Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 9 (2009)

Checking Equality of Matroid Linear Representations and the Cycle Matching Problem*

T.C. Vijayaraghavan†

26th January 2010

Abstract

Given linear representations M_1 and M_2 of matroids over a field $\mathbb{F}$, we consider the problem (denoted by ECLR[$\mathbb{F}$]) of checking if M_1 and M_2 represent the same matroid over $\mathbb{F}$. When $\mathbb{F} = \mathbb{Z}_2$, we show that ECLR[$\mathbb{Z}_2$] is complete for $\oplus$L under logspace many-one reductions.

When $\mathbb{F} = \mathbb{Q}$, given linear representations $M_1, M_2 \in \mathbb{Q}^{m \times n}$ as input, any $X \subseteq \{1, \ldots, n\}$ such that columns corresponding to indexes in X form a minimal linearly dependent set of vectors in one linear representation but the column vectors corresponding to indexes in X are linearly independent in the other linear representation is a witness that M_1 and M_2 represent different matroids over $\mathbb{Q}$. We show that the decision and the search version of this problem are polynomial time equivalent.

We define the CYCLE MATCHING problem of checking if for a pair of undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ given as input with $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2|$, whether the set of vertices in $X \subseteq V_1$ form a cycle in G_1 if and only if the vertices in $X \subseteq V_2$ form a cycle in G_2 also, for all $X \subseteq \{1, \ldots, n\}$. We show that CYCLE MATCHING is complete for L.

Also the problem of counting the number of $X \subseteq \{1, \ldots, n\}$ such that vertices with indexes in X form a cycle in one of the input graphs but not in the other is shown to be #P-complete.

1 Matroid Linear Representations and ECLR

Matroids are combinatorial objects that generalize the notions of linear independence and dependence of vectors in a vector space. The study of computational problems related to matroids and providing efficient algorithms for solving them is an important branch of Combinatorial Optimization [15]. Matroids are also known to generalize connectivity properties between vertices in a graph. As a result several problems on graphs have been re-cast into problems on matroids to seek efficient algorithms for solving them. A classic example is the maximum matching problem for bipartite graphs which can be shown to reduce to the problem of intersection of two linearly representable matroids [15, Section 12.5]. Efficient algorithms for the matroid intersection problem (not necessarily linearly representable matroids) are known and in fact the first polynomial time algorithm for this problem was given by Edmonds in [6, 7].

As one of the main results in this paper, we consider a problem on linearly representable matroids denoted by ECLR. Given two linear representations over a field $\mathbb{F}$, the ECLR[$\mathbb{F}$] problem is to check if the input linear representations represent the same matroid over $\mathbb{F}$.

*This article is the Revision 2 of ECCC Report No.09(2009). Extended abstract of this article was submitted to ISAAC 2008 and STACS 2009.

†Email:vijay@cmi.ac.in,tcvijay@imsc.res.in

ISSN 1433-8092

Before we define the problem formally, we introduce necessary definitions and terminology on matroids. We refer to [13, Chapter 1] for details and clarifications.

2 Preliminaries

2.1 Matroids

Definition 2.1. *A matroid M is a pair $(S, \mathcal{I})$, where S is a finite set and $\mathcal{I}$ is a collection of subsets of S such that*

1. *the empty set $\emptyset$ is in $\mathcal{I}$,*
2. *if $X \in \mathcal{I}$ and $Y \subseteq X$ then $Y \in \mathcal{I}$,*
3. *if $X, Y \in \mathcal{I}$ with $|X| = |Y| + 1$, then there exists $x \in X - Y$ such that $Y \cup \{x\} \in \mathcal{I}$. This condition is also called the* independence augmentation axiom *of a matroid.*

We say that a set $X \subseteq S$ is independent if $X \in \mathcal{I}$. Any subset of S not in $\mathcal{I}$ is said to be a dependent set.

Definition 2.2. *Let $M = (S, \mathcal{I})$ be a matroid and let $X \in \mathcal{I}$. We say that X is a base if $X \not\subseteq Y$, for all $Y \in \mathcal{I}$ with $X \neq Y$. In other words a base is a maximal independent set of M.*

Definition 2.3. *Let $M = (S, \mathcal{I})$ be a matroid, and let $X \subseteq S$. We say that X is a circuit if $X \notin \mathcal{I}$, but every proper subset Y of X is in $\mathcal{I}$. In other words a circuit is a minimal dependent set of M.*

Definition 2.4. *Let $M = (S, \mathcal{I})$ be a matroid where $S = \{1, \ldots, n\}$ and let $\mathbb{F}$ be a field. We say that M is linearly representable over $\mathbb{F}$, if for some $r \in \mathbb{N}$ there exists a matrix $A \in \mathbb{F}^{r \times n}$ such that a set of columns $\{A_{i_1}, \ldots, A_{i_l}\}$ of A are linearly independent over $\mathbb{F}$ if and only if $\{i_1, \ldots, i_l\} \in \mathcal{I}$.*

We now recall some basic results from [13].

Lemma 2.5. [13, Lemma 1.2.2] *If X_1 and X_2 are two distinct bases of a matroid $M = (S, \mathcal{I})$ and there exists $x \in X_1 - X_2$ then we have $y \in X_2 - X_1$ such that $(X_1 - \{x\}) \cup \{y\}$ is also a base of M. This result is also called the* base exchange axiom *of a matroid.*

Theorem 2.6. [13, Theorem 1.2.3] *Let S be a set and let $\mathcal{B}$ be a collection of subsets of S such that $\mathcal{B}$ is non-empty and elements of $\mathcal{B}$ satisfy the base exchange axiom stated in Lemma 2.5. Now if $\mathcal{I}$ is the collection of all subsets of B for all $B \in \mathcal{B}$ then $M = (S, \mathcal{I})$ is a matroid.*

As a consequence of the above theorem we obtain the following result.

Corollary 2.7. *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids such that any set $B \subseteq S$ is a base in M_1 if and only if B is a base in M_2 also. Then both M_1 and M_2 are the same matroid.*

2.2 Logspace Counting Classes

Definition 2.8. [1] *Let $\Sigma = \{0, 1\}$. We say that a function $f : \Sigma^* \to \mathbb{Z}^+$ is in $\#\mathrm{L}$ if there exists a* NL *machine M such that $f(x)$ is the number of accepting computation paths of M on input x.*

Definition 2.9. [4] *Let* $\Sigma = \{0,1\}$ *and* $L \subseteq \Sigma^*$. *We say that a language* $L \in \oplus\mathrm{L}$ *if there exists* $f \in \#\mathrm{L}$ *such that on any input* $x \in \Sigma^*$ *we have* $x \in L$ *if and only if* $f(x) \equiv 1(mod\ 2)$.

Definition 2.10. [8] *Let* $\Sigma = \{0,1\}$ *and let* $\oplus\mathrm{LH}_1 = \oplus\mathrm{L}$. *For* $i \geq 2$ *we say that a language* $L \in \oplus\mathrm{LH}_i$ *if there exists a* NL *machine* M^A *that access a language* $A \in \oplus\mathrm{LH}_{i-1}$ *as an oracle such that on input* $x \in \Sigma^*$, *if* $f(x)$ *denotes the number of accepting computation paths of* M^A *on input* x *then we have* $x \in L$ *if and only if* $f(x) \equiv 1(mod\ 2)$.

We follow the Ruzzo-Simon-Tompa oracle access mechanism [1, 8] *in allowing non-deterministic Turing machines to query oracles. According to this mechanism a non-deterministic Turing machine is allowed to write its queries in the oracle tape in a deterministic manner only.*

We also need the following result from [8].

Theorem 2.11. [8] $\oplus\mathrm{L}^{\oplus\mathrm{L}} = \oplus\mathrm{L}$.

3 ECLR: A problem on matroid linear representations

If M is a matroid linearly representable over a field $\mathbb{F}$ then its linear representation need not be unique. For example the matroid represented by the $n \times n$ identity matrix I_n is the same as the matroid represented by any other $n \times n$ non-singular matrix over a field $\mathbb{F}$. Therefore the problem of deciding if two input linear representations over a field $\mathbb{F}$ represent the same matroid seems very fundamental and interesting. This problem is denoted by ECLR[$\mathbb{F}$] and it is defined as follows.

> **Equality Checking for Linear Representations** (ECLR): Given linear representations $M_1, M_2 \in \mathbb{F}^{m \times n}$, where $\mathbb{F}$ is a field, the ECLR[$\mathbb{F}$] problem is to decide if the matroid represented by M_1 is the same as the matroid represented by M_2 over $\mathbb{F}$.

Note 1. *In* ECLR *we deal with checking if two given linear representations over a field* $\mathbb{F}$ *represent the same matroid. However note that a notion of* equivalence *of two linear representations is also known* [13, Section 6.3]. *We wish to emphasize that in our* ECLR *problem we check if the matroids represented by the input linear representations are the same and not if they are* equivalent *as defined in* [13].

If we assume the basic operations in $\mathbb{F}$ are polynomial time computable then the most obvious complexity upper bound for ECLR[$\mathbb{F}$] is coNP. As a result the question of determining if ECLR[$\mathbb{F}$] has a polynomial time algorithm over any such field $\mathbb{F}$ seems very natural and interesting. Surprisingly however ECLR[$\mathbb{F}$] has not received any attention so far[1] and we do not know of any algorithmic time upper bound or complexity upper bound for this problem.

In Theorem 3.4 we show that ECLR[$\mathbb{Z}_2$] is $\oplus$L-complete. Since $\oplus\mathrm{L} \subseteq \mathrm{NC} \subseteq \mathrm{P}$ our result initiates a rigorous study of showing that ECLR[$\mathbb{F}$] $\in$ P over an arbitrary field $\mathbb{F}$ and in settling the complexity of this problem precisely. In view of this we pose the following question.

> **Open problem:** Let $(\mathbb{F}, +, \circ)$ be a field such that given $\alpha, \beta \in \mathbb{F}$ we can compute $(\alpha + \beta)$, $(\alpha \circ \beta)$ and α^{-1} using space at most $O(\log(|\alpha| + |\beta|))$. Then the question is whether ECLR[$\mathbb{F}$] $\in$ NC. In other words assume that $\mathbb{F}$ is a field such that we can carry out the basic operations involving elements $\alpha, \beta \in \mathbb{F}$ using space at most $O(\log(|\alpha|+|\beta|))$. If $M_1, M_2 \in \mathbb{F}^{m \times n}$ are input linear representations of matroids over $\mathbb{F}$ then the question is to determine if we can decide whether M_1 and M_2 represent the same matroid over $\mathbb{F}$ in NC.

[1] I have defined this problem in my Ph.D. thesis [18].

We believe that ECLR[$\mathbb{F}$] over a field $\mathbb{F}$ as described above indeed has a NC upper bound. In fact we believe that if $\mathbb{F}$ has charcteristic 2 then ECLR[$\mathbb{F}$] $\in$ NC^4.

3.1 Row reduction

We now prove a row reduction result on linear representations of a matroid. Let $M = (S, \mathcal{I})$ be a matroid such that every element of S is in at least one independent set in $\mathcal{I}$. In Theorem 3.1 to be proved, we show that if M is linearly representable by a matrix $A \in \mathbb{F}^{m \times n}$ over a field $\mathbb{F}$ then M is also linearly representable by a submatrix $B \in \mathbb{F}^{r \times n}$ of A such that $r =$ rank $(B) =$ rank (A).

Theorem 3.1. *Let $\mathbb{F}$ be a field and let $A \in \mathbb{F}^{m \times n}$ be a linear representation of the matroid $M = (S, \mathcal{I})$ over $\mathbb{F}$. We assume that every element of S is in at least one independent set in $\mathcal{I}$. Also if $r =$ rank (A) and $B \in \mathbb{F}^{r \times n}$ is a submatrix of A such that $r =$ rank (B) then B is also a linear representation of the matroid M.*

Proof: We have assumed that every element of S is in at least one independent set in M. When we have $r =$ rank $(A) = n$ it is clear that there exists a base of M that contains all the elements in S. Also it is easy to note that this is the only base in M. But the submatrix B is now a $r \times r$ non-singular submatrix of A over $\mathbb{F}$. As a result the matroid represented by B also contains only one base and this base contains all the elements of S. Now using Corollary 2.7 it follows that the matroid represented by B is also M.

