then $31\square$ must be a permutation, so the second \square is 2, and similarly, the third is 3. In other words, there is no flexibility for the subsequent symbols. But 123123 is not suitable a universal cycle (e.g., 123 appears twice while 321 appears zero times).

Universal cycles of S_n can exist when using some type of indirect encoding or representation.

- An *order isomorphic* encoding represents a permutation of $[n]$ using the relative order of n distinct values, but the values are not limited to the set $[n]$. For example, when $n = 4$ the word 1453 encodes the permutation 1342. Johnson proved the best possible existence result: $n + 1$ values are sufficient [15]. Note that 123143 is an order isomorphic universal cycle for the permutations of $n = 3$.
- The *shorthand representation* encodes a permutation using its first $n - 1$ values. For example, when $n = 4$ the word 134 encodes the permutation 1342 as it omits the final redundant value 2. A shorthand universal cycle for the permutations of $[n]$ is equivalent to a universal cycle for the $(n - 1)$ -permutations of $[n]$. These cycles exist and can be generated efficiently [18, 12]. Note that 123132 is a shorthand universal cycle for $n = 3$ (or a universal cycle for the 2-permutations of $\{1, 2, 3\}$).
- A *relaxed shorthand representation* is a word of length n whose values are in $[n]$, however, only it is only required to have $n - 1$ distinct values [24]. If there is a repeated value, then its second copy is deleted and the unique missing value is placed at the end. For example, when $n = 4$ the word 1314 encodes 1342. A shorthand universal cycle is a relaxed shorthand universal cycle since the former ensures that the first $n - 1$ symbols are distinct. Note that 123414231432124313241342 is a relaxed shorthand universal (but not a shorthand universal cycle) for $n = 4$.

Shorthand universal cycles for permutations are arguably the most useful of these approaches. This is because each successive permutation of $[n]$ (including the omitted missing symbol) differs from the previous by a prefix rotation of length n or $n - 1$ [12]. In other words, shorthand universal cycles also provide a nice Gray code for the permutations.

If the permutations must be encoded directly but repetition is allowed, then we have a *superpermutation*. That is, a superpermutation is a word in which every permutation of $[n]$ appears at least once as a substring [6]. (The dual objective of including as many permutations as possible without repetition is trivial; $12 \cdots n 12 \cdots n-1$ is a suitable construction regardless of whether the sequence is linear or cyclic.)

Universal tori for permutations

In this problem we wish to combine the last two generalizations of de Bruijn sequences. In other words, we want to consider universal tori for permutations. As is the case with universal cycles, we need to relax the goal since direct encoding is too restrictive.⁴

In principle we can consider any of the approaches from the previous section. More specifically, we can use an alternate encoding like order isomorphism, shorthand representation,

⁴Exercise left for the reader.

or relaxed shorthand representation. Alternatively, we can allow repeated permutations (as in a superpermutation), or include as many permutations as possible without repetition.

Here we consider two natural generalizations of the shorthand representation. To make the distinction more tangible we use the square window with $r = 2$ and $c = 2$ as an example.

- *Ignored cell.* The bottom-right cell of the window is ignored. So 2-by-2 windows support permutations with $n = 4$. For example $\frac{1}{3} \frac{2}{\square}$ represents the permutation 1234.
- *Omitted cell.* The window is missing one value. So 2-by-2 windows support permutations with $n = 5$. For example $\frac{1}{3} \frac{2}{4}$ represents the permutation 12345.

To get an initial sense for these objects, Joe Sawada programmed a combinatorial search for 2-by-2 tori using both of these generalized shorthand representations. Constructions for the positive results appear in Figure 20.

- When $n = 4$ and the window size is 2-by-2, there do not exist universal tori of permutations using ignored cells for any R-by-C tori.
- When $n = 5$ and the window size is 2-by-2, there exist universal tori of permutations using omitted cells for R-by-C tori when $(R, C) \in \{(2, 60), (4, 30), (6, 20), (8, 15)\}$ but not when $(R, C) \in \{(3, 40), (5, 24)\}$ (with the final case $(R, C) = (10, 12)$ left open).

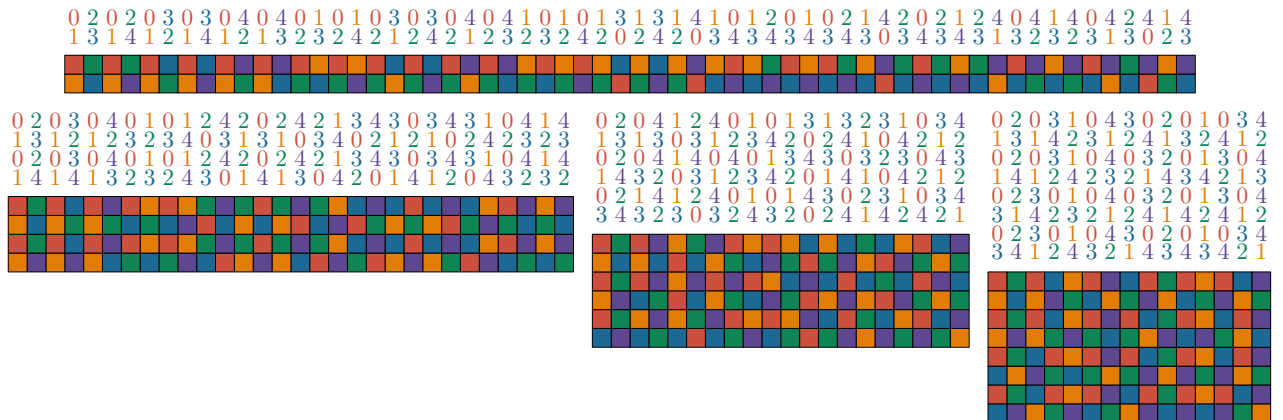


Figure 20: *Universal tori of permutations for $n = 5$ using $(2, 2)$ -windows and an omitted cell. Note that the window contains $n - 1 = 4$ values and the unique missing value is omitted and inferred. For example, the top-left windows $\frac{0}{2} \frac{1}{3}$ encode the permutation 01234 using 0-based values and row-major order. Similarly, in two of the constructions the window to its left $\frac{4}{1} \frac{0}{3}$ wraps-around horizontally and encodes 40132. The images above show that these tori exist for sizes $(R, C) \in \{(2, 60), (4, 30), (6, 20), (8, 15)\}$.*

Question 1: When do universal tori of permutations exist using ignored cells?

Question 2: When do universal tori of permutations exist using omitted cells?

