USING POLYMAKE TO CHECK WHETHER A MATROID IS 1-FLOWING

GORDON ROYLE

ABSTRACT. Description of how to use polymake to check whether a matroid is 1-flowing.

1. Introduction

A binary matroid $M$ is said to be $\{e\}$-flowing if for every element $e \in E(M)$, a certain polytope defined using the circuits of $M$ that pass through $e$ has integral vertices. A matroid is called 1-flowing if it is $\{e\}$-flowing for every element of $E(M)$. In this tutorial-style note, we'll work through an example, namely the matroid $AG(3,2)$ and demonstrate the computational process to checking whether or not it is 1-flowing, using the computer system polymake.

The binary matroid $AG(3,2)$ consists of the binary vectors in $PG(3,2)$ which lie off a hyperplane (i.e. all the affine points) and so can easily be seen to be represented by the matrix consisting of every column vector with first coordinate equal to one.

$$M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}$$

Label the 8 matroid elements (that is, matrix columns) 0, 1, ..., 7 in the natural order. Then $AG(3,2)$ has 14 circuits, all of size 4, as follows:

0123 0145 0167 0246 0257 0347 0356
1247 1256 1346 1357 2345 2367 4567

The automorphism group of $AG(3,2)$ is transitive on the ground set, and so to check whether the matrix is 1-flowing we only need to test that it is $\{0\}$-flowing. Each circuit containing 0 yields one constraint on a set of variables identified with $E(M)\setminus\{0\}$, which we will call $\{x_1, x_2, \ldots, x_7\}$ with the natural correspondence. Each of the circuit constraints is that the sum
of the corresponding variables is at least 1. Thus we get the 7 constraints shown in Table 1.

In addition to these constraints, each variable must be non-negative, and so we add $x_i \geq 0$ to the list of constraints. These gives us a total of fourteen constraints, each determining a half-space in $\mathbb{R}^7$, and the intersection of these half-spaces is an unbounded polytope; an expression for a polytope in this fashion is called an $H$-representation of the polytope ($H$ for half-space, I suppose).

A polytope can also be described by its vertices or extreme points and, if it is unbounded, its rays in addition. The matroid is $\{e\}$-flowing if the polytope described above has integral vertices, that is, its vertices have integer coordinates. The description of a polytope by its vertices (and rays) is called its $V$-representation and so the task is to convert the $H$-representation into the $V$-representation and check integrality.

A program for this purpose, written by Komei Fukuda, is implemented within the computer system polymake with is a computer algebra system for polytopes and associated objects.

## 2. The Computation

The basic process is straightforward:

1. Construct a polytope in polymake using the $H$-representation, and
2. Ask polymake to return the $V$-representation.

In practice, it is almost as straightforward, with just some possible teething troubles in mastering the syntax of polymake. A constraint in polymake of the form

$$a^T x \geq b$$
USING POLYMABLE TO CHECK WHETHER A MATROID IS 1-FLOWING

Circuit Constraint polymake vector
0123 \(x_1 + x_2 + x_3 \geq 1\) [-1,1,1,0,0,0,0,0]
0145 \(x_1 + x_4 + x_5 \geq 1\) [-1,1,0,0,1,1,0,0]
0167 \(x_1 + x_6 + x_7 \geq 1\) [-1,1,0,0,0,0,1,1]
0246 \(x_2 + x_4 + x_6 \geq 1\) [-1,0,1,0,1,0,1,0]
0257 \(x_2 + x_5 + x_7 \geq 1\) [-1,0,1,0,0,1,0,1]
0347 \(x_3 + x_4 + x_7 \geq 1\) [-1,0,0,1,1,0,0,1]
0356 \(x_3 + x_5 + x_6 \geq 1\) [-1,0,0,1,0,1,1,0]

Table 2. Circuit-constraints for checking if AG(3, 2) is \{0\}-flowing

(where a is a (column) vector of coefficients, x the vector of variables and b a vector of constants) must first be re-expressed in the form

\[-b + a^T x \geq 0\]

and then the constant term and coefficients in this expression gathered into one row vector in this order, yielding the vector

\([b, a_1, a_2, \ldots, a_n]\).

Table 2 shows the circuit constraints expressed in polymake form.

