

Exact Matching: Correct Parity and FPT Parameterized by Independence Number

Nicolas El Maalouly  

Department of Computer Science, ETH Zürich, Switzerland

Raphael Steiner  

Department of Computer Science, ETH Zürich, Switzerland

Lasse Wulf  

Institute of Discrete Mathematics, TU Graz, Austria

Abstract

Given an integer k and a graph where every edge is colored either red or blue, the goal of the exact matching problem is to find a perfect matching with the property that exactly k of its edges are red. Soon after Papadimitriou and Yannakakis (JACM 1982) introduced the problem, a randomized polynomial-time algorithm solving the problem was described by Mulmuley et al. (Combinatorica 1987). Despite a lot of effort, it is still not known today whether a deterministic polynomial-time algorithm exists. This makes the exact matching problem an important candidate to test the popular conjecture that the complexity classes P and RP are equal. In a recent article (MFCS 2022), progress was made towards this goal by showing that for bipartite graphs of bounded bipartite independence number, a polynomial time algorithm exists. In terms of parameterized complexity, this algorithm was an XP-algorithm parameterized by the bipartite independence number. In this article, we introduce novel algorithmic techniques that allow us to obtain an FPT-algorithm. If the input is a general graph we show that one can at least compute a perfect matching M which has the correct number of red edges modulo 2, in polynomial time. This is motivated by our last result, in which we prove that an FPT algorithm for general graphs, parameterized by the independence number, reduces to the problem of finding in polynomial time a perfect matching M with at most k red edges and the correct number of red edges modulo 2.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Perfect Matching, Exact Matching, Independence Number, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.28

Related Version *Full Version*: <https://arxiv.org/abs/2207.09797>

Funding *Raphael Steiner*: Supported by an ETH Zurich Postdoctoral Fellowship.

Lasse Wulf: Supported by the Austrian Science Fund (FWF): W1230.

1 Introduction

In the *Exact Matching Problem* (denoted from now on by EM), we are given a graph G together with a fixed coloring of its edges in two colors (red and blue). The question is, for a given integer k , to decide whether there exists a perfect matching M of G with the additional property that exactly k of the edges of the perfect matching M are red. Clearly, if we have the special case that all edges of the graph are red and $k = |V(G)|/2$ then this problem is simply to decide whether there exists a perfect matching in the graph, which is well-known to be decidable in polynomial time [7]. However, when the coloring of the edges is heterogeneous, the problem difficulty seems to increase significantly (see below).



© Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf;
licensed under Creative Commons License CC-BY 4.0

34th International Symposium on Algorithms and Computation (ISAAC 2023).

Editors: Satoru Iwata and Naonori Kakimura; Article No. 28; pp. 28:1–28:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Papadimitriou and Yannakakis [28] initially introduced EM in 1982 and conjectured it to be NP-hard. However, a randomized polynomial-time algorithm solving the problem was described by Mulmuley, Vazirani and Vazirani in 1987 in the course of their celebrated isolation lemma [26]. Given standard complexity theoretic hypotheses, this makes it unlikely for EM to be NP-hard. Despite the existence of a polynomial-time randomized algorithm, as of today it is still not known whether EM can also be solved in deterministic polynomial time. The algorithm of Mulmuley et al. uses polynomial identity testing and is based on the Schwartz-Zippel Lemma [29, 35], which has resisted all attempts of derandomization so far. Indeed, EM is one of the few natural problems which has a randomized polynomial-time algorithm (i.e. it is contained in the complexity class RP) but for which it is not known whether it admits a deterministic polynomial-time algorithm (i.e. it is contained in P). It is a major open conjecture that $RP=P$, and so EM becomes a natural candidate to test this hypothesis.

For this reason, EM has been cited in several papers as an open problem. This includes recent breakthrough papers such as the seminal work on the parallel computation complexity of the matching problem [31], works on planarizing gadgets for perfect matchings [17], works on budgeted, color bounded, or constrained matching problems [3, 22, 24, 25, 30], on multicriteria optimization problems [16] and on matroid intersection problems [5]. It is further known that several different problems relate directly or indirectly to EM. The following is a non-exhaustive list of examples: EM is polynomial-time equivalent to the DNA sequencing problem [4]. EM is equivalent to a variant of the problem of finding a solution of a binary linear equation system with small Hamming weight [2]. EM can be reduced to a special case of the recoverable robust assignment problem [12].

Previous work. Progress in finding deterministic algorithms for EM (and therefore finding positive evidence for the conjecture $P=RP$) has only been made for restricted graph classes: It is known that EM can be solved in deterministic polynomial time for planar and more generally $K_{3,3}$ -minor free graphs [34], as well as graphs of bounded genus [13]. These works use Pfaffian orientations to derandomize the algebraic technique from [26]. EM can also be solved for graphs of bounded treewidth using a dynamic programming approach [8, 32]. In contrast to these classes of sparse graphs, EM on dense graphs seems to be even harder: Already solving the problem on complete graphs and complete bipartite graphs is highly nontrivial. In fact, at least 4 articles just dealing with this special case have appeared in the literature [14, 18, 21, 33]. Recent work [9] made a step forward by showing how to solve EM on graphs of constant independence number, where the *independence number* of a graph G is defined as the largest number α such that G contains an *independent set* of size α , and bipartite graphs of constant bipartite independence number, where the *bipartite independence number* of a bipartite graph G equipped with a bipartition of its vertices is defined as the largest number β such that G contains a *balanced independent set* of size 2β , i.e., an independent set using exactly β vertices from both color classes. This generalizes previous results for complete and complete bipartite graphs which correspond to the special cases $\alpha = 1$ and $\beta = 0$. The authors presented an XP-algorithm, i.e. an algorithm running in time $O(n^{f(\alpha)})$, for the problem. The existence of an FPT algorithm, i.e. an algorithm with running time $f(\alpha)n^{O(1)}$, was left as an open question. The authors also conjectured that counting perfect matching is #P-hard for this class of graphs. This conjecture was later proven in [11] already for $\alpha = 2$ or $\beta = 3$. As a consequence, the Pfaffian derandomization technique is unlikely to work for this class of graphs, because this technique implicitly counts the number of perfect matchings. This makes the graph class of graphs of bounded

independence number a promising frontier to push the limits of deterministic techniques. To the best of our knowledge, these are the only results from the last 40 years showing that EM can be solved in deterministic poly-time for restricted graph classes.

Apart from restricted graph classes, one can also consider parameterized algorithms for EM, using the natural parameter k . Note that an XP-algorithm in this case is trivial to obtain using brute-force guessing (guess the red edges that go in a solution and complete the perfect matching using only blue edges). An FPT algorithm would, however, be highly desirable as it is likely provide a lot of insight into EM. The only progress towards that goal can be found in [8] where some color coding tools were developed but only applied to the almost trivial case of bounded circumference graphs.

Another direction of progress towards solving EM is the study of relaxed versions of it. A first such relaxation would be to lift the requirement for a perfect matching. In [34], however, it was shown that there is a simple deterministic polynomial time algorithm such that given a “Yes” instance of EM, computes an *almost* perfect matching (i.e. of size at least $\frac{n}{2} - 1$) containing k red edges. This result is as close to optimal as possible for this type of relaxation. The study of the other type of relaxation, i.e. relaxing the color constraints, was only recently initiated in [8]. It was shown that there is a deterministic polynomial time algorithm which given a “Yes”-instance of EM outputs a perfect matching with k' red edges, such that $k/2 \leq k' \leq 3k/2$.

Exact matchings modulo 2. A crucial tool in this paper is to consider matchings with k' red edges, where $k' \equiv_2 k$, that is, matchings of the correct parity. Let $r(M)$ denote the number of red edges in a matching M . We define the *Correct Parity Matching Problem* (CPM), where given a red-blue edge-colored graph and an integer k , the goal is to find a perfect matching M such that $r(M) \equiv_2 k$. Note that parity problems (and more general congruency-constrained problems) have been studied in the context of other graph algorithms [19, 27], but are not well studied for the perfect matching problem. A more challenging version of the problem, *Bounded Correct Parity Matching* (BCPM), requires finding a perfect matching M such that $r(M) \equiv_2 k$ and $r(M) \leq k$. In [20] the complexity of the EM problem was even further highlighted by showing that the Exact Matching polytope has exponential extension complexity even when restricted to the bipartite case and to the parity constraint (i.e. CPM in bipartite graphs has exponential extension complexity).

