SIAM J. COMPUT.
Vol. 0, No. 0, pp. STOC16-218–STOC16-235

BIPARTITE PERFECT MATCHING IS IN QUASI-NC*

STEPHEN FENNER†, ROHIT GURJAR‡, AND THOMAS THIERAUF§

Abstract. We show that the bipartite perfect matching problem is in quasi-NC^2. That is, it has uniform circuits of quasi-polynomial size $n^{O(\log n)}$, and $O(\log^2 n)$ 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.

Key words. graph matching, parallel complexity, derandomization, quasi-NC, perfect matching, bipartite graphs, parallel algorithms, Isolation Lemma, matching polytope

AMS subject classifications. 68R10, 68W10, 68Q25, 68W20

DOI. 10.1137/16M1097870

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 [17]. 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 [34]; 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, Upfal, and Wigderson [29] and Mulmuley, Vazirani, and Vazirani [38]. 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, Vazirani, and Vazirani [38] 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.

LEMMA 1.1 (Isolation Lemma [38]). *For a graph $G(V, E)$, let w be a random weight assignment, where edges are assigned weights chosen uniformly and indepen-*

*Received by the editors October 7, 2016; accepted for publication (in revised form) April 12, 2018; published electronically October 24, 2019. A conference version of this paper appeared in the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016) [19].

https://doi.org/10.1137/16M1097870

Funding: The second and third authors were supported by DFG grant TH 472/4.

†Dept. of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208 (fenner.sa@gmail.com, https://cse.sc.edu/~fenner).

‡Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA 91125 (rohitgurjar0@gmail.com, http://www.cse.iitk.ac.in/users/rgurjar).

§Dept. of Electrical Engineering and Computer Science, Aalen University, Aalen 73430, Germany (thomas.thierauf@uni-ulm.de, http://image.informatik.htw-aalen.de/~thierauf/).

dently at random from $\{1, 2, \ldots, 2|E|\}$. *Then* w *is isolating with probability at least* $1/2$.

Also see [44, 18] for alternate proofs and improved probability bounds. *Derandomizing* this lemma means constructing 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 [5].

The Isolation Lemma has been derandomized for some special classes of graphs, e.g., planar bipartite graphs [14, 46], strongly chordal graphs [13], and graphs with a small number of perfect matchings [23, 3]. In this work, we take 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 and Search-PM are in quasi-NC^2. In particular, they can be solved by uniform Boolean circuits of depth $O(\log^2 n)$ and size $n^{O(\log n)}$ for graphs with n nodes. Note that the size is just one $\log n$-exponent away from polynomial size.

Our result also gives an RNC algorithm for PM in bipartite graphs which uses very few random bits. The original RNC algorithm of Lovász [34] uses $O(m \log n)$ random bits. This has been improved by Chari, Rohatgi, and Srinivasan [10] to $O(n \log(m/n))$ random bits. They actually construct an isolating weight assignment using this 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 [11]; that is, the improvement to [10] 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 [21] observed that one can get an RNC algorithm for Search-PM which uses $O(\log^4 n)$ random bits. With our improved decision algorithm, we obtain now an RNC algorithm for Search-PM which uses only $O(\log^2 n)$ random bits. Later on, Goldwasser and Grossman also improved the number of random bits to $O(\log^2 n)$ in a subsequent version of their paper [22]. 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.

In section 5 we show that our approach also gives an alternate NC algorithm for Search-PM in bipartite planar graphs. This case already has known NC algorithms [37, 36, 14]. Our algorithm is in NC^3, while the previous best known upper bound is NC^2 [37, 14].

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, Kulkarni, and Roy [14] 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 of length $< 2r$ is polynomially bounded. This motivates the following strategy which works in $\log n$ rounds: in the ith round, assign weights that 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.

Subsequent work. In one of the follow-up works, Gurjar and Thierauf [25] showed that the linear matroid intersection problem is in quasi-NC. The problem is a generalization of bipartite matching. Given two matroids over the same ground set, the problem asks if there is a common base. In a subsequent work, Gurjar, Thierauf, and Vishnoi [26] further generalized the derandomization of the Isolation Lemma to polytopes with totally unimodular faces. This class of polytopes, in particular, contains the bipartite perfect matching polytope and the common base polytope of two matroids. However, their result is not known to imply quasi-NC algorithms for any new class of problems.

