Bipartite Perfect Matching is in quasi-NC

Stephen Fenner\textsuperscript{1}, Rohit Gurjar\textsuperscript{*2}, and Thomas Thierauf\textsuperscript{*2}

\textsuperscript{1}University of South Carolina
\textsuperscript{2}Aalen University, Germany

October 5, 2016

Abstract

We show that the bipartite perfect matching problem is in quasi-NC\textsuperscript{2}. That is, it has uniform circuits of quasi-polynomial size $n^{O(\log n)}$, and $O((\log^2 n) \text{ depth})$. Previously, only an exponential upper bound was known on the size of such circuits with poly-logarithmic depth.

We obtain our result by an almost complete derandomization of the famous Isolation Lemma when applied to yield an efficient randomized parallel algorithm for the bipartite perfect matching problem.

1 Introduction

The perfect matching problem has been widely studied in complexity theory. It has been of particular interest in the study of derandomization and parallelization. The perfect matching problem, PM, asks whether a given graph contains a perfect matching.

The problem has a polynomial-time algorithm due to Edmonds [Edm65]. However, its parallel complexity is still not completely resolved as of today. The problem can be solved by randomized efficient parallel algorithms due to Lovász [Lov79], i.e., it is in RNC, but it is not known whether randomness is necessary, i.e., whether it is in NC. The class NC represents the problems which have efficient parallel algorithms, i.e., they have uniform circuits of polynomial size and poly-logarithmic depth. For the perfect matching problem, nothing better than an exponential-size circuit was known, in the case of poly-logarithmic depth.

The construction version of the problem, Search-PM, asks to construct a perfect matching in a graph if one exists. It is in RNC due to Karp et al. [KUW86] and Mulmuley et al. [MVV87]. The latter algorithm applies the celebrated Isolation Lemma. Both algorithms work with a weight assignment on the edges of the graph. A weight assignment is called isolating for a graph $G$ if the minimum weight perfect matching in $G$ is unique, if one exists. Mulmuley et al. [MVV87] showed that given an isolating weight assignment with polynomially bounded integer weights for a graph $G$, a perfect matching in $G$ can be constructed in NC. To get an isolating weight assignment they use randomization. This is where the Isolation Lemma comes into play.

\textsuperscript{*}Supported by DFG grant TH 472/4
Lemma 1.1 (Isolation Lemma [MVV87]). For a graph $G(V,E)$, let $w$ be a random weight assignment, where edges are assigned weights chosen uniformly and independently at random from $\{1, 2, \ldots, 2|E|\}$. Then $w$ is isolating with probability $\geq 1/2$.

Derandomizing this lemma means to construct such a weight assignment deterministically in NC. This remains a challenging open question. A general version of this lemma, which considers a family of sets and requires a unique minimum weight set, has also been studied. The general version is related to the polynomial identity testing problem and circuit lower bounds [AM08].

The Isolation Lemma has been derandomized for some special classes of graphs, e.g., planar bipartite graphs [DKR10, TV12], strongly chordal graphs [DK98], and graphs with a small number of perfect matchings [GKS7, AHT07]. In this work, we make a significant step towards the derandomization of the Isolation Lemma for bipartite graphs. In Section 3 we construct an isolating weight assignment for these graphs with quasi-polynomially large weights. Previously, the only known deterministic construction was the trivial one that used exponentially large weights. As a consequence we get that for bipartite graphs, $PM$ is in quasi-$NC^2$ and $Search\cdot PM$ is in quasi-$NC^3$. In particular, $PM$ can be solved by uniform Boolean circuits of depth $O(\log^2 n)$, and size $n^{O(\log n)}$ for graphs with $n$ nodes. The depth goes to $O(\log^3 n)$ for $Search\cdot PM$. Note that the size is just one log-$n$-exponent away from polynomial size.

In previous versions of this paper, we had mistakenly claimed that $Search\cdot PM$ too is in quasi-$NC^2$, which was incorrect.

Our result also gives an RNC-algorithm for $PM$ in bipartite graphs which uses very few random bits. The original RNC-algorithm of Lovász [Lov79] uses $O(m \log n)$ random bits. This has been improved by Chari, Rohatgi, and Srinivasan [CRS95] to $O(n \log(m/n))$ random bits. They actually construct an isolating weight assignment using these many random bits.

To the best of our knowledge, the best upper bound today on the number of random bits is $(n + n \log(m/n))$ by Chen and Kao [CK97], that is, the improvement to [CRS95] was only in the multiplicative factor. In Section 4 we achieve an exponential step down to $O(\log^2 n)$ random bits. Note that this is close to a complete derandomization which would be achieved when the number of random bits comes down to $O(\log n)$. This improves an earlier version of this work, where we had an RNC-algorithm with $O(\log^3 n)$ random bits.

Based on the first version of our paper, Goldwasser and Grossman [GG15] observed that one can get an RNC-algorithm for $Search\cdot PM$ which uses $O(\log^4 n)$ random bits. With our improved decision algorithm, we obtain now an RNC-algorithm for $Search\cdot PM$ which uses only $O(\log^2 n)$ random bits.

In Section 5 we show that our approach also gives an alternate NC-algorithm for $Search\cdot PM$ in bipartite planar graphs. This case already has known NC-algorithms [MN95, MV00, DKR10]. Our algorithm is in NC$^3$, while the previous best known upper bound is already NC$^2$ [MN95, DKR10].

We give a short outline of the main ideas of our approach. For any two perfect matchings of a graph $G$, the edges where they differ form disjoint cycles. For a cycle $C$, its circulation is defined to be the difference of weights of two perfect matchings which differ exactly on the edges of $C$. Datta et al. [DKR10] showed that a weight assignment which ensures nonzero circulation for every cycle is isolating. It is not clear if there exists such a weight assignment with small weights. Instead, we use a weight function that has nonzero circulations only for small cycles. Then, we consider the subgraph $G'$ of $G$ which is the union of minimum
weight perfect matchings in \( G \). In the bipartite case, graph \( G' \) is significantly smaller than the original graph \( G \). In particular, we show that \( G' \) does not contain any cycle with a nonzero circulation. This means that \( G' \) does not contain any small cycles.

Next, we show that for a graph which has no cycles of length \(< r\), the number of cycles \(< 2^r\) is polynomially bounded. This motivates the following strategy which works in \( \log n \) rounds: in the \( i \)-th round, assign weights which ensure nonzero circulations for all cycles with length \(< 2^i\). Since the graph obtained after \((i-1)\)-th rounds has no cycles of length \(< 2^{i-1}\), the number of cycles of length \(< 2^i\) is small. In \( \log n \) rounds, we get a unique minimum weight perfect matching.

