Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 80 (2026)

Maximum Matching and Related Problems in Catalytic Logspace

Srijan Chakraborty, Samir Datta, Aryan Kusre, Partha Mukhopadhyay, Amit Sinhababu
Chennai Mathematical Institute, India
{srijanc, sdatta, aryank, partham, amitks}@cmi.ac.in

May 31, 2026

Abstract

Understanding the power of space-bounded computation with access to catalytic space has been an important theme in complexity theory over the recent years. One of the key algorithmic results in this area is that bipartite maximum matching can be computed in catalytic logspace (CL) with a polynomial-time bound (CLP) [AM25].

In this paper, we show that we can construct a *maximum matching* in *general graphs* in CL, and, in fact, in CLP. We first show that the size of a *maximum matching* in *general graphs* can be determined in CL. Our algorithm is based on the linear-algebraic algorithm for maximum matching by Geelen [Gee00]. We then show that this algorithm, along with some new ideas, can be used to *find* a maximum matching in general graphs. Using a similar algorithm of Geelen [Gee99], we also solve the *maximum rank completion problem* in CLP, which was previously known to be solvable in deterministic polynomial time [Gee99]. This problem turns out to be equivalent to the *linear matroid intersection* problem [Mur95] which has been shown to be in CLP by [AAV26]. Finally, using a PTAS algorithm [BJP18] for approximating the rank in Edmond's problem, we derive a CLP algorithm that can approximate the rank given by any instance of the *Edmond's problem* upto a factor of $(1-\varepsilon)$ for any $\varepsilon \in (0,1)$. An application of this is a CLP bound for approximating the maximum independent matching size in the *linear matroid matching* problem.

1 Introduction

The study of space-bounded computation is a central theme in complexity theory, with a long line of work investigating the power and limitations of logarithmic-space algorithms. A recent direction in this area focuses on models that augment classical space bounds with auxiliary resources, such as catalytic space, where a large workspace is available but must be returned to its initial state at the end of the computation. This model, introduced by Buhrman, Cleve, Koucký, Loff, and Speelman[BCK+14], has led to surprising algorithmic developments and a refined understanding of the role of reversibility and space reuse in computation.

ISSN 1433-8092

A major goal in this line of research is to identify natural problems that can be solved efficiently within catalytic logspace (CL), particularly with polynomial-time bounds (CLP). While several foundational results have established the basic properties of this model, progress on natural combinatorial problems has been more recent. Notably, the breakthrough result of [AM25] shows that bipartite maximum matching can be computed in catalytic logspace with polynomial time, providing one of the first nontrivial examples of a classical graph problem lying in CLP. Over the recent years, there have been many results surrounding catalytic computation such as [BCK+14, Pyn24, CLMP25, CGM+25, AAV26, ACD26, Ede26, CDK+26]. A flowchart for various complexity classes would be: $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{TC}^1 \subseteq \mathsf{NC} \subseteq \mathsf{P} \subseteq \mathsf{ZPP}$ where the only known bounds for catalytic logspace are $\mathsf{TC}^1 \subseteq \mathsf{CL} \subseteq \mathsf{ZPP}$.

Matching problems occupy a central position in combinatorial optimization and theoretical computer science. The maximum matching problem, in both bipartite and general graphs, has been extensively studied, with classical polynomial-time algorithms based on augmenting paths, combinatorial structures such as blossoms, and algebraic formulations [Tut47, Edm65, HK73, Lov79]. In particular, algebraic approaches based on the Tutte matrix and rank computations, such as those developed by Jim Geelen, provide a powerful framework for reasoning about matchings and their generalizations [Gee99, Gee00].

In this paper, we extend the reach of catalytic logspace algorithms for matching problems. We show that a maximum matching in general graphs can be computed in CLP, thereby generalizing the bipartite case. To show that bipartite matching is in CLP, [AM25] uses isolating weights which they sample from the catalytic tape. They can solve the problem in one shot if a given weight assignment is isolating, and otherwise they find a threshold edge by finding appropriate augmenting paths and can free up space. This technique breaks down in general graphs, since finding augmenting paths becomes much more difficult, as one might come across blossoms while doing so. To avoid these issues, we build on the algebraic techniques of Geelen [Gee00] and adapt them to the catalytic setting, carefully managing space reuse. To search for a maximum matching, we again use isolating weights, but not in the conventional sense, in particular, we do not isolate a min-weight matching. As a further application, we show that the *maximum rank matrix completion* problem can also be solved in CLP [Gee99]. This problem, previously known to admit deterministic polynomial-time algorithms, captures a common generalization of linear matroid intersection and bipartite matching [AAV26, AM25], and thus serves as a unifying framework for these problems.

Both maximum matching, and the matrix completion problems are special cases of *symbolic determinant identity testing* (SDIT), which is also known as *Edmonds' problem* [Edm65, Lov79]. Given a matrix $A = \sum_{i \in [m]} A_i x_i$, the SDIT problem asks for the rank of A. There is a simple randomized polynomial-time algorithm for this problem: one can obtain a matrix of maximum rank by substituting random values for $x_1, x_2, \ldots, x_m$ from a sufficiently large set from the field $\mathbb{F}$, using the Polynomial Identity Testing lemma [Zip79, Sch80, DL78].

In [BJP18], Bläser, Jindal, and Pandey study this problem of computing the rank of the symbolic matrix A, and are able to get a PTAS for the same. However, exact computation of the commutative rank of a given matrix space in deterministic polynomial time is the main problem of symbolic identity testing (SDIT) which is directly related to lower bounds [KI04].

Extending the ideas in [BJP18], we derive a CLP algorithm that can approximate the maximum rank of a matrix in any matrix space upto a factor of $(1-\varepsilon)$ for any $\varepsilon \in (0,1)$. An interesting application of this is a CLP algorithm for approximating the size of a maximum independent matching in the linear matroid matching problem.

Our results provide new evidence for the power of catalytic space in enabling efficient algorithms for algebraic and combinatorial problems. More broadly, they suggest that linear-algebraic techniques may play a fundamental role in understanding the capabilities of space-bounded computation with reversible auxiliary memory. This also gives a CL bound for a couple of problems, that we do not yet know how to solve in NC, namely maximum matching size in general graphs, matrix rank completion, $(1-\varepsilon)$ approximating Edmond's rank. Moreover CLP also turns out to be the first well studied subclass of P where we can find a maximum matching in general graphs.

Our results

The maximum matching problem in general graphs admits a well-known algebraic formulation via the Tutte matrix [Tut47]. Given a graph G, the Tutte matrix is a symbolic skew-symmetric matrix in which each edge (i,j) is associated with an indeterminate $x_{i,j}$ (with a sign), and non-edges correspond to zero entries.

The fundamental connection shows that the rank of this matrix (over a suitable field) is twice the size of a maximum matching in G. This characterization enables the use of linear-algebraic techniques for computing matchings, including randomized algorithms based on substituting indeterminates with field elements. In particular, combining this approach with fast parallel algorithms for matrix rank yields an RNC upper bound for maximum matching in general graphs [MVV87], placing the problem among the central examples of efficiently parallelizable combinatorial problems. From the perspective of derandomization and pinning down the parallel complexity, putting matching in NC has been an outstanding question.

The last decade saw advances in understanding the parallel complexity of matching. In particular, building on the breakthrough quasi-NC algorithm for bipartite matching due to Fenner, Gurjar, and Thierauf [FGT16], follow-up work by Svensson and Tarnawski [ST17] obtained quasi-NC algorithms for maximum matching in general graphs.

Despite these advances, a CL algorithm to find a perfect matching in general graphs has remained elusive. We settle this question by proving the following theorem.

Theorem 1.1. *Maximum matchings in a general graph G can be constructed in* CLP.

The maximum rank matrix completion problem asks, given a matrix with some entries fixed and others unspecified, to assign values to the unspecified entries so as to maximize the rank of the resulting matrix. This problem admits a natural algebraic interpretation and captures several combinatorial optimization problems as special cases. In particular, Geelen showed that the problem can be solved in deterministic polynomial time via a greedy algorithm that incrementally assigns values while preserving rank growth [Gee99]. The problem generalizes classical questions such as bipartite matching and linear matroid intersection, and

plays a central role in algebraic approaches to matching and related problems. Extending the techniques used to prove Theorem 1.1, we show the following result.

Theorem 1.2. *The maximum rank completion problem can be solved in* CLP.

In [BJP18], Bläser, Jindal, and Pandey study the problem of computing the commutative rank of a matrix space, a fundamental quantity that captures the maximum rank achievable by linear combinations of given matrices $A_1, A_2, \ldots, A_n$. More precisely, the rank of a matrix space $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ over a sufficiently large field $\mathbb{F}$ is the maximum rank of a matrix in $\mathcal{A}$. If we consider the formal linear combination $A = \sum_{i=1}^{n} A_i x_i$, then we can compute a matrix of maximum rank by substituting random values to $x_1, x_2, \ldots, x_n$ from $\mathbb{F}$ by applying the following Polynomial Identity Testing Lemma [Zip79, Sch80, DL78]:

Lemma 1.3. *Let $P \in R[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$ over an integral domain R. Let S be a finite subset of R and let $r_1, r_2, \ldots, r_n$ be selected at random independently and uniformly from S. Then* $\Pr[P(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}$

As already mentioned, exact computation of the commutative rank of a given matrix space in deterministic polynomial time is hard [KI04]. Nevertheless, Bläser, Jindal, and Pandey [BJP18] could show a deterministic *polynomial-time approximation scheme* (PTAS) that computes a $(1-\varepsilon)$-approximation of the rank for any fixed $\varepsilon > 0$. Their approach combines algebraic and combinatorial techniques to bypass randomness. Our main result in this context is the following.

