A Deterministic Parallel Algorithm 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

A fundamental quest in the 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 doing 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 (Figure 1). 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 [7], 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.

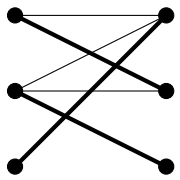


Figure 1: A graph, with the bold edges showing a perfect matching.

Lovász [19] 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 [4]), which is well known to have efficient 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 [18] and later, Mulmuley, Vazirani, and Vazirani [21] 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 $1 \leq i \leq m$, where m is the number of edges. This, in fact, ensures a different weight for each perfect matching. The challenge, however, is to find

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

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.

Note that since there can be exponentially many perfect matchings in a graph, there will definitely be many collisions under a polynomially bounded weight assignment, 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.

LEMMA 1.1 (ISOLATION LEMMA [21]). *Let $G(V, E)$ be a graph, $|E| = m$, and $w \in \{1, 2, \ldots, km\}^E$ be a uniformly random weight assignment on its edges, for some $k \geq 2$. Then w is isolating with probability at least $1 - 1/k$.*

PROOF. Let e be an edge in the graph. We first give an upper bound on the probability that there are two minimum-weight perfect matchings, one containing e and other not containing e. For this, say the weight of every other edge except e has been fixed. Let W be the minimum weight of any perfect matching that avoids e, and let $W' + w(e)$ be the minimum weight of any perfect matching that contains e.

Now, what is the probability that these two minimum weights are equal? Since W and W' are already fixed by the other edges, and $w(e)$ is chosen uniformly and randomly between 1 and km,

$$\Pr\{W = W' + w(e)\} \leq \frac{1}{km}.$$

By the union bound,

$$\Pr\{\exists e \in E \;\; W = W' + w(e)\} \leq \frac{m}{km} = \frac{1}{k}.$$

Now, to finish the proof, observe that there is a unique minimum weight perfect matching if and only if there is no such edge with the above property. □

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 [6, 26], strongly chordal graphs [5], and graphs with a small number of perfect matchings [14, 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 (the graph in Figure 1 is bipartite). 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 [10].

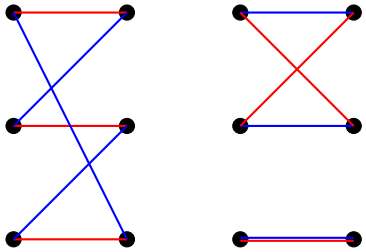


Figure 2: Two perfect matchings, with red and blue edges, respectively. Their union forms a set of disjoint cycles and edges.

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 (see Figure 2). Each of these cycles has even length and has edges alternating from the two perfect matchings. Cycles thus play an important role in isolating a perfect matching. Given a weight assignment on the edges, let us define the *circulation* of an 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 [6]. This observation is the starting point of our cycle elimination technique.

In the 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 2.1). 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. However, it turns out that there are too many cycles in the graph, and it is not possible to ensure nonzero circulations simultaneously for all cycles while keeping the edge weights small (proved in [17]). Instead, what is achievable 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 [10]. 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 [25, 23].

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 resulting graph, we remove all nodes of degree 1, and we contract degree-2 nodes one by one (identifying the two neighbors), until there are no degree-2 nodes left. This creates new small cycles in the graph. We then repeat the procedure of eliminating cycles of length at most $4 \log n$ from the new graph. In each round the number of nodes decreases by a constant fraction. Thus, after $O(\log n)$ rounds, we eliminate all nodes and hence, all cycles.

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 [9] 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* RNC algorithm for bipartite matching [13].

Our crucial technical result (Lemma 2.1) about eliminating cycles has a proof based on linear programming duality. In the next section, we describe a linear programming formulation for bipartite perfect matching and its dual, and then use it to prove our result. Finally in Section 3, we formally describe the weight construction and the three approaches to eliminate all cycles.

2. CYCLE ELIMINATION VIA NONZERO CIRCULATIONS

In this section, we formally describe our main technical tool which enables cycle elimination. Let us first give a formal definition of cycle circulation. For a weight assignment w on the edges of a graph G, the *circulation* $circ_w(C)$ of an even-length cycle $C = (v_1, v_2, \ldots, v_k)$ is defined as the alternating sum of the edge weights around C:

$$circ_w(C) = |w(v_1, v_2) - w(v_2, v_3) + w(v_3, v_4) - \cdots - w(v_k, v_1)|.$$

The definition is independent of the edge we start with because we take the absolute value of the alternating sum.