Recently, Kallampally and Tewari [KT16] used similar techniques to construct an isolating weight assignment for paths in a directed graph, where they study the NL versus UL question.

2 Preliminaries

2.1 Matchings and Complexity

By \( G(V,E) \) we denote a graph with vertex set \( V \) of size \(|V| = n\) and edge set \( E \) of size \(|E| = m\). We consider only undirected graphs in this paper. For a vertex \( v \in V \), let \( \delta(v) \subseteq E \) denote the edges incident on \( v \). A graph is bipartite if there exists a partition \( V = L \cup R \) of the vertices such that all edges are between vertices of \( L \) and \( R \).

In a graph \( G(V,E) \), a matching \( M \subseteq E \) is a subset of edges with no two edges sharing an endpoint. A matching which covers every vertex is called a perfect matching. For any weight assignment \( w : E \to \mathbb{Z} \) on the edges of a graph, the weight of a matching \( M \) is defined to be the sum of weights of all the edges in \( M \), i.e., \( w(M) = \sum_{e \in M} w(e) \).

A weight function \( w \) is called isolating for \( G \), if there is a unique perfect matching of minimum weight in \( G \).

A graph \( G \) is matching-covered if each edge in \( G \) participates in some perfect matching. In the literature, matching-covered is also called 1-extendable and these notions require \( G \) to be connected. Note: in this paper, we use matching-covered also for non-connected graphs!

The perfect matching problem \( \text{PM} \) is to decide whether a given graph has a perfect matching. Its construction version \( \text{Search-PM} \) is to compute a perfect matching of a given graph, or to determine that no perfect matching exists. A bipartite graph \( G(V,E) \) with vertex partition \( V = L \cup R \) can have a perfect matching only when \(|L| = |R| = n/2\). Hence, when we consider bipartite graphs, we will always assume such a partition.

Analogous to \( \text{NC}^k \), Barrington [Bar92] defined the class quasi-\( \text{NC}^k \) as the class of problems which have uniform circuits of quasi-polynomial size \( 2^{O(1)n} \) and poly-logarithmic depth \( O(\log^k n) \). Here, uniformity means that local queries about the circuit can be answered in poly-logarithmic time (see [Bar92] for details). The class quasi-\( \text{NC} \) is the union of classes quasi-\( \text{NC}^k \), over all \( k \geq 0 \).

2.2 An RNC Algorithm for Search-PM

Let us first recall the RNC algorithm of Mulmuley, Vazirani & Vazirani [MVV87] for the construction of a perfect matching (Search-PM). Though the algorithm works for any graph, we will only consider bipartite graphs here.

Let \( G \) be a bipartite graph with vertex partitions \( L = \{u_1, u_2, \ldots, u_{n/2}\} \) and \( R = \{v_1, v_2, \ldots, v_{n/2}\} \), and weight function \( w \). Consider the following \( n/2 \times n/2 \) matrix \( A \) as-
associated with $G$, 

$$A(i, j) = \begin{cases} 
2^w(e), & \text{if } e = (u_i, v_j) \in E, \\
0, & \text{otherwise.}
\end{cases}$$

The algorithm in [MVV87] computes the determinant of $A$. An easy argument shows that this determinant is the signed sum over all perfect matchings in $G$:

$$\det(A) = \sum_{\pi \in S_{n/2}} \sgn(\pi) \prod_{i=1}^{n/2} A(i, \pi(i))$$

(1)

$$= \sum_{M \text{ pm in } G} \sgn(M) 2^{w(M)}$$

(2)

Equation (2) holds because the product $\prod_{i=1}^{n/2} A(i, \pi(i))$ is nonzero if and only if the permutation $\pi$ corresponds to a perfect matching. Here $\sgn(M)$ is the sign of the corresponding permutation. If the graph $G$ does not have a perfect matching, then clearly $\det(A) = 0$. However, even when the graph has perfect matchings, there can be cancellations due to $\sgn(M)$, and $\det(A)$ may become zero. To avoid such cancellations, one needs to design the weight function $w$ cleverly. In particular, if $G$ has a perfect matching and $w$ is isolating, then $\det(A) \neq 0$. This is because the term $2^{w(M)}$ corresponding to the minimum weight perfect matching cannot be canceled with other terms, which are strictly higher powers of 2.

Given an isolating weight assignment for $G$, one can easily construct the minimum weight perfect matching in NC. Let $M^*$ be the unique minimum weight perfect matching in $G$. First we find out $w(M^*)$ by looking at the highest power of 2 dividing $\det(A)$. Then for every edge $e \in E$, compute the determinant of the matrix $A_e$ associated with $G - e$. If the highest power of 2 that divides $\det(A_e)$ is larger than $2^{w(M^*)}$, then $e \in M^*$. Doing this in parallel for each edge, we can find all the edges in $M^*$.

As already explained in the introduction, the Isolation Lemma delivers the isolating weight assignment with high probability. Moreover, the weights chosen by the Isolation Lemma are polynomially bounded. Therefore, the entries in matrix $A$ have polynomially many bits. This suffices to compute the determinant in NC² [Ber84]. Hence, also the construction is in NC². Put together, this yields an RNC-algorithm for Search-PM.

2.3 The Matching Polytope

Matchings are also one of the well-studied objects in polyhedral combinatorics. Matchings have an associated polytope, called the perfect matching polytope. We use some properties of this polytope to construct an isolating weight assignment. The perfect matching polytope also forms the basis of one of the NC-algorithms for bipartite planar matching [MV00].

The perfect matching polytope $\text{PM}(G)$ of a graph $G(V,E)$ with $|E| = m$ edges is a polytope in the edge space, i.e., $\text{PM}(G) \subseteq \mathbb{R}^m$. For any perfect matching $M$ of $G$, consider its incidence vector $x^M = (x^M_e)_{e \in E} \in \mathbb{R}^m$ given by

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

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

Any weight function \( w : E \to \mathbb{R} \) on the edges of a graph \( G \) can be naturally extended to \( \mathbb{R}^m \) as follows: for any \( x = (x_e)_{e \in E} \in \mathbb{R}^m \), define
\[
w(x) = \sum_{e \in E} w(e) x_e.
\]

Clearly, for any perfect matching \( M \), we have \( w(M) = w(x^M) \). In particular, let \( M^* \) be a perfect matching in \( G \) of minimum weight. Then
\[
w(M^*) = \min\{ w(x) \mid x \in \text{PM}(G) \}.
\]

The following lemma gives a simple description of the perfect matching polytope of a bipartite graph \( G \) which is well known, see for example [LP86]. Recall that \( \delta(v) \) denotes the set of edges incident on vertex \( v \).

