Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Exact Perfect Matching

Yanheng Wang

September 18, 2023

Master's Thesis submitted to
Department of Computer Science, ETH Zürich

under the supervision of
Prof. Dr. Bernd Gärtner,
Nicolas El Maalouly

Abstract

The Exact Perfect Matching problem is simple to state: Given a graph G whose edges are coloured either red or blue, is there a perfect matching that contains exactly k reds? Soon after its proposal in [21], an efficient randomised algorithm was found [20]. However, our combinatorial and structural understanding remained limited despite effort for four decades. Neither did people discover deterministic polynomial-time algorithms.

Sporadic progress were made for specific G, initially complete and complete bipartite graphs [14], later graphs of small independence number [6], and recently graphs with certain chordal properties [11]. One goal of the thesis is to present these (not-so-many) tools in a unified and accessible language. In particular, they are subsumed by what we call a *sandwich paradigm* which captures *continuity*. We supplement several new results relating to the paradigm.

Another goal is to depict the challenge behind Exact Perfect Matching. We offer our take from combinatorial, algebraic, and polyhedral perspectives. In addition to some classical material, we develop many cute results and counterexamples to sharpen our understanding of the affairs.

Acknowledgements

I would like to thank my advisor Nicolas for sharing his deep knowledge and optimism in the field, and for suggesting various routes of attack. Our weekly meetings have inspired and shaped the current thesis. I am grateful to Bernd for the many geometry-related discussions which led me to consider a polyhedral approach. I also benefit a lot from an optimisation course by Prof. Rico Zenklusen.

My sincere thanks go to my family for their constant support during my studies. And big thanks to Wenxin, for your encouragement and love.

Table of Contents

1 Introduction

Given some pairs of integers from the range $\{1, \ldots, 2n\}$, can you pick k pairs so that the sorted $2k$ integers alternate in parity? After some trial and error you might find the game challenging, as the pairs could be intertwined and thus every decision deeply correlates with other choices.

This game is a special case of our protagonist, the Exact Perfect Matching problem proposed by Papadimitriou and Yannakakis [21] four decades ago. Here you are given a graph G on $2n$ vertices, some edges being red while the others being blue. Can you pick k reds and $n-k$ blues so that they do not share any common vertex?[1.1]

Many equivalent formulations of the problem exist, some of which are summarised in Chapter 2. But it is most conveniently stated in terms of matchings. A *matching* in G is a set of disjoint edges. It is *perfect* if every vertex is touched. A perfect matching that contains exactly k reds (thus $n-k$ blues) is dubbed a k-PM. Now the problem asks: Does G have a k-PM?

Classical matching theory revolves around the Perfect Matching problem which asks if G has a perfect matching at all. The subject has matured over a century-long development: combinatorial tools found, structures revealed, and algorithms realised. It seems natural to assume that Exact Perfect Matching is no different, and this is utterly wrong. Let us start with two aspects that mark the peculiarity of this innocent-looking variant.

1.1 Discontinuity

The quirks of the problem can be illustrated by a simple instance. It consists of disjoint cycles, each of length four and coloured with "red-blue-red-blue". A perfect matching can pick either zero or two reds from a cycle, and the choices for different cycles are independent. Therefore, the graph admits a k-PM iff $k \in \{0, 2, 4, \ldots, n\}$; the feasibility regime contains plenty of "holes".

A slightly more general instance is the following. It consists of t disjoint cycles of lengths $2\,\ell_1, \ldots, 2\,\ell_t$ respectively. Every cycle is coloured with red and blue in alternation. So k-PM exists iff $k = \sum_{i \in I} \ell_i$ for some subset $I \subseteq \{1, \ldots, t\}$. This models the famous Subset Sum problem where the input values are polynomially bounded. Although we can solve such instance efficiently by a dynamic program, the picture for the feasibility regime is quite obscure. Understanding the structure of these subset sums is still an ongoing direction in additive combinatorics.

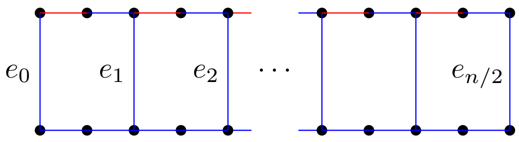
[1.1]. To model our motivational game, we simply link the integers $1, \ldots, 2n$ consecutively in a blue cycle and, for each given pair (a, b), connect the integers a and b by a red edge.

But these are just the tip of the iceberg! Consider a general graph whose cycles are tangled so that their edge choices are interdependent. It becomes less clear how things interact. The moral is that the feasibility of k-PM can be quite discontinuous in k. Knowing the existence of 2-PM and 5-PM, say, does not imply the existence of 3-PM or 4-PM.

One may ask how fractured the feasibility regime can be. Our following result brings the bad news: it can be as fractured as you want.

Theorem 1.1. Let $n \in 2\mathbb{N}$. Given any subset $K \subseteq \{0, 1, \ldots, {}^{n}\!/_{2}\}$, we can construct a bipartite graph on $2n$ vertices such that there exists a k-PM if and only if $k \in K$.

Proof. Consider the following bipartite graph G on $2n$ vertices:



Note that every perfect matching in G must use exactly one vertical edge e_k. Moreover, picking e_k would enforce all reds to its left and kill all reds to its right, thus leading to a k-PM.

Now we obtain a new graph G' from G by keeping the vertical edges $\{e_k : k \in K\}$ and removing the others. Then clearly G' has a k-PM if and only if $k \in K$. □

It is worth comparing a continuity result by Yuster [26]. Informally speaking, if we relax the stringent "perfectness" to "almost perfectness", i.e. allow the matching to take $n-1$ or n edges, then continuity holds.

Theorem 1.2. (Yuster) If a graph admits a k_1-PM and a k_2-PM, then it also admits an almost perfect matching with k reds, for all $k_1 \leqslant k \leqslant k_2$.

Continuity is the overarching theme in Chapters 3 and 4, and we will prove the theorem there after introducing necessary tools. In another pursuit, one can also stick to perfectness but try to identify graphs that exhibit certain amount of continuity. To this end, we will present an algorithmic paradigm inspired by [6, 11] that outputs a $(k \pm \varepsilon)$-PM for all $k_1 \leqslant k \leqslant k_2$. The error term ε quantifies the continuity of the input G. When G has constant modular width t, for example, we show that the error $\varepsilon \leqslant t/2$ is small. We also provide new theorems that can generate small-error graphs based on others. These results supplement previous work, but Theorem 1.1 clearly delineates their limitations and we cannot hope to generalise them to arbitrary graphs.

1.2 Lack of Negative Certificate

A decision problem is in **NP** if every `yes`-instance has an efficiently verifiable certificate. For example, Exact Perfect Matching is in **NP** because if G contains a k-PM then one can present a concrete solution as certificate, allowing others to verify quickly (in fact, in linear time). A decision problem is in **coNP** if every `no`-instance has an efficiently verifiable certificate. For example, the Perfect Matching problem is in **coNP** due to Tutte's theorem (see Theorem 1.5 later).

Note that $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$, so towards a suspected polynomial-time algorithm we must at least show that Exact Perfect Matching is in **coNP**. At present, however, such modest goal seems out of reach even when the graph is bipartite.

To get some insights, let us review the classical matching theorems that once provided **coNP** certificates for Perfect Matching. We begin with bipartite graph $G=(V,E)$ where $V=X\uplus Y$. How can one certify that it does not admit a perfect matching?

There are some quick criteria that exclude the existence of perfect matchings. For example, if we have found a subset $S\subseteq V$ of size $|S|<n$ to which all edges are incident, then no n edges can be disjoint. As another example, if we have found a subset $X_0\subseteq X$ whose neighbourhood is smaller than itself, i.e. $|N(X_0)|<|X_0|$, then by no means can we fully match every vertex in X_0.

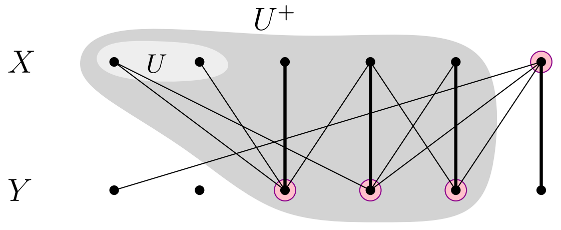
It turns out that both criteria are necessary, meaning that we can *always* use them to certify non-existence of perfect matching in a bipartite graph G.

Theorem 1.3. (Kőnig) A bipartite graph $G=(V,E)$ on $2n$ vertices does not admit a perfect matching iff there is subset $S\subseteq V$ of size $|S|<n$ to which all edges are incident.

We venture to provide a quick proof.

Proof. We have already argued about the ($\Leftarrow$) direction. For the ($\Rightarrow$) direction, write $V=X\uplus Y$ and fix M to be a maximum cardinality matching. Denote by $U\subseteq X$ the set of unmatched vertices in X.

Let U^+ be all the vertices reachable from U via a path that alternates between non-matching and matching edges. Note that such path always travels from X to Y via a non-matching edge, then from Y to X via a matching edge. We claim that $S:=(X\setminus U^+)\cup(Y\cap U^+)$ is the set that we are looking for.



First, every edge is incident to S. Suppose to the contrary that some edge xy is between $X\cap U^+$ and $Y\setminus U^+$, so x is reached via an alternating path whereas y is not. If $xy\in M$ then the path should have visited y prior to x. If $xy\notin M$ then the path can extend to y. Either case comes to a contradiction.

Second, all vertices in S are matched and so $|S|\leqslant|M|<n$. Indeed, $X\setminus U^+\subseteq X\setminus U$ is fully matched by definition. Also is $Y\cap U^+$ fully matched, because otherwise there is an alternating path P leading from an unmatched $x\in X$ to an unmatched $y\in Y$, and $(M\setminus P)\cup(P\setminus M)$ would be a larger matching. □

Theorem 1.4. (Hall) A bipartite graph $G=(X\uplus Y,E)$ does not contain a perfect matching iff there is a subset $X_0\subseteq X$ with $|N(X_0)|<|X_0|$.

Proof. We have already shown the ($\Leftarrow$) direction. For the ($\Rightarrow$) direction, we reuse the notation in the proof of Theorem 1.3 and take $X_0:=X\cap U^+$. Observe that $N(X_0)=Y\cap U^+$ is fully matched to vertices in X_0. But $X_0\supseteq U$ contains unmatched vertices as well, so we conclude $|N(X_0)|<|X_0|$. □

The X_0 in Hall's theorem and the S in Kőnig's theorem are "bottlenecks" that prevent perfect matching in a bipartite graph. Both can serve as a negative certificate. When we turn to general graphs, however, the X_0 is not defined any more. The S is still a sufficient criterion for ruling out perfect matchings, but it is no longer necessary (consider e.g. the triangle graph K_3). Hence we need to hunt for another notion that captures the bottleneck.

A trivial (though not necessary) condition that prevents perfect matching is the number of vertices being odd. Can we strengthen it? Suppose we have found a set $S \subseteq V$ whose removal from G results in several odd-cardinality components $H_1, \ldots, H_t$ (and possibly some even-cardinality components). Each H_i is not perfectly matchable internally, so it has to send at least one vertex to S for help. But if $|S| < t$ then the matching cannot be perfect, and such S is a bottleneck.

To rephrase, let $n_{\text{odd}}(H)$ denote the number of odd-cardinality components in graph H. Then any set $S \subseteq V$ with $|S| < n_{\text{odd}}(G - S)$ shall rule out the existence of a perfect matching in G. It turns out that this criterion is necessary, which we state without proof.

Theorem 1.5. (Tutte) A graph $G = (V, E)$ on $2n$ vertices does not admit a perfect matching iff there is a set S of size $|S| < n_{\text{odd}}(G - S)$.

The situation for Exact Perfect Matching is much more awkward. There do exist some obvious (and not necessary) criteria that rule out the existence of k-PM; for example if k is larger than the size of maximum matching in the red subgraph. But there is no good characterisation in sight. To our knowledge, only two criteria can lead to a negative certificate, but both have a brute-force nature:

- For every set of k disjoint red edges, apply Tutte's theorem to certify that the residual graph, obtained from removing all their endpoints and other red edges, is not perfectly matchable. This requires about $\binom{|E|}{k}$ encoding length and checking time.
- A smarter criterion is the following. If $G = (V, E)$ contains a k-PM, then we can map each selected red edge to its lower vertex. This gives a set of vertices $S \in \binom{V}{k}$ that match to $V \setminus S$ via only red edges. Hence the graph $G_S = (V, E_S)$ where $E_S := \{\text{red } uv : u \in S, v \notin S\} \cup \{\text{blue } uv : u \notin S, v \notin S\}$ should have a perfect matching. Thinking reversely, to refute the existence of k-PM, we may enumerate all subsets $S \in \binom{V}{k}$ and certify via Tutte's theorem that G_S is not perfectly matchable. This requires about $\binom{2n}{k}$ encoding length and checking time.