Before we prove the result for the case when $r < n$ we prove the following claim.

Claim 3.2. *The matrix $B \in \mathbb{F}^{r \times n}$ does not contain any column containing only zeroes.*

Proof of Claim 3.2. Assume that the claim is not true and that the i^{th} column of B, denoted by B_i contains only zeroes for some $1 \leq i \leq n$. We have assumed that every element of S is in at least one independent set in $\mathcal{I}$ of the matroid M. Therefore every column of the linear representation A has at least one non-zero element from $\mathbb{F}$. As a result the i^{th} column of A, denoted by A_i, is non-zero. Now if B_i denotes the i^{th} column of B then since B is submatrix of A it follows that B_i is contained in the column A_i. Moreover rank $(B) = r$ and so there exists a $r \times r$ non-singular submatrix B' of B. Since we have assumed the column B_i contains only zeros it follows that B_i is not in B' and therefore using B' and the column A_i we can then obtain a $(r+1) \times (r+1)$ non-singular submatrix of A. But this contradicts our assumption that $r =$ rank (A). This shows that the column B_i in B contains at least one non-zero entry from $\mathbb{F}$ which proves our claim.

Now assume that $n = |S|$ and that $r < n$. We use induction on n to prove our result. Since $1 \leq r < n$ it follows that $n \geq 2$. Let us consider the base case when $n = 2$ and $r = 1$. Using Claim 3.2 it follows that every column in A contains at least one non-zero entry from $\mathbb{F}$. As a result since $r = 1$ when the number of non-zero rows in A is 1 the row is of the form (α_1, α_2) where $\alpha_1, \alpha_2 \in \mathbb{F}$ and $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$. In this case using Claim 3.2 it follows that B is this non-zero row. Otherwise every row in A is of the form (α, α), where $\alpha \in \mathbb{F}$ and using Claim 3.2 it follows that there are at least 2 non-zero rows in A. As a result we get B to be one of these non-zero rows of A and so the result is true for the case when $n = 2$. By extending these arguments we can show the result to be true if $r = 1$ in a matroid M whenever $n = |S| \geq 2$. This proves the result for all $n \geq 1$ when $r = 1$.

We now use induction on r to complete the proof. Let us inductively assume that if $M = (S, \mathcal{I})$ is a matroid such that $n = |S| \geq 3$ and that M has a linear representation $A \in \mathbb{F}^{m \times n}$ with $1 \leq r =$ rank $(A) \leq (n-2)$ then M is also linearly representable by a submatrix $B \in \mathbb{F}^{r \times n}$ of A such that rank $(B) = r$.

Let $A \in \mathbb{F}^{m \times n}$ be a linear representation of a matroid $M = (S, \mathcal{I})$ over $\mathbb{F}$ such that $n = |S|$ and $r = \text{rank } (A) = (n-1)$. Let $B \in \mathbb{F}^{r \times n}$ be a submatrix of A with rank $(B) = r$ as mentioned in the statement. We need to show that B also represents M over $\mathbb{F}$.

Since $r < n$ using Definition 2.4 and Definition 2.2 we observe that there exists $i \in S$ and a base B of M such that $i \notin B$. We now consider the submatrix A'_i of A obtained by deleting A_i, the i^{th} column of A, from A. It is then clear that the matroid represented by A'_i also has a base of size r. It is obvious that rank $(A'_i) \leq$ rank (A) and so rank $(A'_i) = r$.

Using Claim 3.2 we obtain that every column of B contains at least one non-zero element in $\mathbb{F}$. Similar to the linear representation A, let B_i denote the i^{th} column of B and let B'_i be the submatrix of B obtained by deleting B_i from B. We have $r =$ rank (B) and therefore similar to A'_i we also have rank $(B'_i) = r$. It is easy to note that B'_i is a submatrix of A'_i. Moreover rank $(A'_i) =$ rank $(B'_i) = r$ and the matroid represented by A'_i and the matroid represented by B'_i contain $(n-1)$ elements. Therefore using induction on $|(S - \{i\})| = (n-1)$ it follows that A'_i and B'_i represent the same matroid. Let us call this matroid M_i.

Since we have obtained A'_i by deleting A_i from A, it follows that any base of M is either a base of M_i or it can be obtained from a base of M_i by applying the base exchange axiom stated in Lemma 2.5 to a base of M_i with the element $i \in S$. Also M does not have any other base. But this observation obtained using Lemma 2.5 is also true for the matroid represented by B over $\mathbb{F}$ and the matroid represented by the submatrix B_i of B over $\mathbb{F}$. We also know that the matroid represented by A_i and the matroid represented by B_i is M_i. As a result we obtain that B is also a linear representation of the matroid M. This completes the proof of our result on row reduction for a linear representation of a matroid.

3.2 ECLR[$\mathbb{Z}_2$] is $\oplus$L-complete

We consider the ECLR problem when $\mathbb{F} = \mathbb{Z}_2$. Using results on linear algebraic subroutines such as computing a maximal set of linearly independent vectors from a given set of vectors over $\mathbb{Z}_2$ and a solving system of linear equations over $\mathbb{Z}_2$ from [4] and properties of $\oplus$L from [8] we show that ECLR[$\mathbb{Z}_2$] is $\oplus$L-complete under logspace many-one reductions.

Computing the number of spanning trees modulo 2 of an arbitrary undirected graph has been recently shown to be complete for $\oplus$L under logspace many-one reductions in [3]. Also in [3] computing the permanent of an integer matrix modulo 2^k for a fixed integer $k > 0$ is also shown to be complete for $\oplus$L under logspace many-one reductions. The completeness result for ECLR[$\mathbb{Z}_2$] that we obtain in Theorem 3.4 is a new addition to this list of problems complete for $\oplus$L.

We first show an equivalence relation that characterizes when two linear representations over $\mathbb{Z}_2$ represent the same matroid.

Theorem 3.3. *Given matrices $M_1, M_2 \in \mathbb{Z}_2^{m \times n}$ such that rank (M_1) = rank $(M_2) = m$, we say that M_1 and M_2 are related (denoted by $M_1 \sim M_2$) if there exists an invertible matrix $X \in \mathbb{Z}_2^{m \times m}$ such that $XM_1 = M_2$. Then*

1. *$\sim$ is an equivalence relation, and*
2. *M_1 and M_2 represent the same matroid M over $\mathbb{Z}_2$ if and only if $M_1 \sim M_2$.*

Proof: Let $M_1, M_2 \in \mathbb{Z}_2^{m \times n}$ such that rank $(M_1) =$ rank $(M_2) = m$.

1. It is easy to note that the above relation is reflexive: for any matrix $M \in \mathbb{Z}_2^{m \times n}$ we can take X to be I_m, the $m \times m$ identity matrix. Also $\sim$ is symmetric, since $XM_1 = M_2$ for an invertible matrix X if and only if $M_1 = X^{-1}M_2$. Transitivity follows since given

M_1, M_2 and M_3, where $M_3 \in \mathbb{Z}_2^{m\times n}$ with rank $(M_3) = m$, if $M_1 \sim M_2$ and $M_2 \sim M_3$ then there exists invertible matrices $X_1, X_2 \in \mathbb{Z}_2^{m\times m}$ such that $X_1M_1 = M_2$ and $X_2M_2 = M_3$. Now let $X_3 = X_2X_1$. Then $X_3 \in \mathbb{Z}_2^{m\times m}$ and is invertible. Moreover $X_3M_1 = M_3$ which implies $M_1 \sim M_3$.

2. Let $M_1, M_2 \in \mathbb{Z}_2^{m\times n}$ be linear representations having rank (M_1) = rank $(M_2) = m$ representing the same matroid M. We can identify the columns corresponding to the lexicographically least base containing m elements of the matroid M from these matrices. Let $Y_1, Y_2 \in \mathbb{Z}_2^{m\times m}$ be submatrices of M_1 and M_2 respectively corresponding to this base. Clearly Y_1 and Y_2 are invertible. Thus there exists an invertible matrix $X \in \mathbb{Z}_2^{m\times m}$ such that $XY_1 = Y_2$. More precisely we have $X = Y_2Y_1^{-1}$. Now any column in M_1 can be written as a $\mathbb{Z}_2$-linear combination of columns in Y_1 in a unique way. Also given any set of linearly independent vectors over $\mathbb{Z}_2$, there is exactly one vector in their $\mathbb{Z}_2$ span when all the coefficients of this set of vectors are non-zero. We have also assumed M_1 and M_2 represent the same matroid. As a result any set of columns in M_1 form a circuit if and only if the corresponding set of columns also form a circuit in M_2. Therefore from these observations it follows that $XM_1 = M_2$.

   Conversely if for $M_1, M_2 \in \mathbb{Z}_2^{m\times n}$ we have $M_1 \sim M_2$, then M_1 and M_2 represent the same matroid. This is a consequence of the fact that the set of vectors obtained by computing the product of an invertible matrix with a set of linearly independent vectors is also linearly independent.

Remark 1. *It follows from Theorem 3.3 that the number of equivalence classes under the relation $\sim$ between the linear representations of a matroid M over $\mathbb{Z}_2$ is only one. This observation about the number of equivalence classes is true only for $\mathbb{Z}_2$. This is important since the problem of deciding* ECLR$[\mathbb{Z}_2]$ *reduces to solving a system of linear equations over $\mathbb{Z}_2$. It is easy to see that when considering linear representations of a matroid M over any other field (even with characteristic* 2*) number of equivalence classes under the relation $\sim$ defined in Theorem 3.3 is more than one. We do not know if there is a polynomial time algorithm to solve the* ECLR *problem when the input linear representations are over a field that is not $\mathbb{Z}_2$.*

Theorem 3.4. ECLR$[\mathbb{Z}_2]$ *is* $\oplus$L*-complete.*

Proof: We are given linear representations $M_1, M_2 \in \mathbb{Z}_2^{m\times n}$ as input. In [4] it has been shown that computing a maximal set of linearly independent vectors over $\mathbb{Z}_2$ from a given set of vectors over $\mathbb{Z}_2$ is complete for $\oplus$L under logspace many-one reductions. As a result we can compute rank (M_1) and rank (M_2) in $\oplus$L and check if they are equal. If rank $(M_1) \neq$ rank (M_2) then we output M_1 and M_2 do not represent the same matroid and stop.

Otherwise assume that r = rank (M_1) = rank (M_2). Now once again it follows from [4] that we can obtain a submatrix $A_1 \in \mathbb{Z}_2^{r\times n}$ of M_1 and a submatrix $A_2 \in \mathbb{Z}_2^{r\times n}$ of M_2 such that rank (A_1) = rank $(A_2) = r$ in $\oplus$L. Since the linear representations are over $\mathbb{Z}_2$ using Theorem 3.1 it follows that the matroid represented by A_i is the same as the matroid represented by M_i, for $i = 1, 2$ respectively.

Given these linear representations A_1 and A_2 using Theorem 3.3 it follows that A_1 and A_2 represent the same matroid if and only if there exists a non-singular matrix $X \in \mathbb{Z}_2^{m\times m}$ such that $XA_1 = A_2 (\text{mod } 2)$. Essentially this step is to solve for a system of linear equations over $\mathbb{Z}_2$ which is also shown to be complete for $\oplus$L in [4].

It is easy to see that given the linear representations M_1 and M_2 as input, a logspace machine with access to a $\oplus\text{LH}_3$ oracle can obtain the submatrices A_1 and A_2 and also check for the existence of a matrix $X \in \mathbb{Z}_2^{m\times m}$ such that $XA_1 = A_2$ over $\mathbb{Z}_2$. Since rank (A_1) = rank (A_2)

in $\mathbb{Z}_2$ it follows that X is non-singular. Now using these observations it follows from Theorem 3.3 and Theorem 2.11 that ECLR$[\mathbb{Z}_2] \in \oplus$L. This shows the $\oplus$L upper bound.

Hardness for $\oplus$L follows from the following observation: given a matrix $M \in \mathbb{Z}_2^{n\times n}$ checking if M is non-singular over $\mathbb{Z}_2$ is hard for $\oplus$L [4]. Therefore given a matrix $M \in \mathbb{Z}_2^{n\times n}$ as input we output M and I_n, where I_n is the $n \times n$ identity matrix. Clearly $(M, I_n) \in$ ECLR$[\mathbb{Z}_2]$ if and only if M is non-singular which shows the hardness for $\oplus$L and hence the proof is complete.