**Lemma 2.1.** Let \( G(V,E) \) be a bipartite graph and \( x = (x_e)_{e \in E} \in \mathbb{R}^m \). Then \( x \in \text{PM}(G) \) if and only if
\[
\sum_{e \in \delta(v)} x_e = 1 \quad v \in V, \quad (3)
\]
\[
x_e \geq 0 \quad e \in E, \quad (4)
\]

It is easy to see that any perfect matching point will satisfy these two conditions. In fact, all perfect matching points are vertices of this polytope. The non-trivial part is to show that any point satisfying these two conditions is in the perfect matching polytope [LP86, Chapter 7]. For general graphs, the polytope described by (3) and (4) can have vertices which are not perfect matchings. Thus, the description does not capture the perfect matching polytope for general graphs.

### 2.4 Nice Cycles and Circulation

Let \( G(V,E) \) be a graph with a perfect matching. A cycle \( C \) in \( G \) is a *nice cycle*, if the subgraph \( G - C \) still has a perfect matching. In other words, a nice cycle can be obtained from the symmetric difference of two perfect matchings. Note that a nice cycle is always an even cycle.

For a weight assignment \( w \) on the edges, the *circulation* \( c_w(C) \) of an even length cycle \( C = (v_1, v_2, \ldots, v_k) \) is defined as the alternating sum of the edge weights of \( C \),
\[
c_w(C) = |w(v_1, v_2) - w(v_2, v_3) + w(v_3, v_4) - \cdots - w(v_k, v_1)|.
\]

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

The circulation of nice cycles was one crucial ingredient of the isolation in bipartite planar graphs given by Datta et al. [DKR10].

**Lemma 2.2 ([DKR10]).** Let \( G \) be a graph with a perfect matching, and let \( w \) be a weight function such that all nice cycles in \( G \) have nonzero circulation. Then the minimum perfect matching is unique. That is, \( w \) is isolating.
Proof. Assume that there are two perfect matchings \( M_1, M_2 \) of minimum weight in \( G \). Their symmetric difference \( M_1 \triangle M_2 \) consists of nice cycles. Let \( C \) be a nice cycle in \( M_1 \triangle M_2 \). By the assumption of the lemma, we have \( c_w(C) \neq 0 \). Hence, one can decrease the weight of either \( M_1 \) or \( M_2 \) by altering it on \( C \). As \( M_1 \) and \( M_2 \) are minimal, we get a contradiction. \( \square \)

We will construct an isolating weight function for bipartite graphs. However, our weight function will not necessarily have nonzero circulation on all nice cycles. We start out with a weight assignment which ensures nonzero circulations for a small set of cycles in a black-box way, i.e., without being able to compute the set efficiently. The following lemma describes a standard trick for this (see [FKS84, CRS95]).

Lemma 2.3. Let \( G \) be a graph with \( n \) nodes. Then, for any number \( s \), one can construct a set of \( O(n^2s) \) weight functions with weights bounded by \( O(n^2s) \), such that for any set of \( s \) cycles, one of the weight functions gives nonzero circulation to each of the \( s \) cycles.

Proof. Let us first assign exponentially large weights. Let \( e_1, e_2, \ldots, e_m \) be some enumeration of the edges of \( G \). Define a weight function \( w \) by \( w(e_i) = 2^{i-1} \), for \( i = 1, 2, \ldots, m \). Then clearly every cycle has a nonzero circulation. However, we want to achieve this with small weights.

We consider the weight assignment modulo small numbers, i.e., the weight functions \( \{ w \mod j \mid 2 \leq j \leq t \} \) for some appropriately chosen \( t \). We want to show that for any fixed set of \( s \) cycles \( \{C_1, C_2, \ldots, C_s\} \), one of these assignments will work, when \( t \) is chosen large enough. That is, we want

\[
\exists j \leq t \quad \forall i \leq s : c_{w \mod j}(C_i) \neq 0.
\]

This will be true provided

\[
\exists j \leq t : \prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \pmod{j}.
\]

In other words,

\[
\text{lcm}(2, 3, \ldots, t) \nmid \prod_{i=1}^{s} c_w(C_i).
\]

This can be achieved by setting \( \text{lcm}(2, 3, \ldots, t) > \prod_{i=1}^{s} c_w(C_i) \). The product \( \prod_{i=1}^{s} c_w(C_i) \) is upper bounded by \( 2^{n^2s} \). Furthermore, we have \( \text{lcm}(2, 3, \ldots, t) > 2^t \) for \( t \geq 7 \) (see [Nai82]). Thus, choosing \( t = n^2s \) suffices. Clearly, the weights are bounded by \( t = n^2s \).

3 Isolation in Bipartite Graphs

In this section we present our main result, an almost efficient parallel algorithm for the perfect matching problem.

Theorem 3.1. For bipartite graphs, PM and Search-PM are in quasi-NC\(^2\) and quasi-NC\(^3\), respectively.
Let $G(V,E)$ be the given bipartite graph. In the following discussion, we will assume that $G$ has perfect matchings. Our major challenge is to isolate one of the perfect matchings in $G$ by an appropriate weight function. As we will see later, if $G$ does not have any perfect matchings, then our algorithm will detect this.

Our starting point is Lemma 2.2 which requires nonzero circulations for all nice cycles. Recall from Section 2.2 that the construction algorithm requires the weights to be polynomially bounded. As the number of nice cycles can be exponential in the number of nodes, even the existence of such a weight assignment is not immediately clear. Nonetheless, Datta et al. [DKR10] give a construction of such a weight assignment for bipartite planar graphs. For general bipartite graphs, this is still an open question.

We start with a weight function which gives nonzero circulation only to small cycles (not necessarily nice cycles), say of length 4. There are $\leq n^4$ many such cycles, i.e., polynomially many. Lemma 2.3 describes a way to find such weights. The cost of this weight assignment is proportional to the number of small cycles. Further, it is a black-box construction in the sense that one does not need to know the set of cycles. It just gives a set of weight assignments such that at least one of them has the desired property.

