CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2014, Article 05, pages 1–18
http://cjtcs.cs.uchicago.edu/

Complexity of Testing Reachability in Matroids

B.V. Raghavendra Rao* Jayalal Sarma†

Received January 22, 2012; Revised June 7, 2014; Published July 9, 2014

Abstract: We extend the complexity theoretic framework of reachability problems in graphs to the case of matroids. Given a matroid M and two elements s and t of the ground set, the reachability problem is to test if there is a circuit in M containing both s and t. We show complexity characterizations for several important classes of matroids. In particular, we show: (1) For two important classes of matroids associated with graphs, namely, graphic and bi-circular matroids we show that the reachability problem is L-complete. (2) For transversal matroids, when a basis of M is also given at the input, the problem can be shown to be NL-complete. A general upper bound for this case is the complexity of constructing a matching ($\mathsf{RNC}^2 \cap \mathsf{P}$). (3) For linear matroids representable over $\mathbb{Q}$ and $\mathbb{Z}_p$, we show that the problem characterizes $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ and $\mathsf{Mod}_p\mathsf{L}$ respectively, which provides the first characterizations of these classes in terms of reachability problems.

1 Introduction

Reachability testing in various mathematical structures is a fundamental problem in complexity theory as they encode resource bounded computations in a natural way. The most useful one among them, especially in the context of space bounded computations, is the graph reachability problem, and has exact complexity characterizations both in the case of directed and undirected graphs. For directed graphs

*Department of Computer Science and Engineering, Indian Institute of Technology Madras , Chennai, 600 036, INDIA. bvrr@cse.iitm.ac.in

†Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, 600 036, INDIA. jayalal@cse.iitm.ac.in

Key words and phrases: Space Complexity, Reachability Problems, Matroid Theory

DOI: 10.4086/cjtcs.2014.005

testing reachability (STCON) is known to be complete for the non-deterministic log-space class NL ([9, 17]). From a celebrated result of Reingold [16] it follows that *st*-connectivity for undirected graphs is complete for the deterministic log-space class L. Apart from these two types of graphs, there has been extensive research(see [1] for a survey) aimed at settling the complexity of testing reachability on restricted graphs.

We extend this framework considering the notion of reachability in matroids. Matroids are combinatorial objects introduced by Whitney [19] generalizing the abstract notion of independence in linear algebra and in graph theory. More formally, a matroid M is a combinatorial object defined over a finite set S (of size m) called the *ground set*, equipped with a non-empty family $\mathcal{I}$ of subsets of S (containing the empty subset) which is closed under taking of subsets and satisfies the *exchange axiom* : for any $I_1, I_2 \in \mathcal{I}$ such that $|I_1| > |I_2|$, $\exists x \in I_1 \setminus I_2$, $I_2 \cup \{x\} \in \mathcal{I}$. The sets in $\mathcal{I}$ are called *independent sets*. The rank of the matroid is the size of the maximal independent set. The minimal dependent subsets of S are called *circuits* in the matroid. This provides useful abstractions of many concepts in combinatorics and linear algebra and is well studied [15].

In a matroid $M = (S, \mathcal{I})$, two elements s and t of the ground set S are said to be *reachable* (or *connected*) if there is a circuit which contains both of them. The notion of connectivity in matroids was introduced by Tutte in [18]. Since then, there has been a lot of work on the connectivity properties of a matroid and its relationship to graph connectivity (See Chapters 4 and 8 of [15].) In particular, a matroid is said to be *connected* if every pair of elements of the ground set are connected. Furthermore, if a matroid is not connected, then its ground set can be partitioned uniquely into connected components. The original matroid can be reconstructed as the direct sum of the components.

In order to achieve this decomposition, and to identify the components, a natural computational task is to test connectivity between two given elements. This motivates the problem : Given a matroid $M = (S, \mathcal{I})$ and $s, t \in S$, test whether s and t are reachable (from each other). Complexity characterizations of this problem can be expected to provide new insights, since it generalizes the linear algebraic and combinatorial notions of independence in a unified manner. To the best of our knowledge, this problem has not been explicitly studied before from a complexity theoretic perspective.

In [14], Mihail and Sudan showed that a matroid is connected if and only if a graph associated with it (called base-exchange graph) is connected. This does imply a connection to graph reachability but it is computationally inefficient since the size of the base-exchange graph of a matroid could be exponential in it the size of the ground set of the matroid. In addition, it is unclear how this extends to testing reachability between two given elements in the ground set of the matroid.

Quite naturally, the representation of the input matroid $M = (S, \mathcal{I})$ is important in deciding the complexity of the algorithmic problem. There are several equivalent representations of a matroid. For example, enumerating the maximal independent sets (called bases) or the minimal dependent sets (called circuits) also defines the matroid. These representations, although they can be exponential in the size of the ground set, indeed exist for every matroid, by definition.

For the case of implicit representations, a general framework is the representation of a matroid over a field. A matroid $M = (S, \mathcal{I})$ of rank r is said to be *representable* over a field $\mathbb{F}$ if there is a $\phi : S \to \mathbb{F}^r$ such that, $\forall A \subseteq S$, $A \in \mathcal{I} \iff \phi(A)$ is linearly independent over $\mathbb{F}^r$ as a vector space. However, there are matroids which do not admit linear representations over any field. (For example, the Vamós Matroid, See Proposition 6.1.10, [15].). In contrast, there are matroids (called regular matroids) which admit linear

representations over all fields. We show that for the case of matroids which are representable over finite fields or over $\mathbb{Q}$, testing reachability is complete for the class $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$. In addition, the construction also implies that testing reachability over matroids representable over $\mathbb{Z}_p$ for a prime p, is complete for the class $\mathsf{Mod}_p\mathsf{L}$.

Another natural representation for a matroid is over graphs. For any graph X, there are two natural matroids that can be associated. The first, and the well-known one, is called the *graphic matroid* $M(X)$ of the graph whose ground set is the set of edges of X, and the acyclic sub graphs of X as the independent sets. Graphic matroids are linear but there are linear matroids which are not graphic. (See [15]). We show that reachability in graphic matroids is complete for L by exploiting the close connection between connectivity in graphic matroids and the 2-connectivity of the corresponding graphs representing them.