After installing and starting up polymake (or using the handy online “polymake in a Box” tool found at http://polymake.org/doku.php/boxdoc) we can start to enter commands at the polymake prompt (which by default is the string polymake > and a number).

```
polytope > $circuit_constraints = new Matrix([
polytope (2)> [-1,1,1,0,0,0,0,0],
polytope (3)> [-1,1,0,0,1,1,0,0],
polytope (4)> [-1,1,0,0,0,0,1,1],
polytope (5)> [-1,0,1,0,1,0,1,0],
polytope (6)> [-1,0,1,0,0,1,0,1],
polytope (7)> [-1,0,0,1,1,0,0,1],
polytope (8)> [-1,0,0,1,0,1,1,0]]);
```

We also need the constraints for non-negativity, but these can be constructed programatically rather than manually because the appropriate matrix is just a zero column vector adjacent to an identify matrix,

```
polytope > $nonneg_constraints = zero_vector(7) | unit_matrix(7);
```

and the set of all constraints is obtained from the union of both matrices.
polytope > $all_ineq = $circuit_constraints / $nonneg_constraints;

Let's just check that all is in order

polytope > print_constraints($all_ineq);

<table>
<thead>
<tr>
<th>Index</th>
<th>Inequality</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x1 + x2 + x3 &gt;= 1</td>
</tr>
<tr>
<td>1</td>
<td>x1 + x4 + x5 &gt;= 1</td>
</tr>
<tr>
<td>2</td>
<td>x1 + x6 + x7 &gt;= 1</td>
</tr>
<tr>
<td>3</td>
<td>x2 + x4 + x6 &gt;= 1</td>
</tr>
<tr>
<td>4</td>
<td>x2 + x5 + x7 &gt;= 1</td>
</tr>
<tr>
<td>5</td>
<td>x3 + x4 + x7 &gt;= 1</td>
</tr>
<tr>
<td>6</td>
<td>x3 + x5 + x6 &gt;= 1</td>
</tr>
<tr>
<td>7</td>
<td>x1 &gt;= 0</td>
</tr>
<tr>
<td>8</td>
<td>x2 &gt;= 0</td>
</tr>
<tr>
<td>9</td>
<td>x3 &gt;= 0</td>
</tr>
<tr>
<td>10</td>
<td>x4 &gt;= 0</td>
</tr>
<tr>
<td>11</td>
<td>x5 &gt;= 0</td>
</tr>
<tr>
<td>12</td>
<td>x6 &gt;= 0</td>
</tr>
<tr>
<td>13</td>
<td>x7 &gt;= 0</td>
</tr>
</tbody>
</table>

Finally we can create the polytope and ask for its vertices!

polytope > $p = new Polytope<Rational>(INEQUALITIES=>$all_ineq);

polytope > print $p->VERTICES;

Implementation of the double description method of Motzkin et al.
Copyright by Komei Fukuda.
http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html

<table>
<thead>
<tr>
<th>Index</th>
<th>Vertex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 1 1 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>1 0 0 1 0 1 1 0</td>
</tr>
<tr>
<td>3</td>
<td>1 0 1 0 1 0 1 0</td>
</tr>
<tr>
<td>4</td>
<td>1 0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>1 1 1/3 1/3 1/3 1/3 1/3 1/3</td>
</tr>
<tr>
<td>6</td>
<td>1 1 1 1 1 0 0 0</td>
</tr>
<tr>
<td>7</td>
<td>1 0 0 1 1 0 0 0</td>
</tr>
<tr>
<td>8</td>
<td>0 1 0 0 0 0 1 1</td>
</tr>
<tr>
<td>9</td>
<td>0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>10</td>
<td>0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>11</td>
<td>0 0 0 0 1 0 0 0</td>
</tr>
</tbody>
</table>
Notice that each row listing the vertices has *eight* coordinates, rather than seven. This is because our 7-dimensional space has been embedded as the plane $x_0 = 1$ in an 8-dimensional space in order to accommodate vertices and rays in a uniform manner. The rows with first coordinate equal to 1 are the vertices, while the others are the rays.

We immediately see that this matroid is *not* $\{0\}$-flowing because the point $(1/3, 1/3, 1/3, 1/3, 1/3, 1/3, 1/3)$ is a vertex of the associated polytope.

### 3. Seymour’s Conjecture

The fact that $AG(3, 2)$ is not 1-flowing has of course long been known. And because the property of being 1-flowing is closed under taking minors, this means that no binary matroid with an $AG(3, 2)$ minor is 1-flowing either.

There are *two other* known minor-minimal matroids with no $AG(3, 2)$-minor that are not 1-flowing; these are the dual pair $T_{11}$ and $T_{11}^*$. Seymour conjectured that this is the complete set of excluded minors.

3.1. **Conjecture.** *(Seymour)* A binary matroid is 1-flowing if and only if it has no $AG(3, 2)$, $T_{11}$ or $T_{11}^*$ minor.