We note that Question 2 is a special case of Graham’s Problem 480 [14].

References

- [1] CHUNG, F., DIACONIS, P., AND GRAHAM, R. Universal cycles for combinatorial structures. *Discrete Mathematics* 110, 1-3 (1992), 43–59.
- [2] DE BRUIJN, N. G. A combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 49, 7 (1946), 758–764.
- [3] DE BRUIJN, N. G. Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2^n zeros and ones that show each n -letter word exactly once.
- [4] DRAGON, P. B., HERNANDEZ, O. I., SAWADA, J., WILLIAMS, A., AND WONG, D. Constructing de Bruijn sequences with co-lexicographic order: The k -ary grandmama sequence. *European Journal of Combinatorics* 72 (2018), 1–11.
- [5] DRAGON, P. B., HERNANDEZ, O. I., AND WILLIAMS, A. The grandmama de Bruijn sequence for binary strings. In *LATIN 2016: Theoretical Informatics*. Springer, 2016, pp. 347–361.
- [6] ENGEN, M., AND VATTER, V. Containing all permutations. *The American Mathematical Monthly* 128, 1 (2020), 4–24.
- [7] FAN, C., FAN, S., MA, S., AND SIU, M. On de Bruijn arrays. *Ars Combinatoria* 19A (1985), 205–221.
- [8] FREDRICKSEN, H., AND KESSLER, I. Lexicographic compositions and deBruijn sequences. *Journal of Combinatorial Theory, Series A* 22, 1 (1977), 17–30.
- [9] FREDRICKSEN, H., AND MAIORANA, J. Necklaces of beads in k colors and k -ary de Bruijn sequences. *Discrete Mathematics* 23, 3 (1978), 207–210.
- [10] GABRIC, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Mathematics* 341, 11 (2018), 2977–2987.
- [11] GABRIC, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A successor rule framework for constructing k -ary de Bruijn sequences and universal cycles. *IEEE Transactions on Information Theory* 66, 1 (2019), 679–687.
- [12] HOLROYD, A. E., RUSKEY, F., AND WILLIAMS, A. Shorthand universal cycles for permutations. *Algorithmica* 64, 2 (2012), 215–245.
- [13] HURLBERT, G., AND ISAAK, G. On the de Bruijn torus problem. *Journal of Combinatorial Theory, Series A* 64, 1 (1993), 50–62.
- [14] JACKSON, B., STEVENS, B., AND HURLBERT, G. Research problems on Gray codes and universal cycles. *Discrete Mathematics* 309, 17 (2009), 5341–5348.
- [15] JOHNSON, J. R. Universal cycles for permutations. *Discrete Mathematics* 309, 17 (2009), 5264–5270.

- [16] KNUTH, D. The art of computer programming, volume 4, fascicle 2: generating all tuples and permutations, 2005.
- [17] MARTIN, M. H. A problem in arrangements. *Bull. Amer. Math. Soc.* 40, 12 (1934), 859–864.
- [18] RUSKEY, F., AND WILLIAMS, A. An explicit universal cycle for the $(n-1)$ -permutations of an n -set. *ACM Transactions on Algorithms (TALG)* 6, 3 (2010), 1–12.
- [19] SAWADA, J. De Bruijn sequence and universal cycle constructions. <http://debruijnsequence.org>, 2026.
- [20] SAWADA, J., SEARS, J., TRAUTRIM, A., AND WILLIAMS, A. Concatenation trees: A framework for efficient universal cycle and de Bruijn sequence constructions. *arXiv preprint arXiv:2308.12405* (2023).
- [21] STEVENS, B., AND WILLIAMS, A. The coolest order of binary strings. In *International Conference on Fun with Algorithms* (2012), Springer, pp. 322–333.
- [22] STEVENS, B., AND WILLIAMS, A. The coolest way to generate binary strings. *Theory of Computing Systems* 54, 4 (2014), 551–577.
- [23] VAN AARDENNE-EHRENFEST, T., AND DE BRUIJN, N. G. Circuits and trees in oriented linear graphs. In *Classic papers in combinatorics*. Springer, 1987, pp. 149–163.
- [24] WONG, D. A new universal cycle for permutations. *Graphs and Combinatorics* 33, 6 (2017), 1393–1399.

19: Flip the rightmost 0 and leftmost 1 (R0L1)

(suggested by Aaron Williams)

Consider the following greedy algorithm for combinations (i.e., fixed-weight binary words):

flip the rightmost 0 and the leftmost 1

such that a previously unseen combination is generated. In other words, the *R0L1 algorithm* repeatedly transposes a pair of bits where the first priority is selecting 0s from right-to-left, and the second priority is selecting 1s from left-to-right. For example, the order starting from 0^41^4 appears below with the transposed 0 and 1 shown on the left, and the prefix of the form 0^*1 on the right.

000 <u>0</u> 1111,	<u>0000</u> 1111,	
0001 <u>0</u> 111,	<u>0001</u> 0111,	
00011 <u>0</u> 11,	<u>0001</u> 1011,	
000111 <u>0</u> 1,	<u>0001</u> 1101,	
00 <u>0</u> 11110,	<u>0001</u> 1110,	
001 <u>0</u> 1110,	<u>0010</u> 1110,	
0010 <u>0</u> 111,	<u>0010</u> 0111,	
⋮	⋮	1

This greedy idea may seem somewhat arbitrary at first. However, Figure 21 shows that it creates an order that is similar to lexicographic order when started from 0^a1^b . This similarity is particularly exciting as Gray codes with lexicographic structures tend to have a variety of applications [1, 2, 3].

Question 1: Does the greedy R0L1 algorithm starting from $00 \dots 011 \dots 1$ work? Does the order have the following properties?

- (a) All words with prefix 0^i1 appear before those with prefix $0^{i-1}1$.
- (b) The last word is 1^b0^a .

Answering Question 1 may require understanding how the algorithm works when starting from other words. For example, if (a) and (b) are true, then the order changes the first bit exactly once from $011^{a-1}0^{b-1}$ to $101^{a-1}0^{b-1}$. The algorithm will never flip the first bit again as the words starting with 0 were created, so the second-half is identical to R0L1 starting at $01^{a-1}0^{b-1}$. But understanding how the algorithm works starting from 01^*0^* requires us to know how it works starting from other words. As a result, it might be easier to answer the following stronger question which is verified for $a = 3$ and $b = 3$ in Table 1.

Question 2: Does the R0L1 algorithm work starting from every combination?

If Question 1 or 2 can be answered in the affirmative, then there are several natural follow-up questions including the following.