3.1 The Union of Minimum Weight Perfect Matchings

Let us assign a weight function for bipartite graph $G$ which gives nonzero circulation to a small set of cycles in $G$. Consider a new graph $G_1$ obtained by the union of minimum weight perfect matchings in $G$. Our hope is that $G_1$ is significantly smaller than the original graph $G$. Note that it is not clear if one can efficiently construct $G_1$ from $G$. This is because the determinant of the bi-adjacency matrix with weights in equation (1) from Section 2.2 can still be zero. As we will see, we do not need to construct $G_1$; it is just used in the argument. Our final weight assignment will be completely black-box in this sense.

Our next lemma is the main reason why our technique is restricted to bipartite graphs. It shows that the graph $G_1$ constructed from the minimum weight perfect matchings in $G$ contains no other perfect matchings than these. This lemma was independently proven by Vishwanathan [Vis01]. In Figure 1, we give an example showing that this does not hold in general graphs.

**Lemma 3.2.** Let $G(V,E)$ be a bipartite graph with weight function $w$. Let $E_1$ be the union of all minimum weight perfect matchings in $G$. Then every perfect matching in the graph $G_1(V,E_1)$ has the same weight – the minimum weight of any perfect matching in $G$.

**Proof.** Consider the description of the perfect matching polytope $PM(G)$ given in Lemma 2.1. As the weight function is linear, the points of minimum weight form a face $F$ in $PM(G)$. A face of a polytope is obtained by replacing some of the inequalities in the polytope description by equalities. The only inequalities of the perfect matching polytope for bipartite graphs are of the form (I): $x_e \geq 0$. Thus, for the face $F$ there exists a set $S \subseteq E$ such that for any $x = (x_e)_{e \in E}$, we have $x \in F$ if and only if

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

$$x_e \geq 0 \quad e \in E - S,$$

$$x_e = 0 \quad e \in S.$$


For $e \in S$, the equality $x_e = 0$ means that edge $e$ does not participate in any minimal perfect matching of $G$. Therefore, we have $G_1 = G - S$. The perfect matching polytope $PM(G - S)$ is given by (5) and (6). Hence, face $F$ is the perfect matching polytope of graph $G - S$, i.e., $F = PM(G - S) = PM(G_1)$. Since all points in $F$ have the same weight, it follows that all perfect matchings in $G_1$ have the same weight. Therefore, the minimum weight perfect matchings of $G$ are precisely the perfect matchings in $G_1$.

\[ \]

Figure 1: A non-bipartite weighted graph where every edge is contained in a minimum perfect matching of weight 1. However, the graph also has a perfect matching of weight 3. That is, Lemma 3.2 does not hold for non-bipartite graphs.

Next, we show that $G_1$ is significantly smaller than $G$. As all the perfect matchings in $G_1$ have the same weight, every nice cycle in $G_1$ has zero circulation. However, we need a stronger statement, which the following lemma proves: every cycle in $G_1$ has zero circulation. It is true because one can show that all cycles are in the linear span of nice cycles. In the lemma below, we give a proof using the perfect matching polytope. Note that $G_1$ is matching-covered, as it is a union of perfect matchings.

**Lemma 3.3.** Let $H(V, E)$ be a matching-covered bipartite graph. Let $w$ be a weight function such that every perfect matching in $H$ has the same weight. Then for each cycle $C$ in $H$, we have $c_w(C) = 0$.

**Proof.** Let the weight of each perfect matching be $q$. Any point $\mathbf{x}$ in the perfect matching polytope $PM(H)$ is a convex combination of perfect matchings. Therefore, we also have $w(\mathbf{x}) = q$.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ be all the perfect matching points of $H$, i.e., the corners of $PM(H)$. Consider the average point $\mathbf{x} \in PM(H)$ of the matching points,

\[
\mathbf{x} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_t}{t}.
\]

Since each edge participates in a perfect matching, every coordinate of $\mathbf{x} = (x_e)_{e \in E}$ is nonzero, in fact $x_e \geq 1/t$. Now, consider a cycle $C$ in $H$ with edges $(e_1, e_2, \ldots, e_p)$ in cyclic order. We show that when we move from point $\mathbf{x}$ along the cycle $C$, we remain inside $PM(H)$. This technique of moving along the cycle has been used by Mahajan and Varadarajan [MV00]. To elaborate, we define a new point $\mathbf{y} = (y_e)_{e \in E}$ as follows. For $e \in E$ and $\varepsilon = 1/t$, let

\[
y_e = \begin{cases} 
x_e + (-1)^i \varepsilon, & \text{if } e = e_i, \text{ for some } 1 \leq i \leq p, \\
x_e, & \text{otherwise},\end{cases}
\]

By the choice of $\varepsilon$, we have $y_e \geq 0$ for all $e \in E$. Observe that $\mathbf{y}$ also satisfies equation (3), and therefore, by Lemma 2.1, also lies in the perfect matching polytope $PM(H)$. Hence, $w(\mathbf{y}) = q$. 

8
Consider the vector \( x - y \). We have \( w(x - y) = w(x) - w(y) = q - q = 0 \). The coordinates of \( x - y \) are nonzero only on cycle \( C \), where its entries are alternating \( \varepsilon \) and \( -\varepsilon \). Hence,

\[
w(x - y) = \varepsilon \cdot c_w(C) = 0.
\]

We conclude that \( c_w(C) = 0 \).

After the first version of this paper, Rao, Shpilka, and Wigderson (see [GG15, Lemma 2.4]) came up with an alternate proof of Lemma 3.3, which is based on Hall’s theorem instead of the matching polytope.

We apply Lemma 3.3 to graph \( G_1 \). Recall that we chose the weight function such that all small cycles in \( G \) have a nonzero circulation. From Lemma 3.3, we conclude that \( G_1 \) contains no small cycles.

**Corollary 3.4.** Let \( G(V,E) \) be a bipartite graph with weight function \( w \) and let \( C \) be a cycle in \( G \) such that \( c_w(C) \neq 0 \). Let \( E_1 \) be the union of all minimum weight perfect matchings in \( G \). Then graph \( G_1(V,E_1) \) does not contain cycle \( C \).

Now, we want to repeat this procedure with graph \( G_1 \) with a new weight function. However, in \( G_1 \), there are no more small cycles. Hence, we look at slightly larger cycles. We argue that their number remains polynomially bounded.

Teo and Koh [TK92] showed that the number of shortest cycles in a graph with \( m \) edges is bounded by \( m^2 \). In the following lemma, we extend their argument and give a bound on the number of cycles that have length at most twice the length of shortest cycles.

