Acta Univ. M. Belii Math.
No. 12(2005), pp. 31–41

COMBINATORIAL GENERATION OF MATROID REPRESENTATIONS: THEORY AND PRACTICE

PETR HLINĚNÝ

ABSTRACT. Matroids (also called combinatorial geometries) present a strong combinatorial generalization of graphs and matrices. Unlike isomorph-free generation of graphs, which has been extensively studied both from theoretical and practical points of view, not much research has been done so far about matroid generation. Perhaps the main problem with matroid generation lies in a very complex internal structure of a matroid. That is why we focus on generation of suitable matroid representations, and we outline a way how to exhaustively generate matroid representations over finite fields in reasonable computing time. In particular, we extend here some enumeration results on binary (over the binary field) combinatorial geometries by Kingan et al. We use the matroid generation algorithm of [P. Hliněný, *Equivalence-Free Exhaustive Generation of Matroid Representations*] and its implementation in [MACEK; `http://www.mcs.vuw.ac.nz/research/macek`].

1. INTRODUCTION

Matroids, introduced by Whitney in 1935, present a common generalization of the concept of independency in graphs and matrices. We follow Oxley [10] in our matroid terminology, and we refer the reader to Section 2 for a brief outline of terms.

In graph theory, one often uses pictures to visualize particular graphs. On contrary, such visualization is very difficult in matroid theory – it is almost impossible to give a "nice drawing" of a general matroid in rank higher than 3. That fact makes matroid research more difficult, and brings a strong need for an automated matroid generation routine. (It is often such that proving a theorem in structural matroid theory requires one to check many small cases by hand. As matroid researchers know themselves, checking the "small cases" can be quite long and painful, and prone to errors.) However, great complexity and enormous numbers of (even small) matroids makes the task much harder than generation of, say, graphs.

A promising tractable approach is to generate matroid representations as matrices. Matroids represented by matrices over finite fields play an important role in structural matroid theory, similar to the role that graphs embedded on a surface play in structural graph theory. Hence exhaustive generation routines for

2000 *Mathematics Subject Classification.* Primary: 05A99; Secondary: 05B35, 68R05.
Key words and phrases. MACEK, matroid representation, exhaustive generation
Supported by the Institute of Theoretical Computer Science, project No. 1M0545, and by Czech research grant GAČR 201/05/0050.
An extended abstract to be published in Proceedings AACC 2005.
Submitted: October 14, 2005.

matroid representations over finite fields could have important applications in matroid research. One such a routine is based on the theoretical matroid generation algorithm of [6] (as outlined here in Section 4), which is implemented in our software package MACEK [3]. We have successfully used this tool in matroid structure theory research, namely for getting the excluded minors for matroids of branch-width three [4, 5] over small fields.

This paper (an extended abstract) brings, in addition to a brief overview of matroids and matroid generation, an outline of the Algorithm [6] from the implementation point of view (Algorithm 1). It is then used to obtain significant new results on enumeration of small simple binary matroids (binary combinatorial geometries) in Table 2 (extending the works of Acketa [1] and Kingan et al. [8]). Finally, we add a simple theoretical enumeration result about enumeration of binary matroids in Proposition 6.

2. Basics of Matroids

For clarity, we briefly review basic matroid terminology [10]. A *matroid* is a pair $M = (E, \mathcal{B})$ where $E = E(M)$ is the ground set of M (elements of M), and $\mathcal{B} \subseteq 2^E$ is a nonempty collection of *bases* of M. Moreover, matroid bases satisfy the "exchange axiom"; if $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 - B_2$, then there is $y \in B_2 - B_1$ such that $(B_1 - \{x\}) \cup \{y\} \in \mathcal{B}$. We consider only finite matroids. Subsets of bases are called *independent sets*, and the remaining sets are *dependent.* Minimal dependent sets are called *circuits.* All bases have the same cardinality called the *rank* $\mathrm{r}(M)$ of the matroid. The *rank function* $\mathrm{r}_M(X)$ in M is the maximal cardinality of an independent subset of a set $X \subseteq E(M)$.