Generalizing our work in another direction, Svensson and Tarnawski [43] proved that the matching problem for general graphs is in quasi-NC. Benefitting from some of the ideas in this series of works, Anari and Vazirani [4] gave an NC algorithm for Search-PM in planar graphs. This generalizes the NC bound for Search-PM in bipartite planar graphs [37, 36, 14]. In an independent work, Sankowski [41] also gives the same result.

In a related work, Kallampally and Tewari [27] used techniques similar to the current work 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 and edge set E. Throughout the paper, we use n and m to denote the cardinality of these sets, i.e., $|V| = n$ and $|E| = m$.

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 $w\colon E \to \mathbb{Z}$ 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.

The *perfect matching problem* PM is to decide whether a given graph has a perfect matching. Its construction version 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 NC^k, Barrington [6] defined the class quasi-NC^k as the class of prob-

lems which have uniform Boolean circuits of quasi-polynomial size $2^{\log^{O(1)} n}$ and poly-logarithmic depth $O(\log^k n)$. Here, by *uniform circuits* we mean quasi-polynomial time uniform circuits. The class quasi-NC is the union of classes quasi-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, and Vazirani [38] 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 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 [38] computes the determinant of A. An easy argument shows that this determinant is the signed sum over all perfect matchings in G:

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

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

Equation (2) holds because the product $\prod_{i=1}^{n/2} A(i, \pi(i))$ is nonzero if and only if the permutation π corresponds to a perfect matching. Here $\operatorname{sgn}(M)$ 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 $\operatorname{sgn}(M)$, 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 the matrix A have polynomially many bits. This suffices to compute the determinant in NC^2 [8, 12]. Hence, the construction is also in NC^2. Altogether, 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 [36].

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

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

This vector is referred to 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:

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

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

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

Clearly, for any perfect matching M, we have $w(M) = w(\boldsymbol{x}^M)$. In particular, let M^* be a perfect matching in G of minimum weight. Then

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

The following lemma gives a simple, well-known description of the perfect matching polytope of a bipartite graph G; see, for example, [35]. 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 $\boldsymbol{x} = (x_e)_{e\in E} \in \mathbb{R}^m$. Then $\boldsymbol{x} \in \mathrm{PM}(G)$ if and only if*

$$\sum_{e\in\delta(v)} x_e = 1, \qquad v \in V, \tag{3}$$

$$x_e \geq 0, \qquad e \in E. \tag{4}$$

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 nontrivial part is to show that any point satisfying these two conditions is in the perfect matching polytope (see [35, 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. One has to add exponentially many *odd cut constraints* (see [35, Chapter 7]).

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 after removing the vertices of C the graph 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, Kulkarni, and Roy [14].

LEMMA 2.2 ([14]). *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. ∎

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 [20, 10]).

LEMMA 2.3. *Let G be a graph with n nodes. Then, for any number s, one can construct a set of $O(n^2 s)$ weight functions with weights bounded by $O(n^2 s)$, 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 $\{\, w \bmod j \mid 2 \leq j \leq t \,\}$ 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

$$\exists j \leq t \quad \forall i \leq s : \; c_{w \bmod j}(C_i) \neq 0.$$

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,

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

This can be achieved by setting $\operatorname{lcm}(2, 3, \ldots, t) > \prod_{i=1}^{s} c_w(C_i)$. Since $c_w(C_i) \leq 2^{n^2}$, we have $\prod_{i=1}^{s} c_w(C_i) \leq 2^{n^2 s}$. Furthermore, we have $\operatorname{lcm}(2, 3, \ldots, t) > 2^t$ for $t \geq 7$ (see [39]). Thus choosing $t = n^2 s$ suffices. Clearly, the weights are bounded by $t = n^2 s$. ∎

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.

