COMPUTATIONAL PROOF OF ROTA'S BASIS CONJECTURE FOR MATROIDS OF RANK 4

MICHAEL SZE-YU CHEUNG

Abstract. We prove Rota's basis conjecture for matroids of rank $n \leq 4$ using a computer program. The program starts with n arbitrary bases and uses basis exchange properties to infer new bases until it has n disjoint rainbow bases (solving the conjecture) or no more bases can be inferred (providing a counter-example). $n = 3$ is solved in one case whereas $n = 4$ is solved in around a hundred thousand cases.

1. Introduction

Given n disjoint bases in an n-dimensional vector space, we wish to show that there exist n pairwise disjoint transversals of these bases that are also bases. If we think of the original n bases as each having a color, then we are trying to show that there exists n disjoint "rainbow" bases. To prove this, we know that given two bases, there are theorems that imply the existence of other bases. Hence, we apply these theorems to the n given bases to imply the existence of other bases. Now we apply these theorems to the n given bases along with these other bases to imply the existence of even more bases. This is continued until we have implied the existence of n disjoint "rainbow" bases, or we cannot imply any more bases (in which case the conjecture is false). Due to the many steps involved and cases that must be considered, we use a computer to perform this task.

The concept of independence in a vector space can be generalized into a mathematical structure known as a matroid. This conjecture generalizes to matroids, and all of our techniques for proving it do as well. Hence, we use our algorithm to try and prove or disprove this strengthened conjecture.

Definition 1.1 (matroid)**.** A **matroid** is an ordered pair $M = (S, \mathcal{B})$, where S is a finite set (called the **ground set**) and $\mathcal{B}$ is a collection of subsets of S (called the **bases** of M), that satisfies the following

Date: August 6, 2012.
This research was supported by NSF grant DMS-1004516.
I would like to thank my REU advisors Dr. Joshua Ducey and Dr. Minah Oh.

properties:
M1: $\mathcal{B}$ is nonempty.
M2: Given any two bases B_1, B_2 and an element x of the first basis, we can find some element y of the second basis such that $(B_1 - \{x\}) \cup \{y\}$ is a basis.

Example 1.2 (matroid from vectors in a vector space)**.** The following four column vectors (labeled $1, 2, 3, 4$)

$$\begin{array}{rcccc} \text{elements:} & 1 & 2 & 3 & 4 \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

yield the following matroid: $S = \{1, 2, 3, 4\}, \mathcal{B} = \{\{1, 2, 3\}, \{1, 3, 4\}, \{2, 3, 4\}\}$.

Note that not all matroids arise from vector spaces. Theorems that allow us to infer bases in a matroid from known bases are of interest to us; we shall use the following lemma from White [5].

Lemma 1.3 (White's lemma)**.** *Given any two bases D, E and an ordered partition $\mathscr{O}(D) = (D_1, \ldots, D_k)$ of D, we can find an ordered partition $\mathscr{O}(E) = (E_1, \ldots, E_k)$ of E so that $\forall i \in \{1, \ldots, k\}, (D - D_i) \cup E_i$ is a basis. Note that this requires that $|D_i| = |E_i|$ for all i.*

We will simply refer to "ordered partition" by "partition" if there is no ambiguity. Furthermore, we shall refer to each element of a partition as a "block." In 1.3, if we choose the first block to be $D_1 = D \cap E$, then clearly $E_1 = D \cap E = D_1$. Furthermore, any block containing $D \cap E$ can split into $D \cap E$ and another block contained in $D - E$, so 1.3 is equivalent to the following theorem, where we replace $\mathscr{O}(D), \mathscr{O}(E)$ with $\mathscr{O}(D - E), \mathscr{O}(E - D)$.

Theorem 1.4 (partition basis exchange)**.** *Given any two bases D, E and an ordered partition $\mathscr{O}(D - E) = (D_1, \ldots, D_k)$ of $D - E$, we can find an ordered partition $\mathscr{O}(E - D) = (E_1, \ldots, E_k)$ of $E - D$ so that $\forall i \in \{1, \ldots, k\}, (D - D_i) \cup E_i$ is a basis.*

Notice that we can choose two bases D, E and a partition $\mathscr{O}(D-E)$, but the corresponding partition $\mathscr{O}(E-D)$ is unknown. Hence, we refer to this choice $(D, E, \mathscr{O}(D-E))$ as a "3-tuple," and a partition $\mathscr{O}(E-D)$ as a "candidate." Below is an example showing why 1.3 and 1.4 are equivalent.