Our results. From now on, when we say that an algorithm has *polynomial running time*, we mean a deterministic algorithm, whose running time is bounded by $\text{poly}(n)$, even if α, β, k are not bounded by a constant. We show in this paper how BCPM can be used to solve EM. Precisely, our results are the following:

- We show that EM reduces to BCPM in FPT time parameterized by α in the following sense: There exists an algorithm, which performs a single oracle call to BCPM, and solves EM on general graphs, in running time $f(\alpha)n^{O(1)}$. (The result holds analogously for the bipartite independence number β). Without access to the BCPM oracle, the algorithm outputs a perfect matching with either $k - 1$ or k red edges or deduces that the answer of the given EM-instance is “No” (Section 3).
- CPM can be solved in polynomial time for all graphs (Section 4). This insight is based on a deep result by Lovász [23].
- On bipartite graphs, the more difficult problem BCPM can be solved in polynomial time (Theorem 17). As a consequence, there is an FPT algorithm parameterized by β which solves EM on bipartite graphs (Theorem 2).

Due to space restrictions, proofs of statements marked \star can be found in the extended version of this paper [10].

2 Preliminaries

All graphs considered are simple. For $G = (V, E)$, we let $V(G) := V$ and $E(G) := E$. We always use the letter n to denote the number of vertices of a graph G , i.e. $n = |V(G)|$. An *edge-colored* graph is a tuple (G, col) , where $\text{col} : E \rightarrow \{\text{red}, \text{blue}\}$ prescribes a color to each edge. An instance of EM is a tuple (G, col, k) . Given an instance of EM and a perfect matching (abbreviated PM) M , we define edge weights $w_M : E \rightarrow \mathbb{N}$ as follows: We have $w_M(e) = 0$ if e is a blue edge, $w_M(e) = +1$ if e is a red non-matching edge and $w_M(e) = -1$ if e is a red matching edge. The weight function w_M plays a critical role in many arguments in this paper. When the PM M can be deduced from context, we may write w instead of w_M . In this case, the *weight* of edge e is $w_M(e)$. For G' a subgraph of G , we define $R(G')$ (resp. $B(G')$) to be the set of red (resp. blue) edges in G' , $r(G') := |R(G')|$ to be the number of red edges of G' and $w_M(G')$ to be the sum of the weights of edges in G' . If \mathcal{C} is a set of vertex-disjoint cycles, then we define $w_M(\mathcal{C}) = \sum_{C \in \mathcal{C}} w_M(C)$.

We say that a set of disjoint cycles or paths is M -alternating if for any two adjacent edges in the set, one of them is in M and the other is not. Undirected cycles are considered to have an arbitrary orientation. For a cycle C and $u, v \in C$, $C[u, v]$ is defined as the path from u to v along C (in the fixed but arbitrarily chosen orientation). The term $\text{Ram}(r, s)$ refers to the Ramsey number, i.e. every graph on $\text{Ram}(r, s)$ vertices contains either a clique of size r or an independent set of size s . For simplicity we will use the following upper bound: $\text{Ram}(s + 1, s + 1) < 4^s$ [15].

3 Reducing EM to BCPM in FPT time

The goal of this section is to prove our two main theorems:

► **Theorem 1.** *EM can be reduced to BCPM in FPT time parameterized by the independence number of the graph.*

► **Theorem 2.** *There exists an FPT algorithm for EM on bipartite graphs parameterized by the bipartite independence number of the graph.*

We will first introduce the algorithm and then prove Theorem 1 in Section 3.4 and Theorem 2 in Section 3.5. Finally, in Section 3.6 we will discuss the case where a BCPM oracle can not be used.

3.1 Tools from Prior Work

The algorithm we develop to prove Theorems 1 and 2 will rely on many of the tools developed in [9] and [8]. We start with the two main propositions that we aim to use. The setting of both propositions is the same: We are given some PM M explicitly, and we know that there is another PM M' which we know exists, but we do not know explicitly. We are given the PM M and the number $r(M')$ as input and would like to find either M' itself, or at least another PM M'' with $r(M'') = r(M')$.

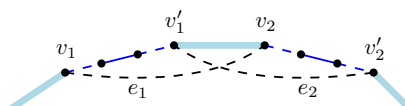
► **Proposition 3** (from [9]). *Let M and M' be two PMs in G such that $|B(M \Delta M')| \leq L$ or $|R(M \Delta M')| \leq L$, for $L \geq 1$. Then there exists a deterministic algorithm running in time $n^{O(L)}$ such that given M and $r(M')$, it outputs a PM M'' with $r(M'') = r(M')$.*

► **Proposition 4** (adapted from [8]). (★) *Given a graph $G = (V, E)$ with edge colors red and blue, let M and M' be two PMs in G such that $|E(M\Delta M')| \leq L$, for $L \geq 1$. Then there exists an algorithm running in time $f(L)n^{O(1)}$ (for $f(L) = L^{O(L)}$) such that given M and $r(M')$, it outputs a PM M'' with $r(M'') = r(M')$.*

The algorithm from Proposition 4 is faster (FPT instead of XP when parameterized by L), but it requires more assumptions on M' . The algorithm from Proposition 3 works by guessing which L edges are in $R(M\Delta M')$ (respectively $B(M\Delta M')$) and then checks if the red (blue) edges can be completed to a PM by using only blue (red) edges. The algorithm from Proposition 4 works using color-coding technique (see [6, Chapter 5] for more details on color coding).

In [9] the authors show that for graphs of small independence number, one could use Proposition 3 to get an XP algorithm (parameterized by the independence number) by bounding either the number of red edges or the number of blue edges in the symmetric difference with a target matching M' . Our aim is to show that we can use the stronger Proposition 4 from [8] to get an FPT algorithm, which would require that we bound both color classes (i.e. the entire symmetric difference). This turns out to be much more difficult to achieve and requires novel algorithmic techniques that we describe in the next section. Our algorithm does, however, start by bounding one of the color classes before bounding the second. For that we simply rely on the tools developed in [9] to avoid starting from scratch. Due to the technicality of some of the used tools, some readers might want to skip the details of the tools from previous work and jump ahead to the next section, only coming back to these definitions and lemmas when needed.

A crucial concept to understand the tools from prior work is a property of the weight function $w = w_M$ as defined in Section 2. Let M and M' be two perfect matchings. It is well-known that the symmetric difference $\mathcal{C} := M\Delta M'$ is a set of edges that forms a vertex-disjoint union of M -alternating cycles. An easy observation is now that $w_M(\mathcal{C})$ counts the difference of red edges between M and M' , that is, we have $r(M') = r(M) + w_M(\mathcal{C})$. This follows directly from the definition of w_M . The second crucial concept is the concept of a *skip*.



■ **Figure 1** A skip formed by two non-matching edges e_1 and e_2 (in black). Matching edges are normal lines, non-matching edges are dashed. The bold lines represent subpaths.

► **Definition 5** (from [9]). *Let M be a PM and C an M -alternating cycle. A skip S is a set of two non-matching edges $e_1 := (v_1, v_2)$ and $e_2 := (v'_1, v'_2)$ with $e_1, e_2 \notin C$ and $v_1, v'_1, v_2, v'_2 \in C$ (appearing in this order along C) such that $C' = e_1 \cup e_2 \cup C \setminus (C[v_1, v'_1] \cup C[v_2, v'_2])$ is an M -alternating cycle, $|C| - |C'| > 0$ and $|w_M(S)| \leq 4$ where $w_M(S) := w_M(C') - w_M(C)$ is called the weight of the skip.*

Let M, C, S, C' be as above. We say that *using the skip S* is the action of replacing the alternating cycle C by the alternating cycle C' . If furthermore M' is another PM and $C \in M\Delta M'$, then we say that using S also modifies M' the following way: We let $M'_{\text{new}} = M'\Delta C\Delta C' = M\Delta((M\Delta M') \setminus C) \cup C'$. In other words, M'_{new} is the matching which has the same symmetric difference from M as M' , except that C was replaced by C' . It is an easy observation that M'_{new} is again a PM and $r(M'_{\text{new}}) = r(M') + w(S)$. This means

that using a positive skip (i.e. a skip of strictly positive weight) increases the cycle weight, using a negative skip decreases it and using a 0 -skip (i.e. a skip of weight 0) does not change the cycle weight. Using a skip always results in a cycle of smaller cardinality. If $P \subseteq C$ is a path and $C[v_1, v'_2] \subseteq P$, then we say that P contains the skip S . Two skips $\{(v_1, v_2), (v'_1, v'_2)\}$ and $\{(u_1, u_2), (u'_1, u'_2)\}$ are called disjoint if they are contained in disjoint paths along the cycle. Note that two disjoint skips can be used independently. Finally, observe that iterating over all skips of a given alternating cycle C can be done in polynomial time by trying all possible combinations of two chords from the cycle C and checking whether they form a skip. This means that if a skip with certain properties is shown to exist, it can also be found in polynomial time.