Theorem 1.4. *Given a matrix space $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$ by a set of matrices $A_1, A_2, \ldots, A_n$ and $\varepsilon > 0$, we can compute a matrix $A \in \mathcal{A}$ whose rank is at least $1-\varepsilon$ times the rank of $\mathcal{A}$, in* CLP.

Since our main result uses ideas from Geelen's algebraic matching algorithm [Gee00], we provide a brief overview of his result. Let T be the Tutte matrix of $G = (V, E)$ where $|V| = n$. For any evaluation of T', let $\operatorname{rank}(T')$ denote its rank. Let $D(T')$ denote the corresponding set of deficiency, i.e., the set of vertices x such that $\operatorname{rank}(T' \setminus \{x\}) = \operatorname{rank}(T')$. Intuitively, the set of vertices corresponds to the Gallai–Edmonds decomposition induced by T' in a linear algebraic sense. Then, Geelen [Gee00] defines a preorder $T'_1 \preceq T'_2$ either

$$\operatorname{rank}(T'_1) < \operatorname{rank}(T'_2) \quad \text{or} \quad (\operatorname{rank}(T'_2) = \operatorname{rank}(T'_1) \quad \text{and} \quad D(T'_1) \subseteq D(T'_2)).$$

If both $\operatorname{rank}(T'_1) = \operatorname{rank}(T'_2)$ and $D(T'_1) = D(T'_2)$ then $T'_1 = T'_2$. The algorithm incrementally assigns values to the indeterminates corresponding to the edges and constructs a sequence $T'_1 \prec T'_2 \prec \cdots \prec T'_m$ where the final matrix achieves the rank of the Tutte matrix.

More precisely, to go from T'_i to T'_{i+1}, the algorithm selects an edge (i, j) and replaces its value by $a \in \{1, 2, \ldots, n\}$ such that $T'_i \prec T'_{i+1}$. If such an assignment does not exist then the algorithm already has found the evaluated matrix achieving the true rank. After finding the size of the maximum matching, they use self-reducibility of matching to search for one. But under our space bounded constraints, the search for a maximum matching in the same way as above, does not work out. Notice that [ACD26] shows a CL reduction from search to weighted decision, that works in the matching context as well, but it is also not

clear how to solve the weighted decision using our ideas. To construct a maximum matching of the graph, we need some substantially new ideas. We again sample weight assignments from the catalytic tape, as [AM25], but we are not able to find min-isolating weights. In stead, we have a non-zero polynomial, the least degree of which turns out to be the weight of the matching that we attempt to isolate. Smaller weighted matchings may disappear in this polynomial. Finally, if we are not able to find a matching even in this way, we show that we are able to save enough space to construct a matching in polynomial space and time, and later restore the catalytic tape. We heavily use the *'compress or compute'* paradigm.

2 Preliminaries

Matchings Let $G = (V, E)$ be a simple undirected graph. A *matching* $M \subseteq E$ is a set of pairwise vertex-disjoint edges. A matching is said to be *maximum* if it has the largest possible cardinality among all matchings in G.

We denote by $\nu(G)$ the size of a maximum matching in G, i.e.,

$$\nu(G) = \max\{|M| : M \subseteq E \text{ is a matching}\}.$$

Gallai–Edmonds Decomposition Let $G = (V, E)$ be a simple graph. Define:

1. $D(G)$: the set of vertices that are unmatched in at least one maximum matching of G,
2. $A(G)$: the set of vertices in $V \setminus D(G)$ that have a neighbor in $D(G)$,
3. $C(G) := V \setminus (A(G) \cup D(G))$.

The partition $(D(G), A(G), C(G))$ is called the *Gallai–Edmonds decomposition* of G.

This decomposition satisfies the following properties:

1. Each connected component of $G[D(G)]$ is factor-critical.
2. $G[C(G)]$ has a perfect matching.
3. In every maximum matching:
   (a) all vertices of $A(G)$ are matched to vertices in $D(G)$,
   (b) each component of $G[D(G)]$ has exactly one unmatched vertex.

Linear Matroids Let $\mathbb{F}$ be a field. A *linear matroid* is a matroid $M = (E, \mathcal{I})$ where E is a finite set and there exists a matrix $A \in \mathbb{F}^{r \times |E|}$ such that a subset $I \subseteq E$ is independent (i.e., $I \in \mathcal{I}$) if and only if the corresponding set of columns of A is linearly independent over $\mathbb{F}$.

Linear Matroid Intersection Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two linear matroids represented over the same ground set E. The *linear matroid intersection problem* asks to find a set

$$I \subseteq E \quad \text{such that } I \in \mathcal{I}_1 \cap \mathcal{I}_2$$

of maximum cardinality.

Linear Matroid Parity Let $M = (E, \mathcal{I})$ be a linear matroid, and suppose the ground set E is partitioned into disjoint pairs:

$$E = \{e_1, f_1, e_2, f_2, \ldots, e_m, f_m\}.$$

The *linear matroid parity problem* asks to find a maximum-size collection of pairs such that the union of the selected elements is independent in M. Formally, find an index set $S \subseteq [m]$ maximizing $|S|$ such that

$$\{e_i, f_i : i \in S\} \in \mathcal{I}.$$

Linear Matroid Matching Let $M = (E, \mathcal{I})$ be a linear matroid, and let $G = (E, F)$ be a graph whose vertex set is the ground set of the matroid.

A set of edges $S \subseteq F$ is a *matroid matching* if:

1. S is a matching in the graph G, and
2. the set of vertices covered by S is independent in the matroid M.

The *linear matroid matching problem* asks to find a matroid matching of maximum cardinality.

Determinant For a square matrix $A \in \mathbb{F}^{n \times n}$, the determinant is defined as:

$$\mathsf{Det}(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$$

where S_n is the symmetric group and $\operatorname{sgn}(\sigma)$ is the parity of the permutation.

Pfaffian For a skew-symmetric matrix $A \in \mathbb{F}^{2n \times 2n}$ where $A^T = -A$, the Pfaffian is defined as:

$$\mathsf{Pf}(A) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{\sigma(2i-1),\sigma(2i)}$$

It holds that $\mathsf{Det}(A) = \mathsf{Pf}(A)^2$.

Catalytic Computation

Definition 2.1. *A catalytic Turing machine M with space $s(n)$ and catalytic space $c(n)$ is a Turing machine that has a read-only input tape of length n, write-only output tape, and $s(n)$ space bounded read-write work tape, and a catalytic tape of size $c(n)$. We say that M computes a function f if for every $x \in \{0,1\}^n$ and $\tau \in \{0,1\}^{c(n)}$, the result of executing M on input x with initial catalytic tape τ i.e. $M(x, \tau)$ satisfies:*

1. *M halts with $f(x)$ on the output tape.*
2. *M halts with the catalytic tape consisting of τ.*

$\mathsf{CSPACE}[s(n), c(n)]$ is the family of functions computable by such a Turing machine. Similarly, $\mathsf{CTISP}[t(n), s(n), c(n)]$ is the family of functions computable by such a Turing machine with the further restriction that the machine simultaneously runs in time $t(n)$.

Definition 2.2 (Catalytic Logspace)**.** *We define catalytic logspace as*

$$\mathsf{CL} = \cup_{k\in\mathbb{N}}\mathsf{CSPACE}[k\log n, n^k]$$

Moreover, CLP *is defined as the set of functions computable by a* CL *machine that simultaneously runs in polynomial time.*

In particular, we know that $\mathsf{CLP} = \mathsf{CL} \cap \mathsf{P}$ [CLMP25]. To compute the rank of various matrices, we shall use the fact that $\mathsf{TC}^1 \subseteq \mathsf{CL}$ [BCK+14]. It is well known that Determinants and Pfaffians are computable in TC^1 and thus in CL.

3 Maximum Matching in CLP

In this section, we prove the decision version of Theorem 1.1.

3.1 A linear algebraic formulation

Fact 3.1. *Let* A *be any skew symmetric matrix. Then*

1. $\operatorname{rank} A$ *is even.*
2. *The rank of* A *is equal to the maximum size of a non-singular principal minor/submatrix of* A*.*

Let $G = (V, E)$ be a simple graph, $|V| = n, |E| = m$, $x_1, \ldots x_m$ be edge variables. We now state the main result in [Gee00] tailored made to our application. The Tutte matrix T of G is the $n \times n$ matrix given by:

$$T_{uv} = \begin{cases} x_i & u < v,\ i = \{u, v\} \in E, \\ -x_i & u > v,\ i = \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 3.2 (Geelen, [Gee00])**.** *Let* T *be the Tutte matrix of a simple graph* $G = (V, E)$*, and let* T' *be an evaluation of* T*. Then either* $\operatorname{rank} T' = \operatorname{rank} T$ *and* $D(T') = D(T) = D(G)$*, or one of the following two statements hold:*

- $\exists i \in E, a \in [n^{10}]$ *such that* $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T' + 2$*.*
- $\exists i \in E, a \in [n^{10}]$ *such that* $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T'$ *and* $D(T') \subsetneq D(T'_{x_i \leftarrow a})$*.*

Recall that $T'_{x_i \leftarrow a}$ is the matrix obtained from T' by changing the value corresponding to the edge variable x_i by a. Moreover, as already mentioned in Section 1 that $D(T)$ is the set of vertices which are deficient in terms of rank. Similarly $D(G)$ is the set of vertices in G which are missed by some maximum matching.

Remark 3.3. *In [BCK+14] it is shown that* $\mathsf{TC}^1 \subseteq \mathsf{CL}$*. This already shows the determinant and matrix rank computation can be performed in* CL*. Clearly, the membership question for the set* D *logspace reduces to the rank computation which is in* CL*.*

Given an evaluation T' of the Tutte matrix T, the following algorithm decides whether the rank of T' is the highest possible rank (or the rank of T).