A less known matroid associated with graphs, named *bi-circular matroids*, has the edge sets in which each connected component contains at most one cycle, as the independent sets. Using a slightly more involved construction, we demonstrate a log-space algorithm for testing connectivity in bi-circular matroids. We also obtain a matching L-hardness result.

Another important class of matroids that we consider is the transversal matroids which arise in matching theory. These matroids are linearly representable over large enough fields. We assume that the input given in this case, is a bipartite graph which represents the transversal matroid (see section 4). It is known [6] that a transversal matroid is binary if and only if it is graphic. In the context of complexity of the reachability problem, we show an analogous situation by showing that the reachability problem for transversal matroids (when given a basis as a part of the input) is complete for NL. We use a known [8](see also [3]) connection between directed graphs and the duals of these matroids (called strict gammoids). However, to apply this bound to the general presentation of transversal matroid, we need to compute a basis, and in general this is as hard as computing a matching in the given graph. In this case we get a bound of $\mathsf{RNC}^2 \cap \mathsf{P}$ using the known [10] upper bounds of the matching problem.

Our Results The results of this paper as as follows :

- Reachability testing problem for graphic matroids (GMR) and for bi-circular matroids (where matroid is given as the graph representing it) are complete for deterministic log-space. (Theorems 3.5 and 3.4.)

- For the case of matroids which are representable over finite fields or over $\mathbb{Q}$, given the representation of the matroids and the elements s and t, testing reachability is complete for the class $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$. In addition, our construction also implies that testing reachability over matroids representable over $\mathbb{Z}_p$ for a prime p, is complete for the class $\mathsf{Mod}_p\mathsf{L}$. (Theorem 3.5.)

- The reachability problem for transversal matroids (when given a basis as a part of the input) is complete for NL. If the basis is not a part of the input, the upper bound is $\mathsf{RNC}^2 \cap \mathsf{P}$. (Theorem 4.4 and Corollary 4.5.)

2 Notations and Preliminaries

We briefly review the space bounded complexity classes that we will be used in the paper. (For a detailed description see [4].) The class of languages that are accepted by deterministic logarithmic space bounded Turing machines is denoted by L and those accepted by non-deterministic log-space bounded machines is denoted by NL. $\oplus\mathsf{L}$ denotes the class of languages L for which there exist a non-deterministic logarithmic space bounded machine M such that for every $x \in \{0,1\}^*$: The number of accepting paths in M on x as input is odd if and only if $x \in L$. $\mathsf{C}_=\mathsf{L}$ denotes the class of languages that are representable by non-deterministic log-space bounded Turing machines such that a string x is in the language if and only if there are equal number of accepting and rejecting paths in the witness machine. It is known that testing singularity of integer matrices is complete for $\mathsf{C}_=\mathsf{L}$ under log-space many-one reductions [2].

Brief Overview of Matroid Theory: We define some additional terms from matroid theory that will be used in the paper. (See e.g., [15] for further details.)

Definition 2.1. A matroid M is an ordered pair $(S,\mathcal{I})$ consisting of a finite set S, called the ground set, and a finite collection $\mathcal{I}$ of subsets of S, called the independent sets, satisfying the following axioms:

1. $\emptyset \in \mathcal{I}$; and
2. $A \in \mathcal{I}, B \subseteq A \implies B \in \mathcal{I}$; and
3. If $I_1, I_2 \in \mathcal{I}$, and $|I_1| > |I_2|$, then $\exists x \in I_1 \setminus I_2$ such that $I_2 \cup \{x\} \in \mathcal{I}$.

The condition 3 above is called the *exchange axiom* or the *independence augmentation axiom*.

Given a matroid $M = (S,\mathcal{I})$, the subsets of S that are not included in $\mathcal{I}$ are called *dependent sets*, and minimal dependent sets are known as *circuits* of M. A *loop* is an element in S that is not independent (and hence dependent) in M. A maximal independent set in $\mathcal{I}$ is called a *basis* of M. Rank of a set $A \subseteq S$, denoted by $\textsc{Rank}(A)$ is the maximum possible size of an independent set in A, i.e

$$\textsc{Rank}(A) = \max_{B \in \mathcal{I}, B \subseteq A} |B|.$$

Rank of M is defined to be $\textsc{Rank}(S)$. Note that a matroid $M = (S,\mathcal{I})$ can also be described uniquely by the set of all circuits, or the set of all bases, or the RANK function, for more details see Chapter 1 in [15].

Two elements $s,t \in S$ are said to be *reachable* or *connected* in M, if there is a circuit C in M with $\{s,t\} \subseteq C$. More generally, M is said to be connected if every par $s \neq t \in S$ are reachable from each other. The notion of connectivity was introduced by Tutte in [18].

A hyperplane of M is a set $H \subseteq S$ of rank $\textsc{Rank}(S) - 1$ and is such that $\forall x \in S \setminus H$, $\textsc{Rank}(H \cup \{x\}) \neq \textsc{Rank}(H)$. Dual of a matroid M denoted by M^* is the matroid defined on the same ground set, where the bases of M^* are set-complements of those of M.

Problem 2.2 (Matroid Reachability)**.** Given a matroid $M = (S,\mathcal{I})$ on a ground set S and $s,t \in S$, test if s and t are reachable in M.

However, the complexity of the matroid reachability problem depends on how a matroid is represented at the input.

One of the simplest ways to represent a matroid is to explicitly list all of the independent sets in $\mathcal{I}$. This input representation is considered in the literature, and see [12] and [13] for a detailed account on complexity of problems on matroids based on various explicit representations of matroids.

Though every matroid has an explicit representation, it could be that the number of independent sets, or bases is exponential in the size of the ground sets. A more compact representation would be via an independence oracle, i.e., the input matroid $M = (S, \mathcal{I})$ is represented by a black-box procedure that decides whether a given set A is independent in M or not. We call this representation as the *independent set oracle* representation. Notice that the general version of the matroid reachability problem with input matroid given as independent set oracle is in NP. Indeed, the non-deterministic machine guesses the elements of S which forms the circuit through s and t, and deterministically verifies, with the help of at most $n-1$ queries whether all its proper subsets are independent. In the next section we give a direct reduction from matroid reachability to the graph reachability problem.