► **Definition 6** (from [9]). *Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles. A 0 -skip set with respect to \mathcal{C} is a set of disjoint skips on cycles of \mathcal{C} such that the total weight of the skips is 0 .*

► **Definition 7** (from [9]). *Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles. A 0 -skip-cycle set with respect to \mathcal{C} is a set of disjoint skips on cycles of \mathcal{C} and/or cycles from \mathcal{C} , such that the total weight of the skips minus the total weight of the cycles is 0 .*

We say that *using a skip-cycle set \mathcal{S}* means to change \mathcal{C} by removing all cycles in \mathcal{S} from \mathcal{C} and by using all skips in \mathcal{S} (i.e. for every $S \in \mathcal{S}$ that is a skip, locate the corresponding cycle $C \in \mathcal{C}$ with S in C and use S on C). A perfect matching M'_{new} is defined in an analogous way as M'_{new} is defined for using a single skip (i.e., such that $M'_{\text{new}} = M \Delta \mathcal{C}_{\text{new}}$). If \mathcal{S} was a 0 -skip-cycle set, then we have $r(M'_{\text{new}}) = r(M')$. Using a 0 -skip-cycle set always decreases the total size of \mathcal{C} . In this paper, it will be a common strategy to locate 0 -skip-cycle sets contained in the symmetric difference $M \Delta M'$ of two PMs. If we manage to find such a 0 -skip-cycle set, then using it on $M \Delta M'$ will produce a new PM M'_{new} such that $r(M'_{\text{new}}) = r(M')$, but $|M \Delta M'_{\text{new}}| < |M \Delta M'|$. Hence we make progress in the sense that we reduce the symmetric difference $M \Delta M'$ while maintaining $r(M')$.

The following lemmas are taken and adapted from [9]. They show that under certain assumptions 0 -skip sets or 0 -skip-cycle sets always exist. They are adapted to also include a proof that the desired objects can be found in polynomial time. We leave their adapted proofs to the appendix.

► **Lemma 8** (adapted from [9]). (\star) *Let M be a PM and P an M -alternating path with $w_M(P) \geq 2t \cdot 4^\alpha$ (resp. $w_M(P) \leq -2t \cdot 4^\alpha$), for $t \geq 1$, then P contains at least t disjoint negative (resp. positive) skips. If P and M are given, then we can also find t such skips in polynomial time.*

► **Lemma 9** (adapted from [9]). (\star) *Let $t \geq 8 \cdot 4^\alpha$ and $t' = 4t^2$. Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles and $C \in \mathcal{C}$ such that $|w_M(C)| \leq t'$ and $|w_M(C)| \geq 2t'$, then \mathcal{C} contains a 0 -skip-cycle set. If \mathcal{C}, M are given, we can also find a 0 -skip-cycle set in polynomial time.*

► **Lemma 10** (adapted from [9]). (\star) *Let $t \geq 3$. Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles such that $|w_M(C)| \leq t$, $|w_M(C)| \leq 2t$ for all $C \in \mathcal{C}$ and $|\mathcal{C}| \geq 10t^3$, then \mathcal{C} contains a 0 -skip-cycle set. If \mathcal{C}, M are given, we can also find a 0 -skip-cycle set in polynomial time.*

► **Lemma 11** (adapted from [9]). (\star) *Let $t \geq 8 \cdot 4^\alpha$. Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles such that $|\mathcal{C}| \leq 10t^3$, $|w_M(C)| \leq 2t$ for all $C \in \mathcal{C}$ and \mathcal{C} contains at least $1000t^6$ blue edges and $1000t^6$ red edges, then \mathcal{C} contains a 0 -skip set. If \mathcal{C}, M are given, then we can also find a 0 -skip set in polynomial time.*

3.2 The Main Algorithm

The aim of this section is to present the algorithm which reduces EM to BCPM in time $f(\alpha)n^{O(1)}$. We first sketch the idea of the algorithm: We assume that the algorithm receives two PMs M and M' as input, such that $r(M) < k < r(M')$ and such that both M and M' already have the correct parity, i.e. $r(M) \equiv_2 r(M') \equiv_2 k$. We will later show how this can be done with an oracle call to BCPM. But even in the case where the BCPM oracle can not be used and M, M' are just some arbitrary PMs with $r(M) < k < r(M')$, our algorithm still computes something meaningful: We show that in this case a PM with k or $k - 1$ red edges will be output, or it will be deduced that the given EM-instance has answer “No”. This variant of the algorithm is further discussed in Section 3.6.

Our algorithm modifies the PMs M and M' many times. But the invariant is maintained that during the whole execution of the algorithm, both the PMs M and M' will never change their parity. The basic idea of the algorithm is to have many iterations, where in each iteration either M is modified such that $r(M)$ increases by 2, or M' is modified such that $r(M')$ decreases by two. Clearly, if we can do such a modification in every iteration, we will eventually arrive at a PM with k red edges. One might ask why we consider modifications of the kind $+2$ and -2 , instead of the kind $+1$ and -1 . The reason for this is that a change of ± 1 might not always be possible, even in complete graphs. To see this, consider the smallest possible modification of a PM. It consists in taking its symmetric difference with an alternating cycle of length four. Such a cycle may add or remove up to two red edges from the matching and it is possible that we only find such cycles adding or removing exactly two red edges. On the converse, if all small cycles add or remove one red edge, we can still achieve a change of two by simply considering two such cycles.

However, reality is more complicated and even a ± 2 modification might not always be possible. The first hurdle is that such a modification might not be possible if $r(M) \ll r(M')$. To combat this hurdle, the algorithm splits into three phases, where in the first phase the PMs M, M' are modified such that they keep their parity and after their modification we have that $r(M), r(M')$ are close to k . Details for phase 1 will be provided in Lemma 12. In the second phase, we will do many iterations, such that in each iteration the algorithm tries to (i) increase $r(M)$ by 2, or (ii) decrease $r(M')$ by 2, or (iii) strictly decrease the cardinality of the symmetric difference $|E(M \Delta M')|$. Finally, it can still happen that neither (i), (ii), or (iii) are possible. However, we prove a key lemma which states that in this situation (and if the given EM-instance is a “Yes” instance), we can use color coding techniques to find a PM M^* in time $f(\alpha)n^{O(1)}$ which is a solution to EM, i.e. $r(M^*) = k$. The algorithm then enters phase 3, where it either finds M^* or deduces that the given EM-instance is a “No”-instance. We now provide the formal description of the algorithm:

Input: A red-blue edge-colored graph (G, col) , a nonnegative integer k . Two PMs M and M' with $r(M) < k < r(M')$.

Phase 1: Find two PMs M_{new} and M'_{new} such that

$$k - 8 \cdot 4^\alpha \leq r(M_{\text{new}}) \leq k \leq r(M'_{\text{new}}) \leq k + 8 \cdot 4^\alpha,$$

and such that the parity is maintained, i.e. $r(M_{\text{new}}) \equiv_2 r(M)$ and $r(M'_{\text{new}}) \equiv_2 r(M')$. Set $M \leftarrow M_{\text{new}}$ and $M' \leftarrow M'_{\text{new}}$. If this step fails, output “EM-instance has no solution”.