These certificates are very inefficient when $k = \omega(1)$; in fact exponential when $k = \Theta(n)$. It raises the question whether we can compress them combinatorially, which leads us to the big challenge to bake colours into classical matching theorems. The main difficulty lies in the type of argument we employ. All known proofs of the theorems of Kőnig's, Hall's and Tutte's use the "alternating path" machinery that we have already seen. Applying such path guides us from a suboptimal solution to a better one smoothly. In the context of Exact Perfect Matching, however, applying the path can change the number of reds abruptly, so we are risking overshoot. The issue will become apparent when you read Chapter 3.

1.3 What Next?

We have been conveying pessimism from the outset to let the reader appreciate the difficulty of Exact Perfect Matching. You might now wonder if it is **NP**-complete after all. But it comes at a surprise that the problem can be solved efficiently by a *randomised* algorithm [20]!

There has been a belief among complexity theorists that every randomised polynomial time algorithm can be efficiently derandomised; see [12] for an evidence. This pushes Exact Perfect Matching to the frontier to test such belief. In particular, can we really remove the randomness in that algorithm?

We do not know. The algorithm, to be detailed in Chapter 5, relates perfect matchings with a symbolic matrix determinant. Randomness plays a key role in probing the determinant, and at present people fail to get rid of it while retaining efficiency. This is consistent with our lack of structural understanding of k-PMs; randomness gets away with it luckily, but conceals the truth on the other hand.

We should mention that the algorithm can be made deterministic when the graph is planar [15] or more generally $K_{3,3}$-free [17]. Chapter 5 contains a simplified analysis, too. Unfortunately, these results do not extend to denser graphs.

1.4 Polytope Descriptions

Classical matching theory can be viewed through the lens of polytopes, too. In this direction, we represent a matching by a 0-1 vector indicating which edges are included. The disjointness condition can be easily enforced by linear inequalities, one for each vertex. This gives rise to an *integer program*. Solving general integer linear programs is an **NP**-complete problem, but when it comes to matchings, the difficulties may be circumvented by a relaxed *linear program*. This means that we do not study the discrete vectors directly but rather their convex hull—thus a polytope. For bipartite graphs, this polytope is especially easy to describe; for general graphs, it has exponentially many facets but they are nicely characterised by Edmonds [4]. Hence, the problem of finding the maximum cardinality matching, and the more general problem of finding a maximum weighted matching, can be solved via the theory of linear programming.

How do these transfer to Exact Perfect Matching? Not much. Even in the bipartite case, and even if we look at $k = 1$, the related polytopes would already be bizarre. Chapter 6 delves in this direction and sheds some light on the polytopes as well as their complexity. It remains a big problem if we can find a description in the same spirit of Edmonds' description of the matching polytope.

1.5 A Step Further

The concluding Chapter 7 initiates the study of a generalised problem called Exact Perfect f-Matching. Here f is a function that specifies an integer "quota" for every vertex. A perfect f-matching picks a multiset of edges so that every vertex v is incident to $f(v)$ of them. When $f \equiv 1$ it degenerates to a perfect matching. The problem asks if the graph admits a perfect f-matching with exactly k reds.

Despite its generality, we will show that the problem reduces to Exact Perfect Matching when f is polynomially bounded. In such case, we present a dynamic program that solves the problem for graphs with small tree-width. This is rather standard—perhaps too boring to be documented by experts—but we did the labour once and for all.

The intriguing case, though, is when we allow exponential values. First, the reduction does not work any more. Second, the determinant-based approach does not seem to adapt, so we are not aware of any efficient randomised algorithm. And third, we fail to prove its hardness. What is the complexity of the problem then? I find it an ideal place to stop and allow reflection.

The goal of this thesis is to organise things in one place, to simplify proofs and establish new connections or insights, and ultimately to serve the reader as an in-depth reference in the field.

2 Notations and Reformulations

This chapter sets up the notations to be used throughout the thesis. We then present several reformulations of Exact Perfect Matching which suggest the essence of the problem from multiple perspectives.

2.1 Notations

Unless otherwise stated, we study a graph $G=(V,E)$ on $|V|=2n$ vertices whose edges are coloured either red or blue: $E=R\uplus B$. When the graph is bipartite, we assume bipartition $V=X\uplus Y$. We usually abbreviate an edge $\{u,v\}$ as uv. The neighbourhood of vertex $v\in V$ is denoted $N(v)$. For a subset of vertices $U\subseteq V$, we write $E[U]$ for the set of edges with both ends in U, and $G[U]:=(U,E[U])$ for the subgraph induced by U.

A *matching* $M\subseteq E$ is a set of edges that do not share vertices. If $|M|=n$ then it is a *perfect matching*; if in addition $|M\cap R|=k$ then we call it a *k-PM*. Vertices incident to a matching edge are said to be *matched* or *covered*.

We assume that G has at least one perfect matching. Denote $k_{\min}$ and $k_{\max}$, respectively, to be the minimum and maximum k for which k-PM exists in G. These numbers can be computed in polynomial time: Place unit weights on red edges and nil weights on blue edges, then invoke Edmonds' primal-dual algorithm to obtain a minimum/maximum weight perfect matching in G.

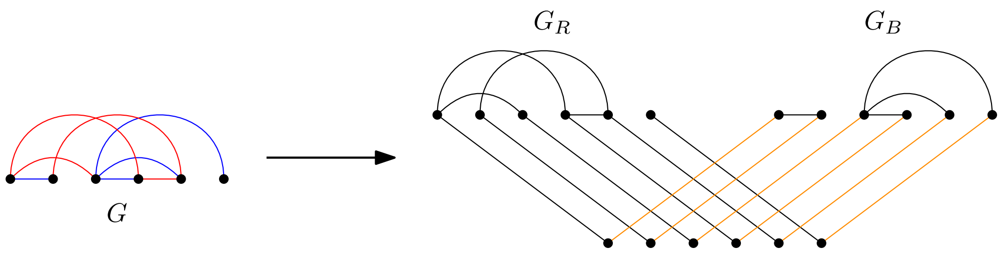
Clearly if G admits a k-PM then $k_{\min}\leqslant k\leqslant k_{\max}$. However, this condition is far from sufficient by Theorem 1.1.

2.2 Making the Reds Disjoint

In general the graph may contain $\Theta(n^2)$ red edges, but it turns out that we can sparsify them without loss of generality.

Lemma 2.1. For every graph G on $2n$ vertices, there is a graph G' on $6n$ vertices with exactly $2n$ disjoint red edges, such that G admits a k-PM iff G' admits a $2k$-PM.

Proof. Assume $G=(V,R\uplus B)$. We separate two graphs $G_R:=(V,R)$ and $G_B:=(V,B)$ that encode the red and blue parts in G, respectively. All edges are now coloured black. On top of these, we make a third copy of V and connect each new vertex to its counterparts in G_R (using a black edge) and G_B (using an orange edge). Call the resulting graph G'. The figure below illustrates the reduction.



For every perfect matching in G', the third copy of vertex v has to make a binary choice: either go for the black, which covers the $v \in G_R$ but leaves open the $v \in G_B$; or go for the orange, which covers the $v \in G_B$ but leaves open the $v \in G_R$. After all decisions are made, the remaining open vertices need to be matched within G_B or G_R. These correspond to vertex v being covered by a blue or a red matching edge in G, respectively.

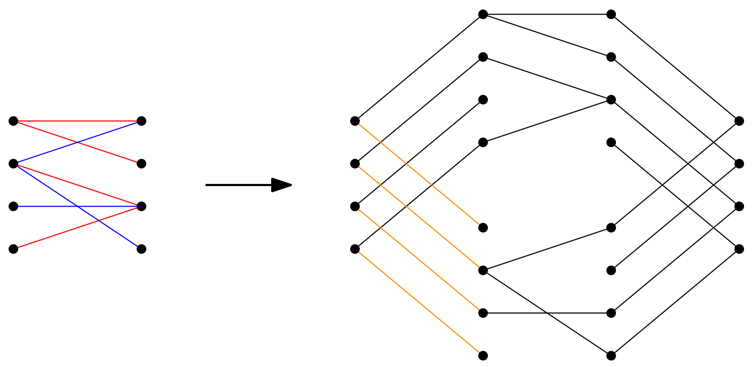
Conversely, every perfect matching in G can be translated to a perfect matching in G', under the same interpretation.

Note that every red matching edge uv in G correspond to two orange matching edges in G': one for u and one for v. Therefore, k-PM in G corresponds to $2k$-PM in G', and the proof is complete. □

For bipartite graphs, we can additionally save half of the orange edges and preserve the parameter k in the transformation.

Lemma 2.2. For any bipartite graph G on $2n$ vertices, there is a bipartite graph G' on $6n$ vertices with exactly n disjoint red edges, such that G admits a k-PM iff G' admits a k-PM.

Proof. Assume $G = (X \uplus Y, R \uplus B)$, and split two graphs $G_R := (X \uplus Y, R)$ and $G_B := (X \uplus Y, B)$. All edges are coloured black. Then we make a copy of X and connect each new vertex to its counterparts in G_R (using a black edge) and G_B (using an orange edge). Finally we make a copy of Y and connect each new vertex to its counterparts in G_R and G_B in black. Clearly the result G' is bipartite. See the figure below for an example.



The correspondence between perfect matchings in G and G' is as before. But this time, every red matching edge $xy \in X \times Y$ in G corresponds to only one orange matching edge in G': the one incident to vertex x. Hence k-PM in G corresponds to k-PM in G', as we claimed. □

In some situations one might want to reduce the instance so that the red edges in the graph form a perfect matching; that is, every vertex is incident to exactly one red edge. This is indeed possible:

Lemma 2.3. For any (bipartite) graph G on $2n$ vertices, there is a (bipartite) graph G' on $6n+4n^2$ vertices whose red edges form a perfect matching, such that G admits a k-PM iff G' admits a $2k$-PM for $k \leqslant n$.

Proof. First we apply the reduction in Lemma 2.1 and obtain a graph H on $6n$ vertices with disjoint orange edges. Exactly $2n$ vertices are not yet incident to orange. Pick any two of them and add a path of length $4n+1$ in between. Colour the path by orange and black in alternation, starting and ending with orange. We repeat the procedure until every vertex gets (exactly) one incident orange edge. This is our graph G'.

Every k-PM in G corresponds to a $2k$-PM in H. The latter naturally maps to a $2k$-PM in G' by "ignoring" the additional paths; that is, using only the black edges along those paths.

Conversely, any $2k$-PM in G' must ignore all additional paths; otherwise it has to use the $2n+1$ oranges along that path, which already exceeds $2k$. □

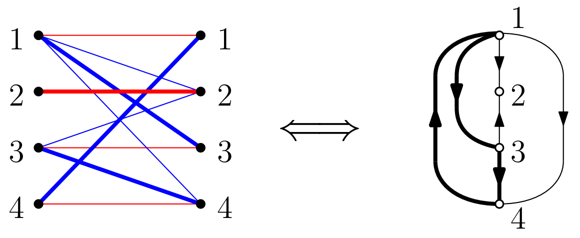
2.3 Exact Cycle Cover

With the last reduction, it is now easy to show that Exact Perfect Matching in bipartite graphs is polynomially equivalent to Exact Cycle Cover in directed graphs. The latter problem asks if you can cover exactly k vertices by disjoint cycles in a directed graph. The relation between the two was already pointed out by Papadimitriou and Yannakakis in their original paper [21].

Theorem 2.4. Exact Perfect Matching in bipartite graphs can be reduced to Exact Cycle Cover in directed graphs within polynomial time, and vice versa.

Proof. Given a bipartite graph, we apply the reduction in Lemma 2.3 to get a bipartite graph $G = (X \uplus Y, R \uplus B)$ where the red edges form a perfect matching. We label the vertices in parts X and Y both by integers $\{1, \ldots, n\}$, so that each red edge goes between "parallel" vertices $i \in X$ and $i \in Y$. Next, orient the blue edges from X to Y, and then contract the red edges. This leads to a directed graph G' on $\{1, \ldots, n\}$.