3 A Direct Reduction to Graph Reachability

In this section, we outline a general reduction from matroid reachability to graph reachability[1].

Theorem 3.1. *Let $M = (S, \mathcal{I})$ be a matroid, B any basis of M. There is a bipartite graph a $G_B^M = (U, V, E)$ with $V \cup U = S$ such that $s, t \in S$ are reachable in G_B^M if and only if s and t are reachable in the matroid M.*

Proof. Without loss of generality, assume that s, t are not loops and $\{s, t\}$ is independent. Let B be a basis of the matroid M with $\{s, t\} \subseteq B$. Construct the following bipartite graph: $G_B^M(U, V, E)$ where $U = B$ and $V = S \setminus B$. The set of edges E is defined as follows.

$$\forall u \in U, v \in V : (u, v) \in E \iff B \setminus (\{u\} \cup \{v\}) \in \mathcal{I}$$

We show that G_B^M satisfies the required property.

($\Leftarrow$) Let C be a circuit in M containing both s and t. Let $k = |C| - 2$. Then there exist two bases B_1 and B_2 such that B_1 contains s but not t and B_2 contains t but not s. (These bases are obtained by growing the independent sets obtained from C by removing s and t respectively.) Let $R = B \setminus \{s, t\}$. By the exchange axiom, there exist e_1' and e_2' such that $\{s, e_1'\} \cup R$ and $\{e_2', t\} \cup R$ are bases of M. Hence, by the exchange axiom $\{s, e_2'\} \cup R$ and $\{e_1', t\} \cup R$ too are bases of M. Thus s and t are connected in G_B^M.

($\Rightarrow$) Suppose two vertices s and t are connected in the bipartite graph G. By the above argument, we can assume that there is a path of length 2 between s and t. Let s, b, t be such a path. It is sufficient to exhibit a hyperplane in M^* that avoids both s and t. Note that V is a basis of M^*. By the definition of G_B^M, we have $V \setminus \{b\} \cup \{s\}$ and $V \setminus \{b\} \cup \{t\}$ are bases in M^*. Now $closure_{M^*}(V \setminus \{b\})$ is the required hyperplane that avoids both s and t. □

[1] This was suggested by an anonymous referee. We are including the complete proof here for completeness.

Let $\mathcal{C}$ denote a complexity class. A matroid family $\mathcal{M}$ is said to admit independence test in $\mathcal{C}$, if for any matroid $M \in \mathcal{M}$, given a subset A of the ground set, testing if A is independent in M can be performed in $\mathcal{C}$. An immediate corollary of Theorem 3.1:

Corollary 3.2. *If a class of matroids $\mathcal{M}$ admits independence test in $\mathcal{C}$, then reachability for matroids in $\mathcal{M}$ can be done in* log *space with queries to $\mathcal{C}$.*

Proof. We assume that $M \in \mathcal{M}$ which admits independence test in the class $\mathcal{C}$. That is given a subset S of the ground set implicitly (given each ground set element there is an algorithm which returns whether it is in S or not), we can check in $\mathcal{C}$, whether S is independent or not.

Now, it suffices to argue that, for a given matroid M, we can find a basis B that contains s and t, and the graph G_B^M corresponding to that basis in Theorem 3.1 can be produced in log space given the independent set oracle. We find the basis B using the independence oracle of the matroid M. That is, given a ground set element g we can decide in log-space whether it is in the basis B or not by iteratively doing the following : in the lexicographic order of the elements, add the ground set elements to the set $\{s,t\}$ maintaining independence. Growing it to a basis this way, we will be able to find out whether $g \in B$ or not. Having found B implicitly, for every $u,v \in U \cup V$, to decide whether there is an edge (u,v) in the graph G_B^M or not, we simply remove the vertices u and v (again implicitly in log-space) and query the oracle $\mathcal{C}$ to see if $B \setminus (\{u\},\{v\})$ is independent or not.

Corollary now follows from the fact that given a graph implicitly (each edge is described by a log-space machine) and give vertices s and t, we can test if t is reachable from s in log space. □

3.1 Reachability on Matroids from Graphs

In this section we apply Corollary 3.2 to the reachability testing problem when the input matroid is defined based on graphs. There are two well studied classes of matroids associated with graphs, and we consider both of them.

Graphic Matroids: As mentioned in the introduction, perhaps the most interesting matroid associated with a graph is the so called cycle matroid (or graphic matroid). We start with a formal definition of the problem that we address in this subsection. *Given an undirected graph $G = (V,E)$ and a pair of edges (e,f), test if there is a (simple) cycle in G, that contains both e,f.* We call this problem GMR (short for Graphic Matroid Reachability). We show that GMR exactly characterizes deterministic log-space.

Theorem 3.3. GMR *is* L*-complete.*

Proof. MEMBERSHIP: Let $G = (V,E)$ be the given graph, and e and f be the two edges. By Corollary 3.2, it is enough to show that given a subset $A \subseteq E$, testing if A is independent in the matroid $M(G)$ can be done in deterministic log-space. Note that, $A \subseteq E$ is independent in $M(G)$ if an only if the sub-graph $G[A]$ of G induced by A is acyclic. A is acyclic if and only if for every edge $g = \{u,v\} \in A$, u and v are not reachable from each other in the graph $G[A \setminus g]$. This can be done via queries to undirected graph reachability algorithm of Reingold [16], and hence can be done in deterministic log-space.
HARDNESS: The reachability problem in undirected forests (UFA) is known to be L-complete ([5]). Let G be a forest and s and t be the two designated nodes given at the input. Construct a new graph G' by

creating an additional vertex u, and connecting it to both s and t. It is clear that there exists a path between s and t in G if and only if there is a (simple) cycle containing the edges (s,u) and (u,t). This gives the reduction. □

Bi-circular matroids: We can extend the above result to the reachability problem in the case of bi-circular matroids. These are another class of linearly representable matroids associated with graphs and were introduced in [11]. Given an undirected graph $G = (V,E)$, a bi-circular matroid of G is the matroid $B(G) = (E,\mathcal{I})$, where $\mathcal{I}$ is the set of of pseudo-forests of G (pseudo-forests are undirected graphs in which every connected component has at most one cycle). The circuits of $B(G)$, called bi-circular sub-graphs of G (also called bicycles), are those which contain two simple cycles, which either (1) are connected by a simple path (2) intersect on a path or (3) intersect at a vertex (see Figure 1). Applying Corollary 3.2 to solve matroid reachability, will need implementation of the independence set oracle. An independence set tester needs to detect the absence of any bi-cycle in a give subset of edges. However it turns out that testing for existence of a bicycle is easier which in turn gives a direct algorithm for reachability testing in bi-circular matroids, without appealing to Corollary 3.2.