Algorithm 1 IsMaxRank$(T', T, G = (V, E))$

```
k ← rank T'                                   ▷ Can be done in CL
maxrank ← true
for i ∈ E, a ∈ [n^10] do
    T'' = T'_{x_i ← a}
    if (k < rank T'') ∨ (k = rank T'' ∧ D(T') ⊊ D(T'')) then ▷ Can be checked in CL
        maxrank ← false
    end if
end for
return maxrank
```

Lemma 3.4. *Suppose, we are given an evaluation of T, namely T'. Then we can check in* CL *if* $\operatorname{rank} T = \operatorname{rank} T'$ *or not.*

Proof. Algorithm 1 gives the required procedure, the proof of which directly follows from Theorem 3.2 and Remark 3.3. □

Next we show two uniqueness lemmas, which will turn out to be essential ingredients in the design of our catalytic algorithms.

Lemma 3.5 $(2A)$**.** *Let T' be an evaluation of T and $i \in E, a \in [n^{10}]$ such that* $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T' + 2$*, and in T', x_i is a_i. Then $\forall \tilde{a} \neq a_i$, we have* $\operatorname{rank} T'_{x_i \leftarrow \tilde{a}} = \operatorname{rank} T' + 2$.

Proof. Let $k = \operatorname{rank} T'$. Let $X \subseteq V, |X| = k+2$ be a maximal principal minor of T that is non-singular in $T'_{x_i \leftarrow a}[X]$. Notice that $L(y) = \mathsf{Pf}(T'_{x_i \leftarrow y}[X])$ is linear in y. Since $L(a) \neq 0, L(a_i) = 0$, clearly $\forall \tilde{a} \neq a_i, L(\tilde{a}) \neq 0$ i.e $T'_{x_i \leftarrow \tilde{a}}[X]$ is non-singular. Furthermore, on changing two entries of T' (i.e. by changing the variable x_i) the rank can go up by atmost 2. Therefore, $\operatorname{rank} T'_{x_i \leftarrow \tilde{a}} = \operatorname{rank} T' + 2 \ \forall \tilde{a} \neq a$. □

Lemma 3.6 $(2B)$**.** *Let T' be an evaluation of T and $i \in E, a \in [n^{10}]$ such that* $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T' = k$ *where in T', x_i is set to a_i. Suppose $\exists u \in D(T'_{x_i \leftarrow a}) \setminus D(T')$. Then $\forall \tilde{a} \neq a_i$, we have $u \in D(T'_{x_i \leftarrow \tilde{a}})$.*

Proof. From the given assumptions, we have a $X_u \subseteq V, |X_u| = k$ such that $T'_{x_i \leftarrow a}[X_u]$ is non-singular and $u \notin X_u$. Since $u \notin D(T')$, $T'[X_u]$ is singular.

Claim 3.7. $\forall \tilde{a} \neq a_i$, $\operatorname{rank} T'_{x_i \leftarrow \tilde{a}} \leq k$.

Proof. Suppose $\operatorname{rank} T'_{x_i \leftarrow \tilde{a}} > k$ for some $\tilde{a}$. Therefore, $\tilde{a} \notin \{a, a_i\}$. Let X be the principal minor of size larger than k such that $T'_{x_i \leftarrow \tilde{a}}[X]$ is non-singular. But then $\widetilde{L}(y) = \mathsf{Pf}(T'_{x_i \leftarrow y}[X])$ is linear and $\widetilde{L}(a) = \widetilde{L}(a_i) = 0$. Therefore, $\widetilde{L}(\tilde{a}) = 0$ gives a contradiction. □

Again $L(y) = \mathsf{Pf}(T'_{x_i \leftarrow y}[X_u])$ is linear in y, such that $L(a) \neq 0, L(a_i) = 0$. Therefore, $\forall \tilde{a} \neq a_i$ we have $L(\tilde{a}) \neq 0$ i.e $T'_{x_i \leftarrow \tilde{a}}[X_u]$ is non-singular and by the above claim X_u is of maximum possible size (i.e. $\operatorname{rank} T'_{x_i \leftarrow \tilde{a}} = |X_u|$), thus $u \in D(T'_{x_i \leftarrow \tilde{a}})$. □

3.2 A compress or compute algorithm

Now we show that the size of a maximum matching in a general graph can be computed in CL. Let G be a graph on n vertices and m edges, and let T be its Tutte matrix with variables $x_1, \ldots, x_m$ corresponding to the edges.

We sample an assignment $a_1, \ldots, a_m$ for these variables from the catalytic tape. Either this assignment maximizes the rank of the evaluated matrix T', or it does not. We can verify this using Algorithm 1. If the rank is indeed maximized, we proceed to restore the modified catalytic tape.

Otherwise, if the rank is not maximized, we consider two cases:

(a) Suppose that by perturbing one of the variables x_i, the rank of T' can be increased. In this case, by Lemma 3.5, for all values other than a_i, the rank increases. Therefore, we discard a_i from the catalytic tape and instead store the index i along with the rank. Later, we can recover a_i as the unique value for which the rank remains unchanged.

(b) Otherwise, perturbing some variable x_i does not increase the rank, but enlarges the set $D(T')$, i.e., there exists a vertex u that gets added to $D(T')$. By Lemma 3.6, a_i is the unique value for which $u \notin D(T')$. Thus, we again discard a_i and store its index i together with u. As before, a_i can be recovered when needed using this characterization.

Finally, if the sampled assignment already yields the maximum rank, we are done. Otherwise, we have freed enough space to recompute the rank from scratch using the algorithm of [Gee00]. A formal proof follows.

Theorem 3.8. *Given a simple graph $G = (V, E)$ on n vertices and m edges, we can in* CLP *find the size of the maximum matching, $\nu(G)$ of G.*

Proof. Let $x_1, \ldots, x_m$ be edge variables. Consider the Tutte matrix T, given by

$$T_{uv} = \begin{cases} x_i & u < v,\ i = \{u, v\} \in E, \\ -x_i & u > v,\ i = \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Assume that the catalytic tape is divided into blocks $(\mathcal{C}_1, \ldots, \mathcal{C}_N, \mathcal{B})$ where $\mathcal{C}_r = (a_1, \ldots, a_m) \forall r \in [N]$ with each $a_i \in [n^{10}]$ is $10 \log n$ bits long, and $\mathcal{B}$ is an extra block which shall be used for rank computations; N is n^3. The algorithm proceeds as follows: We iterate over all $\mathcal{C}_1, \ldots, \mathcal{C}_N$, suppose we are processing $\mathcal{C}_r = (a_1, \ldots, a_m)$, then we follow the following steps (as described in Algorithm 2):

1. Let T' be the evaluation of T given by $\mathcal{C}_r$. We execute Algorithm 2 which either reports $\nu(G)$ or not. If we get $\nu(G)$ from this call, then we proceed to call Algorithm 3 on all previous catalytic blocks, i.e. jump to Step 4 with the temporary variable $index \leftarrow r - 1$. Now suppose that Algorithm 2 does not return $\nu(G)$. Therefore, from Theorem 3.2, the only two possibilities are:

   (*a*) $\exists i \in E, a \in [n^{10}]$ such that $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T' + 2$.

   (*b*) $\exists i \in E, a \in [n^{10}]$ such that $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T' \wedge D(T') \subsetneq D(T'_{x_i \leftarrow a})$.

   Consider the two cases $2A, 2B$ that deals with (a) and (b) respectively, as follows–

Algorithm 2 ProcessAssignment$(\mathcal{C}, T, G = (V, E))$

Input: We are given $\mathcal{C} = (a_1, \ldots, a_m)$ as an initial catalytic string, where $a_i \in [n^{10}] \forall i \in E$.
$T' \leftarrow T_{\forall i \in E : x_i \leftarrow a_i}$, $k \leftarrow \operatorname{rank} T'$
if IsMaxRank(T', T, G) **then** ▷ Call Algorithm 1
    return $\nu(G) = \dfrac{k}{2}, D(G) = D(T')$
end if
$case \leftarrow 2$
for $i \in E, a \in [n^{10}]$ **do**
    $T'' \leftarrow T'_{x_i \leftarrow a}$
    if $\operatorname{rank} T'' = k + 2$ **then**
        $case \leftarrow 2A$
        Reset $\mathcal{C} \leftarrow (2A, i, k, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m)$.
        return ▷ Terminate the algorithm
    end if
end for
for $i \in E, a \in [n^{10}]$ **do**
    $T'' \leftarrow T'_{x_i \leftarrow a}$
    if $(\operatorname{rank} T'' = k) \wedge (D(T') \subsetneq D(T''))$ **then**
        $case \leftarrow 2B$
        Let $u \in D(T'') \setminus D(T')$ ▷ Pick any such u arbitrarily
        Reset $\mathcal{C} \leftarrow (2B, i, u, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m)$.
        return ▷ Terminate the algorithm
    end if
end for

Algorithm 3 RestoreAssignment$(\mathcal{C}, T, G = (V, E))$

Input: We are given $\mathcal{C} = (case, j, \beta, a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_m), a_i \in [n^{10}] \forall i \in E$.
$T' \leftarrow T_{\forall i \in E \setminus \{j\} : x_i \leftarrow a_i}$
if $case = 2A$ **then**
    $k \leftarrow \beta$
    for $a \in [n^{10}]$ **do**
        if $k = \operatorname{rank} T'_{x_j \leftarrow a}$ **then**
            Reset $\mathcal{C} \leftarrow (a_1, \ldots, a_{j-1}, a_j \leftarrow a, a_{j+1}, \ldots, a_m)$
        end if
    end for
else
    $u \leftarrow \beta$
    for $a \in [n^{10}]$ **do**
        if $u \notin D(T'_{x_j \leftarrow a})$ **then**
            Reset $\mathcal{C} \leftarrow (a_1, \ldots, a_{j-1}, a_j \leftarrow a, a_{j+1}, \ldots, a_m)$
        end if
    end for
end if

2A. In this case, after finding an i as in (a), from Lemma 3.5, we know that for no other substitution $x_i \leftarrow a$ and $a \neq a_i$, is $\operatorname{rank} T'_{x_i \leftarrow a} = \operatorname{rank} T'$. Therefore, we replace $\mathcal{C}_r$ with $(2A, i, \operatorname{rank} T', a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m)$. This saves $10m \log n - \mathcal{O}(1) - \log m - \log n - 10(m-1)\log n > 6 \log n$ (here $\mathcal{O}(1)$ comes from writing $2A$).

