Linear Matroid Intersection Is in Catalytic Logspace

Aryan Agarwala
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Yaroslav Alekseev
Technion – Israel Institute of Technology, Haifa, Israel

Antoine Vinciguerra
Technion – Israel Institute of Technology, Haifa, Israel

Abstract

Linear matroid intersection is an important problem in combinatorial optimization. Given two linear matroids over the same ground set, the linear matroid intersection problem asks you to find a common independent set of maximum size. The deep interest in linear matroid intersection is due to the fact that it generalises many classical problems in theoretical computer science, such as bipartite matching, edge disjoint spanning trees, rainbow spanning tree, and many more.

We study this problem in the model of catalytic computation: space-bounded machines are granted access to *catalytic space*, which is additional working memory that is full with arbitrary data that must be preserved at the end of its computation.

Although linear matroid intersection has had a polynomial time algorithm for over 50 years, it remains an important open problem to show that linear matroid intersection belongs to any well studied subclass of P. We address this problem for the class catalytic logspace (CL) with a polynomial time bound (CLP).

Recently, Agarwala and Mertz (2025) showed that bipartite maximum matching can be computed in the class $\mathsf{CLP} \subseteq \mathsf{P}$. This was the first subclass of P shown to contain bipartite matching, and additionally the first problem outside TC^1 shown to be contained in CL. We significantly improve the result of Agarwala and Mertz by showing that linear matroid intersection can be computed in CLP.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Submodular optimization and polymatroids; Theory of computation → Computational complexity and cryptography; Mathematics of computing → Graph algorithms

Keywords and phrases Catalytic Computing, Computational Complexity, Matroid Theory, Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.3

Related Version *Full Version*: https://arxiv.org/abs/2509.06435

Funding *Yaroslav Alekseev*: Supported by ISF grant 507/24.
Antoine Vinciguerra: Supported by ISF grant 507/24.

Acknowledgements All three authors thank Ian Mertz and Yuval Filmus for extensive discussions.

1 Introduction

1.1 Catalytic Computing

Catalytic computation was introduced by Buhrman et al. [13] in order to study the power of used space. In this model, a space-bounded Turing machine is augmented with an additional read-write tape, known as the *catalytic tape*. The catalytic tape is initialized adversarially with some arbitrary content τ. The Turing machine may use this tape freely, with the requirement that upon termination the catalytic tape must be reset to its original state τ.

CL is the class of problems that can be solved by a catalytic machine with a work tape of size $O(\log n)$ and a catalytic tape of size poly(n). CLP is the class formed by the additional restriction that the machine must run in polynomial time.

Although it was earlier informally conjectured [22] that used space could not provide additional computational power, Buhrman et al. [13] showed the surprising result that CLP[1] is likely much stronger than L:

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{TC}^1 \subseteq \mathsf{CLP} \subseteq \mathsf{CL} \subseteq \mathsf{LOSSY} \subseteq \mathsf{ZPP}$$

Following the work of [13], catalytic computation has been a subject of growing interest, and many variants of the model have been studied, including non-deterministic and randomized [15, 23, 17, 37], non-uniform [52, 55, 20, 21], error-prone [31, 28], communication [53], and many more [30, 10, 9, 14] (see surveys by Koucký [36] and Mertz [48]). This interest in catalytic computation culminated in space efficient tree evaluation algorithms by Cook and Mertz [18, 19, 20, 21], which recently led to the breakthrough result $\mathsf{TIME}(t) \subseteq \mathsf{SPACE}(\sqrt{t \log t})$ by Ryan Williams [59].

Despite this long line of work, however, the exact strength of catalytic computation remains unclear. Of particular interest is the relationship between CL and the NC hierarchy. Buhrman et al. [13] showed that $\mathsf{TC}^1 \subseteq \mathsf{CLP}$, so the two natural questions which follow are:

1. Is $\mathsf{NC}^2 \subseteq \mathsf{CLP}$? That is, can the TC^1 inclusion of [13] be strengthened?
2. Is $\mathsf{CLP} \subseteq \mathsf{NC}$? That is, can CLP be shown to be contained in the NC (or equivalently TC) hierarchy?

On the first problem, Alekseev et al. [4] recently made progress by showing that SAC^2 can be solved with $O(\log^2 n/ \log\log n)$ free space and $2^{O(\log^{1+\epsilon} n)}$ catalytic space. On the second problem, Agarwala and Mertz [1] recently presented a barrier by showing that bipartite matching, which is currently incomparable to the NC hierarchy, is contained in CLP. This was the first new problem shown to lie in CL, and thus CLP, since the decade old result $\mathsf{TC}^1 \subseteq \mathsf{CLP}$ [13]. A natural open problem posed in [1] is to extend their framework to solve harder problems in CLP. One such problem is linear matroid intersection.

1.2 Linear Matroid Intersection

A *matroid*, defined by Whitney [58], is a set-independence structure which naturally arises in many combinatorial optimization problems. Formally, a matroid is a pair $M = (S, \mathcal{I})$, where S is some finite set and $\mathcal{I} \subseteq 2^S$ is a collection of subsets of S called independent sets. The independent sets are required to satisfy three properties: the empty set is independent, the independent sets are downward closed, and the augmentation property. See the preliminaries for a formal definition.

In the *matroid intersection* problem, one is given two matroids over the same ground set, say $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$. The goal is to find $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size. This problem is inherently challenging because while M_1 and M_2 are matroids, their intersection $(S, \mathcal{I}_1 \cap \mathcal{I}_2)$ may not be. In this paper we work exclusively with a well studied class of matroids known as *linear matroids*.

A linear matroid $M = (S, \mathcal{I})$ is one where the elements are vectors of a vector space, i.e, $S \subseteq \mathbb{F}^m$, and the independent sets $\mathcal{I}$ are exactly the sets of vectors in S which are linearly independent. The Linear Matroid Intersection problem is the matroid intersection problem where the input matroids M_1 and M_2 are linear matroids.

[1] Buhrman et al. did not define the class CLP in [13], but proved the equivalent statement that there is a CL machine running in polynomial time which simulates TC^1. The class CLP was defined later in order to study the question of whether $\mathsf{CL} \subseteq \mathsf{P}$.

Many important problems in combinatorial optimization are special cases of linear matroid intersection. For example:

- **Bipartite maximum matching:** Given a bipartite graph G, output a matching of maximum size.
- **Rainbow spanning tree:** Given an edge coloured graph G, output a spanning tree consisting of distinctly coloured edges.
- **Edge disjoint spanning trees:** Given a graph G, output two edge disjoint spanning trees.

Bipartite matching, in particular, is closely related to linear matroid intersection. As mentioned above, bipartite matching is a special case of linear matroid intersection. On the other hand, algorithms for bipartite matching tend to influence algorithms for linear matroid intersection. The augmenting paths framework for bipartite matching [41] led to polynomial time algorithms for linear matroid intersection [3, 42], the isolation lemma framework for bipartite matching [49] led to an RNC algorithm for linear matroid intersection [50], and the Quasi-NC algorithm for bipartite matching [27] led to a Quasi-NC algorithm for linear matroid intersection [32].

Recently, Agarwala and Mertz [1] showed that bipartite maximum matching is in CLP. It is then a natural question to ask whether their techniques can be extended to work for linear matroid intersection.

1.3 Isolation Lemma

A key part of this paper is the celebrated *isolation lemma*. Let S be a ground set and $\mathcal{I} \subseteq 2^S$ be *any* collection of subsets of S. The isolation lemma states that if one assigns polynomially-bounded integer weights uniformly and independently at random to each element of S, then the minimum weight set $I \in \mathcal{I}$ will be unique with high probability.

Since its introduction by Mulmuley, Vazirani, and Vazirani [49], the isolation lemma has served as a key tool in designing randomised algorithms for a wide range of classical problems [51, 43, 50, 32, 35, 54, 11, 34, 57, 7, 1, 6]. Because of this broad applicability, the question of derandomizing the lemma has attracted significant attention [16, 7, 2, 33].
The work of Narayanan et al. [50] used the isolation lemma to obtain an RNC algorithm for linear matroid intersection. In particular, given input matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$, they need polynomially bounded weights $w : S \to \mathbb{Z}$ such that the minimum weight maximum sized common independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ is unique. Gurjar and Thierauf [32] partially derandomized the isolation lemma for linear matroid intersection in Quasi-NC, but obtain a weight assignment which has large quasi-polynomially (instead of polynomially) bounded weights. It is a long-standing open problem to fully derandomize the isolation lemma for linear matroid intersection in NC.

Agarwala and Mertz [1] made progress on this problem by providing a derandomization for the case of bipartite matching in CL. It is then a natural goal to obtain a similar CL derandomization of the isolation lemma for linear matroid intersection.