1	000111	001011	001101	001110	010011	010101	010110	011001	011010	011100
2	001011	000111	000111	000111	000111	000111	000111	001011	001011	001101
3	001101	001101	001011	001011	001011	001011	001011	000111	000111	000111
4	001110	001110	001110	001101	001101	001101	001101	001101	001101	001011
5	010110	010110	010110	010101	001110	001110	001110	001110	001110	001110
6	010011	010011	010011	010011	010110	010110	011010	010110	010110	010110
7	010101	010101	010101	010110	010101	010011	010011	010011	010011	010011
8	011001	011001	011001	011010	011001	011001	010101	010101	010101	010101
9	011010	011010	011010	011001	011010	011010	011001	011100	011001	011001
10	011100	011100	011100	011100	011100	011100	011100	011010	011100	011010
11	101100	101100	101100	101100	101100	101100	101100	101010	101100	101010
12	100101	100101	100101	100101	100101	100101	100101	100011	100101	100011
13	100011	100011	100011	100011	100011	100011	100011	100101	100011	100101
14	100110	100110	100110	100110	100110	100110	100110	100110	100110	100110
15	101010	101010	101010	101010	101010	101010	101010	101100	101010	101100
16	101001	101001	101001	101001	101001	101001	101001	101001	101001	101001
17	110001	110001	110001	110001	110001	110001	110001	110001	110001	110001
18	110010	110010	110010	110010	110010	110010	110010	110010	110010	110010
19	110100	110100	110100	110100	110100	110100	110100	110100	110100	110100
20	111000	111000	111000	111000	111000	111000	111000	111000	111000	111000

1	100011	100101	100110	101001	101010	101100	110001	110010	110100	111000
2	000111	000111	000111	001011	001011	001101	010011	010011	010101	011001
3	001011	001011	001011	000111	000111	000111	000111	000111	000111	001011
4	001101	001101	001101	001101	001101	001011	001011	001011	001011	000111
5	001110	001110	001110	001110	001110	001110	001101	001101	001101	001101
6	010110	010110	010110	010110	010110	010110	001110	001110	001110	001110
7	010011	010011	010011	010011	010011	010011	010110	010110	010110	010110
8	010101	010101	010101	010101	010101	010101	010101	010101	010011	010011
9	011001	011001	011001	011001	011001	011001	011001	011001	011001	010101
10	011010	011010	011010	011010	011010	011010	011010	011010	011010	011100
11	011100	011100	011100	011100	011100	011100	011100	011100	011100	011010
12	101100	101100	101100	101100	101100	110100	101100	101100	101100	101010
13	100101	101001	100101	100101	100101	100101	100101	100101	100101	100011
14	100110	100011	100011	100011	100011	100011	100011	100011	100011	100101
15	101010	100110	101001	100110	100110	100110	100110	100110	100110	100110
16	101001	101010	101010	101010	110010	101010	101010	101010	101010	101100
17	110001	110010	110010	110010	110001	101001	101001	101001	101001	101001
18	110010	110001	110001	110001	110100	110001	111000	110001	110001	110001
19	110100	110100	110100	110100	111000	110010	110010	110100	110010	110010
20	111000	111000	111000	111000	101001	111000	110100	111000	111000	110100

Table 1: The greedy ROL1 algorithm works starting from any combination with $a = 3$ copies of 0 and $b = 3$ copies of 1.

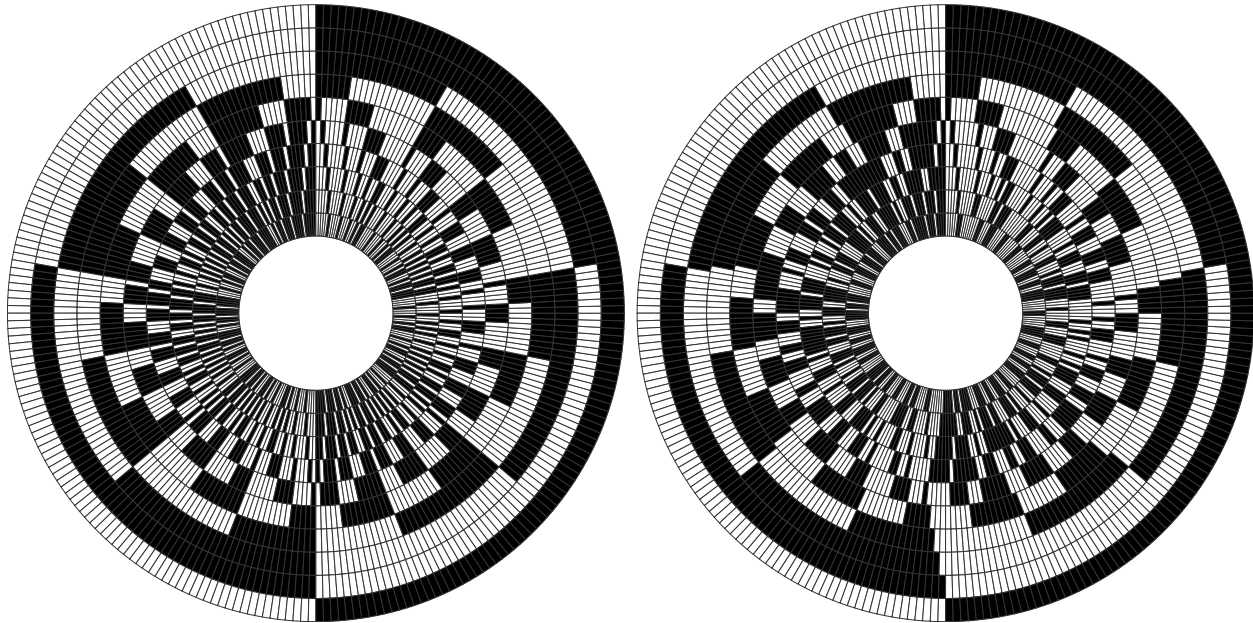


Figure 21: *Lexicographic order (left) and the greedy R0L1 order (right) starting at 0^51^5 .*

Question 3: Can the R0L1 order(s) be generated efficiently?

Question 4: Is there a generalization to all binary words or s-words?

References

- [1] MERINO, A., MUTZE, T., AND WILLIAMS, A. All your bases are belong to us: listing all bases of a matroid by greedy exchanges. In *11th International Conference on Fun with Algorithms (FUN 2022)* (2022), vol. 226, Schloss Dagstuhl—Leibniz-Zentrum für Informatik, p. 22.
- [2] RUSKEY, F., AND WILLIAMS, A. The coolest way to generate combinations. *Discrete Mathematics* 309, 17 (2009), 5305–5320.
- [3] WILLIAMS, A. The greedy Gray code algorithm. In *Workshop on Algorithms and Data Structures* (2013), Springer, pp. 525–536.