Phase 2: If either M or M' is a solution matching we are done. Otherwise repeat the following three steps until either M or M' is a solution matching or until every step (i),(ii), and (iii) fails in the same iteration:

- (i) Invoke the algorithm of Proposition 3 with respect to the matching M and $L = 2$ in order to try to find a PM M_{new} with $r(M_{\text{new}}) = r(M) + 2$. If such a PM is found, let $M \leftarrow M_{\text{new}}$, otherwise do not modify M and consider step (i) as failed.
- (ii) Invoke the algorithm of Proposition 3 with respect to the matching M' and $L = 2$ in order to try to find a PM M'_{new} with $r(M'_{\text{new}}) = r(M') - 2$. If such a PM is found, let $M' \leftarrow M'_{\text{new}}$, otherwise do not modify M' and consider step (ii) as failed.
- (iii) Invoke the algorithms of Lemma 9, Lemma 10 or Lemma 11 (with $t = 256 \cdot 4^{2\alpha}$), to try to find a 0-skip or a 0-skip-cycle set in $M\Delta M'$. If such an object is found, then use it (i.e. change M' accordingly) to reduce $|E(M\Delta M')|$. Otherwise do not modify M, M' and consider step (iii) as failed.

Phase 3: If either M or M' is a solution matching we are done. Otherwise invoke the algorithm of Proposition 4 with $L = 2^{\alpha^{O(1)}}$ (for appropriately large constants) on the matching M to try to find a PM M^* with $r(M^*) = k$. If such a PM M^* is found, then output it. Otherwise output “EM-instance has no solution”.

This completes the description of the algorithm. The remainder of this section is dedicated to its proof. First, we prove that phase 1 can be completed correctly in polynomial time (Lemma 12). It is not so difficult to prove that phase 2 requires only polynomial time (as there are at most n^2 iterations). Finally, we prove in our main lemma (Lemma 15) that if steps (i),(ii),(iii) all fail simultaneously, then phase 3 is guaranteed to succeed. This is the most difficult lemma to prove. In Section 3.4 we summarize the proof and explain how to obtain the two initial matchings M and M' required as input for phase 1.

Finally, we describe the modifications necessary for bipartite graphs (Section 3.5) and for cases where the BCPM oracle is not available (Section 3.6).

3.3 Proof of the main lemmas

The following lemmas help us prove the correctness and polynomial running time of the algorithm.

► **Lemma 12.** *Given a “Yes” instance of EM and two PMs M and M' with $r(M) \leq k \leq r(M')$, there exists a deterministic polynomial time algorithm that outputs two PMs M_1 and M_2 with $r(M_1) \equiv_2 r(M)$, $r(M_2) \equiv_2 r(M')$ and $k - 8 \cdot 4^\alpha \leq r(M_1) \leq k \leq r(M_2) \leq k + 8 \cdot 4^\alpha$.*

Proof. As long as $r(M) < k - 8 \cdot 4^\alpha$ we will consider two cases:

- All cycles $C \in M\Delta M'$ have weight $w_M(C) \leq 4 \cdot 4^\alpha$. In this case $M\Delta M'$ must contain at least two strictly positive cycles C_1 and C_2 . If $w_M(C_1) \equiv_2 0$ then we replace M by $M\Delta C_1$ and iterate (note that $r(M) < r(M\Delta C_1) \leq k$ and $r(M\Delta C_1) \equiv_2 r(M)$). The case $w_M(C_2) \equiv_2 0$ is similar. Otherwise we replace M by $M\Delta(C \cup C')$ and iterate (note that $r(M) < r(M\Delta(C \cup C')) \leq k$ and $r(M\Delta(C \cup C')) \equiv_2 r(M)$).
- There exists $C \in M\Delta M'$ with $w_M(C) > 4 \cdot 4^\alpha$. Observe that $C \in M'\Delta M$ and $w_{M'}(C) = -w_M(C) \leq -4 \cdot 4^\alpha$. By Lemma 8 applied to M' and $w_{M'}$, we have that C contains two positive skips (with respect to M' and $w_{M'}$). If any of the skips has even weight, we use it to increase the weight of $w_{M'}(C)$ and iterate (note that $r(M)$ increases since using a skip in $M'\Delta M$ modifies M). Otherwise we use both skips. In either case, $r(M)$ increases and its parity is preserved. Note that $r(M)$ can increase by at most 8 given that a skip must have weight at most 4 by definition.

In both cases $r(M)$ increases after every iteration. So there can be at most $O(n)$ iterations, each running in polynomial time, until $k - 8 \cdot 4^\alpha \leq r(M) \leq k$. Now we apply a similar procedure to decrease $r(M')$. As long as $r(M') > k + 8 \cdot 4^\alpha$ we will consider two cases:

- All cycles in $M' \Delta M$ have weight $w_{M'}$ more than $-4 \cdot 4^\alpha$. In this case $M' \Delta M$ must contain at least two strictly negative cycles C_1 and C_2 . If $w_{M'}(C_1) \equiv_2 0$ then we replace M' by $M' \Delta C_1$ and iterate (note that $k \leq r(M' \Delta C_1) < r(M')$ and $r(M' \Delta C_1) \equiv_2 r(M')$). The case $w_{M'}(C_2) \equiv_2 0$ is similar. Otherwise we replace M' by $M' \Delta (C \cup C')$ and iterate (note that $k \leq r(M \Delta (C \cup C')) < r(M')$ and $r(M \Delta (C \cup C')) \equiv_2 r(M')$).
- There exists $C \in M' \Delta M$ with $w_{M'}(C) < -4 \cdot 4^\alpha$. Observe that $C \in M \Delta M'$ with $w_M(C) = -w_{M'}(C) \geq 4 \cdot 4^\alpha$. By Lemma 8 applied to M and w_M , C contains two negative skips (with respect to M and w_M). If any of the skips has even weight, we use it to reduce $w_M(C)$ and iterate (note that $r(M')$ decreases since using a skip in $M \Delta M'$ modifies M'). Otherwise we use both skips. In either case, $r(M')$ decreases and its parity is preserved. Note that $r(M')$ can decrease by at most 8 given that a skip must have weight at least -4 by definition.

In both cases $r(M')$ decreases after every iteration. So there can be at most $O(n)$ iterations, each running in polynomial time, until $k \leq r(M') \leq k + 8 \cdot 4^\alpha$. Finally the algorithm terminates by outputting $M_1 := M$ and $M_2 := M'$. ◀

► **Lemma 13.** *Let M be a PM and \mathcal{C} a set of disjoint M -alternating cycles with the following properties:*

- \mathcal{C} does not contain monochromatic cycles.
- $|E(\mathcal{C})| \geq 2t^3$.
- $|R(\mathcal{C})| \leq t$ (resp. $|B(\mathcal{C})| \leq t$).

Then \mathcal{C} contains a blue (resp. red) M -alternating path of length at least t .

Proof. We will consider the case when $|R(\mathcal{C})| \leq t$. The case $|B(\mathcal{C})| \leq t$ is proven similarly by swapping the two colors. First observe that if \mathcal{C} contains at most t red edges and no monochromatic cycles, then $|\mathcal{C}| \leq t$. So by the pigeonhole principle, \mathcal{C} must contain a cycle C with $|E(C)| \geq 2t^2$. Consider the set of maximal blue subpaths of C and let p_B be the number of these paths. As every such path is accompanied by a red edge, we have $p_B \leq t$. Finally, C has at least $2t^2 - t$ blue edges, so by the pigeonhole principle one of the blue paths must have length at least $(2t^2 - t)/t \geq t$. ◀

The above lemma simply shows that if only one color class is bounded, there must be long monochromatic paths of the other color. The next lemma shows that the existence of long monochromatic paths in turn implies the existence of small cycles.