1.4 Our Results

In this paper we prove the following:

▶ **Theorem 1.**

Linear Matroid Intersection $\in$ CLP

This result is interesting for two reasons:

1. Linear Matroid Intersection is now, informally speaking, the hardest problem known to be solvable in CL. The previous strongest inclusion was bipartite matching [1], which is a special case of linear matroid intersection. Thus, our result constitutes a stronger barrier against the CL $\subseteq$ NC conjecture [36, 48].
2. This is the first algorithm for Linear Matroid Intersection which uses sublinear free space and polynomial time with access to any additional resources, such as randomness, nondeterminism, or, in our case, catalytic space. As far as we are aware, the only other sublinear space algorithm uses $O(\log^2 n)$ space but runs in quasi-polynomial time, as a corollary of the fact that Linear Matroid Intersection $\in$ Quasi-NC^2 [32].

Moreover, our algorithm, which is a natural extension of the algorithm of [1], derandomizes the isolation lemma for linear matroid intersection in CLP.

1.5 Proof Overview

We present here a high-level overview of our proof. Our proof structure is largely inspired by the techniques used to prove that maximum bipartite matching is in CLP [1], with extra machinery needed to handle the more complicated structure of matroid intersection.

The core idea of the proof is to construct an isolating weight assignment on the catalytic tape, and then apply the algorithm of of [50] to compute a maximum size common independent set. Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be the input linear matroids. We start by dividing the catalytic tape into three sections:

1. A weight assignment $W : S \to \mathbb{Z}$,
2. A set of "reserve weights" used to modify W, and
3. A section used as catalytic space for the computation of catalytic subroutines.

Our algorithm proceeds iteratively with a counter k, starting with $k = 0$. At each step, we maintain the invariant that the current weight assignment W induces a unique minimum weight (or *isolates* a) size k common independent set $I_k \in \mathcal{I}_1 \cap \mathcal{I}_2$. We then perform the following steps:

1. First, we check if I_k is a common independent set of maximum size. If it is, we output I_k as our solution.
2. If I_k is not a maximum size common independent, we check if W also isolates a size $k+1$ common independent set I_{k+1}. If it does, we increment k by one and repeat the process from step 1.
3. If W does not isolate a size $k+1$ common independent set, we must start again with a new weight assignment. This is the crucial step. We swap the weight of a special element s, in the first section, with an arbitrary weight from the reserve section. We then use a novel compression-decompression algorithm to compress the reserve section. We reset our counter k to 0 and begin the process again with the modified weights.

This iterative process stops when either:

1. Step 1 is successful and we find a maximum sized common independent set through a good weight assignment W' on the catalytic tape, or
2. We visit step 3 poly(n) many times, at which point we have freed up poly(n) space on the catalytic tape through compression, and we may use this space to run a standard polynomial-time algorithm for linear matroid intersection.

We can then use our decompression algorithm to revert the catalytic tape to its original state. This is an implementation of the compress-or-random framework introduced by Cook et al. [17].

Our main contribution is a novel compression and decompression scheme. Note that if our weight assignment W does not isolate a size $k+1$ common independent set, then there must exist at least two size $k+1$ common independent sets I and I'. We can thus find an element $s \in I \setminus I'$. We call s a *threshold element.* [2] Agarwala and Mertz [1] observe that, in the context of bipartite matching, the weight of a threshold element can be deleted and later reconstructed given s and k. However, linear matroid intersection differs from bipartite matching in two ways:

1. In the case of bipartite matching, one can always ensure that $s \notin I_k$ (s is not in the isolated size k matching). In the case of linear matroids, this is not always possible – all threshold elements may be in I_k.
2. Agarwala and Mertz [1] use a structure known as the *residual graph* in their compression and decompression procedure. The key property they use is that shortest paths in the residual graph are in bijection with minimum weight size $k+1$ matchings of the original graph. We use a similar structure, known as the *exchange graph*, for matroids. However, this bijection property no longer holds.

We handle both of these issues using *inclusion* and *exclusion* matroids. In particular, in order to execute compression and decompression using a threshold element s, we need to answer two questions:

1. What is the minimum weight of a size $k+1$ common independent set containing s (excluding the weight of s itself)? These independent sets are characterised by the inclusion matroid.
2. What is the minimum weight of a $k+1$ common independent set forbidden from containing s? These independent sets are characterised by the exclusion matroid.

Agarwala and Mertz [1] showed that, given a threshold element s, and both of the aforementioned values, one can recover the weight of s as the difference of the first and second value. Thus, our main contribution is a CLP algorithm which computes both of these values.

1.6 Organization of the Paper

Our paper is divided into five main sections:

In Section 2 we formally introduce catalytic classes and matroids, and describe some generic catalytic subroutines that we will use later. In Section 3 we present a CLP algorithm which, given access to a weight function W which isolates a size k common independent set, constructs and outputs the isolated set. This is largely based on the algorithm in [50]. In Section 4, we present a CLP algorithm which decides whether a common independent set I_k is of maximum size. In Section 5 we present a CLP algorithm which either certifies that W isolates a size $k+1$ common independent set, or identifies a threshold element. We need these algorithms in the case where I_k is not of maximum size. In Section 6, we present our compression and decompression procedures, and describe the final CLP algorithm for Linear Matroid Intersection.

[2] The concept of threshold elements was introduced by Mulmuley, Vazirani, and Vazirani [49] in their proof of the isolation lemma.

2 Preliminaries

We use notation $\mathbb{Z}^{\leq c}$ to denote the non-negative integers of value at most c.

For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

Let $G(V, E)$ be a graph, for any walk P of G, we define the hop-length of P to be the number of edges in P. This is in order to distinguish from the weight of a walk when we work with weighted graphs.

2.1 Catalytic Computation

Our main computational model in this paper is the catalytic space model:

▶ **Definition 2** (Catalytic machines). *Let $s := s(n)$ and $c := c(n)$. A* catalytic Turing machine *with space s and* catalytic space *c is a Turing machine M with a read-only input tape of length n, a write-only output tape, a read-write work tape of length s, and a second read-write work tape of length c called the* catalytic tape*, which will be initialized to an adversarial string τ.*

We say that M computes a function f if for every $x \in \{0,1\}^n$ and $\tau \in \{0,1\}^c$, the result of executing M on input x with initial catalytic tape τ fulfils two properties: 1) M halts with $f(x)$ written on the output tape; and 2) M halts with the catalytic tape in state τ.

Such machines naturally give rise to complexity classes of interest:

▶ **Definition 3** (Catalytic classes). *We define $\mathsf{CSPACE}[s, c]$ to be the family of functions computable by catalytic Turing machines with space s and catalytic space c. We also define* catalytic logspace *as*

$$\mathsf{CL} := \bigcup_{d \in \mathbb{N}} \mathsf{CSPACE}[d \log n, n^d]$$

Furthermore we define CLP as the family of functions computable by CL machines that are additionally restricted to run in polynomial time for every initial catalytic tape τ.

Important to this work will be the fact, due to Buhrman et al. [13], that CLP can simulate log-depth threshold circuits:

▶ **Theorem 4** ([13]).

$$\mathsf{TC}^1 \subseteq \mathsf{CLP}$$

The algorithm we present in this paper is in CLP. We do not explicitly argue this due to the following theorem from Cook et al. [17], that any problem solvable independently in CL and in P, can be solved in CLP.

▶ **Theorem 5** ([17]).

$$\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$$

Due to the fact that linear matroid intersection is known to be in P [26, 25, 24], the main goal of this paper is to prove that there is a CL algorithm for Linear Matroid Intersection:

▶ **Theorem 6.**

$$\mathsf{Linear\ Matroid\ Intersection} \in \mathsf{CL}$$

We thus obtain Theorem 1 as a corollary of Theorem 6 and Theorem 5.

2.2 Matroids

We denote by $S \subseteq [n]$ a finite set, where $n \in \mathbb{N}$.

▶ **Definition 7** (Matroid). *A matroid M is a tuple $(S, \mathcal{I})$, where S is called the ground set, and $\mathcal{I} \subseteq 2^S$ is a collection of subsets of S, known as "independent sets". The following properties must hold for $(S, \mathcal{I})$ to be a matroid:*

- *$\emptyset \in \mathcal{I}$. The empty set is independent.*
- *If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$. The independent sets are downward closed.*
- *If $A, B \in \mathcal{I}$ where $|A| > |B|$, then there exists $x \in A \setminus B$ such that $B \cup \{x\} \in I$.*

The inclusion-wise maximal sets of $\mathcal{I}$ are referred to as the "bases" of M. The rank of M is defined to be the size of the largest independent set in $\mathcal{I}$:

$$\text{rank}(M) \coloneqq \max_{I \in \mathcal{I}} |I|.$$

In the rest of the paper, we will consider *weighted* matroids.