Theorem 3.4. *Testing reachability in bi-circular matroids is* L*-complete.*

Proof. MEMBERSHIP: Let $\langle G,e,f\rangle$ be the given graph representing the bi-circular matroid $B(G)$. By the definition of $B(G)$, $e = (a,b)$ and $f = (c,d)$ are reachable in $B(G)$ if and only if there is a bi-circular sub graph containing e and f. We refer the reader to Figure 1 for the forms in which this bi-circular sub graph can appear in the graph. Hence, it requires careful applications of connectivity testing algorithm in order to test this. We present the algorithm in Algorithm 2.

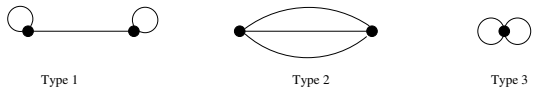

Figure 1: Bicycles are subdivisions of one of these graphs

Correctness: The proof of correctness of the above algorithm is an involved case analysis which we will present below. The interesting part of the proof is the correctness of step 32, where we can argue that checking one more additional path P' in step 33 suffices to handle all the cases that arise in the situation.

We refer to the algorithm for the notations. Suppose G has a bicycle A containing e and f. Let D_1 and D_2 be the two cycles in A. First we argue the correctness of the procedure ObtainCycle and steps 5-9. If steps 3.1-3.4 of ObtainCycle successfully find paths P and P' then we have the required cycle C at the end of ObtainCycle. But, it is possible that step 3.3 of ObtainCycle fails. In this case, removal of P destroys all the cycles containing both e and f. However, this combined with Step 4.4 of ObtainCycle ensures that there is a bicycle containing both e and f and hence we can accept the input and halt.

For the rest of the algorithm we have the following cases:

Case 1 : *A is of Type-1c.* In this case the procedure ObtainCycle will return ACCEPT, and hence the algorithm will succeed.

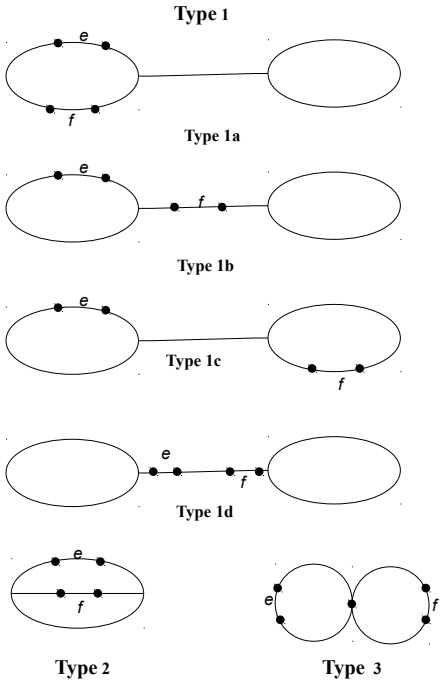

Figure 2: Different types of bicycles that connect the edges e and f.

Case 2 : *A is of Type-2* (See Figure 2). In this case there is a cycle C containing e and f and hence Step 5 will be performed and we can assume that we have a such a cycle C. Now, irrespective of whether C is one of the cycles (formed by two of the three parallel paths) or not, step 13 of the algorithm will succeed for some pair of vertices w and w'. Hence the input will be successfully accepted.

So in the following cases we need to deal with bicycles of type 1 and 3 only.

Case 3 : *Both e and f are inside the same cycle*. Without loss of generality suppose that D_1 contains both e and f. If $C = D_1$ (Where C is the cycle obtained at the end of Step 5), then irrespective of A being Type 3 or 1 one of the steps 5-9 will report success hence the algorithm will ACCEPT. The trouble is when $C \neq D_1$. We have the following sub-cases:

Sub-case 1 C intersects (edge intersection) with D_1 (or D_2) alone. In this case, the edge induced sub-graph $C \cup A$ contains another bicycle A' of Type 2 containing both e and f and having C as one of its cycles. Hence Step 13 will detect this bicycle and report success.

Sub-case 2 C intersects both D_1 and D_2. (Note that we mean edge-wise intersection.) Irrespective of the type of the bicycle A, G contains another bicycle A' of type 2 containing both e and f.

Algorithm 1 Procedure: ObtainCycle

Require: $G = (V,E), e = (a,b), f = (c,d) \in E$

Ensure: Test if there is a cycle containing e and f, or a bicycle of Type 1c containing e and f or return NULL (see Figure 1)

1: **for all** $u \in \{a,b\}$, $v \in \{c,d\}$ **do**

2: // $\{u'\} = \{a,b\} \setminus \{u\}$ and $\{v'\} = \{c,d\} \setminus \{v\}$.

3: 1. Find a simple path P between u and v.

2. Remove the vertices appearing in this path P from the graph G to get G'.

3. Find a path P' between u' and v' in G'.

4. If there is no such path and if u' and v' are connected in $F \setminus \{e,f\}$ then return(ACCEPT).

4: **end for**

5: C denotes a simple cycle containing e and f obtained by joining the two vertex-disjoint paths P and P' using e and f if one such exists, else return(NULL).

6: return C.