**Lemma 3.5.** Let \( H \) be a graph with \( n \) nodes that has no cycles of length \( \leq r \), for some even \( r \geq 4 \). Then \( H \) has \( \leq n^4 \) cycles of length \( \leq 2r \).

**Proof.** Let \( C = (v_0, v_1, \ldots, v_{\ell-1}) \) be a cycle of length \( \ell \leq 2r \) in \( G \). Let \( f = \ell/4 \). We successively choose four nodes on \( C \) with distance \( \leq \lfloor f \rfloor \leq r/2 \) and associate them with \( C \). We start with \( u_0 = v_0 \) and define \( u_i = v_{[i]f} \), for \( i = 1, 2, 3 \). Note that the distance between \( u_3 \) and \( u_0 \) is also \( \leq \lfloor f \rfloor \). Since we could choose any node of \( C \) as starting point \( u_0 \), the four nodes \((u_0, u_1, u_2, u_3)\) associated with \( C \) are not uniquely defined. However, they uniquely describe \( C \).

**Claim 1.** Cycle \( C \) is the only cycle in \( H \) of length \( \leq 2r \) that is associated with \((u_0, u_1, u_2, u_3)\).

**Proof.** Suppose \( C' \neq C \) would be another such cycle. Let \( p \neq p' \) be paths of \( C \) and \( C' \), respectively, that connect the same \( u \)-nodes. Note that \( p \) and \( p' \) create a cycle in \( H \) of length at most

\[
|p| + |p'| \leq \frac{r}{2} + \frac{r}{2} \leq r,
\]

which is a contradiction. This proves the claim.

There are \( \leq n^4 \) ways to choose 4 nodes and their order. By Claim 1 this gives a bound on the number of cycles of length \( \leq 2r \).

**Lemma 3.5** suggests the following strategy how to continue from \( G_1 \): in each successive round, we double the length of the cycles and adapt the weight function to give nonzero circulations to these slightly longer cycles. By Lemma 3.3 we have that any cycle with nonzero circulation disappears from the new graph obtained by taking only the minimum
perfect matchings from the previous graph. Thus, in \( \log n \) rounds we reach a graph with no cycles, i.e., with a unique perfect matching. Now, we put all the ingredients together and formally define our weight assignment.

### 3.2 Constructing the Weight Assignment

Let \( G(V, E) = G_0 \) be a bipartite graph with \( n \) nodes that has perfect matchings. Define \( k = \lceil \log n \rceil - 1 \), which is the number of rounds we will need. We define subgraphs \( G_i \) and weight assignments \( w_i \), for \( i = 0, 1, 2, \ldots, k - 1 \),

\[
  w_i: \text{ a weight function such that all cycles in } G_i \text{ of length } \leq 2^i + 2 \text{ have nonzero circulations.}
\]

\( G_{i+1} \): the union of minimum weight perfect matchings in \( G_i \) according to weight \( w_i \).

By the definition of \( G_i \), any two perfect matchings in \( G_i \) have the same weight, not only according to \( w_i \), but also to \( w_j \) for all \( j < i \), for any \( 1 \leq i \leq k \).

By Lemma 3.3, graph \( G_i \) does not contain any cycles of length \( \leq 2^i + 1 \) for each \( 1 \leq i \leq k \). In particular, \( G_k \) does not have any cycles, since \( 2^{k+1} \geq n \). Therefore \( G_k \) has a unique perfect matching.

Our final weight function \( w \) will be a combination of \( w_0, w_1, \ldots, w_{k-1} \). We combine them in a way that the weight assignment in a later round does not interfere with the order of perfect matchings given by earlier round weights. Let \( B \) be a number greater than the weight of any edge under any of these weight assignments. Then, define

\[
  w = w_0B^{k-1} + w_1B^{k-2} + \cdots + w_{k-1}B^0.
\]

(8)

In the definition of \( w \), the precedence decreases from \( w_0 \) to \( w_{k-1} \). That is, for any two perfect matchings \( M_1 \) and \( M_2 \) in \( G_0 \), we have \( w(M_1) < w(M_2) \), if and only if there exists an \( 0 \leq i \leq k - 1 \) such that

\[
  \begin{align*}
    w_j(M_1) &= w_j(M_2), \quad \text{for } j < i, \\
    w_i(M_1) &< w_i(M_2).
  \end{align*}
\]

As a consequence, the perfect matchings left in \( G_i \) have a strictly smaller weight with respect to \( w \) than the ones in \( G_{i-1} \) that did not make it to \( G_i \).

**Lemma 3.6.** For any \( 1 \leq i \leq k \), let \( M_1 \) be a perfect matching in \( G_i \) and \( M_2 \) be a perfect matching in \( G_{i-1} \) which is not in \( G_i \). Then \( w(M_1) < w(M_2) \).

**Proof.** Since \( M_1 \) and \( M_2 \) are perfect matchings in \( G_{i-1} \), we have \( w_j(M_1) = w_j(M_2) \), for all \( j < i - 1 \), as observed above. From the definition of \( G_i \) and Lemma 3.2, it follows that \( w_{i-1}(M_1) < w_{i-1}(M_2) \). Hence we get that \( w(M_1) < w(M_2) \).

It follows that the unique perfect matching in \( G_k \) has a strictly smaller weight with respect to \( w \) than all other perfect matchings.

**Corollary 3.7.** The weight assignment \( w \) defined in (8) is isolating for \( G_0 \).
It remains to bound the values of the weights assigned. By Lemma 3.5, the number
of cycles that we handle in each round is \( \leq n^4 \). Therefore, each \( w_i \) needs to give nonzero
circulations to \( \leq n^4 \) cycles, for \( 0 \leq i < k \). By Lemma 2.3 with \( s = n^4 \), this yields a set
of \( O(n^6) \) weight assignments with weights bounded by \( O(n^6) \).

Recall that the number \( B \) used in equation (8) is the highest weight assigned by any \( w_i \). Hence, we also have \( B = O(n^6) \). Therefore the weights in the assignment \( w \) in equation (8) are bounded by \( B^k = O(n^{6 \log n}) \). That is, the weights have \( O(\log^2 n) \) bits.

For each \( w_i \) we have \( O(n^6) \) possibilities and we do not know which one would work.
Therefore we try all of them in parallel. In total, we need to try \( O(n^{6k}) = O(n^{6 \log n}) \) weight assignments.