If G is a (multi)graph, then its *cycle matroid* on the ground set $E(G)$ is denoted by $M(G)$. The independent sets of $M(G)$ are acyclic subsets (forests) in G, and the circuits of $M(G)$ are the cycles in G. In fact, a lot of matroid terminology is inherited from graphs. Another example of a matroid is a finite set of vectors with usual linear dependency. If $\boldsymbol{A}$ is a matrix, then the matroid M formed by the column vectors of $\boldsymbol{A}$ is called the *vector matroid* of $\boldsymbol{A}$, and $\boldsymbol{A}$ is called a *(matrix) representation* of M. (Not all matroids are vector matroids.) Two matrix representations are *(strongly) equivalent* if one can be obtained from the other by pivoting, non-zero scaling, and permuting rows or columns. (Note that, unlike Oxley [10], we do not allow for automorphisms of the underlying field here.) A matroid M is *representable* over a field $\mathbb{F}$ if M has a matrix representation over $\mathbb{F}$. Matroids representable over all fields are *regular*, those over $GF(2)$, $GF(3)$ are called *binary*, *ternary*, resp. See Figure 1.

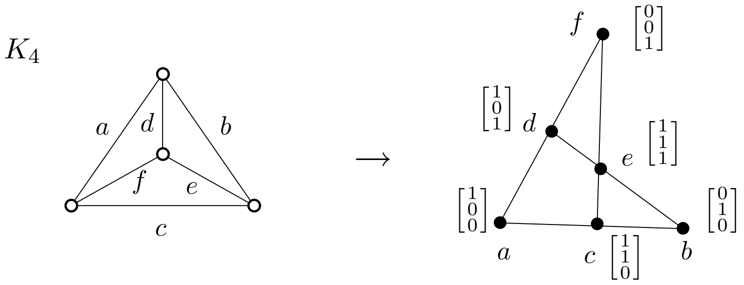
K_4
a
b
c
d
e
f
$\longrightarrow$
f $\begin{bmatrix}0\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\\1\end{bmatrix}$ d
e $\begin{bmatrix}1\\1\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\\0\end{bmatrix}$ a
c $\begin{bmatrix}1\\1\\0\end{bmatrix}$
b $\begin{bmatrix}0\\1\\0\end{bmatrix}$

FIGURE 1. An example of a vector representation of the cycle matroid $M(K_4)$. The matroid elements are depicted by dots, and their (linear) dependency is shown using lines.

A matroid M is *simple* if M contains no loops or parallel elements. The *dual* matroid M^* of M is defined on the same ground set E, and the bases of M^* are the set-complements of the bases of M. A *deletion* $M \setminus e$ of an element e simply removes e from the ground set of M. A *contraction* M/e is defined using duality $M/e = (M^* \setminus e)^*$. (This corresponds to contracting an edge in a graph.) Conversely, a matroid M' is a *one-element extension* (*coextension*) of M if $M = M' \setminus e$ ($M = M'/e$) for some e. A *minor* N of a matroid M is obtained by a sequence of deletions and contractions of elements (the order of which does not matter), and M is then an *extension* of N.

An important concept in structural matroid theory is *connectivity*, which is just slightly different from traditional graph connectivity. The *connectivity function* λ_M of a matroid M is defined for $A \subseteq E$ by $\lambda_M(A) = r_M(A) + r_M(E-A) - r(M) + 1$. A partition $(A, E-A)$ is called a *k-separation* if $\lambda_M(A) \leq k$ and both $|A|, |E-A| \geq k$. Geometrically the spans of the two sides of a k-separation intersect in a subspace of rank less than k, and in a graph view the two edge parts $A, E-A$ share $\leq k$ vertices provided they are both connected. A matroid M is *n-connected*, for $n > 1$, if it has no k-separation for $k = 1, 2, \ldots, n-1$, and $|E(M)| \geq 2n-2$. For instance, matroid connectivity is preserved under duality, and a 3-connected matroid has to be simple.

3. About Exhaustive Matroid Generation

Unlike for graphs or other common combinatorial objects, the history of matroid enumeration is rather short. We refer to [8] for a nice overview and bibliography. In particular, we mention here the old work of Acketa [1] on enumeration and construction of small binary matroids, which is closely related to further mentioned work [8] and to our contribution in Section 5.

