Polytope samplers for inference in ill-posed inverse problems

Edoardo M. Airoldi, Bertrand Haas
Department of Statistics, Harvard University

Abstract

We consider linear ill-posed inverse problems $y = Ax$, in which we want to infer many count parameters x from few count observations y, where the matrix A is binary and has some unimodularity property. Such problems are typical in applications such as contingency table analysis and network tomography (on which we present testing results). These properties of A have a geometrical implication for the solution space: It is a convex integer polytope. We develop a novel approach to characterize this polytope in terms of its vertices; by taking advantage of the geometrical intuitions behind the Hermite normal form decomposition of the matrix A, and of a newly defined pivoting operation to travel across vertices. Next, we use this characterization to develop three (exact) polytope samplers for x with emphasis on uniform distributions. We showcase one of these samplers on simulated and real data.

1 INTRODUCTION

Problem settings where we have low dimensional data and a high dimensional parameter space arise often in model-assisted discovery of mechanistic principles in the biological and social sciences. We consider linear ill-posed inverse problems $y = Ax$ with $x, y \geq 0$, where the dimension of y is less than the dimension of x. Given y and A we want to make inferences on x.

This flavor of ill-posed linear inverse problem arise as the core inference tasks underlying a number of applications, including image super-resolution and positron emission tomography where we want to combine many 2D images in a 3D image consistent with 2D constraints (Shepp and Kruskal, 1978; Vardi et al., 1985); blind source separation where there are more sources than observations (e.g., sound tracks) available (Lee et al., 1999; Parra and Sajda, 2003); inference on cells of a contingency table where two-way and multi-way margins are given (Bishop et al., 1975; Dobra et al., 2006), and network tomography (Vanderbei and Iannone, 1994; Vardi, 1996; Tebaldi and West, 1998; Cao et al., 2000; Medina et al., 2002; Zhang et al., 2003; Liang and Yu, 2003; Airoldi and Faloutsos, 2004; Lawrence et al., 2006; Fang et al., 2007; Blocker and Airoldi, 2011). Two main approaches to these problems have been proposed in the literature: sequential MCMC methods, in which intelligently designed proposal distributions attempt to explore the space of solutions efficiently, and algebraic geometry methods, in which difficult calculations precisely characterize the space of solutions in terms of structured polynomials (Diaconis and Sturmfels, 1998; Chen et al., 2005; Dobra et al., 2006; Dobra and Fienberg, 2008; Dobra et al., 2009). The MCMC based are widely used, but approximate. The algebraic methods are exact and elegant, but already computationally infeasible in low dimensional problems.

Here, we focus on applications where the matrix A is a $\{0, 1\}$-matrix that is *unimodular* (definition below). Ill-posed inverse problems with such properties arise naturally when sampling contingency tables and in network tomography, in which x and y have the additional constraint of being integer-valued. Surprisingly, in such problems the space of solutions has not been characterized. However, it has a simple geometrical structure: it is an integer convex polytope.

In this paper we develop *polytope samplers*, a fresh new approach to inference in ill-posed linear inverse problems. The innovation of our approach is two-fold: (i) we develop new algorithms to identify all the vertices of the solution polytope, and (ii) we develop three strategies to build a sampling distribution on the polytope, as a generalization of the Dirichlet distribution over the simplex. We accomplish these tasks by taking advantage of the geometry of the problem, which is well specified by the Hermite normal form decomposition of the matrix A.

Proofs of new results are in the online suplementary material. We provide references for existing results.

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15. Copyright 2011 by the authors.

2 IDENTIFYING THE SOLUTION POLYTOPE

Definition 1. *A square integer matrix is* unimodular *if it is invertible as an integer matrix (in particular its determinant is* ± 1*). By extension we define a rectangular matrix to be* unimodular *if it has full rank and each square submatrix of maximal size is either unimodular or singular (*0 *determinant). A matrix is* totally unimodular *(or* TUM*) if each square submatrix (of any size) is either unimodular or singular (in particular entries of a TUM matrix are in* $\{-1, 0, +1\}$*).*

Our goal is to sample solutions of

$$Ax = y, \quad x \geq 0 \tag{1}$$

where A is a given unimodular $m \times n$ matrix with $\{0, 1\}$ entries and y is a given integer positive vector (solutions to 1 generalize straightforwardly to solutions of the problem where y is non-negative).

2.1 Integer geometry preliminaries

Definition 2. *A* polyhedron *of dimension* d *in* $\mathbb{R}^n$*,* $n \geq d$ *is the intersection of finitely many half spaces that is not contained in an affine space of dimension* $d-1$*. A* polytope *of dimension* d *is defined as the convex hull of finitely many points not all lying in a* $d-1$ *affine space.*

It is straightforward that the space of solutions to 1 is a polyhedron of dimension $n - m$. That this polyhedron is bounded (and therefore a polytope) results from A having non-negative entries and y being strictly positive. Moreover we have:

Lemma 2.1 (Sup. Mat.)**.** *The space of real solutions* x *to equation 1 is an integral polytope.*

Proof. The vertices are the intersections of the affine solution space of $Ax = y$ with the $(n-m)$-coordinate planes bording the non-negative orthant. So a vertex x has $n-m$ zero coordinates. Let's gather the rest of the coordinates into a positive integer vector x' of dimension m. And let's gather the corresponding columns of A into a square matrix A_1; so we get the equation $A_1 x' = y$. If A_1 was singular, the latter system would have either none or infinitely many solutions, which would contradict that x is a vertex. So A_1 is unimodular and $x' = A_1^{-1} y$. And since y is integer, x' is also integer. □

Since A is unimodular, it has at least one unimodular submatrix A_1 of maximal size. Without loss of generality we can assume A is decomposed into blocks $A = (A_1, A_2)$. It is straightforward that

$$(A_1, A_2) \begin{pmatrix} A_1^{-1} & -A_1^{-1} A_2 \\ 0 & I_{n-m} \end{pmatrix} = (I_m, 0)$$

This is the *Hermite normal decomposition* of A, and the $n \times n$ square matrix on the left hand side is denoted Q as is usual in such decomposition. So Q is naturally decomposed into two or four blocks $Q = (Q_1, Q_2) = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$. Here $Q_1 = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix}$, $Q_{11} = A_1^{-1}$, etc. In particular, it should be clear from the Hermite normal decomposition that the columns of Q_2 generate the null-space of A.