Example 1.5 (partition basis exchange 1)**.** If $D = \{a_1, a_2, a_3\}$ and $E = \{a_1, b_2, b_3\}$, then any subset of D containing a_1 must be swapped for a subset of E containing a_1. Suppose not, that is, suppose some other subset of D (a_2 or a_3) must be swapped for a_1. If we swap say a_2 for a_1 in E, then we have that $(D - \{a_2\}) \cup \{a_1\} = \{a_1, a_1, a_3\}$ is a basis, a contradiction.

Below is an example applying 1.4

Example 1.6 (partition basis exchange 2)**.** If we are given the 3-tuple $D = \{a_1, a_2, a_3, a_4\}, E = \{b_1, b_2, b_3, b_4\}, \mathscr{O}(D-E) = (\{a_1, a_2\}, \{a_3\}, \{a_4\})$, then possible candidates for $\mathscr{O}(E-D)$ are all partitions with one block of order two followed by two blocks of order one:
$(\{b_1, b_2\}, \{b_3\}, \{b_4\})$
$(\{b_1, b_2\}, \{b_4\}, \{b_3\})$
$(\{b_1, b_3\}, \{b_2\}, \{b_4\})$
$(\{b_1, b_3\}, \{b_4\}, \{b_2\})$
$(\{b_1, b_4\}, \{b_2\}, \{b_3\})$
$(\{b_1, b_4\}, \{b_3\}, \{b_2\})$
$(\{b_2, b_3\}, \{b_1\}, \{b_4\})$
$(\{b_2, b_3\}, \{b_4\}, \{b_1\})$
$(\{b_2, b_4\}, \{b_1\}, \{b_3\})$
$(\{b_2, b_4\}, \{b_3\}, \{b_1\})$
$(\{b_3, b_4\}, \{b_1\}, \{b_2\})$
$(\{b_3, b_4\}, \{b_2\}, \{b_1\})$

We then have twelve cases, one for each $\mathscr{O}(E - D)$, each implying the existence of three bases. Case 1 ($\mathscr{O}(E-D) = (\{b_1, b_2\}, \{b_3\}, \{b_4\})$) implies the existence of these bases by 1.4:
$\{b_1, b_2, a_3, a_4\}$
$\{a_1, a_2, b_3, a_4\}$
$\{a_1, a_2, a_3, b_4\}$

Notice that if the partition of $D - E$ consists of one-element blocks of $D - E$, then we have the following corollary:

Corollary 1.7 (bijective basis exchange)**.** *Given any two bases D, E, there is a bijective function $f : D - E \to E - D$ such that $\forall x \in D - E, (D - \{x\}) \cup \{f(x)\}$ is a basis.*

Notice further that if the partition of $D-E$ contains only two blocks $\{D_1, D-E-D_1\}$, then the corresponding partition of $E-D$ contains two blocks $\{E_1, E-D-E_1\}$, so 1.4 implies the existence of two bases,

$(D-D_1)\cup E_1$ and $(D-(D-E-D_1))\cup(E-D-E_1)=D_1\cup(E-E_1)$. This yields the following corollary:

Corollary 1.8 (subset basis exchange). *Given any two bases B_1, B_2 and a subset A_1 of B_1-B_2, there exists a subset A_2 of B_2-B_1 such that $(B_1-A_1)\cup A_2$ and $(B_2-A_2)\cup A_1$ are bases.*

Example 1.9 (subset basis exchange). For $n=3$, we are given bases $D_1=\{a_1,a_2,a_3\}, D_2=\{b_1,b_2,b_3\}, D_3=\{c_1,c_2,c_3\}$. If we choose $B_1=D_1, B_2=D_2, A_1=\{a_1,a_2\}$, A_2 could be either $\{b_1,b_2\},\{b_1,b_3\}$, or $\{b_2,b_3\}$. Hence, we must consider three cases - one for each possible A_2. Case 1 ($A_2=\{b_1,b_2\}$) provides two new bases: $D_4=\{a_3,b_1,b_2\}, D_5=\{a_1,a_2,b_3\}$. Now if we choose $B_1=D_5, B_2=D_3, A_1=\{b_3\}$, then A_2 could be either $\{c_1\},\{c_2\}$, or $\{c_3\}$. Hence, we have another three cases (Case 1-1, 1-2, 1-3).