3.3 An equivalence between search and decision for ECLR$[\mathbb{Q}]$

In this section we consider the ECLR problem when the input linear representations are over $\mathbb{Q}$, the set of rationals. Let $M_1, M_2 \in \mathbb{Q}^{m\times n}$ be the input linear representations. Given a set of indexes, columns corresponding to which are linearly dependent in one linear representation but are linearly independent in the other linear representation is a witness that M_1 and M_2 represent different matroids over $\mathbb{Q}$. We show that the search and the decision version of this problem are polynomial time equivalent. More precisely, assume that there is a polynomial time algorithm that decides ECLR$[\mathbb{Q}]$ and that the input linear representations M_1 and M_2 represent different matroids. The polynomial time procedure described below outputs a set of indexes such that columns corresponding to these indexes form a circuit in M_i but the corresponding columns do not form a circuit in M_j using ECLR(M_1, M_2) as an oracle, where $1 \leq i, j \leq 2$ and $i \neq j$.

Given linear representations $M_1, M_2 \in \mathbb{Q}^{m\times n}$, let ECLR$(M_1, M_2)$ be the function that outputs 1 if the matroid represented by M_1 and M_2 are the same, and it outputs 0 otherwise. Also if $X \subseteq S = \{1, \ldots, n\}$ and $j \in \{1, 2\}$, let $M_j^{(X)}$ denote the submatrix of M_j obtained by retaining columns whose indexes are in X and deleting any other column whose indexes is not in X. We denote the matroid so obtained from M_j by $(S, \mathcal{I}_j^{(X)})$, where $\mathcal{I}_j^{(X)} = \{X \cap I | \text{for all } I \in \mathcal{I}_j\}$. We start by assuming that M_1 and M_2 represent different matroids.

Let $i = 1$, $X = \{1, \ldots, i\}$, and $Y = \emptyset$. We query the ECLR$(M_1^{(X)}, M_2^{(X)})$ oracle for increasing values of i until the oracle outputs 0 for the smallest $1 \leq i \leq n$. Once we obtain this element $i \in S$ we re-initialize $X = \{1, \ldots, (i-1)\}$ and $Y = Y \cup \{i\}$. If the columns corresponding to elements in Y form a circuit in M_k but are linearly independent in M_l, where $1 \leq k, l \leq 2$ with $k \neq l$, then we output Y and stop.

Otherwise if the set Y does not witness that the input linear representations represent different matroids then we find the smallest $j \in \{1, \ldots, (i-1)\}$ such that the ECLR$(M_1^{(X'\cup Y)}, M_2^{(X'\cup Y)})$ oracle outputs 0, where $X' = \{1, \ldots, j\}$. Once we obtain this $j \in S$ we once again re-initialize $X = X'$ and $Y = Y \cup \{j\}$. Whenever we augment a new element to Y, we also check if the columns corresponding to elements in Y form a circuit in M_k but are linearly independent in M_l, where $1 \leq k, l \leq 2$ with $k \neq l$. We iterate this step of augmenting elements to Y until we obtain the desired set $Y \subseteq \{1, \ldots, i\} \subseteq S$ that forms a circuit in one linear representation but not in the other and hence witnesses that the input linear representations represent different matroids over $\mathbb{Q}$.

The steps given above involve retaining some set of columns of the given input matrices and querying the ECLR oracle. Clearly these steps are polynomial time computable and hence the claim follows.

4 Cycle Matching Problem

The Cycle Matching Problem is defined as follows.

CYCLE MATCHING: Given a pair of undirected graphs $G_1 = (V_1, E_1)$ and $G_2 =$

(V_2, E_2) as input with $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2|$, the CYCLE MATCHING problem is to check if the set of vertices in $X \subseteq V_1$ form a cycle in G_1 if and only if the set of vertices in $X \subseteq V_2$ form a cycle in G_2 also, for all $X \subseteq \{1, \ldots, n\}$.

It is not hard to view the CYCLE MATCHING problem as the graph theoretic analogue of the ECLR problem. This follows from the fact that we can associate a linearly representable matroid over any given field $\mathbb{F}$ to a given simple undirected graph. The matroid so obtained is called a *cycle matroid* or a *graphic matroid* [13, pages 11 and 12]. Here the edges of the graph are the underlying elements and a set of edges form a cycle if and only if the corresponding columns of the linear representation form a minimal linearly dependent set of vectors over $\mathbb{F}$. In fact the incidence matrix of a simple undirected graph G is itself a linear representation for the cycle matroid of G over $\mathbb{Z}_2$. Thus it seems natural to expect a $\oplus$L upper bound for CYCLE MATCHING. However as one of the main results we show that the CYCLE MATCHING problem is in fact complete for L. We obtain the upper bound of L for CYCLE MATCHING in Theorem 4.22. We show that CYCLE MATCHING is hard for L in Theorem 4.24.

Assume that we are given a pair of simple undirected connected graphs (G_1, G_2) that do not have any cut-vertices as input to the CYCLE MATCHING problem. If (G_1, G_2) is a "yes" instance of CYCLE MATCHING then it turns out that G_1 and G_2 are also isomorphic. Given such a "yes" instance (G_1, G_2) of CYCLE MATCHING we identify a subset of the set of all isomorphisms from G_1 to G_2 and call these isomorphisms as *Cymatch Isomorphisms* of G_1 and G_2. We also say that G_1 and G_2 are *cymatch isomorphic*. The definition of a cymatch isomorphism along with related results leading to the proof of the theorem we have stated above is proved in Section 4.2 and Theorem 4.19. Due to the upper bound result that CYCLE MATCHING $\in$ L shown in Theorem 4.22 we are therefore able to show that it is possible to decide if there exists a cymatch isomorphism from G_1 to G_2 (and also compute a cymatch isomorphism from G_1 to G_2) in L in Corollary 4.23.

It should be remarked that for a cymatch isomorphism to exist between a given pair of input graphs, every two vertices in each of the input graphs need be in at least one cycle and the input graphs should also agree on all subsets of vertices that form a cycle (giving a rigorous proof of this statement is the main objective for the rest of this paper). Since the condition stated above demands the input graphs to agree on the adjacency of vertices to a large extent, it is not surprising that there exists an isomorphism between the given input graphs when they satisfy the condition stated above. While [12, 10, 9] prove results showing that the Graph Isomorphism problem for certain restricted graph classes are complete for L, using our results on CYCLE MATCHING we are able to identify yet another non-trivial class of pairs of undirected graphs[2] for which we can decide Graph Isomorphism in L. (Also Köbler in [11] gives a recent survey of results on the Graph Isomorphism problem for restricted graph classes that are decidable in NC.)

Reingold in [16] presents a $O(\log n)$ space algorithm for the undirected *st*-connectivity problem and as a consequence it is also shown in [16] that SL = L. Due to this result it follows that the list of problems in [2] that are shown to be complete for SL are also complete for L. Therefore the results obtained in [5, 12, 10, 2, 9] give an almost complete list of problems that are known to be L-complete. It seems that the CYCLE MATCHING problem has not received any attention so far and our result that CYCLE MATCHING is L-complete shown in Section 4.4 and other related results we obtain are new and unknown.

[2]The precise statement would assume a promise on the input pair of undirected graphs (G_1, G_2) so that G_1 and G_2 are isomorphic if and only if they are cymatch isomorphic.

Definition 4.1. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be a pair of undirected graphs with $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2|$ such that (G_1, G_2) is a "no" instance of* CYCLE MATCHING. *We say that a vertex $x \in V_1$ causes cycle mismatch if there exists $S \subseteq \{1, \ldots, n\}$ such that $x \in S$ and the vertices in S form a cycle in G_i but the vertices in S do not form a cycle in G_j, where $1 \leq i, j \leq 2$ and $i \neq j$.*

4.1 Preliminaries

Definition 4.2. *A cut-edge in an undirected graph G is an edge whose deletion from the graph increases the number of components in G.*

Definition 4.3. *A cut-vertex in an undirected graph G is a vertex whose deletion from the graph increases the number of components in G.*

Theorem 4.4. [19, Theorem 1.2.14] *Given an undirected graph G, an edge is a cut-edge if and only if it does not belongs to any cycle in G.*

Corollary 4.5. *Let G be a undirected graph. Then G does not contain cut-edges if and only if for every edge in G is contained in at least one cycle in G.*

Proof: Let G be an undirected graph. Now if G does not contain any cut-edge then it follows from Theorem 4.4 that every edge in G is contained in at least one cycle in G. Conversely if every edge in G is always contained in at least one cycle in G then once again using Theorem 4.4 it follows that any edge in G is not a cut-edge.

Note 2. *If G is an undirected graph then the subgraph of G formed by those vertices which are the end vertices of edges that are not cut-edges is unique.*

Lemma 4.6. *Let $G = (V, E)$ be an undirected graph that does not contain cut-edges and isolated vertices. Also assume that $|V| \geq 3$ and let $u, v \in V$. Then there exists a cycle containing u and v in G if and only if there exists a path connecting u and v in $G' = (V', E')$ where $V' = V - \{w\}$ and $E' = E - \{(w, x) | x \in V\}$ for any $w \in V - \{u, v\}$.*

Proof: Assume that there exists a cycle containing u and v in G. From this cycle we can then obtain two paths connecting u and v. Now for any $w \in V - \{u, v\}$ if w is not in the cycle containing u and v then the result is obviously true. Otherwise if w is a vertex in the cycle then we can always choose the path in G connecting u and v that does not contain w. Clearly this path exists in G' also.

Conversely assume that there always exists a path connecting u and v in $G' = (V', E')$ where $V' = V - \{w\}$ and $E' = E - \{(w, x) | x \in V\}$, for any $w \in V - \{u, v\}$. We use induction on the length of a shortest path connecting u and v in G' to prove the result. For the case when this path in G' is the edge (u, v), since (u, v) is not a cut-edge in G using Corollary 4.5 it follows that there exists a cycle containing u and v in G.

Now inductively assume that if the length of the shortest path connecting u and v in G' is $(l - 1)$ for $l \geq 2$, then there exists a cycle containing u and v in G.

Let P be a path of length $l \geq 2$ connecting u and v in G'. Let w_1 be the neighbour of u in P and let w_2 be the neighbour of v in P. Since there exists a path of length less than or equal to $(l - 1)$ connecting u and w_2 in G', using induction on $(l - 1)$ it follows that there exists a cycle containing vertices u and w_2 in G. Similarly there exists a cycle containing w_1 and v in G. If $w_1 \neq w_2$ then it is easy to construct a cycle containing u and v from the edges $(u, w_1), (w_2, v)$ and the edges in these two cycles. When $w_1 = w_2$ using our assumption about G it follows that there exists a path connecting u and v in $G'' = (V'', E'')$ where $V'' = V - \{w_1\}$

and $E'' = E - \{(w_1, x) | x \in V\}$. Now using the shortest path connecting u and v in G'' and the edges $(u, w_1), (w_1, v) \in E$ we can obtain a cycle containing u and v in G. This completes the proof.

As a consequence we obtain the following result.

Corollary 4.7. *Let $G = (V, E)$ be an undirected graph that does not contain cut-edges and isolated vertices such that $|V| \geq 3$. Then G does not contain cut-vertices if and only if for any $u, v \in V$ such that $u \neq v$ we have a cycle containing u and v.*

Proof: Let $G = (V, E)$ be an undirected graph that does not contain any cut-edge and isolated vertices. Assume that G does not contain cut-vertices. As a result for any $u, v \in V$ and $w \in V - \{u, v\}$ there always exists a path connecting u and v in $G' = (V', E')$ where $V' = V - \{w\}$ and $E' = E - \{(w, x) | x \in V\}$. Therefore using Lemma 4.6 it follows that there exists a cycle containing u and v in G. Conversely, if for every pair $u, v \in V$ there exists a cycle containing u and v in G then there exists a path connecting u and v in G', where $G' = (V', E')$ with $V' = V - \{w\}$ and $E' = E - \{(w, x) | x \in V\}$ for all $w \in V - \{u, v\}$. As a result it follows from Lemma 4.6 that w is not a cut-vertex. Since this is true for every subset $\{u, v, w\} \in V$ containing 3 distinct vertices it follows that G does not have cut-vertices.