A weighted matroid is a matroid $M = (S, \mathcal{I})$, where the elements of S are given integer weights $W \colon S \to \mathbb{Z}$. The weight of an independent set $I \subseteq S$ is defined to be $W(I) = \sum_{s \in I} W(s)$.

▶ **Definition 8** (Common Independent Set). *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be weighted matroids with weights $W \colon S \to \mathbb{Z}$.*

$I \subseteq S$ is defined to be a "common independent set" of M_1 and M_2 if $I \in \mathcal{I}_1 \cap \mathcal{I}_2$.

Additionally, we define $\min_k(M_1, M_2) = \min\{W(I) \mid I \in \mathcal{I}_1 \cap \mathcal{I}_2, |I| = k\}$ to be the minimum weight of a common independent set of size k, and $\mathcal{I}^k_{\min}(M_1, M_2) = \{I \in \mathcal{I}_1 \cap \mathcal{I}_2 \mid W(I) = \min_k(M_1, M_2)\}$ to be the set of minimum weight size k common independent sets.

Naively, a matroid may have an exponential (in $|S|$) number of independent sets. A fundamental challenge of formalizing computational tasks on matroids is in finding a succinct description of the independent sets. In this work, we study a large class of matroids known as *linear matroids*, which can be represented succinctly by matrices.

▶ **Definition 9** (Linear Matroids). *Let $M = (S = \{s_1, \ldots, s_n\}, \mathcal{I})$ be a matroid, and A be a matrix of dimensions $m \times n$ over a field $\mathbb{F}$. For $i \in [n]$, let $A_i \in \mathbb{F}^m$ refer to the i^{th} column of the matrix A. A is a linear representation of the matroid M if:*

$$\forall I \subseteq [n], \ \{s_i \mid i \in I\} \in \mathcal{I} \iff \{A_i \mid i \in I\} \textit{ is linearly independent over } \mathbb{F}^m$$

A matroid M is defined to be a linear matroid if it can be linearly represented by a matrix A. For notational convenience, when no confusion arises, we will often refer to the matroid M by its corresponding matrix.

▶ **Definition 10** (Linear Matroid Intersection). *The* Linear Matroid Intersection *problem takes as input two linear matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ in the form of their linear representations, and outputs a maximum sized common independent set I of M_1 and M_2.*

Let us now define for every matroid $M = (S, \mathcal{I})$, and every element $s \in S$, two associated matroids.

▶ **Definition 11** (Exclusion and Inclusion Matroids). *Let $M = (S, \mathcal{I})$, $W \colon S \to \mathbb{Z}$ be a weighted matroid, and let $s \in S$.*

The exclusion matroid $M - s$ *is defined as* $M - s = (S \setminus \{s\}, \{I \in \mathcal{I} \mid s \notin I)$ *with weights* $W \restriction_{S\setminus\{s\}}$.

The inclusion matroid $M \restriction_s$ *is defined as* $M \restriction_s = (S \setminus \{s\},\ \{I \setminus \{s\} \mid I \in \mathcal{I},\ s \in I\})$ *with weights* $W \restriction_{S\setminus\{s\}}$. [3]

▶ **Definition 12** (Membership Oracle). *For a matroid* $M = (S, \mathcal{I})$*, a membership oracle* $\mathcal{O}$ *takes as input* $I \subseteq S$ *and accepts if and only if* $I \in \mathcal{I}$.

A key property of linear matroids, and thus inclusion/exclusion matroids built from linear matroids, is that membership can be decided in CL.

▶ **Lemma 13.** *Given a linear matroid* $M = (S, \mathcal{I})$ *and* $I \subseteq S$*, there exists a* CL *algorithm which tests whether* $I \in \mathcal{I}$.

Proof. Testing whether $I \in \mathcal{I}$ involves only testing whether the linear representation of M, when restriction to the columns of I, has rank $|I|$. This can be done in TC^1 and thus CL [5, 13]. ◀

▶ **Lemma 14.** *Given a linear matroid* $M = (S, \mathcal{I})$*,* $s \in S$*, and* $I \subseteq S \setminus \{s\}$*, there exist* CL *algorithms to test whether* I *is independent for both the exclusion matroid* $M - s$ *and the inclusion matroid* $M \restriction_s$.

Proof. For the exclusion matroid, I is independent if and only if I is independent in M. For the inclusion matroid, I is independent if and only if $I \cup \{s\}$ is independent in M. Both can be tested in CL using Lemma 13. ◀

From now on, unless explicitly stated otherwise, all matroids in this paper will be either linear matroids, or inclusion/exclusion matroids built from linear matroids.

2.3 Graph Algorithms

We first show two weighted reachability problems that are in NL.

▶ **Lemma 15.** *There exists an* NL *algorithm which, given a directed graph* $G = (V, E)$*, vertex weights* $l : V \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$*, sets* $X_1, X_2 \subseteq V$*, and an integer* $L \in \mathbb{Z}^{\leq \mathrm{poly}(n)}$*, decides whether there exists in* G *an* $X_1 - X_2$ *walk of weight at most* L *and hop-length at most* $|V|$.

Proof. We non-deterministically explore the graph with a walk. First, we non-deterministically pick the first vertex v_1 of the walk. If $v_1 \notin X_1$, we reject. Assume that at stage j the algorithm has explored the walk $\{v_1, v_2, \ldots, v_j\}$. We store the last vertex v_j, the weight of the walk $cw = \sum_{i=1}^{j} l(v_i)$, and the hop-length $j - 1$. There are now three cases:

1. If $v_j \in X_2$ and $cw \leq L$, we accept.
2. Else if $j \geq |V|$, we reject.
3. Else, we non-deterministically pick a vertex v_{j+1}. If $(v_j, v_{j+1}) \notin E$, we reject. Otherwise we continue to the next stage.

This procedure always terminates eventually because, if case 1 is never reached, then j grows by 1 at each stage and eventually exceeds $|V|$, at which point case 2 is reached. It accepts if and only if it finds a walk whose weight is at most L and hop-length at most $|V|$. ◀

[3] The exclusion matroid is always a matroid by definition. The inclusion matroid is a matroid if and only if s is not a loop. That is, $\{s\} \in \mathcal{I}$. We will assume, without loss of generality, that this is always the case.

▶ **Corollary 16.** *There exists an* NL *algorithm which, Given a directed graph* $G = (V, E)$*, vertex weights* $l : V \to \mathbb{Z}^{\leq \text{poly}(n)}$*, sets* $X_1, X_2 \subseteq V$*, such that* G *does not have any negative weight cycles with respect to* l*, decides whether there exists an* $X_1 - X_2$ *simple path of weight at most* L*.*

Proof. If all cycles in the graph have non-negative weight, then it is easy to see that there exists a simple path of weight $\leq L$ if and only if there exists a walk of weight $\leq L$ and hop-length $\leq |V|$. Thus, we can simply apply Lemma 15 to solve this in NL. ◀

We deduce the following catalytic algorithm.

▶ **Lemma 17.** *There exists a* CL *algorithm which, given a directed graph* $G = (V, E)$ *with vertex weights* $l : V \to \mathbb{Z}^{\leq \text{poly}(n)}$ *such that* G *does not have any negative weight cycles with respect to* l*, along with sets* $X_1, X_2 \subseteq V$*, computes the minimum weight of a simple* X_1*-*X_2 *path, or concludes that such a path does not exist.*

Proof. Since NL $\subseteq$ CL [13], the CL algorithm can simply iterate through all possible L, starting from $L = 0$, in ascending order, and pick the first one such that Corollary 16 returns true. If Corollary 16 does not return true for any $L \leq \sum_{v \in V} l(v) = \text{poly}(n)$, then our CL algorithm can conclude that there does not exist any X_1-X_2 path in G. ◀

Finally, we present a catalytic subroutine to find the minimum weight and minimum weight hop-length simple cycle in a graph. This algorithm proceeds by a folklore reduction to minimum weight bipartite perfect matching, and then uses the algorithm of [1] as a black-box. See the full version of this paper for a proof.

▶ **Lemma 18.** *There exists a* CL *algorithm which, given a directed graph* $G = (V, E)$ *with vertex weights* $l : V \to \mathbb{Z}^{\leq \text{poly}(n)}$*, such that* G *does not have any negative weight cycles with respect to* l*, along with a special vertex* $c \in V$*, computes a minimum weight minimum hop-length simple cycle of* G *containing* c*.*

3 Isolation Lemma and Minimum Weight k Linear Matroid Intersection

Recent advancements in the complexity of linear matroid intersection rely on the isolation lemma (see [49, 50, 32]). This lemma is crucial because it allows these problems to be solved in parallel by demonstrating that a random weight assignment will, with high probability, yield a unique optimal solution.