In 1989, Rota made a conjecture about another basis exchange property of matroids.

Conjecture 1.10 (Rota's basis conjecture). *Let M be a matroid of rank n and let $\mathcal{B}^*=\{B_1,\ldots,B_n\}$ be a collection of n disjoint bases in M. Then there exists n pairwise disjoint transversals of $\mathcal{B}^*$ that are bases.*

If we think of each of the n given bases as having a color, we can call these pairwise disjoint transversals "rainbow" bases. Furthermore, we can define the rainbowness of a basis in this matroid as being the number of bases in $\mathcal{B}^*$ that it has a non-empty intersection with. For example, all bases in $\mathcal{B}^*$ have rainbowness 1 and all rainbow bases have rainbowness n.

2. Algorithm

The n bases given by 1.10 do not fully describe the matroid, since 1.4 implies the existence of other bases. In fact, we can "complete" the matroid by inferring new bases until no new bases can be inferred. At this point, our collection of known bases fully describes a matroid, since this collection along with the n^2 elements from the bases in $\mathcal{B}^*$ satisfies *M1* and 1.4, which implies *M2*. While in the process of completing this matroid, if we ever have n disjoint rainbow bases, we have proven the conjecture and don't need to finish completing the matroid. On the other hand, if we have completed the matroid yet there is no set of n disjoint rainbow bases in it, then this matroid is a counter-example to 1.10.

Specifically, we start with the given n bases (assuming nothing about these bases besides the fact that they satisfy the axioms of a matroid) and proceed in a series of steps until we have proven the existence of n disjoint rainbow bases (or arrived at a counter-example). At each step, we choose a 3-tuple $(D, E, \mathscr{O}(D-E))$ and apply 1.4 to prove the existence of new bases. We will need to consider a separate case for each valid $\mathscr{O}(E-D)$. It is preferable to prove 1.10 in as few cases as possible, so we first look for 3-tuples where we can eliminate all but one candidate $\mathscr{O}(E-D)$. This way, we can prove the existence of new bases without having to branch off into more cases. If no such 3-tuple exists, we must find the "best" 3-tuple that will branch off into the least number of cases and prove 1.10 in as few steps as possible. To do this, we compute a score for each 3-tuple based on the number of cases that must be considered (one case for each candidate $\mathscr{O}(E-D)$) and the quality of the new bases resulting from each case.

2.1. **3-Tuples.** Consider the 3-tuple $(D, E, \mathscr{O}(D-E))$ again. If there is some $\mathscr{O}(E-D)$ such that applying 1.4 will result in no new bases, then skip this 3-tuple.

Scores for 3-tuples are computed by finding the case with the lowest case score and dividing that score by $\ln|C|$ where $|C|$ is the number of cases. Case scores are computed by adding together the basis scores for each new basis corresponding to that case. In computing the basis scores, we require a fixed list of "element score values" and "subscore factors." We try to choose values and factors that minimize the number of cases that must be considered, though this is largely experimental and by no means rigorous. A new basis score is computed as follows.

(1) Group the elements in the new basis by which starting basis in $\mathcal{B}^*$ the element belongs to. Each group might contain anywhere from 0 to n elements.
(2) For each existing rainbow basis, compute a subscore as follows:
    (a) We first compute an element score for each of the elements of the rainbow basis.
    (b) Associate each element in the rainbow basis to a group in the new basis based on which starting basis in $\mathcal{B}^*$ the element of the rainbow basis belongs to.
    (c) For each element of the rainbow basis, find the number of elements in its associated group that are not equal to it. Call this the disjointness of this element of the rainbow

basis. The element score corresponding to each element of the rainbow basis is a value determined by the disjointness of that element. For instance, disjointness $\{0, 1, 2, 3\}$ could correspond to element scores of $\{0, 1, 3, 10\}$. Call this list $\{0, 1, 3, 10\}$ the "element score values." For example, if our rainbow basis contains an a_3 and its corresponding group is $\{a_3, a_4, a_5\}$, then since a_4, a_5 are different from a_3, the element a_3 of the rainbow basis has disjointness 2. This would yield an element score of 3. Once again, the element scores are chosen by the user, more or less experimentally. However, we should require that the element score values are strictly increasing, since we would like to award a higher score to (i.e. prefer) bases that are more disjoint from currently known rainbow bases.