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 subsection 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, Kulkarni, and Roy [14] 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 (1) from subsection 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 [48]. 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 $\mathrm{PM}(G)$ given in Lemma 2.1. As the weight function is linear, the points of minimum weight form a face F in $\mathrm{PM}(G)$. 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 (4): $x_e \geq 0$. Thus, for the face F there exists a set $S \subseteq E$ such that for any $\boldsymbol{x} = (x_e)_{e \in E}$, we have $\boldsymbol{x} \in F$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1, \qquad v \in V, \tag{5}$$

$$x_e \geq 0, \qquad e \in E - S, \tag{6}$$

$$x_e = 0, \qquad e \in S. \tag{7}$$

For $e \in S$, the equality $x_e = 0$ means that edge e does not participate in any

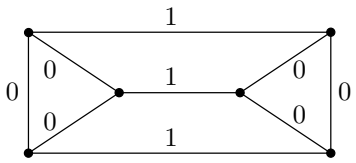

FIG. 1. *A nonbipartite weighted graph where every edge is contained in a minimum perfect matching of weight* 1. *However, the graph also has a perfect matching of weight* 3. *That is, Lemma* 3.2 *does not hold for nonbipartite graphs.*

minimal perfect matching of G. Therefore, we have $G_1 = G - S$. The perfect matching polytope $\mathrm{PM}(G - S)$ is given by (5) and (6). Hence, face F is the perfect matching polytope of graph $G - S$, i.e., $F = \mathrm{PM}(G - S) = \mathrm{PM}(G_1)$. 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. ∎

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 by definition, every edge of G_1 belongs to some perfect matching.

LEMMA 3.3. *Let $H(V, E)$ be a bipartite graph where every edge belongs to some perfect matching. 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 $\boldsymbol{x}$ in the perfect matching polytope $\mathrm{PM}(H)$ is a convex combination of perfect matchings. Therefore, we also have $w(\boldsymbol{x}) = q$.

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

$$\boldsymbol{x} = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_t}{t}.$$

Since each edge participates in a perfect matching, every coordinate of $\boldsymbol{x} = (x_e)_{e \in E}$ 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 $\boldsymbol{x}$ along the cycle C, we remain inside $\mathrm{PM}(H)$. This technique of moving along the cycle has been used by Mahajan and Varadarajan [36]. To elaborate, we define a new point $\boldsymbol{y} = (y_e)_{e \in E}$ 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 ε, we have $y_e \geq 0$ for all $e \in E$. Observe that $\boldsymbol{y}$ also satisfies (3) and, therefore, by Lemma 2.1, also lies in the perfect matching polytope $\mathrm{PM}(H)$. Hence, $w(\boldsymbol{y}) = q$.

Consider the vector $\boldsymbol{x} - \boldsymbol{y}$. We have $w(\boldsymbol{x} - \boldsymbol{y}) = w(\boldsymbol{x}) - w(\boldsymbol{y}) = q - q = 0$. The coordinates of $\boldsymbol{x} - \boldsymbol{y}$ are nonzero only on cycle C, where its entries are alternating ε

and $-\varepsilon$. Hence,

$$w(\boldsymbol{x}-\boldsymbol{y}) = \varepsilon \cdot c_w(C) = 0.$$

We conclude that $c_w(C) = 0$. ∎

After the first version of this paper, Rao, Shpilka, and Wigderson (see [22, 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 [45] 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 at most 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 at most $\lceil f \rceil \leq r/2$ and *associate* them with C. We start with $u_0 = v_0$ and define $u_i = v_{\lceil if \rceil}$ for $i = 1, 2, 3$. Note that the distance between u_3 and u_0 is also at most $\lceil f \rceil$. 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 3.6. *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 at most n^4 ways to choose 4 nodes and their order. By Claim 3.6, this gives a bound on the number of cycles of length $\leq 2r$. ∎

Lemma 3.5 suggests the following strategy for continuing 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: A weight function such that all cycles in G_i of length $\leq 2^{i+2}$ 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 perfect matching under any of these weight assignments. Then, define

$$w = w_0 B^{k-1} + w_1 B^{k-2} + \cdots + w_{k-1} B^0. \tag{8}$$

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

$$\begin{aligned} w_j(M_1) &= w_j(M_2) \qquad \text{for } j < i, \\ w_i(M_1) &< w_i(M_2). \end{aligned}$$

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 3.7. *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 3.8. *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 does not exceed n^4. Therefore, each w_i needs to give nonzero circulations to at most 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 (8) is greater than the weight of any perfect matching with respect to any w_i. Hence, we have $B = O(n^7)$. Therefore, the weights in the assignment w in (8) are bounded by $B^k = n^{O(\log n)}$. 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 will work. Therefore, we try all of them in parallel. In total, we need to try $O(n^{6k}) = n^{O(\log n)}$ weight assignments.