Clearly, every weight assignment can be constructed in quasi-NC\(^1\) with circuit
size \( 2^{O(\log^2 n)} \).

**Lemma 3.8.** In quasi-NC\(^1\), one can construct a set of \( O(n^{6 \log n}) \) integer weight functions on
\([n/2] \times [n/2] \), where the weights have \( O(\log^2 n) \) bits, such that for any given bipartite graph
with \( n \) nodes, one of the weight functions is isolating.

With the above weight functions, we can decide the existence of a perfect matching in a
bipartite graph in quasi-NC\(^2\) as follows: Recall the bi-adjacency matrix \( A \) from Section 2.2
which has entry \( 2^{w(e)} \) for edge \( e \). We test nonzeroness of \( \det(A) \) for each of the constructed
weight functions in parallel. If the given graph has a perfect matching, then one of the
weight functions isolates a perfect matching. As we discussed in Section 2.2 for this weight
function \( \det(A) \) will be nonzero. When there is no perfect matching, then \( \det(A) \) will be
zero for any weight function. As our weights have \( O(\log^2 n) \) bits, the determinant entries
have quasi-polynomially many bits. So, computing \( \det(A) \) is not efficient. Instead, we
test whether \( \det(A) \neq 0 \) (mod \( p \)) for some \( 2 \leq p \leq t \). Clearly, choosing a \( t \) such that
\( \text{lcm}(2, 3, \ldots, t) > \det(A) \) will suffice. Since \( \text{lcm}(2, 3, \ldots, t) > 2^t \) for \( t \geq 7 \) (See [Nai82]), we can choose \( t = \log \det(A) = 2^{O(\log^2 n)} \). While computing modulo \( p \), the determinant entries
will be small, and hence, \( \det(A) \mod p \) can be computed in NC\(^2\) [Ber84, BCP84]. As we need to compute
\( 2^{O(\log^2 n)} \)-many determinants in parallel, our algorithm is in quasi-NC\(^2\) with
circuit size \( 2^{O(\log^2 n)} \).

To construct a perfect matching, we follow the algorithm of Mulmuley et al. [MVV87] from
Section 2.2 with each of our weight functions. For a weight function \( w \) which is isolating, the algorithm outputs the unique minimum weight perfect matching \( M \). If we have a weight
function \( w' \) which is not isolating, still \( \det(A) \) might be non-zero with respect to \( w' \). In
this case, the algorithm computes a set of edges \( M' \) that might or might not be a perfect
matching. However, it is easy to verify if \( M' \) is indeed a perfect matching, and in this case,
we will output \( M' \).

Here, the algorithm needs to compute \( \det(A) \) exactly, which we argue can be done
in quasi-NC\(^3\) with circuit size \( 2^{O(\log^2 n)} \). Berkowitz [Ber84] reduced the determinant computation
to \( \log n \) rounds of matrix squaring. Usual matrix squaring is in NC\(^1\), but as our
matrix entries have \( 2^{O(\log^2 n)} \) bits, it falls into quasi-NC\(^2\) with circuit size \( 2^{O(\log^2 n)} \). Hence,
\( \det(A) \) can be computed in quasi-NC\(^3\). This finishes the proof of Theorem 3.1.
4 An RNC-Algorithm with Few Random Bits

We can also present our result for bipartite perfect matching in an alternate way. Instead of quasi-NC, we can get an RNC-circuit, but with only poly-logarithmically many, namely $O(\log^2 n)$ random bits. Note that for a complete derandomization, it would suffice to bring the number of random bits down to $O(\log n)$. Then there are only polynomially many random strings which can all be tested in NC. Hence we are only one log-factor away from a complete derandomization.

4.1 Decision Version

First, let us look at the decision version.

**Theorem 4.1.** For bipartite graphs, there is an RNC$^2$-algorithm for PM which uses $O(\log^2 n)$ random bits.

To prove Theorem 4.1, consider our algorithm from Section 3. There are two reasons that we need quasi-polynomially large circuits: (i) we need to try quasi-polynomially many different weight assignments and (ii) each weight assignment has quasi-polynomially large weights. We show how to come down to polynomial bounds in both cases by using randomization.

To solve the first problem, we modify Lemma 2.3 to get a random weight assignment which works with high probability (see [CRS95, KS01]).

**Lemma 4.2.** Let $G$ be a graph with $n$ nodes and $s \geq 1$. There is a random weight assignment $w$ which uses $O(\log(ns))$ random bits and assigns weights bounded by $O(n^3s \log ns)$, i.e., with $O(\log ns)$ bits, such that for any set of $s$ cycles, $w$ gives nonzero circulation to each of the $s$ cycles with probability at least $1 - 1/n$.

**Proof.** We follow the construction of Lemma 2.3 and give exponential weights modulo small numbers. Here, we use only prime numbers as moduli. Let $w(e_i) = 2^{i-1}$. Choose a random number $p$ among the first $t$ prime numbers. We take our random weight assignment to be $w \mod p$.

We want to show that, with high probability, this weight function gives nonzero circulation to every cycle in $\{C_1, C_2, \ldots, C_s\}$. In other words, $\prod_{i=1}^s c_w(C_i) \not\equiv 0 \pmod{p}$. As the product is bounded by $2^{n^2s}$, it has at most $n^2s$ prime factors. Choose $t = n^3s$. Then a random prime works with probability $\geq (1 - 1/n)$. As the $t$-th prime number is bounded by $2t \log t$, the weights are bounded by $2t \log t = O(n^3s \log ns)$. Hence, the weights have $O(\log ns)$ bits. A random prime with $O(\log ns)$ bits can be constructed using $O(\log ns)$ random bits (see [KS01]).

Recall from Section 3.2 that for a bipartite graph $G$ with $n$ nodes, we had $k = \lceil \log n \rceil - 1$ rounds and constructed one weight function in each round. We do the same here, however, we use the random scheme from Lemma 4.2 to choose each of the weight functions $w_0, w_1, \ldots, w_{k-1}$ independently. The probability that all of them provide nonzero circulation on their respective set of cycles is at least $1 - k/n \geq 1 - \log n/n$ using the union bound.