2.2 Finding a first vertex of the solution polytope

After defining the operation of *pivoting* on a matrix (algorithm 1) which is central to our algorithms, we restrict our attention to matrices A that are TUM. However, Corollary 0.1 (in sup. mat.), shows that A can be only unimodular for our algorithms to work.

Lemma 2.2 (Sup. Mat.)**.** *If* A *is TUM, then blocks* Q_2 *and* Q_{12} *of* Q *introduced above are also TUM.*

In particular Q_{12} has entries in $\{-1, 0, +1\}$.

Proposition 2.3 (Sup. Mat.)**.** *Given a TUM matrix* Q_2 *as above, a row index* i*, and a column index* j *as input, the algorithm 1 returns a matrix* Q_2' *of the same form than* Q_2*, that is,* $Q_2' = \begin{pmatrix} Q_{12}' \\ I_{n-m} \end{pmatrix}$*. Moreover,* Q_2' *is also TUM.*

Algorithm 1 Pivot(Q_2, i, j)

```
Q2'[, j] ← Q2[, j]
for k = 1, ..., n excluding k = j do
    Q2'[, k] ← Q2[, k] − Q2[i, j] * Q2[i, k] * Q2[, j]
Q2'[, j] ← Q2'[k, j] * Q2'[, j]
Swap rows i and m + j of Q2'
return Q2'
```

For short-hand we call a *vertex*, a vertex of the solution polytope. A *coordinate* m*-plane* is the set of all points with a given set of $n - m$ coordinates always zero. Vertices of the solution polytope are intersections of the $n - m$ dimensional affine space $Ax = y$ with the portions of coordinate m-planes that border the non-negative orthant. Let $x' = Q^{-1} x$, so finding solutions to system $y = Ax$ amounts to finding solutions to $AQx' = y$, $Qx' \geq 0$. Since $AQ = (I_m, 0)$, the system $AQx' = y$ has the obvious solution $x' = \binom{y}{0}$. Let $y' = Q_{11} y$ so a solution to $Ax = y$ is $x = Qx' = (Q_1, Q_2)x' = Q_1 y = \binom{y'}{0}$.

Since x has $n - m$ zero coordinates it belongs to a coordinate m-plane. However this might not be in the portion that borders the non-negative orthant, that is, x might have negative coordinates. We call the intersection of the affine space $Ax = y$ with a coordinate m-plane a *pseudo-vertex* and we use the letter ν to denote it.

We have a first pseudo vertex $\nu^{(0)} = \binom{y'}{0}$, in the x-space, but we want a true vertex. We will construct a sequence $\nu^{(0)}, \nu^{(1)}, \dots$ such that $\nu^{(i+1)}$ has one less negative coordinate than $\nu^{(i)}$, or its smallest negative coordinate in absolute value is strictly smaller than the one from $\nu^{(i)}$. Therefore the sequence terminates at a real vertex $\nu^{(s)}$. More precisely, the sequence will be a sequence of triplets $(\nu^{(0)}, Q_2^{(0)}, \pi^{(0)}), (\nu^{(1)}, Q_2^{(1)}, \pi^{(1)}), \dots, (\nu^{(s)}, Q_2^{(s)}, \pi^{(s)})$, where $Q_2^{(0)} = Q_2$ computed above, $\pi^{(0)}$ is the identity permutation and one triplet is deduced from the previous as described in section 2.3.

2.3 The generic local geometry at a pseudo-vertex

A coordinate m-plane is the intersection of $n-m$ coordinate hyperplanes. Each of these hyperplane $x_i = 0$ intersects the affine space $Ax = y$ into a hyperplane H_i of the affine space that we call a *c-hyperplane*. Generically [1] a pseudo-vertex is the intersection of $n-m$ c-hyperplane, that is, it has exactly m non-zero coordinates (non-generically it might have less than m). The union of all these hyperplanes borders cones inside which, locally, all points have the same sign vector. We call these cones the *sign cones*.

Adding a scalar factor of $Q_2[, j]$ to $\nu^{(0)}$ changes the zero coordinate at position $m + j$ into a non-zero coordinate. Therefore by continuously adding infinitesimal factors of $Q_2[, j]$ we get a point moving away from the c-hyperlane $x_{m+j} = 0$, but remaining in the c-hyperplanes $x_{m+1} = \dots = x_{m+j-1} = x_{m+j+1} = \dots = x_n = 0$. That is, we get a point moving along the edge of a sign-cone.

One of the sign-cones with apex $\nu^{(0)}$ has a sign vector with least number of -1; let's call it the *most positive sign-cone*. Adding a positive scalar factor of $Q_2[, j]$ to $\nu^{(0)}$ changes the zero $m + j$ coordinate into a positive coordinate. Therefore if the infinitesimal factors are positive the moving point moves along the corresponding edge of the most positive sign-cone. That is, the columns of Q_2 define the directions of the edges of the most positive sign-cone at $\nu^{(0)}$. Let's define a *base* at $\nu^{(0)}$ to be the set of vectors $B = \{b_1, b_2, \dots\}$ defining the directions of the edges of the most positive sign-cone. We just showed that we can take $b_j = Q_2[, j]$, so we call Q_2 a *base matrix* at $\nu^{(0)}$. See figure 1.

[1] A d-dimensional polyhedron with its defining half-spaces in generic position has exactly d supporting hyperplane intersecting at a vertex, but no constraint on the shape of the facets. A polytope with its defining points in generic position has facets that are simplexes but no constraint on the degree of the vertices. Our solution polytope is defined as a bounded polyhedron so generically the (pseudo)-vertices have degree $d = n - m$.

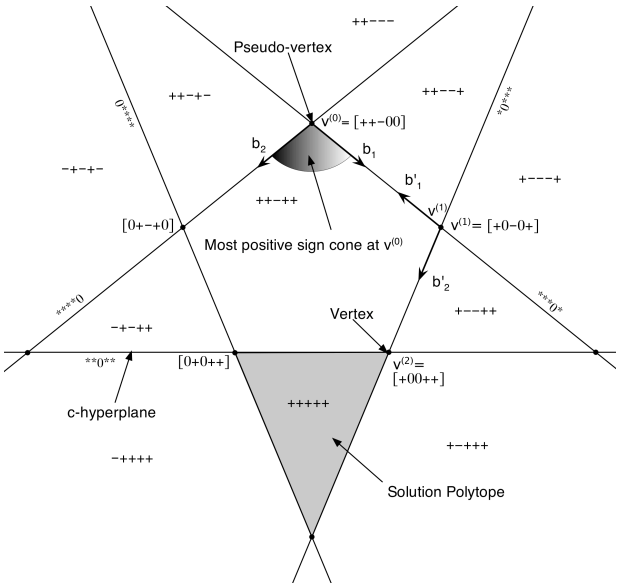


Figure 1: An example of 2-dimensional solution space with $n = 5$ and where $\nu^{(1)}$ has the same number of negative coordinates than $\nu^{(0)}$ for any choice of base vector.