Hence Step 13 must succeed. We demonstrate this for one particular case, for the remaining cases the argument is similar. Suppose A is of type 1. A' contains C, the path from one of the bifurcating points of C and D_1 to one of the bifurcating points of C and D_2. Now, A' contains 3 parallel paths and hence is a bicycle of Type 2. Since A' contains C, A' is as required. See Figure 3

Case 4 : $e \in D_1$ and $f \in D_2$. Type 1 and 3 are the only possibilities here for A. Now we consider Step 3 of the algorithm. If $C_1 = D_1$, then the Steps 16-27 will succeed depending on the type of the bicycle A. When $C_1 \neq D_1$ as in Case 2, we have the following sub-cases:

Sub-case 1 C_1 intersects D_1 alone. If A is of Type 1, then there is an A' of Type 1, having C_1 in place of D_1 and containing both e and f. Hence Steps 23-25 will detect A' and report success. If A is of Type 3, then G has another bicycle A' of type 3 or type 1 depending on if the intersection vertex between D_1 and D_2 is contained inside C_1 or not. Note that A' contains C_1 as one of its cycles. Hence either Step 18-21 or Step 23-25 is guaranteed to succeed depending on the type of A'.

Sub-case 2 C_1 intersects both D_1 and D_2. As done in the sub-case 2 in the Case 2 above, we can come up with a different bicycle of Type 2 that contains both e and f. Hence this will be detected by step 13 and hence the algorithm will succeed.

Case 4 : e and f are not contained in any of the two cycles D_1 and D_2. This necessitates the witness bicycle A to be of *Type 1 c* . Then surely e and f must lie inside a path Q that connects the two circuits D_1 and D_2 in A. Suppose the path P obtained in Step 28 agrees with Q. Then, clearly Step 30 will succeed and accept. The problem is when P does not agree with Q and intersects at least one of D_1 and D_2. Without loss of generality , assume that Q induces the ordering a,b,c,d

Algorithm 2 Testing bi-circular Reachability in graphs

Require: $G = (V,E)$, $e = (a,b), f = (c,d) \in E$

Ensure: Test if there is a bicycle (see Figure 1) containing e and f.

If e and f are not in the same connected component in G then **reject** and **halt**
if ObtainCycle(G, e,f)= ACCEPT **then**
  ACCEPT and HALT
end if
if ObtainCycle(G, e,f)= C **then**
  for all ($w \in C$) **do**
    If there is a cycle C' in G such that $C \cap C' = \{w\}$, then **accept** and **halt**
    for all ($z \notin C : z \rightsquigarrow w$) **do**
      If $G \setminus C$ has a cycle containing z, then **accept** and **halt**.
    end for
  end for
  for all $w, w' \in C$ **do**
    If there is a path between w and w' avoiding all other vertices of C then **accept** and **halt**.
  end for
end if
for all $p \in \{e, f\}$ **do**
  // $q \in \{e, f\} \setminus \{p\}$
  if p is contained in a simple cycle C_1 in G **then**
    for all w in C_1 **do**
      If G contains a simple cycle C_2 containing q with one of its vertices reachable from w and C_2 is vertex-disjoint from C_1 then **accept** and **halt**.
    end for
    // There is no such cycle C_2 for any $w \in C_1$
    for all w in C_1 such that P is a path from w to q **do**
      If there is a cycle C_2 vertex disjoint from P and C_1, reachable from w, then **accept** and **halt**.
    end for
  end if
end for
// Let P be a path between e and f.
// Let u ($u \in \{a,b\}$) and v ($v \in \{c,d\}$) be the start and end vertex of P respectively.
If ($\exists w \in V$ such that $w \rightsquigarrow e$ in $G \setminus P$)
  $\wedge$ ($G \setminus P$ has a simple cycle C_1 containing w)
  $\wedge$ ($G \setminus (C_1 \cup P)$ contains a cycle C_2 such that $\exists w' \in C_2$ with $w \rightsquigarrow f$)
then **accept**
if there is a path P' between u and v in G **then**
  Redo step 28-30 with P' in the place of P and **accept** and **halt** if it succeeds.
end if
reject

among the incidence vertices of e and f. We consider cases based on the ordering of these vertices induced by P. (Recall that we have $e = (a,b)$ and $f = (c,d)$). We have two sub-cases.

Sub-case 1 (P intersects one of D_1 or D_2) Without loss of generality suppose P intersects with D_1. Our analysis depends on the ordering of the vertices a,b,c and d induced by P.

(b,a,d,c) In this case G contains cycle D containing both the edges e and f and a witness bicycle A' of Type 1 or 3 (this depends on whether the vertex d is contained in the cycle D_2 or not), such that D is contained inside A'. Hence Step 2 will detect this situation and accept. See Figure 4 for a pictorial demonstration.

(b,a,c,d) In this case G contains a circuit D that contains e. Also, there is a witness bicycle A' of Type 1 containing D as one of its cycles. Hence Steps 6-15 will detect this and report success.

(a,b,c,d) Here too, there is a cycle D containing e and a bicycle A' of type 1 containing this cycle.

(a,b,d,c) : Can be handled in a way similar to the above case.

Sub-case 1 (P intersects both of D_1 and D_2) In this case, removing P destroys both the cycles and hence step 30 need not detect it. We will have another case analysis based on the ordering of of the vertices a,b,c and d induced by the path P.

(a,b,c,d) With this ordering, we can find two cycles D and D' with $e \in D$ and $f \in D'$ and such that they are connected by a simple path (A part of Q). So this gives a bicycle A' of Type 1 with both the edges contained in cycles. Hence this case will be detected by Steps 16-27 and accepted.

(a,b,d,c) Here we get a cycle D containing the edge e and a cycle D'. D and D' are connected by a path (a part of Q) containing f. So we have a bicycle A' of Type 1 and this will be detected by Steps 16-27.

(b,a,c,d) Symmetric to the case above.

(b,a,d,c) In this case, once we remove the path P both the cycles D_1 and D_2 vanish and hence Step 30 will not be able to detect a bicycle containing e and f. If P intersects the path Q on at least one edge, then it immediately gives another bicycle of *Type 1c* so earlier steps would detect a bicycle. Thus one can assume that P does not have any edge common to Q ,and Q remains intact even after removing the edges in P. Hence Step 31 will find this path and Step 32 shall succeed now as the ordering of the vertices a,b,c and d is changed in this new path P'. See Figure 5 for pictorial demonstration.

