Parallel Algorithms for Bipartite Perfect Matching

Stephen Fenner
University of South Carolina
USA
fenner.sa@gmail.com

Rohit Gurjar
California Institute of Technology
USA
rohitgurjar0@gmail.com

Thomas Thierauf*
Aalen University
Germany
thomas.thierauf@uni-ulm.de

ABSTRACT

One of the fundamental questions in theory of computing is to understand the power of randomness. It is not known whether every problem with an efficient randomized algorithm also has one that does not use randomness. One of the extensively studied problems under this theme is that of perfect matching. The perfect matching problem has a randomized parallel (NC) algorithm based on the Isolation Lemma of Mulmuley, Vazirani and Vazirani. It is a long-standing open question whether this algorithm can be derandomized. In this work, we give an almost complete derandomization of the Isolation Lemma for perfect matchings in bipartite graphs. This gives us a deterministic parallel (quasi-NC) algorithm for the bipartite perfect matching problem.

Derandomization of the Isolation Lemma means that we deterministically construct a weight assignment so that the minimum weight perfect matching is unique. We present three different ways of this construction with a common main idea.

1. INTRODUCTION

A perfect matching in a graph is a subset of edges such that every vertex has exactly one edge incident on it from the subset. The perfect matching problem, PM, asks whether a given graph contains a perfect matching. The problem has played an important role in the study of algorithms and complexity. The first polynomial-time algorithm for the problem was given by Edmonds [6], which, in fact, motivated him to propose polynomial time as a measure of efficient computation.

Perfect matching was also one of the first problems to be studied from the perspective of parallel algorithms. A parallel algorithm is one where we allow use of polynomially many processors running in parallel. And to consider a parallel algorithm as efficient, we require the running time to be much smaller than a polynomial. In particular, the complexity class NC is defined as the set of problems which can be solved by a parallel computer with polynomially many processors in poly-logarithmic time.

Lovász [16] gave an efficient randomized parallel algorithm for the matching problem, putting it in the complexity class RNC (randomized NC). The essence of his parallel algorithm was a randomized reduction from the matching problem to a determinant computation. A determinant computation in turn reduces to matrix multiplication (see [3]), which is well-known to have parallel algorithms.

One of the central themes in the theory of computation is to understand the power of randomness, i.e., whether all problems with an efficient randomized algorithm also have a deterministic one. The matching problem has been widely studied under this theme. It has been a long-standing open question whether randomness is necessary for a parallel matching algorithm, i.e., whether the problem is in NC.

One can also ask for a parallel algorithm to construct a perfect matching in the graph if one exists (Search-PM). Note that there is a standard search-to-decision reduction for the matching problem, but it does not work in parallel. Karp, Upfal, and Wigderson [15] and later, Mulmuley, Vazirani, and Vazirani [18] gave RNC algorithms for Search-PM. The latter work introduced the celebrated Isolation Lemma and used it to solve Search-PM in RNC. They assign some weights to the edges of the graph, call a weight assignment *isolating* for a graph G if there is a *unique* minimum weight perfect matching in G. Here, the weight of a perfect matching is simply the sum of the weights of the edges in it. Given an isolating weight assignment with polynomially bounded integer weights, they can find the minimum weight perfect matching in G in NC (again via determinant computations).

Note that if we allow exponentially large weights then it is trivial to construct an isolating weight assignment: assign weight 2^i to the i-th edge for $1 \le i \le m$, where m is the number of edges. This, in fact, ensures a different weight for each perfect matching. The challenge, however, is to find an isolating weight assignment with polynomially bounded weights. This is where the Isolation Lemma comes in: it states that if each edge is assigned a random weight from a polynomially bounded range then such a weight assignment is isolating with high probability.

LEMMA 1.1 (ISOLATION LEMMA [18]). *Let $G(V,E)$ be a graph and $w \in \{1,2,\ldots,k|E|\}^E$ be a uniformly random weight assignment on its edges, for some $k \ge 2$. Then w is isolating with probability at least $1 - 1/k$.*

Note that since there can be exponentially many perfect matchings in a graph, under a polynomially bounded

The original version of this paper is entitled "Bipartite Perfect Matching is in quasi-NC" and was published in the Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC), 2016
*Supported by DFG grant TH 472/4

weight assignment, there will definitely be many collisions, i.e., many perfect matchings will get the same weight. The beauty of the Isolation Lemma is that for the minimum weight, there will be a unique perfect matching with high probability.