Note 3. *If G is an undirected graph then the subgraph of G formed by vertices that are not cut-vertices is unique.*

We now prove results on simple undirected connected graphs based on the degree of vertices in it.

Lemma 4.8. *Let $G = (V, E)$ be a simple undirected connected graph that does not contain cut-vertices. Let $x_1, x_2, y_1, y_2, u, v \in V$ be such that $d(x_1) \geq 3$, $d(x_2) \geq 3$, $d(y_1) \geq 3$, $d(y_2) \geq 3$ and $d(u) = d(v) = 2$. Also assume that there exists a path P_1 connecting x_1 and y_1 which contains u and which does not contain v. Similarly assume that there exists a path P_2 connecting x_2 and y_2 which contains v and which does not contain u. If apart from x_i and y_i every other vertex in the path P_i is of degree 2, where $1 \leq i \leq 2$, then there exists a cycle in G that contains x_1, u, y_1 and x_2 but which does not contain v.*

(The result is also true when we assume $x_1 = x_2$ and $y_1 = y_2$ and the paths P_1 and P_2 do not have any vertex in common other than x_1 and y_1.)

Proof: We have assumed G to be a simple undirected connected graph and that $d(x_i) \geq 3$ and $d(y_i) \geq 3$ for $1 \leq i \leq 2$. As a result there exists at least 3 paths that connect x_i and y_i in G, where $1 \leq i \leq 2$. Let Q and Q' be two paths connecting x_2 and y_2 such that both these paths do not contain v. Also assume that the paths Q and Q' do not have any vertex in common other than x_2 and y_2.

Let $S \subseteq V$ denote the set of vertices in the path P_2. Then $v \in S$. Consider the subgraph of G formed by the vertices in $(V - S) \cup \{x_2, y_2\}$. Let us denote this subgraph by H. It is easy to note that both the paths Q and Q' exist in H also. Also we can obtain a cycle containing x_2 and y_2 from the paths Q and Q' in H. This shows that H does not contain cut-vertices. Using Corollary 4.7 it is clear that there exists a cycle containing u and x_2 in H. This cycle also contains x_1 and y_1 since these vertices are the end vertices of the path P_1 that contains u. Since this cycle is in H it does not contain v and it is in G also. This completes the proof.

Lemma 4.9. *Let $G = (V, E)$ be a simple undirected connected graph that does not contain cut-vertices such that $|V| \geq 3$. Let D denote the number of vertices in G that have degree at least 3. Then*

1. *$D = 0$ if and only if G is a cycle,*

2. *$D \neq 1$ in G,*

3. *if $D = 2$ and we have $u, v, w \in V$ such that $d(u) \geq 3$, $d(v) \geq 3$ and $d(w) = 2$ then there always exists a cycle in G that contains u and v but which does not contain w,*

4. *if $D \geq 3$ then for any $u, v \in V$ such that $d(u) \geq 3$ and $d(v) \geq 3$ there always exists a cycle in G that contains u but which does not contain v.*

Proof:

1. If for a simple undirected connected graph G that does not contain cut-vertices we have $|V| \geq 3$ and $D = 0$ then every vertex is of degree 2 and so G is obviously a cycle. Conversely, if G is a simple undirected connected graph that is a cycle then it does not have any vertices having degree greater than or equal to 3. This implies $D = 0$.

2. Let G be a simple undirected connected graph that does not contain cut-vertices. Then either G is a cycle in which case we have shown $D = 0$ or the number of vertices having degree greater than or equal to 3 is at least 2. This shows we cannot have $D = 1$.

3. Since $D = 2$ it is clear that G is not a cycle. Also G is simple undirected and connected. Therefore there are at least 3 paths connecting u and v. As a result we get $|V| \geq 4$. Moreover if $x, w \in V - \{u, v\}$ since $D = 2$ it follows that $d(x) = d(w) = 2$. We can also assume without loss of generality that x and w are in distinct paths connecting u and v. Clearly the path connecting u and v that contains x and the path connecting u and v that contains w do not have any vertex in common other than u and v. Now using Lemma 4.8 it follows that there exists a cycle containing u, x and v that does not contain w.

4. Assume that $D \geq 3$. Let $u, v, w \in V$ be such that $d(u) \geq 3$, $d(v) \geq 3$ and $d(w) \geq 3$. Since G is simple undirected connected we have $|V| \geq 4$. Let $x \in V - \{u, v, w\}$. Using Corollary 4.7 we can obtain a cycle C_1 that contains u and w and a cycle C_2 that contains u and x. If C_1 or C_2 do not contain v then the proof is complete.

   Otherwise assume that both C_1 and C_2 contain u and v. Consider the case when there exists at least one vertex in $V - \{u, v\}$ that is in C_1 and in C_2. From amongst these vertices in $V - \{u, v\}$ that are common to C_1 and C_2 we can then choose a vertex y such that using the edges in C_1 and in C_2 that connect u and y we can obtain a cycle that contains u but which does not contain v in G.

   Instead if C_1 and C_2 do not have any vertex in common other than u and v, using Corollary 4.7 it follows that there exists a cycle C_3 in G that contains w and x. As we had observed above we can now construct a cycle in G that contains u and which does not contain v from the paths connecting u and w in C_1, the paths connecting u and x in C_2 and the paths connecting w and x in C_3. This completes the proof.

4.2 Cymatch Permutation

Definition 4.10. *Let $G = (V, E)$ be a simple undirected connected graph that does not contain cut-vertices such that $|V| \geq 3$. We define a Cymatch Permutation of V to be a permutation ψ of V such that for $u, v \in V$ and $u \neq v$ if we have $\psi(u) = v$ then either*

1. *we have $d(u) = d(v) = 2$ and G is a cycle containing vertices u and v, or*

2. *we have $d(u) = d(v) = 2$ and there exists $x, y \in V$ with $d(x) \geq 3$ and $d(y) \geq 3$ such that both u and v are contained in a path P that connects x and y in G. Moreover in this path P apart from x and y every other vertex has degree 2, or*

3. *we have* $d(u) = d(v) = 2$ *and there exists* $x, y \in V$ *with* $d(x) \geq 3$ *and* $d(y) \geq 3$ *such that* $(x, u), (u, y), (y, v), (x, v) \in E$, *or*

4. *we have* $d(u) = d(v) \geq 3$ *and these are the only vertices in* G *that have degree greater than or equal to* 3. *Apart from* u *and* v *every other vertex in* G *has degree* 2. *There are at least* 3 *paths in* G *that connect* u *and* v *and every vertex in* G *that has degree* 2 *is contained in one of these paths connecting* u *and* v.

Otherwise if $G = (V, E)$ *contains at least* 3 *vertices whose degree is greater than or equal to* 3 *then for all* $x \in V$ *such that* $d(x) \geq 3$ *we have* $\psi(x) = x$. *If* $x, y \in V$ *and* $d(x) \neq d(y)$ *then we always have* $\psi(x) \neq y$.

Fact 4.11. *Let* $G = (V, E)$ *be a simple undirected connected graph that does not contain cut-vertices such that* $|V| \geq 3$. *Then the identity permutation on* V *is a cymatch permutation of* V.

Proof: Proof is obvious from Definition 4.10.

Definition 4.12. *Let* $G = (V, E)$ *be a simple undirected connected graph that does not contain cut-vertices such that* $|V| \geq 4$. *We say that a vertex* $v \in V$ *is a* δ*-vertex in* G *if* $d(v) = 2$ *and there exists* $x, u, y \in V$ *with* $d(u) = 2$, $d(x) \geq 3$ *and* $d(y) \geq 3$ *such that* $(x, u), (u, y), (y, v), (x, v) \in E$ *in* G.

Proposition 4.13. *Let* $G = (V, E)$ *be a simple undirected connected graph that does not contain cut-vertices such that there are* 2 *vertices having degree greater than or equal to* 3. *If a set of vertices* $S \subseteq V$ *form a cycle in* G *then the number of* δ*-vertices in* S *is at most* 2. *Also if the number of* δ*-vertices in* S *is* 2 *then* $|S| = 4$.

Proof: Since the number of vertices in G having degree greater than or equal to 3 is 2 it is clear that G is not a cycle. Assume that we have $S \subseteq V$ such that vertices in S form a cycle in G. If S does not contain any δ-vertices then the result immediately follows. Otherwise if we have a δ-vertex $v \in S$ it follows from Definition 4.10 that there exists another δ-vertex $u \in S$ such that $(x, u), (u, y), (y, v), (x, v) \in E$. Since vertices in S form a cycle it follows that the number of δ-vertices in S is at most 2. Moreover when $u, v \in S$ since S is a cycle the edges that form the cycle are once again $(x, u), (u, y), (y, v), (x, v)$ which implies $|S| = 4$.

Remark 2. *In Definition 4.10 we have assumed* $G = (V, E)$ *is a simple undirected connected graph that does contain cut-vertices such that* $|V| \geq 3$. *As a result if* $v \in V$ *then* $d(v) \geq 2$. *Moreover it follows from Corollary 4.7 that given* $u, v \in V$ *there always exists a cycle that contains* u *and* v *in* G.

a. *In defining a cymatch permutation* ψ, *if for* $u, v \in V$ *with* $u \neq v$ *we have* $\psi(u) = v$ *and* $d(u) = d(v) = 2$ *then* G *is a cycle or* u *and* v *are contained in a path that connects* $x, y \in V$ *such that* $d(x) \geq 3$ *and* $d(y) \geq 3$. *Also this path contains only vertices of degree* 2 *apart from* x *and* y *in* G.

b. *Yet another instance when a cymatch permutation* ψ *is such that* $\psi(u) = v$ *where* $u, v \in V$ *with* $u \neq v$ *and* $d(u) = d(v) = 2$ *is when there exists* $x, y \in V$ *with* $d(x) \geq 3$ *and* $d(y) \geq 3$ *such that* $(x, u), (u, y), (y, v), (x, v) \in E$. *In other words we have* $\psi(u) = v$ *when both* u *and* v *are* δ*-vertices (Definition 4.12) connecting* x *and* y *in* G. *The* δ*-vertices case is the only instance where a cymatch permutation maps* $u \in V$ *with* $d(u) = 2$ *to* $v \in V$ *with* $d(v) = 2$ *where* $u \neq v$ *such that there exists* $x, y \in V$ *with* $d(x) \geq 3$ *and* $d(y) \geq 3$ *connecting* u *and* v *in* G. *The above two cases in (a & b) exhaust the possibility of a cymatch permutation to map a vertex of degree* 2 *in* G *to yet another vertex of degree* 2 *in* G.

c. *If for $u, v \in V$ with $u \neq v$ and $d(u) = d(v) \geq 3$ we have a cymatch permutation ψ such that $\psi(u) = v$ then G is such that there does not exist any $w \in V$ with $d(w) \geq 3$ where $w \neq u$ and $w \neq v$. In these instances since we assume G is simple undirected connected it is clear that $|V| \geq 4$ and that there are at least 3 paths connecting u and v. Also every other vertex in G has degree 2 and it lies in one of the paths that connect u and v. Moreover G does not contain cut-vertices. As a result every vertex in G is contained in at least one cycle and every cycle contains both u and v. (Note that our assumption that $d(u) = d(v)$ in Definition 4.10 is redundant. In other words if we assume G is simple undirected connected and that it does not contain cut-vertices such that there are only 2 vertices $u, v \in V$ that have degree greater than or equal to 3 then we can show that $d(u) = d(v)$. A simple way of proving this statement is to observe that if P and P' are two paths that connect u and v in G such that the end vertices of P and P' are also u and v then the vertices that are common to P and P' are u and v only. As a result since we assume $D = 2$ we need to have $d(u) = d(u)$.)*

d. *Let $X = \{x \in V | d(x) \geq 3\}$ and $D = |X|$. In defining a cymatch permutation ψ the only class of graphs where we allow $\psi(x) = y$ for $x, y \in X$ such that $x \neq y$ is when $D = 2$. In any other case if G is such that $D \geq 3$ then we have $\psi(x) = x$ for all $x \in X$. We impose this restriction for $x \in X$ so that we avoid the possibility of a cycle mismatch (Definition 4.1) that can occur due to vertices of degree 3 in G (consequence of Lemma 4.9) when the pair $(G, \psi(G))$ is given as input to* CYCLE MATCHING. *This ensures that the definition of a cymatch permutation in Definition 4.10 is consistent with* CYCLE MATCHING. *Also note that a cymatch permutation ψ of V for a simple undirected connected graph $G = (V, E)$ that does not contain cut-vertices is always such that if $u, v \in V$ with $u \neq v$ and $d(x) \neq d(y)$ then we do not have $\psi(x) = y$.*

Lemma 4.14. *Let $G = (V, E)$ be a simple undirected connected graph that does not contain cut-vertices such that $V = \{1, \ldots, n\}$ and $n = |V| \geq 3$. Let θ be a cymatch permutation of V. Also let $G' = (V', E')$ be a simple undirected connected graph that does not contain cut-vertices such that $V' = V$ and $E' = \{(\theta(u), \theta(v)) | (u, v) \in E\}$, where $u, v \in V$. Then for any $S = \{u_1, \ldots u_l\} \subseteq V$ such that $(u_i, u_j) \in E$ we have $\theta(S) \subseteq V'$ and $(\theta(u_i), \theta(u_j)) \in E'$, where $j = i (mod\ l) + 1$ and $1 \leq i \leq l$. In other words, if a set of vertices in S form a cycle in G then the vertices in $\theta(S)$ form a cycle in G'.*

Let $v \in V$. If $v \in (S \triangle \theta(S))$[3] then v is a δ-vertex.