2B. Again, first we find an index i, and value a witnessing case (b). Now, we pick a $u \in D(T'_{x_i \leftarrow a}) \setminus D(T')$. Again from Lemma 3.6 we know that a_i is the unique value for which $D(T')$ does not contain u. Hence we replace $\mathcal{C}_r$ with $(2B, i, u, a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_m)$. This again saves $10m \log n - \mathcal{O}(1) - \log m - \log n - 10(m-1)\log n > 6 \log n$ bits.

3. Suppose we just processed $\mathcal{C}_N$ and still did not find $\nu(G)$. In this case, we have freed up at least $6N \log n = \mathcal{O}(n^3 \log n)$ space for $N = n^3$. We shift the catalytic strings to make this space contiguous, and run any standard polynomial time algorithm for matching and find $\nu(G)$. Next set $index = N$ and go to step 4.

4. We have either reached this step if we have processed all catalytic blocks, then $index = N$, and otherwise we have jumped from step 1 with some value of $index$. In either case, we now recompute the modified catalytic strings $\mathcal{C}_1, \ldots, \mathcal{C}_{index}$. For each $r \in [index]$, we execute Algorithm 3 on $\mathcal{C}_r = (case, j, \beta, a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_m)$. Here is how it works:

   (a) This is when $case = 2A$. Therefore, we have $k = \beta$ to be the rank of T' with the evaluation of T given by the original catalytic tape (before we made any modified any catalytic bits). Therefore, we find the unique a such that $\operatorname{rank} T'_{x_j \leftarrow a} = k$, and restore the catalytic tape by setting $a_j \leftarrow a$. The correctness is again given by Lemma 3.5.

   (b) This time $case = 2B$. Therefore $u = \beta$. Again find the unique a such that $u \notin D(T'_{x_j \leftarrow a})$ and restore $a_j \leftarrow a$. The correctness follows from Lemma 3.6.

The above algorithm clearly runs in polynomial time and the catalytic space used is $\mathcal{O}(mn^2 \log n + |\mathcal{B}|) = \mathsf{poly}(n)$ with work space $\mathcal{O}(\log n)$, and the catalytic tape is always restored when the algorithm halts. Thus the above gives the required CLP algorithm to compute $\nu(G)$. □

Corollary 3.8.1. *Given a simple graph $G = (V, E)$, we can in* CLP *compute the Gallai-Edmonds Decomposition of G.*

Proof. Notice that in the above algorithm, if we succeed in finding a 'good' catalytic tape (in Algorithm 2), then we have also found $D(T)$ which is equal to $D(G)$. In case we do not find a 'good' catalytic tape, we have saved enough free space to compute $D(G)$ from scratch using [Gee00]. After computing $D(G)$ it is easy to get $A(G)$ and $C(G)$ as $A = (V \setminus D) \cap N_G[D], C = V \setminus (A \cup D)$ where $N_G[D]$ is the set of neighbors of $D(G)$ in G. □

Corollary 3.8.2. *Given a simple graph G and its Tutte matrix T, in* CLP *we can find an assignment T' of T that maximizes the rank of T.*

Proof. In the above algorithm, if we find a good catalytic tape via Algorithm 2, then the catalytic bits themselves give an evaluation T' that is of the highest possible rank. In case we have not found any 'good' catalytic tapes, we have freed up enough space to run Geelen's algorithm [Gee00] directly. □

Remark 3.9. *Notice that from Corollary 3.8.2 we get a* CL *algorithm that finds an evaluation T' of T in the following sense:*

- *Either the program saves enough space, and finds T', and can later restore the catalytic tape whenever necessary.*
- *Or, it finds the evaluation T' from a given string (read as assignments to the variables) on the catalytic tape.*

This shall be used implicitly when we prove Lemma 4.2 and theorem 4.3.

4 Hunt for a Maximum Matching

Finally we show how to search for a maximum matching in CL. We break it down into two steps, the second step being a standard reduction, but the first step is fairly technical and uses new ideas:

1. First, we show how to search for a perfect matching in a general graphs, given that the graph contains a perfect matching. We shall crucially use the fact that we can find an assignment to the Tutte matrix that achieves full rank, Corollary 3.8.2.
2. Next we give a procedure to find a maximum matching in general graphs, by using (1).

4.1 Perfect Matching Search in general graphs

Throughout this section, we assume that we are given a simple graph $G = (V, E)$, that has a perfect matching.

Notation For a weight assignment $W : E \to \mathbb{N}$ on the edges and an evaluated Tutte matrix T', define $T' \circ W$ as the skew-symmetric matrix:

$$(T' \circ W)_{uv} = \begin{cases} T'_{uv} \cdot z^{W(uv)} & \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Let T' be an evaluation of T such that $\operatorname{rank} T' = n$, and let W be a weight assignment on the edges. Denote $A = T' \circ W$, and for any edge $e = (i, j)$, denote A_e as the matrix obtained from A by multiplying A_{ij}, A_{ji} with the variable y and setting $W(e) = 0$, i.e. $(A_e)_{ij}, (A_e)_{ji}$ are $T'_{ij}y$ and $T'_{ji}y$ respectively.

Let $\mathcal{M}$ be the set of all perfect matchings of G. For a perfect matching $M \in \mathcal{M}$ in G, denote by c_M the coefficient[1] of the monomial corresponding to M in $\mathsf{Pf}(T')$. Therefore, we have

$$\mathsf{Pf}(A) = \sum_{M \in \mathcal{M}} c_M \cdot z^{W(M)}$$

Observe that we can write $\mathsf{Pf}(A_e) = P_0(z) + yP_1(z)$ where P_0, P_1 are univariate polynomials in z. Furthermore, for $w \in \mathbb{N}$, let

$$c_w = \sum_{M : W(M) = w} c_M,$$

[1] i.e. Let T'_X be the same matrix as T' but with T'_{ij}, T'_{ji} multiplied by x_{ij} if $(i, j) \in E$. Now let $x_M = \Pi_{e \in M} x_e$. Then c_M is the coefficient of x_M in $\mathsf{Pf}(T'_X)$.

and for $e \in E$:

$$c_w^{\bar{e}} = \sum_{M:W(M)=w, e\notin M} c_M \text{ and } c_w^e = c_w - c_w^{\bar{e}} = \sum_{M:W(M)=w, e\in M} c_M.$$

Therefore, we have that:

$$P_0(z) = \sum_{w\in\mathbb{N}} c_w^{\bar{e}} z^w, P_1(z) = \sum_{w\in\mathbb{N}} c_{w+W(e)}^e z^w$$

We know that T' is full rank, hence $\mathsf{Pf}(A)$ is a non-zero polynomial in z, since at $z = 1$, $\mathsf{Pf}(A)|_{z=1} = \mathsf{Pf}(T') \neq 0$. Let $c_{w_0} z^{w_0}$ be the minimum degree term of $\mathsf{Pf}(A)$ with the non-zero coefficient c_{w_0}. For an edge $e \in E$, let $c_{w_{\bar{e}}}^{\bar{e}} z^{w_{\bar{e}}}$ denote the minimum degree monomial in P_0 not containing the variable y (if such a monomial exists, i.e. if P_0 is a non-zero polynomial in z), and $c_{w_e+W(e)}^e z^{w_e}$ be the minimum degree monomial containing y in P_1 (if it exists, i.e. if P_1 is non-zero).

A high level overview Notice that w_0 may not be the weight of the min-weight perfect matching in G, since the corresponding term for the minimum weight matching may cancel out in $\mathsf{Pf}(A)$. But, nevertheless, we attempt to isolate a matching of weight w_0, though not in the conventional sense i.e. there may be multiple matchings with this weight. Notice that if W isolates a unique minimum weight perfect matching of G with total weight w^*, then the set $\{e \in E : c_{w^*}^{\bar{e}} = 0\}$ is precisely this unique perfect matching. But, in our context there might not be a unique w_0 weight matching, but nevertheless, we show that either of the two holds:

1. the set $\{e \in E : c_{w_0}^{\bar{e}} = 0\}$ is a perfect matching of weight w_0.
2. We can find an edge $e \in E$, such that $W(e) = w_{\bar{e}} - w_e$, and using this we shall develop a 'compress or compute' algorithm.

We make these notions formal in the next lemma. The following is the crucial lemma that helps us find a perfect matching in CLP. can

Lemma 4.1. *With the above notation, let w_0 be the least degree such that z^{w_0} is present with non-zero coefficient in $\mathsf{Pf}(A)$. Then, one of the following holds:*

1. *$\exists e \in E$ such that $W(e) = w_{\bar{e}} - w_e$ where $z^{w_{\bar{e}}}$ is the least degree monomial with non-zero coefficient in P_0 and z^{w_e} is the least degree monomial with non-zero coefficient in P_1 and $\mathsf{Pf}(A_e) = P_0 + yP_1$.*
2. *$M = \{e \in E : c_{w_0}^{\bar{e}} = 0\}$ is a perfect matching. Moreover M is exactly the set of edges $e \in E$ such that P_0 does not contain the monomial z^{w_0}.*

Proof. Notice that $c_{w_0} \neq 0$ and $\forall w < w_0, c_w = 0$. We divide the proof into the following two mutually exclusive cases:

`Case 1.` $\exists e \in E$, such that both $c_{w_0}^e \neq 0$ and $c_{w_0}^{\bar{e}} \neq 0$.

`Case 2.` $\forall e \in E$, at least one of $c_{w_0}^e$ and $c_{w_0}^{\bar{e}}$ is 0.