Note that every k-PM in G picks k red (parallel) edges, and $n-k$ blue (non-parallel) edges. These blues form a perfect matching between $I \cap X$ and $I \cap Y$ for some $I \subseteq \{1, \ldots, n\}$ of size $n-k$. Every blue edge ij corresponds to a directed edge $i \to j$ in G', so the latter together form a cycle cover in $G'[I]$. Conversely, cycles in G' that cover $n-k$ vertices can be interpreted as a k-PM in G. Therefore, Exact Perfect Matching reduces to Exact Cycle Cover.



The reduction also works in reversal. Namely, given a directed graph, we can construct a bipartite graph by splitting each vertex into two copies joined by a parallel red edge, and replacing each directed edge $i \to j$ with an undirected blue edge from the left copy of i to the right copy of j. We omit the rest of the argument as it is similar to the other direction. □

2.4 Max Weight Extendable k-Set

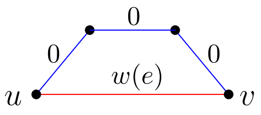
Next we introduce a less obvious reformulation, first studied by El Maalouly [5]. An edge subset $F \subseteq E$ is *extendable* if it is contained in some perfect matching. Consider a weight function $w: E \to \mathbb{N}_0$ and write $w(F) := \sum_{e \in F} w(e)$ for $F \subseteq E$. The Max Weight Extendable k-Set problem is stated as (1) below. For convenience, we also list the Exact Perfect Matching problem as (3) and an intermediate problem as (2).

(1) Given $G = (V, E, w)$ and $k \in \mathbb{N}$, maximise $w(F)$ over all extendable sets F of size k.
(2) Given $G = (V, R \uplus B, w)$ and $k \in \mathbb{N}$, maximise $w(M)$ over all k-PMs M.
(3) Given $G = (V, R \uplus B)$ and $k \in \mathbb{N}$, find a k-PM.

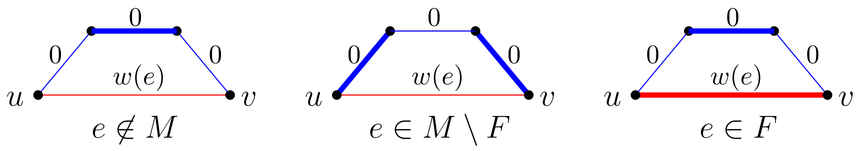
That the three problems are equivalent was shown in [5] and [8]. Here we give a different and simpler proof.

Reducing (1) to (2)

Note that (1) is the same as optimising $w(F)$ over all pairs (F, M) where $F \in \binom{M}{k}$ is contained in perfect matching M. Given an instance of (1), we colour every edge $e = uv$ red and attach a parallel blue path of length three and zero weight:



This gadget can model the three possible states of the edge:



So a valid pair (F, M) in the old graph one-one corresponds to a k-PM M' in the new graph, and $w(F)$ equals the weight of M'. Hence the problem reduces to (2).

Reducing (2) with polynomial weight to (3)

Given an instance of (2), we fix a constant $L > w(E)$ and replace every red edge e by a red-blue alternating path P_e of length $2\,(L + w(e)) - 1$ that starts and ends with reds. Note that such construction is efficient only if the weights are polynomially bounded.

Every perfect matching M in the old graph corresponds to a perfect matching M' in the new graph, and vice versa: M picks a red edge e iff M' picks all the $L + w(e)$ reds in P_e. Moreover $|M' \cap R| = |M \cap R| \cdot L + w(M)$. Therefore, a k-PM in the old graph corresponds to some k'-PM in the new graph for $k\,L \leqslant k' = k\,L + w(M) < (k+1)\,L$. The large cushion L prevented the mixing of different number of reds.

To solve (2), we invoke (3) multiple times to decide whether there is a k'-PM in the new graph, for $k' = (k+1)\,L - 1$ down to $k\,L$. If at some point we detect such a perfect matching then we stop. It maps to k-PM of maximum weight $k' - k\,L$ in the old graph.

Reducing (3) to (1)

Given an instance of (3), we replace

- each red edge by a path of length three with weights $2, 3, 2$; and

- each blue edge by an edge of weight 0.

Every perfect matching M in the old graph G naturally corresponds to a perfect matching M' in the new graph G', and vice versa. For each $e \in E(G)$,

- if $e \in R \cap M$ then it contributes to M' two edges of weight 2;
- if $e \in R \setminus M$ then it contributes to M' an edge of weight 3;
- if $e \in B \cap M$ then it contributes to M' an edge of weight 0;
- if $e \in B \setminus M$ then it does not contribute to M'.

Denote $r := |R|$ and $i := |M \cap R|$. Then M' has $2i$ edges of weight 2 and $r - i$ edges of weight 3; other edges have weight 0. Note that only $r + i$ edges have non-zero weights. Consider the heaviest $k' := r + k$ edges $F' \subseteq M'$.

- If $i \leqslant k$ then all non-zero edges are included in F', so $w(F') = 2 \cdot 2i + 3\,(r - i) = 3\,r + i$.
- Otherwise $i > k$, thus $r - i < k' < r + i$. So all the 3's are included but some 2's are missed by F'. It follows that $w(F') = 2\,(k' - (r - i)) + 3\,(r - i) = 3\,r - i + 2\,k$.

Apparently $w(F') \leqslant 3\,r + k$. It attains the maximum possible value iff $i = k$, that is, when M has k reds.

Remark. For problem (3) one can assume $r = \Theta(n)$ via Lemma 2.1. Then reduction (3)$\Rightarrow$(1) constructs a graph G' on $n + 2\,r = \Theta(n)$ vertices. Suppose we are able to ε-approximate (1), i.e. report a pair (F, M) such that $w(F)$ is within $1 \pm \varepsilon$ times the optimum $3\,r + k = \Theta(n)$. It corresponds to a matching M with $i = k \pm \Theta(n)$ reds. In the most interesting (and trickiest) scenario $k = \Theta(n)$, this gives a constant-factor approximation for (3). Similarly, if we are able to additively approximate (1) then we can also do so for (3).

3

Combinatorial Structures

Starting from this chapter, we investigate the combinatorial structure of (coloured) perfect matchings. We will borrow classical tools from matching theory and incorporate edge colours. Then we can present a proof of the Continuity Theorem, as well as a characterisation for complete graphs due to Karzanov [14]. Despite some success, we are still in short of a general characterisation like the theorems of Kőnig's, Hall's and Tutte's.

3.1 Symmetric Difference of Perfect Matchings

If there is one central notion in the matching theory, then it is the notion of symmetric difference. The *symmetric difference* of two sets S, S' is defined as

$$S \oplus S' := (S \setminus S') \cup (S' \setminus S).$$

In other words, it is the "exlusive or" that captures where S and S' differ. Clearly $\oplus$ is commutative and associative, and $S \oplus S = \emptyset$.

Applied to matching theory, we usually have a preliminary matching M at hand and would like to "improve" it to a matching M' with certain property. Conceptually $M \oplus M'$ tells us how to bridge the gap in one shot. What makes the notion immensely useful for us is that $M \oplus M'$ takes rather regular form when $M, M' \subseteq E$ are perfect matchings.

Definition 3.1. Let $F, F' \subseteq E$. A path/cycle is $\{F, F'\}$*-alternating* if its edges alternate between F and F'. For convenience, $\{F, E \setminus F\}$-alternating is abbreviated as F*-alternating*.

Lemma 3.2. Let M be a perfect matching.

(i) If M' is a perfect matching, then $\mathcal{C} := M \oplus M'$ is a collection of disjoint $\{M, M'\}$-alternating cycles.

(ii) If $\mathcal{C}$ is a collection of disjoint M-alternating cycles, then $M' := M \oplus \mathcal{C}$ is a perfect matching.

Proof. To see (i), consider any vertex v. It is incident to exactly one edge $e \in M$ and one edge $e' \in M'$ because M, M' are perfect. If $e = e'$ then they cancel out in $M \oplus M'$; otherwise both remain. So v is incident to either zero or two edges in $M \oplus M'$, implying that $M \oplus M'$ forms a collection of disjoint cycles. Clearly each cycle must be $\{M, M'\}$-alternating.

To see (ii), first note that $M \oplus M' = \mathcal{C}$, so M and M' differ on $\mathcal{C}$ only. Hence, to show that M' is a perfect matching, it suffices to argue that each vertex v lying on $\mathcal{C}$ is incident to exactly one edge from M'.

To this end, v is incident to exactly two edges $e, e' \in \mathcal{C}$ due to disjointness of the cycles. Recall that $\mathcal{C}$ is M-alternating, thus exactly one edge is M, say $e \in M$ and $e' \notin M$. So $e \notin M'$ and $e' \in M'$, and v is indeed incident to exactly one edge from M'. $\square$

The lemma basically says "to obtain another perfect matching is equivalent to finding alternating cycles". It opens up the possibility to move from M to a target M' incrementally. Since the difference $M \oplus M'$ consists of one or more disjoint alternating cycles, we may search for and apply the cycles in discrete steps, patching one cycle on the current perfect matching at a time. We will illustrate the idea when we prove Theorem 1.2 in the next section.

3.2 Accounting the Change of Reds

So far we have only accounted the graphical structure by the term "alternating cycle". But we also care about the edge colours, in particular how the number of reds changes after a modification. This motivates us to define a weight function $E \to \{0, -1, +1\}$ with respect to a matching M:

$$\delta(e|M) := \begin{cases} 0 & e \in B, \\ -1 & e \in R \cap M, \\ +1 & e \in R \setminus M. \end{cases}$$

We abuse the notation and write $\delta(F|M) := \sum_{e \in F} \delta(e|M)$ for any subset $F \subseteq E$. Then $\delta(F|M)$ captures the net growth of reds when we move from M to $M \oplus F$. In other words, $|R \cap (M \oplus F)| = |R \cap M| + \delta(F|M)$.

With these tools, we are ready to prove the Continuity Theorem 1.2 via an interpolation argument.

Proof of Theorem 1.2. Assume $k_1 < k < k_2$; for otherwise the theorem is trivial. Let M_1 and M_2 be arbitrary k_1- and k_2-PM, respectively. Consider the symmetric difference $\mathcal{C} := M_1 \oplus M_2$, which consists of alternating cycles $C_1, \ldots, C_t$. Then

$$k_2 = k_1 + \delta(\mathcal{C}|M_1) = k_1 + \sum_{i=1}^{t} \delta(C_i|M_1).$$

Hence there exists a minimal i_0 such that

$$k_1 + \sum_{i=1}^{i_0} \delta(C_i|M_1) \;<\; k \;\leqslant\; k_1 + \sum_{i=1}^{i_0+1} \delta(C_i|M_1);$$

in particular $\delta(C_{i_0}|M_1) > 0$. We apply $C_1 \cup \cdots \cup C_{i_0}$ on matching M_1, boosting its number of reds near but below k. If we in addition apply C_{i_0+1}, either we get to exactly k and finish, or there is an overshoot.

To resolve such overshoot, we shall not apply C_{i_0+1} in its entirety. Instead, we extract an alternating subpath from C_{i_0+1} that exactly fills the gap

$$\Delta := k - \left(k_1 + \sum_{i=0}^{i_0} \delta(C_i|M_1) \right) > 0.$$

To this end, observe that C_{i_0+1} must contain a red M_2 edge followed by a blue M_1 edge; otherwise $\delta(C_{i_0+1}|M_1) \leqslant 0$, a contradiction. Hence we cut C_{i_0+1} at this blue M_1 edge and linearize it to an alternating path $P = v_1 \ldots v_{2\ell}$, where $v_1 v_2 \in M_2$ is red and $v_{2\ell-1} v_{2\ell} \in M_2$. Note also that $\delta(P) = \delta(C_{i_0+1}) > \Delta$.

We gradually grow a subpath $Q := v_1 \dots v_{2j+1}$ for $j = 1, \dots, \ell - 1$. This is an alternating path that starts with M_2 and ends with M_1. Initially $\delta(Q) \leqslant 1$. Increasing j by one shall change $\delta(Q)$ by at most one. Eventually $\delta(Q) \geqslant \delta(P) - 1 \geqslant \Delta$. Hence there exists an intermediate j with $\delta(Q) = \Delta$. Applying this alternating path Q (in addition to $C_1 \cup \dots \cup C_{i_0}$) on M_1 will boost the number of reds to exactly k. There is one final caveat: before applying Q, we must drop the blue edge $v_1 v_{2\ell}$ (the "cut") from M_1 to ensure that v_1 is unmatched; otherwise applying Q would not produce a valid matching. This is the only loss in our argument, and the resulting matching is almost perfect. □

3.3 Characterisation for Complete Graphs

Conceivably, the discontinuous feasibility of k-PMs arises from sparsity of G. If the graph is dense enough, we should be able to move from one perfect matching to another smoothly. This direction was first pursued in the early work of Karzanov [14], where he characterised the feasibility regimes for complete and complete bipartite graphs. Here we focus on $G = K_{n,n}$ and present a streamlined proof suggested by [9, 10]; the case $G = K_{2n}$ is quite similar. Throughout we assume that G is not monochromatic—otherwise the problem is trivial.