Also let $X = \{x \in V | d(x) \geq 3\}$ and $D = |X|$. If $D = 2$ then $0 \leq |(S \triangle \theta(S))| \leq 2$.

Proof: (In this proof we do not explicitly mention or refer to $G' = (V', E')$ that is obtained by relabelling vertices of G under the cymatch permutation θ. Instead if $S \subseteq V$ then we always consider vertices in $\theta(S)$ and this suffices to prove our result.)

Let $X = \{x \in V | d(x) \geq 3\}$ and $D = |X|$. We prove the result based on the value of D. When $D = 0$ it follows from Lemma 4.9 that G is a cycle in which case the result is true as mentioned in Remark 2(a). It also follows from Lemma 4.9 that $D \neq 1$.

Let G be such that $D = 2$ and let $X = \{x, y\}$. Since θ is a cymatch permutation of V it follows from Remark 2(d) that either ($\theta(x) = x$ and $\theta(y) = y$) or ($\theta(x) = y$ and $\theta(y) = x$). Moreover we have noted in Remark 2(c) that when $D = 2$ every cycle in G contains x and y. As a result if $S \subseteq V$ forms a cycle then $x, y \in S$ and $x, y \in \theta(S)$.

Since θ is a cymatch permutation of V and $D = 2$, we use Remark 2(c) to recall the minimum requirements about the adjacency of vertices in G. We have $|V| \geq 4$ and for $u, v \in (V - X)$ we

[3]Given two sets A and B we denote the the symmetric difference of A and B by $(A \triangle B)$. That is $(A \triangle B) = (A \cup B) - (A \cap B)$.

always have $d(u) = d(v) = 2$. Also assume that $\theta(u) = v$. Since θ is a cymatch permutation, as mentioned in Remark 2(a & b) it is clear that either we have a path connecting x and y in G that contains u and v or vertices u and v are δ-vertices in G.

For the case when we have a path connecting x and y in G that contains u and v it is clear that when $u, v \in S$ we have $u, v \in (S \cap \theta(S))$. Therefore if there are no δ-vertices in S then $\theta(S) = S$. This implies $|(S \triangle \theta(S))| = 0$ and so the result is true for this case.

Otherwise let us assume that $u \in S$ and that u is the only δ-vertex in S. Since $D = 2$ if $\theta(u) = v$ then from Remark 2(b) we get that v is also a δ-vertex in G that is adjacent to x and y. When $\theta(u) = v$ and $u = v$ we have $\theta(S) = S$ in which case the result is true since vertices in S form a cycle. If $u \neq v$ then $v \notin S$ and so from Remark 2(a) it follows that $\theta(S - \{u\}) = (S - \{u\})$ and these vertices form a path connecting x and y in G. Therefore from these observations we get $\theta(S) = (S - \{u\}) \cup \{v\}$. Since v is a δ-vertex adjacent to x and y, vertices in $(S - \{u\}) \cup \{v\}$ also form a cycle in G. In this case we therefore get $0 \leq |(S \triangle \theta(S))| \leq 1$.

When $D = 2$ and $u, v \in S$ are δ-vertices in G such that $u \neq v$ it follows from Proposition 4.13 that vertices in S form a cycle of length 4. That is $S = \{x, u, y, z\}$ and $(x, u), (u, y), (y, v), (x, v) \in E$. Now using Remark 2(b) it follows that both $\theta(u)$ and $\theta(v)$ are δ-vertices connecting x and y in G. Moreover $\theta\{x, y\} = \{x, y\}$ and so vertices in $\theta(S)$ also form a cycle. As a result we have $0 \leq |\{u, v\} \cap \theta\{u, v\}| \leq 2$ and so we have $0 \leq |(S \triangle \theta(S))| \leq 2$. From Proposition 4.13 we also know that the number of δ-vertices in S is at most 2 and so the result is proved for the case when $D = 2$.

The proof for the case when $D \geq 3$ in G is similar. We need to observe that if $x \in X$ then $\theta(x) = x$. As a result if vertices in $S \subseteq V$ form a cycle in G and if $x \in (S \cap X)$ then $\theta(x) = x$. Rest of the proof for this case when $D \geq 3$ is similar to the case when $D = 2$ and follows from Definition 4.10, Remark 2 and Proposition 4.13. To be more precise, given $S \subseteq V$ we need to consider all $S' \subset S$ such that vertices in S' have degree 2 in G and these vertices form a path connecting vertices x and y where $x, y \in (S \cap X)$. When $S' = \{u\}$ the path so obtained is of length 2 and when $|S'| \geq 2$ the path formed by vertices in S' has length at least 3. When $S' = \{u\}$ and u is a δ-vertex using Remark 2(b) it follows that $\theta(u) = v$ where v is also a δ-vertex connecting x and y. Note that when $S' = \{u\}$ it is not always necessary to have u to be a δ-vertex. However similar to the case when $D = 2$ it is easy to see that $\theta(S') = S'$. When $|S'| \geq 2$ and the length of the path formed by the vertices in S' is at least 3 we have $\theta(S') = S'$ and the proof is similar to the case when $D = 2$. As a result it follows that vertices in $\theta(S')$ once again form a path connecting x and y in G. This shows that vertices in $\theta(S)$ also form a cycle in G.

Let $v \in V$ such that $v \in (S \triangle \theta(S))$. We have already observed that if $S' \subset S$ such that vertices in S' have degree 2 in G and they form a path connecting $x, y \in (S \cap X)$ then $\theta(S') = S'$. Moreover when $S' = \{u\}$ we may have the cymatch permutation ψ such that $\psi(u) = u$ or that $\psi(u) = v$ where $u, v \in V$ and $u \neq v$. Clearly in the latter case $\theta(S') \neq S'$ and we have $u, v \in (S \triangle \theta(S))$. But it follows from Definition 4.10 and Definition 4.12 that both u and v are δ-vertices. This is true for all $S' \subset S \subseteq V$ such that vertices in S form a cycle in G and $|S'| = 1$ which completes the proof.

Definition 4.15. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be simple undirected graphs such that $V_1 = V_2 = \{1, \ldots, n\}$. We define an isomorphism from G_1 to G_2 to be a mapping $\theta : V_1 \to V_2$ such that for $u, v \in V_1$ if $u \neq v$ then $\theta(u) \neq \theta(v)$ and ($(u, v) \in E_1$ if and only if $(\theta(u), \theta(v)) \in E_2$). When such a mapping $\theta : G_1 \to G_2$ exists we say that G_1 and G_2 are isomorphic.*

Definition 4.16. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be simple undirected connected graphs such that G_1 does not contain cut-vertices and G_2 does not contain cut-vertices. Also let $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2| \geq 3$. We define a Cymatch Isomorphism ψ from G_1 to G_2*

to be an isomorphism from G_1 to G_2 given by the mapping $\psi : V_1 \to V_2$ such that ψ is also a cymatch permutation of V_1. When such a mapping $\psi : V_1 \to V_2$ exists we say that G_1 and G_2 are cymatch isomorphic.

We now prove results on cymatch isomorphism when a pair of simple undirected connected graphs that do not contain cut-vertices is given as input.

Proposition 4.17. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be cycles such that $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2| \geq 3$. Then G_1 and G_2 are cymatch isomorphic.*

Proof: Since G_1 is a cycle on n vertices it follows from the definition of a cymatch permutation (Definition 4.10) that any permutation θ of V_1 is a cymatch permutation of V_1. We also know that any two cycles on $n \geq 3$ vertices are isomorphic. As a result if ψ is an isomorphism from G_1 to G_2 then ψ is also a cymatch permutation of V_1. This shows that G_1 and G_2 are cymatch isomorphic.

Theorem 4.18. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be simple undirected connected graphs such that $V_1 = V_2 = \{1, \ldots, n\}$ and $n \geq 3$. Also assume that G_1 does not contain cut-vertices and G_2 does not contain cut-vertices. Then G_1 and G_2 are cymatch isomorphic if and only if there is a cymatch permutation $\psi : V_1 \to V_2$ such that for any $S \subseteq V_1$, if the vertices in S form a cycle in G_1 then vertices in $\psi(S) \subseteq V_2$ form a cycle in G_2.*

Proof: Let ψ be a cymatch isomorphism from G_1 to G_2. It follows from Definition 4.16 that ψ is a cymatch permutation of V_1. Moreover ψ is an isomorphism from G_1 to G_2. Therefore if $u, v \in V_1$ and if $(u, v) \in E_1$ then $(\psi(u), \psi(v)) \in E_2$. As a result if $S \subseteq V_1$ such that vertices in S form a cycle in G_1 then $\psi(S) \subseteq V_2$ and vertices in $\psi(S)$ form a cycle in G_2.

Conversely, let ψ be a cymatch permutation of V_1 such that for any $S \subseteq V_1$ if the vertices in S form a cycle in G_1 then the vertices in $\psi(S) \subseteq V_2$ form a cycle in G_2. It follows from Lemma 4.14 that if $S \subseteq V_1$ and the vertices in S form a cycle in G_1 then the vertices in $\psi(S)$ also form a cycle in G_1.

When G_1 and G_2 are cycles on n vertices the result follows immediately from Proposition 4.17. Otherwise let $X_i = \{x \in V_i | d(x) \geq 3\}$ and $D_i = |X_i|$, for $1 \leq i \leq 2$. Assume that $D_i \geq 2$, for $1 \leq i \leq 2$. Since $D_1 \geq 2$ and ψ is a cymatch permutation of V_1, we recall the minimum requirements about V_1 and adjacency of vertices in V_1 from Remark 2(c & d). Since G_1 and G_2 are not cycles and ψ is a cymatch permutation of V_1 we need to have $X_2 = X_1$ and that $D_2 = D_1$ in G_2. If not, it follows from Lemma 4.9 that we get a cycle mismatch (Definition 4.1) between G_1 and G_2 caused by a vertex in $(X_1 \triangle X_2)$. But this leads to a contradiction for G_1 due to the result proved in Lemma 4.14 since ψ is a cymatch isomorphism from G_1 to G_2.

So assume that $X_1 = X_2$ and $D_1 = D_2 \geq 2$. We now recall the proof of Lemma 4.14. Since ψ is a cymatch permutation of V_1 we have $\psi(X_1) = X_1$. In addition if $D_1 = 2$ then ψ may be a cycle of length 2 on X_1 or an identity mapping on X_1. Otherwise if $D_1 \geq 3$ we have $\psi(x) = x$ for all $x \in X_1$.