► **Lemma 14.** *Let M be a PM and C an M -alternating cycle. Let $P \subseteq C$ be a blue (resp. red) M -alternating path of length at least $6 \text{Ram}(\text{Ram}(4, \alpha + 1), \alpha + 1)$, starting with a non-matching edge and not containing 0-skips. Then there must be two edges $e_1 := (b_1, b_2)$ and $e_2 := (w_1, w_2)$ with endpoints on P , at least one of which must be red (resp. blue), such that $C' = e_1 \cup e_2 \cup C[b_1, w_1] \cup C[b_2, w_2]$ is an M -alternating cycle with $0 < w_M(C') \leq 2$ (resp. $-2 \leq w_M(C') < 0$) and containing a number of red (resp. blue) edges equal to the absolute value of its weight.*

Proof. We will only deal with the case when P is blue, the other case is treated similarly (by switching the roles of the two colors in the proof). We assume that P has an arbitrary orientation which is used to define the start and end vertices of subpaths of P . First, we divide P into a set of consecutive paths \mathcal{P} of length 6 each, starting with the first non-matching edge. Let \mathcal{P}_1 be the set of paths formed by the first 3 edges of each path in \mathcal{P} . The set of start vertices of paths in \mathcal{P}_1 has size at least $\text{Ram}(\text{Ram}(4, \alpha + 1), \alpha + 1)$ so it must contain a clique Q of size $\text{Ram}(4, \alpha + 1)$. Let \mathcal{P}_2 be the set of paths in \mathcal{P}_1 with start vertices in Q . The set of end vertices of paths in \mathcal{P}_2 must contain a clique Q' of size 4 (see Figure 2). Let

$\mathcal{P}_3 := \{P_1, P_2, P_3, P_4\}$ be the set of paths in \mathcal{P}_1 with end vertices in Q' . Let s_i and t_i be the start and end vertices of path P_i for $i \in \{1, 2, 3, 4\}$. Observe that any two distinct paths $P_i, P_j \in \mathcal{P}_3$ have their endpoints connected by the edges (s_i, s_j) and (t_i, t_j) and a skip is created this way. If both edges were blue, we would get a 0-skip (since the whole path P has only blue edges). Letting $i = 2, j = 3$, we see that one of the edges (s_2, s_3) or (t_2, t_3) must be red. Suppose (s_2, s_3) is red. Observe that $(t_1, t_2) \cup (s_2, s_3) \cup C[t_1, s_2] \cup C[t_2, s_3]$ is a cycle of weight $+1$ or $+2$ (depending on whether (t_1, t_2) is red or blue, since all other edges are blue) and containing at most 2 red edges. Similarly, suppose (t_2, t_3) is red. Observe that $(t_2, t_3) \cup (s_3, s_4) \cup C[t_2, s_3] \cup C[t_3, s_4]$ is a cycle of weight $+1$ or $+2$ (depending on whether (s_3, s_4) is red or blue) and the number of red (resp. blue) edges it contains is equal to the absolute value of its weight. \blacktriangleleft

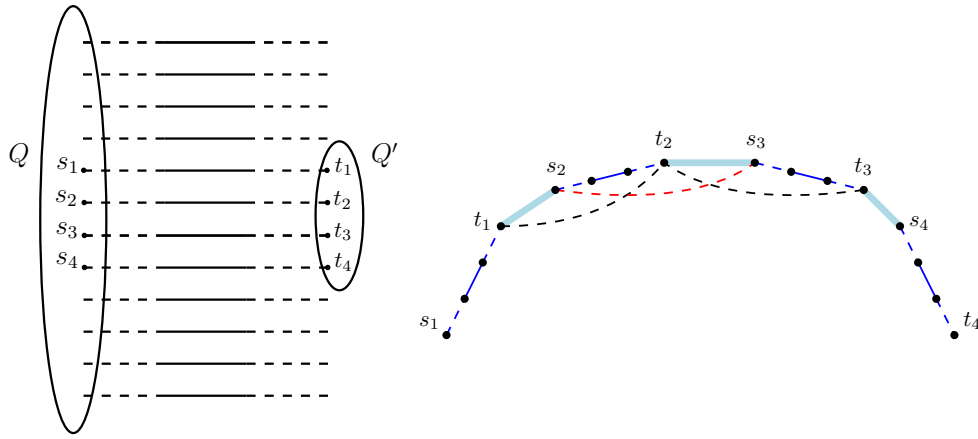


Figure 2 Left: The set of paths \mathcal{P}_2 , of size $\text{Ram}(4, \alpha + 1)$, and the cliques Q and Q' . Right: The paths from \mathcal{P}_3 along the blue path P and the vertices s_i, t_j . Matching edges are normal lines, non-matching edges are dashed. The bold lines between t_i and s_{i+1} represent subpaths. Observe that $\{(s_2, s_3), (t_2, t_3)\}$ forms a skip and $(t_1, t_2) \cup (s_2, s_3) \cup C[t_1, s_2] \cup C[t_2, s_3]$ is an alternating cycle.

Finally, we are ready to prove our main lemma. Roughly speaking, it states that if all steps (i),(ii) and (iii) in phase two of the algorithm fail, then phase 3 is guaranteed to succeed. More specifically, it states that if we cannot make small progress towards a solution then we are ready to apply Proposition 4 and find one in FPT time. Small progress here means either getting the number of red edges in M or M' closer to k , or making their symmetric difference smaller.

► Lemma 15. *Let M and M' be two PMs with the following properties:*

- (a) $r(M) < k - 1, r(M') > k$.
- (b) $|w_M(M \Delta M')| \leq t$ for $t = 256 \cdot 4^{2\alpha}$.
- (c) *There is no PM M_1 such that $r(M_1) = r(M) + 2$ and $|R(M \Delta M_1)| = 2$.*
- (d) *There is no PM M'_1 such that $r(M'_1) = r(M') - 2$ and $|B(M' \Delta M'_1)| = 2$.*
- (e) $M \Delta M'$ does not contain any 0-skip.
- (f) *The algorithms of Lemma 9, Lemma 10 and Lemma 11 all fail to find a 0-skip-cycle set in $M \Delta M'$.*

If there is at least one PM with k red edges, then there exists a PM M^ such that $r(M^*) = k$ and $|E(M \Delta M^*)| = 2^{\alpha^{O(1)}}$.*

Proof. We start by giving a high level overview and the intuition behind the proof. First we note that properties (a) and (b) will be guaranteed after phase 1 of the algorithm and they state that both M and M' are close to k in terms of number of red edges. Second, properties (c) and (d) state that our algorithm is unable to make small progress in terms of getting the number of red edges in M or M' closer to k . Finally properties (e) and (f) state that our algorithm is unable to make small progress in term of making the symmetric difference between M and M' smaller.

Our final goal is to bound the symmetric difference between M and some solution M^* . We will do that by contradiction to one of the given properties. Observe that if the conditions for Lemma 14 are met, i.e. there is a long blue only M -alternating path, then the lemma guarantees that small progress towards getting the number of red edges closer to k is possible. The same holds for long red only M' -alternating paths. Note, however, that we might need to apply the lemma twice in order to ensure that the progress is in increments or decrements of two, thus contradicting either property (c) or (d). This way we can bound the length of blue only M -alternating paths. Then if red only M -alternating paths are also bounded, Lemma 13 implies that either both colors are bounded, in which case we are done, or none of them is. In the latter case, we use the machinery developed in [9] (see Section 3.1) to reach the contradiction (remember that the goal there was to bound one color class), and this requires properties (a) and (b) to hold. The same holds if blue only M' -alternating paths are bounded.

The only remaining obstacles are long red only M -alternating or blue only M' -alternating paths. To deal with that, we try to reduce the symmetric difference between M and M' such that long monochromatic M -alternating paths are also M' -alternating (and the contradiction above can again be reached) since the two matchings do not differ by that many edges. Bounding the symmetric difference between M and M' relies on a contradiction to properties (e) or (f). It follows the same steps as bounding the symmetric difference between M and M^* , but with the added benefit that paths in this symmetric difference are both M and M' -alternating, which avoids the problem of long red only M -alternating or blue only M' -alternating paths.

To summarise, we start by bounding $|E(M\Delta M')|$ (first bounding one color class, then the second). Then we are able to bound $|E(M\Delta M^*)|$ (again one color class at a time).

Detailed proof. We will start by showing that one color class of $M\Delta M'$ must be bounded. This allows us to then bound $|E(M\Delta M')|$. We then consider the solution matching M^* that minimizes $|E(M\Delta M^*)|$ and start by bounding the number of blue edges in $M\Delta M^*$. Finally, we also show that the number of red edges in $M\Delta M^*$ is bounded, thus bounding $|E(M\Delta M^*)|$.

Bounding one color class of $M\Delta M'$. Since we failed to reduce $|E(M\Delta M')|$ using the algorithm of Lemma 9, the weight of all cycles in $M\Delta M'$ must be bounded: $|w(C)| \leq 2t$ for all $C \in M\Delta M'$. Since we failed to reduce $|E(M\Delta M')|$ using the algorithm of Lemma 10, the number of cycles in $M\Delta M'$ must be bounded: $|M\Delta M'| \leq 10t^3$. Finally, since we failed to reduce $|E(M\Delta M')|$ using the algorithm of Lemma 11, $|B(M\Delta M')|$ or $|R(M\Delta M')|$ must be bounded (by $1000t^6$). Let $t' = \max(1000t^6, 20 \text{Ram}(\text{Ram}(4, \alpha + 1), \alpha + 1))$ (note that $1000t^6 = 2^{O(\alpha)}$, $\text{Ram}(4, \alpha + 1) = \alpha^{O(1)}$ and $\text{Ram}(\text{Ram}(4, \alpha + 1), \alpha + 1) = 2^{\alpha^{O(1)}}$ so $t' = 2^{\alpha^{O(1)}}$).