Now, instead of combining them to form a single weight assignment, we use a different variable for each weight assignment. We modify the construction of matrix $A$ from Section 2.2.
Let \( L = \{u_1, u_2, \ldots, u_{n/2}\} \) and \( R = \{v_1, v_2, \ldots, v_{n/2}\} \) be the vertex partition of \( G \). For variables \( x_0, x_1, \ldots, x_{k-1} \), define an \( n/2 \times n/2 \) matrix \( A \) by

\[
A(i, j) = \begin{cases} 
    x_0^{w_0(e)} x_1^{w_1(e)} \cdots x_{k-1}^{w_{k-1}(e)}, & \text{if } e = (u_i, v_j) \in E, \\
    0, & \text{otherwise.}
\end{cases}
\]

From arguments similar to those in Section 2.2, one can write

\[
\det(A) = \sum_{M \text{ pm in } G} \text{sgn}(M) x_0^{w_0(M)} x_1^{w_1(M)} \cdots x_{k-1}^{w_{k-1}(M)},
\]

From the construction of the weight assignments it follows that if the graph has a perfect matching then the lexicographically minimum term in \( \det(A) \), with respect to the exponents of variables \( x_0, x_1, \ldots, x_{k-1} \) in this precedence order, comes from a unique perfect matching. Thus, we get the following lemma.

**Lemma 4.3.** \( \det(A) \neq 0 \iff G \text{ has a perfect matching.} \)

Recall that each \( w_i \) needs to give nonzero circulations to \( n^4 \) cycles. Thus, the weights obtained by the scheme of Lemma 4.2 will be bounded by \( O(n^7 \log n) \). Therefore, the weight of a matching will be bounded by \( O(n^8 \log n) \). Hence, \( \det(A) \) is a polynomial of individual degree \( O(n^8 \log n) \) with \( \log n \) variables. To test if \( \det(A) \) is nonzero one can apply the standard randomized polynomial identity test [Sch80, Zip79, DL78]. That is, to plug in random values for variables \( x_i \), independently from \( \{1, 2, \ldots, n^9\} \). If \( \det(A) \neq 0 \), then the evaluation is nonzero with high probability.

We bound the number of random bits. For a weight assignment \( w_i \), we need \( O(\log ns) \) random bits from Lemma 4.2 where \( s = n^4 \). Thus, the number of random bits required for all \( w_i \)'s together is \( O(k \log n) = O(\log^2 n) \). Finally, we need to plug in \( O(\log n) \) random bits for each \( x_i \). This again requires \( O(\log^2 n) \) random bits.

We bound the circuit size. The weight construction involves taking exponential weights modulo small primes by Lemma 4.2. Primality testing can be done by the brute force algorithm in \( \text{NC}^2 \), as the numbers involved have \( O(\log n) \) bits. Thus, the weight assignments can be constructed in \( \text{NC}^2 \). Moreover, the determinant with polynomially bounded entries can be computed in \( \text{NC}^2 \) [Ber84].

In summary, we get an \( \text{RNC}^2 \)-algorithm that uses \( O(\log^2 n) \) random bits as claimed in Theorem 4.1.

### 4.2 Search Version

We get a similar algorithm for Search-PM using also only \( O(\log^2 n) \) random bits. This improves the RNC-algorithm of Goldwasser and Grossman [GG15] based on an earlier version of this paper that uses \( O(\log^4 n) \) random bits. Their RNC-algorithm has an additional property: it is pseudo-deterministic, i.e., it outputs the same perfect matching for almost all choices of random bits. Our algorithm does not have this property.

**Theorem 4.4.** For bipartite graphs, there is an \( \text{RNC}^3 \)-algorithm for Search-PM which uses \( O(\log^2 n) \) random bits.
Let again $G(V, E)$ be the given bipartite graph with vertex partition $L = \{u_1, u_2, \ldots, u_{n/2}\}$ and $R = \{v_1, v_2, \ldots, v_{n/2}\}$. We construct the weight assignments $w_0, w_1, \ldots, w_{k-1}$ as in Lemma 4.2 in the randomized decision version. Let $M^*$ be the unique minimum weight perfect matching in $G$ with respect to the combined weight function $w$. Let $w_r(M^*) = w^*_r$, for $0 \leq r < k$.

Recall from Section 3.2 the sequence of subgraphs $G_1, G_2, \ldots, G_k$ of $G = G_0$, where $G_{r+1}$ consists of the minimum perfect matchings of $G_r$ according to weight $w_r$. In order to compute $M^*$, we would like to actually construct all the graphs $G_1, G_2, \ldots, G_k$. However, it is not clear how to achieve this with $O(\log^2 n)$ random bits. Instead, we will construct a sequence of graphs $H_1, H_2, \ldots, H_k$ such that $H_r$ will be a subgraph of $G_r$, for each $1 \leq r \leq k$. Furthermore, each $H_r$ will contain the matching $M^*$. Recall that $G_k$ consists of the unique perfect matching $M^*$. Hence, once we have $H_k = G_k$, we are done.

Let $H_0 = G$ and $0 \leq r < k$. We describe the $r$-th round. Suppose we have constructed the graph $H_r(V, E_r)$ and want to compute $H_{r+1}$. An edge will appear in $H_{r+1}$ only if it participates in a matching $M$ with $w_r(M) = w^*_r$. Thus, we will have that $H_{r+1}$ is a subgraph of $G_{r+1}$. For an edge $e$, let $X^{w(e)}_r$ denote the product

$$X^{w(e)}_r = x^{w(e)}_{r} x^{w_{r+1}(e)}_{r+1} \cdots x^{w_{k-1}(e)}_{k-1}. $$

For a matching $M$, the term $X^{w(M)}_r$ is defined similarly. Let $N(e)$ denote the set of edges which are neighbors of an edge $e$ in $G_r$, i.e. all edges $e' \neq e$ that share an endpoint with $e$. For an edge $e \in E_r$, define the $n/2 \times n/2$ matrix $A_e$ as

$$A_e(i, j) = \begin{cases} X^{w(e')}_{r}, & \text{if } e' = (u_i, v_j) \in E_r - N(e), \\ 0, & \text{otherwise}. \end{cases} $$

Note that the matrix $A_e$ has a zero entry for each neighboring edge of $e$. Thus, its determinant is a sum over all perfect matchings which contain $e$. That is,

$$\det(A_e) = \sum_{M \text{ pm in } H_r, e \in M} \text{sgn}(M) X^{w(M)}_r. $$

Consider the coefficient $c_e$ of $x^{w_r}_{r}$ in $\det(A_e)$,

$$c_e = \sum_{M \text{ pm in } H_r, w_r(M) = w^*_r, e \in M} \text{sgn}(M) X^{w(M)}_{r+1}. $$

Define the graph $H_{r+1}$ to be the union of all the edges $e$ for which the polynomial $c_e$ is nonzero. We claim that $c_e$ is nonzero for each edge in $M^*$ and thus, these edges appear in $H_{r+1}$: For any edge $e \in M^*$, the polynomial $c_e$ will contain the term $X^{w(M^*)}_{r+1}$. As the matching $M^*$ is isolated in $H_r$ with respect to the weight vector $(w_{r+1}, \ldots, w_{k-1})$, the polynomial $c_e$ is nonzero. Note that the polynomial $c_e$ can be zero for an edge $e$ which participates in a matching $M$ with $w_r(M) = w^*_r$. Therefore $H_{r+1}$ is a subgraph of $G_{r+1}$.