Also if we have a set of vertices $S' \subset S \subseteq V_1$ such that every vertex in S' has degree 2 and these vertices form a path of length at least 3 connecting $x, y \in (S \cap X_1)$ in G_1 then $\psi(S') = S'$. Even though we assume $\psi(S')$ forms a path in G_2 note that if $u, v \in S'$ and if $(u, v) \in E_1$ then we need not have $(\psi(u), \psi(v)) \in E_2$. In fact we may not even have $(\psi(u), \psi(v)) \in E_1$. That is, if $u, v \in S'$ and $(u, v) \in E_1$ then any of the following possibilities can occur.

- $(\psi(u), \psi(v)) \in E_1$ and $(\psi(u), \psi(v)) \in E_2$, or
- $(\psi(u), \psi(v)) \in E_1$ and $(\psi(u), \psi(v)) \notin E_2$, or
- $(\psi(u), \psi(v)) \notin E_1$ and $(\psi(u), \psi(v)) \in E_2$, or

- $(\psi(u), \psi(v)) \notin E_1$ and $(\psi(u), \psi(v)) \notin E_2$.

However since ψ is a cymatch permutation, as in the proof of Lemma 4.14, we always have $\psi(S') = S'$ and S' forms a path that connects $x, y \in X_1$.

When $S' = \{u\}$ such that $d(u) = 2$ and u is not a δ-vertex in G_1 it follows from the definition of a cymatch permutation (Definition 4.10) that $\psi(S') = S'$. For the case when $S' = \{u\}$ and u is a δ-vertex that connects vertices $x, y \in X_1$ we have $\psi(u) = v$ where v is also a δ-vertex in G_1 that connects the same pair of vertices $x, y \in X_1$. From our assumption on ψ it is clear that these observations on the adjacency of vertices in $\psi(S')$ is true in G_2 also.

We now define $\theta : V_1 \to V_2$ based on ψ and show that θ is a cymatch permutation of V_1 that is also an isomorphism from G_1 to G_2. If $v \in V_1$ such that $v \in X_1$ (recall that $X_1 = X_2$) or v is such that $d(v) = 2$ such that v is adjacent to vertices $x, y \in X_1$ then we define $\theta(v) = \psi(v)$ (note when we have $d(v) = 2$ we also take care of the case of v being a δ-vertex in G_1).

Otherwise let $v \in S'$ where $S' \subseteq V_1$ and $|S'| \geq 2$ as described above. We recall that $d(v) = 2$ and there exists $x, y \in X_1$ such that v is contained in a path formed by vertices in S' of length at least 3 connecting x and y in G_1. Moreover $|S'| \geq 2$ and $\psi(S') = S'$. Assume that the length of the path connecting x and v in G_1 formed by vertices in $(\{x\} \cup S')$ is l. Then $1 \leq l \leq |S'|$. We then define $\theta(v) = u$ if the length of the path connecting $\psi(x)$ and u in G_2 formed by the vertices in $(\psi(x) \cup S')$ is l.

We have already shown that $X_1 = X_2$ and it follows from the definition of θ that $\theta(X_1) = X_2$. Also note that the only instance when we may have θ to be a non-identity mapping on X_1 is when $D_1 = 2$. Here we may have θ to be a cycle of length 2 on X_1 if ψ is a cycle of length 2 on X_1. If $u \in V_1$ such that $d(u) = 2$ and u is adjacent to vertices $x, y \in X_1$ then $\theta(u)$ is also a vertex having degree 2 in G_2 that is once again adjacent to $x, y \in X_2$ (note that this includes δ-vertices in G_1). Also for any $S' \subseteq V_1$ that contains only vertices of degree 2 such that $|S'| \geq 2$ as described above, we have $\theta(S') = S' \subseteq V_2$. In addition, as stated in Lemma 4.14, θ also ensures that if $u, v \in S'$ and $(x, u) \in E_1$ or $(u, v) \in E_1$ or $(v, y) \in E_1$ then $(\theta(x), \theta(u)) \in E_2$ or $(\theta(u), \theta(v)) \in E_2$ or $(\theta(v), \theta(y)) \in E_2$ respectively in G_2 also. This shows that if a set of vertices $S \subseteq V_1$ forms a cycle in G_1 then $\theta(S)$ also forms a cycle in V_1. We therefore get θ to be a cymatch permutation of V_1. The above stated properties of θ also show that θ is an isomorphism from G_1 to G_2. This shows that G_1 and G_2 are cymatch isomorphic and hence the proof is complete.

Theorem 4.19. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be a pair of simple undirected connected graphs such that $V_1 = V_2 = \{1, \ldots n\}$ and $n = |V_1| = |V_2| \geq 3$. Also assume that G_1 does not contain cut-vertices and G_2 does not contain cut-vertices. Then (G_1, G_2) is a "yes" instance of* CYCLE MATCHING *if and only if there exists a cymatch permutation ψ of V_1 that is also a cymatch isomorphism from G_1 to G_2.*

In other words, (G_1, G_2) is a "yes" instance of CYCLE MATCHING *if and only if G_1 and G_2 are cymatch isomorphic.*

Proof: Let (G_1, G_2) be a "yes" instance of CYCLE MATCHING. We know that the identity permutation of V_1 is itself a cymatch permutation of V_1 (Fact 4.11). Let us denote the identity permutation of V_1 by e. Since (G_1, G_2) is a "yes" instance of CYCLE MATCHING if a set of vertices $S \subseteq V_1$ form a cycle in G_1 then vertices in S form a cycle in G_2 also. Also $e(S) = S$ for all $S \subseteq V_1$. Therefore using Theorem 4.18 it follows that G_1 and G_2 are cymatch isomorphic.

Conversely assume that G_1 and G_2 are cymatch isomorphic and let θ be a cymatch isomorphism from G_1 to G_2. Then it follows from the definition of a cymatch isomorphism (Definition 4.16) that θ is a cymatch permutation of V_1 that is also an isomorphism from G_1 to G_2. Also using Theorem 4.18 it follows that if vertices in $S \subseteq V_1$ form a cycle in G_1 then $\theta(S) \subseteq V_2$ and $\theta(S)$

forms a cycle in G_2. If for all subsets of vertices $S \subseteq V_1$ that form a cycle in G_1 we have $\theta(S) = S$ then it is immediate that (G_1, G_2) is a "yes" instance of CYCLE MATCHING.

Otherwise it follows from Lemma 4.14 that if $v \in (S \triangle \theta(S))$ then v is a δ-vertex. We now define another permutation ψ on V_1 as follows. If $v \in V_1$ and v is a δ-vertex in G_1 then $\psi(v) = v$ and for all other vertices $u \in V_1$ we let $\psi(u) = \theta(u)$. Since ψ maps a δ-vertex to itself and is defined to be θ on every other vertex in V_1 it is easy to observe that ψ is also a cymatch permutation of V_1. Also ψ is an isomorphism from G_1 to G_2. More importantly note that if vertices in $S \subseteq V_1$ form a cycle in G_1 then $\psi(S) = S \subseteq V_2$ and vertices in S forms a cycle in G_2. This shows that (G_1, G_2) is a "yes" instance of CYCLE MATCHING.

4.3 Subroutines computable in L

We now give some subroutines that are computable in L.

Lemma 4.20. *Given an undirected graph G and an edge e in G, there is a logspace algorithm to test if e is a cut-edge in G.*

Proof: Let $e = (u, v) \in E(G)$. It follows from Definition 4.2, that e is a cut-edge if and only if there is does not exist any path connecting u and v in the graph $G - \{e\}$. It is easy to obtain $G - \{e\}$ from G in logspace. We then use Reingold's undirected *st*-connectivity algorithm [16] to test if there is path connecting u and v in $G - \{e\}$. If it exists then we output e is not a cut-edge. Otherwise we output e is a cut-edge.

Since Reingold's undirected *st*-connectivity algorithm [16] uses at most $O(\log(|V| + |E|))$ space and we need at most $O(\log(|V| + |E|))$ space to keep track of the edge e we can determine if e is a cut-edge in G in L.

Using Lemma 4.20 it is clear that we can obtain the subgraph H of G that does not contain any cut-edge in L. Moreover we also know that this subgraph H is unique (Note 2). We outline the set of instructions as a subroutine.

CUT-EDGE FREE SUBGRAPH(G)
Input: An undirected graph G.
Output: The subgraph H of G such that H does not contain cut-edges that are in G.
Complexity upper bound: L using Lemma 4.20.
for (each $e \in E(G)$)**do**
  if (e is not a cut-edge) **then**
    Output e.
  endif
endfor

Lemma 4.21. *Given an undirected graph G and a vertex v in G, there is a logspace algorithm to test if v is a cut-vertex in G.*

Proof: Let $v \in V(G)$. It follows from Definition 4.3 that v is a cut-vertex if and only if there exists vertices u and w in the graph G such that there exists a path connecting u and w in G and that there does not exist any path connecting u and w in $G - \{v\}$. Given vertices u and w in G, using Reingold's undirected *st*-connectivity algorithm [16] we test if there exists a path connecting u and w in G. Also once again using Reingold's undirected *st*-connectivity algorithm [16] we test if there exists a path connecting u and w in $G - \{v\}$. If for some pair of vertices $u, w \in V(G)$ such a path exists in G but it does not exist in $G - \{v\}$ then we output v is a cut-vertex in G. Otherwise we output v is not a cut-vertex in G.

Since Reingold's undirected *st*-connectivity algorithm [16] uses at most $O(\log(|V| + |E|))$ space and we need at most $O(\log(|V| + |E|))$ space to keep track of vertices u, v, w and obtain

the subgraph formed by vertices $G - \{v\}$ in every stage, we can determine if v is a cut-vertex in L.

Let G be an undirected graph and assume that we obtain the subgraph H as the output of CUT-EDGE FREE SUBGRAPH(G). Then using Corollary 4.7 and Lemma 4.21 we can determine if a given pair of vertices $u, v \in V(H)$ are contained in a cycle in H (and therefore in G also) in L. We outline the set of instructions in the following subroutine.

CUT-VERTEX FREE PATHS(H,u,v)
Input: An undirected graph H and vertices $u, v \in H$ such that H does not contain cut-edges.
Output: "yes" if there exists a cycle containing u and v in H. Output "no" otherwise.
Complexity upper bound: L using Lemma 4.21.
if (there exists a path connecting u and v in H) **then**
  for (each $w \in V(H)$) **do**
    if (($w \neq u$) and ($w \neq v$) and (w is a cut-vertex)) **then**
      if (there exists a path connecting u and v in $H - \{w\}$) **then**
        Output "yes" and stop.
      endif
    endif
  endfor
  Output "yes" and stop.
endif
Output "no" and stop.

Let H be an undirected graph such that H does not contain cut-edges. Also let $v \in V(H)$ be a vertex that is not a cut-vertex. Then the following subroutine outputs a subset $S \subseteq V(H)$ such that $v \in S$ and there always exists a cycle contaning any two vertices from S in H. Using Reingold's undirected *st*-connectivity algorithm and the L upper bound obtained for CUT-VERTEX FREE PATHS(H,u,v) we are able to show that we can output S in L.

SPLIT CUT-VERTEX GRAPHS(H,v)
Input: An undirected graph H such that H does not contain cut-edges and a vertex $v \in V(H)$ such that v is not a cut-vertex.
Output: A set of vertices $S \subseteq V(H)$ that form a subgraph H_v of H such that $v \in S$ and there always exists a cycle containing any two vertices from S in H.
Complexity upper bound: L using the upper bound for CUT-VERTEX FREE PATHS(H,u,v).
for (each $u \in V(H)$) **do**
  if ($u \neq v$) **then**
    if (CUT-VERTEX FREE PATHS(H,u,v)="yes") **then**
      Output u.
    endif
  endif
endfor
Output v and stop.

Remark 3. *It follows from our assumption on H and using Corollary 4.7 that given two vertices $u, w \in S$ there always exists a cycle containing u and w in H_v and therefore in H also.*

4.4 CYCLE MATCHING is L-complete

We now prove one of our main results that CYCLE MATCHING is in L.

Theorem 4.22. CYCLE MATCHING *is in* L.

Proof: Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the pair of undirected graphs with $V_1 = V_2 = \{1, \ldots, n\}$ and $n = |V_1| = |V_2|$ given as input to the CYCLE MATCHING problem. Let $N = (|(V_1 \cup V_2)| + |(E_1 \cup E_2)|)$ be the size of the input graph pairs (G_1, G_2). We now describe our logspace algorithm for deciding if (G_1, G_2) is a "yes" instance of CYCLE MATCHING.