One way to obtain a deterministic parallel (NC) algorithm for the perfect matching problem is to derandomize this lemma. That is, to *deterministically* construct such a polynomially bounded isolating weight assignment in NC. This has remained a challenging open question.

Derandomization of the Isolation Lemma has been known for some special classes of graphs, for example, planar bipartite graphs [5, 22], strongly chordal graphs [4], and graphs with a small number of perfect matchings [11, 1]. Here, we present an almost complete derandomization of the Isolation Lemma for bipartite graphs. The class of bipartite graphs appears very naturally in the study of perfect matchings. A graph is bipartite if there is a partition of its vertex set into two parts such that each edge connects a vertex from one part to a vertex from the other. Thus, a perfect matching in a bipartite graph matches every vertex in one part to exactly one vertex in the other.

In Section 3, we construct an isolating weight assignment for bipartite graphs with quasi-polynomially large ($n^{O(\log n)}$) weights, where n is the number of vertices in the graph. Note that this is slightly worse than what we would have ideally liked, which is – polynomially bounded weights. Hence, we do not get an NC algorithm. Instead, we get that for bipartite graphs, the problems PM and Search-PM are in quasi-NC^2. That is, the problems can be solved in $O(\log^2 n)$ time using $n^{O(\log n)}$ parallel processors. A more detailed exposition is in the conference version of the paper [8].

THEOREM 1.2. *For bipartite graphs,* PM *and* Search-PM *are in quasi-*NC^2.

The isolation technique

At the heart of our isolation approach is a cycle elimination technique. It is easy to see that if we take a union of two perfect matchings, we get a set of disjoint cycles and singleton edges. Each of these cycles has edges alternating from the two perfect matchings. And thus, cycles play an important role in isolating a perfect matching. Given a weight assignment on the edges, let us define the *circulation* of a (even) cycle C to be the difference of weights between the set of odd-numbered edges and the set of even-numbered edges in cyclic order around C. Clearly, if all the cycles in the union of two perfect matchings have zero circulations, then the two perfect matchings will have the same weight. It turns out that the converse is also true when the two perfect matchings under consideration are of the minimum weight [5]. This observation is the starting point of our cycle elimination technique.

In case of bipartite graphs, this observation can be further generalized. We show that for any weight assignment w on the edges of a bipartite graph, if we consider the union of all the minimum weight perfect matchings, then it has only those cycles which have zero circulation (Lemma 2.2). This means that if we design the weights w such that a particular cycle C has a nonzero circulation, then C does not appear in the union of minimum weight perfect matchings, i.e., at least one of the edges in C does not participate in any minimum weight perfect matchings. This is the way we will be eliminating cycles.

If we eliminate all cycles this way, we will get a unique minimum weight perfect matching, for if there were two minimum weight perfect matchings, their union would contain a cycle with zero circulation. However, it is not possible to ensure nonzero circulations simultaneously for all cycles while keeping the edge weights small (proved in [14]). What we can get is nonzero circulation for any *polynomially large* set of cycles using well-known hashing techniques. In short, we can eliminate any desired set of a small number of cycles at once. With this tool in hand we would like to eliminate all cycles—whose number can be exponentially large—in a small number of rounds.

We present three different ways of achieving this. The first two of these have appeared before in different versions of our paper [8]. The third has not appeared anywhere before.

1. In the first approach, in the i-th round, we eliminate all cycles of length at most 2^{i+1}. Hence, we eliminate all cycles in $\log n$ rounds. Each round is efficient because if a graph does not have any cycles of length at most ℓ, then the number of cycles up to length 2ℓ is polynomially bounded [21, 19].

2. In the second approach, first we eliminate all cycles of length at most $4 \log n$. The bound we have on the number of such cycles is quasi-polynomial in n. Alon, Hoory, and Linial [2] have shown that any graph which does not contain any cycle of length $\leq 4 \log n$ must have average degree at most 2.5, and thus must have at least a constant fraction of nodes with degree 2 or less. From the obtained graph, we ignore degree-1 vertices, and we revise our notion of cycle length to ignore degree-2 vertices in the cycle. We then repeat the procedure of eliminating cycles of length at most $4 \log n$, based on this revised notion of cycle length. The number of such cycles remains quasi-polynomially bounded. In each round the number of degree > 2 nodes decreases by a constant fraction, and thus, after $O(\log n)$ rounds, we are left with all the nodes having degree ≤ 2. At this stage, we have $O(n)$ cycles left, which can be eliminated in one more round.