Hence the correctness when e and f are reachable in $B(G)$. When e and f are not reachable in $B(G)$, none of the steps in the algorithm will succeed as success in any step of the algorithm corresponds to detection of a bicycle.

Space Analysis: In the space analysis we crucially use the log-space algorithm for testing connectivity in an undirected graph, and also the fact that such a path can be computed in deterministic log-space if it exists [16]. As explained in the proof of theorem 3.3 we can test if two edges $e = (a,b)$ and $f = (c,d)$ are contained inside a cycle in L.

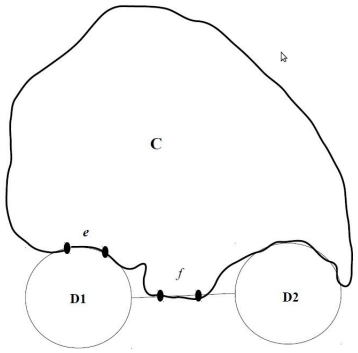

Figure 3: Formation of bicycle A' in sub-case 2 of case 3

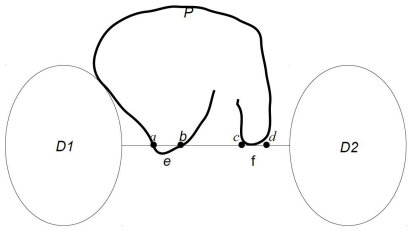

Figure 4: An example of Sub-case 1 in Case 4

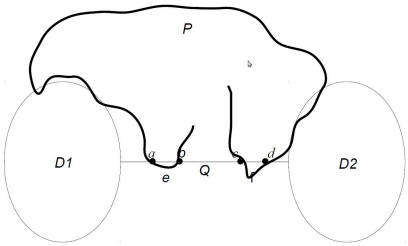

Figure 5: An example of Sub-case 2 in Case 4

First note that we can test if there is a simple cycle containing the edges e and f. There is a cycle containing e and f if and only if every pair of end vertices of e and f are contained inside a cycle. The latter condition can be tested using at most n queries to undirected graph reachability and hence can be performed in L. Step 5 needs a level 2 composition of log-space algorithms. The first one is to compute the path P and the next one to compute P'. The remaining steps of the algorithm would require a 3 level composition of log-space algorithms. First one is to compute a cycle C. The next one runs through all $w \in C$ and $z \in V \setminus C$ and tests reachability (in the graph) of (w,z). This requires C as its input. The last level is to test if there is a circuit containing z using the undirected st-connectivity. This uses outputs of all the two previous algorithms. So we have a level 3 composition of log-space algorithms and hence this can be done in log-space. Using similar implementations, we can see that the remaining steps too can be performed in log-space.

HARDNESS: For hardness, we reduce from undirected forest accessibility (UFA) which is known [5] to be complete for L. let G be a forest with s and t as designated start and terminal nodes. Define G' based on G as follows : Add three additional vertices u, v and w. Add edges (s,u) and (u,t), add a simple cycle $(s,v),(v,w),(w,s)$. Now it is easy to see that, s and t are reachable in G if and only if there is a bi-circular graph in G' containing the edge (s,u) and (s,v). □

3.2 Linear Matroids

We consider the reachability problem for linear matroids. We define the Linear Matroid Reachability problem (LMR) as follows:

Given a linear matroid as an $m \times n$ matrix A over a field $\mathbb{F}$ and a pair of indices e, f. Test if there is a minimal dependent set of columns in A, that contains both columns indexed by $e, f \in [n]$. Formally, let A_e, A_f denote the columns of the matrix A indexed by e and f respectively.

$$\text{LMR}(\mathbb{F}) = \left\{ \langle A, e, f \rangle : A_e \text{ and } A_f \text{ are reachable in the matroid } M(A). \right\}$$

Let

$$\text{SINGULAR}(\mathbb{F}) = \{A \mid A \in \mathbb{F}^{n \times n} \;\; A \text{ is singular.}\}$$

As testing independence in linear matroids reduces to testing if a matrix is singular or not, applying Corollary 3.2 we have,

Lemma 3.5. $\text{LMR}(\mathbb{F})$ *can be decided in* $\mathsf{L}^{\text{SINGULAR}(\mathbb{F})}$.

Now we turn to lower bound for the problem LMR. In what follows, we show that $\text{LMR}(\mathbb{Q})$ is hard for the class $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ and $\text{LMR}(\mathbb{Z}_p)$ is hard for $\mathsf{Mod}_p\mathsf{L}$ under log-space Turing reductions.

Let p be a prime, and let $\mathbb{F}$ be one of either $\mathbb{Q}$, $\mathbb{Z}_p$. Recall that $\text{FSLE}(\mathbb{F})$ denotes the problem of testing feasibility of a system of linear equations over $\mathbb{F}$. That is, given a matrix $A \in \mathbb{F}^{m \times n}$ and a vector $b \in \mathbb{F}^m$, test if there exists an $a \in \mathbb{F}^n$ such that $Aa = b$. It is known that $\text{FSLE}(\mathbb{Q})$ is complete for $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$. (Theorem 3.2 in [2].)

Lemma 3.6. $\text{FSLE}(\mathbb{F}) \leq_T^{\mathsf{L}} \text{LMR}(\mathbb{F})$.