▶ **Definition 19** (Isolation)**.** *Let* U *be a set and* $W : U \mapsto \mathbb{Z}$*. Let* $\mathcal{F}$ *be a family of subsets of* U*. We say that* W *isolates a set* $S \in \mathcal{F}$ *from* $\mathcal{F}$ *if for all sets* $T \in \mathcal{F}, T \neq S$*, we have* $W(T) > W(S)$*.*

It was proven in [49], that given a bipartite graph, an edge e, and an edge weight assignment W with the promise that W isolates a perfect matching M of G, there is an L^{DET} machine (where DET is the matrix determinant problem) which decides if $e \in M$. A similar statement was proven for linear matroid intersection in [50].

Observing that DET $\in$ TC^1 $\subseteq$ CL, it was proven in [1] that given a weight assignment W which isolates a size k matching M_k of G, there exists a CL algorithm which outputs this matching. In this section, we will prove the same statement for linear matroid intersection.

▶ **Theorem 20** ([50]). *There exists a* CL *algorithm which:*
1. *Takes as input weighted linear matroids* $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W\colon S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ *with linear representations* L_1 *and* L_2 *respectively, both with dimensions* $m \times n$*.*
2. *If the minimum weight perfect common independent set* I *(i.e.* $|I| = m$*) of* M_1 *and* M_2 *is unique with respect to* W*, it produces as output the set* I*.*

Proof. The only computationally heavy subroutines used in the algorithm of [50] are:
1. Computing the product of $\mathrm{poly}(n) \times \mathrm{poly}(n)$ dimension matrices with $\mathrm{poly}(n)$ bit entries. This is in $\mathsf{L} \subseteq \mathsf{CL}$.
2. Computing the determinant of a $\mathrm{poly}(n) \times \mathrm{poly}(n)$ dimension matrix with $\mathrm{poly}(n)$ bit entries. This is in $\mathsf{GapL} \subseteq \mathsf{TC}^1 \subseteq \mathsf{CL}$ [47].
3. Interpolating the $\mathrm{poly}(n)$-bit coefficients of a CL computable univariate polynomial of degree $\mathrm{poly}(n)$. It is known that this reduces to multiplying the inverse of a $\mathrm{poly}(n) \times \mathrm{poly}(n)$ dimension Vandermonde matrix having $\mathrm{poly}(n)$ bit entries, with a $\mathrm{poly}(n)$ sized vector consisting of $\mathrm{poly}(n)$ bit entries. Computing the inverse of a matrix reduces to computing its determinant and its cofactors, so this is in $\mathsf{TC}^1 \subseteq \mathsf{CL}$ by combining the previous two facts.

These facts combine to show that the algorithm described in Theorem 4.1 of [50] can be implemented in CL. ◀

We now reduce the computation of an isolated size k common independent set to the computation of an isolated perfect common independent set.

▶ **Lemma 21.** *There exists a* L *machine which:*
1. *Takes as input weighted linear matroids* $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W\colon S \to Z^{\leq \mathrm{poly}(n)}$ *with linear representations* L_1 *and* L_2*.*
2. *Computes weighted linear matroids* $M_1' = (S \cup S', \mathcal{I}_1'),\ M_2' = (S \cup S', \mathcal{I}_2'),\ W' : S \cup S' \to Z^{\leq \mathrm{poly}(n)}$ *such that if* W *isolates a size* k *common independent set* I_k *of* M_1 *and* M_2*, then* W' *isolates a perfect common independent set* I *of* M_1' *and* M_2' *which satisfies* $I \cap S = I_k$*.*

The proof of this lemma is technical but not particularly insightful. See the full version of this paper for a proof.

▶ **Lemma 22.** *There exists a* CL *algorithm which, given as input linear matroids* $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W\colon S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$*, and* $k \in [|S|]$ *such that* W *isolates a size* k *common independent set* I *of* M_1 *and* M_2*, computes* I*.*

Proof. We can simply apply the reduction in Lemma 21, and then the algorithm in Theorem 20, in order to obtain I. ◀

Finally, let us present a slight modification of Lemma 22. Our compression and decompression scheme will need to be able to compute an isolated size k common independent set I_k without having access to the weight of a "compressed" element s. The following lemma shows that this is possible in CL.

▶ **Lemma 23.** *Let* $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W\colon S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ *be weighted linear matroids, and* $k \in [|S|]$ *be such that* W *isolates a size* k *common independent set* I_k *of* M_1 *and* M_2*.*

Let $s \in S$ *be a "compressed" element and let* $b = \mathbb{I}\left[s \in I_k\right]$*.*

There exists a CL *algorithm which, given as input the matroids* M_1 *and* M_2*,* k*, the element* s*, the compressed weight assignment* $W \restriction_{S \setminus s}$*, and the boolean* b*, computes* I_k*.*

Proof. Let $|W| = \sum_{x \in S \setminus s} |W(x)|$. Consider the following weight function W':

$$W'(x) = \begin{cases} W(x) & x \neq s \\ |W| \cdot (-1)^b & x = s \end{cases}$$

Clearly, $W'(x) \leq \text{poly}(n)$, and I_k is the unique minimum weight size k common independent set of M_1 and M_2 under weights W'.

We can now simply apply the algorithm in Lemma 22 on $M_1,\ M_2$ with weights W' in order to obtain I_k in CL. ◀

4 Maximum Common Independent Set, and the Exchange Graph

In the previous section, we showed that if a weight assignment W isolates a size k common independent set I_k, we can compute I_k in CL.

Our goal is now to decide whether this set I_k is a maximum sized common independent set. If it is, we can simply output it as our final solution.

In order to check if I_k is maximum, we will use a standard graph-theoretic tool from matroid theory known as the *exchange graph*. This graph generalizes the use of residual graphs and augmenting paths for maximum bipartite matching. For maximum bipartite matching, maximality testing can be reduced to a s-t reachability problem in the residual graph [8]. A similar statement is true for matroid intersection in the exchange graph.

▶ **Definition 24** (Exchange Graph). *Let $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W \colon S \to \mathbb{Z}$ be weighted matroids, and let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ be a common independent set.*

The exchange graph $\mathcal{E}_{M_1,M_2,I} = (S, E)$ is a vertex weighted directed graph. Let $x \notin I$ and $y \in I$. The edge set E is defined as follows:

1. $(y, x) \in E \iff I - \{y\} \cup \{x\} \in \mathcal{I}_1$
2. $(x, y) \in E \iff I - \{y\} \cup \{x\} \in \mathcal{I}_2$.

The vertex weights $l : S \to \mathbb{Z}$ are defined by:

$$l(s) = \begin{cases} W(s), & \text{if } s \notin I \\ -W(s), & \text{if } s \in I \end{cases}$$

Furthermore, $\mathcal{E}_{M_1,M_2,I}$ has two special vertex sets X_1 and X_2 defined as follows:

$$X_1 = \{x \in S \setminus I \mid I \cup \{x\} \in \mathcal{I}_1\}$$

$$X_2 = \{x \in S \setminus I \mid I \cup \{x\} \in \mathcal{I}_2\}$$

We observe that we can compute the Exchange Graph in CL.

▶ **Lemma 25.** *There exists a* CL *algorithm which:*

1. *Takes as input weighted matroids $M_1 = (S, \mathcal{I}_1),\ M_2 = (S, \mathcal{I}_2),\ W \colon S \to \mathbb{Z}^{\leq \text{poly}(n)}$ such that membership testing for M_1 and M_2 can be done in* CL*, along with $I \in \mathcal{I}_1 \cap \mathcal{I}_2$,*
2. *Computes the exchange graph $\mathcal{E}_{M_1,M_2,I}$.*

Proof. There are four parts of $\mathcal{E}$ that the machine needs to compute: the vertices, the edges, the vertex weights, and the special sets X_1, X_2:

1. The vertices of $\mathcal{E}_{M_1,M_2,I}$ are simply S.
2. The vertex weight of any vertex $x \in S$ is simply $-W(x)$ if $x \in I$, and $W(x)$ otherwise.

3. The edges of G can be computed as follows: for $x \in I$ and $y \notin I$, the edge (x, y) is in the graph if and only if $I - \{x\} \cup \{y\} \in \mathcal{I}_1$. Since membership testing for M_1 is assumed to be in CL, this can be decided in CL. The inclusion of edge (y, x) can be determined in the same way for M_2.
4. X_1 and X_2 can be computed as follows: $x \in X_i \iff I \cup \{x\} \in \mathcal{I}_i$. Again, this is simply membership testing for M_1 and M_2.

This completes the proof. ◀

The exchange graph is a well-known and extensively studied object. Notably, the problem of determining whether a common independent set I is of maximum size can be solved by deciding s-t reachability in the exchange graph.