3. In the third approach, instead of considering the lengths of the cycles, we try to pick as many edge-disjoint cycles as possible and eliminate them. Note that edge-disjointness ensures that we will eliminate at least as many edges as cycles. Erdős and Pósa [7] showed that any graph with m edges and n nodes contains $\Omega(\frac{m-n}{\log(m-n)})$ edge-disjoint cycles. A careful argument shows that in $O(\log^2 n)$ rounds, we eliminate enough edges so that no cycles are left.

As we will see later, the first approach is more efficient than the other two. We still think it is interesting to see different ways of achieving isolation, as they might lead to better ideas for getting isolation with polynomially bounded weights or isolation in other settings. Another interesting point is that our second approach was used in designing a *pseudo-deterministic*-NC algorithm for bipartite matching [10].

Our crucial technical result (Lemma 2.2) about eliminating cycles has a geometric proof that is based on the *perfect matching polytope* of a bipartite graph. In the next section, we define the perfect matching polytope and use its properties to prove the result. Then in Section 3, we formally

describe the weight construction and the three approaches to eliminate all cycles.

2. THE PERFECT MATCHING POLYTOPE

Perfect matchings have an associated polytope, called the *perfect matching polytope*. The perfect matching polytope $\mathrm{PM}(G)$ of a graph $G(V,E)$ is a polytope in the (real) edge space, i.e., $\mathrm{PM}(G) \subseteq \mathbb{R}^E$. For any perfect matching M of G, consider its incidence vector $x^M = (x_e^M)_{e\in E} \in \mathbb{R}^E$ given by

$$x_e^M = \begin{cases} 1, & \text{if } e \in M, \\ 0, & \text{otherwise.} \end{cases}$$

This vector is referred as a *perfect matching point* for any perfect matching M. The *perfect matching polytope* of a graph G is defined to be the convex hull of all its perfect matching points:

$$\mathrm{PM}(G) = \mathrm{conv}\{\, x^M \mid M \text{ is a perfect matching in } G \,\}.$$

The corners of $\mathrm{PM}(G)$ are exactly the perfect matching points of G. For any weight function $w\colon E \to \mathbb{R}$ on the edges of a graph G, consider the linear function

$$w \cdot x = \sum_{e\in E} w(e)\, x_e$$

on $\mathbb{R}^E$. Clearly, for any perfect matching M, we have

$$w(M) = w \cdot x^M.$$

In particular, let M^* be a perfect matching in G of minimum weight. Then

$$w(M^*) = \min\{\, w \cdot x \mid x \in \mathrm{PM}(G) \,\}.$$

The task to isolate a perfect matching can now be rephrased as: *construct a weight function $w\colon E \to \mathbb{N}$ such that $w\cdot x$ has a unique minimum point in* $\mathrm{PM}(G)$.

It is well known that for bipartite graphs, the perfect matching polytope has a simple description in terms of linear inequalities.

LEMMA 2.1 (SEE [17]). *Let $G(V,E)$ be a bipartite graph. The perfect matching polytope $\mathrm{PM}(G)$ is given by*

$$\sum_{e\in\delta(v)} x_e = 1 \qquad v \in V, \tag{1}$$

$$x_e \geq 0 \qquad e \in E, \tag{2}$$

where $\delta(v)$ denotes the set of edges incident on vertex v.

It is easy to see that any integer solution to (1) and (2) is a perfect matching point. The conditions simply say that each vertex should have exactly one edge incident to it. The non-trivial part of the lemma is that the perfect matching points are *all* the corners of the polytope. That is, any real solution to (1) and (2) is a convex combination of some perfect matching points.

For general graphs, the polytope described by (1) and (2) can have corners which are not perfect matchings points. Thus, the description does not capture the perfect matching polytope for general graphs.

Here we depart from the usual definition of "edge space" as a vector space over $\mathbb{Z}/2\mathbb{Z}$.