Proof. Let $Ax = b$ be an instance of FSLE. $Ax = b$ is feasible if and only if the matrix Ab has a dependent subset of column vectors containing b if and only if there is a minimally dependent subset of column vectors in Ab containing b. This can be tested by testing if there is a column v in A such that v and b are reachable in the matroid $M(Ab)$. □

Combining this with the upper bounds described above, we get the following theorem. For the case of $\mathbb{F} = \mathbb{Z}_p$ for some prime p, we get a similar result, combining lemma 3.6 and using the fact that $\mathsf{L}^{\mathsf{Mod}_p\mathsf{L}} = \mathsf{Mod}_p\mathsf{L}$ (see [7]). Combining this with the above upper bounds, we get:

Theorem 3.7. $\mathrm{LMR}(\mathbb{Q})$ *is complete for the class* $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ *under log-space Turing reductions. For any prime* p, $\mathrm{LMR}(\mathbb{Z}_p)$ *is* $\mathsf{Mod}_p\mathsf{L}$*-complete under log-space Turing reductions.*

4 Transversal Matroids

We begin with the definition of a transversal matroid. Let $S = \{1,\ldots,n\}$ be a finite set and $\mathcal{A} = \{A_1,\ldots,A_m\}$ be a collection of subsets of S. A transversal of $\mathcal{A}$ is a $T \subseteq S$, such that there is a bijection $\pi : T \to \{1,\ldots,m\}$ with $i \in T \iff i \in A_{\pi(i)}$. T is a partial transversal if π is an injective map. The set of all partial transversals of $\mathcal{A}$ form a matroid with S as its ground set, called *transversal matroid* denoted by $M[\mathcal{A}]$. A matroid $M = (S,\mathcal{I})$ is said to be a transversal matroid if it is isomorphic to $M[\mathcal{A}]$ for some collection of subsets $\mathcal{A}$.

We can also define a transversal matroid via set of matchings of a bipartite graph. Given S and $\mathcal{A}$ as above, define a bipartite graph $X = (S,J,E)$ where $J = \{1,\ldots,m\}$ and $E = \{(i,j) \mid 1 \le i \le n, 1 \le j \le m, i \in A_j\}$. Now the subsets B of S, such that the induced sub graph $X[B] = (B,J,E')$ contains a matching that saturates B are exactly the independent sets of $M[\mathcal{A}]$. We study the reachability problem on a transversal matroid where we assume that the input is given as a bipartite graph $X = (S,J,E)$.

Problem 1. (TMR) Given an undirected bipartite graph $X = (S,J,E)$, a basis of the transversal matroid defined by X and a pair of vertices $s,t \in S$, test if there is a circuit in the transversal matroid M defined by X, that contains both s,t. We call this problem TMR (short for Transversal Matroid Reachability).

Note that applying Theorem 3.1 for TMR requires queries to the bipartite matching. In the following, we prove that TMR is NL-complete under log-space many-one reductions.

We first define the notion of strict gammoids that we use in the proofs. Let $G = (V,E)$ be a directed graph, and let $B \subseteq V$. Now define $L(G,B)$ as the set of all subsets of vertices of V that are matched into B, i.e,

$$L(G,B) = \left\{ B' \subseteq V \;\middle|\; \begin{array}{l} \exists \text{ an injection } \pi : B' \to B \text{ such that for all } v \in B', \\ G \text{ contains edge-disjoint paths between } v \text{ and } \pi(v) \end{array} \right\}$$

It is known that $L(G,B)$ forms the independent sets of a matroid M called the strict gammoid studied by Ingleton and Piff [8] (see also [3]).

We start with a reduction from the problem of testing st-connectivity in directed graphs (STCON) to TMR.

Lemma 4.1. STCON *is log-space many-one reducible to* TMR

Proof. Let $G = (V,E)$ be the given directed graph and s and t be the designated start and terminal nodes. Without loss of generality, we assume that G is acyclic (the general version can be reduced to this case, see [9]) and that there are no outgoing edges from t. (removing outgoing edges from t does not change reachability from s to t.) Let $G' = (V',E')$ be the graph obtained from G as follows: choose, $k > 2$. $V' = V \cup \{t_1,\ldots,t_k\}$. $E' = E \cup \{(t,t_1),\ldots,(t_{k-1},t_k)\}$.

Now choose $B = \{t,t_1,\ldots,t_k\}$, and consider the strict gammoid M defined by the sets of independent sets $L(G',B)$ for this particular B. First we establish the following connection with st- connectivity in the original graph G.

Claim 4.2. *M has a hyperplane avoiding s and t $\iff$ G has a directed s to t path.*

Proof. Suppose there is an s to t path in G. Then we construct a hyperplane Q in M as follows: Q contains $t_1,t_2,\ldots,t_k$ and all the vertices in V except $s,v_1,\ldots,v_r$ where there is a directed v_i-to-t path in G. By the definition of M, Q is a hyperplane as it is dependent and that adding any of t,s, $v_1,\ldots,v_r$ increases the rank of Q.

Conversely, suppose there is hyperplane Q that avoids both s and t. Clearly, $\{t_1,\ldots,t_k\} \subseteq Q$. As $s \notin Q$, adding s to Q must increase the rank of Q. This can happen only if there is an s to t path in G. □

We use the following fact [8] from matroid theory.

Strict gammoids are precisely the duals of transversal matroids. Thus, testing reachability between the elements s and t in the matroid M^* is equivalent to testing if there is a hyperplane avoiding s and t in the matroid M and hence by the above claim, is equivalent to testing if there is a directed path from s to t.

To proceed further we need to give the description of the transversal matroid M^* in deterministic log-space. For this, we construct a bipartite graph $X = (U,J,F)$ based on G' as follows: $U = V'$, $J = \{\widehat{v} \mid v \in V'\}$ and $F = \{(v,\widehat{v}) \mid v \in V'\} \cup \{(u,\widehat{v}) \mid (u,v) \in E'\}$. By construction (see Theorem 2.4.4 in [15]), M^*, the dual of the strict gammoid M is the transversal matroid defined by the induced sub graph $X' = (U,J \setminus \widehat{B},F)$ of X, where $\widehat{B} = \{\widehat{b} \mid b \in B\}$. It is easy to see that X' can be obtained in deterministic log space when G is given as the input. Observe that $V - \{t\}$ is a basis that we will require to produce as per the definition of TMR problem. Hence the reduction follows. □

The above reduction shows that TMR is hard for NL under log-space many-one reductions. We remark that it is possible to obtain a linear representation of the matroid M^* over the field of fractions of the ring of formal power series in indeterminates. To obtain this simply define the matrix where (i,j) entry is $-x_{ij}$ if $(i,j) \in F$(of the graph X') and 0 otherwise, and the resulting matrix represents M^*(see also [3]). This gives a hardness result for transversal matroids when the input is a linear representation as well.

Now we turn to upper bounds. We use the fact that we are given a basis of the matroid as a part of the input to obtain an NL upper bound.