(d) Average the nonzero element scores of each element of the rainbow basis. Multiply this number by a factor determined by the number of nonzero subscores. For instance, the factors corresponding to $\{0, 1, 2, 3\}$ nonzero subscores could be $\{0, 1, 2, 4\}$. Call this list of factors the "subscore factors." For instance, given element scores $\{0, 5, 0, 3, 4\}$, the average of the nonzero element scores is 4, and since there are 3 nonzero element scores, we multiply it by the subscore factor 4 for a subscore of 16. These subscore factors are also chosen experimentally, but once again we should require that the subscore factors are strictly increasing.

(3) The basis score is the sum of all subscores divided by the number of rainbow bases. We divide to normalize the score across different steps of the algorithm, since earlier steps have less rainbow bases whereas later steps have more. This normalization is actually not necessary in the algorithm since within a particular step of a particular case, the number of rainbow bases will be the same. However, it is useful for gathering statistics and theoretical discussion.

Example 2.1 (subscore calculation)**.** Let the element score values be {0,2,3,4,5} and the subscore factors {0,1,3,7,15,31} Suppose our starting bases are $\{a_1, a_2, a_3, a_4\}, \{b_1, b_2, b_3, b_4\}, \{c_1, c_2, c_3, c_4\}, \{d_1, d_2, d_3, d_4\}$ and our new basis is $\{a_1, a_2, b_2, d_4\}$. Then our four groups are $\{a_1, a_2\}, \{b_2\}, \emptyset, \{d_4\}$. To calculate the subscore for the existing rainbow basis $\{a_3, b_2, c_2, d_2\}$, we compute the element scores of each element in the rainbow basis (Figure 1). From this, the average of the nonzero subscores is $\frac{3+2}{2} = 2.5$.

Group	$\{a_1, a_2\}$	$\{b_2\}$	$\emptyset$	$\{d_4\}$
Element	a_3	b_2	c_2	d_2
Disjoint	$\{a_1, a_2\}$	$\emptyset$	$\emptyset$	$\{d_4\}$
Disjoint#	2	0	0	1
Subsubscore	3	0	0	2

FIGURE 1. Table illustrating the calculation of a subscore. "Element" refers an element in the rainbow basis. "Disjoint" refers to the bases in the group that are disjoint from the corresponding element.

Since we have two nonzero subscores, we multiply by the corresponding subscore factor of 3, for a final subscore of 7.5.

This score is designed to hopefully choose the optimal 3-tuple each time to reach n disjoint rainbow bases as quickly as possible. Of course, this score does not rigorously determine which 3-tuple is the best, but we believe it to be a good tradeoff between time spent computing the score and time saved by choosing a better 3-tuple (and hence proving 1.10 in less cases and steps). The reason why we divide by $\ln |C|$ at the end is because at each step that we apply 1.4 we get a more or less constant number of new bases (maybe 1 to 4 for $n = 4$), whereas the number of new cases is somewhat proportional to the current number of cases. We illustrate this concept with two examples.

Example 2.2 (3-tuple scoring 1)**.** Suppose at step s some 3-tuple branches off into 3 cases, with the worst case (that is, the case with the lowest case score) scoring 5. This would score $\frac{5}{\ln 3} = \frac{10}{\ln 9}$. Now if we branch off into 3 cases with the worst case scoring 5 (again) at step $s+1$ for each of the 3 cases, we will have 9 cases and a combined score of 10, which corresponds with the right hand side (RHS) of the equation. Alternatively, a 3-tuple at step s could branch off into 9 cases, with the worst case scoring 10. By our metric, this would again yield the same score. Hence, we believe our metric finds an appropriate trade-off between number of cases and the combined quality of the new bases resulting from each case.