▶ **Lemma 26** ([25, 26]). *A common independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ is of maximum size if and only if there is no path from X_1 to X_2 in the exchange graph $\mathcal{E}_{M_1,M_2,I}$.*

An immediate corollary of is the following:

▶ **Lemma 27.** *There exists a CL algorithm which:*
1. *Takes as input weighted matroids $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ such that membership testing for M_1 and M_2 can be done in CL, along with a size k common independent set I_k of M_1 and M_2.*
2. *Decides whether a size $k+1$ common independent set of M_1 and M_2 exists.*

Proof. The following CL algorithm works:
1. Compute the exchange graph $\mathcal{E}_{M_1,M_2,I_k}$ (Lemma 25).
2. Test whether an $X_1 - X_2$ path in $\mathcal{E}_{M_1,M_2,I_k}$ exists. (Lemma 17).

Correctness follows directly from Lemma 26. ◀

5 Checking if W Isolates a Size $k+1$ Common Independent Set

In the last two sections, we provided catalytic algorithms for computing an isolated size k common independent set I_k, and then testing whether I_k is maximum. If not, we know that a size $k+1$ common independent set exists. Our goal in this section is to either certify that W also isolates a size $k+1$ common independent set, or to show how to compress (and later decompress) the catalytic tape if it doesn't.

5.1 More Facts About the Exchange Graph

We presented in Lemma 26 a connection between the paths in the exchange graph and the existence of common independent sets of size $k+1$. We now present slightly deeper properties about this connection:

▶ **Theorem 28** ([12, 38, 39, 40, 29, 25, 26]). *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}$ be weighted matroids. Let I be a minimum weight, not necessarily unique, size k common independent set of M_1 and M_2.*

Let P be a minimum weight minimum hop-length path from X_1 to X_2 in $\mathcal{E}_{M_1,M_2,I_k}$.

The following is known:
1. *$\mathcal{E}_{M_1,M_2,I_k}$ does not contain any negative weight cycles.*
2. *$I_{k+1} = I_k \Delta P$ is a minimum weight size $k+1$ common independent set of M_1 and M_2.*

Theorem 28 is described in detail in Section 41.3 of the textbook "Combinatorial Optimization" by Alexander Schrijver [56]. [4]

Additionally, in the case where W isolates a size k common independent set, we observe that all cycles must have strictly positive weight.

▶ **Lemma 29.** *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}$ be weighted matroids such that W isolates a size k common independent set I_k of M_1 and M_2. $\mathcal{E}_{M_1,M_2,I_k}$ does not contain any weight 0 cycles.*

Proof. Lemma 41.5α from [56] states that if $\mathcal{E}_{M_1,M_2,I_k}$ contains a cycle C, then there exists a size k common independent set $I'_k \neq I_k$ such that either $W(I'_k) < W(I_k)$ or $W(I'_k) \leq W(I_k) + l(C)$. Thus, if there were a cycle C such that $l(C) = 0$, it would imply that I_k is not the unique minimum weight size k common independent set. ◀

Now, we showed in Section 3 that, when W isolates a size k common independent set I_k, we can construct I_k. We need to additionally compute, for any $s \in S$, the (not necessarily unique) minimum weight size k common independent I^{+s} such that $s \in I^{+s}$, and similarly the minimum weight size k common independent I^{-s} such that $s \notin I^{-s}$. Note that either I^{+s} or I^{-s} is I_k. In order to compute the other, we need the following fact:

▶ **Lemma 30.** *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ be weighted matroids which admit a unique minimum weight size k common independent set I_k. Let $s \in S$.*

Let C^* be the minimum weight minimum hop-length cycle C in $\mathcal{E}_{M_1,M_2,I_k}$ such that $s \in C$. Then $I = I_k \Delta C^*$ is a minimum weight size k common independent set of M_1 and M_2 such that $s \in I \iff s \notin I_k$. That is,

$$I_k \Delta C^* \in \underset{I \in \{I \in \mathcal{I}_1 \cap \mathcal{I}_2 \ \mid \ |I| = k, \ s \in I \Delta I_k\}}{\arg\min} W(I)$$

Proof. First, we will show that $I_k \Delta C^*$ is a common independent set of M_1 and M_2. Lemma 41.5α from [56] states that if $I_k \Delta C^* \notin \mathcal{I}_1 \cap \mathcal{I}_2$, then there exists either a negative weight cycle in $\mathcal{E}_{M_1,M_2,I_k}$ (which is impossible by Theorem 28), or there exists another cycle $C \subsetneq C^*$ with $l(C) \leq l(C^*)$. The latter would contradict the fact that C^* has minimum hop-length amongst the minimum weight cycles. Therefore, neither of these cases are possible, and hence $I_k \Delta C^*$ is necessarily a common independent set of M_1 and M_2. Moreover, $(I_k \Delta C^*) \Delta I_k = C^*$. Thus, $s \in (I_k \Delta C^*) \Delta I_k$.

Now, we will prove the minimality of $I_k \Delta C^*$. Let $I' \in \mathcal{I}_1 \cap \mathcal{I}_2$ be a size k common independent set such that $s \in I' \Delta I_k$. Theorem 41.5 of [56] shows that $I' \Delta I_k$ is the union of disjoint cycles in $\mathcal{E}_{M_1,M_2,I_k}$. Let these cycles be $C_1, \ldots, C_m$. Since $s \in I' \Delta I_k$, there must exist a cycle containing s, assume that it is C_1. $W(I') = W(I_k) + \sum_{i=1}^{m} l(C_i) \geq W(I_k) + l(C_1) \geq W(I_k) + l(C^*)$. The second to last step was due to the fact that all cycles have non-negative weight in $\mathcal{E}_{M_1,M_2,I_k}$. The last step was due to the fact that $l(C^*)$ is the minimum weight amongst all cycles in $\mathcal{E}_{M_1,M_2,I_k}$ which contain s.

This completes the proof. ◀

[4] Schrijver's description is in terms of maximum weight common independent sets and uses the weight function $-l$. However, it is simple to see that these descriptions are equivalent, simply by substituting the matroid weights $-w$ into Schrijver's theorems.

5.2 Threshold Elements

In order to handle the case where W does not isolate a size $k+1$ common independent set, we need to introduce the concept of *threshold elements.*

▶ **Definition 31** (Threshold Elements). *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \text{poly}(n)}$ be weighted matroids which admit a unique minimum weight size k common independent set I_k.*

An element $s \in S$ is a $k+1$-threshold element if there exist two size $k+1$ minimum weight common independent sets, I and I', such that $s \in I$ and $s \notin I'$.

The set of $k+1$-threshold elements of M_1 and M_2 is denoted by T^{k+1}.

▶ **Lemma 32** (Threshold elements exist $\iff$ the minimum weight size $k+1$ common independent set is not unique). *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}$ be weighted matroids, and let $k \in [|S|]$ such that W isolates a size k common independent set I_k of M_1 and M_2, and a size $k+1$ common independent set of M_1 and M_2 exists. The following statements are equivalent:*

1. *The minimum weight size $k+1$ common independent set of M_1 and M_2 is not unique.*
2. $|T^{k+1}| \geq 1$.

Proof. This trivially holds:

1. (1) $\implies$ (2): Let I_1 and I_2 be minimum weight size $k+1$ common independent sets of M_1 and M_2. Any element $s \in I_1 \setminus I_2$ is a threshold element. Thus, $|\mathcal{I}^{k+1}_{\min}| > 1 \implies |T^{k+1}_{M_1,M_2}| \geq 1$.
2. (2) $\implies$ (1): By definition of $T^{k+1}_{M_1,M_2}$, for any $s \in T^{k+1}_{M_1,M_2}$, there exist $X, X' \in \mathcal{I}^{k+1}_{\min}$ such that $X \neq X'$. Thus, $|T^{k+1}_{M_1,M_2}| \geq 1 \implies |\mathcal{I}^{k+1}_{\min}| > 1$

   This completes the proof. ◀

The goal is therefore simply to decide whether a threshold element exists, and if it does then to find one. In order to do this, first observe that we can define a threshold element in terms of inclusion and exclusion matroids.

▶ **Observation 33.** $s \in T^{k+1} \iff \min_k(M_1 \restriction_s, M_2 \restriction_s) + w(s) = \min_{k+1}(M_1 - s, M_2 - s)$.

We will show that for any $s \in S$, both sides of this equation can be computed in CL. Thus, we can simply iterate over all $s \in S$, compute both sides of this equation, and test if equality holds – this suffices to both decide whether a threshold element exists, and to find one if it does.

5.3 The Hunt for Threshold Elements

In this subsection, we will describe how to decide in CL if an element $s \in S$ is a threshold element, and thus how to find a threshold element if it exists. By the equation in Observation 33, this reduces to computing for a fixed s the weights $a = \min_k(M_1 \restriction_s, M_2 \restriction_s)$, $b = \min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$, and then simply checking if $b = a + w(s)$.