We will show that if Case 1 holds, then for the same edge e such that $c_{w_0}^e \neq 0$ and $c_{w_0}^{\bar{e}} \neq 0$ holds, we have the equality $W(e) = w_{\bar{e}} - w_e$. And if Case 2 holds, we shall show that $M = \{e \in E : c_{w_0}^{\bar{e}} = 0\}$ is a perfect matching in G. We start with the first case.

`Case 1.` Suppose that there exists an edge $e \in E$ such that $c_{w_0}^e$ and $c_{w_0}^{\bar{e}}$ both are nonzero. Write as before

$$\mathsf{Pf}\,(A_e) = P_0(z) + yP_1(z)$$

and $w_{\bar{e}}, w_e$ are the minimum degrees in P_0, P_1 respectively. Since $c^e_{w_0}$ and $c^{\bar{e}}_{w_0}$ both are nonzero, we have $w_{\bar{e}}, w_e - W(e) \leq w_0$. If $w_{\bar{e}} = w_e + W(e)$ then we are done. Else, $w_{\bar{e}} < w_e + W(e)$ or $w_{\bar{e}} > w_e + W(e)$ are the two cases we consider:

1. In the first case, we have $w_{\bar{e}} < w_e + W(e)$, and therefore the coefficient of $z^{w_{\bar{e}} - W(e)}$ is zero in P_1, i.e, $c^e_{w_{\bar{e}}} = 0$. Thus, observe that $c_{w_{\bar{e}}} = c^e_{w_{\bar{e}}} + c^{\bar{e}}_{w_{\bar{e}}} \neq 0$, implying that $w_{\bar{e}} \geq w_0$. Combining with the fact that $w_{\bar{e}} \leq w_0$ we have $w_{\bar{e}} = w_0$. But then we have $c^e_{w_0} = 0$ that contradicts our assumption on e.
2. The second case is symmetric, with the assumption $w_{\bar{e}} > w_e + W(e)$. We again see that the coefficient of $z^{w_e + W(e)}$ in P_0 is zero, i.e, $c^{\bar{e}}_{w_e + W(e)} = 0$. This in turn means $c_{w_e + W(e)} \neq 0$, implying $w_e + W(e) \geq w_0$. Combining with the fact that $w_e + W(e) \leq w_0$, we have the equality $w_e + W(e) = w_0$. But then we have $c^{\bar{e}}_{w_0} = 0$ giving a contradiction.

We have therefore shown that $W(e) = w_{\bar{e}} - w_e$. Now we move on to the other case.

`Case 2.` We now assume that $\forall e \in E$, at least one of $c^e_{w_0}$ and $c^{\bar{e}}_{w_0}$ is 0. Let $M = \{e \in E : c^{\bar{e}}_{w_0} = 0\}$. Since $c_{w_0} \neq 0$, we have $M = \{e \in E : c^e_{w_0} \neq 0 \wedge c^{\bar{e}}_{w_0} = 0\}$. We wish to show that M is a perfect matching of G. We proceed by contradiction. Consider the graph $G' = (V, M)$. If M is not a perfect matching, there is either a vertex with degree 0 in G', or some vertex has degree at least 2 in G'. We divide the proof into these two cases and show a contradiction in both of them:

1. Suppose v has degree 0 in G'. Let $e_1, \ldots, e_k$ be the edges in G that are incident on v. For all e_i we know that $e_i \notin M \implies c^{e_i}_{w_0} = 0$. But then we have
$$c_{w_0} = \sum_{i \in [k]} c^{e_i}_{w_0} = 0$$
which is a contradiction, since $c_{w_0} \neq 0$.
2. Now suppose a vertex v has degree ℓ in G' where $\ell > 1$. Suppose $e_1, \ldots, e_k$ are the edges incident on v in G and $e_1, \ldots, e_\ell \in M, e_{\ell+1}, \ldots, e_k \notin M$. Therefore, we have the following:
$$\forall i \in [\ell] : c^{\bar{e}_i}_{w_0} = 0, \forall j > \ell : c^{e_j}_{w_0} = 0$$
and therefore,
$$\forall i \in [\ell] : c^{\bar{e}_i}_{w_0} = \sum_{j \in [\ell] \setminus \{i\}} c^{e_j}_{w_0} = 0.$$
Summing the above for all $i \in [\ell]$, we get
$$\sum_{i \in [\ell]} c^{\bar{e}_i}_{w_0} = \sum_{i \in [\ell]} \sum_{j \in [\ell] \setminus \{i\}} c^{e_j}_{w_0} = (\ell - 1) \sum_{i \in [\ell]} c^{e_i}_{w_0} = 0.$$
But since $\ell > 1$, we have:
$$c_{w_0} = \sum_{i \in [\ell]} c^{e_i}_{w_0} = 0.$$
Again this gives a contradiction since $c_{w_0} \neq 0$.

By the above argument, we have that every vertex has degree exactly 1 in G'. This happens precisely when M is a perfect matching of G, and we are done. □

Finally, we come to the following algorithm:

Lemma 4.2. *There is a* CLP *procedure that, given a simple undirected graph G on n vertices and m edges which contains a perfect matching, returns a perfect matching of G.*

Proof. Assume that the catalytic tape is given as n^3 weight assignments $(W_1, W_2, \ldots, W_{n^3})$ where each $W_r : E \to [n^{10}]$ is given as $(W_i(e_1), \ldots, W_i(e_m))$. We assume that we are given an assignment to the Tutte matrix T' by Corollary 3.8.2 in the sense of Remark 3.9. Then we do the following, by processing $W_1, \ldots, W_{n^3}$ sequentially:

1. Suppose, we are processing W_r. Set $A = T' \circ W_r$. For all $e \in E$ check whether $W_r(e) = w_{\bar{e}} - w_e$ where $z^{w_{\bar{e}}}$ is the least degree monomial with non-zero coefficient (can be found using interpolating) in P_0 and z^{w_e} is the least degree monomial with non-zero coefficient in P_1. (recall that A_e is the matrix obtained from A by multiplying A_{ij}, A_{ji} by the variable y and setting $W(e) = 0$, and $\mathsf{Pf}(A_e) = P_0 + yP_1$) If we find such an edge e_i (i.e. the i^{th} edge), then reset the current catalytic tape string to $(i, W_r(e_1), \ldots, W_r(e_{i-1}), W_r(e_{i+1}), \ldots, e_m)$. This frees up $10 \log n - \log m \geq 8 \log n$ bits of space. On the other hand, if we did not find such an edge, using Lemma 4.1, we find a perfect matching as follows: Compute w_0 to be the minimum degree such that z^{w_0} is present in $\mathsf{Pf}(A)$. Find all edges e such that P_0 does not contain the monomial z^{w_0} where $\mathsf{Pf}(A_e) = P_0 + yP_1$. This set of edges precisely forms the perfect matching $M = \{e \in E : c^{\bar{e}}_{w_0} = 0\}$ by Lemma 4.1. After finding this matching and returning it, we move to step 3 (to restore the previous catalytic strings) with the temporary variable $index \leftarrow r - 1$.
2. Suppose that we just processed W_{n^3} and still did not find a perfect matching. In this case, we have saved up $\mathcal{O}(n^3 \log n)$ bits of space, where we can run any standard algorithm (say [Gee00] with the self-reduction step) to search for a perfect matching. Then we set $index \leftarrow n^3$ and go to step 3 to restore the catalytic strings.
3. We restore all the weights $W_1, \ldots, W_{index}$ as follows: Suppose we are restoring W. We are given the catalytic string $(i, W(e_1), \ldots, W(e_{i-1}), W(e_{i+1}), \ldots, W(e_m))$. Compute A_{e_i} (can be done since we set $W(e_i) = 0$). Compute $w_{\bar{e}_i}, w_{e_i}$ such that $z^{w_{\bar{e}_i}}$ is the least degree monomial with non-zero coefficient in P_0 and $z^{w_{e_i}}$ is the least degree monomial with non-zero coefficient in P_1. Set $W(e_i) = w_{\bar{e}_i} - w_{e_i}$. Reset W to $(W(e_1), \ldots, W(e_m))$.

Clearly we have always restored the catalytic tape to its initial state when our computation halts, and we always return a perfect matching. The space bounds are: $\mathcal{O}(\log n)$ workspace, $\mathcal{O}(n^3 m \log n)$ catalytic space, and the algorithm runs in polynomial time. □

4.2 Maximum Matching Search

Finally, we have the main result of this section, as follows:

Theorem 4.3. *Given a simple graph $G = (V, E)$, we can construct a maximum matching M of cardinality $\nu(G)$ in* CLP.

Proof. Let $G = (V, E)$ be a graph, and suppose we are given the size $k = \nu(G)$ of a maximum matching in G by Theorem 3.8. Let $d = n - 2k$.

Construct a graph $G' = (V', E')$ as follows:

Add d new dummy vertices $w_1, \ldots, w_d$. Let $V' = V \cup \{w_1, \ldots, w_d\}$, and $E' = E \cup \{(w_i, v) : i \in [d], v \in V\}$.

Clearly, there perfect matching in G'. Suppose M is a matching of size k in G. Then exactly $|V| - 2k = d$ vertices of V are unmatched by M. Match each dummy vertex w_i to a distinct unmatched vertex of V. Together with M, this gives a perfect matching of G'.

Conversely, suppose G' has a perfect matching M'. Since there are exactly d dummy vertices, and each dummy vertex can only contribute one matched edge, exactly d edges of M' are incident to dummy vertices (since no edge is between two dummy vertices). Removing all such edges leaves a maximum matching M in G.

Therefore, we can find a maximum matching of G in CLP as follows: First construct G' in logspace, and compute a perfect matching M' of G' using Lemma 4.2. Next, using the above stated logspace procedure, compute a maximum M matching of G by removing the edges of M' incident on the dummy vertices.

