**Petr Hliněný**

# MACEK:

## Practical computations with represented matroids

Dept. of Computer Science, FEI VŠB – TU Ostrava

17. listopadu 15, 708 33 Ostrava

and

Faculty of Informatics, Masaryk University in Brno

Botanická 68a, 602 00 Brno
Czech Republic

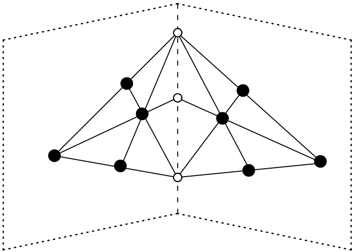
e-mail: hlineny@fi.muni.cz
http://www.fi.muni.cz/~hlineny

MACEK is now supported by Czech research grant GAČR 201/05/0050.

# 1 Matroids and MACEK

**Question:** What really are **matroids**?

- A common combinatorial generalization of graphs and finite geometries.
- A new look at (some) structural graph properties.



**Question:** What can matroids bring to us?

- Interesting objects to study (and difficult, indeed!).
- More general view some concepts brings interesting applications (e.g. the greedy algorithm, or recently the graph rank-width).

## 1.1 Definitions

A **matroid** $M$ on $E$ is a set system $\mathcal{B} \subseteq 2^E$ of *bases*, satisf. the exch. axiom

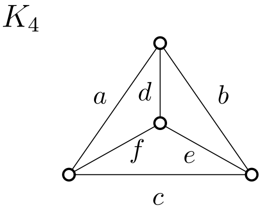
$$\forall B_1, B_2 \in \mathcal{B} \text{ a } \forall x \in B_1 - B_2,\ \exists y \in B_2 - B_1 :\ (B_1 - \{x\}) \cup \{y\} \in \mathcal{B}\,.$$

The subsets of bases are called *independent*.

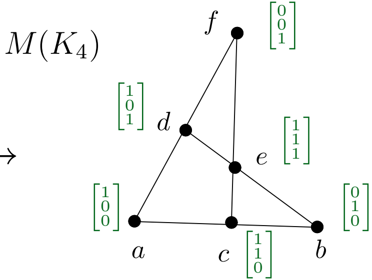
### Matroids coming from graphs and from vectors

**Cycle matroid of a graph** $M(G)$ – on the edges of $G$, where acyclic sets are independent.

**Vector matroid of a matrix** $M(\boldsymbol{A})$ – on the (column) vectors of $\boldsymbol{A}$, with usual linear independence.




$\longrightarrow$



## **Matrix representation** $\boldsymbol{A}$ of a matroid $M$ – the vector matroid

- Elements of $M$ are vectors over $\mathbb{F}$ – the columns of a matrix

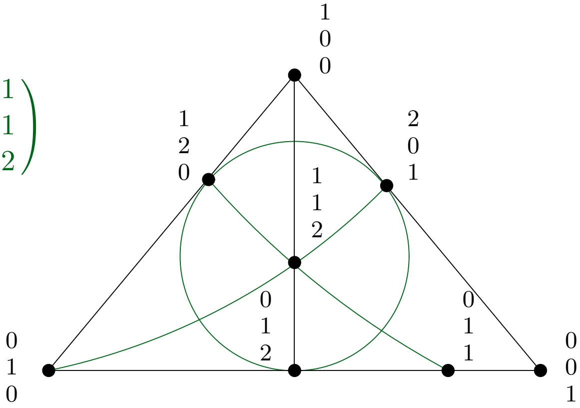
$$\boldsymbol{A} \in \mathbb{F}^{r \times n}.$$

- Matroid independence is usual linear independence.
- *Equivalence* of representations $\simeq$ row operations on matrices.

Not all matroids have matrix represent. over chosen $\mathbb{F}$, some even over no $\mathbb{F}$ at all.

An example – a matrix representation of a rank-$3$ matroid with $8$ elements over $GF(3)$:

$$\begin{pmatrix} 1 & 0 & 0 & & 1 & 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & & 0 & 1 & 2 & 1 & 2 \end{pmatrix}$$



## 1.2 Representing matroids in MACEK

**Matrix representation $\boldsymbol{A}' = [\boldsymbol{I} \,|\, \boldsymbol{A}]$** $\rightarrow$ the *reduced representation* $\boldsymbol{A}$ (stripping the unit submatrix).

$$\begin{pmatrix} 1 & 0 & 0 & & 1 & 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & & 0 & 1 & 2 & 1 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 & 2 \end{pmatrix}$$

- Now matroid elements label both the columns and rows of $\boldsymbol{A}$.
- The rows *display* a basis of $M(\boldsymbol{A})$.
- *Pivoting* changes to other bases...
- (Matrix equivalence now means a sequence of pivots and non-zero scalings.)

Normally, matrix representations in MACEK are **unlabeled**!
(Though some default labels are printed out for readability...)

## 1.3 Computing matroid properties

- Printing out thorough information about matroids:
  bases, flats, separations, connectivity, girth, automorphism group, representability over other fields, etc.

- Testing matroid properties (including batch-processing):
  minors, isomorphism, connectivity, representability, branch-width, etc.