2.3.1 From one pseudo-vertex to the next in the generic case

By leaving $\nu^{(0)}$ we leave a c-hyperplane, and by moving along an edge of the most positive sign-cone long enough we meet a next c-hyperplane (say $x_k = 0$, $1 \le k \le m$), that is, a next pseudo-vertex, $\nu^{(1)}$. The points along the edge of the most positive cone have one more positive coordinate than $\nu^{(0)}$ (at position $m + j$), and $\nu^{(1)}$ has one more zero coordinate than the points on the edge (at position k). Therefore $\nu^{(1)}$ cannot have more negative coordinates than $\nu^{(0)}$. It has either less (which is what we eventually want) if $\nu^{(0)}[k] < 0$, or it has the same number if $\nu^{(0)}[k] > 0$.

When we are at $\nu^{(0)}$ we have $n - m$ choices of edge to move along. If we could choose an edge which guarantees that we $\nu^{(1)}$ has less negative coordinate than $\nu^{(0)}$ we would certainly choose this one. However, such a choice does not always exist (see figure 1). Instead, we choose an edge so that even if $\nu^{(1)}$ has the same number of negative coordinates, we still are better off. That is, we choose the edge such that the smallest (in absolute value) negative coordinate of $\nu^{(1)}$ is strictly smaller than the smallest negative coordinate of $\nu^{(0)}$. To explain this we need lemma 2.4.

Lemma 2.4 (Sup. Mat.). *If $\nu^{(0)}[i] \le 0$, then the row vector $Q_2[i,]$ has at least one entry equal to 1.*

Let i be the argument of $\min |\nu^{(0)}[u]|$ over all u where $\nu^{(0)}[u] < 0$. Lemma 2.4 ensures there is a j such that $b_j[i] = 1$. Let k be the argument of $\min |\nu^{(0)}[u]|$ over all u where $\nu^{(0)}[u] < 0$ and $b_j[u] = +1$ or $\nu^{(0)}[u] > 0$

and $b_j[u] = -1$. Finally let

$$\nu^{(1)} = \nu^{(0)} + \nu^{(0)}[k]b_j$$

If $k = i$, then $\nu^{(1)}[i] = 0$ and $\nu^{(1)}$ has one less negative coordinate than $\nu^{(0)}$. If $k \neq i$, then $0 > \nu^{(1)}[i] > \nu^{(0)}[i]$, and $\nu^{(1)}[i]$ is still the minimum in absolute value among all $\nu^{(1)}$'s negative coordinates.

Once we got $\nu^{(1)}$ we need to reshape it so that it has the same form as $\nu^{(0)}$ (we want a pseudo vertex to have its last $n-m$ coordinates zero). This is easy: We just swap coordinates $\nu^{(1)}[k] = 0$ with $\nu^{(1)}[m+j] > 0$ (and we still write it $\nu^{(1)}$). We record this in the permutation vector $\pi^{(1)} = \text{Swap}(\pi^{(0)}, k, m+j)$, the result of swapping entries k and $m+j$ in $\pi^{(0)}$. And for the same reason of form preserving, we perform a Pivot operation (see algorithm 1) to get $Q_2^{(1)} = \text{Pivot}(Q_2, k, j)$.

We just described how to get from $\nu^{(0)}$ to $\nu^{(1)}$. Clearly the same apply from $\nu^{(i)}$ to $\nu^{(i+1)}$.

2.3.2 From one pseudo-vertex to the next in the non-generic case

In the generic case, a pseudo-vertex $\nu^{(0)}$ is the intersection of exactly $n-m$ c-hyperplanes. In the non-generic case, $\nu^{(0)}$ is the intersection of $z > 0$ extra c-hyperplane, that is, $\nu^{(0)}$ has exactly $m-z$ non-zero coordinates. This might pose a problem in the algorithm described above. Assume for instance $\nu^{(0)}[1] = 0$ and $2 = \arg\min |\nu^{(0)}[i]|$ over all i with $\nu^{(0)} < 0$. It may happen that for all base vector b_j with $b_j[2] = +1$ we have as well $b_j[1] = -1$. So any positive factor of b_j will create a negative coordinate at $\nu^{(1)}[1]$, thus possibly increasing the number of negative coordinates.

A solution to this problem is to pretend the z extra zero coordinates are actually not zero but infinitesimal (*virtual*) negative values $\delta_1, \delta_2, \ldots, \delta_z$. This resolves the singularity by blowing it up into a *virtual polytope*. So $\nu^{(0)}$ splits into the (generic) pseudo-vertices $\nu_1^{(0)}, \nu_2^{(0)}, \ldots$ of the virtual polytope. Each new pseudo vertex $\nu_i^{(0)}$, being generic, has its own generic base and the base we got for $\nu^{(0)}$ is actually a base for a the virtual pseudo vertex $\nu_0^{(0)}$. We can now navigate along the virtual pseudo vertices as if they were normal generic ones and find a path up to the next real (non-virtual) pseudo vertex $\nu^{(1)}$. (see figure 2 or 6 in the suplementary material). Therefore we get:

Proposition 2.5 (Sup. Mat.)**.** *Given a pseudo vertex, its base matrix, and its permutation vector, (ν, Q_2, π), algorithm 2 returns a vertex of the solution polytope.*

2.4 Finding all the vertices

Our strategy to find all the vertices amounts to actually find the one-skeletton of the solution polytope (all vertices and all edges). We use this extra structure in the polytope sampler 3.2. We do that by starting at a known vertex (which we got in section 2.2), then finding all its neighbors, moving to a neighbor, finding all its neighbors, etc. We chose to do so with a breadth-first search kind of algorithm (although a depth-first