Perhaps the main problem with matroid generation lies in a very complex internal structure of a matroid – a single matroid on n elements carries an amount of information exponential in n. That is why we are looking for generation of suitable matroid representations which would be easier to handle. Incidently, such are matrix representations of matroids over finite fields, which happen to be interesting and useful in matroid structure theory. We refer the works of Dharmatilake [2] and Kingan [7] as two examples of tasks of matroid-representations generation for the purpose of obtaining theoretical structural results.

Dharmatilake [2] used computer generation of binary matroid representations in order to find the binary excluded minors for matroids of branch-width three. He, in fact, found all 10 (3 new) of them, but his search was not finished. We have completed [4] the search exhaustively using our (faster) generation routine; and moreover, we have found [5] all the ternary excluded minors for branch-width three and generated many more such excluded minors over larger fields.

Among some recent works we cite Kingan et al. [8], and (unpublished) Pendavingh [11]. Kingan et al. present a generation algorithm for matroid representations, which we compare with our generation routine in Section 4, and we significantly extend their enumeration results in Section 5. On the other hand, Pendavingh systematically generates matroids in general (as set systems), which is computationally very hard. The point of our interest in his work is that some nontrivial outcomes of his search (e.g. the small excluded minors for representability over $GF(5)$ and $GF(7)$) agree with results we can obtain with our tools.

As in the above mentioned cases, it often turns out that 3-connected matroids are most useful when assisting theory research, as 3-connected matroids capture the interesting structural properties. Moreover, every matroid is easily decomposable into 3-connected components. A special feature of our generation routine is that it allows to generate exclusively 3-connected matroids, and thus significantly speeding up the process in many cases. For an illustration we present (see also [6]) our exhaustive enumeration results of small representable 3-connected matroids in Table 1. (It is hardly imaginable generating those amounts of matroids without the 3-connectivity restriction. To our knowledge, no similar large-scale enumeration of 3-connected representable matroids has been carried out till now.)

Table 1. The numbers of small 3-connected matroids represented over small fields.

representable \ *elements*	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-regular:	0	0	0	2	2	4	17	70	337	2080	16739	181834
$GF(3)$, non-regular:	1	0	1	6	23	120	1045	14116	330470	?	?	?

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

representable \ *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-$GF(2,3,4,5)$:	0	0	0	18	1922	252438	?	?
– non-isomorphic:	0	0	0	10	277	97106	?	?
$GF(8)$, non-$GF(2,3,4,5,7)$:	0	0	0	0	94	?	?	?
– non-isomorphic:	0	0	0	0	20	?	?	?

4. Matroid Extension Generation Algorithm

A simple approach to combinatorial generation of certain objects is the following: Exhaustively construct all possible "presentations" of the objects of given size (called [9] also labeled objects or labeled representations), and then select one representative of each isomorphism class by means of an isomorphism tester. Since there are typically many presentations for each one resulting object, and the procedure requires to isomorph-test each pair of the constructed presentations, that approach quickly becomes infeasible. Moreover, even worse specific problem – with *non-uniqueness*, arises when a matroid has several inequivalent representations (quite often over large fields), and not all of them extend further to representations of the extensions of this matroid (cf. [8]).

A better solution is provided by the technique of a "canonical construction path" which has been explicitly formulated by McKay [9], and used (explicitly or implicitly) in many combinatorial searches. Its idea is briefly described here: 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). Adapt the construction process so that all non-canonical extensions at each step are thrown away immediately (assuming that the canonical lexicographic order is hereditary). In this way every object is generated only once, and no explicit isomorphism tests are necessary. (Though the isomorphism problem may be implicitly contained in the definition of the canonical order.)

It appears that the canonical construction path technique has not been fully implemented before in matroid generation. The works mentioned in Section 3 usually use variations of the above simple generation method. Although Kingan et al. [8] claim to use the canonical construction path method, they implement it only partially — there is no definition of a canonical order, but an isomorphism tester is used to select the representatives of one-element matroid extensions.