Bounding $|E(M\Delta M')|$. First, we show that property (c) implies that there is no blue M -alternating path of length at least t' in the graph. Suppose such a path exists. Divide the path into two blue paths P_1 and P_2 of length at least $t'/2$ each. From Lemma 14 applied to each of the paths P_1 and P_2 , we get that there exists two disjoint M -alternating cycles C_1 and C_2 with $0 < w_M(C_1) \leq 2$, $0 < w_M(C_2) \leq 2$ and each containing a number of red edges equal to the absolute value of their weight. If C_1 contains two red edges, let $M_1 := M\Delta C_1$. Otherwise if C_2 contains two red edges, let $M_1 := M\Delta C_2$. Finally, if both C_1 and C_2 contain only one red edge, let $M_1 := M\Delta(C_1 \cup C_2)$. Observe that $|R(M\Delta M_1)| = 2$ and $r(M_1) = r(M) + 2$, contradicting property (c).

Next we show that property (d) implies that there is no red M' -alternating path of length at least t' in the graph. Divide the path into two red paths P_1 and P_2 of length at least $t'/2$ each. From Lemma 14 applied to each of the paths P_1 and P_2 , we get that there exists two disjoint M -alternating (with respect to M') cycles C_1 and C_2 with $-2 \leq w_{M'}(C_1) < 0$, $-2 \leq w_{M'}(C_2) < 0$ and each containing a number of blue edges equal to the absolute value of their weight. If C_1 contains two blue edges, let $M'_1 := M'\Delta C_1$. Otherwise if C_2 contains two blue edges, let $M'_1 := M'\Delta C_2$. Finally, if both C_1 and C_2 contain only one blue edge, let $M'_1 := M'\Delta(C_1 \cup C_2)$. Observe that $|B(M'\Delta M'_1)| = 2$ and $r(M'_1) = r(M') - 2$, contradicting property (d).

Suppose $|R(M\Delta M')| \geq 2t'^3$. Then by the previous paragraph $|B(M\Delta M')| \leq t'$. Note that $M\Delta M'$ contains no monochromatic cycle, as this would be a 0-skip cycle set, therefore by Lemma 13, $M'\Delta M$ contains a red M' -alternating path of length at least t' . But this contradicts property (c). Now suppose $|B(M\Delta M')| \geq 2t'^3$. Then $|R(M\Delta M')| \leq t'$ and by Lemma 13, $M\Delta M'$ contains a blue M -alternating path of length at least t' , contradicting property (c). So we get $|E(M\Delta M')| = |B(M\Delta M')| + |R(M\Delta M')| \leq 4t'^3$.

Bounding $|B(M\Delta M^*)|$. Now let M^* among all those PMs with k red edges be the one which minimizes $|E(M\Delta M^*)|$. Note that $|w_M(M\Delta M^*)| \leq |w_M(M\Delta M')| \leq t$. Since $|E(M\Delta M^*)|$ is minimal, $M\Delta M^*$ cannot contain a 0-skip-cycle set. By Lemma 9, the weight of all cycles in $M\Delta M^*$ must be bounded: $|w(C)| \leq 2t$ for all $C \in M\Delta M^*$. By Lemma 10, the number of cycles in $M\Delta M^*$ must be bounded: $|M\Delta M^*| \leq 10t^3$. Finally, by Lemma 11, $|B(M\Delta M^*)|$ or $|R(M\Delta M^*)|$ must be bounded (by $1000t^6 \leq t'$). Suppose $|B(M\Delta M^*)| > 2t'^3$. So $|R(M\Delta M^*)| \leq t'$ and by Lemma 13, $M\Delta M^*$ contains a blue M -alternating path of length t' , contradicting property (c). So $|B(M\Delta M^*)| \leq 2t'^3$.

Bounding $|E(M\Delta M^*)|$. Let $t'' = 4t'^4$. Suppose $|R(M\Delta M^*)| > 2t''^3$, by Lemma 13 $M^*\Delta M$ contains a red path P with $|P| \geq t''$. Observe that $M'\Delta M^* = (M\Delta M^*)\Delta(M\Delta M')$ and $P \subseteq M\Delta M^*$ so

$$P \cap (M'\Delta M^*) = P \setminus (P \cap (M\Delta M')).$$

We have $|P \cap (M\Delta M')| \leq |E(M\Delta M')| < 4t'^3$, so if all paths in $P \cap (M'\Delta M^*)$ have length at most t' then $|P| < 4t'^4$, a contradiction. So there must be a path $P' \subseteq P \cap (M'\Delta M^*)$ of length at least t' . Note that P' is a red M' -alternating path, contradicting property (d). So we have $|E(M\Delta M^*)| \leq 4t''^3 = t''^{O(1)} = t'^{O(1)} = 2^{\alpha^{O(1)}}$. ◀

3.4 Main theorem for general graphs

Proof of Theorem 1. Suppose we have a polynomial time oracle for BCPM. We start by solving CPM on the given instance. This can be done in polynomial time (as we prove later in Theorem 19) and will give us a PM M_p with $r(M_p) \equiv_2 k$. If $r(M_p) \geq k$, let $M' := M_p$

and use an oracle call to BCPM to get M with $r(M) \equiv_2 k$ and $r(M) \leq k$. Otherwise, if $r(M_p) \geq k$, let $M := M_p$ and use an oracle call to BCPM to get M' with $r(M') \equiv_2 k$ and $r(M') \geq k$ (this can simply be done by swapping the red and blue colors and using $k' = n/2 - k$ as parameter for the BCPM oracle). In both cases, we obtain PMs M, M' such that $r(M) \equiv_2 k \equiv_2 r(M')$ and $r(M) \leq k \leq r(M')$.

Note that if this step fails (in the sense that the CPM or BCPM call returns “false”), then the EM instance has no solution. Otherwise we apply the algorithm of Section 3.2 on the EM-instance with M and M' as input. Our goal now is to prove that if the EM-instance is a “YES” instance, then the following must be true:

- (a) Phase 1 runs in polynomial time and outputs two PMs M and M' such that $k - 8 \cdot 4^\alpha \leq r(M) \leq k \leq r(M') \leq k + 8 \cdot 4^\alpha$, $r(M) \equiv_2 r(M') \equiv_2 k$.
- (b) Phase 2 runs in polynomial time and either outputs a PM with k red edges (and the algorithm terminates) or a PM M such that there exists a PM M^* with $r(M^*) = k$ and $|E(M\Delta M^*)| \leq 2^{\alpha^{O(1)}}$ (for appropriately large constants).
- (c) If the algorithm did not terminate in Phase 2, then Phase 3 runs in time $f(\alpha)n^{O(1)}$ and outputs a PM with k red edges.

It is easy to see that if all the above items hold, then the algorithm runs in time $f(\alpha)n^{O(1)}$ and always outputs a PM with k red edges if one exists. Note that (a) and (c) follow directly from Lemma 12 and Proposition 4 respectively.