- Some operations over a matroid:
  deletions/contractions of elements, pivoting, generating other representations of the same matroid, etc.

- A command-line user interface, very suitable for batch-jobs.

- Matroid generation. . .

# 2 Exhaustive Generation

A simple approach to combinatorial generation:

- Exhaustively construct all possible "presentations" of the objects.
- Then select one representative of each isomorphism class by means of an isomorphism tester.
- Slow, and problems with ineq. repres. giving different extensions...

The "*canonical construction path*" technique [McKay]:

- Select a small *base* object.
- Then, out of all ways how to construct our big object by single-element steps from the base object (*construction paths*), define the lexicographically smallest one (the *canonical* construction path).
- During generation, throw immediately away non-canonical extensions at each step.
- A big advantage – no explicit pairwise-isomorphism tests are necessary!

## 2.1 Canonically Generating Matroids

Actually, generating inequivalent matrix representations. . .

Matrix representation $\boldsymbol{A}' = [\boldsymbol{I} \mid \boldsymbol{A}]$ $\rightarrow$ reduced representation $\boldsymbol{A}$ (stripping the unit submatrix).

– Base object $\sim$ a submatrix (minor),

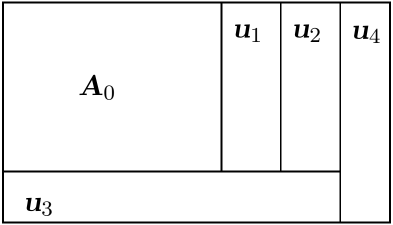
– construction path $\sim$ an *elimination sequence*

— in reverse order, stripping the excess rows and columns one by one,

– canonical ordering $\sim$ *lexic. order* on the excess vectors after unit-scaling,

– in a picture:

$\boldsymbol{A} =$




$\rightarrow$ an elimination sequence $S = (\, \boldsymbol{A}_0, \boldsymbol{A}, \; (1101)_2 \,)$.

**Algorithm 2.1.** *Recursive generation of (up to) $\ell$-step extensions of the matroid generated by a matrix $\boldsymbol{A}_0$ over $\mathbb{F}$.*

$S_0 = (\boldsymbol{A}_0, \boldsymbol{A}_0, \emptyset)$
`matroid-generate`$(S_0)$;

`procedure matroid-generate`$(S = (\boldsymbol{A}_0, \boldsymbol{A}, q))$
    `output` the matroid generated by $\boldsymbol{A}$;
    `if` length$(S) \geq \ell$ `then exit`;
    $s_0 =$ number of rows of $\boldsymbol{A}$; $s_1 =$ number of columns of $\boldsymbol{A}$;
    `for` $x \in \{0,1\}$, and $\vec{z} \in \mathbb{F}^{s_x}$ `do`
        $q_1 = (q, x)$;
        $\boldsymbol{A}_1 = \boldsymbol{A}$ with added $\vec{z}$ as the last row $(x = 0)$ or column $(x = 1)$;
        $S_1 = (\boldsymbol{A}_0, \boldsymbol{A}_1, q_1)$;
        `if` $\neg$ `unit-check`$(S_1)$ `then continue`;
        `if` $\neg$ `sequence-check`$(S_1)$ `then continue`;
        `if` $\neg$ `structure-check`$(S_1)$ `then continue`;
        `if` $\neg$ `canonical-check`$(S_1)$ `then continue`;
        `matroid-generate`$(S_1)$;
    `done`
`end.`

- `unit-check`: unit-scaling of the vectors.
- `sequence-check`: user-specified, like connectivity,etc.
- `structure-check`: user-specified, inherited to all minors.
- `canonical-check`:

**Algorithm 2.2.** *Testing canonical elimination sequence $S$ with base $\boldsymbol{A}_0$.*

`procedure canonical-check`$(S = (\boldsymbol{A}_0, \boldsymbol{A}, q)\,)$
    `for` $q' \leq_{\text{lexicographically}} q$, and all $\boldsymbol{A}'$ equivalent to $\boldsymbol{A}$
                such that $\boldsymbol{A}_0$ is a top-left submatrix of $\boldsymbol{A}'$ `do`
        $k =$ length$(S)$; $S' = (\boldsymbol{A}_0, \boldsymbol{A}', q')$;
        $S'_i =$ the $i$-th step subsequence of $S'$, $i = 1, 2, \dots, k$;
        `if` $\neg$`unit-check`$(S'_i)$, $i = 1, \dots, k$ `then continue`;
        `if` $\neg$`sequence-check`$(S'_i)$, $i = 1, \dots, k$ `then continue`;
        `if` $q' <_{\text{lexicographically}} q$, `or`
                $(\vec{u}'_1, \dots, \vec{u}'_k)$ of $S' <_{\text{lex.}} (\vec{u}_1, \dots, \vec{u}_k)$ of $S$ `then`
            `return false`;
    `done`
    `return true`;
`end.`

## 2.2 Using Generation in MACEK

- Generating all inequivalent (multi-step) extensions of a given matroid over a fixed finite field. (Easy to split for independent parallel generation.)
- Generation can internally maintain additional structural properties (simplicity, $3$-connectivity, excluded minors, etc).
- More tools are provided for involved filtering of generated extensions.