Algorithm 2 Get_Vertex(ν, Q_2, π)

while ν has a negative coordinate **do**
  blow up ν if it is non-generic
  $I \leftarrow \{i : \nu[i] < 0\}$
  $i \leftarrow \arg\min |\nu[s]|$ for $s \in I$
  Find j such that $Q_2[i, j] = 1$
  $K \leftarrow \{k : Q_2[k, j] = -1, \nu[k] > 0\}$
  $k \leftarrow \arg\min |\nu[s]|$ for $s \in K \cup \{i\}$
  $\nu \leftarrow \nu + |\nu[k]| * Q_2[, j]$
  $\nu \leftarrow \text{Swap}(\nu, k, j)$
  $Q_2 \leftarrow \text{Pivot}(Q_2, k, j)$
  $\pi \leftarrow \text{Swap}(\pi, k, j)$
return (ν, Q_2, π)

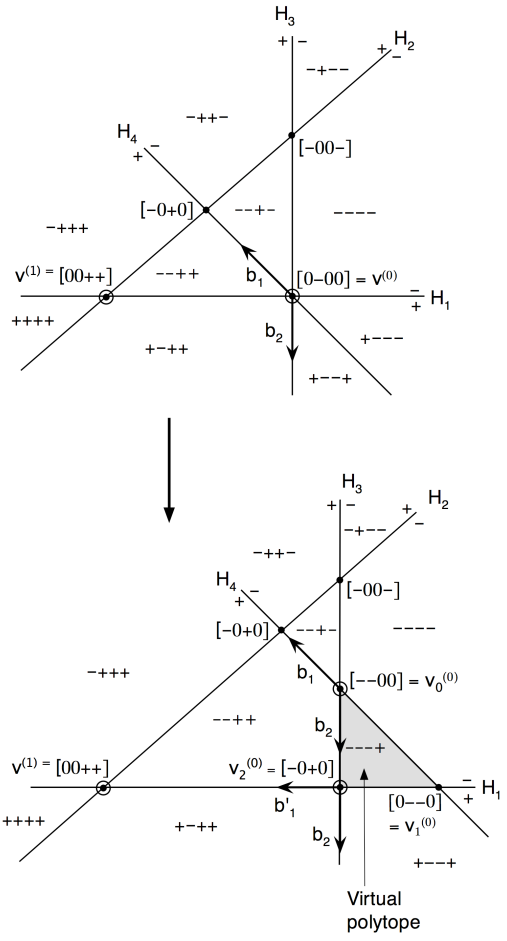


Figure 2: The virtual polytope method solves the problem of not being able to move from $\nu^{(0)}$ to $\nu^{(1)}$ along the given base vectors.

kind of algorithm would work as well). So we use two data-structures:

- A one-skeletton sk which is an ordered list of pairs (V, N) where $V \in \mathbb{R}^n$ is a vertex, and N is the list of indexes of all the other pairs (V', N') in sk such that V and V' are neighbors.
- A vertex queue vq containing triplets (v, B, π) where v is a vertex in compact form (its permuted $n-m$ non-zero components), B is a set of base vectors (directions to positive neighbors of v), and π is the permutation that allows us to recover the expanded form $V \in \mathbb{R}^n$ of the vertex from its compact form $v \in \mathbb{R}^m$.

For conciseness of description, we present the alogorithm (3) only in the generic case (all vertices are generic). In the non-generic case, when we encounter a non-generic vertex v, we blow it up into a virtual polytope like in section 2.3.2 and gather in B the positive directions from the matrices $Q_2^{(i)}$ at each virtual vertex $v^{(i)}$.

Algorithm 3 Get_Skel(v_0, Q_{12}, π_0) (generic case)

```
    Initiate queue vq and one-skeleton sk
    Push (v_0, Q_12, π_0) into queue vq
    while vq not empty do
      (v_0, Q_12, π_0) ← Pop(vq)
 5:   Append n − m zeros to v_0 and V_0 ← π_0(v_0)
      Append (V_0, ∅) to sk
      for j = 1 to n − m do
        k ← arg min(v_0[s]) over {s : Q_2[s, j] = −1}
        v_1 ← v_0 + v_0[k]Q_2[, j]
10:     π_1 ← Swap(π_0, k, m + j)
        if (v_1, ·, π_1) is not in vq then
          get V_1 as in line 5
          if V_1 is not in sk then
            Q'_2 ← Pivot(Q_2, k, j)
15:         append (v_1, Q'_12, π_1) to vq
          else
            Let s be the index of V_1 in sk
            Let t be the index of V_0 in sk
            Add s to the index set of V_0-neighbors
20:         Add t to the index set of V_1-neighbors
25: return sk
```

Proposition 2.6 (Sup. Mat.)**.** *Algorithm 3 finds the one-skeleton (all r vertices and all edges) of the solution polytope in time $O(r\,n\,m)$.*

2.5 Final Remarks

Algorithm 3 has two drawbacks. The first one is inherent to the problem and is that the number of vertices, r, seems to grow exponentially with the dimension of the problem. Therefore even a running time of $O(r)$ will be intractable in large dimensions. The second one is that the memory requirement is also $O(r\,n\,m)$, since we store in the queue the base-matrix Q_{12} for each vertex. Therefore, for large dimensions the memory requirement will also make it intractable. Other clever and efficient algorithms, recently brought to the attention of the authors, also run in $O(r\,n\,m)$ (see for example Avis et al. (2009)) but require only $O(1)$ memory. However they output only the vertices, not the full one-skeleton.

3 THREE POLYTOPE SAMPLERS

We developed three strategies for drawing exact samples from the solution polytope that we identified in Section 2. We focus on uniform sampling from a measure with unit volume.

3.1 Basic polytope sampler

The following algorithm straightforwardly produces a distribution on P.

Algorithm 4 Basic_Polytope_Sampler(P)

```
Associate coefficients α_0, ..., α_{r−1} to P's vertices
V_0, ..., V_{r−1}
Draw a vector Z ∼ Dirichlet (α_0, α_1, ..., α_{r−1})
return X ← Σ Z[i]V_i
```

Notice that this algorithm amounts to lift the polytope to a simplex S of dimension $r-1$, draw a point with a Dirichlet distribution, and project it back. A Dirichlet distribution on S cannot project to an exact uniform distribution on P, but by letting $\alpha_0 = \alpha_1 = \cdots = 1/(r-s)$, we get a distribution on P that appoximates very well the uniform distribution (see figure 4). Indeed the density of the projection over a small s-volume element is proportional to the complementary $(r-1-s)$-volume above it. We compensate with a distribution on the simplex with weights inversely proportional to this excess of dimension, that is, $1/(1+r-1-s)$.