□

5 Rank of Mixed Matrices

In this section, we prove Theorem 1.2.

5.1 Mixed matrices

Definition 5.1. *Let A be an $r \times c$ matrix where each entry of A is either an indeterminate, or an element of $\mathbb{Q}$, where the set of variables is $X = \{x_{ij} : i \in [r], j \in [c], A_{ij} = x_{ij}\}$.*

Mixed matrices can also be written as $A = K + Q$ where Q is a scalar matrix over $\mathbb{Q}$ and K is an indeterminate matrix (where no two indeterminates in two coordinates are the same, in the same sense as Definition 5.1). For any $r \times c$ matrix A whose rows are indexed by L_r and columns by L_c, define $D(A) = \{i \in L_r \cup L_c : \operatorname{rank}(A \setminus \{i\}) = \operatorname{rank} A\}$. Let $E = \{(i, j) : A_{ij} \in X\}$, further index these tuples such that $E = \{i : i = (i', j'), A_{i'j'} \in X\}$.

Theorem 5.1 (Geelen, [Gee99])**.** *Let A be a mixed matrix over the variable set X where $|X| = m$, and let A' be an evaluation of A. Then either* $\operatorname{rank} A' = \operatorname{rank} A$ *and $D(A') = D(A)$, or one of the following two statements hold:*

- $\exists i \in E, a \in [n^{10}]$ *such that* $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' + 1$.
- $\exists i \in E, a \in [n^{10}]$ *such that* $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' \wedge D(A') \subsetneq D(A'_{x_i \leftarrow a})$.

Analogously to Lemmas 3.5 and 3.6 ($2A, 2B$), we show the following uniqueness lemmata, which will be crucial for our CL algorithm to work.

Lemma 5.2 ($2A'$)**.** *Let A' be an evaluation of A and $i \in E, a \in [n^{10}]$ such that $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' + 1$, and in A', x_i is a_i. Then $\forall \tilde{a} \neq a_i$, we have $\operatorname{rank} A'_{x_i \leftarrow \tilde{a}} = \operatorname{rank} A' + 1$.*

Proof. Let $k = \operatorname{rank} A'$. Let $X \subseteq L_r, Y \subseteq L_c, |X| = |Y| = k$ be a minor of A that is non-singular in $A'_{x_i \leftarrow a}$. Notice that $L(y) = \mathsf{Det}(A'_{x_i \leftarrow y}[X, Y])$ is linear in y. Since $L(a) \neq 0, L(a_i) = 0$, clearly $\forall \tilde{a} \neq a_i, L(\tilde{a}) \neq 0$ i.e $A'_{x_i \leftarrow \tilde{a}}[X, Y]$ is non-singular. Furthermore, on changing one entries of A' (i.e. by changing the variable x_i) the rank can go up by atmost 1. Therefore, $\operatorname{rank} A'_{x_i \leftarrow \tilde{a}} = \operatorname{rank} A' + 1$ holds $\forall \tilde{a} \neq a$. □

Lemma 5.3 ($2B'$)**.** *Let A' be an evaluation of A and $i \in E, a \in [n^{10}]$ such that $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' = k$ where in A', x_i is set to a_i. Suppose $\exists u \in D(A'_{x_i \leftarrow a}) \setminus D(A')$. Then $\forall \tilde{a} \neq a_i$, we have $u \in D(A'_{x_i \leftarrow \tilde{a}})$.*

Proof. From the given assumptions, we have a $X_u \subseteq L_r, Y_u \subseteq L_c, |X_u| = |Y_u| = k$ such that $A'_{x_i \leftarrow a}[X_u, Y_u]$ is non-singular and $u \notin X_u \cup Y_u$. Since $u \notin D(A')$, $A'[X_u, Y_u]$ is singular.

Claim 5.4. $\forall \tilde{a} \neq a_i$, $\operatorname{rank} A'_{x_i \leftarrow \tilde{a}} \leq k$.

Proof. Suppose $\operatorname{rank} A'_{x_i \leftarrow \tilde{a}} > k$ for some $\tilde{a}$. Therefore, $\tilde{a} \notin \{a, a_i\}$. Therefore, we can find X, Y as a square minor of size larger than k such that $A'_{x_i \leftarrow \tilde{a}}[X, Y]$ is non-singular. But then $\widetilde{L}(y) = \mathsf{Det}(A'_{x_i \leftarrow y}[X, Y])$ is linear and $\widetilde{L}(a) = \widetilde{L}(a_i) = 0$. Therefore, $\widetilde{L}(\tilde{a}) = 0$ gives a contradiction. □

Again $L(y) = \mathsf{Det}(A'_{x_i \leftarrow y}[X_u, Y_u])$ is linear in y, such that $L(a) \neq 0, L(a_i) = 0$. Therefore, $\forall \tilde{a} \neq a_i$ we have $L(\tilde{a}) \neq 0$ i.e $A'_{x_i \leftarrow \tilde{a}}[X_u]$ is non-singular and by the above claim X_u, Y_u is of maximum possible size (i.e. $\operatorname{rank} A'_{x_i \leftarrow \tilde{a}} = |X_u| = |Y_u| = k$), thus $u \in D(A'_{x_i \leftarrow \tilde{a}})$. □

Lemma 5.5. *Suppose that we are given an evaluation of a mixed matrix A, namely A'. Then we can check, in* CL *if $\operatorname{rank} A = \operatorname{rank} A'$ or not.*

Proof. The proof is very similar to Algorithm 1 and uses Theorem 5.1. For all i and all perturbations $x_i \leftarrow a (\forall a \in [m^{10}])$ check if the rank increases by 1 or if the D set increases. If for no such perturbation this happens, we conclude by Theorem 5.1 that $\operatorname{rank} A' = \operatorname{rank} A$, and otherwise we know that $\operatorname{rank} A' < \operatorname{rank} A$. □

5.2 A catalytic rank algorithm

In this section, we show that the rank of mixed matrices can be computed in CL. Let A be a mixed matrix with variables $x_1, \ldots, x_m$. The algorithm is essentially the same as the matching algorithm described in Section 3. We sample an assignment $a_1, \ldots, a_m$ for these variables from the catalytic tape. Either this assignment maximizes the rank of the evaluated matrix A', or it does not. We can verify this using Lemma 5.5. If the rank is indeed maximized, we proceed to restore the modified catalytic tape.

Otherwise, if the rank is not maximized, we consider two cases:

(a) Suppose that by perturbing one of the variables x_i, the rank of A' can be increased. In this case, by Lemma 5.2, for all values other than a_i, the rank increases. Therefore, we discard a_i from the catalytic tape and store the index i together with the rank. Later, we can recover a_i as the unique value for which the rank remains unchanged.

(b) Otherwise, perturbing some variable x_i does not increase the rank, but enlarges the set $D(A')$, i.e., there exists a vertex u that gets added to $D(A')$. By Lemma 5.3, a_i is the unique value for which $u \notin D(A')$. Thus, we again discard a_i and store its index i together with u. As before, a_i can be recovered when needed using this characterization.

Finally, if the sampled assignment already yields the maximum rank, we are done. Otherwise, we have freed enough space to recompute the rank from scratch using [Gee99]. A formal proof follows.

Theorem 5.6. *Let A be an $r \times c$ dimensional mixed matrix on variables X indexed over the set E of size $m \leq n^2$, and $n = r + c$. Then, there is a* CLP *procedure that finds* $\operatorname{rank} A$*, and an assignment to X (and thus an evaluation of A, namely A') that is of the maximum possible rank, i.e.,* $\operatorname{rank} A' = \operatorname{rank} A$.

Proof. The algorithm mirrors the proof of Theorem 3.8. We assume that the catalytic tape is divided into blocks $(\mathcal{C}_1, \ldots, \mathcal{C}_N, \mathcal{B})$ where $\mathcal{C}_t = (a_1, \ldots, a_m) \forall t \in [N]$ with each $a_i \in [n^{10}]$ is $10 \log n$ bits long, and $\mathcal{B}$ is for rank computations; $N = n^3$. The algorithm proceeds as follows: We iterate over all $\mathcal{C}_1, \ldots, \mathcal{C}_N$ as follows, suppose we are processing $\mathcal{C}_t$:

1. Let A' be the evaluation of A given by $\mathcal{C}_t$ i.e. by putting $x_i \leftarrow a_i \forall i \in [m]$. We first check if $\operatorname{rank} A' = \operatorname{rank} A$ or not, using Lemma 5.5. If yes, then we have found a required assignment, and we proceed to restore all previous catalytic blocks, i.e. jump to Step 4 with the temporary variable $index \leftarrow t-1$. Now suppose we did not get the required assignment. Therefore, from Theorem 5.1, the only two possibilities are:

   (a) $\exists i \in E, a \in [n^{10}]$ such that $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' + 1$.

   (b) $\exists i \in E, a \in [n^{10}]$ such that $\operatorname{rank} A'_{x_i \leftarrow a} = \operatorname{rank} A' \wedge D(A') \subsetneq D(A'_{x_i \leftarrow a})$.

   Consider the two cases $2A', 2B'$ that deals with (a) and (b) respectively, as follows–

$2A'$. In this case, after finding an i (we enumerate over all i to find such an i) as in (a), from Lemma 5.2, we know that for all substitutions $x_i \leftarrow a$ and $a \neq a_i$, we have $\operatorname{rank} A'_{x_i \leftarrow a} \neq \operatorname{rank} A'$. Therefore, we replace $\mathcal{C}_t$ with $(2A', i, \operatorname{rank} A', a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m)$. This saves $6 \log n$.

$2B'$. Again, first we find an index i, and value a witnessing case (b). Now, we pick a $u \in D(A'_{x_i \leftarrow a}) \setminus D(A')$. Again from Lemma 5.3 we know that a_i is the unique value for which $D(A')$ does not contain u. Hence we replace $\mathcal{C}_r$ with $(2B', i, u, a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_m)$. This again saves $6 \log n$ bits.