To prove (b) first observe that as long as $r(M) \neq k$ and $r(M') \neq k$, all steps in phase 2 maintain the following invariants: $r(M) \leq k \leq r(M')$ and $r(M) \equiv_2 r(M') \equiv_2 k$. To see this, note that $r(M)$ and $r(M')$ can only change by 2 every step and they start with the same parity as k . So in order for $r(M)$ to go above k or $r(M')$ to go below k they would need to pass by k , at which point the algorithm terminates. Also observe that if any of the steps does not fail, then either $r(M') - r(M)$ decreases or $|E(M\Delta M')|$ decreases while $r(M') - r(M)$ remains unchanged. So if we consider as a measure of progress $r(M') - r(M)$ and $|E(M\Delta M')|$ ordered lexicographically (where progress is towards smaller values of the measure), then we always make progress (i.e. the measure strictly decreases). Note that $r(M') - r(M) \leq n$ and is always non-negative and the same holds for $|E(M\Delta M')|$. So the algorithm can perform at most n^2 iterations in phase 2. Since every iteration runs in polynomial time (this is true for steps (i) and (ii) by Proposition 3 and for step (iii) by Lemma 9, Lemma 10 and Lemma 11), we get that phase 2 runs in polynomial time. Now observe that the algorithm only terminates in phase 2 if either M or M' is a solution (i.e. it has k red edges). So it remains to show that if the algorithm does not terminate in this phase then there exists a PM M^* with $r(M^*) = k$ and $|E(M\Delta M^*)| \leq 2^{\alpha^{O(1)}}$. Observe that in case of non-termination, all the conditions of Lemma 15 are met:

- (a) $r(M) < k - 1$, $r(M') > k$: follows from the invariants and M, M' not being solutions.
- (b) $|w_M(M\Delta M')| \leq 256 \cdot 4^{2\alpha}$: follows from $r(M') - r(M) \leq 16 \cdot 4^\alpha$.
- (c) There is no PM M_1 such that $r(M_1) = r(M) + 2$ and $|R(M\Delta M_1)| = 2$: follows from the failure of (i).
- (d) There is no PM M'_1 such that $r(M'_1) = r(M') - 2$ and $|B(M'\Delta M'_1)| = 2$: follows from the failure of (ii).
- (e) $M\Delta M'$ does not contain any 0-skip: follows from the failure of (iii).
- (f) The algorithms of Lemma 9, Lemma 10 and Lemma 11 all fail to find a 0-skip-cycle set in $M\Delta M'$: follows from the failure of (iii).

So by Lemma 15 we get the desired result. ◀

3.5 Main theorem for bipartite graphs

In order to prove the main theorem for the bipartite case (Theorem 2), we start by proving a similar result to the main theorem on general graph that is adapted to bipartite graphs, i.e., we use the bipartite independence number of the graph (the proof can be found in the extended version [10]).

► **Lemma 16.** (\star) *EM on bipartite graphs can be reduced to BCPM on bipartite graphs in FPT time parameterized by the bipartite independence number of the graph.*

It remains to show that there is a deterministic polynomial time algorithm for BCPM on bipartite graphs. This result can be derived from the more general result of [1] on network matrices, as noted in [20], even for the more general weighted version of the problem. To make it more accessible, we reprove it using a standard dynamic programming techniques. The high level approach, as briefly described in [20], is the following: start by computing a minimum weight perfect matching, in our case a perfect matching with minimum number of red edges, and if the number of red edges is even then find a minimum odd weight alternating cycle and output the symmetric difference. We could not find formal proof of correctness and running time for this algorithm in the literature, therefore we provide one in the extended version of this paper [10].

► **Theorem 17.** (\star) *There is a deterministic polynomial time algorithm for BCPM on bipartite graphs.*

3.6 Main theorem without oracle access

Although an FPT algorithm parameterized by the independence number for general graphs still requires an algorithm for BCPM, the following theorem shows that without relying on BCPM the algorithm developed in this section can still output a PM that is very close to optimal, i.e. it contains either k or $k - 1$ red edges (the proof is similar to that of Theorem 1 and left for the extended version [10]).

► **Theorem 18.** (\star) *There exists an algorithm such that given a “Yes” instance of EM, it outputs a perfect matching with either $k - 1$ or k red edges in time $f(\alpha)n^{O(1)}$.*

This strengthens the results of [8] by reducing the constraint violation to at most one red edge at the expense of an FPT (parameterized by the independence number) instead of a polynomial running time.

4 Correct Parity Matching for General Graphs

While solving BCPM for general graphs remains an open problem, in this section we present a solution to the easier problem of CPM which is only concerned with the parity of the number of red edges.

► **Theorem 19.** (\star) *There is a deterministic polynomial time algorithm for CPM.*

We will establish Theorem 19 as a consequence of a deep result by Lovász [23] on the *linear hull* of perfect matchings of a graph. We first need to introduce some notation which we adopt from [23]. Let a (not necessarily bipartite) graph $G = (V, E)$ and a field \mathbb{F} be given, and let us denote by \mathcal{M} the set of perfect matchings of G . Then the linear hull of perfect matchings $\text{lin}_{\mathbb{F}}(\mathcal{M})$ is the linear subspace of \mathbb{F}^E , generated by the characteristic vectors of

perfect matchings in G . Concretely, $\text{lin}_{\mathbb{F}}(\mathcal{M})$ is the linear span of $\{1_M \mid M \in \mathcal{M}\}$, where for every perfect matching M the vector $1_M \in \mathbb{F}^E$ is defined by $1_M(e) = 1$ for every $e \in M$ and $1_M(e) = 0$ for every $e \in E \setminus M$.

We will make use of the following result of Lovász [23].

► **Theorem 20** ([23]). *For every finite field \mathbb{F} there is a deterministic polynomial-time algorithm that, given as input a graph G , returns a linear basis of $\text{lin}_{\mathbb{F}}(\mathcal{M})$.*

The importance of this result by Lovász for solving the CPM is explained through the following lemma. As usual, for two vectors $x, y \in \mathbb{F}^E$ we denote by $\langle x, y \rangle := \sum_{e \in E} x_e y_e \in \mathbb{F}$ their scalar product.

► **Lemma 21.** *Let $G = (V, E)$ be a graph equipped with a coloring of its edges with colors red and blue. Let \mathbb{F}_2 denote the 2-element field and let $\{x_1, \dots, x_d\} \subset \mathbb{F}_2^E$ be a linear basis of $\text{lin}_{\mathbb{F}_2}(\mathcal{M})$. Let $r \in \mathbb{F}_2^E$ be defined by $r_e := 1$ for all red edges $e \in E$ and $r_e = 0$ for all blue edges $e \in E$. Then the following two statements are equivalent:*

1. *There exists a perfect matching M in G containing an odd number of red edges.*
2. *There exists $i \in \{1, \dots, d\}$ such that $\langle x_i, r \rangle = 1$.*

Proof. Suppose first that there exists a perfect matching M in G containing an odd number of red edges. Then the incidence vector $1_M \in \text{lin}_{\mathbb{F}_2}(\mathcal{M})$ can be represented as a linear combination of the basis elements x_1, \dots, x_d , in other words, there exists $I \subseteq \{1, \dots, d\}$ such that

$$1_M = \sum_{i \in I} x_i.$$

Taking scalar products with r we get

$$\langle 1_M, r \rangle = \sum_{i \in I} \langle x_i, r \rangle.$$

Note that the scalar product on the left hand side equals the number of red edges in M taken modulo 2, and hence it equals 1. But then at least one of the scalar products on the right hand side must also be non-zero, i.e., there exists $i \in I$ with $\langle x_i, r \rangle = 1$, as desired.

Conversely, suppose there exists $i \in \{1, \dots, d\}$ such that $\langle x_i, r \rangle = 1$. Then by virtue of $\text{lin}_{\mathbb{F}_2}(\mathcal{M})$ being spanned by the characteristic vectors of perfect matchings in M , there exists a list of perfect matchings M_1, \dots, M_t in G such that $x_i = \sum_{j=1}^t 1_{M_j}$. Using the same argument as above, i.e., by taking scalar products with r and using that $\langle x_i, r \rangle = 1$, we find that there must exist $j \in \{1, \dots, t\}$ with $\langle 1_{M_j}, r \rangle = 1$, which means that M_j is a perfect matching of G with an odd number of red edges. This concludes the proof. ◀

We may now deduce the following.

► **Corollary 22.** *There exists a deterministic polynomial-time algorithm, that, given as input a red-blue edge-colored graph $G = (V, E)$ and a number $k \in \mathbb{Z}$, decides whether or not G contains a perfect matching M with $r(M) \equiv_2 k$.*

Proof. Suppose first that k is odd. We use Theorem 20 to compute in deterministic polynomial time a linear basis x_1, \dots, x_d of $\text{lin}_{\mathbb{F}_2}(\mathcal{M})$. Note that since $\text{lin}_{\mathbb{F}_2}(\mathcal{M})$ is a subspace of \mathbb{F}_2^E , its dimension satisfies $d \leq |E|$. Next we generate the incidence vector r of red edges as in the previous lemma, and compute the scalar products $\langle x_i, r \rangle$ for $i = 1, \dots, d$ in polynomial time. If at least one of these products equals 1, we return that a perfect matching M with $r(M) \equiv_2 k$ exists, and otherwise we return that such a matching does not exist. The correctness of this output follows by Lemma 21.