Example 2.3 (3-tuple scoring 2)**.** Suppose at some step in the program we have a choice between two 3-tuples. One branches off into 3 cases at one application of 1.4, with the worst case yielding a normalized score of 5. Another branches off into 8 cases with the worst case yielding two bases scoring a total of 9.5 normalized. By our metric, these 3-tuples are about the same quality, since $\frac{9.5}{\ln 8} \approx 4.57$ and $\frac{5}{\ln 3} \approx 4.55$.

2.2. **Eliminating candidates.** In order to reduce the number of cases, we try to eliminate as many of the candidate partitions $\mathscr{O}(E-D)$ as possible. If the addition of the new bases proves 1.10, then we can eliminate that candidate. If two candidates result in two similar cases, we can eliminate one of the two candidates.

2.2.1. *Permutation cases.* We formalize the concept of two cases being similar (so that only one must be considered). Define some permutation σ of the elements of the ground set S of a matroid. Note that we can permute all subsets of S with σ as well. Given a collection $\mathcal{B}$ of bases, we can permute all the bases of $\mathcal{B}$ (since they are all subsets of S) to produce a new collection of subsets of S (not necessarily all bases). Hence, we can define the action of σ on a collection of bases in this way.

To test two cases for similarity, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the collection of all known bases after applying 1.4 on the first and second case. If some permutation σ of $\mathcal{B}_1$ yields $\mathcal{B}_2$, then these cases are similar and only one must be considered.

Example 2.4. Consider $B_1 = \{a_1, a_2\}, B_2 = \{b_1, b_2\}, \mathscr{O}(D) = (\{a_1\}, \{a_2\})$; we have candidates $(\{b_1\}, \{b_2\}), (\{b_2\}, \{b_1\})$ that each produce two new bases B_3, B_4, resulting in two collections of bases:

Collection/Basis	B_1	B_2	B_3	B_4
$\mathcal{B}_1$	$\{a_1, a_2\}$	$\{b_1, b_2\}$	$\{b_1, a_2\}$	$\{a_1, b_2\}$
$\mathcal{B}_2$	$\{a_1, a_2\}$	$\{b_1, b_2\}$	$\{b_2, a_2\}$	$\{a_1, b_1\}$

However, B_1 and B_2 are unchanged under the permutation σ that swaps b_1 with b_2, and the new bases $B_3 = \{a_2, b_1\}, \{a_2, b_2\}$ and $B_4 = \{a_1, b_2\}, \{a_1, b_1\}$ are equivalent after applying σ to one or the other. Hence, one of these cases can be eliminated.

2.2.2. *Superset cases.* Ultimately, we hope to prove that one can always find n disjoint rainbow bases starting with the initial collection $\mathcal{B}^* = \{B_1, \ldots, B_n\}$. On the way, we prove cases that start with $\mathcal{B}^*$ plus some other bases that we have inferred using 1.4. Suppose we prove a case C_1 that starts with the collection of bases $\mathcal{B}_1$. While proving another case C_2, we arrive at a collection of bases that contains $\mathcal{B}_1$. Then we no longer need to consider C_2 since we can prove the conjecture for C_2 by following the same steps as C_1, ignoring the additional bases that C_2 has.

In fact, we only need to consider the difference between the new bases inferred for each case. For instance, suppose case 1 results in a new basis X and case 2 results in a new basis Y. If the conjecture is

Unproven case	1-2	2-1	2-2	2-3-1	2-3-2
Proven case	1-1	1	2-1	1	2-2
Missing bases	L	H, I	O	H, I	L, M