Let us call a graph *stiff* if all its connected components are complete. We branch four types on the structures of $G_R := (V, R)$ and $G_B := (V, B)$:

(1) G_R and G_B are both stiff.

(2) G_R or G_B is not stiff.

  (2.1) G_R is not stiff, whereas G_B is stiff and admits a perfect matching;

  (2.2) G_B is not stiff, whereas G_R is stiff and admits a perfect matching;

  (2.3) otherwise.

For the main plot we only care about differentiating types (1) and (2). In the following, an (r, b)-*cycle* refers to a cycle with r reds and b blues.

Lemma 3.3. Type (1) is characterised by any of the two conditions:

(i) Both G_R and G_B are disjoint unions of two complete bipartite graphs. That is, $G_R = K_{X_1,Y_1} \cup K_{X_2,Y_2}$ and $G_B = K_{X_1,Y_2} \cup K_{X_2,Y_1}$.

(ii) Every cycle in G has an even number of reds and an even number of blues.

Proof. First let us assume (1). Then neither G_R nor G_B is connected; otherwise G would be monochromatic. On the other hand, G_R can have at most two components. To see this, suppose it had at least three components K_{X_i,Y_i} $(i = 1, 2, 3, \dots, t)$, then the edges bridging X_i, Y_j are blue for all $i \neq j$, which implies that G_B is connected, a contradiction. Hence G_R has exactly two components. By symmetry G_B has exactly two components, too. This establishes (i).

Next we assume (i). Traverse any cycle in G. Every time we jump from one component of G_B to the other (necessarily along a red edge), we have to return along another red edge later. During this excursion we encounter no other reds, thus the total number of traversed reds is even. So is the number of traversed blues. This establishes (ii).

Finally we assume (ii). Suppose some component of G_R is not complete; denote $X_0 \subseteq X$ and $Y_0 \subseteq Y$ its two parts in the bipartition. Then there exists a blue edge $xy \in X_0 \times Y_0$. Since x and y lie in the same red component, they are connected via some red path. It together with xy form a cycle containing exactly one blue, which contradicts (ii). Therefore G_R is stiff; analogously G_B is stiff as well. This establishes (1). □

Lemma 3.4. Suppose G has type (1) and $k_{\min} \leqslant k \leqslant k_{\max}$. Then k-PM exists iff $k_{\min} \equiv k \equiv k_{\max} \pmod 2$.

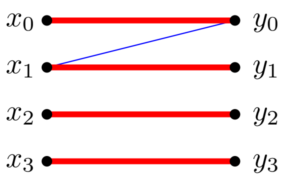
Proof. Here is an important observation: The value $\delta(C|M)$ is even for every matching M and M-alternating cycle C. Indeed, from Lemma 3.3 we know that C contains an even number of reds. Each such red contributes either $+1$ or -1 to $\delta(C|M)$, whereas other edges contribute nothing. So $\delta(C|M)$ has to be even.

Let M_1, M_2 be perfect matchings with $k_{\min}$ and $k_{\max}$ reds, respectively. Suppose there is a k-PM M. Every cycle C in $\mathcal{C} := M_1 \oplus M$ has an even $\delta(C|M_1)$, which implies $k = k_{\min} + \sum_{C \in \mathcal{C}} \delta(C|M_1) \equiv k_{\min} \pmod 2$. Similarly $k \equiv k_{\max} \pmod 2$.

For the reverse direction, if $k_{\min} = k_{\max}$ then there is nothing to prove. Hence assume $k_{\min} < k_{\max}$. We employ a very similar argument to that of Theorem 1.2. The symmetric difference $M_1 \oplus M_2$ consists of alternating cycles. We iteratively apply the cycles on M_1, until the next cycle $C =: (x_1, y_1, \ldots, x_\ell, y_\ell)$ overshoots the target k. Without loss of generality we assume $x_1 y_1 \in M_1$. Denote by $0 < \Delta < \delta(C|M_1)$ the remaining gap to k, which must be even by our observation. We shortcut the cycle C into an M_1-alternating $C' := (x_1, y_1, \ldots, x_i, y_i)$ for $i = 2, \ldots, \ell$. Initially $\delta(C'|M_1) \in \{0, 2\}$ and eventually $\delta(C'|M_1) = \delta(C|M_1)$. Increasing i by one changes δ by either zero or two (again by our observation). So there exists an intermediate i for which $\delta(C'|M_1) = \Delta$. Applying this cycle C' on the current M_1 shall close up the gap exactly. $\square$

Lemma 3.5. Suppose G has type (2), $k_{\min} \leqslant k \leqslant k_{\max}$ and $2 \leqslant k \leqslant n-2$. Then k-PM always exists.

Proof. We prove by induction on n. The base cases $n \leqslant 4$ can be argued by enumeration. Now assume $n \geqslant 5$. By symmetry of the two colours, we may also assume $k \geqslant \lceil n/2 \rceil \geqslant 3$. We apply Theorem 1.2 to find an almost perfect matching M with exactly k reds. If it turns out to be perfect then we are done. Otherwise, let x_0 and y_0 be the two unmatched vertices, where y_0 is incident to the dropped blue edge $x_1 y_0$ (the $v_1 v_{2\ell}$ in the proof of Theorem 1.2). The vertex x_1 is matched to some red edge $x_1 y_1$ (the $v_1 v_2$ in previous proof). Take two other red matching edges $x_2 y_2$ and $x_3 y_3$, which exist since $k \geqslant 3$. Apparently, if $x_0 y_0$ is blue then $M^+ := M + x_0 y_0$ is a k-PM. So below we assume that $x_0 y_0$ is red, thus M^+ is a $(k+1)$-PM. Here is an illustration of the setup, where bold edges represent M^+.



Consider the graph $G' := G - \{x_2, y_2\}$ with $n' = n-1$ and $k' := k-1$. Note that $2 \leqslant k' \leqslant n'-2$. If G' admits a k'-PM then we happily insert the edge $x_2 y_2$ and obtain a k-PM in G. If not, then G' has type (1) by induction. Via an analogous reasoning, we may assume that $G'' := G - \{x_3, y_3\}$ has type (1), too.

What do they tell us about the graph G? Applying Lemma 3.3 on G'', we deduce that

- $x_0 y_1$ needs to be blue;
- $x_1 y_2$ and $x_2 y_1$ have the same colour; and
- $x_0 y_2$ and $x_2 y_0$ have the same colour different from the previous pair.

Anyway (x_i, y_2, x_2, y_i) is a $(4,0)$-cycle for some $i \in \{0, 1\}$. We apply Lemma 3.3 on G'' again to conclude two properties:

- $x_i y$ and $x_2 y$ have the same colour, for all $y \in Y \setminus \{y_3\}$; and

- $x\,y_i$ and $x\,y_2$ have the same colour, for all $x \in X \setminus \{x_3\}$.

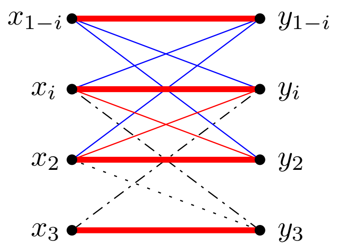
The properties might or might not hold for y_3 and x_3. (The argument certainly does not because the concerned edges appear in neither G' nor G''.) So let us make two cases:

(i) Suppose the two properties extend to y_3 and x_3. Decompose $G'_R =: K_{X'_1,Y'_1} \cup K_{X'_2,Y'_2}$ and $G'_B =: K_{X'_1,Y'_2} \cup K_{X'_2,Y'_1}$ where $X'_1 \cup X'_2 = X \setminus \{x_2\}$ and $Y'_1 \cup Y'_2 = Y \setminus \{y_2\}$. Without loss of generality $x_i \in X'_1$ and thus $y_i \in Y'_1$. Applying the properties, we have

$$\begin{array}{lcl} x_2\,y \in R & \iff & x_i\,y \in R \iff y \in Y'_1 \\ x\,y_2 \in R & \iff & x\,y_i \in R \iff x \in X'_1 \end{array}$$

In other words, all edges between x_2, Y'_1 (resp. X'_1, y_2) are red, and all edges between x_2, Y'_2 (resp. X'_2, y_2) are blue. So with $X_1 := X'_1 \cup \{x_2\}$ and $Y_1 := Y'_1 \cup \{y_2\}$ we have K_{X_1,Y_1} and $K_{X'_2,Y'_2}$ being red, and K_{X_1,Y'_2} and $K_{X'_2,Y_1}$ being blue. Therefore G has type (1) by Lemma 3.3, a contradiction.

(ii) So the properties do not always hold. With symmetry we may assume that y_3 is the violator, so $x_i\,y_3$ has colour c and $x_2\,y_3$ has the opposite colour $\bar{c}$. Hence $x_3\,y_i$ has colour c by Lemma 3.3 on G'. Now observe that $C := (x_i, y_i, x_3, y_3, x_2, y_2)$ is an M^+-alternating $(5,1)$-cycle with $\delta(C|M^+) = -1$. Applying it on M^+ would produce a k-PM as desired. See the figure below for a visualisation.

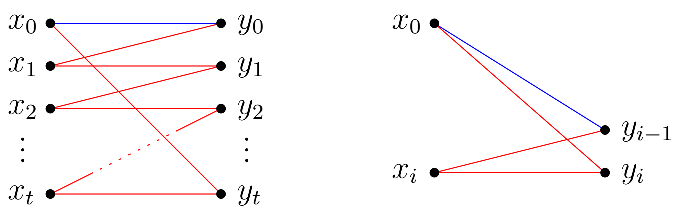


□

What remains are the corner cases $k \in \{0, 1, n-1, n\}$. The preparatory lemma below allows us to shorten a cycle that contains one blue.

Lemma 3.6. If G contains a $(2t+1, 1)$-cycle for some $t \geqslant 0$, then it contains a $(3,1)$-cycle.

Proof. Let $(x_0, y_0, \ldots, x_t, y_t)$ be a $(2t+1, 1)$-cycle where $x_0\,y_0$ is the only blue edge. Take the minimal i such that $x_0\,y_i \in R$. It exists because $x_0\,y_t \in R$. By minimality we know $x_0\,y_{i-1} \in B$, thus (x_0, y_{i-1}, x_i, y_i) is a $(3,1)$-cycle.



□

Lemma 3.7.

G_R is not stiff $\iff$ G contains $(3,1)$-cycle.
G_B is not stiff $\iff$ G contains $(1,3)$-cycle.

Proof. The two statements are symmetric, so we prove the first only. Fix a non-complete component of G_R and let $X_0 \subseteq X$ and $Y_0 \subseteq Y$ be its two parts. So there exists a blue edge $x_0\,y_0 \in X_0 \times Y_0$. As x_0 and y_0 lie in the same red component, they are connected via some red path $y_0\,x_1\,y_1 \ldots x_t\,y_t\,x_0$. It together with $x_0\,y_0$ gives a $(2t+1,1)$-cycle. The claim then follows from Lemma 3.6.

Conversely, if G contains a $(3,1)$-cycle, then the four vertices on the cycle are in the same red component H. This H is not complete as witnessed by the blue edge. □

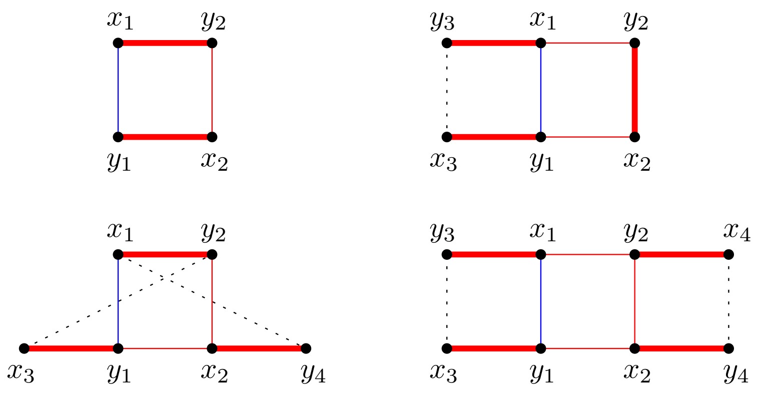
The following lemma is our final missing piece.