This is the sampler we implemented to obtain the results in Section 4.

3.2 Triangulation polytope sampler

Here we find a triangulation of the polytope P (a decomposition of P into simplexes), and then sample points from P in two steps: sample a simplex; sample a point in the simplex (details in algorithm 5).

By taking all $\alpha_i = 1$ and $p_i = \mathrm{Vol}(S_i)/\mathrm{Vol}(P)$, we clearly get a uniform distribution on P. Notice that by using interior points to the polytope P, one can decompose P into more simplexes, therefore providing a richer family of distributions. In section 2.4 we got the

Algorithm 5 Triangulation_Polytope_Sampler(P)

Find a triangulation $P = S_1 \cup S_2 \cup \cdots \cup S_t$
Associate weights p_i to simplexes S_i with $\sum_i p_i = 1$

Associate coefficients α_i to vertices V_i of P
Draw a simplex S_i with probability p_i.
Let $v_{i0}, v_{i1}, \ldots, v_{in}$ be the vertices of S_i
Draw a vector $Z \sim$ Dirichlet $(\alpha_{i0}, \alpha_{i1}, \ldots, \alpha_{is})$
return $X \leftarrow \sum Z[j]V_j$ for $j = i_0, \ldots, i_s$

1-skeleton of P. If we continue and get the 2-skeleton, the 3-skeleton, and finally the full face-decomposition of P, the we can work a triangulation of P recursively as follows:

- In every 2-face of P, fix a point x and triangulate the face by joining the edges of the face to x.
- In every k-face of P, fix a point x and triangulate the face by joining the (previously constructed) simplexes of its $(k-1)$-faces to x.

Other well known triangulation algorithms, like the *placing* or the *pulling* algorithm can be used (see De Loera et al. (2010)).

3.3 Moment-map polytope sampler

Here we define a generalization of the Dirichlet distribution by placing a distribution on positive orthant of the affine space $\mathbb{R}^s$ or the projective space $\mathbb{P}(\mathbb{R})^s$ and use the moment map to induce a distribution on the polytope.

Let $v_1, v_2, \ldots, v_r$ be the vertices of the s-dimensional solution polytope P. Let

$$d_i = \sum_{j=1}^{r} v_j[i], \quad d_0 = \max d_i$$

And set an extra coordinate $v_i[0] = d_0 - d_i$ to each vertex. This amounts to lifting P to a polytope $P' \subset \{\sum_{i=0}^{s} x_i = d_0\} \subset \mathbb{R}^{s+1}$.

For $x = (x_0, x_1, \ldots, x_s) \in \mathbb{R}^{s+1}_{>0}$ and $i = 1, \ldots r$, define the monomials

$$m_i(x) = x_0^{v_i[0]}\, x_1^{v_i[1]} \ldots x_s^{v_i[s]}$$

So each monomial has degree d_0. Associate a coefficient α_i to each vertex v_i, and set

$$w_i(x) = \frac{\alpha_i m_i(x)}{\sum_{j=1}^{r} \alpha_j m_j(x)} v_i$$

Notice that $w_i(\lambda x) = w_i(x)$ for any non-zero λ. The moment map (projective version) is defined as

$$\begin{array}{rcl} \Phi : \mathbb{R}^{s+1}_{>0} & \to & \mathrm{int}(P) \\ x & \mapsto & w_1(x) + \cdots + w_r(x) \end{array}$$

Theorem 3.1. Φ *is an isomorphism from* $\mathbb{P}(\mathbb{R})^s_+$ *to* $\mathrm{int}(P)$.

Where $\mathbb{P}(\mathbb{R})^s_+$ is the orthant of the real projective s-dimensional space where all coordinates have same sign. See for instance Fulton (1993), chap. 4, for a proof of the affine version.

This results helps us to sample points on the polytope: Draw Z for a distribution on $\mathbb{P}(\mathbb{R})^s_+$ (for instance from a distribution on the s-sphere or s-simplex) or on $\mathbb{R}^{s+1}$. Then set $X = \Phi(Z)$.

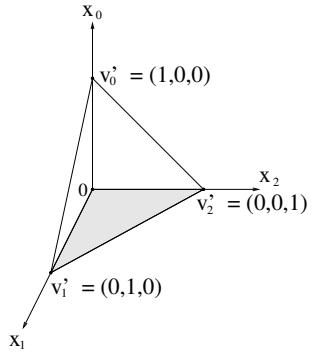


Figure 3: The moment map associated to this triangle yields the Dirichlet distribution.

As an illustration, consider as polytope $P \subset \mathbb{R}^2$ the triangle with vertices $(0,0), (0,1), (1,0)$. We firstly lift it to the triangle P' with vertices $v'_0 = (1,0,0), v'_1 = (0,1,0), v'_2 = (0,0,1)$ (see figure 3). The monomials are simply

$$m_0 = x_0, \quad m_1 = x_1, \quad m_2 = x_2$$

Set the coefficients $\alpha_i = 1$ for $i = 0, 1, 2$. The map is $\Phi(x) = (x_0, x_1, x_2)' / \sum_i x_i$. We easily verify that if we take $Z_i \sim Gamma(a_i)$ then we get

$$X = \frac{1}{\sum Z_i} \begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \end{pmatrix} \sim \text{Dirichlet}(a_1, a_2, a_3)$$

Which shows that the resulting distribution for general polytopes is a direct generalization of the Dirichlet distribution.

4 RESULTS

We tested the basic sampler in multiple settings. We evaluated the goodness of the basic polytope sampler via simulations in a situation where exact sampling is also possible. We considered the network tomography problem posed by Vardi (1996), and compared the efficiency and accuracy of the basic polytope sampler to the Metropolis-Hastings (MH) in Gibbs proposed by Tebaldi and West (1998). We also considered the problem of sampling two-way contingency tables given

fixe margins, corresponding to both decomposable and non-decomposable models. In both problems, the matrix A is totally unimodular.