20: Generalizing the permutation language series to s -words

(suggested by Aaron Williams)

One of the biggest successes in combinatorial generation and Gray codes over the past decade is the *Permutation Languages* series. It began in Summer 2018 when Torsten Mütze suggested to visitors Liz Hartung and Aaron Williams that they try to find a Gray code for Baxter permutations $Av_n(2\overline{41}3, 3\overline{14}2)$ with an eye towards Gray codes for rectangulations. Two different but equivalent solutions were quickly discovered: one using local recursion and the other a greedy algorithm. The result was a broad generalization of plain changes for permutations to a jump Gray codes for zig-zag languages using Algorithm J. Since then it has had many applications including but not limited to (i) pattern avoidance in permutations [10], (ii) lattice congruences [11], (iii) rectangulations [13], (iv) elimination trees [6], (v) acyclic orientations [5], (vi) pattern avoidance in binary trees [9], (vii) supersolvable hyperplane arrangements [1, 2].

While the series continues to find new applications, there does seem to be certain limitations. For example, it is not clear how one would obtain results for k -Catalan objects (e.g., k -ary trees) as there is no known pattern avoidance result using permutations. However, there is such a result for pattern avoidance within s -words⁵ which contain s_i copies of i for a signature $s = (s_1, s_2, \dots, s_m)$ (or $s = (s_0, s_1, \dots, s_{m-1})$ for 0-based symbols).

At this past year's *Permutation Patterns Conference (2025)* a generalization of zig-zag languages and Algorithm J to s -words was proposed [4] (also on arXiv [3]). It was discovered during the SMALL REU at Williams College along with eight talented undergraduate students from Canada, Europe, and the United States. In some cases (e.g., rectangulations [13]) it is not clear if this generalization could lead to new results. In other cases (e.g., lattice congruences [11] and pattern avoidance in binary trees [9]) there is additional unpublished work. This problem highlights some of what is known and avenues for future work, but everyone familiar with the Permutation Languages series is invited to participate.

Permutations to Stirling s -Words of type 212

While our proposed generalization applies to all s -words, a natural first result involves Stirling s -words. Recall that a *Stirling word* is a permutation of $1, 1, 2, 2, \dots, n, n$ with the property that only smaller values can appear between the two copies of value i . More generally, a *Stirling s -word* is a words with signature s that avoids the pattern 121.

In this section we instead consider these words with smaller/larger reversed. That is, a *Stirling s -word of type 212* is an s -word that avoids the pattern 212.

Stirling s -words of type 212 are particularly nice in the context of generalizing zig-zag languages and Algorithm J. This is because the copies of a value j cannot be split up when they move over a smaller value $i < j$, since otherwise a copy of the forbidden 212 pattern will be created. Another way of stating this advantage is that Stirling s -words of type 212 can be represented by an inversion word that contains only one entry per unique value rather than one entry per symbol in the word.

⁵These are also called multiset permutations, words with fixed-content or Parikh vector, or s -permutations. With regard to the last option, s -word leads to more appropriate variable names when programming.

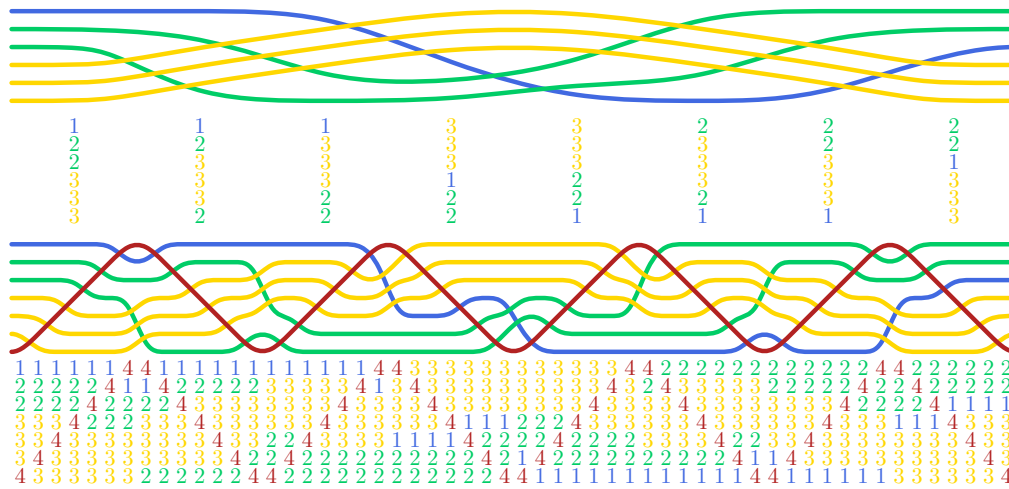


Figure 22: Plain changes for Stirling s -words of type 212 with signatures $s = (1, 2, 3)$ (top) and $s = (1, 2, 3, 1)$ (bottom) [4].

Given this discussion it is fairly easy to see how to generalize plain changes from permutations to Stirling s -words of type 212: greedily swap the largest value. In this context a *swap* moves every copy of a value j over one copy of a smaller value $i < j$. The resulting order recreates the familiar zig-zag pattern of local recursion seen in Figure 22.

Hamilton paths in the generalization of permutohedra involving trees [7] are also created (but differences in conventions lead to a change from decreasing trees to increasing trees).