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

LEMMA 3.9. *In* quasi-NC^1*, one can construct a set of* $n^{O(\log n)}$ *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 subsection 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 subsection 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. The determinant can still be computed in parallel using the Berkowitz algorithm [8] with Chinese remaindering, but it requires circuits of quasi-polynomial size $2^{O(\log^2 n)}$, as we explain next.

Beame, Cook, and Hoover [7] have shown that Chinese remaindering with n-bit numbers is in NC^1.

LEMMA 3.10 ([7]). *Given* (i) *prime numbers* $p_1, p_2, \ldots, p_n \le n^2$*,* (ii) *their product* $p_1 p_2 \cdots p_n$*, and* (iii) *for some number* $D \le p_1 p_2 \cdots p_n$*, the remainders* $D \bmod p_i$*, for* $i = 1, 2, \ldots, n$*, then* D *can be computed in* NC^1*.*

In our case, the number $D = \det(A)$ has $2^{O(\log^2 n)}$ bits. Hence we take the first $2^{O(\log^2 n)}$ prime numbers. Then their product exceeds D. The primes will not be computed by the circuit, but instead are hardwired into it. Similarly, all values $2^w \bmod p$ for all $w \le 2^{O(\log^2 n)}$ and $p \le 2^{O(\log^2 n)}$ will be hardwired into the circuit. Now, to compute $\det(A) \bmod p$ for some weight function w, we first pick the matrix entries modulo these primes from the hardwired values, and then compute $\det(A) \bmod p$ in NC^2 [8, 9]. Finally, we compute $\det(A)$ by Lemma 3.10, which yields a quasi-NC^2 circuit in our case.

To construct a perfect matching, we follow the algorithm of Mulmuley, Vazirani, and Vazirani [38] from subsection 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, $\det(A)$ might still be nonzero 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 whether M' is indeed a perfect matching, and in this case, we will output M'. This finishes the proof of Theorem 3.1.

4. An RNC algorithm with a 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 and succeeds with a high probability.*

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 [10, 32]).

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 $O(n^3 s \log ns)$, 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 $w \bmod p$.

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 $2^{n^2 s}$, it has at most $n^2 s$ prime factors. Choose $t = n^3 s$. Then a random prime works with probability at least $1 - 1/n$. As the tth prime number is bounded by $2t \log t$, the weights are bounded by $2t \log t = O(n^3 s \log ns)$. 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 [32]). ∎

Recall from subsection 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 subsection 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 bipartition 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 subsection 2.2, one can write

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

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. *If the combined weight function $(w_0, w_1, \ldots, w_{k-1})$ is isolating, then $\det(A) \neq 0$ if and only if G 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 [42, 49, 16]. That is, 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 each 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 NC^2, as the numbers involved have $O(\log n)$ bits. Thus, the weight assignments can be constructed in NC^2. Moreover, the determinant with polynomially bounded entries can be computed in NC^2 [8].

In summary, we get an 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 again only $O(\log^2 n)$ random bits.

THEOREM 4.4. *For bipartite graphs, there is an* RNC^3 *algorithm for* Search-PM *which uses* $O(\log^2 n)$ *random bits and has a high success probability.*

Let again $G(V, E)$ be the given bipartite graph with vertex bipartition $L = \{u_1, u_2, \ldots, u_{n/2}\}$ 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 subsection 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 $1 \leq r \leq k$.

The reason that we get subgraphs H_r instead of all of G_r will become clear below. Very roughly, it is because here we work with the final weight function w already in the rth round instead of just w_r. Therefore, only the edges in M^* are guaranteed to survive in H_r. But this suffices for our purpose: 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 rth 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 and a perfect matching M, we define the products

$$\boldsymbol{X}_r^{\boldsymbol{w}(e)} = x_r^{w_r(e)} \; x_{r+1}^{w_{r+1}(e)} \; \cdots \; x_{k-1}^{w_{k-1}(e)},$$
$$\boldsymbol{X}_r^{\boldsymbol{w}(M)} = x_r^{w_r(M)} \; x_{r+1}^{w_{r+1}(M)} \; \cdots \; x_{k-1}^{w_{k-1}(M)}.$$

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

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

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,

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

Consider the coefficient c_e of $x_r^{w_r^*}$ in $\det(A_e)$,

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

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 $e \in M^*$, and thus these edges appear in H_{r+1}: For any edge $e \in M^*$, the polynomial c_e will contain the term $\boldsymbol{X}_{r+1}^{\boldsymbol{w}(M^*)}$. 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 whether 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 $\{1, 2, \ldots, n^{11}\}$. 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 $x_r^{w_r^*}$. This can be done in NC^2 [9, 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 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 NC^3.