3. Suppose we just processed $\mathcal{C}_N$ and still did not find a required assignment. In this case, we have freed up at least $\mathcal{O}(n^3 \log n)$ space for $N = n^3$. We shift the catalytic strings to make this space contiguous, and run Geelen's algorithm to find an assignment that maximizes the rank. Next, set $index = N$ and go to step 4.

4. In either case, we now recompute the modified catalytic strings $\mathcal{C}_1, \ldots, \mathcal{C}_{index}$. For each $t \in [index]$, we execute the following with $\mathcal{C}_t = (case, j, \beta, a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_m)$:

   (a) This is when $case = 2A'$. Therefore, we have $k = \beta$ to be the rank of A' with the evaluation of T given by the original catalytic tape. Thus, we find the unique a such that $\operatorname{rank} A'_{x_j \leftarrow a} = k$, and restore $a_j \leftarrow a$. The correctness is again given by Lemma 5.2.

   (b) This time $case = 2B'$. Therefore $u = \beta$. Again find the unique a such that $u \notin D(T'_{x_j \leftarrow a})$ and restore $a_j \leftarrow a$. The correctness follows from Lemma 5.3.

   Finally we reset $\mathcal{C}_t \leftarrow (a_1, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_m)$.

The above algorithm clearly runs in polynomial time and the catalytic space used is $\mathcal{O}(mn^2 \log n + |\mathcal{B}|) = \mathsf{poly}(n)$ with work space $\mathcal{O}(\log n)$, and the catalytic tape is always restored when the algorithm halts. Thus the above gives the required CLP algorithm that finds $\operatorname{rank} A$ and an assignment $\vec{a}$ such that $\operatorname{rank} A = \operatorname{rank} A_{x_i \leftarrow a_i}$. □

Corollary 5.6.1. *Given two linear matroids M_1 and M_2, we can find the cardinality of the maximum common independent set in* CLP.

Proof. Let M_1 and M_2 be matroids on a common ground set E, $|E| = n$. Suppose M_1 and M_2 have their representation matrices A_1 and A_2 repectively. Let $\{x_1 \ldots, x_n\}$ be the set of variables. Consider the following matrix:

$$A := \left(\begin{array}{c|ccc} 0 & \multicolumn{3}{c}{A_1} \\ \hline & x_1 & & \\ A_2^T & & \ddots & \\ & & & x_n \end{array}\right).$$

In [Mur95, Mur00], Murota showed that the size of a maximum common independent set of M_1 and M_2 is $\operatorname{rank} A - n$. Since A is a mixed matrix, and we can compute $\operatorname{rank} A$ using Theorem 5.6, and we are done. □

6 $(1-\varepsilon)$-approximation for Edmonds's Problem

In this section we prove Theorem 1.4.

Given a matrix space $\mathcal{A} = \langle A_1, \ldots, A_m \rangle$ where each matrix is a $n \times n$ matrix over some underlying field $\mathbb{F}$ of size greater that some polynomial in n, say $|\mathbb{F}| > n^c$ for a carefully chosen c that depends inversely on ε. We want to approximate the maximum rank that any matrix in $\mathcal{A}$ can attain. Consider the indeterminates $x_1, \ldots, x_m$, we want to approximate the rank of $\sum_{i \in [m]} A_i x_i =: \operatorname{rank}(\mathcal{A})$.

Theorem 6.1 ([BJP18])**.** *Let $\mathcal{A} = \langle A_1, \ldots, A_m \rangle \leq \mathbb{F}^{n \times n}$, and $\varepsilon \in (0, 1)$ be a fixed parameter with $\ell = \lceil \frac{1}{\varepsilon} - 1 \rceil$. Let $a_i \in \mathbb{F}$ for $i \in [m]$ and S be any subset of $\mathbb{F}$ of size n^c. Then either:*

1. $\operatorname{rank}\left(\sum_{i \in [m]} A_i a_i\right) \geq (1 - \varepsilon) \cdot \operatorname{rank}\left(\sum_{i \in [m]} A_i x_i\right)$.

2. *or,* $\exists I = \{i_1, \ldots, i_\ell\} \in \binom{[m]}{\ell}$ *and* $\lambda_{i_1} \ldots, \lambda_{i_\ell} \in S$ *such that*

$$\operatorname{rank}\Big(\sum_{i\in[m]\setminus I} A_i a_i + \sum_{i\in I} A_i\lambda_i\Big) > \operatorname{rank}\Big(\sum_{i\in[m]} A_i a_i\Big)$$

Henceforth, we fix S as the first n^c elements of $\mathbb{F}$ in the canonical order. Even though we do not have any uniqueness lemma like Lemmas 3.5 and 3.6, we show the following which is essentially an application of the Schwartz-Zippel lemma, and shall be crucial for our purposes.

Lemma 6.2. *Suppose* $\vec{a} = a_1, \ldots, a_m \in \mathbb{F}$ *and* $\exists I = \{i_1, \ldots, i_\ell\} \in \binom{[m]}{\ell}$ *and* $\lambda_{i_1} \ldots, \lambda_{i_\ell} \in S$ *such that*

$$\operatorname{rank}\Big(\sum_{i\in[m]\setminus I} A_i a_i + \sum_{i\in I} A_i\lambda_i\Big) > \operatorname{rank}\Big(\sum_{i\in[m]} A_i a_i\Big).$$

Let

$$\mathfrak{I}_{\vec{a},I} = \Big\{\lambda'_{i_1}, \ldots, \lambda'_{i_\ell} \in S : \operatorname{rank}\Big(\sum_{i\in[m]\setminus I} A_i a_i + \sum_{i\in I} A_i\lambda'_i\Big) = \operatorname{rank}\Big(\sum_{i\in[m]} A_i a_i\Big)\Big\}.$$

Then $|\mathfrak{I}_{\vec{a},I}| \le n^{c\ell-c+1}$.

Proof. Let $A' = \sum_{i\in[m]} A_i a_i$, $A'' = \sum_{i\in[m]\setminus I} A_i a_i + \sum_{i\in I} A_i\lambda_i$. Let $X, Y \subseteq [n]$ be of maximum possible size such that $A''[X, Y]$ is a square non-singular matrix, but $A'[X, Y]$ is singular. Let

$$p(x_{i_1}, \ldots, x_{i_\ell}) = \mathsf{Det}\Big(\sum_{i\in[m]\setminus I} A_i a_i[X, Y] + \sum_{i\in I} A_i x_i[X, Y]\Big)$$

be a polynomial in the indeterminates $x_{i_1}, \ldots, x_{i_\ell}$. Clearly p is of degree at most n (to be precise, the degree is at most $|X| \le n$), and $p(\lambda_{i_1}, \ldots, \lambda_{i_\ell}) \neq 0$. Therefore, from Lemma 1.3, the number of roots of p in S^ℓ is upper bounded by $|S^\ell|.(n/|S|) = n^{c\ell-c+1}$. Moreover, for any $(\lambda'_{i_1}, \ldots, \lambda'_{i_\ell}) \in \mathfrak{I}_{\vec{a},I}$, we have $p(\lambda'_{i_1}, \ldots, \lambda'_{i_\ell}) = 0$. Therefore, $|\mathfrak{I}_{\vec{a},I}| \le n^{c\ell-c+1}$. □

We shall approximate $\operatorname{rank}(\mathcal{A})$ upto a factor of $(1-\varepsilon)$ in CL. We sample $x_1 \leftarrow a_1, \ldots, x_m \leftarrow a_m$ from the catalytic tape. Either we already have landed up at an approximately good assignment A' in which case we begin reconstructing the modified catalytic strings. Otherwise, we know that we can find $I = (i_1, \ldots, i_\ell)$ and a perturbation to $x_{i_1}, \ldots, x_{i_\ell}$ for which the rank increases. But then, the above lemma implies that $|\mathfrak{I}_{\vec{a},I}|$ is smaller than $|S|^\ell$ by a polynomially large factor. Notice that $I \in \mathfrak{I}_{\vec{a},I}$, and therefore I is the j^{th} tuple in $\mathfrak{I}_{\vec{a},I}$ for some j that requires a small number of bits to represent. Therefore, we forget $a_{i_1}, \ldots, a_{i_\ell}$ from $\vec{a}$ and remember instead I, j and the rank of A'. We can later reconstruct $a_{i_1}, \ldots, a_{i_\ell}$ by finding the j^{th} tuple of $\mathfrak{I}_{\vec{a},I}$. Again, if we never find a suitable catalytic tape, then we have saved enough space to run [BJP18] and find a $(1-\varepsilon)$ approximation of the rank, and a required assignment. We now present the proof in detail:

Theorem 6.3. *For any constant* $\varepsilon \in (0, 1)$*, and large enough field* $\mathbb{F}$ *such that* $|\mathbb{F}| \ge n^{\mathcal{O}(1/\varepsilon)}$*, suppose we are given an instance of Edmond's problem,* $\mathcal{A} = \langle A_1, \ldots, A_m\rangle$*. Then, we have a* CLP *algorithm that returns a matrix* $A \in \mathcal{A}$ *such that* $\operatorname{rank} A \ge (1-\varepsilon)\cdot\operatorname{rank}(\mathcal{A})$.