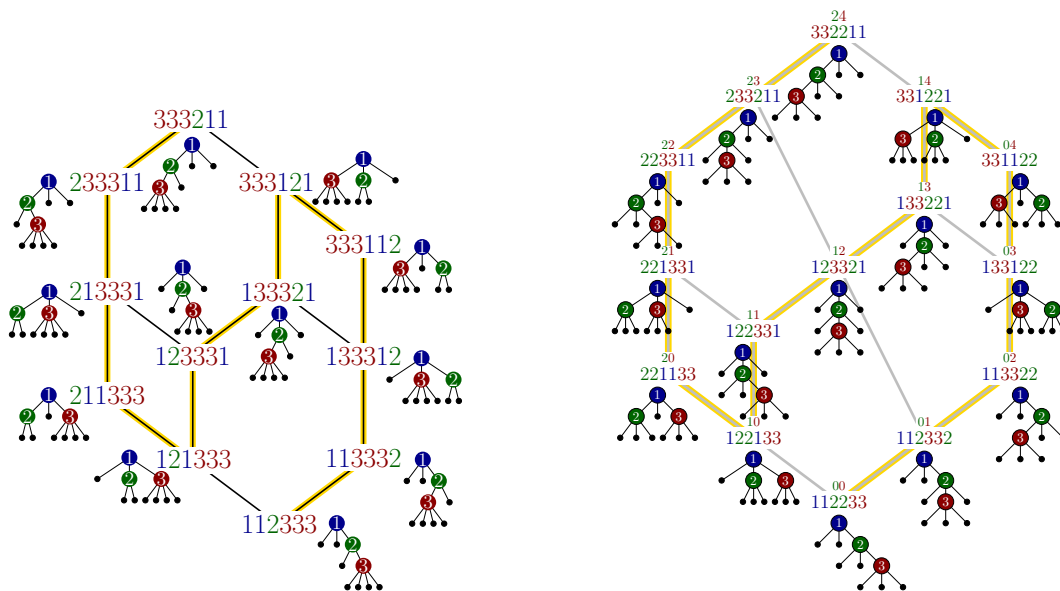


Figure 23: Hamilton paths in s -permutohedra for $s = (2, 1, 3)$ and $s = (2, 2, 2)$. The second example also shows the corresponding ± 1 Gray code for the inversion words.

Note: In the rest of this problem we primarily use the standard notion of Stirling s -words. In other words, Stirling s -words of type 212 are more of a special case.

Algorithm B for bumps

When working with other languages we want to be able to move some copies of a value j (rather than all of them) over more than one smaller value $i < j$ (rather than one of them). This leads to our notion of a bump, which is a generalization of a jump [10]. To define the operation we first define the rank of each symbol in a word and a one-sided run.

Let $w = w_1 w_2 \cdots w_n$ be an s -word with $s = (s_1, s_2, \dots, s_m)$. Each $v \in [m]$ is a *value* and each w_i is a *digit*. Each digit has a unique index (i.e., w_i has index i) but also a unique *rank* based on its value and position relative to the other digits with the same value. More specifically, the *rank* of each digit is between 1 and n and it is obtained by increasing value then left-to-right index. For example, the ranks of 123332 are 124563 .

The *right-maximal run* (or *right-run*) at index i is $w_i w_{i+1} \cdots w_j$ with $w_i = \cdots = w_j$ and $j = n$ or $w_j \neq w_{j+1}$. Note that these runs can also be specified by the rank of the digit rather than its index. A *right-bump* moves a right-run to the right past some smaller digits. Left notions are defined in the same way, and a *bump* is a left-bump or right-bump.

A bump has various parameters including value v , rank r , index i , width w , and distance d . For example, $11211\overline{33}311$ creates $1\overline{33}1211311$ by a left-bump with value $v = 3$ (i.e., this is the larger value that moves) and width $w = 2$ (i.e., two copies $\overline{33}$ move left) and distance $d = 4$ (i.e., four digits are passed $\overline{1211}$) starting at index $i = 7$ (i.e., $w_7 = 3$) or rank $r = 8$.

Let L be an *s-word language* (i.e., a subset of s -words for some s). Suppose a bump at rank r changes $w \in L$ to w' . The bump is *minimal* for its rank and direction if its distance is positive and minimized with respect to $w' \in L$. Similarly define a *minimal left-bump*, and a bump to be *minimal* if it is a minimal right-bump or a minimal left-bump. Note that minimal bumps on $w \in L$ are uniquely determined by rank and direction. Our generalization of Algorithm J replaces values with ranks, and prefers rightwards bumps.

Algorithm B attempts to generate an s -word language L with $s = (s_1, s_2, \dots, s_m)$ as follows.

- B1** *Initialize*. Visit the given initial word w (or by default visit $w = 1^{s_1} 2^{s_2} \cdots m^{s_m}$).
- B2** *Greedy*. Let w be the most recent word visited. Visit a new word by applying to w a minimal bump prioritized by largest rank then rightward over leftward. Halt if no such bump exists. Otherwise, repeat **B2**.

Zig-zag languages for s -words

Our generalization of zig-zag languages can be defined in terms of jumps. Say that a jump is *maximum* if it uses the longest distance from an index in a direction. In other words, it moves a digit in one direction until it reaches a value that is greater than or equal to it. A *zig-zag language* is then an s -word language that is closed under maximum jumps. Note that closure under maximum jumps also implies closure under maximum bumps.

Zig-zag languages include $Av_s(\alpha)$ (i.e., the avoidance of pattern α) when α 's largest values are *internal* (i.e., not first or last) and *isolated* (i.e., not consecutive). Zig-zag languages are also closed under intersection (and union). As a result, the *peakless s-words* $Av_s(132, 231, 121)$ are a zig-zag language. Indeed, these words are a subset of every zig-zag language as any peakless s -word is created from any s -word by a series of maximum jumps.

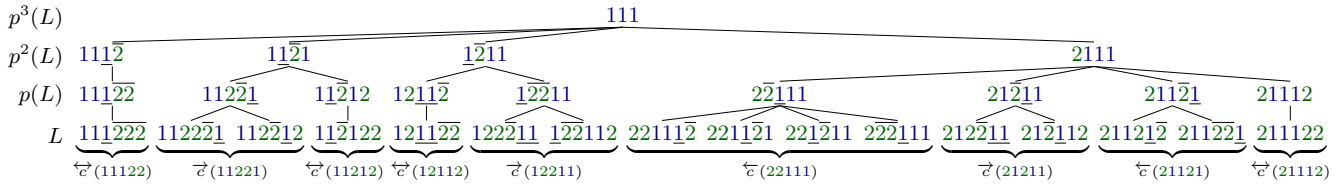


Figure 24: Algorithm B’s Gray code for $L = Av_{3,3}(12121)$ and its ancestor languages. Note that successive children jump the rightmost largest value either to the left or right (or nowhere if there is no room) as these are Algorithm B’s highest-priority operations. Then a bump changes the last child of one parent to the first child of the next parent, and the rightmost largest value may join these bumps. For example, the jump $22111 = 21211$ in the parent language widens to become the bump $222111 = 212211$ in L .

We can prove that Algorithm B generates any zig-zag language using an inductive argument that is illustrated in Figure 24. But Algorithm B also generates other languages including $Av_s(212)$ (i.e., Stirling s -words of type 212). To prove this we can more generally define *zig/zag languages* (as opposed to zig-zag languages) by allowing values to either be stationary or closed under maximum-jumps, which is an approach taken in some of the Permutation Languages papers.