Lemma 3.8. Suppose G has type (2).

- If $k_{\max} = n$, then $(n-1)$-PM exists iff G contains $(3,1)$-cycle.
- If $k_{\min} = 0$, then 1-PM exists iff G contains $(1,3)$-cycle.

Proof. The statements are symmetric and we prove the first only. Let M be an n-PM, that is a perfect matching in G_R. To show the forward implication, let M' be an $(n-1)$-PM and consider the symmetric difference $M \oplus M'$. Only one alternating cycle contains blue, and it must be a $(2t+1,1)$-cycle for some $t \geqslant 0$. Hence there is a $(3,1)$-cycle by Lemma 3.6.

For the reverse implication, let (x_1, y_1, x_2, y_2) be a $(3,1)$-cycle where $x_1\,y_1$ is blue. Consider the vertices matched to these four in M. We argue that there is an M-alternating cycle C on the at most eight vertices in consideration, with $\delta(C|M) = -1$. Let us enumerate all cases (modulo symmetry):



In the top-left case, we simply take $C = (x_1, y_1, x_2, y_2)$. In the bottom-left case, if $x_3\,y_2$ is red then we take $C = (x_1, y_1, x_3, y_2)$; similarly if $x_1\,y_4$ is red then we take $C = (x_2, y_2, x_1, y_4)$; otherwise we take $C = (x_1, y_2, x_3, y_1, x_2, y_4)$. The remaining two cases are similar. □

Theorem 3.9. (Karzanov) In the respective four types, G admits a k-PM for $k_{\min} \leqslant k \leqslant k_{\max}$ iff

(1) $k_{\min} \equiv k \equiv k_{\max} \pmod 2$.
(2.1) $k \neq 1$.
(2.2) $k \neq n-1$.
(2.3) always.

Proof. If G has type (1) then the feasibility is captured by Lemma 3.4.

If G has type (2.1) then $k_{\min} = 0$. When $2 \leqslant k \leqslant n-2$ the feasibility is captured by Lemma 3.5. When $k \in \{0, k_{\max}\}$ the feasibility is trivial. So the only uncovered cases are (i) $k = 1$ and (ii) $k = n-1 < k_{\max}$. Note that G contains $(3,1)$-cycle but no $(1,3)$-cycle by Lemma 3.7. Hence (i) is infeasible and (ii) is feasible by Lemma 3.8.

Types (2.2) and (2.3) are handled analogously. □

4

The Sandwich Paradigm

As we saw in Karzanov's theorem, complete (bipartite) graphs enjoy continuity: they admit k-PM for basically every k, except in the stiff case where the feasibility of k jumps by two. A key insight by Karzanov is that we can shorten long cycles in a symmetric difference down to 4-cycles; applying the latter on a perfect matching can change the number of reds by at most two.

Of course, similar properties extend to "sufficiently structured" graphs. This line of research was initiated by El Maalouly et al. [6, 7] where the authors studied graphs of bounded (bipartite) independence number. In another extension, Haslebacher [11] defined the doubly chordal property to frame some other graphs, such as unit interval graphs and chain graphs. Both work follow a recipe that we summarise as the *sandwich paradigm*.

We will start with its basic version which can return us a $(k \pm \varepsilon)$-PM, where $\varepsilon \in \mathbb{N}$ is an error term depending on G. Along the way we will analyse the error for a wealth of graph classes, some of which not yet considered in the literature. Then we will refine the paradigm and kill the error for some graphs. As an interlude we supply three "lifting theorems" that generate even more graphs on which the sandwich paradigm is applicable.

4.1 The Basic Version

We maintain two perfect matchings M_1 and M_2 such that $|M_1 \cap R| \leqslant k \leqslant |M_2 \cap R|$. In each iteration we try to close their "gap" in certain sense. Eventually the process terminates and we catch our target in the middle.

There are many ways to measure the gap. The most obvious one is $|M_2 \cap R| - |M_1 \cap R|$; another candidate is the cardinality $|M_1 \oplus M_2|$. But it need not be numeric. In general we may use any function $d(M_1, M_2)$ whose codomain is a totally ordered set of polynomial size. The basic sandwich paradigm then goes as follows:

Algorithm Sandwich

```
assume $k_{\min} \leqslant k \leqslant k_{\max}$
let $M_1, M_2$ be perfect matchings with $k_{\min}$ and $k_{\max}$ reds, respectively
**while** $|M_2 \cap R| - |M_1 \cap R| > 2\varepsilon$ **do**
    compute a perfect matching $M$ such that $d(M, M_2), d(M_1, M) < d(M_1, M_2)$
    **if** $|M \cap R| \leqslant k$ **then** $M_1 := M$, **else** $M_2 := M$
**return** $M_1$ or $M_2$
```

To implement the paradigm, one must specify the function d, the tolerable error $\varepsilon \in \mathbb{N}$, and an efficient procedure that computes the M under the loop condition. Given all these, the paradigm shall terminate in polynomial time since each iteration strictly decreases the gap $d(M_1, M_2)$. Note that the sandwich $|M_1 \cap R| \leqslant k \leqslant |M_2 \cap R|$ is maintained by the replacement rule, so upon termination we obtain a perfect matching with $k \pm \varepsilon$ reds. We stress that the number of reds does *not* necessarily move towards k in every single iteration.

The output guarantee is of course weaker than our ultimate goal. Nevertheless, a $(k \pm \varepsilon)$-PM is fairly good approximation, where the error ε quantifies continuity of the feasibility regime of G. For some graphs, we may actually use the approximate solution as a stepping stone to jump to a k-PM. We will come back to this point later.

In the following we showcase the paradigm on a handful of input graphs. Here is a brief summary:

Input graph	**Gap function d**	**Error ε**
independence number α	$\lvert M_1 \oplus M_2\rvert$	$4^{\alpha+1}$
bipartite independence number β	$\lvert M_1 \oplus M_2\rvert$	$\beta + 1$
modular width t	$\lvert M_1 \oplus M_2\rvert$	$t/2$
1-extendable bipartite, face complexity t	$\sum_{C \subseteq M_1 \oplus M_2} \text{area}(C)$	$t/4$

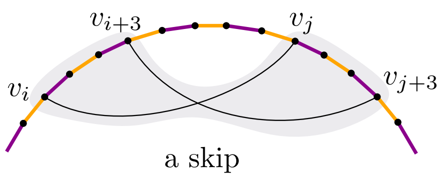
4.1.1 Graphs of small independence number

The *independence number* α of a graph G is the maximum cardinality of an independent set in G. Note that $\alpha = 1$ iff G is complete. Generally, if α is small then G has dense local structure:

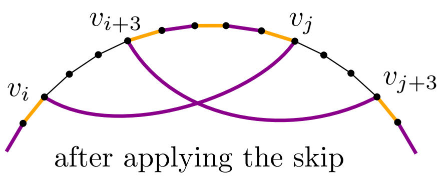
Lemma 4.1. Let $G = (V, E)$ be a graph of independence number α. Then for every set $S \subseteq V$ of $4^{\alpha+1}$ vertices, $G[S]$ contains a clique of size $\alpha + 1$.

Proof. Denote $H := G[S]$. From basic Ramsey theory we know that either H or $\overline{H}$ contains a clique of size $\alpha + 1$. But the latter case is impossible because such a clique corresponds to an independent set of size $\alpha + 1$ in H. □

Definition 4.2. An $M_1 \to M_2$ *skip* is an M_1-alternating cycle $v_i \rightsquigarrow v_{i+3} \to v_{j+3} \rightsquigarrow v_j \to v_i$ where $v_i \dots v_{j+3}$ is a subpath in $M_1 \oplus M_2$, and $v_i v_j$, $v_{i+3} v_{j+3}$ are two chords.



a skip



after applying the skip

An $M_1 \to M_2$ skip D has length 8, thus $-4 \leqslant \delta(D|M_1) \leqslant 4$. It contains four edges from M_1, two edges from M_2, and two chords from neither. Applying it on M_1 would reduce the cardinality of $M_1 \oplus M_2$ by exactly $4 + 2 - 2 = 4$.

Given $M_1 \oplus M_2$, we can list all skips in $O(n^2)$ time because fixing one chord that appears in the skip would determine the other.

Lemma 4.3. Every $\{M_1, M_2\}$-alternating path P of length $|P| \geqslant 4^{\alpha+2}$ gives rise to an $M_1 \to M_2$ skip.

Proof. Orient P along one direction and cut it into $|P|/4 \geqslant 4^{\alpha+1}$ disjoint subpaths, each having length three and pattern "$M_1 M_2 M_1$". Collect the heads of these paths into a set S. By Lemma 4.1 there is a subset $S_0 \subseteq S$ of size $\alpha+1$ such that $G[S_0]$ is a clique. Let T_0 be the tails corresponding to S_0. As $|T_0| = \alpha + 1$ is larger than the independence number, there exist adjacent vertices $t_1, t_2 \in T_0$. Let $s_1, s_2 \in S_0$ be their corresponding heads, respectively. Then the cycle $s_1 \rightsquigarrow t_1 \to t_2 \rightsquigarrow s_2 \to s_1$ forms an $M_1 \to M_2$ skip. □

It is time to flesh out Sandwich. Choosing function $d(M_1, M_2) := |M_1 \oplus M_2|$ and error $\varepsilon := 4^{\alpha+1}$, we describe a procedure that computes M. Under the loop condition we have $d(M_1, M_2) > 4\,\varepsilon$. We pick an arbitrary cycle C from $M_1 \oplus M_2$.

- If $|C| < d(M_1, M_2)$ then we simply take $M := M_1 \oplus C$. Both $d(M, M_2) = d(M_1, M_2) - |C|$ and $d(M_1, M) = |C|$ are strictly less than $d(M_1, M_2)$.
- Otherwise $|C| = d(M_1, M_2) > 4\,\varepsilon$. Then Lemma 4.3 guarantees an $M_1 \to M_2$ skip D. We take $M := M_1 \oplus D$, thus $d(M, M_2) = d(M_1, M_2) - 4 < d(M_1, M_2)$ and $d(M_1, M) = 8 \ll d(M_1, M_2)$.

4.1.2 Bipartite graphs of small bipartite independence number

For bipartite graph G we have $\alpha \geqslant n$ as each side of the bipartition is an independent set. So the independence number is not a good parameterisation. This motivates the study of *bipartite independence number* β. Call a set $S \subseteq V$ *balanced* if $|S \cap X| = |S \cap Y|$. We define 2β to be the maximum cardinality of a balanced independent set.

With $d(M_1, M_2) := |M_1 \oplus M_2|$ and $\varepsilon := \beta + 1$, we describe a procedure that computes M. Again, under the loop condition we have $d(M_1, M_2) > 4\,\varepsilon$. Pick an arbitrary cycle C from $M_1 \oplus M_2$.

- If $|C| < d(M_1, M_2)$ then we take $M := M_1 \oplus C$ as before.
- If $|C| = d(M_1, M_2) > 4\,\varepsilon$ then we walk along $C =: (v_1, v_2, \ldots, v_{|C|})$, starting from an edge $v_1 v_2 \in M_1$. Without loss of generality we assume $v_1 \in X$. Let $X_0 := \{v_1, v_3, \ldots, v_{2\beta+1}\} \subseteq X$ and $Y_0 := \{v_{2\beta+4}, v_{2\beta+6}, \ldots, v_{4\beta+4}\} \subseteq Y$. Since $X_0 \cup Y_0$ is a balanced set of size $2\,(\beta+1)$, there exist vertices $v_{2i-1} \in X_0$ and $v_{2j} \in Y_0$ that are adjacent. Hence $D := (v_{2i-1}, v_{2i}, \ldots, v_{2j-1}, v_{2j})$ is an M_1-alternating cycle of length at most $4\,\varepsilon$. Take $M := M_1 \oplus D$ and note that $d(M, M_2) = d(M_1, M_2) - |D| + 2$ and $d(M_1, M) = |D|$ are less than $d(M_1, M_2)$.

You might see a pattern in this sort of argument. Before we move on to the next graph class, let us abstract the key structure.

Definition 4.4. A graph is *ε-chordal* if every even cycle C of length $|C| > 4\,\varepsilon$ has a chord that splits C into two odd-length paths.