4.1 Simulation results

The simple polytope sampler worked well in practice. We drew samples in simple known polytopes (see for

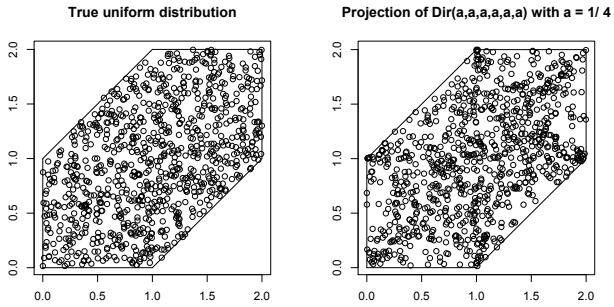
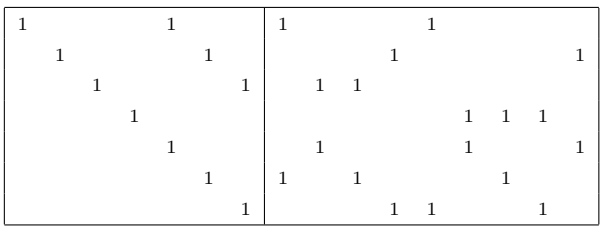


Figure 4: 900 uniform draws from a 2-polytope (left), versus approximately uniform draws obtained using algorithm 4 (right).

example figure 4) corresponding to totally unimodular matrices for generic and non-generic cases. A battery of statistical tests including two-sample Kolmogorov-Smirnov and Wilcoxon-Mann-Whitney were not able to distinguish between the exact and the polytope sampler samples.

4.2 Network tomography

We applied the polytope sampler to solve the network tomography problem on a network consisting of one router and 4 computers. We used the data from the Bell Labs router-1 data analyzed in Cao et al. (2001) consisting of 24-hour worth of traffic data sampled every 5 minutes (287 time points). The fixed routing protocol in the network of interest leads to a totally unimodular matrix A of size 7×16, since 16 origin-destination (OD) traffic loads X contribute to 7 observed aggregate traffic loads Y. We rearranged the columns of A to have a nice-looking unimodular submatrix A_1 (for clarity we display only the 1s):



The goal of the analysis is to estimate the OD traffic from aggregate measurements Y. Different values of Y might lead to generic and non-generic solution polytopes.

We compared the performance of our sampler to the Metropolis-Hasting based solution sampler proposed by Tebaldi and West (1998). We ran the latter to produce 10 independent chains of 100,000 solutions at 7 time points, with a cumulative CPU time of about 9 hours. In terms of efficiency, this algorithm mixed badly at six time points corresponding to particularly narrow polytopes, where it required more than one billion samples before moving. Our polytope sampler produced exact samples at each draw (1 Million draw at each of the 7 time point), and ran with a cumulative CPU time of about 8 minutes (compare with 9 hours!), including finding all the vertices for each solution polytope (less than 2 sec for all 7 polytopes). For example the vector of observed aggregate traffic at 3pm, measured in bytes is y = (12053370, 14424000, 249121410, 1485729, 248498220, 2365508, 15357294), and yields a solution polytope with 232 vertices. In Figure 5, the modes of the posterior distributions on the OD loads are excellent estimates for the ground truth (black triangles), available for this data set.

We also computed the average L_1 and L_2 distance between the two solutions and the ground truth, across OD loads and over 24 hours. The average distance between the MH solution and the ground truth solution is of the order of 55 Mb (in L_1), and 20 Mb (in L_2). The average distance between the basic polytope sampler solution and the ground truth solution is of the order of 17 Mb (in L_1), and 6 Mb (in L_2). In practice, the advantage of using the polytope sampler over a MH with a specialized proposal was therefore quite substantial.

4.3 Two-way, multi-way contingency tables

We considered the problem of sampling two-way and multi-way contingency tables given fixed margins, corresponding to both decomposable and non-decomposable models (Bishop et al., 1975), four cases in total. The four examples we considered required the algorithms for identifying the solution polytope in both the generic and non-generic cases, and an algorithm variants (not presented here) for dealing with matrices A that are non-totally unimodular. The simulation results substantiate the claim that the basic polytope sampler is a feasible strategy to tackle the more complicated problem of sampling multi-way contingency tables.

5 CONCLUDING REMARKS

The polytope samplers provide a fresh new approach to inference for ill-posed inverse problems, $y = A\ x$, $y, x \geq 0$. The innovation is in our ability to leverage the geometrical intuition underlying some properties of the matrix A that arise in the real applications we consider: namely, network tomography and sampling