The main theoretical contribution of our work is in a complete adaptation of the canonical construction path technique for exhaustive generation of matroid representations over finite fields. (We can, moreover, generate matroid representations over so called [12] "partial fields".) The adaptation is not at all obvious due to mentioned specific difficulties with matroid representations. Our base object is a common minor of all generated matroid representations, and one-element extensions and coextensions (see Section 2) play the role of single-element steps.

The following points summarize the important advantages of our approach:

- We actually generate all inequivalent matroid representations, which means that we naturally overcome the difficulties caused by non-uniqueness of a matroid representation over large fields.
- We can generate all matroids containing a given minor, by selecting the minor as the base object. This is often useful or even necessary when assisting matroid structure research.
- The construction path definition is quite flexible, and we can control various properties of it. Most importantly, we can generate exclusively 3-connected matroids by using Seymour's splitter theorem [13].
- We can easily distribute branches of a large computation among several (or many) independent computers.
- Finally, the implementation of our generation algorithm [3] appears to be faster than the other methods and programs mentioned in Section 3.

Elimination sequences of matroid representations. Before giving formal statements about our algorithm, we have to clarify some terms concerning matroid representations. As we already know, a (vector) representation of a matroid M is a matrix $\boldsymbol{A}$ such that the elements of M label the column vectors of $\boldsymbol{A}$, and matroid independence coincides with usual linear independence of the vectors. We always assume that the matrix $\boldsymbol{A}$ has full row rank, and so we can write $\boldsymbol{A} = [\boldsymbol{I} \,|\, \boldsymbol{A}']$ (*standard form*), where the columns of the identity matrix $\boldsymbol{I}$ are labeled by a basis B_I, which is said to be *displayed* by this representation $\boldsymbol{A}$. Naturally, we can now strip the identity matrix $\boldsymbol{I}$, and work with the *reduced representation* $\boldsymbol{A}'$ instead, where the rows of $\boldsymbol{A}'$ are labeled by the basis B_I and the columns by $E(M) - B_I$.

It is useful to notice that each regular square submatrix $\boldsymbol{D}$ of reduced $\boldsymbol{A}'$ is in a one-to-one correspondence with the basis $B_I \Delta E_D$ where $E_D \subseteq E(M)$ are the elements labeling the rows and columns of $\boldsymbol{D}$ in $\boldsymbol{A}'$. Every basis of M can be displayed by suitable pivoting in a reduced representation. Matroid duality corresponds to a transposition of a reduced representation, and contracting / deleting a matroid element corresponds to deleting a row / column from a reduced representation. So every minor of M can be shown as a submatrix in a suitable reduced representation of M.

Hence any $r \times c$ matrix $\boldsymbol{X}$ is a reduced representation of an $(r+c)$-element rank-r matroid. The definition of a (strong) matrix equivalence of matroid representations from Section 2 clearly applies in the same way also to reduced representations. In this sense an equivalence class of matrix representations over a field $\mathbb{F}$ shall be called an $\mathbb{F}$-*represented matroid*. Represented matroids refine the isomorphism classes of all $\mathbb{F}$-representable matroids, and it may easily happen over fields larger than $GF(3)$ that different represented matroids are isomorphic (non-uniqueness of representations). ¿From now on, when speaking about matroids, we implicitly *consider represented matroids* in the defined sense, and we treat them as unlabeled objects. We moreover say that a represented matroid N is a *represented minor* of M if some matrix of N is a submatrix of some matrix of M. (Notice that this is a stronger requirement than a usual minor.)

To understand the coming notion of an elimination sequence, imagine a matroid N that is a represented minor of M, and representations $\boldsymbol{A}_N$ and $\boldsymbol{A}_M$ of those such that $\boldsymbol{A}_N$ actually is a submatrix of $\boldsymbol{A}_M$. Then $\boldsymbol{A}_M$ can be reduced to $\boldsymbol{A}_N$ in steps by removing single columns or rows. Conversely, we can generate represented extensions of N in steps by adding single columns or rows to the base matrix $\boldsymbol{A}_0$. (By allowing both one-element extensions and co-extensions, we add more flexibility to our generation routine, compared to [8]. The added flexibility is necessary, for example, for handling generation of 3-connected matroids.)