2.1 Faces of the perfect matching polytope

Since $w \cdot x$ is a linear function, the points in $\mathrm{PM}(G)$ that minimize $w \cdot x$ will form a face of $\mathrm{PM}(G)$. The corners of this minimizing face will all correspond to minimum weight perfect matchings. Any face of a polytope can be obtained by replacing some of the inequalities in its description by equalities. In the case of $\mathrm{PM}(G)$, these inequalities are just the non-negativity constraints (2). Thus, by Lemma 2.1, for any face F of $\mathrm{PM}(G)$, there exists a set $S \subseteq E$ of edges such that F is described by (1) and (2), and $x_e = 0$ for $e \in S$.

Now, for any weight function w, let F_w be the face of the polytope $\mathrm{PM}(G)$ minimizing $w \cdot x$. Let

$$S_w = \{\, e \in E \mid F_w \text{ satisfies } x_e = 0 \,\}.$$

Intuitively, the edges in S_w do not participate in any minimum weight perfect matching with respect to w. Define $E_w = E - S_w$ and $G_w = (V, E_w)$. Hence, G_w is the subgraph of G that contains exactly those edges that participate in some minimum weight perfect matching in G.

2.2 Cycles and their circulations

As mentioned before, cycles play an important role in the context of perfect matchings, and also in our arguments. For an even cycle $C = (e_1, e_2, \ldots, e_p)$, we define its *circulation vector* $\chi^C = (\chi_e^C)_{e\in E}$ by

$$\chi_e^C = \begin{cases} (-1)^j & \text{if } e = e_j, \text{ for some } 1 \leq j \leq p, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the definition actually depends on the starting edge e_1. For our purposes, it does not matter which edge of a cycle is chosen as e_1. Observe that χ^C satisfies

$$\sum_{e\in\delta(v)} \chi_e^C = 0 \qquad v \in V. \tag{3}$$

By equation (3), the circulation vector χ^C lies parallel to the affine subspace of $\mathbb{R}^E$ defined by (1). More general, we define the *circulation of C* with respect to weight w by

$$w \cdot \chi^C = \sum_{j=1}^{p} (-1)^j w(e_j).$$

The following lemma is crucial for our weight construction. It shows that for any cycle C in G_w (i.e., the union of minimum weight perfect matchings w.r.t. w), we have $w\cdot\chi^C = 0$.

LEMMA 2.2. *Let w be a weight function on the edges of a graph G. Let C be a cycle in the subgraph G_w. Then $w \cdot \chi^C = 0$.*

PROOF. Recall that F_w is described by (1) and (2), and $x_e = 0$, for $e \in S_w$. By definition of G_w, the cycle C does not have any edge from S_w. Thus, χ^C also satisfies $x_e = 0$ for all $e \in S_w$, and furthermore, $\sum_{e\in\delta(v)} \chi_e^C = 0$ for all $v \in V$ by equation (3). Therefore χ^C lies parallel to F_w.

By definition, all points $x \in F_w$ have the same weight, i.e., $w \cdot x = c_0$, for some constant c_0. Hence, vector w is orthogonal to the affine span of F_w. We conclude that w is also orthogonal to χ^C, and therefore $w \cdot \chi^C = 0$. $\square$

3. CONSTRUCTING AN ISOLATING WEIGHT ASSIGNMENT

Lemma 2.2 gives us a way to eliminate cycles. Suppose C is a cycle in graph G. If we construct a weight assignment w such that $w \cdot \chi^C \neq 0$ then the cycle C will not be present in G_w, i.e., at least one edge of C will be missing.

We will be applying this technique on a small set of chosen cycles. As mentioned earlier, there are standard ways to construct a weight function which ensures nonzero circulations for any small set of cycles simultaneously, see for example [9].

LEMMA 3.1. *Let $\mathcal{C}$ be any set of s cycles in graph $G(V, E)$ and let $E = \{e_1, e_2, \ldots, e_m\}$. For $j \in \mathbb{N}$, we define weights*

$$w_{(\bmod j)}(e_i) := 2^{i-1} \bmod j, \quad \textit{for } i = 1, 2, \ldots, m.$$

Then there exists a $j \leq ms$ such that

$$w_{(\bmod j)} \cdot \chi^C \neq 0, \quad \textit{for all } C \in \mathcal{C}.$$

Note that the above lemma actually gives a list of weight functions such that for any desired set of cycles, at least one of the weight functions in the list works. Also observe that weight of any edge under any of these functions is bounded by ms. That is, the weights are polynomially bounded as long as the number of cycles is.