The following lemma about circulations of cycles gives us a way to eliminate cycles. For a weight assignment w on the edges of a graph G, let G_w be the union of minimum weight perfect matchings, i.e., it is a subgraph of G that has exactly those edges that are present in some minimum weight perfect matching in G.

LEMMA 2.1 (ZERO CIRCULATION). *Let w be a weight function on the edges of a bipartite graph G. Let C be a cycle in the subgraph G_w. Then $circ_w(C) = 0$.*

The following corollary is immediate, which shows how the above lemma can be used to eliminate cycles.

COROLLARY 2.2 (CYCLE ELIMINATION). *Let C be a cycle in a bipartite graph G and w be a weight function on its edges such that $circ_w(C) \neq 0$. Then C is not present in G_w.*

There are several ways to prove Lemma 2.1.

1. In our original paper [10], we presented a proof based on properties of the *perfect matching polytope*. In the argument, the center point of the polytope is slightly moved along cycle C, so that the point stays in the polytope. This implies that the circulation of C must be zero.

2. After the first version of our paper was published, Rao, Shpilka, and Wigderson (see [13, Lemma 2.4]) came up with an alternate proof of Lemma 2.1, similar to ours, but based on *Hall's Theorem* instead of the matching polytope.

3. In a column for SIGACT News [11], we gave a *geometric proof*, where we just argue via vectors being parallel or perpendicular to each other. One might consider this the shortest and most elegant of the proofs.

4. Nevertheless, in Section 2.1 below, we present a fourth proof that we find very nice. It is based on *linear programming duality*.

2.1 Linear programming formulation for perfect matching

The minimum weight perfect matching problem on bipartite graphs has a simple and well-known linear programming (LP) formulation. Let G be a bipartite graph with vertex set V and edge set E. Then the following linear program captures the minimum weight perfect matching problem (see, for example, [20]).

$$\min \sum_{e \in E} w_e x_e$$

$$x_e \geq 0 \quad \forall e \in E \tag{1}$$

$$\sum_{e \in \delta(v)} x_e = 1 \quad \forall v \in V, \tag{2}$$

where $\delta(v)$ denotes the set of edges incident on a vertex v. The linear program has one variable x_e for each edge in the graph. Intuitively, $x_e = 1$ represents that the edge e is present in the perfect matching and $x_e = 0$ represents that e is not in the perfect matching. Then, Equation (2) is simply

saying that a perfect matching contains exactly one edge from the set of edges incident on a particular vertex. The objective function asks to minimize the sum of the weights of the edges present in a perfect matching, that is, the weight of a perfect matching.

An important point to note here is that the above LP formulation works only for bipartite graphs, and this is the reason our proof does not work for general graphs.

From the standard theory of LP duality, the following is the dual linear program for minimum weight perfect matching. Since in the primal LP above we had an equality constraint for each vertex, here we have a variable for each vertex.

$$\max \sum_{u \in V} y_u$$
$$y_u + y_v \leq w_e \quad \forall e = (u, v) \in E. \tag{3}$$

Note that the dual variables do not have any non-negativity constraint, since the primal constraints are equalities.

It follows from strong LP duality that the optimal values of these two linear programs are equal. This will be crucial for the proof of Lemma 2.1.

PROOF OF LEMMA 2.1. Let $e = (u, v)$ be any edge participating in a minimum weight perfect matching, in other words, e is an edge in G_w. Let $y \in \mathbb{R}^V$ be a dual optimal solution. We claim that the dual constraint (3) corresponding to e is tight, i.e.,

$$y_u + y_v = w_e. \tag{4}$$

This can be seen as follows: for any minimum weight perfect matching M, its weight by definition is the primal optimal value, and thus, by strong duality must be equal to the dual optimal value. That is,

$$w(M) = \sum_{v \in V} y_v.$$

Note that a sum over all vertices is the same as a sum over the end points of all the edges in a perfect matching. Thus,

$$\sum_{e \in M} w(e) = \sum_{e=(u,v) \in M} (y_u + y_v). \tag{5}$$

Together with (3), equation (5) implies equation (4).

Now, let $C = (u_1, u_2, \ldots, u_{2k})$ be a cycle in G_w. Since each edge in C is part of some minimum weight perfect matching, by (4), all the edges of C are tight w.r.t. the dual optimal solution y. Hence,

$$\begin{aligned} circ_w(C) &= w(u_1, u_2) - w(u_2, u_3) + \cdots - w(u_{2k}, u_1) \\ &= (y_{u_1} + y_{u_2}) - (y_{u_2} + y_{u_3}) + \cdots - (y_{u_{2k}} + y_{u_1}) \\ &= 0. \end{aligned}$$

Hence, every cycle in G_w has zero circulation. □

3. CONSTRUCTING AN ISOLATING WEIGHT ASSIGNMENT

Corollary 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 $circ_w(C) \neq 0$ 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 [12].