For this, let us first make the following observation:

▶ **Lemma 34.** *There exists a CL algorithm which, given as input $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \text{poly}(n)}$ such that membership testing for M_1 and M_2 can be done in CL, along with a minimum weight size k common independent set I_k, computes* $\min_{k+1}(M_1, M_2)$.

Proof. The following algorithm works:
1. Compute $\mathcal{E}_{M_1,M_2,I_k}$ (Lemma 25).
2. Compute the minimum weight of an X_1-X_2 path in $\mathcal{E}_{M_1,M_2,I_k}$ (Lemma 17). Let it be L.
3. Output $w(I_k) + L$.

Correctness follows from Theorem 28. ◀

Now, note that $\min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$ is simply the minimum weight of a size $k+1$ common independent set of M_1 and M_2 *not containing* s. Lemma 34 tells us that in order to compute this, it suffices to compute a minimum weight size k common independent set not containing s – let such a set be I^{-s}. Similarly, $\min_k(M_1 \restriction_s, M_2 \restriction_s)$ is simply the minimum weight of a size $k+1$ common independent set *containing* s, but excluding the weight of s itself. Lemma 34 tells us that in order to compute this, it suffices to compute a minimum weight size k common independent set containing s – let such a set be I^{+s}. Simply note that either I^{+s} or I^{-s} is I_k. Lemma 30 implies that the other can be computed by finding a minimum weight cycle C in $\mathcal{E}_{M_1,M_2,I_k}$ using Lemma 18, and then taking the symmetric difference of I_k and C.

▶ **Lemma 35.** *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \text{poly}(n)}$ be weighted linear matroids which admit a unique minimum weight size k common independent set I_k. Let $s \in S$ be an element, and let $b = \mathbb{I}[s \in I_k]$ indicate whether $s \in I_k$.*
Given M_1, M_2, $W \restriction_{S \setminus s}$, k, s, b, there exists a CL *algorithm which either:*
1. *Outputs a minimum weight size k common independent set I'_k of M_1 and M_2 such that $s \in I'_k \iff s \notin I_k$, or*
2. *Certifies that such a common independent set does not exist.*

Proof. Consider $W' : S \to \mathbb{Z}^{\leq \text{poly}(n)}$ such that $W'(x) = W(x)$, for $x \neq s$, and $W'(s) = (1 + \sum_{x \in S, x \neq s} |W(s)|) \cdot (-1)^b$.

The minimum weight size k common independent set of M_1 and M_2 with respect to W' is unique and must be I_k. Thus, it can be computed using Lemma 22.

For every $S_1, S_2 \subseteq S$ such that $s \in S_1 \iff s \in S_2$, we have $W'(S_1) \leq W'(S_2) \iff W(S_1) \leq W(S_2)$. Thus, the minimum weight size k common independent set I of M_1 and M_2 under W' such that $s \in I \iff s \notin I_k$ is the same as that under W, and must be I'_k (though I'_k need not be unique). The algorithm is fairly straightforward:
1. Compute I_k (Lemma 22).
2. Compute $\mathcal{E}_{M_1,M_2,I_k}$ under weights W' (using Lemma 25). Let the lengths be l'.
3. Compute a minimum weight minimum hop-length cycle C^* of $\mathcal{E}_{M_1,M_2,I_k}$ containing s under weights l' (using Lemma 18). If no such cycle exists, certify the second case of the statement.
4. Else, output $C^* \Delta I_k$.

The correctness of the algorithm follows from Lemma 30. ◀

▶ **Lemma 36.** *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \text{poly}(n)}$ be weighted linear matroids which admit a unique minimum weight size k common independent set I_k. Let $s \in S$ and $b = \mathbb{I}[s \in I_k]$.*
Given M_1, M_2, $W \restriction_{S \setminus s}$, s, $b = \mathbb{I}[s \in I_k]$, there exists a CL *algorithm which either:*
1. *Outputs $\min_k(M_1 \restriction_s, M_2 \restriction_s)$, or*
2. *Certifies that a size k common independent set of $M_1 \restriction_s$ and $M_2 \restriction_s$ does not exist.*

Similarly, there exists another CL *algorithm which either:*
1. *Outputs* $\min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$*, or*
2. *Certifies that a size* $k+1$ *common independent set of* $M_1 - \{s\}$ *and* $M_2 - \{s\}$ *does not exist.*

Proof. First for the inclusion matroid. Note that any $J \subseteq S \setminus \{s\}$ is a minimum weight size $k-1$ common independent set of $M_1 \restriction_s$ and $M_2 \restriction_s$ if and only if $I = J \cup \{s\}$ is a minimum weight size k common independent set of M_1 and M_2 including s.

Thus, the problem of computing a minimum weight size $k-1$ common independent set of $M_1 \restriction_s$ and $M_2 \restriction_s$ reduces simply to computing I (or certifying that it does not exist). This can be done as follows:
1. Compute I_k (Lemma 22).
2. If $b = 1$, we have $I = I_k$.
3. If $b = 0$, we can compute I using Lemma 35. If no such set exists, we can certify the second case of the statement.

Now, we have $J = I \setminus \{s\}$, a minimum weight size $k-1$ common independent set of $M_1 \restriction_s$ and $M_2 \restriction_s$. We can now:
1. Test if a size k common independent set of $M_1 \restriction_s$ and $M_2 \restriction_s$ exists (using Lemma 27). If not, we can certify the second case of the statement.
2. Else, we can compute and output $\min_k(M_1 \restriction_s, M_2 \restriction_s)$ (using Lemma 34).

For the exclusion matroid case, the algorithm is similar:
1. Compute I_k (using Lemma 23).
2. Compute a minimum weight size k common independent set I'_k of $M_1 - \{s\}$ and $M_2 - \{s\}$:
   a. If $b = 0$, this is exactly I_k.
   b. If $b = 1$, this is a minimum weight size k common independent set of M_1 and M_2 which does not include s. This can be computed using Lemma 35. If such a set does not exist, we can certify the second case of the statement.
3. Test if a size $k+1$ common independent set of $M_1 - \{s\}$ and $M_2 - \{s\}$ exists (using Lemma 27). If such a set does not exist, we can certify the second case of the statement.
4. Compute $\min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$ (using Lemma 34).

This completes the proof. ◀

Lemma 36 enables us to compute both sides of the equation in Observation 33. We now need to check, for all $s \in S$, if s satisfies this equation. If it does, s will be our threshold element.

▶ **Lemma 37.** *There exists a* CL *algorithm which, given weighted linear matroids* $M_1 = (S, \mathcal{I}_1)$*,* $M_2 = (S, \mathcal{I}_2)$*,* $W : S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ *such that the minimum weight size* k *common independent set of* M_1 *and* M_2*,* I_k*, is unique, and a size* $k+1$ *common independent set of* M_1 *and* M_2 *exists:*
1. *Computes* $s \in T^{k+1}$*. Or*
2. *Certifies that the minimum weight size* $k+1$ *common independent set of* M_1 *and* M_2 *is unique.*

Proof. For all $s \in S$, we will do the following test:
1. Compute I_k (Using Lemma 22).
2. Compute $\mathcal{E}_{M_1,M_2,I_k}$ (Using Lemma 25).
3. Compute $a = \min_k(M_1 \restriction_s, M_2 \restriction_s)$ and $b = \min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$ (Using Lemma 36).
4. If either a or b doesn't exist, continue to the next s.
5. Else, check if $b = a + W(s)$. If yes, then $s \in T^{k+1}$, so we can output s and terminate. Else, $s \notin T^{k+1}$, so we can continue to the next s.

The existence of a threshold element s is guaranteed by Lemma 32 if and only if the minimum weight size $k+1$ common independent set of M_1 and M_2 is not unique. Thus, if such an s is found, we can simply output it. If no such s is found, we can certify that the minimum weight size $k+1$ common independent set of M_1 and M_2 is unique. ◀

6 Final Algorithm

We are now ready to describe our compression and decompression algorithms.