The isolating weight assignment is now constructed in rounds. The strategy is to keep eliminating a small number (poly or quasi-poly) of cycles in each round by giving them nonzero circulations. This is repeated until we are left with no cycles. In every round, we add the new weight function to the current weight function on a smaller scale. This is to ensure that the new weights do not interfere significantly with the circulations of cycles which have been already eliminated in earlier rounds.

In more detail, if w_i is the current weight function in the i-th round, then in the next round, we will consider the weight function $w_{i+1} = Nw_i + w'$, for some weight function w' and a large enough number N. The number N is chosen to be larger than $n \cdot \max_e w'(e)$, which ensures that Nw_i gets precedence over w'. The weight function w' is designed to ensure nonzero circulations for a desired set of cycles in G_{w_i}. These cycles will not appear in $G_{w_{i+1}}$. We will keep eliminating cycles in this way until we obtain a w such that G_w has no cycles. Recall that G_w is defined to be the union of minimum weight perfect matchings with respect to w, and thus, contains at least one perfect matching. Since G_w has no cycles, it must have a unique perfect matching, and so, w is isolating for G. Figure 1 shows a graph where an isolating weight assignment is constructed in 3 rounds using our Approach 1, described below.

Bound on the weights.

If we want to assign nonzero circulations to at most s cycles in each round, then the weights are bounded by ms by Lemma 3.1. If there are k such rounds, the bound on the weights becomes $O((nms)^k)$. As we will see later, the quantity $(nms)^k$ will be quasi-polynomially bounded.

Recall that Lemma 3.1 gives a list of ms candidate weight functions such that at least one of them gives nonzero circulations to all the s cycles chosen in a round. We need to try all possible $(ms)^k$ combinations of these candidate functions coming from each round. Our quasi-NC algorithm tries all these combinations in parallel.

Now, the crucial question left in our isolating weight construction is this: how to eliminate all cycles, which are possibly exponentially many, in a small number of rounds, while in each round, only eliminating a small number of cycles. We present three different approaches for this. Each approach will have a different criterion for choosing a small set of cycles which are to be eliminated in a round. The rest of the procedure is common to all three approaches. The following table gives, for each approach, the number of cycles chosen in each round and the number of rounds required to eliminate all cycles. Here we use $m \leq n^2$.

	Number of cycles in each round	Number of rounds	Bound on the weights
Approach 1	n^4	$O(\log n)$	$n^{O(\log n)}$
Approach 2	$n^{O(\log n)}$	$O(\log n)$	$n^{O(\log^2 n)}$
Approach 3	$O(n^2)$	$O(\log^2 n)$	$n^{O(\log^2 n)}$

3.1 Approach 1: Doubling the lengths of the cycles

Here, the idea is to double the length of the cycles which we want to eliminate in each round. There will be $\log n$ rounds. In the i-th round we eliminate all cycles of length at most 2^{i+1}, and thus eliminate all cycles in $\log n$ rounds. The following lemma shows that if we have already eliminated all the cycles of length at most 2^i then the number of cycles of length 2^{i+1} is polynomially bounded, for any i.

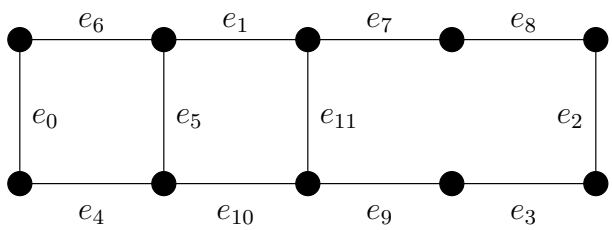
LEMMA 3.2 ([19]). *Let H be a graph with n nodes that has no cycles of length at most r, for some even number $r \geq 4$. Then H has at most n^4 cycles of length at most $2r$.*

PROOF. Let C be a cycle of length $\leq 2r$ in G. We choose 4 vertices u_0, u_1, u_2, u_3 on C which divide it into 4 almost equal parts. We *associate* the tuple (u_0, u_1, u_2, u_3) with C. We claim that C is the only cycle associated with (u_0, u_1, u_2, u_3). For the sake of contradiction, let there be another such cycle C'. Let $p \neq p'$ be paths of C and C', respectively, that connect the same u-nodes. As the four segments of C and C' are of equal length, we have $|p|, |p'| \leq r/2$. Thus p and p' create a cycle of length $\leq r$, which is a contradiction. Hence, a tuple is associated with only one cycle. The number of tuples of four nodes is bounded by n^4, and so is the number of cycles of length $\leq 2r$. $\square$