Hence graphs of bipartite independence number β are $(\beta+1)$-chordal. We may prove the following general theorem:

Theorem 4.5. Consider applying Sandwich on an ε-chordal graph with gap function $d(M_1, M_2) := |M_1 \oplus M_2|$ and error ε. Then under the loop condition we may always find a desired M in quadratic time.

Proof. Pick an arbitrary cycle C from $M_1 \oplus M_2$.

- If $|C| < d(M_1, M_2)$ then we take $M := M_1 \oplus C$.
- If $|C| = d(M_1, M_2) > 4\,\varepsilon$ then there exists a chord e that splits C into two odd-length paths. One of the paths P starts and ends with M_1 edges. Hence $D := P + e$ is an M_1-alternating cycle. We take $M := M_1 \oplus D$. □

4.1.3 Graphs of small neighbourhood diversity

A graph has *neighbourhood diversity* at most t if we can partition its vertices into t bags such that

(i) the induced subgraph on every bag is either complete or empty; and

(ii) there is either no edge or full edges between every two bags.

We show that these graphs are $t/2$-chordal. Let $C=(v_1,v_2,\dots,v_{|C|})$ be a cycle of length $|C|>2t$. Consider the more than t vertices at even indices. By pigeon-hole principle, two such vertices v_{2i} and v_{2j} must lie in the same bag b. Assume that v_{2i-1} is in bag a (possibly $a=b$). From $v_{2i-1}v_{2i}\in E$ and properties (i)(ii), we deduce that $v_{2i-1}v_{2j}\in E$ as well. This is exactly the chord we are looking for.

4.1.4 Graphs of small modular width

As a generalisation, we consider a "recursive version" of neighbourhood diversity. A graph has *modular width* at most t if we can partition its vertices into t bags such that

(i) the induced subgraph of every bag is either complete, empty, or has modular width at most t; and

(ii) there is either no edge or full edges between every two bags.

In other words, we allow each bag to contain sub-bags recursively. This accommodates much richer graph structures. Unfortunately the previous argument breaks when $a=b$, as we cannot guarantee $v_{2i-1}v_{2j}\in E$.

Here is the tweak. Let $C=(v_1,v_2,\dots,v_{|C|})$ be a cycle of length $|C|>2t$. If all vertices on C lie in the same top-level bag, then we "zoom in" and inspect its sub-bags. If still all vertices on C lie in the same sub-bag then we recurse likewise. Eventually we either descend to a level where vertices begin to differentiate, or we hit the bottom and could not descend any more. In the latter case $G[\{v_1,\dots,v_{|C|}\}]$ must be complete because it is non-empty, thus we may take for example v_1v_4 as the desired chord.

It remains to consider the former case. Look at the vertices at even indices. By pigeon-hole principle there are vertices v_{2i} and v_{2j} in the same bag b (at current level).

- If v_{2i-1} is in a different bag, then we deduce from property (ii) that $v_{2i-1}v_j\in E$ is a desired chord.
- If v_{2i+1} is in a different bag, then we deduce from property (ii) that $v_{2i+1}v_j\in E$ is a desired chord.
- Otherwise v_{2i-1},v_{2i},v_{2i+1} are all in bag b. Continue walking along the cycle as long as we are in bag b. Eventually we will get to some vertex v_ℓ in a different bag $a\neq b$. Note that v_ℓ is adjacent to every vertex in bag b due to property (ii). In particular $v_{\ell-3}v_\ell\in E$ is a desired chord.

In contrast to all graph classes mentioned before, the last case highlights a tricky structure: we have no control over the parity of ℓ. So the chord $v_{\ell-3}v_\ell$ completes an M_1-alternating cycle with an unknown half of C. This will become a hurdle if we wish to jump to a k-PM using the method in Section 4.4.

Of particular interest is a subclass called *cographs*. These are graphs that do not contain P_4 (path of length three) as an induced subgraph. Equivalently, they are graphs of modular width at most two. The tricky structure manifests already in cographs, and we do not know any efficient deterministic algorithm that finds k-PM in them.

4.1.5 Planar 1-extendable bipartite graphs

Finally we investigate a graph class that needs special argument. A graph is *1-extendable* if it is connected and every edge appears in some perfect matching. Such graphs play a key role in the structural decomposition of perfectly matchable graphs. We refer to Chapters 5 and 6 in Lovász and Plummer [18] for more information, although we do not use any property beyond the definition.

We assume in addition that the graph is bipartite and planar, in which every face is bounded by at most t vertices. These include the square and hexagonal tilings of finite size; the former was mentioned in Haslebacher [11].

Take $\varepsilon := t/4$ and define the gap function as

$$d(M_1, M_2) := \sum_{C \subseteq M_1 \oplus M_2} \operatorname{area}(C),$$

where $\operatorname{area}(C)$ is the area bounded by the Jordan curve C.

- If there are two or more cycles in $M_1 \oplus M_2$ then we take any of them, say C, and let $M := M_1 \oplus C$. Since both $M \oplus M_2$ and $M_1 \oplus M$ are proper subsets of $M_1 \oplus M_2$, the area sums are strictly smaller.
- So assume there is only one cycle $C = M_1 \oplus M_2$. Under the loop condition its length is more than $4\varepsilon > t$, so it does not bound a face. In particular, some vertex $u \in C$ has a neighbour v in the interior of C. Consider a perfect matching $M_3 \ni uv$, which exists by 1-extendability. We observe that the symmetric difference $M_1 \oplus M_3$, when restricted to the interior of C, is an $\{M_1, M_3\}$-alternating path P that starts and ends with M_3 edges touching the cycle C. Therefore, this P together with one half of C forms an M_1-alternating cycle D. By letting $M := M_1 \oplus D$ we are able to reduce the area in both $M \oplus M_2$ and $M_1 \oplus M$.

4.2 The Parity Variant

Is there a graph class where the error term ε can be killed? Haslebacher [11] isolated an important class called doubly chordal graphs. Note that property (i) in the definition below means that they are 1-chordal (cf. Definition 4.4).

Definition 4.6. A graph is *doubly chordal* if

(i) every even cycle $C = (v_1, \ldots, v_\ell)$ of length $\ell \geqslant 6$ has a chord $v_i v_j$ where i and j have different parity;

(ii) every even cycle $C = (v_1, \ldots, v_\ell)$ of length $\ell \geqslant 8$ has two chords $v_a v_d$ and $v_b v_c$ where v_a, v_b, v_c, v_d appear in circular order and $a \equiv c \not\equiv b \equiv d \pmod{2}$.

Yet we cannot apply Sandwich directly with $\varepsilon := 0$ on such graphs. Think about the scenario where $|M_1 \cap R| = k - 1 < k + 1 = |M_2 \cap R|$ and $M_1 \oplus M_2$ is a 4-cycle. To get to a k-PM, one must correct the parity via some external alternating cycles of total weight 1, which are challenging to find. Hence it was devised to avoid the parity issue altogether by approaching the target in even strides [11].

Algorithm Parity-Sandwich

```
assume $k_{\min} \leqslant k \leqslant k_{\max}$
```

```
let $M_1$ be a perfect matchings with $\leqslant k$ reds and $|M_1 \cap R| \equiv k \pmod 2$
let $M_2$ be a perfect matchings with $\geqslant k$ reds and $|M_2 \cap R| \equiv k \pmod 2$
while $|M_2 \cap R| - |M_1 \cap R| \geqslant 4$ do
    compute a perfect matching $M$ such that
        $d(M, M_2), d(M_1, M) < d(M_1, M_2)$ and $|M \cap R| \equiv k \pmod 2$
    if $|M \cap R| \leqslant k$ then $M_1 := M$, else $M_2 := M$
return $M_1$ or $M_2$, depending on which has $k$ reds
```

Note that $|M_1 \cap R| \equiv |M_2 \cap R| \equiv k \pmod 2$ during the loop, so upon termination we have $|M_2 \cap R| - |M_1 \cap R| \leqslant 2$, meaning at least one of M_1, M_2 has k reds.

To implement Parity-Sandwich, we define the gap function

$$d(M_1, M_2) := (|M_1 \oplus M_2|, \ \ \#\text{cycles in } M_1 \oplus M_2) \in \mathbb{N}^2$$

and order the codomain lexicographically: we prefer smaller symmetric difference and, if there is a tie, we prefer more cycles. Next we specify how we compute M in the loop.

- If $M_1 \oplus M_2$ contains more than two cycles, then let $\mathcal{C}_1, \mathcal{C}_2$ be two of them and define $\mathcal{C}_3 := \mathcal{C}_1 \oplus \mathcal{C}_2$. Observe that $\delta(\mathcal{C}_i | M_1)$ is even for some i, hence $M := M_1 \oplus \mathcal{C}_i$ has the correct parity. Regardless of the i that we used, $|M \oplus M_2|, |M_1 \oplus M| < |M_1 \oplus M_2|$, so the gap strictly decreases.
- If $M_1 \oplus M_2$ contains exactly two cycles C, D, then we need to be careful.
  - If both cycles have length 4, then $\delta(C|M_1), \delta(D|M_1) \leqslant 2$. But the loop condition says $\delta(C|M_1) + \delta(D|M_1) = \delta(M_2|M_1) \geqslant 4$, so both have weight exactly 2. Thus we may simply take $M := M_1 \oplus C$.
  - Otherwise, say C has length at least 6, then from property (i) we can shorten it to an M_1-alternating cycle $\mathcal{C}_1$ by a chord. Let $\mathcal{C}_2 := D$ and $\mathcal{C}_3 := \mathcal{C}_1 \oplus \mathcal{C}_2$. Again, $\delta(\mathcal{C}_i | M_1)$ is even for some i, so we may take $M := M_1 \oplus \mathcal{C}_i$. The shortening ensures that gap decreases in all scenarios.
- If $M_1 \oplus M_2$ is a single cycle C then it has length at least 8, so we may find two chords $v_a v_d$ and $v_b v_c$ by property (ii). They make two disjoint cycles $\mathcal{C}_1 := v_a \to v_d \rightsquigarrow v_a$ and $\mathcal{C}_2 := v_b \rightsquigarrow v_c \to v_b$ where "$\rightsquigarrow$" travels along the cycle C. Define $\mathcal{C}_3 := \mathcal{C}_1 \oplus \mathcal{C}_2$. Due to the parity requirements in (ii), these are all M_1-alternating or all M_2-alternating; we may assume the former case by symmetry. Since $\delta(\mathcal{C}_i | M_1)$ is even for some i, we may take $M := M_1 \oplus \mathcal{C}_i$.

  For the gap, $|M \oplus M_2|, |M_1 \oplus M| \leqslant |M_1 \oplus M_2|$ always holds. In fact, when $i \in \{1, 2\}$ the inequalities are strict. The only troublesome case is when $i = 3$ and the two chords sit "back-to-back", i.e. $b = a + 1$ and $d = c + 1$. In this scenario, $|M \oplus M_2| = 4 < |M_1 \oplus M_2|$ but $|M_1 \oplus M| = |\mathcal{C}_3| = |M_1 \oplus M_2|$. Fortunately, $M_1 \oplus M = \mathcal{C}_3$ has two cycles whereas $M_1 \oplus M_2 = C$ has only one. So we indeed have $d(M_1, M), d(M, M_2) < d(M_1, M_2)$.

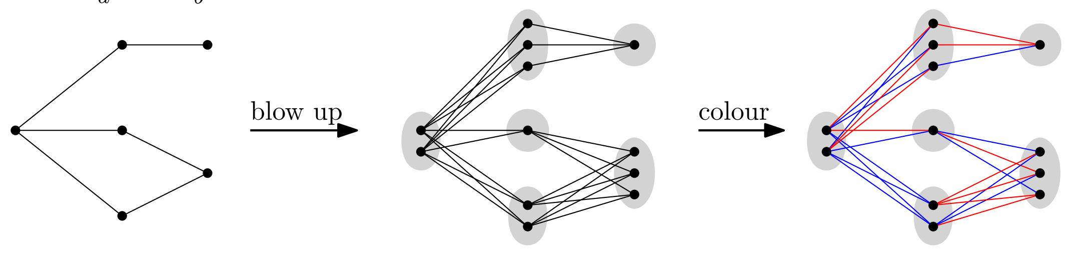
At this point we are still missing a crucial piece: how do we initialise the M_1, M_2 before the loop? This is a highly non-trivial problem. For bipartite graphs, one can use a $k_{\min}$-PM (resp. $k_{\max}$-PM) and correct its parity via a dynamic program [7]. For unit-interval graphs there is a polynomial time algorithm, too [11]. However, for general graphs the problem is widely open.