Catalan objects to k -Catalan objects

It is well-known that the Catalan numbers count $Av_n(\alpha)$ for all patterns of length three (i.e., any single $\alpha \in \{123, 132, 213, 231, 312, 321\}$). More generally, the *s -Catalan numbers* generalize the Catalan numbers (using signature $s = (1, 1, \dots, 1)$) and k -Catalan numbers (using signature $s = (k, \dots, k)$) and they count $Av_s(\alpha, \beta)$ for certain pairs of patterns (α, β) including the following

- $(231, 121)$ [12]
- $(231, 221)$ [8]
- $(123, 122)$ [14]

with the general result presented in the latter. The first pair is particularly helpful as $Av_s(231, 121)$ (or equivalently, $Av_s(132, 121)$) form a zig-zag language. This leads to new Gray codes including those shown in Figure 25.

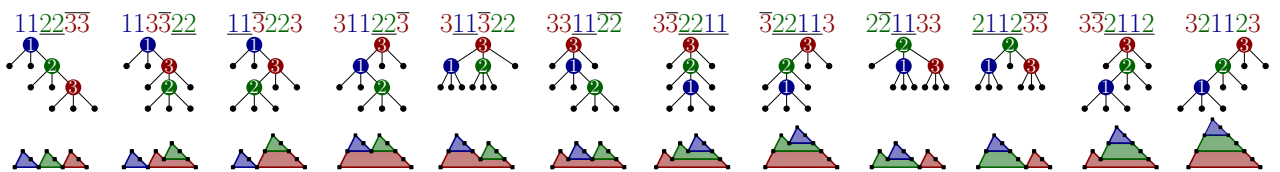


Figure 25: Pattern-avoiding s -words $Av_s(132, 121)$ for $s = (2, 2, 2)$ with Gray codes for 3-Catalan objects. The ternary trees differ by moves preserving inorder (i.e., visit self before last child).

References

- [1] BRENNER, S., CARDINAL, J., MCCONVILLE, T., MERINO, A., AND MÜTZE, T. Combinatorial generation via permutation languages. VII. Supersolvable hyperplane arrangements. *arXiv preprint arXiv:2507.14327* (2025).
- [2] BRENNER, S., CARDINAL, J., MCCONVILLE, T., MERINO, A., AND MÜTZE, T. Traversing regions of supersolvable hyperplane arrangements and their lattice quotients. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2026), SIAM, pp. 4953–4968.
- [3] BUICK, S., GOERTZ, M., LASTMANN, A., PAL, K., QIAN, H., TACHENY, S., WILLIAMS, A., WILLIAMS, L., AND ZHAI, Y. Exhaustive generation of pattern-avoiding s-words. *arXiv preprint arXiv:2508.16039* (2025).
- [4] BUICK, S., MADELEINE, G., LASTMANN, A., PAL, K., QIAN, H., TACHENY, S., WILLIAMS, A., WILLIAMS, L., AND ZHAI, Y. Exhaustive generation of pattern-avoiding s-words. In *Proceedings of the 23rd International Conference on Permutation Patterns* (2025).
- [5] CARDINAL, J., HOANG, H. P., MERINO, A., MIČKA, O., AND MÜTZE, T. Combinatorial generation via permutation languages. V. Acyclic orientations. *SIAM Journal on Discrete Mathematics* 37, 3 (2023), 1509–1547.
- [6] CARDINAL, J., MERINO, A., AND MÜTZE, T. Combinatorial generation via permutation languages. IV. Elimination trees. *ACM Transactions on Algorithms* 21, 1 (2024), 1–41.
- [7] CEBALLOS, C., AND PONS, V. The s-weak order and s-permutahedra I: Combinatorics and lattice structure. *SIAM Journal on Discrete Mathematics* 38, 4 (2024), 2855–2895.
- [8] DEFANT, C., AND KRAVITZ, N. Stack-sorting for words. *Australasian Journal of Combinatorics* 77, 1 (2020), 51–68.
- [9] GREGOR, P., MÜTZE, T., AND NAMRATA. Combinatorial generation via permutation languages. VI. Binary trees. *European Journal of Combinatorics* 122 (2024), 104020.
- [10] HARTUNG, E., HOANG, H. P., MÜTZE, T., AND WILLIAMS, A. Combinatorial generation via permutation languages. I. Fundamentals. *Transactions of the American Mathematical Society* 375 (2022), 2255–2291.
- [11] HOANG, H. P., AND MÜTZE, T. Combinatorial generation via permutation languages. II. Lattice congruences. *Israel Journal of Mathematics* 244, 1 (2021), 359–417.
- [12] KUBA, M., AND PANHOLZER, A. Enumeration formulae for pattern restricted Stirling permutations. *Discrete Mathematics* 312, 21 (2012), 3179–3194.
- [13] MERINO, A., AND MÜTZE, T. Combinatorial generation via permutation languages. III. Rectangulations. *Discrete & Computational Geometry* 70, 1 (2023), 51–122.
- [14] WILLIAMS, A. Pattern avoidance for k-Catalan sequences. In *Proceedings of the 21st International Conference on Permutation Patterns* (2023).

21: Anti-Gray codes

(suggested by Hung Hoang and Arturo Merino)

Given a graph $G = (V, E)$, we can construct its *k*th power by adding an edge between all pairs of distinct vertices that are at distance at most k . That is, $G^k = (V, E^k)$ where

$$E^k = \{uv \in \binom{V}{2} \mid d_G(u, v) \leq k\}.$$

Such graphs are intimately related to k -Gray codes, as a k -Gray code is simply a Hamiltonian path or cycle in G^k . Sufficient conditions for the existence of Hamilton cycles in G^k are provided by the classical results of Sekanina [2] and Fleischner [1].

Theorem 1. *If G is connected, then G^3 contains a Hamilton cycle. Furthermore, if G is 2-connected, then G^2 contains a Hamilton cycle.*

Thus, for a connected graph G , we can define its *Gray code number*, denoted $GC(G)$, as the smallest k such that G^k is Hamiltonian. By the aforementioned results, we know that $GC(G) \in \{1, 2, 3\}$ for any connected graph G .

We propose the study of the following variant: instead of looking at powers, we examine *copowers*. That is, we connect vertices that are at distance *at least* k . Formally, the *k*th copower of G is given by $G^{\geq k} = (V, E^{\geq k})$, where

$$E^{\geq k} = \left\{ uv \in \binom{V}{2} \mid d_G(u, v) \geq k \right\}.$$

Analogously, we define the *anti-Gray code number*, denoted $aGC(G)$, as the *largest* k such that $G^{\geq k}$ is Hamiltonian.