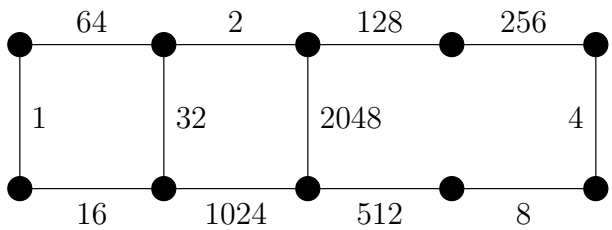
3.2 Approach 2: Eliminating small cycles implies a small average degree

Here the idea is to use a result of Alon, Hoory, & Linial [2] which states that a graph with no small cycles must have many nodes of degree 2. To get an intuitive understanding of this, consider a graph where each node has degree at least 3: do a breadth-first search of the graph starting from an arbitrary node until depth $\log n$. When one reaches a node v via an edge e, there are at least 2 edges incident on v other than e. So, the search tree contains a binary tree of depth $\log n$. The nodes in the tree cannot be all distinct, because otherwise we would have strictly more than $2^{\log n} = n$ nodes. A node that appears twice in the search tree gives us a cycle of length at most $2 \log n$. In other words, if there are no cycles of length at most $2 \log n$, then the graph must have a node with degree 2 or less. Alon, Hoory, & Linial [2] generalize this intuition to show that as the length of the shortest cycle increases, the average degree gets closer to 2.

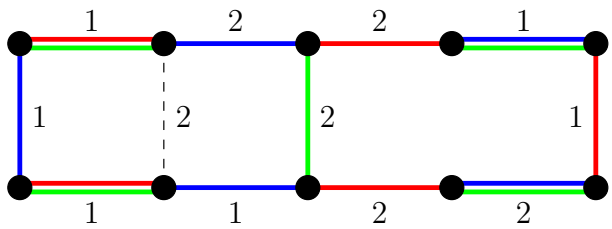
LEMMA 3.3 ([2]). *Let H be a graph with no cycles of length $< 4 \log n - 2$. Then H has average degree < 2.5.*



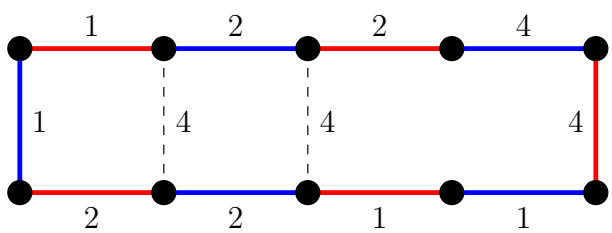
G is a 10-node graph with 12 edges $e_0, \ldots, e_{11}$.



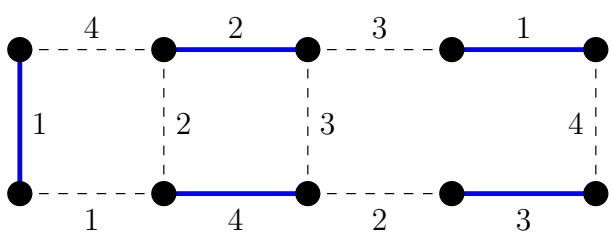
The initial weights are $w(e_i) := 2^i$.



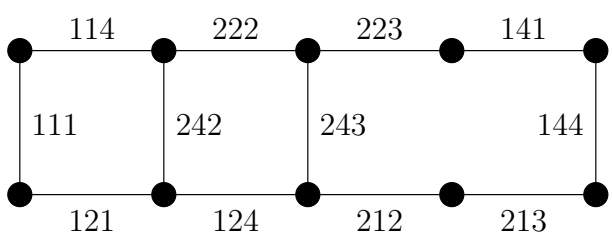
Take w mod 3. The 4-cycles are gone, but only e_5 is removed in the derived graph—the union of 3 min matchings (blue, red, green).



Take w mod 7. The 6-cycles are gone, but only e_{11} is removed in the derived graph—the union of the blue and red matchings (the green does not survive).