For the construction of $H_{r+1}$, we need to test if $c_e$ is nonzero, for each edge $e$ in $H_r$. As argued above in the decision part, the degree of $c_e$ is $O(n^8 \log^2 n)$. We apply the standard zero-test, i.e., we plug in random values for the variables $x_{r+1}, \ldots, x_{k-1}$ independently from
The probability that the evaluation will be nonzero is at least \(1 - O(\log^2 n/n^3)\). To compute this evaluation, we plug in values of \(x_{r+1}, \ldots, x_{k-1}\) in \(\det(A_e)\) and find the coefficient of \(x_{r+1}^{w_e}\). This can be done in \(\text{NC}^2\) [BCPS84, Corollary 4.4]. For all the edges, we use the same random values for variables \(x_{r+1}, \ldots, x_{k-1}\) in each identity test. The probability that the test works successfully for each edge is at least \(1 - O(\log^2 n/n)\) by the union bound.

We continue this for \(k\) rounds to find \(H_k\), which is a perfect matching.

We need again \(O(\log^2 n)\) random bits for the weight assignments \(w_0, w_1, \ldots, w_{k-1}\) and the values for the \(x_i\)’s. Note that we use the same random bits for \(x_i\) in all \(k\) rounds. This decreases the success probability, which is now at least \(1 - O(\log^3 n/n)\) by the union bound.

In \(\text{NC}^2\), we can construct the weight assignments and compute the determinants in each round. As we have \(k = O(\log n)\) rounds, the overall complexity becomes \(\text{NC}^3\).

5 Extensions and Related Problems

There are many problems related to perfect matching (see for example [KR98, Chapter 14 and 15]). Some of them are subtree isomorphism [KL89], maximum flow [KUW86], and constructing a depth-first search tree [AAK90, AA87]. These problems have an \(\text{NC}\)-reduction to bipartite perfect matching and thus fall in quasi-\(\text{NC}\).

A generalization of perfect matchings are \(b\)-factors in a graph. The bipartite \(b\)-factor problem also falls in quasi-\(\text{NC}\) via a reduction to perfect matching (see, for example, [LP86, Section 10.1]). We can also find an isolating weight assignment for bipartite \(b\)-factors directly, using the same construction from Section 3.2. There are other versions of the matching problem, where our techniques apply. Below, we elaborate them.

5.1 Bipartite Planar Graphs

The Search-PM problem already has some known \(\text{NC}\)-algorithms in the case of bipartite planar graphs [MN95, MV00, DKR10]. The one by Mahajan and Varadarajan [MV00] is in \(\text{NC}^3\), while the other two are in \(\text{NC}^2\). Our approach from the previous section can be modified to give an alternate \(\text{NC}^3\)-algorithm for this case.

The weights in our scheme in Section 3.2 become quasi-polynomial because we need to combine the different weight functions from \(\log n\) rounds using a different scale. To solve this problem, we use the fact that in planar graphs, one can count the number of perfect matchings of a given weight in \(\text{NC}^2\) by the Pfaffian orientation technique [Kas67, Vaz89]. As a consequence, we can actually construct the graphs \(G_i\) in each round in \(\text{NC}^2\). Thereby we avoid having to combine the weight functions from different rounds.

In more detail, in the \(i\)-th round, we need to compute the union of minimum weight perfect matchings in \(G_{i-1}\) according to \(w_{i-1}\). For each edge \(e\), we decide in parallel if deleting \(e\) reduces the count of minimum weight perfect matchings. If yes, then edge \(e\) should be present in \(G_i\). As it takes \(\log n\) rounds to reach a single perfect matching, the algorithm is in \(\text{NC}^3\).

5.2 Weighted Perfect Matchings and Maximum Matchings

A generalization of the perfect matching problem is the weighted perfect matching problem (weight-PM), where we are given a weighted graph, and we want to compute a perfect matching of minimum weight. There is no \(\text{NC}\)-reduction known from weight-PM to the perfect matching problem. However, the isolation technique works for this problem as well, when the
weights are small integers. We put the given weights on a higher scale and put the weights constructed by our scheme in Section 3 on a lower scale. This ensures that a minimum weight perfect matching according to the combined weight function also has minimum weight according to the given weight assignment. Our scheme ensures that there is a unique minimum weight perfect matching. One can construct this perfect matching following the algorithm of Mulmuley et al. [MVV87] (Section 2.2).

**Corollary 5.1.** For bipartite graphs, weight-PM with quasi-polynomially bounded integer weights is in quasi-NC².

The maximum matching problem asks to find a maximum size matching in a given graph. It is well known that the maximum matching problem (MM) is NC-equivalent to the perfect matching problem (see for example [GKMT13]). The equivalence holds for both decision versions and the construction versions. The reductions also preserve bipartiteness of the graph. Thus, we get the following corollary.

**Corollary 5.2.** For bipartite graphs, MM is in quasi-NC².

**Discussion**

The major open question remains whether one can do isolation with polynomially bounded weights. Our construction requires quasi-polynomial weights because it takes log n rounds to reach a unique perfect matching and the graphs obtained in the successive rounds cannot be constructed. To get polynomially bounded weights one needs to circumvent this.

For non-bipartite graphs, the isolation question is open even in the planar case. For this case, our approach fails in its first step: Lemma 3.2 no longer holds as demonstrated in Figure 1. Can one assign weights in a way which ensures that the union of minimum weight perfect matchings is significantly smaller than the original graph?

Another generalization of bipartite matching (and b-matching) is the matroid intersection problem. Given two matroids over the same ground set, the problem asks if there is a common basis. The problem has a polynomial time algorithm [Edm79] and also lies in RNC (for linear matroids) [NSV94]. Can we find an NC or quasi-NC algorithm?

**Acknowledgements**

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 techniques used in Section 4 and Jacobo Torán for discussions on the number of shortest cycles.

**References**