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*

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

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

$$circ_{w_{(\mathrm{mod}\, j)}}(C) \neq 0, \quad \textit{for all } C \in \mathcal{C}.$$

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 the 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.

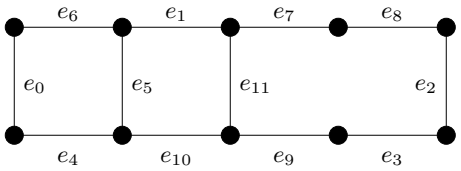
In more detail, if w_i is the current weight function in the i-th round, then in the next round, we will consider the weight function $w_{i+1} = Nw_i + w'$, for some weight function w' and a large enough number N. The number N is chosen to be larger than $n \cdot \max_e w'(e)$, which ensures that Nw_i 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 3 shows a graph where an isolating weight assignment is constructed in 3 rounds using our Approach 1, described below.

Bound on the weights.

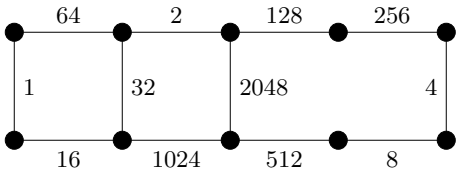
If we want to assign nonzero circulations to at most s cycles in each round, then the weights are bounded by ms by Lemma 3.1. If there are k such rounds, the bound on the weights becomes $O((nms)^k)$. As we will see later, the quantity $(nms)^k$ will be quasi-polynomially bounded.

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.

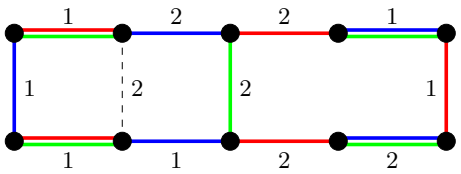
Now, the crucial question left in our isolating weight construction is this: how to eliminate all cycles, which are possibly exponentially many, in a small number of rounds, while only eliminating a small number of cycles in each round. We present three different approaches for this. Each approach will 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



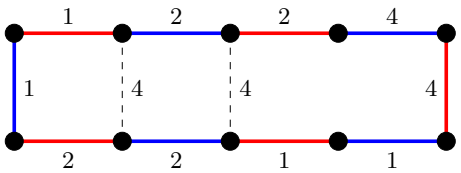
G is a 10-node graph with 12 edges $e_0, \ldots, e_{11}$.



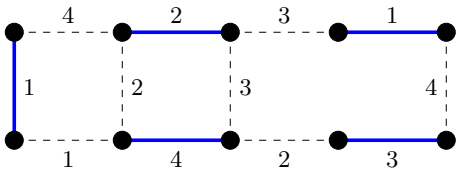
The initial weights are $w(e_i) := 2^i$.



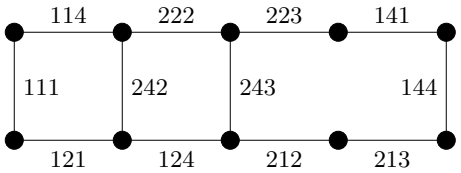
Take $w_{(\mathrm{mod}\,3)}$. The 4-cycles are gone, but only e_5 is removed in the derived graph—the union of 3 min matchings (blue, red, green).



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



Take $w_{(\mathrm{mod}\,5)}$. 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 3: Iterative construction of an isolating weight assignment on a bipartite graph

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 that 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 we 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 ([23]). *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 $\leq 2r$. □

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 $< 4 \log n$. Then H has average degree < 2.5.*

In this approach, we start by eliminating all cycles in graph G of length $\leq 4 \log n$. It is easy to see that the number of such cycles will be bounded by $n^{4 \log n}$. Lemma 3.3 implies that after this, a constant fraction of the nodes in G have degree ≤ 2.

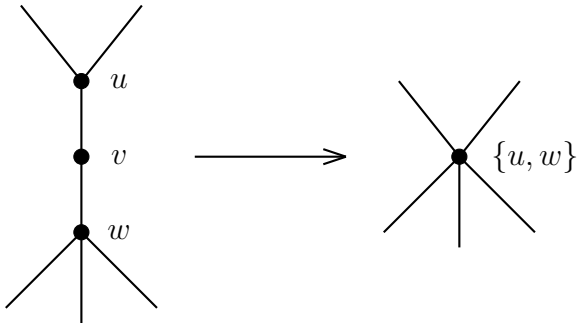


Figure 4: Deleting a degree-2 node v and identifying its two neighbors u and w — an operation which preserves perfect matchings and cycles.