Algorithm 4 ProcessAssignment$(\mathcal{C}, \mathcal{A})$

Given is $\mathcal{C} = (a_1, \ldots, a_m)$ where $a_i \in S \forall i \in [m]$.
$A' \leftarrow \sum_{i \in [m]} A_i a_i, k \leftarrow \operatorname{rank} A'$
$flag \leftarrow 0$
for $I = \{i_1, \ldots, i_\ell\} \in \binom{[m]}{\ell}, \lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S$ **do**
    $A'' \leftarrow \sum_{i \in [m] \setminus I} A_i a_i + \sum_{i \in I} A_i \lambda_i$ ▷ $I = \{i_1, \ldots, i_\ell\}$
    if $\operatorname{rank} A'' > \operatorname{rank} A'$ **then**
        $flag \leftarrow 1$
    end if
end for
if $flag = 0$ **then**
    return A'
end if
$j \leftarrow 0$
for $I = \{i_1, \ldots, i_\ell\} \in \binom{[m]}{\ell}, \lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S$ **do**
    $A'' \leftarrow \sum_{i \in [m] \setminus I} A_i a_i + \sum_{i \in I} A_i \lambda_i$
    if $\operatorname{rank} A'' = k$ **then**
        $j \leftarrow j + 1$
    end if
    if $(\lambda_{i_1} = a_{i_1}) \wedge \ldots \wedge (\lambda_{i_\ell} = a_{i_\ell})$ **then**
        Reset $\mathcal{C} \leftarrow (i_1 \ldots, i_\ell, j, k, \vec{a}^{-I})$ ▷ $\vec{a}^{-I}$ is $\{a_1 \ldots, a_m\} \setminus \{a_{i_1}, \ldots, a_{i_\ell}\}$
        return ▷ Terminate the algorithm
    end if
end for

Algorithm 5 RestoreAssignment$(\mathcal{C}, \mathcal{A})$

Input: We are given $\mathcal{C} = (i_1 \ldots, i_\ell, j, k, \vec{a}^{-I})$.
$j' = 0$
for $\lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S$ **do**
    $A'' \leftarrow \sum_{i \in [m] \setminus I} A_i a_i + \sum_{i \in I} A_i \lambda_i$
    if $\operatorname{rank} A'' = k$ **then**
        $j' \leftarrow j' + 1$
    end if
    if $j = j'$ **then**
        $a_{i_1} \leftarrow \lambda_{i_1}, \ldots, a_{i_\ell} \leftarrow \lambda_{i_\ell}$
        Reset $\mathcal{C} \leftarrow (a_1, \ldots, a_m)$
        return ▷ Terminate the algorithm
    end if
end for

Proof. Let S be the first n^c elements of $\mathbb{F}$ that therefore require $c \log n$ bits to store. We shall throughout the algorithm, work with S. Fix $\ell = \lceil 1/\varepsilon - 1 \rceil, N = n^3$ and $c = 2\ell + 3$. Assume that the catalytic tape is divided into blocks $(C_1, \ldots, C_N, \mathcal{B})$ where $C_t = (a_1, \ldots, a_m) \forall t \in [N]$ and each $a_i \in S$ takes up $c \log n$ bits of storage. $\mathcal{B}$ is again for rank computations. Let $A = A_1 x_1 + \ldots A_m x_m$ where $x_1, \ldots, x_m$ are variables. The algorithm proceeds as follows: We iterate over all $\mathcal{C}_1, \ldots, \mathcal{C}_N$ as follows, suppose that we are processing $\mathcal{C}_t = (a_1, \ldots, a_m)$:

1. We Execute Algorithm 4. Let A' be the evaluation of A given by $\mathcal{C}_t$ and k be its rank i.e. by putting $x_i \leftarrow a_i \forall i \in [m]$. If for no perturbation in S for any ℓ variables, gives a matrix of larger rank than A' then we have found A' which approximates the rank of A upto a factor of $(1 - \varepsilon)$ by Theorem 6.1. Thus we jump to Step 4 to reconstruct the previous catalytic blocks by setting $index = t - 1$. On the other hand, suppose for some $I = (i_1, \ldots, i_\ell) \in \binom{[m]}{\ell}$, and some perturbation $\lambda_{i_1}, \ldots, \lambda_{i_\ell}$, the rank of A' increases. Then, from Lemma 6.2 we know that $\mathfrak{I}_{\vec{a},I}$ is of size at most $n^{c\ell - c + 1}$. Therefore, we enumerate over all $\lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S$ and check if $(\lambda_{i_1}, \ldots, \lambda_{i_\ell}) \in \mathfrak{I}_{\vec{a},I}$. This way, we find j such that $(a_{i_1}, \ldots, a_{i_\ell})$ is the j^{th} tuple in $\mathfrak{I}_{\vec{a},I}$. Finally, we reset $\mathcal{C} \leftarrow (i_1 \ldots, i_\ell, j, k, \vec{a}^{-I})$ where $\vec{a}^{-I}$ is $\{a_1 \ldots, a_m\} \setminus \{a_{i_1}, \ldots, a_{i_\ell}\}$. Notice that the index j can be written with memory only $(c\ell - c + 1) \log n$. Therefore, in this process, we have freed up $cm \log n - \ell \log n - (c\ell - c + 1) \log n - \log n - c(m - \ell) \log n = (c\ell - \ell - c\ell + c - 1 - 1) \log n = \mathcal{O}(\log n)$ since we have chosen $c = 2\ell + 3$.

2. Suppose we just processed $\mathcal{C}_N$ and still did not find a required assignment. In this case, we have freed up at least $\mathcal{O}(n^3 \log n)$ space for $N = n^3$. We shift the catalytic strings to make this space contiguous, and run [BJP18] algorithm to find an assignment A' achieves the $(1 - \varepsilon)$ approximation for $\text{rank}(\mathcal{A})$. Next set $index = N$ and go to step 4.

3. In either case, we now recompute the modified catalytic strings $\mathcal{C}_1, \ldots, \mathcal{C}_{index}$. For each $t \in [index]$, we execute Algorithm 5 with $\mathcal{C}_t = (i_1 \ldots, i_\ell, j, k, \vec{a}^{-I})$ as follows:

   We wish to compute the j^{th} tuple from the set $\mathfrak{I}_{\vec{a},I}$. Notice that

   $$\mathfrak{I}_{\vec{a},I} = \left\{ \lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S : \text{rank} \left(\sum_{i \in [m] \setminus I} A_i a_i + \sum_{i \in I} A_i \lambda_i \right) = k \right\}.$$

   Therefore we enumerate over all $\lambda_{i_1}, \ldots, \lambda_{i_\ell} \in S$ until we find the j^{th} tuple $(a_{i_1}, \ldots, a_{i_\ell}) \in \mathfrak{I}_{\vec{a},I}$. Finally we reset $\mathcal{C} \leftarrow (a_1, \ldots, a_m)$.

The above algorithm clearly runs in polynomial time $\mathsf{poly}(n^\ell)$, and the catalytic space used is $\mathcal{O}(c \cdot mn^2 \log n + |\mathcal{B}|) = \mathsf{poly}(n, \ell)$ with work space $\mathcal{O}(\ell \log n)$, and the catalytic tape is always restored when the algorithm halts. Thus the above gives the required CLP algorithm that finds a matrix $A \in \mathcal{A}$ such that $\text{rank}\, A \geq (1 - \varepsilon) \cdot \text{rank}(\mathcal{A})$. □

Corollary 6.3.1. *Given an instance of Linear Matroid Matching M, we have a* CLP *algorithm that approximates the size of a maximum independent matching of M upto a factor of $(1 - \varepsilon)$.*

Proof. Let V be the ground set of the matroid and T be the indeterminate Tutte matrix corresponding to the underlying graph on V given by the instance M. Let

A be the linear representation matrix of M. Then [Mur00] shows that the size of the maximum independent matching in M is exactly $1/2$ times $\text{rank}(ATA^T)$. Therefore, Theorem 6.3 gives the desired result. □

References

[AAV26] Aryan Agarwala, Yaroslav Alekseev, and Antoine Vinciguerra. Linear matroid intersection is in catalytic logspace. In *Proceedings of the 17th Innovations in Theoretical Computer Science Conference (ITCS 2026)*, volume 362 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:23, 2026.

[ACD26] V. Arvind, Srijan Chakraborty, and Samir Datta. Derandomizing isolation in catalytic logspace. 2026.

[AM25] Aryan Agarwala and Ian Mertz. Bipartite matching is in catalytic logspace. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 360–372. IEEE, 2025.

[BCK+14] Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Catalytic computation. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 101–110. ACM, 2014.

[BJP18] Markus Bläser, Gorav Jindal, and Anurag Pandey. A deterministic ptas for the commutative rank of matrix spaces. *Theory of Computing*, 14(3):1–21, 2018.

[CDK+26] Petr Chmel, Aditi Dudeja, Michal Koucký, Ian Mertz, and Ninad Rajgopal. Frontier space-time algorithms using only full memory. 2026.

[CGM+25] James Cook, Surendra Ghentiyala, Ian Mertz, Edward Pyne, and Nathan S. Sheffield. The structure of in-place space-bounded computation, 2025.

[CLMP25] James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. The structure of catalytic space: Capturing randomness and time via compression. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*, pages 554–564, 2025.

[DL78] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[Ede26] Roman Edenhofer. A space-space trade-off for directed st-connectivity. 2026.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[FGT16] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 754–763, 2016.

[Gee99] James F. Geelen. Maximum rank matrix completion. *Linear Algebra and its Applications*, 288(1-3):211–217, 1999.

[Gee00] Jim Geelen. An algebraic matching algorithm. *Combinatorica*, 20(4):577–588, 2000.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. *Fundamenta Mathematicae*, 103:119–130, 1979.

[Mur95] Kazuo Murota. Mixed matrices: irreducibility and decomposition. *Discrete Applied Mathematics*, 56(2-3):205–221, 1995.

[Mur00] Kazuo Murota. *Matrices and Matroids for Systems Analysis*, volume 20 of *Algorithms and Combinatorics*. Springer, 2000.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Pyn24] Edward Pyne. Derandomizing logspace with a small shared hard drive. In Rahul Santhanam, editor, *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 4:1–4:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[ST17] Ola Svensson and Jakub Tarnawski. Maximum matching in general graphs in quasi-nc. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 696–707, 2017.

[Tut47] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22:107–111, 1947.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation (EUROSAM)*, pages 216–226, 1979.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il