Definition. *Let $\boldsymbol{A}$, $\boldsymbol{A}_0$ be matrices over $\mathbb{F}$ of dimensions $r \times c$, $r_0 \times c_0$, respectively. A triple $S = (\boldsymbol{A}_0, \boldsymbol{A}, q)$ is called an* elimination sequence *of length k if the following are true: (See an illustration in Figure 2.)*

- *$\boldsymbol{A}_0$ (called a* base*) is a top-left submatrix of $\boldsymbol{A}$, and $k = r + c - r_0 - c_0$.*
- *q is a $\{0,1\}$-sequence of length k containing $r - r_0$ zeros. Moreover, there is a sequence of matrices $\boldsymbol{A}_1, \dots, \boldsymbol{A}_k = \boldsymbol{A}$ such that $\boldsymbol{A}_i$, $1 \le i \le k$ is obtained from $\boldsymbol{A}_{i-1}$ by adding one row ($q_i = 0$) or one column ($q_i = 1$) vector.*
- *Denote by $\vec{u}_i$ the vector that extends $\boldsymbol{A}_{i-1}$ into $\boldsymbol{A}_i$, $1 \le i \le k$. Then each $\vec{u}_i$ is in a "unit form" with respect to $\boldsymbol{A}_{i-1}$. (If we generate only connected matroids, that simply means the first non-zero entry is scaled to 1 for each row or column in $\boldsymbol{A}$ not intersecting $\boldsymbol{A}_0$. Obviously, non-zero scaling does not change the generated matroid.)*

A column vector $\vec{x}$ is in a unit form *with respect to a matrix $\boldsymbol{Y}$ if the following holds: Two rows of $\boldsymbol{Y}$ are said to be connected if they both have nonzero entries in a common column of $\boldsymbol{Y}$. In the vector $\vec{x}$, the first nonzero entry of every connected components of rows with respect to $\boldsymbol{Y}$ must be 1. The unit form of a row vector is defined analogously.*

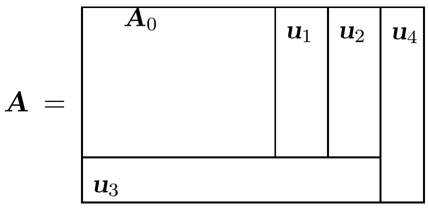

FIGURE 2. An illustration of an elimination sequence $(\boldsymbol{A}_0, \boldsymbol{A}, q)$ where $q = (1, 1, 0, 1)$.

Definition. *Let $S = (\boldsymbol{A}_0, \boldsymbol{A}, q)$ and $S' = (\boldsymbol{A}_0, \boldsymbol{A}', q')$ be two elimination sequences of length k and the same base $\boldsymbol{A}_0$. Then $S \prec S'$ (is lexicographically smaller than) if q is lexicographically smaller than q', or $q = q'$ and the sequence of vectors $(\vec{u}_1, \ldots, \vec{u}_k)$ is lexicographically smaller than the sequence $(\vec{u}'_1, \ldots, \vec{u}'_k)$.*

An elimination sequence $S = (\boldsymbol{A}_0, \boldsymbol{A}, q)$ is said to *produce* the matroid represented by $\boldsymbol{A}$. Out of all elimination sequences S with base $\boldsymbol{A}_0$ producing a represented matroid M, the unique lexicographically minimal one is called the *canonical elimination sequence* of M with respect to $\boldsymbol{A}_0$ and $\mathbb{F}$. The main contribution of our full theoretical paper [6] is in a detailed proof of the following basic generation theorem (and its corollaries).