▶ **Lemma 38.** *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be linear matroids given as input. Let (w, r, τ) be a catalytic tape where:*

1. *$|w| = |S| \cdot 10 \log|S|$. This section of the catalytic tape is interpreted as a weight assignment $w : S \to \mathbb{Z}^{\leq |S|^{10}}$.*
2. *$|r| = 10 \log|S|$. This section of the catalytic tape is interpreted as a "reserve" weight in $\mathbb{Z}^{\leq |S|^{10}}$.*
3. *$|\tau| = \mathrm{poly}(n)$. This section of the catalytic tape has no special interpretation, it simply catalytic space to be used by our catalytic subroutines.*

There exist a pair of CL *algorithms $\mathcal{C}omp$ and $\mathcal{D}ecomp$ with the following behaviour:*

1. *$\mathcal{C}omp$, when run on inputs M_1, M_2, $k \in [|S|]$, $s \in S$ with the catalytic tape containing (w, r, τ), such that w does not isolate a size $k+1$ common independent set of M_1 and M_2 (k is the minimum such size), and $s \in T^{k+1}$, outputs nothing but changes the catalytic tape to a string (w', r', τ) such that*
   a. *$w'(e) = w(e)$ for all $e \neq s$, and $w'(s) = r$.*
   b. *$r' = (0^{8 \log |S| - 1}, s, k, b = \mathbb{I}[s \in I_k])$*
2. *$\mathcal{D}ecomp$, when run on inputs M_1 and M_2 with the catalytic tape (w', r', τ) outputs nothing, but returns the catalytic tape to its original state τ.*

Proof. $\mathcal{C}omp$ simply swaps $w(s)$ with r, and then writes $(0^{8 \log |S| - 1}, s, k, b = \mathbb{I}[s \in I_k])$ in the place of r.

We now focus on the $\mathcal{D}ecomp$ procedure. Let us first make the following observation.

▶ **Lemma 39.** *Let $M_1 = (S, \mathcal{I}_1)$, $M_2 = (S, \mathcal{I}_2)$, $W : S \to \mathbb{Z}^{\leq \mathrm{poly}(n)}$ be weighted linear matroids such that the minimum weight size k common independent set of M_1 and M_2, I_k, is unique, and a size $k+1$ common independent set of M_1 and M_2 exists.*

Given M_1, M_2, $s \in T^{k+1}$, $b = \mathbb{I}[s \in I_k]$, and $W \restriction_{S \setminus \{s\}}$: $\min_{k+1}(M_1, M_2) - w(s)$ and $\min_{k+1}(M_1, M_2)$ can be computed in CL.

Proof. The following works:

1. For $\min_{k+1}(M_1, M_2) - w(s)$, simply note that this value is exactly $\min_k(M_1 \restriction_s, M_2 \restriction_s)$, which can be computed using Lemma 36.
2. On the other hand, $\min_{k+1}(M_1, M_2)$ is exactly $\min_{k+1}(M_1 - \{s\}, M_2 - \{s\})$, which can be computed again using Lemma 36. ◀

Recall that $r' = (0^{8 \log |S| - 1}, s, k, b = \mathbb{I}[s \in I_k])$. Therefore, using Lemma 39, we can compute $a = \min_{k+1}(M_1, M_2) - w(s)$ and $b = \min_{k+1}(M_1, M_2)$ from the information stored in r', and thus compute $w(s) = b - a$. Once we have $w(s)$ on the work tape, we can swap r' and $w'(s)$, and then swap $w'(s)$ and $w(s)$. This is guaranteed to return the catalytic tape to its original state.

This completes the proof. ◀

We are now ready to present our final algorithm. The idea is to proceed in stages: in each stage, if the weight assignment w on the catalytic tape isolates common independent sets of all sizes, we can compute and output a maximum sized common independent set. Else there exists k such that a size $k+1$ common independent set exists, but is not isolated by w. In this case we can use our compression procedure on the catalytic tape. After $\text{poly}(n)$ stages, we will have either solved the problem, or freed up polynomial space on the catalytic tape, at which point we can run Edmonds' polynomial time algorithm [25, 26, 24] directly on the catalytic tape.

Let us present this CL algorithm in detail.

▶ **Theorem 40** (Restatement of Theorem 6)**.** Linear Matroid Intersection *is in* CL.

Proof. Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be linear matroids given as input, and let us define $n = |S|$.

Let T be the time taken by a strongly polynomial time algorithm $\mathcal{A}$ to solve linear matroid intersection on M_1 and M_2 (we only need T to be a $\text{poly}(n)$ upper bound, so it is easy to compute in CL).

The catalytic tape is partitioned as follows:

- The first section comprises of $10 \cdot n \log n$ bits. It is interpreted as n blocks of $10 \log n$ bits each, which correspond to a weight assignment $W : S \to \mathbb{Z}^{\leq n^{10}}$.
- The second section will hold $T \cdot 10 \log n$ bits. It is interpreted as T blocks of $10 \log n$ bits each, each block corresponding to an element of $\mathbb{Z}^{\leq n^{10}}$. Let these be $r_1, \ldots, r_T$
- The final section τ has no special interpretation. It is simply catalytic space to be used by our catalytic subroutines.

On the work tape we maintain two counters $k \in [n]$ and $j \in [T]$. k is initialised to 0 and j is initialised to 1.

The algorithm iterates through k while maintaining the invariant that the weights W isolate a size k common independent I_k of M_1 and M_2. At each step of the iteration, we do the following:

1. Check maximality.
   The algorithm first checks if I_k is a maximum sized common independent set of M_1 and M_2 by computing it using Lemma 22 and then using the algorithm in Lemma 27. If it is of maximum size, the algorithm simply outputs I_k.
2. Check if W isolates a $k+1$-size common independent set.
   If I_k is not a maximum sized common independent set of M_1 and M_2, the algorithm checks if W isolates a size $k+1$ common independent set, or else finds a threshold element $s \in T^{k+1}$ using Lemma 37. If the algorithm does not find any threshold element, it can simply increment k and return to step 1. Else, it proceeds to the next step.
3. Compression.
   We have a threshold element $s \in T^{k+1}$. We can now use the $\mathcal{C}omp$ procedure defined in Lemma 38, with the catalytic tape (W, r_j, τ). The weight of s on the catalytic tape is replaced with r_j, and more importantly the place of r_j now has the first $> 7 \log n$ bits equal to 0. We increment the counter j by 1, set k to 0, and return to step 1.

Each iteration either increments k or increments j, the latter of which increases monotonically. If k reaches $|S|$ at any point, we have a maximum sized common independent set of M_1 and M_2 by definition, and step 1 will output this. Else, if j exceeds T, we have compressed each r_i for $i \in [T]$. This means that the first $7 \log n$ bits of each block r_i are all 0s. This acts as free space, which we can use to run the algorithm $\mathcal{A}$ and solve linear matroid intersection, and output the result. Once this is done, we can proceed to the next part.

Decompression. The algorithm simply needs to revert the changes it made to the catalytic tape via the $\mathcal{C}omp$ procedure. It does this by simply iteratively decrementing j, and then running the $\mathcal{D}ecomp$ procedure on M_1 and M_2 using the catalytic tape (w, r_j, τ), until j reaches 0, at which point the catalytic tape is fully restored. ◀

7 Conclusion and Open Problems

In this paper, we solve Linear Matroid Intersection in catalytic logspace by derandomizing the isolation lemma, extending the bipartite matching result of [1]. We thus present the hardest problem yet known to be in CL. A natural question is if we can solve harder problems in CL. We briefly discuss two candidates:

1. Linear matroid parity. This problem admits a deterministic polynomial time algorithm [44, 45, 46], thus a CL algorithm may be the next logical step. Just as bipartite matching is a special case of linear matroid intersection, non-bipartite matching is a special case of the linear matroid parity problem.
2. Exact linear matroid intersection. This problem admits a randomized polynomial time algorithm [49], but is not known to be in P. Thus, a CL algorithm for this would show a strong barrier towards proving $\mathsf{CL} \subseteq \mathsf{P}$. Exact matching is a special case of this problem.

Additionally, it would be interesting if resources other than catalytic space, such as nondeterminism, could lead to similar sublinear free space and polynomial time algorithms for linear matroid intersection, or even bipartite matching.

References

1 Aryan Agarwala and Ian Mertz. Bipartite matching is in catalytic logspace. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2025.

2 Manindra Agrawal, Rohit Gurjar, and Thomas Thierauf. Impossibility of derandomizing the isolation lemma for all families. *Electron. Colloquium Comput. Complex.*, TR20-098:98, 2020. URL: `https://eccc.weizmann.ac.il/report/2020/098`.

3 Martin Aigner and Thomas A. Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158:231–245, 1971. `doi:10.1090/S0002-9947-1971-0286689-5`.

4 Yaroslav Alekseev, Yuval Filmus, Ian Mertz, Alexander Smal, and Antoine Vinciguerra. Catalytic computing and register programs beyond log-depth. In Pawel Gawrychowski, Filip Mazowiecki, and Michal Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, Warsaw, Poland, August 25-29, 2025*, volume 345 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. `doi:10.4230/LIPIcs.MFCS.2025.6`.

5 Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 161–167. ACM, 1996. `doi:10.1145/237814.237856`.

6 Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC model. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, Seattle, Washington, USA, January 12-14, 2020*, volume 151 of *LIPIcs*, pages 54:1–54:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ITCS.2020.54`.

7 Vikraman Arvind and Partha Mukhopadhyay. Derandomizing the isolation lemma and lower bounds for circuit size. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, volume 5171 of *Lecture Notes in Computer Science*, pages 276–289. Springer, 2008. doi:10.1007/978-3-540-85363-3_23.