5. Extensions and related problems. There are many problems related to perfect matching (see, for example, [30, Chapter 14 and 15]). The problems that have an NC-reduction to bipartite perfect matching include subtree isomorphism [33], maximum flow with polynomially bounded capacities [29], minimum cut and minimum s-t-cut for directed/undirected graphs (reduces to maximum flow [40]), and constructing a depth-first search tree in a graph [2, 1]. As a corollary to our result, we get them all in quasi-NC. Note that the minimum cut problem is easier for undirected graphs, and it already has an NC algorithm in that case [28].

A generalization of perfect matchings are b-factors in a graph. The bipartite b-factor problem also falls in quasi-NC via a reduction to perfect matching (see, for example, [35, section 10.1]). We can also find an isolating weight assignment for bipartite b-factors directly, using the same construction from subsection 3.2. There are other versions of the matching problem where our techniques apply. We elaborate on some of them below.

5.1. Bipartite planar graphs. The Search-PM problem already has some known NC algorithms in the case of bipartite planar graphs [37, 36, 14]. The one by Mahajan and Varadarajan [36] is in NC^3, while the other two are in NC^2. Our approach from the previous section can be modified to give an alternate NC^3 algorithm for this case.

The weights in our scheme in subsection 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 NC^2 by the Pfaffian orientation technique [31, 47]. As a consequence, we can actually construct the graphs G_i in each round in NC^2. Thereby we avoid having to combine the weight functions from different rounds.

In more detail, in the ith 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. We have to do this for each of the $O(n^6)$ possibilities of w_{i-1}. We know there is at least one choice of w_{i-1} which ensures G_i does not contain any cycles of length $\leq 2^{i+1}$. We can test this in NC^2 for each G_i via the shortest path algorithm (see, for example, [15]). In the $(i+1)$th round, we work with that G_i which has this property.

As it takes $\log n$ rounds to reach a single perfect matching, the overall algorithm is in 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 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, Vazirani, and Vazirani [38] (subsection 2.2).

COROLLARY 5.1. *For bipartite graphs,* weight-PM *is in* quasi-NC^2 *when the given weights are quasi-polynomially bounded integers.*

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, [24]). 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-NC^2.

Discussion. The major open question remains whether one can do isolation with *polynomially bounded* weights. Then we would have bipartite perfect matching in NC. 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.

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 techniques used in section 4, and Jacobo Torán for discussions on the number of shortest cycles. We thank the anonymous reviewers for helpful suggestions.

REFERENCES

[1] A. Aggarwal and R. J. Anderson, *A random NC algorithm for depth first search*, in Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 1987, pp. 325–334, https://doi.org/10.1145/28395.28430.

[2] A. Aggarwal, R. J. Anderson, and M.-Y. Kao, *Parallel depth-first search in general directed graphs*, SIAM J. Comput., 19 (1990), pp. 397–409, https://doi.org/10.1137/0219025.

[3] M. Agrawal, T. M. Hoang, and T. Thierauf, *The polynomially bounded perfect matching problem is in NC^2*, in Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Comput. Sci. 4393, Springer, Berlin, Heidelberg, 2007, pp. 489–499, https://doi.org/10.1007/978-3-540-70918-3_42.

[4] N. Anari and V. V. Vazirani, *Planar Graph Perfect Matching Is in NC*, preprint, https://arxiv.org/abs/1709.07822, 2017.

[5] V. Arvind and P. Mukhopadhyay, *Derandomizing the isolation lemma and lower bounds for circuit size*, in Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, Springer, Berlin, 2008, pp. 276–289, https://doi.org/10.1007/978-3-540-85363-3_23.

[6] D. A. M. Barrington, *Quasipolynomial size circuit classes*, in Proceedings of the 7th Annual Structure in Complexity Theory Conference, IEEE, Washington, DC, 1992, pp. 86–93, https://doi.org/10.1109/SCT.1992.215383.