Having many nodes of degree ≤ 2 is quite useful when we are interested in perfect matchings because they provide a way to shrink the graph while preserving perfect matchings.

1. Let v be node of degree 1 in G and u be the unique neighbor of v. Recall that our graph after every round is always a union of perfect matchings. Therefore u has degree 1 as well. Hence, we can simply delete u and v from G.

2. Let v be a node of degree 2 in G with its neighbors u and w. Now, construct a new graph G' from G by deleting v and identifying u and w to get a single node $\{u, w\}$ (see Figure 4). We refer to this operation as *collapsing* the node v. Observe that perfect matchings of G and G' are in one-to-one correspondence.

Note also that any cycle in G appears in G' with the degree-2 nodes cut out, i.e., cycles get shorter in G'.

To further proceed in this approach, we first collapse all degree-2 nodes in G (one by one) and delete all degree-1 nodes. Let G_0 be the resulting graph. Since there were a constant fraction of nodes with degree ≤ 2 in G, the number of nodes in G_0 decreases by a constant fraction. Note also that all nodes in G_0 have degree ≥ 3. By Lemma 3.3, graph G_0 again has small cycles of length $\leq 4 \log n$. Now, we can repeat the procedure of eliminating all cycles of length $\leq 4 \log n$ with G_0.

In every round, the number of nodes in the graph decreases by a constant fraction. Thus, in $O(\log n)$ rounds, we eliminate all cycles and reach the empty graph. One can easily obtain a unique perfect matching in the original graph G, by reversing all the degree-1 deletions and degree-2 collapses.

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

In this approach we do not consider the lengths 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 [9] 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 3.4. *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 $|\mathcal{C}| = k$. Let H' be any 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 [9] on the number of edge-disjoint cycles. □

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 $m-n+1$ [20, 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 DEVELOPMENTS

After years of inactivity, our result inspired a series of follow-up works on parallel algorithms for perfect matching and the Isolation Lemma. In one direction, our isolation approach was generalized to the broader settings of matroid intersection and polytopes with totally unimodular faces, respectively [15, 16]. For these general settings, the right substitute for cycles are integer vectors parallel to a face of the associated 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 [15, 16] for details). It is not clear, however, if our second and third approaches work in these settings.

In another direction, Svensson and Tarnawski [24] generalized the isolation result to perfect matchings in *general graphs*. They use the basic framework of our first approach as the starting point, but they need to combine the technique of eliminating cycles with a second parameter (*contractability*) to control the progress in subsequent rounds.

The techniques developed by us and by Svensson and Tarnawski were used by Anari and Vazirani [3] to compute a perfect matching in *planar graphs* in NC (see also [22]), which also was a long-standing open problem. They show that the sets to be contracted, odd sets of vertices that form a tight cut in the LP-constraints, can be computed in NC. In a subsequent work, the NC algorithm was further generalized to one-crossing-minor-free graphs [8].

The Isolation Lemma has applications in many different settings—in particular, in design of randomized algorithms. The main open question that remains is for what other settings can one derandomize the Isolation Lemma. We conjecture that our isolation approach works for any family of sets whose corresponding polytopes are described by 0/1 constraints.

Acknowledgments

We would like to thank Manindra Agrawal and Nitin Saxena for their constant encouragement and very helpful discussions. We thank Arpita Korwar for discussions on some other techniques used in this research, Jacobo Torán for discussions on the number of shortest cycles, and Nisheeth Vishnoi for helpful comments.

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] N. Anari and V. V. Vazirani. Planar graph perfect matching is in NC. Technical Report arXiv:1709.07822, CoRR, 2017.

[4] 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.

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

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

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

[8] D. Eppstein and V. V. Vazirani. NC algorithms for perfect matching and maximum flow in one-crossing-minor-free graphs. Technical Report arXiv:1802.00084, CoRR, 2018.

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

[10] 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.

[11] S. Fenner, R. Gurjar, and T. Thierauf. Guest column: Parallel algorithms for perfect matching. *SIGACT News*, 48(1):102–109, Mar. 2017.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

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

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

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

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

[22] P. Sankowski. Planar perfect matching is in NC. Technical Report arXiv:1709.07869, CoRR, 2017.

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

[24] 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.

[25] 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.

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