Haslebacher [11] proceeded to show that chain graphs (which are bipartite) and unit-interval graphs are doubly chordal, hence yielding efficient Parity-Sandwich on these inputs. We will later add another graph class to the list.

4.3 Lifting Theorems

Having seen many graphs on which the sandwich paradigm works fairly well, we might wonder if we can use them as building blocks to generate more such graphs. This section develops two "lifting" theorems of this kind. The core operation for building new graphs is defined below.

Definition 4.7. A *blow-up* of graph H is constructed as follows. Each node a in H is duplicated as a set of vertices S_a. If $a\,b$ is an edge in H then we fully connect all pairs between S_a and S_b.



Theorem 4.8. Let $\varepsilon \in \mathbb{N}$. A blow-up of a bipartite ε-chordal graph is again bipartite ε-chordal.

Proof. Let G be a blow-up of a bipartite ε-chordal graph H. Note that G is bipartite: all vertices in S_a lie in the same part of a, for every node $a \in H$. It remains to show that G is ε-chordal.

Take an even cycle $C = (v_1, v_2, \ldots, v_\ell)$ in G of length $\ell > 4\,\varepsilon$. We project it back to H and obtain a closed *walk* $(a_1, a_2, \ldots, a_\ell)$ in H. Note that $v_i \in S_{a_i}$ for all i.

- If the closed walk is a cycle, then it has a chord $a_i\,a_j$ because H is ε-chordal. This chord naturally gives rise to a chord $v_i\,v_j$ of C.
- Otherwise, some node is visited twice in the walk, say $a_i = a_j$. Consider the vertices v_{i-1} and v_{i+1} adjacent to v_i. By definition of a blow-up, both are adjacent to v_j as well since $a_i = a_j$. But it cannot happen that $v_{i-1}\,v_j \in C$ and $v_{i+1}\,v_j \in C$ simultaneously, for otherwise $\ell = 4 \leqslant 4\,\varepsilon$. Hence at least one of these edges is a chord of C.

In both cases we found a chord of C, which must split C into two odd-length paths due to bipartiteness. □

The theorem is trivial when applied to graphs of modular width t because blow-ups preserve the parameter. On the other hand, it teaches us something valuable when applied to graphs of bipartite independence number β, because the parameter can become arbitrarily large in a blow-up.

Theorem 4.9. A blow-up of a tree is bipartite doubly chordal.

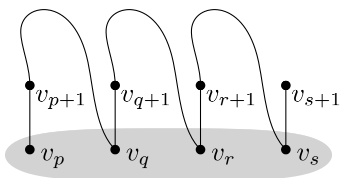
Proof. Let T be a tree. Assume that each node a is blown up to S_a. The resulting graph G is clearly bipartite, so there is no need to check parity when we verify Definition 4.6.

Fix an arbitrary cycle $C = (v_1, v_2, \ldots, v_\ell)$. We project it back to T and obtain a closed walk on the tree, which spans a subtree $T' \subseteq T$ on at least two nodes.

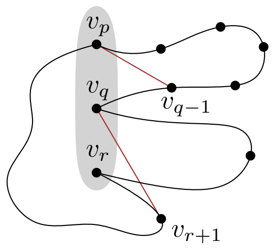
(i) Assuming length $\ell = 6$, we will find a chord in C. Take a leaf $b \in T'$ and a neighbouring node $a \in T'$. Apparently there exists i such that $v_i \in S_a$, $v_{i+1} \in S_b$ and $v_{i+2} \in S_a$. Now consider the next vertex v_{i+3}. It must be adjacent to all vertices in S_a, in particular to v_i. But the edge $v_i\,v_{i+3}$ is not in C because $\ell > 4$, so it is a chord.

(ii) Assuming length $\ell \geqslant 8$, we will find two chords in C that satisfy the definition. Let $a \in T'$ be a node with maximum $|C \cap S_a|$.

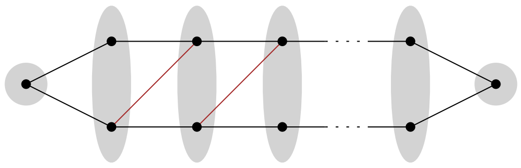
- If $|C \cap S_a| \geqslant 4$ then we assume that v_p, v_q, v_r, v_s are four consecutive visits to S_a. Hence the four successors $v_{p+1}, v_{q+1}, v_{r+1}, v_{s+1}$, though not necessarily project to the same node in T', are adjacent to all the four vertices. In particular $v_p v_{q+1}$ and $v_r v_{s+1}$ are two desired chords.



- If $|C \cap S_a| = 3$ then we assume that v_p, v_q, v_r are the three visits to S_a. They split C into three subpaths: $(v_p, v_{p+1}, \ldots, v_q)$, $(v_q, v_{q+1}, \ldots, v_r)$ and $(v_r, v_{r+1}, \ldots, v_p)$, each of even length. Some of them must have length more than 2 because $\ell \geqslant 8$. Say $(v_p, v_{p+1}, \ldots, v_q)$ is the path, thus $v_{q-1} \neq v_{p+1}$. Then the edge $v_p v_{q-1}$ is a chord. For the other chord, we may take $v_q v_{r+1}$.

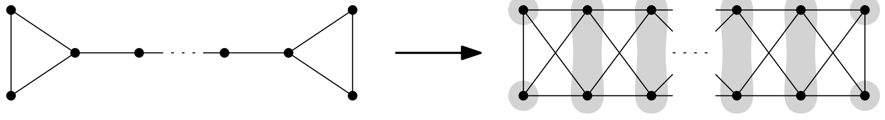


- If $|C \cap S_a| = 2$ then T' must be a path and C has the following form:



There are at least three intermediate layers because $\ell \geqslant 8$. So we may take the two chords shown in brown in the figure. □

The previous two theorems deal with blow-ups of bipartite graphs. In contrast, the blow-up of a non-bipartite ε-chordal graph can be much more complicated. Consider the pathological example below. The graph on the left is trivially doubly chordal, but its blow-up on the right is not ε-chordal for any constant ε because it contains a long outer cycle whose chords all have wrong parity.



Nevertheless, we are able to prove a lifting theorem for graphs of small independence number. Of special interest is that the argument blends two different types of properties.

Theorem 4.10. Suppose H is a graph of independence number α, and let $\varepsilon := 10\,\alpha \cdot 4^{\alpha+1}$. Then Sandwich is capable of finding a $(k \pm \varepsilon)$-PM in any blow-up G of H.

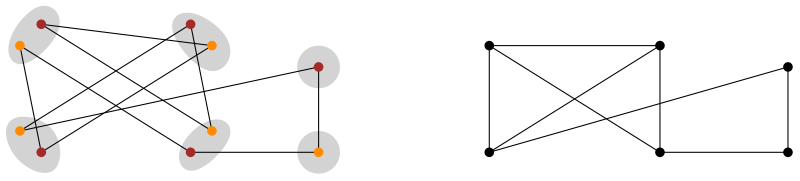
Proof. Take $d(M_1, M_2) := |M_1 \oplus M_2|$ as usual. We describe a procedure that computes M in Sandwich.

Under the loop condition we have $d(M_1, M_2) > 4\,\varepsilon$. We pick a cycle $C = (v_1, v_2, \ldots, v_\ell)$ from $M_1 \oplus M_2$. If $\ell < d(M_1, M_2)$ then we take $M := M_1 \oplus C$. Else $\ell = d(M_1, M_2) > 4\,\varepsilon$. Project the cycle to a closed walk $(a_1, a_2, \ldots, a_\ell)$ on H, which spans a subgraph $H' \subseteq H$.

> **Observation:** If $a_i = a_j$ for some distinct indices $i \equiv j \pmod 2$ then we consider v_{i-1} and v_{i+1}, both adjacent to v_i and thus to v_j as well. Note that at least one of the edges $v_{i-1} v_j$ and $v_{i+1} v_j$ is a chord of C because $\ell > 4$. The chord splits C into two odd-length paths, and together with one of the paths gives an M_1-alternating cycle D. So we may set $M := M_1 \oplus D$.

We can now assemble our main argument.

- First assume that some node is visited three times, say $a_p = a_q = a_r$. Then two of the three indices have the same parity, thus we may apply the observation.
- Next assume that every node is visited at most twice, thus H' has maximum degree $\Delta \leqslant 4$. Let A be the nodes visited exactly twice. By the observation, we only need to deal with the case where every $a \in A$ appears in the walk at both even and odd indices.



cycle C graph H' of the projected cycle

  - If $|A| > 5\,\alpha$ then we find a maximal independent set $A_0 \subseteq A$ in $H'[A]$ (hence also in H'). Note that $|A_0| \geqslant \frac{|A|}{\Delta + 1} > \alpha$, so there exist two nodes $a, b \in A_0$ that are adjacent in H (but not in H'). Take even indices i, j such that $a_i = a$ and $a_j = b$. Then the two vertices v_i, v_j are adjacent in G but not in C. That is, $v_i v_j$ is a chord of C. From here our usual argument applies: The chord and one half of C completes an M_1-alternating cycle that we may use to modify M_1.
  - If $|A| \leqslant 5\,\alpha$ then we remove the corresponding vertices from C. This breaks it into at most $10\,\alpha$ paths, one of which has length at least $\frac{\ell}{10\,\alpha} > 4^{\alpha+2}$. So we may apply Lemma 4.3 to find an $M_1 \rightarrow M_2$ skip D. We then take $M := M_1 \oplus D$. □

4.4 Educated Jump

Although Parity-Sandwich can eliminate the error for doubly chordal graphs, the trick is too specific. For most other instances we have to live with error in the output. The problem then becomes: Can we jump from a $(k \pm \varepsilon)$-PM M to some k-PM M^* directly?

In the rest of the section, let us write $\mathcal{C} := M \oplus M^*$. Note that $-\varepsilon \leqslant \delta(\mathcal{C} | M^*) \leqslant \varepsilon$, but it does not necessarily imply a small $|\mathcal{C}|$.

Suppose we managed to show $|\mathcal{C}| \leqslant c(\varepsilon)$ for some function c. Then we can jump to M^* by trying every possible $\mathcal{C}$ of size at most $c(\varepsilon)$ and applying it to M. The running time is $n^{O(c(\varepsilon))}$, which lands in the complexity class **XP**.

Alternatively, suppose we managed the weaker claim that $|\mathcal{C} \cap R| \leqslant c(\varepsilon)$ or $|\mathcal{C} \cap B| \leqslant c(\varepsilon)$. Still we can jump to M^* in $n^{O(c(\varepsilon))}$ time:

Guess a set $R_0 \subseteq R$ of size $\leqslant c(\varepsilon)$. Pretend that $R_0 = \mathcal{C} \cap R$. Recover $M^* \cap R = (M \cap R) \oplus R_0$. Verify (i) $|M^* \cap R| = k$, and (ii) there exist $n-k$ disjoint blues that do not touch $M^* \cap R$. If not then we guess again.

If none of the guesses work, then we switch the roles of red and blue, replace k with $n-k$, and restart the game.

For graphs of independence number α, the "closeness claims" were found in [6, 7]. Here we will sketch the proof qualitatively and skip detailed calculations. We loosely say that a quantity is "small" if its absolute value is bounded by a function of α.

Definition 4.11. An $M^* \to M$ *kit* is a collection $\mathcal{D}$ of disjoint cycles with $\delta(\mathcal{D}|M^*) = 0$. Each cycle needs to be an $M^* \to M$ skip or come from $M \oplus M^*$.

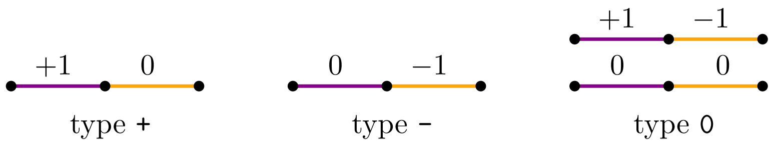
Theorem 4.12. If there is no $M^* \to M$ kit then

(i) every cycle C in $\mathcal{C}$ has small weight $\delta(C|M^*)$;
(ii) the number of cycles in $\mathcal{C}$ is small;
(iii) $|\mathcal{C} \cap R|$ or $|\mathcal{C} \cap B|$ is small.