[7] P. W. Beame, S. A. Cook, and H. J. Hoover, *Log depth circuits for division and related problems*, SIAM J. Comput., 15 (1986), pp. 994–1003, https://doi.org/10.1137/0215070.

[8] S. J. Berkowitz, *On computing the determinant in small parallel time using a small number of processors*, Inform. Process. Lett., 18 (1984), pp. 147–150, https://doi.org/10.1016/0020-0190(84)90018-8.

[9] A. Borodin, S. Cook, and N. Pippenger, *Parallel computation for well-endowed rings and space-bounded probabilistic machines*, Inform. and Control, 58 (1983), pp. 113–136, https://doi.org/10.1016/S0019-9958(83)80060-6.

[10] S. Chari, P. Rohatgi, and A. Srinivasan, *Randomness-optimal unique element isolation with applications to perfect matching and related problems*, SIAM J. Comput., 24 (1995), pp. 1036–1050, https://doi.org/10.1137/S0097539793250330.

[11] Z.-Z. Chen and M.-Y. Kao, *Reducing randomness via irrational numbers*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 1997, pp. 200–209, https://doi.org/10.1145/258533.258583.

[12] L. Csanky, *Fast parallel matrix inversion algorithms*, SIAM J. Comput., 5 (1976), pp. 618–623, https://doi.org/10.1137/0205040.

[13] E. Dahlhaus and M. Karpinski, *Matching and multidimensional matching in chordal and strongly chordal graphs*, Discrete Appl. Math., 84 (1998), pp. 79–91, https://doi.org/10.1016/S0166-218X(98)00006-7.

[14] S. Datta, R. Kulkarni, and S. Roy, *Deterministically isolating a perfect matching in bipartite planar graphs*, Theory Comput. Syst., 47 (2010), pp. 737–757, https://doi.org/10.1007/s00224-009-9204-8.

[15] E. Dekel, D. Nassimi, and S. Sahni, *Parallel matrix and graph algorithms*, SIAM J. Comput., 10 (1981), pp. 657–675, https://doi.org/10.1137/0210049.

[16] R. A. Demillo and R. J. Lipton, *A probabilistic remark on algebraic program testing*, Inform. Process. Lett., 7 (1978), pp. 193–195, https://doi.org/10.1016/0020-0190(78)90067-4.

[17] J. Edmonds, *Paths, trees, and flowers*, Canad. J. Math., 17 (1965), p. 449–467, https://doi.org/10.4153/CJM-1965-045-4.

[18] V. Faber and D. G. Harris, *Tight Bounds and Conjectures for the Isolation Lemma*, preprint, https://arxiv.org/abs/1604.07035, 2016.

[19] S. Fenner, R. Gurjar, and T. Thierauf, *Bipartite perfect matching is in quasi-NC*, in Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016) (Cambridge, MA), ACM, New York, 2016, pp. 754–763, https://doi.org/10.1145/2897518.2897564.