Theorem. *Let $\boldsymbol{A}_0$ be a matrix over a finite field $\mathbb{F}$. Algorithm 1 outputs the canonical elimination sequence S with base $\boldsymbol{A}_0$ producing M; for every $\mathbb{F}$-represented matroid M of given size that can be produced by some elimination sequence with base $\boldsymbol{A}_0$ (subject to further user-defined restrictions on the sequences).*

There are, roughly saying, two kinds of "user-defined" restrictions that can be applied to generated sequences. The first kind – structural restrictions, control chosen (hereditary) properties of the matroid represented by the resulting matrix $\boldsymbol{A}$ of S. Such is, for example, a restriction excluding certain minors in generated matroids. The second kind – sequential restrictions, control the proper order in which matrix lines can be added to produce the resulting matrix $\boldsymbol{A}$. Such is, say, a matroid connectivity restriction at each extension step $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k = \boldsymbol{A}$ of S. For instance, the promised exhaustive generation of 3-connected matroids can be handled in single-element steps (with a small exception of wheels and whirls as the base minors) using Seymour's Splitter Theorem [13]:

Theorem. (Seymour) *Let M, N be 3-connected matroids such that N is a minor of M. Suppose that if N is a wheel (a whirl), then M has no larger wheel (no larger whirl) as a minor. Then there is a 3-connected matroid N_1 such that $|E(N_1)| = |E(N)| + 1$, and that M has an N_1-minor.*

Extensions generation scheme. In this paper, we outline the theoretical algorithm of Theorem 4 [6] from an application point of view, precisely as it is implemented in Macek [3].

Algorithm 1. *Recursive generation of (up to) ℓ-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$\big(S = (\boldsymbol{A}_0, \boldsymbol{A}, q)\big)$
    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 ¬ canonical-check($S_1$) then continue;
        matroid-generate($S_1$);
    done
end.
```

The procedure `unit-check` tests, whether the last-added vector $\vec{z}$ is in the unit form with respect to $\boldsymbol{A}$. (Actually, to save computing time, we directly generate vectors $\vec{z}$ of the cycle in the unit form. An analogous note applies also to further Algorithm 2.)

The procedures `sequence-check` and `structure-check` realize the above mentioned user-defined restrictions on elimination sequences. Two rules have to be observed to make the scheme work properly and efficiently: The properties tested in `structure-check` should inherit to all minors of the matroid represented by $\boldsymbol{A}$, and the tests in `sequence-check` should be computationally fast.

Finally, the most complex procedure `canonical-check` determines (by brute force) whether the given sequence S is canonical. It is implemented, using the same(!) `unit-check` and `sequence-check` procedures as Algorithm 1, as follows:

Algorithm 2. *Testing canonical elimination sequence S with base $\boldsymbol{A}_0$.*

```
procedure canonical-check$\big(S = (\boldsymbol{A}_0, \boldsymbol{A}, q)\big)$
    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
        $k = \text{length}(S)$; $S' = (\boldsymbol{A}_0, \boldsymbol{A}', q')$;
        $S'_i =$ the $i$-th step subsequence of $S'$, $i = 1, 2, \ldots, k$;
        if ¬ unit-check($S'_i$), $i = 1, \ldots, k$  then continue;
        if ¬ sequence-check($S'_i$), $i = 1, \ldots, k$  then continue;
        if  $q' <_{\text{lexicographically}} q$, or
                    $(\vec{u}'_1, \ldots, \vec{u}'_k)$ of $S'$ $<_{\text{lex.}}$ $(\vec{u}_1, \ldots, \vec{u}_k)$ of $S$  then
            return false;
    done
    return true;
end.
```

5. Practical Computations

As already noted above, we have used Algorithm's 1 implementation in Macek [3] for carrying out several successful matroid generation projects. (It is worth to mention here that Macek implements also many other structural computations over matroids, like minor, connectivity, or isomorphism tests, generation of representations of a fixed matroid over other fields, etc.) Mainly we have focused on generation of 3-connected represented matroids. The success of it is based on the following corollary of Theorems 4 and 4:

Corrolary. *Assume the procedure* `sequence-check` *implements a 3-connectivity test for each extension step of the sequence, and that $\boldsymbol{A}_0$ over $\mathbb{F}$ represents a 3-connected matroid that is not a wheel or a whirl. Then Algorithm 1 outputs all the 3-connected $\mathbb{F}$-represented matroids of given size having the matroid of $\boldsymbol{A}_0$ as a represented minor, without repetition.*

Using suitable structural results on minors in representable matroids, one may exhaustively generate 3-connected matroids using this corollary. In this way we have proved the complete list of binary excluded minors for the matroids of branch-width three [4] (extending Dharmatilake [2]), and found all such ternary excluded minors [5]. Similarly we compute [6] the enumeration results on 3-connected representable matroids summarized in Table 1.

Our MACEK package is available for free under the GPL license. Moreover, we have added a new trial online interface to MACEK accessible from `http://www.cs.vsb.cz/hlineny/MACEK`. This java-applet based interface allows everybody to run small matroid computations without necessity to install the whole package. Many motivation examples are presented in the online documentation. Two such examples are described next.

The matroid called R_{10} is well known for being a splitter for the class of regular matroids. (A *splitter* has no 3-connected extension or coextension in its class.) Using MACEK, one can easily prove that R_{10} is a splitter also for the class of all near-regular matroids (those representable over all fields larger than $GF(2)$, at least). Running MACEK (through the online interface) with arguments `'near-reg'` `'!extend'` `'R10'` shows:

```
      MACEK 1.2.01 (26/06/05) starting...
  ~126~   Generated 0 non-equiv 3-conn row co-extensions of the sequence [R10] (5x5|5x5).
  ~126~   Generated 0 non-equiv 3-conn column extensions of the sequence [R10] (5x5|5x5).
```

One can also get all inequivalent representations of a matroid over a field. Running with `'GF(7)'` `'!verbose;!represgen (S) all'` `'U36 U37'` shows:

```
      MACEK 1.2.01 (26/06/05) starting...
  ~675~   There are 140 GF(7)-representations of #1 matroid [U36] (3x3, GF(7)).
  ~676~   There are 120 GF(7)-representations of #2 matroid [U37] (3x4, GF(7)).
```

At last, the reader may naturally ask how the results of our computations compare to others. Unfortunately, there are not many established results on matroid enumeration so far. Out of a few published ones, we focus in this work on the enumeration of small simple binary matroids, started by Acketa [1] by hand, and continued by Kingan et al. [8] using their matroid computing package. This part brings the main new computational results of our paper.

Noticing that a simple rank-r matroid M on more than r elements always contains a circuit C of length at least 3, and that deleting any element e not in C preserves simplicity of $M \setminus e$, we obtain the following conclusion of Theorem 4:

Corrolary. *Assume the procedure* `sequence-check` *implements just a simplicity test for each extension step of the sequence, and that* $\boldsymbol{A}_0 = \binom{1}{1}$ *over* $GF(2)$*. Then Algorithm 1 outputs all the simple binary matroids of given size* $n \leq \ell + 3$ *and rank* $r < n$*, without repetition.*

The results we have obtained using the corollary and MACEK are summarized in Table 2. We have independently verified all the enumeration results of Kingan et al. [8] on up to 11 elements, and moreover we have added significant new results for matroids on 12 and 13 elements. For instance, the number of 8-element rank-5 simple binary matroids can be obtained by running MACEK with `'GF(2)'` `'@ext-simple;!extendsize 5 3'` `'1;1'`.