We obtain subgraphs of G_1 and G_2 using the CUT-EDGE FREE SUBGRAPH(G_1) and CUT-EDGE FREE SUBGRAPH(G_2) subroutines in Section 4.3. Let the subgraphs obtained from CUT-EDGE FREE SUBGRAPH(G_1) and CUT-EDGE FREE SUBGRAPH(G_2) be G'_1 and G'_2 respectively. We then use Reingold's undirected *st*-connectivity algorithm to obtain connected components of G'_1 and G'_2. Assume that $H_1 = (V_1^H, E_1^H)$ is a connected component of G'_1 and $H_2 = (V_2^H, E_2^H)$ is a connected component of G'_2. Note that whenever $(V_1^H \cap V_2^H) \neq \emptyset$ then it is essential to have $V_1^H = V_2^H$. Otherwise since H_1 and H_2 are subgraphs of G'_1 and G'_2, it follows from Corollary 4.5 and Note 2 that if a vertex v is in V_1^H but however v is not in V_2^H then we get a cycle mismatch (Definition 4.1) between H_1 and H_2 that is caused by v. This shows that (H_1, H_2) is a "no" instance of CYCLE MATCHING and since H_i is a subgraph of G_i, for $1 \leq i \leq 2$, we get (G_1, G_2) is also a "no" instance of CYCLE MATCHING. In this case we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. Since CUT-EDGE FREE SUBGRAPH(G_i) outputs the edges of the subgraphs G'_1 and G'_2 using space at most $O(\log N)$ and Reingold's undirected *st*-connectivity algorithm also uses at most $O(\log N)$ space we can output all the connected components of G'_i, for $1 \leq i \leq 2$, in L.

Assume that we have obtained undirected connected graphs $H_i = (V_i^H, E_i^H)$ as output from the CUT-EDGE FREE SUBGRAPH(G_i) subroutine, where $V_1^H = V_2^H$ and $1 \leq i \leq 2$. We first check if there exists a vertex that has a loop in H_1 if and only if the vertex has a loop in H_2 also. Similarly we also check whether a pair of vertices in $u, v \in \{1, \ldots, n\}$ form a cycle of length 2 in H_1 if and only if the pair u, v also form a cycle of length 2 in H_2. If one of these conditions is not true then we get a cycle mismatch caused by vertices having loops or cycles of length 2 between H_1 and H_2 and so (H_1, H_2) is a "no" instance of CYCLE MATCHING. Therefore in this case we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. It is easy to note that checking for the existence of loops and parallel edges in H_1 and H_2 between vertices requires constant space and so the algorithm requires at most $O(\log N)$ space.

From now onwards we assume that the input graphs $H_1 = (V_1^H, E_1^H)$ and $H_2 = (V_2^H, E_2^H)$ are simple undirected connected graphs and that both these graphs do not contain cut-edges. We also assume that $V_1^H = V_2^H$. Since H_1 and H_2 are subgraphs of G_1 and G_2, using Lemma 4.21 we can identify cut-vertices in H_1 and H_2 using space at most $O(\log N)$. Following this we also split H_i into subgraphs using SPLIT CUT-VERTEX GRAPHS(H_i,u) subroutine where $u \in V_i^H$, where u is not a cut-vertex in H_i, for $1 \leq i \leq 2$. It is then clear that every subgraph of H_i that is output is simple undirected connected and neither contains cut-edges nor cut-vertices. Similar to the pair (H_1, H_2) it is also clear (as mentioned in Note 3) that if we obtain two connected subgraphs H'_1 and H'_2 as the output of SPLIT CUT-VERTEX GRAPHS(H_1,u) and SPLIT CUT-VERTEX GRAPHS(H_2,u) respectively, and if the vertex sets of H'_1 and H'_2 have a non-empty intersection then then the vertex set of H'_1 and H'_2 is the same. Otherwise we have a vertex that causes cycle mismatch between H'_1 and H'_2 and since H'_1 and H'_2 are subgraphs of G_1 and G_2 respectively, we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. Once again given the pair (H_1, H_2) as input it follows from Lemma 4.21 and the complexity upper bound for SPLIT CUT-VERTEX GRAPHS(H_i,u) that we can decide if u is not a cut-vertex in H_i and also split H_i graphs into subgraphs H'_i, for $1 \leq i \leq 2$, using space at most $O(\log N)$.

Now assume that the vertex sets of H'_1 and H'_2 is the same. In order to check if (H'_1, H'_2) is a "yes" instance of CYCLE MATCHING we use Theorem 4.19 to verify if there exists a cymatch permutation of $V_1^{H'}$ that is also an isomorphism of G_1 and G_2. The proof of

Theorem 4.19 depends on Theorem 4.18. More precisely if we assume the existence of a cymatch permutation θ of $V_1^{H'}$ which when viewed as a mapping between $V_1^{H'}$ and $V_2^{H'}$ also preserves cycles between H_1' and H_2' we are able to construct a cymatch isomorphism between H_1' and H_2' and hence conclude (H_1', H_2') is a "yes" instance of CYCLE MATCHING. A closer examination of the proofs of Theorem 4.19 and Theorem 4.18 shows that if (H_1', H_2') is a "yes" instance of CYCLE MATCHING then we are able to construct a cymatch isomorphism between H_1' and H_2' even with the identity mapping on $V_1^{H'}$ as a cymatch permutation of $V_1^{H'}$.

As a result using ideas from Theorem 4.18 and Theorem 4.19 in order to verify if (H_1', H_2') is a "yes" instance of CYCLE MATCHING it suffices to do the following.

- if both H_1' and H_2' are cycles then output "yes" to the pair (H_1', H_2').
- otherwise let $X_i = \{x \in V_i^{H'} | d(x) \geq 3\}$ and $D_i = |X_i|$, for $1 \leq i \leq 2$. If $X_1 \neq X_2$ then we get a cycle mismatch between H_1' and H_2' and so (H_1', H_2') is a "no" instance of CYCLE MATCHING. Since H_1' and H_2' are subgraphs of G_1 and G_2 respectively, we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop.
- assume that $X_1 = X_2$ and hence $D_1 = D_2 \geq 2$. We then consider the subgraph of H_1' obtained by deleting all the vertices of X_1 from H_1'. Let us denoted this subgraph by H_{11}'. Similarly we consider the subgraph of H_2' obtained by deleting all vertices in X_2 from H_2'. Let us denote this subgraph by H_{22}'. It is easy to note that any vertex in H_{11}' and H_{22}' has degree $0, 1$ or 2. In other words we only have isolated vertices and paths in H_{11}' and H_{22}'.
- for every vertex v, we check if v is an isolated vertex in H_{11}' if and only if v is an isolated vertex in H_{22}' also. In addition we also check if v is adjacent to the same pair of vertices in X_1 and in X_2. If one of these two conditions is not true then (H_1', H_2') is a "no" instance of CYCLE MATCHING. Since H_1' and H_2' are subgraphs of G_1 and G_2 respectively we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. This step takes care of δ-vertices that exist in H_1' and H_2'. Note that this step also takes care of the case when we obtain v to be an isolated vertex in H_{ii}' since $(x, v), (v, y) \in E_i$ but v is not a δ-vertex in H_i', where $d(x) \geq 3, d(y) \geq 3$ and $d(v) = 2$ for $1 \leq i \leq 2$.
- otherwise we consider paths formed by vertices in H_{11}' and H_{22}'. If a set of vertices $S \subseteq V_1^{H'}$ forms a path in H_{11}' then vertices in $S \subseteq V_2^{H'}$ should also form a path in H_{22}'. Also we check if the end vertices of the path formed by vertices in S are adjacent to the same pair of vertices in X_1 and in X_2. If one of these two conditions is not true then (H_1', H_2') is a "no" instance of CYCLE MATCHING. Since H_1' and H_2' are subgraphs of G_1 and G_2 respectively we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. This step takes care of the case when we have a path of length at least 3 connecting vertices in X_1 and in X_2 in H_1' and H_2' respectively.

We have assumed H_1' and H_2' to be simple undirected connected graphs and that both these graphs neither contain cut-edges nor cut-vertices. The proof of correctness of the claim that steps described above correctly decide if (H_1', H_2') is a "yes" instance of CYCLE MATCHING follows from Fact 4.11 and the proof of Theorem 4.19.

We repeat the steps described above for every pair of simple undirected connected graphs (H_1', H_2') that do not contain cut-vertices obtained from SPLIT CUT-VERTEX GRAPHS(H_i,u), for $i = 1, 2$ respectively, such that both of H_1' and H_2' have the same vertex set. If there exists some pair (H_1', H_2') which is a "no" instance of CYCLE MATCHING then we output (G_1, G_2) is a "no" instance of CYCLE MATCHING and stop. Otherwise if for all pairs of subgraphs

(H'_1, H'_2) we get a "yes" output then (H_1, H_2) is a "yes" instance. Once again we need to repeat this step for every pair (H_1, H_2) that is output by the CUT-EDGE FREE SUBGRAPH(G_i) subroutine for $1 \leq i \leq 2$ respectively, and check if for all pairs (H_1, H_2) we get a "yes" instance. If so we output (G_1, G_2) is a "yes" instance of CYCLE MATCHING and stop. Otherwise if for some pair (H_1, H_2) we get a "no" output then we output (G_1, G_2) is a "no" instance of the CYCLE MATCHING problem and stop.

Using results from [5] it follows that given a simple undirected graph G on n vertices we can check if G is a cycle using at most $O(\log N)$ space. If we are given (H'_1, H'_2) as input then obtaining the pair of subgraphs (H'_{11}, H'_{22}) as described above involves deleting vertices of degree greater than or equal to 3. This is clearly computable in $O(\log N)$. Also checking for isolated vertices in H'_{ii} and their neighbours in H'_i, for $1 \leq i \leq 2$, is also computable in $O(\log N)$. Similarly we can also check for subsets of vertices that form paths in H'_{11} and H'_{22} and the neighbours of the end vertices of these paths in H'_1 and H'_2 respectively using at most $O(\log N)$ space.

We have also showed an upper bound of $O(\log N)$ in every stage to output subgraph pairs (H'_1, H'_2) from (H_1, H_2). Also note that if we have an edge $(u, v) \in E_i$ then it occurs in exactly one subgraph of H_i that is output by the SPLIT CUT-VERTEX GRAPHS(H_i,u) subroutine, for $1 \leq i \leq 2$. As a result it is easy to note that given (H_1, H_2) the number of subgraph pairs (H'_1, H'_2) that we output is also upper bounded by a polynomial in the size of (G_1, G_2). We can keep track of these subgraph pairs that is output using at most $O(\log N)$ space. Moreover number of subgraph pairs (H_1, H_2) of (G_1, G_2) is upper bounded by the number of components in (G_1, G_2). As a result we need at most $O(\log N)$ space to keep track of these graph pairs. Using these observations it follows that we can decide if (G_1, G_2) is a "yes" instance of CYCLE MATCHING using space at most $O(\log N)$. This shows CYCLE MATCHING $\in$ L.

Corollary 4.23. *Let (G_1, G_2) be a pair of simple undirected connected graphs such that G_1 and G_2 do not contain cut-vertices. Then we can decide if G_1 and G_2 are cymatch isomorphic is in* L. *Also if G_1 and G_2 are cymatch isomorphic then we can also construct a cymatch isomorphism from G_1 to G_2 in* FL.

Theorem 4.24. CYCLE MATCHING *is hard for* L

Proof: Using Reingold's undirected *st*-connectivity algorithm [16], given an undirected graph G and vertices u, v in G we can decide whether there exists a path connecting u and v in G in L. As a result it follows that the *st*-connectivity problem for undirected graphs is complete for L. Now given an undirected graph $G = (V, E)$ and vertices $s, t \in V$, we output $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_1 = V_2 = V$, $E_1 = E$ and $E_2 = E \cup \{(s, t)\}$. It is clear that if there does not exist any path connecting s and t in G, then the vertices corresponding to any $S \subseteq V$ form a cycle in G_1 if and only if the vertices in S form a cycle in G_2. Thus the pair (G_1, G_2) is an "yes" instance of the CYCLE MATCHING problem. Otherwise if there is a path connecting s and t in G then there exists at least one set $S \subseteq V$ such that $s, t \in S$ and the vertices in S form a cycle in G_2 but vertices in S do not form a cycle in G_1. In this case the pair (G_1, G_2) is a "no" instance of the CYCLE MATCHING problem. Given the input G we can construct G_1 and G_2 easily (even in L-uniform AC^0). The fact that L is closed under complement then completes the proof.