Take w mod 5. The 10-cycle is now gone and only the blue matching survives in the derived graph.



Combining the reduced weights gives us a weight function that isolates the blue matching as unique with min weight. Numbers can be interpreted in any radix ≥ 5 in this example.

Figure 1: Iterative construction of an isolating weight assignment on a bipartite graph

Now, in this approach we eliminate all cycles of length less than $4\log n - 2$ in the first round. The Lemma 3.3 implies that after the first round, a constant fraction of the nodes in the graph have degree ≤ 2. In the subsequent rounds, we again eliminate cycles of length less than $4\log n - 2$, except that while counting the length of a cycle, we ignore degree-2 vertices (thus treating paths with degree-2 interior vertices as single edges). It is easy to see that the number of such cycles will be always bounded by $n^{4\log n - 2}$. The following lemma shows that in each round, we reduce the number of degree > 2 nodes by a constant fraction.

LEMMA 3.4. *Let $G(V, E)$ be a graph with n nodes. Let $U \subseteq V$ be its set of degree > 2 nodes. Let $G_1(V, E_1)$ be a subgraph of G such that any cycle in G_1 contains at least $\ell = 4\log n - 2$ nodes from U. Then a constant fraction of the nodes in U have degree ≤ 2 in G_1.*

PROOF. Let $T = V - U$ be the set of nodes of degree ≤ 2 in G. In the following, by *degree of a node* we mean its degree in G_1. First, we delete all degree 1 nodes from G_1. For the degree 2 nodes in T, let us identify them with one of their two neighbors. More specifically, if there is a path consisting of degree 2 nodes of T, we delete all the nodes in the path and connect its two endpoints outside T by an edge.

In summary, we deleted all nodes in T and the nodes of degree 1 in U from G_1. Let G_1' be the resulting graph. The nodes in G_1' are exactly those nodes in U whose degree is > 1 in G_1. Note that the degree of any node in G_1' is the same as in G_1.

By the assumption of the lemma, any cycle in G_1' has length $\geq \ell$. From Lemma 3.3 we have that G_1' has average degree < 2.5. it follows that at least a constant fraction of its nodes have degree 2. This, in turn, means that at least a constant fraction of the nodes in U have degree ≤ 2 in G_1. $\square$

After repeating this procedure for $O(\log n)$ rounds, we will reach a constant number of degree > 2 nodes. That is, there will be only $O(n)$ cycles, which can be eliminated in the next round.

3.3 Approach 3: Eliminating a maximum size set of edge-disjoint cycles

In this approach we do not consider the length of the cycles. Instead, in each round we pick as many edge-disjoint cycles as possible. Recall that eliminating a cycle means that at least one of its edges will not be present in the graph in the next round. Hence when we eliminate a set of edge-disjoint cycles, we will eliminate at least as many edges. Once we remove enough edges, we will be left with no cycles.

Let G be a graph with n vertices and m edges. The number of cycles picked in each round is trivially bounded by m. The non-trivial part is to come up with a lower bound. Erdős and Pósa [7] showed that G has at least $\frac{m-n}{O(\log(m-n))}$ edge-disjoint cycles. We will argue that if we eliminate a maximum size set of edge-disjoint cycles in a round, then the quantity $m - n$ decreases by a significant fraction in every round.

LEMMA 3.5. *Let G be a connected graph with n vertices and m edges. Let $\mathcal{C}$ be a maximum size set of edge-disjoint cycles in G. Let H be any subgraph of G obtained by deleting at least one edge from each cycle in $\mathcal{C}$. Then for any connected component H_1 of H with n_1 vertices and m_1 edges,*

$$m_1 - n_1 \leq (m - n)\left(1 - \frac{1}{O(\log(m-n))}\right).$$

PROOF. Let $|\mathcal{C}| = k$. Define a graph H' to be a subgraph of G such that H is a subgraph of H' and for each cycle in $\mathcal{C}$, *exactly* one edge is missing in H'. Note that H' is still connected, since the cycles in $\mathcal{C}$ are edge-disjoint. The difference between the number of edges and vertices of H' is $m - n - k$.