Currently, we know the following:

Proposition 2. *Let Q_n be the n -cube, and Π_n the n -permutahedron. Then, $aGC(Q_n) = n - 1$ and $aGC(\Pi_n) = \binom{n}{2} - 1$.*

Proof sketch. For both it is easy to see that $Q_n^{\geq n}$ and $\Pi_n^{\geq \binom{n}{2}}$ are matchings. In particular, they are disconnected, yielding the upper bound.

For the lower bound there is a construction from the BRGC/SJT. (Turn upside down to spoil yourself.)

For the Cube: If n is even: Take the standard BRGC sequence s_1, \dots, s_{2^n} . The cycle is $s_1, \bar{s}_2, s_3, \bar{s}_4, \dots$ (alternating complements). If n is odd: Let B_1 be the first half of the BRGC. Interleave B_1 with B_1 (the element-wise complement of the first half).

For the Permutahedron: If $\binom{n}{2}$ is even: Take the SJT sequence $\pi_1, \dots, \pi_{\binom{n}{2}}$. The cycle is $\pi_1, \pi_2^R, \pi_3, \pi_4^R, \dots$ (alternating reversals). If $\binom{n}{2}$ is odd: Take the first half of the SJT, S_1 , and interleave it with S_1^R (the element-wise reversal of the first half).

This finishes the proof. □

A first natural question is for which other graphs we can compute the anti-Gray code number.

Question 1: For which other graph families can we compute the anti-Gray code number? In particular, if \mathcal{A}_n is the n -associahedron, what can be said about $\text{aGC}(\mathcal{A}_n)$.

It would be particularly interesting if we could expand the zigzag framework to anti-Gray codes.

Question 2: Is there an anti-Gray code version of the zigzag framework?

Finally, we ask if there are meta-theorems for anti-Gray codes similar to those for graph powers.

Question 3: Are there Sekanina/Fleischner type results for anti-Gray codes?

References

- [1] FLEISCHNER, H. The square of every two-connected graph is Hamiltonian. *J. Combinatorial Theory Ser. B* 16 (1974), 29–34.
- [2] SEKANINA, M. On an ordering of the set of vertices of a connected graph. *Spisy Přírod. Fak. Univ. Brno 1960* (1960), 137–141.

22: Hamiltonicity of generalized Heawood graphs

(suggested by Torsten Mütze)

Consider the infinite tiling of the plane with regular hexagons, and label each hexagon by a triple $\mathbf{a} = (a_1, a_2, a_3)$ of non-negative integers according to the three directions as shown in Figure 26.

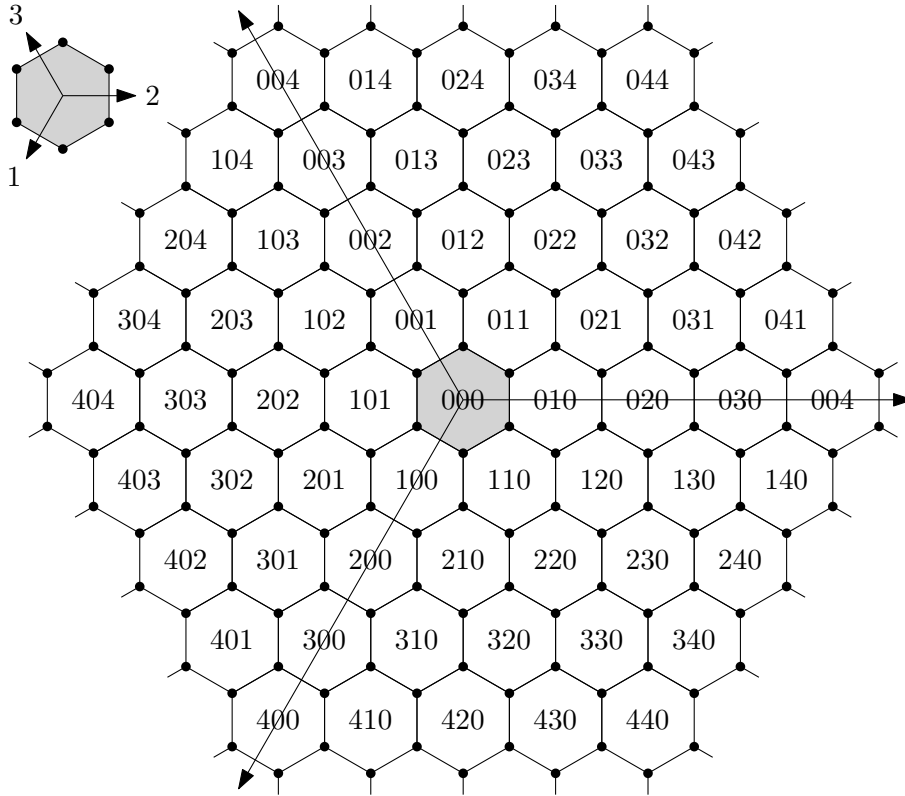


Figure 26: Infinite hexagonal tiling with labeling of hexagons.

For a triple $\mathbf{b} = (b_1, b_2, b_3) \in \mathbb{N}_{>0}^3$, we define the \mathbf{b} -Heawood graph $H_{\mathbf{b}}$ as the graph obtained from the infinite hexagonal grid as follows: We take the subset of hexagons whose labels (a_1, a_2, a_3) satisfy $a_i \leq b_i$ for $i = 1, 2, 3$, and we identify boundary vertices of the resulting hexagonal patch if they lie on pairs of opposite sides (like on a torus); see Figure 27. The classical Heawood graph is the special case $H_{(1,1,1)}$. Note that $H_{\mathbf{b}}$ is 3-regular and vertex-transitive.

Ceballos and Doolittle [1] raised the following problem.

Question 1: Does $H_{\mathbf{b}}$ have a Hamilton cycle, for any $\mathbf{b} \in \mathbb{N}_{>0}^3$?

Actually, they conjecture that there is always a Hamilton cycle that alternately uses edges along only two of the three possible directions; see Figure 28.

Question 2: Does $H_{\mathbf{b}}$ have a Hamilton cycle that alternately uses only two of the three possible edge directions, for any $\mathbf{b} \in \mathbb{N}_{>0}^3$?