FIGURE 2. Details of superset cases in Figure 3

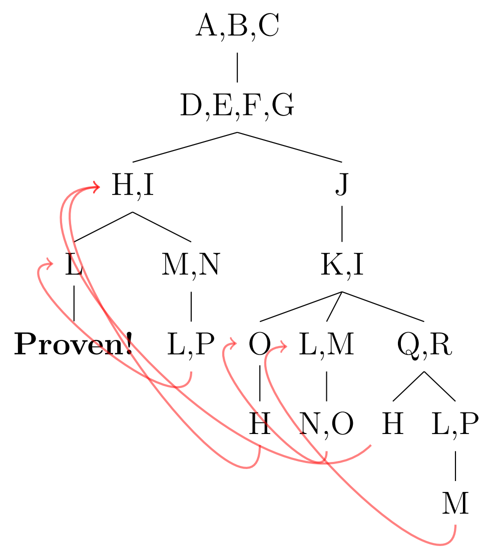

FIGURE 3. Case tree. Orange lines indicate a superset case

proven for case 1, and case 2 eventually infers a new basis X, then case 2 has also been proven, following the same steps as case 1. Consider Figure 3, which illustrates an application of this concept. Each letter represents a basis. As more bases are inferred, we find the need to branch off into different cases. The program considers cases on the left first, and proves the conjecture for case 1-1. However, case 1-2 infers the basis L, meaning it is proven through case 1-1. We call case 1-2 a superset case. We list all superset cases for this example in Figure 2. Here, an "unproven case" is proven through a "proven case" by inferring the "missing bases" that the unproven case needs to contain all the bases of the proven case.

2.3. **Exchange graph.** Again from White, there is another way to infer new bases that does not require branching off into cases [5]:

Definition 2.5 (exchange graph)**.** An exchange graph G for a collection of bases $\mathcal{B}$ in a matroid M is a directed multigraph whose vertices are the members of the ground set of the matroid. There is an edge from a to b labeled B_1 if and only if $b \in B_1 \in \mathcal{B}$ and $(B_1 - \{b\}) \cup \{a\} \in \mathcal{B}$.

Definition 2.6 (shortcut). Given a path or cycle $v_1 \xrightarrow{B_1} v_2 \xrightarrow{B_2} \dots \xrightarrow{B_{n-1}} v_n$, we say this path or cycle has a shortcut if and only if we can find $i < j - 1$ such that there is an edge directed from v_i to v_j.

Theorem 2.7 (no shortcuts). *Let $v_1 \xrightarrow{B_1} v_2 \xrightarrow{B_2} \dots \xrightarrow{B_{n-1}} v_n$ be a path or cycle with no shortcuts. The edge labels are not necessarily distinct but the vertices are, except in the case of a cycle where $v_1 = v_n$. Now let $i \in \{1, \dots, n-1\}$ and let $a_1 \xrightarrow{B_i} b_1, \dots, a_k \xrightarrow{B_i} b_k$ be all ordered vertex pairs with an edge labeled by B_i. Then $(B_i - \{b_1, \dots, b_k\}) \cup \{a_1, \dots, a_k\}$ is a basis in M.*

Recall that each candidate of each 3-tuple infers a number of new bases. These new bases will introduce new edges in the exchange graph, which may imply the existence of more new bases.

3. Results

Rota's basis conjecture has been completely proven for $n = 3$ in three cases by Chan [1], who uses 1.8 and other clever techniques. Furthermore, Onn proved that when n is even, the Alon-Tarsi conjecture implies 1.10 for all matroids arising from vector spaces (except vector spaces of certain characteristics) [4]. The Alon-Tarsi conjecture has been proven for $n = p + 1$ for all p prime by [2]. Furthermore, 1.10 has been verified for all paving matroids [3]. Hence, $n = 4$ for matroids is still open.