Since H is obtained by deleting possibly some more edges from H', for any connected component of H, the difference between the number of edges and vertices cannot be larger than $m-n-k$. Now, the lemma follows from the above lower bound of Erdős and Pósa [7] on the number of edge-disjoint cycles. $\square$

Let us repeat the procedure of eliminating a maximum size set of edge-disjoint cycles. It follows from the lemma that after $O(\log^2 n)$ rounds, each component of the obtained graph will have a constant difference between the number of edges and vertices. At this stage, each component will have only constantly many cycles. And so, in one more round we will eliminate all cycles.

A different view on the third approach is by considering the *dimension* of the perfect matching polytope. For a connected bipartite graph, where each of its edges belong to some perfect matching, the perfect matching polytope has dimension $m - n + 1$ [17, Theorem 7.6.2]. Thus, the argument of this approach can also be viewed as decreasing the dimension of the perfect matching polytope by a fraction in each round and eventually reaching dimension zero, i.e., just one perfect matching point.

4. FURTHER GENERALIZATIONS

In a series of follow-up works our isolation approach was generalized to the broader settings of matroid intersection and polytopes with totally unimodular faces, respectively [12, 13]. For a these general settings, the right substitute for the cycle circulation vectors are integer vectors parallel to a face of the polytope. Following our first approach, if one eliminates vectors of length $\leq 2^i$, then there are only polynomially many vectors of length $\leq 2^{i+1}$, in their respective settings (see [12, 13] for details). However, it is not clear if our second and third approaches work in these settings. It will be interesting to find other polytopes for which these isolation approaches work.

In another direction, Svensson and Tarnawaki [20] generalized the isolation result to perfect matchings in *general* graphs. They use the basic framework of our first approach as the starting point. However, the approach does not seem to work as it is and they need several other ideas. In particular, it is not clear if one can show the desired upper bound on the number of vectors parallel to a face. To overcome this issue, they use the technique of *contraction*. They gradually contract larger and larger sets of vertices which enables them to just work with an upper bound on the number of cycles (or closed walks).

Acknowledgments

We would like to thank Manindra Agrawal and Nitin Saxena for their constant encouragement and very helpful dis-

cussions. We thank Arpita Korwar for discussions on some other techniques used in this research, and Jacobo Torán for discussions on the number of shortest cycles.

5. REFERENCES

[1] M. Agrawal, T. M. Hoang, and T. Thierauf. The polynomially bounded perfect matching problem is in NC^2. In *24th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *Lecture Notes in Computer Science*, pages 489–499. Springer Berlin Heidelberg, 2007.

[2] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002.

[3] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147 – 150, 1984.

[4] E. Dahlhaus and M. Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84:79–91, 1998.

[5] S. Datta, R. Kulkarni, and S. Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47:737–757, 2010.

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

[7] P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math. Debrecen*, 9:3–12, 1962.

[8] S. Fenner, R. Gurjar, and T. Thierauf. Bipartite Perfect Matching is in quasi-NC. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC)*, 2016. arXiv:1601.06319; ECCC TR15-177.

[9] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. ACM*, 31(3):538–544, June 1984.

[10] S. Goldwasser and O. Grossman. Bipartite perfect matching in pseudo-deterministic NC. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 87:1–87:13, 2017.

[11] D. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract). In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 166–172, 1987.

[12] R. Gurjar and T. Thierauf. Linear matroid intersection is in quasi-NC. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 821–830, 2017.

[13] R. Gurjar, T. Thierauf, and N. K. Vishnoi. Isolating a vertex via lattices: Polytopes with totally unimodular faces. *CoRR*, abs/1708.02222, 2017.

[14] D. Kane, S. Lovett, and S. Rao. The independence number of the birkhoff polytope graph, and applications to maximally recoverable codes. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 252–259, 2017.

[15] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[16] L. Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory*, pages 565–574, 1979.

[17] L. Lovász and M. D. Plummer. *Matching Theory*. North-Holland mathematics studies. Elsevier Science Ltd, 1986.

[18] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[19] A. Subramanian. A polynomial bound on the number of light cycles in an undirected graph. *Information Processing Letters*, 53(4):173 – 176, 1995.

[20] O. Svensson and J. Tarnawski. The matching problem in general graphs is in quasi-NC. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707, 2017.

[21] C. P. Teo and K. M. Koh. The number of shortest cycles and the chromatic uniqueness of a graph. *Journal of Graph Theory*, 16(1):7–15, 1992.

[22] R. Tewari and N. Vinodchandran. Green's theorem and isolation in planar graphs. *Information and Computation*, 215:1–7, 2012.