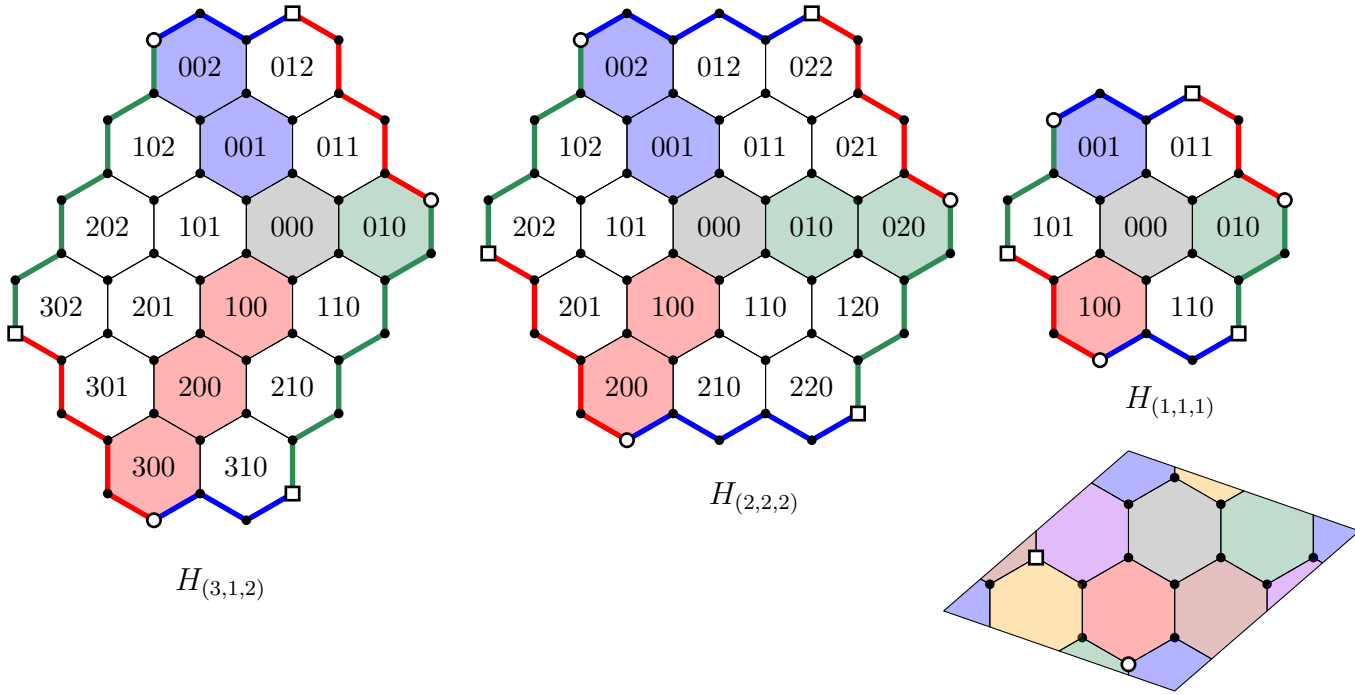


Figure 27: Three instances of \mathbf{b} -Heawood graphs, namely for $\mathbf{b} = (3, 1, 2)$, $\mathbf{b} = (2, 2, 2)$ and $\mathbf{b} = (1, 1, 1)$, from left to right. The bottom right shows an alternative drawing of the graph $H_{(1,1,1)}$.

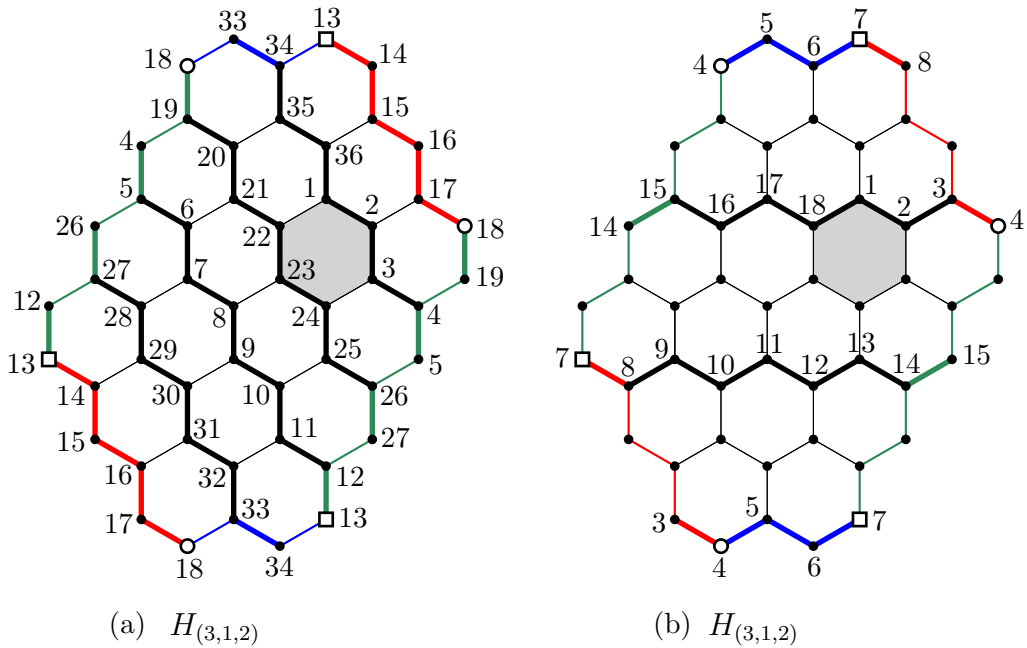


Figure 28: (a) A Hamilton cycle in $H_{(3,1,2)}$ that alternately uses only two of the three possible edge directions (vertices labeled $1, \dots, 36$ along the cycle). (b) A cycle in $H_{(3,1,2)}$ that uses two directions other than the ones from (a) but that is not a Hamilton cycle (vertices labeled $1, \dots, 18$ along the cycle).

In fact, this definition of \mathbf{b} -Heawood graph can be generalized to any vector $\mathbf{b} \in \mathbb{N}_{>0}^{d+1}$ by starting with an infinite tiling of \mathbb{R}^d with d -dimensional permutahedra, taking a subpatch according to \mathbf{b} and identifying opposite facets; for details see [1]. These graphs are $(d + 1)$ -

regular and vertex-transitive. Question 2 then generalizes naturally to the following.

Question 3: Does $H_{\mathbf{b}}$ have a Hamilton cycle that alternately uses only two of the $d + 1$ possible edge directions, for any $d \geq 2$ and $\mathbf{b} \in \mathbb{N}_{>0}^d$?

References

- [1] CEBALLOS, C., AND DOOLITTLE, J. Generalized Heawood graphs and triangulations of tori. <https://arxiv.org/abs/2307.11859>, 2023.