```
      MACEK 1.2.09 (26/08/05) starting...
                ......
  ~652~   In total 15 (co-)extensions of 1 matrix-sequence generated for "size" over GF(2).
```

(One needs the latest version $\geq$ 1.2.09 of MACEK to run this.) Actually, a faster way to generate all such rank-5 matroids on up to 9 elements is using `'GF(2)'` `'@ext-simple;!extend rrrcc'` `'1;1'`. Try this yourself.

For interested readers, we add a short summary of real running times of our enumerations (after some tuning-up). All the results on at most 11 elements are

TABLE 2. An enumeration of small simple binary matroids – binary combinatorial geometries. (The *-marked numbers are the new contributions.)

rank \ *elements*	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

quite easy, they take just seconds or at most minutes to finish. For 12 elements, the computation took approximately 3 hours (rank 6), 13 hours (rank 7), and 16 hours (rank 9); for 13 elements it took 11 hours (rank 6), 110 hours (rank 7), 260 hours (rank 8), 300 hours (rank 9), and 140 hours (rank 10). (Computing times are normalized to 1GHz CPU.)

6. CONCLUSIONS

We have presented a matroid extension generation scheme which fully implements the idea of a generation via a canonical construction path by McKay. Compared with other known matroid generation programs, the implementation of our algorithm appears to be remarkably faster and more powerful in generation of matroids representable over finite fields. We have already used it to prove some results of theoretical interest, and significantly extended known enumeration results of small matroids. In current work we are extending the enumeration in Table 2 to ternary and other representable matroids. (For instance, we have so far verified the ternary results of [8].)

A simple observation contained in [8] reads: There is one r-element rank-r simple binary matroid, and $r-1$ of $r+1$-element rank-r simple binary matroids. Partially inspired by our enumeration results from Table 2, we have derived the following claim. (One could actually obtain a closed formula for this sum in a straightforward way, but that would be a complicated and boring expression.)

Proposition. *The number of $r+2$-element rank-r simple binary matroids is given by*

$$\sum_{i=3}^{r} \sum_{j=0}^{\lfloor i/2 \rfloor} \max\left\{0, r-2i+j+3\right\}.$$

Sketch of proof. Using standard arguments about binary matroids, it is easy to argue that an $r+2$-element rank-r simple binary matroid M can be decomposed into $E(M) = C_1 \cup C_2 \cup F_0$ such that C_1, C_2, and $C_1 \Delta C_2$ provided they intersect, are all circuits; and that F_0 are coloops – elements contained in no dependent set. (Actually, such matroids are always cycle matroids of simple graphs.) Now consider C_1 the shortest circuit in M, and denote $i = |C_1| \geq 3$, $j = |C_1 \cap C_2| \geq 0$. To avoid symmetric duplicate cases, assume $j \leq \lfloor i/2 \rfloor$. Then the length of C_2 is at least $|C_1| = i$ and at most $|E(M)| - |C_1 - C_2| = r + 2 + j - i$; and so, fixing valid values of i, j, there are $r + 3 + j - 2i$ choices for the circuit C_2. The above formula follows in a straightforward way. ∎

Acknowledgments

The large-scale computations summarized in Tables 1 and 2 have been run on the `minos` computing cluster at West Bohemia University (ITI center).

References

[1] D.M. Acketa, *A construction of non-simple matroids on at most 8 elements*, J. Combin. Inform. System Sci. 9 (1984), 121–132.

[2] J.S. Dharmatilake, *Binary Matroids of Branch-Width* 3, PhD. dissertation, Ohio State University, 1994.

[3] P. Hliněný, *The* Macek *Program*, `http://www.mcs.vuw.ac.nz/research/macek`, `http://www.cs.vsb.cz/hlineny/MACEK`, 2002–2005, version 1.2.09.

[4] P. Hliněný, *On the Excluded Minors for Matroids of Branch-Width Three*, Electronic Journal of Combinatorics 9 (2002), #R32.

[5] P. Hliněný, *Using a Computer in Matroid Theory Research*, Acta Math. Univ. M. Belii 11 (2004), 27–44.

[6] P. Hliněný, *Equivalence-Free Exhaustive Generation of Matroid Representations*, Discrete Appl. Math., accepted (2005).

[7] S.R. Kingan, Matroid Structure, Ph.D. dissertation, 1994, Louisiana State University.

[8] R.J. Kingan, S.R. Kingan, W. Myrvold, *On matroid generation*, Proceedings of the Thirty-Fourth Southeastern International Conference on Combinatorics, Graph Theory, and Computing, Congressus Numerantium 164 (2003), 95–109.

[9] B.D. McKay, *Isomorph-free exhaustive generation*, J. Algorithms 26 (1998), 306–324.

[10] J.G. Oxley, Matroid Theory, Oxford University Press 1992.

[11] R. Pendavingh, personal communication, 2004.

[12] C. Semple, G.P. Whittle, *Partial Fields and Matroid Representation*, Advances in Appl. Math. 17 (1996), 184–208.

[13] P.D. Seymour, *Decomposition of Regular Matroids*, J. Combin. Theory Ser. B 28 (1980), 305–359.

Faculty of Informatics,, Masaryk University in Brno,, Botanická 68a, 602 00 Brno, Czech Rep., Department of Computer Science,,, VŠB – Technical University Ostrava, `hlineny@fi.muni.cz`