8 Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957. doi:10.1073/pnas.43.9.842.

9 Sagar Bisoyi, Krishnamoorthy Dinesh, Bhabya Rai, and Jayalal Sarma. Almost-catalytic computation. *Electron. Colloquium Comput. Complex.*, TR24-140, 2024. URL: https://eccc.weizmann.ac.il/report/2024/140.

10 Sagar Bisoyi, Krishnamoorthy Dinesh, and Jayalal Sarma. On pure space vs catalytic space. *Theor. Comput. Sci.*, 921:112–126, 2022. doi:10.1016/j.tcs.2022.04.005.

11 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):4:1–4:17, 2009. doi:10.1145/1490270.1490274.

12 Richard A Brualdi. Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society*, 1(2):161–167, 1969.

13 Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 857–866. ACM, 2014. doi:10.1145/2591796.2591874.

14 Harry Buhrman, Marten Folkertsma, Ian Mertz, Florian Speelman, Sergii Strelchuk, Sathyawageeswar Subramanian, and Quinten Tupker. Quantum catalytic space. In Bill Fefferman, editor, *20th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2025, Indian Institute of Science, Bengaluru, India, September 15-19, 2025*, volume 350 of *LIPIcs*, pages 11:1–11:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.TQC.2025.11.

15 Harry Buhrman, Michal Koucký, Bruno Loff, and Florian Speelman. Catalytic space: Nondeterminism and hierarchy. *Theory Comput. Syst.*, 62(1):116–135, 2018. doi:10.1007/s00224-017-9784-7.

16 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 458–467. ACM, 1993. doi:10.1145/167088.167213.

17 James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. The structure of catalytic space: Capturing randomness and time via compression. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 554–564. ACM, 2025. doi:10.1145/3717823.3718112.

18 James Cook and Ian Mertz. Catalytic approaches to the tree evaluation problem. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 752–760. ACM, 2020. doi:10.1145/3357713.3384316.

19 James Cook and Ian Mertz. Encodings and the tree evaluation problem. *Electron. Colloquium Comput. Complex.*, TR21-054, 2021. URL: https://eccc.weizmann.ac.il/report/2021/054.

20 James Cook and Ian Mertz. Trading time and space in catalytic branching programs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, Philadelphia, PA, USA, July 20-23, 2022*, volume 234 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.8.

21 James Cook and Ian Mertz. Tree evaluation is in space O(log n · log log n). In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1268–1278. ACM, 2024. `doi:10.1145/3618260.3649664`.

22 Stephen A. Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and branching programs for tree evaluation. *ACM Trans. Comput. Theory*, 3(2):4:1–4:43, 2012. `doi:10.1145/2077336.2077337`.

23 Samir Datta, Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Randomized and symmetric catalytic computation. In Henning Fernau, editor, *Computer Science - Theory and Applications - 15th International Computer Science Symposium in Russia, CSR 2020, Yekaterinburg, Russia, June 29 - July 3, 2020, Proceedings*, volume 12159 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2020. `doi:10.1007/978-3-030-50026-9_15`.

24 Jack Edmonds. Matroid intersection. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization I*, volume 4 of *Annals of Discrete Mathematics*, pages 39–49. North-Holland, Amsterdam, 1979. `doi:10.1016/S0167-5060(08)70817-3`.

25 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, France, March 5-9, 2001, Revised Papers*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26, New York, 2001. Springer. `doi:10.1007/3-540-36478-1_2`.

26 Jack Edmonds. Matroid partition. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 199–217. Springer, 2010. `doi:10.1007/978-3-540-68279-0_7`.

27 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 754–763. ACM, 2016. `doi:10.1145/2897518.2897564`.

28 Marten Folkertsma, Ian Mertz, Florian Speelman, and Quinten Tupker. Fully characterizing lossy catalytic computation. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, Columbia University, New York, NY, USA, January 7-10, 2025*, volume 325 of *LIPIcs*, pages 50:1–50:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. `doi:10.4230/LIPIcs.ITCS.2025.50`.

29 Satoru Fujishige. A primal approach to the independent assignment problem. *Journal of the Operations Research Society of Japan*, 20(1):1–15, 1977.

30 Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Unambiguous catalytic computation. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, Bombay, India, December 11-13, 2019*, volume 150 of *LIPIcs*, pages 16:1–16:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.FSTTCS.2019.16`.

31 Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. Lossy catalytic computation. *CoRR*, abs/2408.14670, 2024. `doi:10.48550/arXiv.2408.14670`.

32 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 821–830. ACM, 2017. `doi:10.1145/3055399.3055440`.

33 Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. Isolating a vertex via lattices: Polytopes with totally unimodular faces. *SIAM J. Comput.*, 50(2):636–661, 2021. `doi:10.1137/19M1290802`.

34 Vivek Anand T. Kallampally and Raghunath Tewari. Trading determinism for time in space bounded computations. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, Kraków, Poland, August 22-26, 2016*, volume 58 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.MFCS.2016.10.

35 Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223. ACM, 2001. doi:10.1145/380752.380801.

36 Michal Koucký. Catalytic computation. *Bull. EATCS*, 118, 2016. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/400.

37 Michal Koucký, Ian Mertz, Edward Pyne, and Sasha Sami. Collapsing catalytic classes. *Electron. Colloquium Comput. Complex.*, TR25-019, 2025. URL: https://eccc.weizmann.ac.il/report/2025/019.

38 Stein Krogdahl. A combinatorial base for some optimal matroid intersection algorithms. Technical report, Computer Science Department, Stanford University, Stanford, California, 1974.

39 Stein Krogdahl. A combinatorial proof for a weighted matroid intersection algorithm. *https://submission.dagstuhl.de/images/dblp-gray.pngComputer Science Report*, 17, 1976.

40 Stein Krogdahl. The dependence graph for bases in matroids. *Discret. Math.*, 19(1):47–59, 1977. doi:10.1016/0012-365X(77)90118-2.

41 Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

42 Eugene L. Lawler. Matroid intersection algorithms. *Math. Program.*, 9(1):31–56, 1975. doi:10.1007/BF01681329.

43 Andrzej Lingas and Mia Persson. A fast parallel algorithm for minimum-cost small integral flows. *Algorithmica*, 72(2):607–619, 2015. doi:10.1007/s00453-013-9865-1.

44 László Lovász. The matroid matching problem. *Algebraic methods in graph theory*, 2:495–517, 1978.

45 László Lovász. On determinants, matchings, and random algorithms. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, volume 79, pages 565–574. Akademie-Verlag, Berlin, 1979.

46 László Lovász. Matroid matching and some applications. *J. Comb. Theory B*, 28(2):208–236, 1980. doi:10.1016/0095-8956(80)90066-0.

47 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chic. J. Theor. Comput. Sci.*, 1997, 1997. URL: http://cjtcs.cs.uchicago.edu/articles/1997/5/contents.html.

48 Ian Mertz. Reusing space: Techniques and open problems. *Bull. EATCS*, 141:57–106, 2023. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/780.

49 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 345–354. ACM, 1987. doi:10.1145/28395.383347.

50 H. Narayanan, Huzur Saran, and Vijay V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arboresences and edge-disjoint spanning trees. *SIAM J. Comput.*, 23(2):387–397, 1994. doi:10.1137/S0097539791195245.

51 James B. Orlin and Clifford Stein. Parallel algorithms for the assignment and minimum-cost flow problems. *Oper. Res. Lett.*, 14(4):181–186, 1993. doi:10.1016/0167-6377(93)90068-R.

52 Aaron Potechin. A note on amortized branching program complexity. In Ryan O'Donnell and Ryan O'Donnell, editors, *32nd Computational Complexity Conference, CCC 2017, Riga, Latvia, July 6-9, 2017*, volume 79 of *LIPIcs*, pages 4:1–4:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.4.

53 Edward Pyne, Nathan S. Sheffield, and William Wang. Catalytic communication. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, Columbia University, New York, NY, USA, January 7-10, 2025*, volume 325 of *LIPIcs*, pages 79:1–79:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ITCS.2025.79.

54 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.

55 Robert Robere and Jeroen Zuiddam. Amortized circuit complexity, formal complexity measures, and catalytic algorithms. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 759–769. IEEE, 2021. doi:10.1109/FOCS52979.2021.00079.

56 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer Berlin Heidelberg, 2003.

57 Dieter van Melkebeek and Gautam Prakriya. Derandomizing isolation in space-bounded settings. *SIAM J. Comput.*, 48(3):979–1021, 2019. doi:10.1137/17M1130538.

58 Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, July 1935. doi:10.2307/2371182.

59 R. Ryan Williams. Simulating time with square-root space. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 13–23. ACM, 2025. doi:10.1145/3717823.3718225.