[20] M. L. Fredman, J. Komlós, and E. Szemerédi, *Storing a sparse table with $O(1)$ worst case access time*, J. ACM, 31 (1984), pp. 538–544, https://doi.org/10.1145/828.1884.
[21] S. Goldwasser and O. Grossman, *Perfect Bipartite Matching in Pseudo-Deterministic RNC*, in Electronic Colloquium on Computational Complexity, 2015, TR15-208, https://eccc.weizmann.ac.il/report/2015/208.
[22] S. Goldwasser and O. Grossman, *Bipartite perfect matching in pseudo-deterministic NC*, in Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, Warsaw, Poland, 2017, 87, https://doi.org/10.4230/LIPIcs.ICALP.2017.87.
[23] D. Grigoriev and M. Karpinski, *The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract)*, in Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, Washington, DC, 1987, pp. 166–172, https://doi.org/10.1109/SFCS.1987.56.
[24] R. Gurjar, A. Korwar, J. Messner, and T. Thierauf, *Exact perfect matching in complete graphs*, ACM Trans. Comput. Theory, 9 (2017), 8, https://doi.org/10.1145/3041402.
[25] 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, Canada), ACM, New York, 2017, pp. 821–830, https://doi.org/10.1145/3055399.3055440.
[26] R. Gurjar, T. Thierauf, and N. K. Vishnoi, *Isolating a Vertex via Lattices: Polytopes with Totally Unimodular Faces*, preprint, https://arxiv.org/abs/1708.02222, 2017.
[27] V. A. T. Kallampally and R. Tewari, *Trading determinism for time in space bounded computations*, in 41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, Kraków, Poland, 2016, 10, https://doi.org/10.4230/LIPIcs.MFCS.2016.10.
[28] D. R. Karger and R. Motwani, *An NC algorithm for minimum cuts*, SIAM J. Comput., 26 (1997), pp. 255–272, https://doi.org/10.1137/S0097539794273083.
[29] R. M. Karp, E. Upfal, and A. Wigderson, *Constructing a perfect matching is in random NC*, Combinatorica, 6 (1986), pp. 35–48, https://doi.org/10.1007/BF02579407.
[30] M. Karpinski and W. Rytter, *Fast Parallel Algorithms for Graph Matching Problems*, Oxford University Press, Oxford, UK, 1998.
[31] P. W. Kasteleyn, *Graph theory and crystal physics*, in Graph Theory and Theoretical Physics, Academic Press, London, 1967, pp. 43–110.
[32] A. Klivans and D. A. Spielman, *Randomness efficient identity testing of multivariate polynomials*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 2001, pp. 216–223, https://doi.org/10.1145/380752.380801.
[33] A. Lingas and M. Karpinski, *Subtree isomorphism is NC reducible to bipartite perfect matching*, Inform. Process. Lett., 30 (1989), pp. 27–32, https://doi.org/10.1016/0020-0190(89)90170-1.
[34] L. Lovász, *On determinants, matchings, and random algorithms*, in Fundamentals of Computation Theory, Akademie-Verlag, Berlin, 1979, pp. 565–574.
[35] L. Lovász and M. D. Plummer, *Matching Theory*, North–Holland Math. Stud. 121, North–Holland, Amsterdam, 1986.
[36] M. Mahajan and K. R. Varadarajan, *A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 2000, pp. 351–357, https://doi.org/10.1145/335305.335346.
[37] G. L. Miller and J. (S.) Naor, *Flow in planar graphs with multiple sources and sinks*, SIAM J. Comput., 24 (1995), pp. 1002–1017, https://doi.org/10.1137/S0097539789162997.
[38] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, *Matching is as easy as matrix inversion*, Combinatorica, 7 (1987), pp. 105–113, https://doi.org/10.1007/BF02579206.
[39] M. Nair, *On Chebyshev-type inequalities for primes*, Amer. Math. Monthly, 89 (1982), pp. 126–129.
[40] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Upper Saddle River, NJ, 1982.
[41] P. Sankowski, *NC Algorithms for Weighted Planar Perfect Matching and Related Problems*, preprint, https://arxiv.org/abs/1709.07869, 2017.
[42] J. T. Schwartz, *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM, 27 (1980), pp. 701–717, https://doi.org/10.1145/322217.322225.
[43] O. Svensson and J. Tarnawski, *The matching problem in general graphs is in quasi-NC*, in Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017) (Berkeley, CA), IEEE, Washington, DC, 2017, pp. 696–707, https://doi.org/10.1109/FOCS.2017.70.

[44] N. Ta-Shma, *A simple proof of the Isolation Lemma*, in Electronic Colloquium on Computational Complexity, 2015, TR15-080, https://eccc.weizmann.ac.il/report/2015/080/.
[45] C. P. Teo and K. M. Koh, *The number of shortest cycles and the chromatic uniqueness of a graph*, J. Graph Theory, 16 (1992), pp. 7–15, https://doi.org/10.1002/jgt.3190160103.
[46] R. Tewari and N. Vinodchandran, *Green's theorem and isolation in planar graphs*, Inform. and Comput., 215 (2012), pp. 1–7, https://doi.org/10.1016/j.ic.2012.03.002.
[47] V. V. Vazirani, *NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems*, Inform. and Comput., 80 (1989), pp. 152–164, https://doi.org/10.1016/0890-5401(89)90017-5.
[48] S. Vishwanathan, *Personal communication*, 2001.
[49] R. Zippel, *Probabilistic algorithms for sparse polynomials*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation (EUROSAM), Springer-Verlag, Berlin, 1979, pp. 216–226, https://doi.org/10.1007/3-540-09519-5_73.