Next suppose k is even. Let G' be the red, blue-edge colored graph which is obtained as the disjoint union of G with its given edge-coloring and a disjoint new edge of color red. The perfect matchings of G' are exactly the perfect matchings of G together with the additional new red edge, and hence G contains a perfect matching M with $r(M) \equiv_2 k$ if and only if G' contains a perfect matching with an odd number of red edges. Thus we can decide whether such a matching exists by invoking the algorithm from the case $k = 1$ described above with G' as the input. ◀

It is now easy to use the above decision-version of the CPM to solve the CPM itself by a standard edge-deletion procedure.

Proof of Theorem 19. Let $G = (V, E)$ be the input graph with a given red, blue-edge coloring, and let further $k \in \mathbb{Z}$ be given. We use Corollary 22 to decide if G contains a perfect matching M with $r(M) \equiv_2 k$. If it does not, then the algorithm stops with this conclusion. Otherwise, we search through the edges $e \in E$ one by one, and for each such edge test (again using Corollary 22) whether $G - e$ contains a perfect matching M with $r(M) \equiv_2 k$.

Suppose first we find an edge $e \in E$ such that $G - e$ contains a perfect matching M with $r(M) \equiv_2 k$. In this case we make a recursive call of the algorithm to $G - e$, which will return a perfect matching with the correct parity in $G - e$. We can then return this matching, as it is also a perfect matching in G with the correct parity of red edges.

Otherwise, we find that there exists no $e \in E$ such that $G - e$ contains a perfect matching M with $r(M) \equiv_2 k$. But as we know that G does contain a perfect matching M with $r(M) \equiv_2 k$, this means that all edges of G are contained in M , and hence we may return the set of edges of G and thereby find a solution to the CPM. ◀

5 Conclusion and Open Problems

So far, EM has only been solved for very sparse graphs (i.e. bounded tree-width and bounded genus graphs) and very dense graphs (i.e. bounded independence number graphs). The techniques used are quite different between these two cases. Especially in the case of dense graphs, many previous works considered only complete (bipartite) graphs without much progress. Only recently, the results were extended to the case of bounded independence number, leading to XP algorithms parameterized by α or β . Looking for FPT algorithms was the natural next step. In this paper, we could resolve the bipartite case fully, while the non-bipartite case could only be resolved partially. However, our results in the non-bipartite case still yield the following two non-trivial, independent insights: (i) To obtain an FPT algorithm parameterized by α it suffices to solve BCPM (Theorem 1) and the easier problem CPM can always be solved (Theorem 19). (ii) An FPT algorithm parameterized by α can w.l.o.g. assume to start with a PM with $k - 1$ red edges (Theorem 18).

We hope that these insights can be the starting point for future work to obtain an FPT algorithm parameterized by α , or, even better, an FPT algorithm parameterized by k . The latter would be considered quite a breakthrough as it is likely to require a lot of deep understanding of the structure and patterns behind EM, given the difficulty of making progress towards it.

References

- 1 Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1206–1219, 2017.
- 2 Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Solving linear equations parameterized by hamming weight. *Algorithmica*, 75(2):322–338, 2016.

- 3 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128(1):355–372, 2011.
- 4 Jacek Błazewicz, Piotr Formanowicz, Marta Kasprzak, Petra Schuurman, and Gerhard J Woeginger. A polynomial time equivalence between DNA sequencing and the exact perfect matching problem. *Discrete Optimization*, 4(2):154–162, 2007.
- 5 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13(2):258–273, 1992.
- 6 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshтанov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.
- 7 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- 8 Nicolas El Maalouly. Exact matching: Algorithms and related problems. *arXiv preprint*, 2022. [arXiv:2203.13899](https://arxiv.org/abs/2203.13899).
- 9 Nicolas El Maalouly and Raphael Steiner. Exact Matching in Graphs of Bounded Independence Number. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:14, 2022.
- 10 Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf. Exact matching: Correct parity and FPT parameterized by independence number. *CoRR*, abs/2207.09797, 2022. doi: 10.48550/arXiv.2207.09797.
- 11 Nicolas El Maalouly and Yanheng Wang. Counting perfect matchings in dense graphs is hard. *arXiv preprint*, 2022. [arXiv:2210.15014](https://arxiv.org/abs/2210.15014).
- 12 Dennis Fischer, Tim A Hartmann, Stefan Lendl, and Gerhard J Woeginger. An investigation of the recoverable robust assignment problem. *arXiv preprint*, 2020. [arXiv:2010.11456](https://arxiv.org/abs/2010.11456).
- 13 Anna Galluccio and Martin Loebl. On the theory of Pfaffian orientations. I. Perfect matchings and permanents. *Electronic Journal of Combinatorics*, 6:R6, 1999.
- 14 Hans-Florian Geerdes and Jácint Szabó. A unified proof for Karzanov’s exact matching theorem. Technical Report QP-2011-02, Egerváry Research Group, Budapest, 2011.
- 15 Ronald L Graham, Bruce L Rothschild, and Joel H Spencer. *Ramsey theory*, volume 20. John Wiley & Sons, 1991.
- 16 Fabrizio Grandoni and Rico Zenklusen. Optimization with more than one budget. *arXiv preprint*, 2010. [arXiv:1002.2147](https://arxiv.org/abs/1002.2147).
- 17 Rohit Gurjar, Arpita Korwar, Jochen Messner, Simon Straub, and Thomas Thierauf. Planarizing gadgets for perfect matching do not exist. In *International Symposium on Mathematical Foundations of Computer Science*, pages 478–490. Springer, 2012.
- 18 Rohit Gurjar, Arpita Korwar, Jochen Messner, and Thomas Thierauf. Exact perfect matching in complete graphs. *ACM Transactions on Computation Theory (TOCT)*, 9(2):1–20, 2017.
- 19 Edith Hemaspaandra, Holger Spakowski, and Mayur Thakur. Complexity of cycle length modularity problems in graphs. In *Latin American Symposium on Theoretical Informatics*, pages 509–518. Springer, 2004.
- 20 Xinrui Jia, Ola Svensson, and Weiqiang Yuan. The exact bipartite matching polytope has exponential extension complexity. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1635–1654. SIAM, 2023.
- 21 AV Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics*, 23(1):8–13, 1987.
- 22 Steven Kelk and Georgios Stamoulis. Integrality gaps for colorful matchings. *Discrete Optimization*, 32:73–92, 2019.
- 23 László Lovász. Matching structure and the matching lattice. *Journal of Combinatorial Theory, Series B*, 43:187–222, 1987.

- 24 Monaldo Mastrolilli and Georgios Stamoulis. Constrained matching problems in bipartite graphs. In *International Symposium on Combinatorial Optimization*, pages 344–355. Springer, 2012.
- 25 Monaldo Mastrolilli and Georgios Stamoulis. Bi-criteria and approximation algorithms for restricted matchings. *Theoretical Computer Science*, 540:115–132, 2014.
- 26 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.
- 27 Martin Nägele, Benny Sudakov, and Rico Zenklusen. Submodular minimization under congruency constraints. *Combinatorica*, 39(6):1351–1386, 2019.
- 28 Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.
- 29 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.
- 30 Georgios Stamoulis. Approximation algorithms for bounded color matchings via convex decompositions. In *International Symposium on Mathematical Foundations of Computer Science*, pages 625–636. Springer, 2014.
- 31 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-NC. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 696–707. Ieee, 2017.
- 32 Moshe Y Vardi and Zhiwei Zhang. Quantum-inspired perfect matching under vertex-color constraints. *arXiv preprint*, 2022. [arXiv:2209.13063](https://arxiv.org/abs/2209.13063).
- 33 Tongnyoung Yi, Katta G Murty, and Cosimo Spera. Matchings in colored bipartite networks. *Discrete Applied Mathematics*, 121(1-3):261–277, 2002.
- 34 Raphael Yuster. Almost exact matchings. *Algorithmica*, 63(1):39–50, 2012.
- 35 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International Symposium on Symbolic and Algebraic Computation (EUROSAM 1979)*, pages 216–226. Springer, 1979.