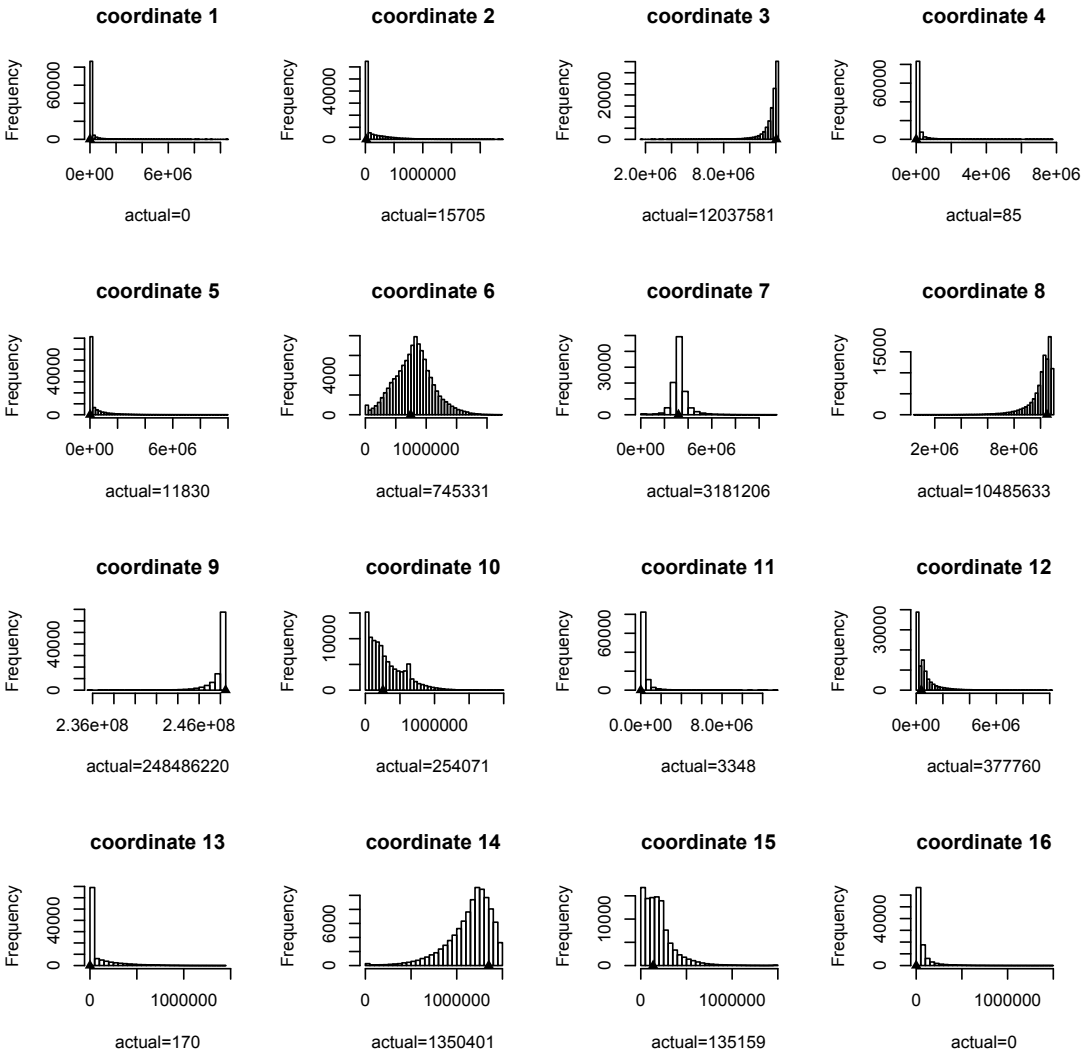


Figure 5: OD loads posterior distributions at 3pm.

contingency tables given fixed margins. For reasonably sized problems our approach shows to be very competitive with current known methods: Sequential MCMC methods (Chen et al., 2005) (which prove much slower mainly due to their sequential nature and less accurate) and methods based on algebraic geometry (Diaconis and Sturmfels, 1998; Dobra et al., 2009) (which seem to become intractable in lower dimensions than for ours).

We developed three polytope sampling strategies and implemented one, the basic polytope sampler. These different strategies have advantages and disadvantages. The basic sampler is easy to implement, however the sampler draws approximately uniform samples. The triangulation sampler draws exactly uniform samples, and allows for finer decompositions of the polytope by using additional interior points. However, partitioning a polytope into mutually exclusive simplexes is a non-trivial operation. An efficient algorithm to produce such decompositions is needed. The moment-map sampler is a direct method that generalizes the construction of a Dirichlet distribution. However the computation of the Jacobian of the moment map may be costly, and the distribution is difficult to control—the Jacobian may not be simple. Numerical methods to compute distributions involving a complicated Jacobian should alleviate this difficulty.

Our work suggest broader applicability of polytope samplers. To what extent relaxations to conditions on the matrix A maintain the general geometric structure that the polytope samplers leverage? Our results establish new and exciting research directions.

References

E. M. Airoldi and C. Faloutsos. Recovering latent time-series from their observed sums: network tomography with particle filters. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 10, pages 30–39, 2004.

D. Avis, D. Bremner, and A. Deza. *Polyhedral Computation*. American Mathematical Society, 2009.

Y. Bishop, S. E. Fienberg, and P. Holland. *Discrete multivariate analysis: theory and practice*. The MIT press, 1975.

A. W. Blocker and E. M. Airoldi. Deconvolution of mixing time series on a graph. Manuscript no. 1105.2526 on arXiv.org, May 2011.

J. Cao, D. Davis, S. Van Der Viel, and B. Yu. Time-varying network tomography: router link data. *Journal of the American Statistical Association*, 95: 1063–75, 2000.

J. Cao, D. Davis, S. Van Der Viel, B. Yu, and Z. Zu. A scalable method for estimating network traffic matrices from link counts. Technical report, Bell Labs, 2001.

Y. Chen, P. Diaconis, S.P. Holmes, and J.S. Liu. Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, 100(469):109–120, 2005.

J. A. De Loera, J. Rambau, and F. Santos. *Triangulations: Structures for Algorithms and Applications*. Springer, 1st edition, 2010.

P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363–397, 1998.

A. Dobra and S.E. Fienberg. The generalized shuttle algorithm. In P. Gibilisco, E. Riccomagno, M. P. Rogantin, and H. P. Wynn, editors, *Algebraic and geometric methods in statistics*, pages 135–156. Cambridge University Press, 2008.

A. Dobra, C. Tebaldi, and M. West. Data augmentation in multi-way contingency tables with fixed marginal totals. *Journal of Statistical Planning and Inference*, 136:355–372, 2006.

A. Dobra, S. E. Fienberg, A. Rinaldo, S. Slavkovic, and Y. Zhou. Algebraic statistics and contingency table problems: Estimation and disclosure limitation. In S. Sullivant and M. Putinar, editors, *Emerging applications of algebraic geometry*, volume 149 of *IMA*, pages 63–88. Springer, 2009.

J. Fang, Y. Vardi, and C.-H. Zhang. An iterative tomogravity algorithm for the estimation of network traffic. In R. Liu, W. Strawderman, and C.-H. Zhang, editors, *Complex Datasets and Inverse Problems: Tomography, Networks and Beyond*, volume 54 of *Lecture Notes–Monograph Series*. IMS, 2007.

W. Fulton. *Introduction to Toric Varieties*. Princeton University Press, 1993.

E. Lawrence, G. Michailidis, V. Nair, and B. Xi. Network tomography: A review and recent developments,. In J. Fan and H. L. Koul, editors, *Frontiers in Statistics*. Imperial College Press, 2006.

T.-W. Lee, M. S. Lewicki, M. Girolami, and T. J. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, 1999.

G. Liang and B. Yu. Pseudo-likelihood estimations in network tomography. In *Proceedings of IEEE INFOCOM*, 2003.

A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. *SIGCOMM Computer Communication Review*, 32(4):161–174, 2002.

L. Parra and P. Sajda. Blind source separation via generalized eigenvalue decomposition. *Journal of Machine Learning Resaerch*, 4:1261–1269, 2003.