Using our algorithm, we have created a program that can completely prove Rota's basis conjecture for any n. The program generates a proof while it runs; the full proof of $n = 3$ is shown below. We have proven $n = 3$ in one case and $n = 4$ in around a hundred thousand cases. For $n \geq 5$, the program cannot complete in a reasonable amount of time. However, we can still use the program to look for counter-examples for these cases.

```
**********************START**********************
B0 =   a1   a2   a3
B1 =   b1   b2   b3
B2 =   c1   c2   c3
B3 =   a1   a3   b1
B4 =   a2   b2   b3
B5 =   a1   b1   c1
```

```
B6 =    a3    c2    c3
B7 =    a2    a3    b3
B8 =    b2    c2    c3
B9 =    a2    b2    c2
B10 =   b2    b3    c3
************************************

-a3 (in B7)     -c3 (solves)
B7(a2)/B6(c2)
B11 =   a3    b3    c2
B12 =   a2    a3    c3

-c3 (in B10)    -a3 (solves)
B10(b2)/B6(c2)
B13 =   b3    c2    c3
B14 =   a3    b2    c3

-b3 (in B10)    -a3 (solves)
B10(b2)/B7(a2)
B15 =   a2    b3    c3
B16 =   a3    b2    b3

-c2 (in B11)    -a2 (solves)
B11(a3)/B9(b2)
B17 =   b2    b3    c2
B18 =   a2    a3    c2

-a2 (in B12)    -b2 (solves)
B12(a3)/B9(c2)
B19 =   a2    c2    c3
B20 =   a2    a3    b2

-a3 (in B6)     -b1 (permutation of a1: a1 a2 a3 b1 b3 b2 c1 c3 c2 )
B6(c2)/B3(a1)
B21 =   a1    a3    c3
B22 =   a3    b1    c2

-a1 (solves)    -c3 (solves)
B4(b3)/B21(a3)
```

```
B23 =    a1        b3        c3

-a1 (solves)    -c1 (solves)
B20(a3)/B5(b1)
B24 =    a2        b1        b2
B25 =    a1        a3        c1

-a2 (in B24)    -c3 (solves)
B24(b1)/B12(a3)
B26 =    a2        b1        c3

-b2 (solves)    -b3 (solves)
B25(a1)/B17(c2)
B27 =    a3        c1        c2
B28 =    a1        b2        b3

-a1 (in B28)    -b1 (solves)
B28(b2)/B5(c1)
B29 =    a1        b3        c1
B30 =    a1        b1        b2

-b1 (in B26)    -a3 (solves)
B26(a2)/B3(a1)
B31 =    a1        b1        c3
B32 =    a2        a3        b1

-b1 (solves)    -b3 (solves)
B25(a1)/B1(b2)
B33 =    a3        b2        c1
B34 =    a1        b1        b3

-b3 (in B29)    -a3 (solves)
B29(a1)/B7(a2)
B35 =    a2        b3        c1
B36 =    a1        a3        b3

-c1 (solves)    -c3 (solves)
B28(b3)/B2(c2)
B37 =    a1        b2        c2
```

B38 = b3 c1 c3

********************SOLVED*****************
BASES =
0: a1 a2 a3
1: b1 b2 b3
2: c1 c2 c3
3: a1 a3 b1
4: a2 b2 b3
5: a1 b1 c1
6: a3 c2 c3
7: a2 a3 b3
8: b2 c2 c3
9: a2 b2 c2
10: b2 b3 c3
11: a3 b3 c2
12: a2 a3 c3
13: b3 c2 c3
14: a3 b2 c3
15: a2 b3 c3
16: a3 b2 b3
17: b2 b3 c2
18: a2 a3 c2
19: a2 c2 c3
20: a2 a3 b2
21: a1 a3 c3
22: a3 b1 c2
23: a1 b3 c3
24: a2 b1 b2
25: a1 a3 c1
26: a2 b1 c3
27: a3 c1 c2
28: a1 b2 b3
29: a1 b3 c1
30: a1 b1 b2
31: a1 b1 c3
32: a2 a3 b1
33: a3 b2 c1
34: a1 b1 b3

```
35: a2 b3 c1
36: a1 a3 b3
37: a1 b2 c2
38: b3 c1 c3
RAINBOW BASES =
0: a1 b1 c1
1: a2 b2 c2
2: a3 b3 c2
3: a3 b2 c3
4: a2 b3 c3
5: a3 b1 c2
6: a1 b3 c3
7: a2 b1 c3
8: a1 b3 c1
9: a1 b1 c3
10: a3 b2 c1
11: a2 b3 c1
12: a1 b2 c2

-a1 (solves)   -b1 (solves)   -b3 (solves)
B8(c2)/B34(?)
********************SOLVED*****************

NUMBER OF CASES = 1
```

References

1. Wendy Chan, *An exchange property of matroid*, Discrete Mathematics **146** (1995), 299–302.
2. Arthur A Drisko, *On the number of even and odd latin squares of orderp+1*, Advances in Mathematics **128** (1997), no. 1, 20 – 35.
3. Jim Geelen and Peter J. Humphries, *Rota's basis conjecture for paving matroids.*
4. Shmuel Onn, *A colorful determinantal identity, a conjecture of rota, and latin squares*, Department of Operations Research (1997).
5. Neil White, *Theory of matroids*, Cambridge University Press, 1986.

James Madison University, Harrisonburg, VA 22801
Current address: 57 Berkshire Way, East Brunswick, NJ 08816
E-mail address: cheungm@etown.edu