4.5 A hard counting problem from CYCLE MATCHING

Definition 4.25. *We say that a function $f : \{0,1\}^* \to \mathbb{Z}^+$ is in* #P *if there exists a* NP *machine M such that $f(x)$ is the number of accepting computation paths of M on input x.*

Definition 4.26. *We say that a function $f : \{0,1\}^* \to \mathbb{Z}$ is polynomial time many-one reducible to $g : \{0,1\}^* \to \mathbb{Z}$, if for any input $x \in \{0,1\}^*$ we can compute $f(x)$ when we are given $g(x)$ as input in time polynomial in $|x|$.*

In this section we show that given a pair of input graphs (G_1, G_2) with the same number of vertices, the problem of counting the number of $X \subseteq \{1, \ldots, n\}$ such that vertices corresponding to X form a cycle in one of the input graphs but not the other is #P-complete. We then conclude the results of this paper with interesting observations on counting witnesses for "no" instances of CYCLE MATCHING.

It is well known that counting the number of cycles in an undirected graph G is complete for #P under polynomial time many-one reductions (we give a proof of this result in Theorem 4.27 in the Appendix). We now return to the counting problem defined with respect to CYCLE MATCHING. Let G be the input graph on n vertices. Consider the graph G' formed by a path on n vertices. Clearly G' does not contain any cycle. Therefore the number of cycles in G is equal to the number of subsets of vertices that form a cycle in G, but these subsets do not form a cycle in G'. This reduction is polynomial time computable and this shows that the problem of counting the number of subsets of $\{1, \ldots, n\}$ such that vertices in one of the given pair of input graphs form a cycle but these subsets of vertices do not form a cycle in the other input graph is #P-hard under polynomial time many-one reductions. Since this problem is trivially in #P we get #P-completeness under polynomial time many-one reductions.

Remark 4. *Assume that (G_1, G_2) is a no instance of the* CYCLE MATCHING *problem. Let $\mathcal{F}_1$ denote the collection of subsets of $\{1, \ldots, n\}$ such that if $S \in \mathcal{F}_1$ then the vertices in S form a cycle in G_i but the vertices in S do not form a cycle in G_j for $1 \leq i, j \leq 2$ and $i \neq j$. We have shown that computing the size of $\mathcal{F}_1$ is* #P*-complete.*

We have also shown that CYCLE MATCHING *is in* L *in Theorem 4.22. We recall the proof of Theorem 4.22 and the algorithm described in it. Given a pair of input graphs (G_1, G_2) the algorithm checks for any mismatch due to loops and due to cycles of length 2 between G_1 and G_2. If G_1 and G_2 agree on the above two conditions then it obtains at most polynomially many subgraphs H'_{11} and H'_{22} of G_1 and G_2 respectively which are then used to decide if (G_1, G_2) is a "no" instance of* CYCLE MATCHING*. When (G_1, G_2) is a "no" instance of* CYCLE MATCHING *we may get a witness for this "no" instance from every such subgraph pair (H'_{11}, H'_{22}) that is output. It is easy to note that a witness for the "no" instance is an isolated vertex or a path formed by vertices in H'_{11} that is not in H'_{22}. Let $\mathcal{F}_2$ denote the collection of all subsets of $\{1, \ldots, n\}$ such that any if $S \subseteq \{1, \ldots, n\}$ and $S \in \mathcal{F}_2$ then vertices in S satisfy one of the conditions we have mentioned above (as in Theorem 4.22) and hence witnesses that (G_1, G_2) is a "no" instance of* CYCLE MATCHING*. Clearly we can compute the size of $\mathcal{F}_2$ in* FL*.*

These two observations we have arrived at seem intriguing and surprising since in spite of the vast difference in the computational complexity of computing the sizes of $\mathcal{F}_1$ and $\mathcal{F}_2$, any element in $\mathcal{F}_1$ or any element in $\mathcal{F}_2$ independently witnesses the fact that (G_1, G_2) is a "no" instance of CYCLE MATCHING*.*

Acknowledgements

I derived the motivation to obtain the results shown in this manuscript in the Microsoft Research Theory Day Workshop held in Bangalore during December 2007. I thank Ravindran Kannan for organising this workshop and enabling me to participate in it. I also thank Ravindran Kannan for the sustained encouragement and support he has given me since then.

I thank Samir Datta and Srikanth Srinivasan for useful discussions. I also thank the anonymous third referee of the STACS 2009 conference for correcting Remark 1 that the number of equivalence classes under the equivalence relation $\sim$ shown in Theorem 3.3 is more than one even for linear representations over a field $\mathbb{F}$ of characteristic 2 representing the same matroid M if $\mathbb{F}$ is not $\mathbb{Z}_2$.

I am grateful to Jaikumar Radhakrishnan for hosting my visit to the School of Technology and Computer Science, TIFR during December 2009. I am also grateful to Jaikumar Radhakrishnan for useful discussions during this visit and in particular for providing counter examples and explaining that the algorithm I have described for CYCLE MATCHING in earlier versions[4] is incorrect.

References

[1] Eric Allender and Mitsunori Ogihara. Relationships among PL, #L and the Determinant. *RAIRO-Theoretical Informatics and Applications*, 30:1-21, 1996.

[2] Càrme Alvarez and Raymond Greenlaw. A compendium of problems complete for symmetric logspace. *Computational Complexity*, 9:123-145, 2000.

[3] Mark Braverman, Raghav Kulkarni, and Sambuddha Roy. Space efficient counting in graphs on surfaces. *Computational Complexity*, 18(4):601-649, 2009.

[4] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and Importance of Logspace-MOD-Classes. *Mathematical Systems Theory*, 25(3):223-237, 1992.

[5] Stephen Cook and Pierre McKenzie. Problems complete for Deterministic Logarithmic Space. *Journal of Algorithms*, 8:385-394, 1987.

[6] J. Edmonds. Minimum partition of a matroid into independent subsets. *J. Res. National Bureau of Standards*, 69B:67-72, 1965.

[7] J. Edmonds. Matroids and the Greedy Algorithm. *Mathematical Programming*, 1:127-136, 1971.

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

[9] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for Graph Isomorphism. *Journal of Computer and System Sciences*, 66(3):549-566, 2003.

[10] Birgit Jenner, Klaus-Jörn Lange, and Pierre McKenzie. *Tree Isomorphism and Some Other Complete Problems for Deterministic Logspace.* publication #1059, DIRO, Université de Montreál, mars, 1997.

[11] Johannes Köbler. On Graph Isomorphism for Restricted Graph Classes. *In Logical Approaches to Computational Barriers, Proceedings of the Second Conference on Computability in Europe (CiE 2006)*, LNCS 3988, pages 241-256, 2006.

[12] Steve Lindell. A Logspace Algorithm for Tree Canonization (Extended Abstract). *Proceedings of the* 24^{th} *Annual ACM Symposium on Theory of Computing (STOC '92)*, pages 400-404, 1992.

[4] Revision 1 of ECCC Report No.09(2009)

[13] James Oxley. *Matroid Theory.* Oxford University Press, 2006.

[14] Christos Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

[15] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall of India Private Limited, 2001.

[16] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):1-24, 2008.

[17] Leslie Valiant. Completeness for Parity Problems. *Proceedings of the* 11^{th} *International Computing and Combinatorics Conference (COCOON '05)*, pages 1-8, 2005.

[18] T.C. Vijayaraghavan. *Classifying certain Algebraic Problems using Logspace Counting Classes.* Ph.D Thesis, The Institute of Mathematical Sciences, Homi Bhabha National Institute, India, December 2008.

[19] Douglas West. *Introduction to Graph Theory*, Second edition. Prentice-Hall of India Private Limited, 2003.

Appendix

We show that the problem of counting the number of cycles in an undirected graph G is complete for #P under polynomial time many-one reductions. We show that the problem of counting the number of Hamilton cycles in an undirected graph polynomial time many-one reduces to our problem. As it is standard in reductions between graph problems [14, Chapters 9 and 18][17] we also employ a gadget construction in our proof.

Theorem 4.27. *Let $G = (V, E)$ be an input undirected graph such that $n = |V|$. Then counting the number of cycles in G is complete for* #P *under polynomial time many-one reductions.*

Proof: Let $G = (V, E)$ be an input undirected graph and let n denote the size of the graph G. We know that counting the number of Hamilton cycles in G is complete for #P under polynomial time many-one reductions [14, Theorem 18.2]. We show that the above problem is polynomial time many-one reducible to the problem of counting the number of cycles in G.

Given a graph $G = (V, E)$ let l denote the length of a cycle in G. Then we know that $3 \leq l \leq n$. Also if N denotes the number of cycles in G then $N = \Sigma_{i=3}^{n} m_i$, where m_i denotes the number of cycles of length i in G. It is easy to observe that $N < 2^n$ and therefore it follows that $m_i < 2^n$ for all $3 \leq i \leq n$.

We now replace every edge in G by a collection of n^4 paths of length 2 each to obtain another graph $G' = (V', E')$. It follows from the definition of G' that every cycle in G' has even length. Also if we have a set of edges in G that form a cycle C_l of length l such that these edges are incident to a set of vertices $S \subseteq V$ in G, then $l = |S|$ and we obtain l^{n^4} cycles of length $2l$ each in G' from C_l under this reduction. We also have cycles of length 4 in G'. However the number of cycles of length 4 is bounded by a polynomial in $n = |G|$.

Elaborating on this observation if we have m_l cycles of length l in G and we have m_{l+1} cycles of length $(l + 1)$ in G, where $3 \leq l \leq (n - 1)$, then we have $(l^{n^4} m_l)$ cycles of length $2l$ in G' and $((l + 1)^{n^4} m_{l+1})$ cycles of length $2(l + 1)$ in G'.

Claim 4.28. *If $3 \leq l \leq (n - 1)$ then $(l + 1)^{n^4} > 2^n \Sigma_{i=3}^{l} (i^{n^4})$.*

Proof of Claim 4.28. We use induction on l to prove the claim. When $l = 3$ it is easy to see that $(4^{n^4}) > (2^n(3^{n^4}))$ if $n \geq 3$. Now assume the result to be true for some $3 \leq l \leq (n-1)$ and consider $((l+1)^{n^4})$. Using induction on l it follows that $l^{n^4} > 2^n \Sigma_{i=3}^{(l-1)}(i^{n^4})$. It is also clear that $(l+1)^{n^4} > 2(l^{n^4})$ when $3 \leq l \leq (n-1)$ and so the claim is true.

It is clear that if N' denotes the number of cycles in G' then $N' = \#(C_4) + \Sigma_{i=3}^{n}(i^{n^4} m'_i)$, where $\#(C_4)$ denotes the number of cycles of length 4 in G' and m'_i denotes the number of cycles of length $2i$ in G'. We know that $\#(C_4)$ is a polynomial in $n = |G|$ and $m'_i = m_i < 2^n$. Given G it is also easy to compute $\#(C_4)$. It is easy to note that we can upper bound $\Sigma_{i=3}^{j}(i^{n^4} m'_i)$ for each $3 \leq j \leq n$ by $2^{p(n)}$, where $p(n)$ is a polynomial in n. As a result using Claim 4.28 it follows that the binary representation of N' is then a concatenation of the binary representation of m_i for decreasing values of $n > i \geq 3$ followed by $\#(C_4)$ such that m_{i+1} and m_i are seperated by sufficiently many 0's. As a result if we are given the binary representation of N' then we can obtain m_i for all $3 \leq i \leq n$ in time polynomial in $|G|$. In fact the number of Hamilton cycles in G is the leading polynomially many bits of N' which can also be clearly computed in time polynomial in $|G|$.

Since in time polynomial in n we can output G' when we are given the graph G as input it follows that counting the number of cycles in an undirected graph is #P-hard under polynomial time many-one reductions. It is also easy to see that our counting problem is in #P from which we obtain #P-completeness for our problem under polynomial time many-one reductions.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de