### How can MACEK help in research?

- Some computer–assisted proofs
  (e.g. [P. Hliněný, *On the Excluded Minors for Matroids of Branch-Width Three*, Electronic Journal of Combinatorics 9 (2002), #R32.])
- And a very easy generation of nasty counterexamples. . .
- Say, want to check whether $R_{10}$ is a splitter for the class of near-regular matroids? (Piece of cake. . . )

# 3 Matroid Enumeration Results

Enumeration of *binary combinatorial geometries* (i.e. simple binary matroids).

- Acketa [1984], by hand.
- Kingan, Kingan, Myrvold [2003], using computer and Oid.
- Our new computer generation [2005] with MACEK (* new entries):

| *rank**el.* | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | | 1 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | | | 1 | 3 | 4 | 5 | 6 | 5 | 4 | 3 | 2 | 1 |
| 5 | | | | 1 | 4 | 8 | 15 | 29 | 46 | 64 | 89 | * **112** |
| 6 | | | | | 1 | 5 | 14 | 38 | 105 | 273 | * **700** | * **1794** |
| 7 | | | | | | 1 | 6 | 22 | 80 | 312 | * **1285** | * **5632** |
| 8 | | | | | | | 1 | 7 | 32 | 151 | * **821** | * **5098** |
| 9 | | | | | | | | 1 | 8 | 44 | 266 | * **1948** |
| 10 | | | | | | | | | 1 | 9 | 59 | * **440** |
| 11 | | | | | | | | | | 1 | 10 | * **76** |
| 12 | | | | | | | | | | | 1 | 11 |
| 13 | | | | | | | | | | | | 1 |

The numbers of *labeled / unlabeled represented matroids* over small fields.

- The unlabeled case not studied so far to our knowledge.
- A really simple task for MACEK!

| *repr.* \ *matroid* | $U_{2,4}$ | $U_{2,5}$ | $U_{2,6}$ | $U_{3,6}$ | $\mathcal{W}^3$ | $U_{2,7}$ | $U_{3,7}$ |
|---|---|---|---|---|---|---|---|
| $GF(5)$ | 3/1 | 6/1 | 6/1 | 6/1 | 3/2 | 0/0 | 0/0 |
| $GF(7)$ | 5/2 | 20/1 | 60/1 | 140/3 | 5/3 | 120/1 | 120/1 |
| $GF(8)$ | 6/1 | 30/1 | 120/1 | 390/5 | 6/3 | 360/1 | 1200/2 |
| $GF(9)$ | 7/2 | 42/2 | 210/2 | 882/7 | 7/4 | 840/1 | 6120/4 |

The numbers of small *$3$-connected matroids* representable over small fields (generated all as unlabeled represented matroids).

- Computed [2003–4] with MACEK, but no such independent results exist to compare with (to our knowledge).

| *represent.* $\setminus$ *elem.* | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| regular: | 0 | 0 | 1 | 0 | 1 | 4 | 7 | 10 | 33 | 84 | 260 | 908 |
| $GF(2)$, non-reg: | 0 | 0 | 0 | 2 | 2 | 4 | 17 | 70 | 337 | 2080 | 16739 | 181834 |
| $GF(3)$, non-reg: | 1 | 0 | 1 | 6 | 23 | 120 | 1045 | 14116 | 330470 | ? | ? | ? |

(Next we present both the numbers of non-equivalent and of non-isomorphic ones.)

| *representable* $\setminus$ *elements* | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|
| $GF(4)$, non-$GF(2,3)$: | 0 | 2 | 2 | 8 | 78 | 1040 | 26494 | 1241588 |
| –non-isomorphic: | 0 | 2 | 2 | 8 | 69 | 748 | 15305 | ? |
| $GF(5)$, non-$GF(2,3,4)$: | 0 | 0 | 3 | 16 | 271 | 8336 | 497558 | ? |
| –non-isomorphic: | 0 | 0 | 3 | 12 | 192 | 6590 | ? | ? |
| $GF(7)$, non-$GF(2,-,5)$: | 0 | 0 | 0 | 18 | 1922 | 252438 | ? | ? |
| –non-isomorphic: | 0 | 0 | 0 | 10 | 277 | 97106 | ? | ? |
| $GF(8)$, non-$GF(2,-,7)$: | 0 | 0 | 0 | 0 | 94 | ? | ? | ? |
| –non-isomorphic: | 0 | 0 | 0 | 0 | 20 | ? | ? | ? |

# 4 Conclusions

**Want to try?** Go to

`http://www.cs.vsb.cz/hlineny/MACEK` ,

read the manual and find out whether MACEK is useful for you...

(Now with a new online interface - TRY IT yourself easily!)

**What about correctness?**

– Theoretical correctness of MACEK's algorithms.

– Debugging self-tests implemented in the program code.

– Some highly nontrivial self-reducing computations for comparism.

---

Anyway,

what is "MACEK" in Czech?