A. Schrijver. *Theory of Linear and Integer Programming*. Interscience Series in Discrete Mathematics and Optimization. J. Wiley, 1998.

L. A. Shepp and J. B. Kruskal. Computerized tomography: The new medical x-ray technology. *The American Mathematical Monthly*, 85(6):420–439, 1978.

C. Tebaldi and M. West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–573, 1998.

R. J. Vanderbei and J. Iannone. An em approach to od matrix estimation. Technical Report SOR 94-04, Princeton University, 1994.

Y. Vardi. Network tomography: estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91:365–377, 1996.

Y. Vardi, L. A. Shepp, and L. Kaufman. A statistical model for positron emission tomography. *Journal of the American Statistical Association*, 80(389):8–20, 1985.

Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proceedings of SIGCOMM*, 2003.

SUPPLEMENTARY MATERIAL

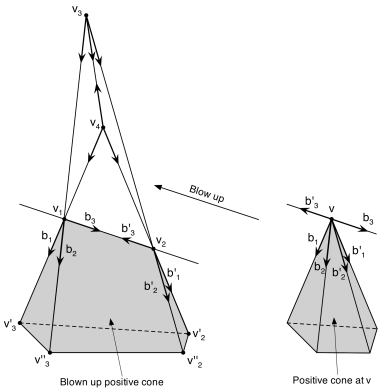


Figure 6: A singular (non-generic) vertex v is blown-up into a virtual polytope with vertices v_1, v_2, v_3, v_4. Notice the base at each virtual vertex contains only 2 directions along the actual positive sign cone.

We have so far restricted our attention to matrices A that are TUM. This was useful to guarantee that Q_{12} has entries in $\{-1, 0. + 1\}$ and that the pivoting operation actually works to pass from one vertex to the next. However, as a corollary to lemma **??** we get that the difference between two neighbor vertices is an integer, therefore a multiple of a column of Q_2. Since by varying y we can get all possible positive vertices (since A_1 is unimodular), we get that Q_2 has the sam form before and after pivoting:

Corollary 0.1. *If A is unimodular, $Q_2' = Pivot(Q_2, i, j)$ has the same form that Q_2, that is, $Q_2' = \begin{pmatrix} Q_{12}' \\ I_{n-m} \end{pmatrix}$.*

This means that our algorithms work for A unimodular as well.

The following proposition is rather obvious and its proof is left to the reader.

Proposition 0.2. *If A is totally unimodular, then the matrix A' obtained from A by any one of the following operations is still totally unimodular.*

1. *Permuting rows and columns.*
2. *Removing a row or column from A.*
3. *Adding to A one more row or column containing only 0's except one 1.*
4. *Adding to A one more row or column already in A.*
5. *Multiplying a row or column by -1.*

Following Schrijver (1998) we call a $m \times m$ submatrix of a full rank integer $m \times n$ matrix, $n \geq m$, a basis if it has full rank.

Theorem 0.3 (Theorem 19.5 in Schrijver (1998))**.** *Let A be an integral matrix. The following two assertions are equivalent.*

1. *For every basis A_1 of A, the matrix $A_1^{-1}A$ is integral*
2. *For every basis A_1 of A, the matrix $A_1^{-1}A$ is totally unimodular*

Lemma 2.2 becomes a corollary of this theorem.

Proof. (of Lemma 2.2) A is totally unimodular, therefore every basis A_1 is unimodular, therefore A_1^{-1} is integral and so is $A_1^{-1}A = (I, A_1^{-1}A_2)$. Therefore $(I, A_1^{-1}A_2)$ is totally unimodular. Therefore, $Q_{12} = -A_1^{-1}A_2$ is totally unimodular (a consequence of prop 0.2). Therefore $Q_2 = \binom{Q_{12}}{I}$ is totally unimodular. □

Proof. (of Proposition 2.3) The pivoting operation of algorithm 1 is easily seen to be a combination of operations from proposition 0.2 and the pivoting operation defined in Schrijver (1998) by:

$$\begin{pmatrix} \epsilon & c \\ b & D \end{pmatrix} \rightarrow \begin{pmatrix} \epsilon & -\epsilon c \\ \epsilon b & D - \epsilon bc \end{pmatrix}$$

The latter operation is proved in Schrijver (1998) to preserve TUM. □

Proof. (of Lemma 2.4) Any solution has the form $\nu^{(0)} + Q_2 w$ with w a non-negative $n - m$ vector. Let $J = \{j : Q_2[i, j] \neq 0\}$ and assume $Q_2[i, j] < 0$ for all $j \in J$. Any non-zero scalar $Q_2[i,]w$ would be negative, so $y'[i] + Q_2[i,]w$ would again be negative and so would never be the coordinate of a solution. Therefore the columns b_j, $j \in J$ are unecessary to express any solution in the form $\binom{y'}{0} + Q_2 w$. The solution polytope has dimension $n - m$, and therefore contains at least $n - m + 1$ linearly independent solutions. One solution might be $\binom{y'}{0}$ (if $\nu^{(0)}$ is actually a vertex), but there are at least $n - m$ other linearly independent solutions. However, the remaining columns $Q_2[, k]$, $k \notin J$, being less than $n - m$ in number, cannot express linearly independent $n - m$ solutions; a contradiction. □

Proof. (of Proposition 2.6) Both in the queue vq and the one-skeleton data structure sk, vertices are accompanied by their base-matrix (or set of base-matrices if it is a non-generic vertex) and their corresponding permutations. We place our first vertex in the queue, then each vertex v popped out of the queue is appended to the one-skeleton data structure and checked

for its neighbors. So vq contains vertices with neighbors unchecked and sk contains vertices with neighbors checked. If a neighbor v' is not already in vq, we check if it is already in sk so far (if it is already in vq, we ignore it and look at the next one). If it is in sk, we update the list N of neighbors of both v and v' in sk. If it is not in sk, then it is a new vertex and we add it to the queue. This should be clear that this exhausts all vertices and their neighboring relations. For the running time, the number of iterations of the while loop on line 3 is equal to the number of vertices v. For each iteration, the dominating operation is the pivot operation (which occurs only when a vertex is appended to the queue). Checking in line 11 and 13 that a vertex neighbor (that is, a vertex candidate) is not in the queue and not in the skeleton can be done in $O(1)$ time with a hash table sufficiently large. □