Lemma 4.3. TMR *log-space Turing reduces to* STCON.

Proof. Let N be the input transversal matroid represented by the bipartite graph $X = (U,V,E)$ and a matching T where U(say) represents the ground set of N. Let s, $t \in U$ be the two designated elements whose reachability in N is to be tested. We start with a construction from [15]. Let $B = U \setminus B_0$ be the basis of N corresponding to the given matching T. Let Q be a matching in X in the sub graph induced by

$U \setminus B_0$. (Note that such a matching exists and it matches all the vertices in $U \setminus B_0$.) Let $J \subseteq V$ be the vertices that are matched to $U \setminus B_0$. If $j \in J$ is joined to $v \in U \setminus B_0$ in the matching Q, then relabel j as $\widehat{v}$. Now, for each $u \in B_0$ add a new vertex $\widehat{u}$ to V and connect $\widehat{u}$ to u and none else. Let X' be the new graph obtained from this relabelling and addition of vertices.

Now construct a directed graph $G = (U, F)$, where the directed edge $(u, v) \in F$ if and only if the edge $(u, \widehat{v})$ is present in X'. Let $L(G, B_0)$ be defined as in the proof of Lemma 4.1. As $L(G, B_0)$ defines the set of independent sets of the matroid N^* which is the dual of N. Thus, N has a circuit containing s and t if and only if N^* has a hyperplane that avoids both s and t. Let H be any hyperplane of N^*. By definition, $rank(H) = |B_0| - 1$. (note that $rank(N^*) = |B_0|$.) Hence there exists a vertex $v \in B_0$ such that for all $u \in H$, v is not reachable (via a directed path) from u and for all $w \in U \setminus H$, there is a directed path from w to v in G. Hence we can conclude that N^* contains a hyperplane avoiding s and t if and only if there exists $v \in B_0$ such that there are directed paths from s to v and t to v in the graph G. This latter condition can be tested in NL once we obtain the graph G. The graph G can be obtained by trivial computation from the given bipartite graph. □

Combining Lemmas 4.1 and 4.3 we have:

Theorem 4.4. TMR *is complete for* NL *under* log*-space many-one reductions.*

If we are not given the basis B which is essentially a matching in the given bipartite graph, we need to invoke known algorithms [10] to find one. The rest of the computation can be done in NL. This relaxes our upper bound in the case of the general version of the transversal matroid reachability. We note this as a corollary.

Corollary 4.5. *Given a bipartite graph $X(U, V, E)$ and two vertices $s, t \in U$, the problem of testing if s is reachable from t in the transversal matroid M defined by X is in* $\mathsf{RNC}^2 \cap \mathsf{P}$.

5 Acknowledgements

We thank Kristoffer Hansen for pointing out an error in an earlier proof of Theorem 3.3 and the anonymous referee for pointing out a simple proof of Lemma 3.6. We also thank Meena Mahajan for helpful suggestions and pointers. We thank the anonymous referee who pointed out the possible simplification of our proofs as presented in section 3. Part of the work was done when the second author was a postdoctoral researcher at the Institute for Theoretical Computer Science, Tsinghua University, Beijing, and was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

References

[1] Eric Allender. Reachability Problems: An Update. In *Computability in Europe (CiE 2007)*, pages 25–27, 2007. 2

[2] Eric Allender, Robert Beals, and Mitsunori Ogihara. The Complexity of Matrix Rank and Feasible System of Linear Equations. *Computational Complexity 8(2): 99-126*, 8(2):99–126, 1999. STOC 1996. 4, 13

[3] Federico Ardila. Tranversal and Co-tranversal Matroids via their Representations. In *Proceedings of 19th International Conference on Formal Power Series and Algebraic Combinatorics*, 2007. 3, 14, 15

[4] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, April 2009. 4

[5] Stephen A. Cook and Pierre McKenzie. Problems Complete for Deterministic Logarithmic Space. *Journal of Algorithms*, 8(3):385–394, 1987. 6, 13

[6] J. de Sousa and D. J. A. Welsh. A Characterisation of Binary Transversal Structures. *Journal of Mathematical Analysis and Applications*, 40(1):55 – 59, 1972. 3

[7] Ulrich Hertrampf, Steffen Reith, and Heribert Vollmer. A Note on Closure Properties of Logspace mod Classes. *Information Processing Letters*, 75(3):91–93, 2000. 14

[8] A. Ingleton and M. Piff. Gammoids and Tranversal Matroids. *Journal of Combinatorial Theory Ser. B*, 15:51–68, 1973. 3, 14, 15

[9] Neil D. Jones. Space-bounded Reducibility among Combinatorial Problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975. 2, 15

[10] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a Perfect Matching is in Random NC. *Combinatorica*, 6(1):35–48, 1986. 3, 16

[11] L.R. Matthews. Bicircular Matroids. *Quarterly Journal of Mathematics (Oxford), Second Series*, 28:213–227, 1977. 7

[12] Dillon Mayhew. *Matroids and Complexity*. PhD thesis, University of Oxford, 2005. 5

[13] Dillon Mayhew. Matroid Complexity and Nonsuccinct Descriptions. *SIAM Journal of Discrete Mathematics*, 22(2):455–466, 2008. 5

[14] Milena Mihail and Madhu Sudan. Connectivity Properties of Matroids. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1992. 2

[15] James G. Oxley. *Matroid theory*. Oxford University Press, New York, 1992. 2, 3, 4, 15

[16] Omer Reingold. Undirected ST-connectivity in Log-space. *Journal of the Association for Computing Machinery (JACM)*, 55(4):17:1–17:24, September 2008. 2, 6, 11

[17] Walter J. Savitch. Maze Recognizing Automata and Nondeterministic Tape Complexity. *Journal of Computer and System Sciences*, 7(4):389–403, 1973. 2

[18] W. T. Tutte. Connectivity in Matroids. *Canadian Journal of Mathematics*, 18:1301–1324, 1966. 2, 4

[19] Hassler Whitney. On the Abstract Properties of Linear Dependence. *American Journal of Mathematics*, 57(3):509–533, 1935. 2