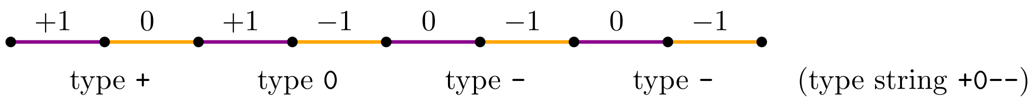
From here a closeness claim follows: Whenever $|\mathcal{C} \cap R|$ and $|\mathcal{C} \cap B|$ are large, there is an $M^* \to M$ kit $\mathcal{D}$ by Theorem 4.12. Conceptually applying $\mathcal{D}$ on M^* shall preserve the number of reds while reducing $|M \oplus M^*| = |\mathcal{C} \cap R| + |\mathcal{C} \cap B|$. So eventually one of $|\mathcal{C} \cap R|$ and $|\mathcal{C} \cap B|$ has to be small.

Let us now overview the proof of Theorem 4.12. Suppose the conclusion does not hold, then we seek many M^*-alternating cycles of small positive weights, and also many of small negative weights. One can choose a subset of these cycles to balance the sign. This gives an $M^* \to M$ kit that contradicts the assumption.

The main device in finding the positive/negative cycles is Lemma 4.3. But now we care about the sign of the weight, so some refinement is needed. For an $\{M, M^*\}$-alternating path, we always group two consecutive edges as a whole. Four configurations are possible:



The *type string* of a path is naturally the sequence of types when we walk through the path in stride of two; see the figure below for illustration. We will use regular expressions such as `+0*+` to denote a string that starts with `+`, continues with an indefinite number of `0`'s, and ends with `+`.



Lemma 4.13. Every 4^α disjoint $\{M, M^*\}$-alternating paths with type string listed in the first column, give rise to an $M^* \to M$ skip with weight stated in the second column:

disjoint paths of type...	yield a skip of weight...
`+0*+`	$2 \leqslant \delta \leqslant 4$
`-0*-`	$-4 \leqslant \delta \leqslant -2$
`00` led by blue / `+0*-`	$0 \leqslant \delta \leqslant 2$
`00` led by red / `-0*+`	$-2 \leqslant \delta \leqslant 0$

We omit the proof as it mirrors the argument of Lemma 4.3. Two corollaries are in order.

Corollary 4.14. An M^*-alternating cycle C of large weight $\delta(C|M^*)$ leads to plenty of disjoint $M^* \to M$ skips, all in the same sign of $\delta(C|M^*)$.

Proof. Assume for example $\delta(C|M^*) \gg 0$. Scan the type string of C in one direction. Whenever we see the pattern `+0*+`, we mark it and continue with the remaining string. Observe that the path between neighbouring marks has weight $\leqslant 0$, and every mark contributes weight 2. So there must be a large number of marks. Applying Lemma 4.13 (first row) finishes the proof. □

Corollary 4.15. An M^*-alternating cycle C of small weight $\delta(C|M^*)$ but plenty of reds (resp. blues) leads to plenty of disjoint skips of negative (resp. positive) weights.

Proof. We split the type string of C in two passes. In the first pass we split by `+0*+` (called positive marks). In the second pass we further split by `-0*-` (called negative marks). Every resulting fragment has form `(+0*-0*)*` or `(-0*+0*)*`; that is, has alternating signs. In particular, it has weight in $\{-1, 0, 1\}$ which is smaller in amplitude than a mark.

- The number of marks is large. Then both positive and negative marks are abundant, since otherwise $\delta(C|M^*)$ cannot be small. Applying Lemma 4.3 (first and second rows) gives us plenty of positive and negative skips.
- The number of marks is small. Let us consider all the fragments as well as the `0*` sections in the marks. Say C contains plenty of reds. Note that all but a small number of reds appear in these paths, so at least one path P contains many reds. Let c count the total number of `+` and `-` in P.
  - If c is large then P contains a large number of `-0*+` subpaths.
  - If c is small then P contains a small number of maximal `0*` subpaths. One of them contains many reds; every other red leads a `00` path.

  Either way, Lemma 4.13 (last row) gives us many negative skips. □

Proof of Theorem 4.12. Suppose some cycle in $\mathcal{C}$ has large positive weight, say. It produces many $M^* \to M$ skips of positive weights by Corollary 4.14. On the other hand, recall that $-\varepsilon \leqslant \delta(\mathcal{C}|M^*) \leqslant \varepsilon$ is small, so either there is a cycle in $\mathcal{C}$ of large negative weight (which produces many $M^* \to M$ skips of negative weights again by Corollary 4.14), or there are many cycles in $\mathcal{C}$ of small negative weights. Anyway, we may choose a subset of these positive and negative cycles to balance the sign, contradicting that an $M^* \to M$ kit does not exist. We have proven property (i).

Next suppose the number of cycles in $\mathcal{C}$ is large. Then there are many positive *and* many negative cycles because of property (i) and that $\delta(\mathcal{C}|M^*)$ is small. So there is a subset of cycles that balance the sign, contradicting that an $M^* \to M$ kit does not exist. We have established property (ii).

Finally, suppose both $|\mathcal{C} \cap R|$ and $|\mathcal{C} \cap B|$ are large. Then property (ii) implies that $\mathcal{C}$ contains a cycle C with many reds, and a cycle C' with many blues. Note that C and C' could be the same; in that case we simply cut it into two paths without compromising the properties, and our remaining argument gets through. By Corollary 4.15, C implies many negative $M^* \to M$ skips, and C' implies many positive $M^* \to M$ skips. They together contain an $M^* \to M$ kit, a contradiction. Hence we have shown property (iii). □

The analyses above treated M and M^* as blackboxes: we only used the fact that they have $k \pm \varepsilon$ and k reds, respectively. But in reality we have more control over M. By tweaking SANDWICH one can actually generate an M so that $|\mathcal{C}| = |M \oplus M^*|$ is provably small, which strengthens Theorem 4.12. Combined with a trick called colour coding, we can turn it into an algorithm that finds M^* in **FPT** time $c(\alpha)^{O(c(\alpha))} \cdot \text{poly}(n)$ when the input graph is bipartite [7].

So what is the tweak of SANDWICH? Here is a sketch. We define the gap function as

$$d(M_1, M_2) := (|M_2 \cap R| - |M_1 \cap R|,\ |M_1 \oplus M_2|) \in \mathbb{N}^2,$$

ordered lexicographically. In each iteration we try one of the four modifications:

(1) add two disjoint reds to $M_1 \cap R$, so that all vertices unmatched by reds can be matched up by blues;
(2) remove two reds from $M_2 \cap R$, so that all vertices unmatched by reds can be matched up by blues;
(3) find and apply an $M_1 \to M_2$ kit;
(4) find and apply an $M_2 \to M_1$ kit.

If none would work then we give up and exit the loop.

Theorem 4.16. $|M_1 \oplus M_2|$ is small upon exit.

Proof. The proof of Theorem 4.12 can be used to show

(i) every cycle C in $M_1 \oplus M_2$ has small weight $\delta(C|M_1)$;
(ii) the number of cycles in $M_1 \oplus M_2$ is small;
(iii) $|(M_1 \oplus M_2) \cap R|$ or $|(M_1 \oplus M_2) \cap B|$ is small.

Assume that $|(M_1 \oplus M_2) \cap R|$ is small while $|(M_1 \oplus M_2) \cap B|$ is large. Then by (ii) we know that there is a cycle C in $M_1 \oplus M_2$ with many blues (and few reds). The reds split C into blue paths, one of which (denoted P) is long. We apply Lemma 4.13 on two disjoint sections of P, which yields an $M_1 \to M_2$ skip C_i for $i = 1, 2$. Note that $\delta_i := \delta(C_i|M_1) \in \{0, 1, 2\}$ since all but possibly the two chords are blue. If $\delta_i = 0$ then trial (3) should have succeeded; and if $\delta_1, \delta_2 \in \{1, 2\}$ then trial (1) should have succeeded. So we reach a contradiction. The case that $|(M_1 \oplus M_2) \cap R|$ is large while $|(M_1 \oplus M_2) \cap B|$ is small can be treated symmetrically. □

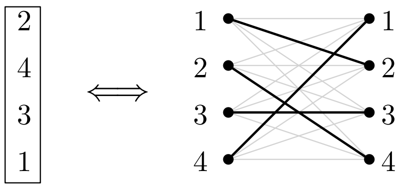
Now denote $\mathcal{C} := |M_1 \oplus M^*|$. With a more involved application of Ramsey theory, one can show that $|\mathcal{C}|$ is small for some M^*. Basically, if $|\mathcal{C} \cap B|$ is large then we can find a local structure that contains either an $M^* \to M_1$ kit or two $M_1 \to M^*$ skips of positive weights. In the former case we reduce $|\mathcal{C}|$. In the latter case trial (1) should have succeeded, which is impossible. Symmetrically, if $|\mathcal{C} \cap R|$ is large then we can find a local structure that contains either an $M^* \to M_2$ kit or two $M_2 \to M^*$ skips of negative weights. In the former case we reduce $|\mathcal{C}|$. In the latter case trial (2) should have succeeded, which is impossible.

5 Algebraic Algorithms

The only successful attack of Exact Perfect Matching to date uses algebraic encoding and randomisation [20]. Beneath the algebra, however, runs an elegant combinatorial argument where alternating cycles re-enter at a new level. Inspecting the argument, one can derandomise it for planar graphs and more generally $K_{3,3}$-free graphs [15, 25]. This chapter will cover the techniques in detail. Denote by S_n the set of all permutations of $\{1, \ldots, n\}$.

5.1 Algebraic Encoding for Bipartite Matching

Every permutation $\pi \in S_n$ can be interpreted as a perfect matching $\{1\pi(1), \ldots, n\pi(n)\}$ in the complete bipartite graph $K_{n,n}$, and vice versa:



Assume G is bipartite. Let us represent it by an $n \times n$ matrix A,

$$A_{uv} := \begin{cases} x_{uv}\, y & uv \in R \\ x_{uv} & uv \in B \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{x} := (x_e)_{e \in E}$ and y are variables. Recall that the determinant of A is defined as

$$\det A := \sum_{\pi \in S_n} \operatorname{sign}(\pi) \prod_{u=1}^{n} A_{u,\pi(u)}.$$

Here $\operatorname{sign}(\pi) \in \{-1, +1\}$ is the parity of π, but its precise definition is inconsequential until very late. The trailing product reflects how π fits in G. If the product is zero, then some edge $u\pi(u)$ is absent and thus π is not a perfect matching in G. If the product contributes some non-zero monomial $(\prod_{u=1}^{n} x_{u\pi(u)}) \cdot y^d$, then π is a d-PM in G. Because the monomials contributed from different π's are distinct, they cannot cancel each other however the sign is defined.

Overall $\det A$ is a giant polynomial in $\boldsymbol{x}$ and y. We may group it as $\sum_{d=0}^{n} c_d(\boldsymbol{x})\, y^d$ where each $c_d(\boldsymbol{x})$ is a multilinear polynomial in $\boldsymbol{x}$ and tracks information of all d-PMs. It follows that a k-PM exists iff $c_k(\boldsymbol{x}) \not\equiv 0$.

The only obstacle is the *computation* of $c_k(\boldsymbol{x})$. One might suggest computing $\det A$ symbolically and then deriving $c_k(\boldsymbol{x})$, but this is too costly. A Gaussian elimination on symbolic matrices, for example, will produce intermediate quotients of exponentially high degrees. More to the point, the $c_k(\boldsymbol{x})$ itself might contain exponentially many monomials, which makes it difficult to represent in the first place! So can we decide $c_k(\boldsymbol{x}) \not\equiv 0$ without computing the polynomial? From here the story unfolds…

5.2 Detection of Non-Zero Polynomials

Consider what happens if we evaluate $c_k(\boldsymbol{x})$ at a random point. If $c_k(\boldsymbol{x}) \equiv 0$ then the result is zero, of course. If $c_k(\boldsymbol{x}) \not\equiv 0$ then the result ought to be non-zero with positive probability. In fact quite significant a probability, as the lemma below shows.

The Schwartz-Zippel Lemma. Let $f \not\equiv 0$ be a polynomial in $\boldsymbol{x} = (x_1, \ldots, x_t)$ over finite field $\mathbb{F}$. Assume its degree is at most n. If we sample $\boldsymbol{r} = (r_1, \ldots, r_t) \in \mathbb{F}^t$ uniformly at random, then $\mathbb{P}(f(\boldsymbol{r}) = 0) \leqslant n / |\mathbb{F}|$.

Proof. By induction on the number of variables t. The base case $t = 1$ follows from the fact that any univariate polynomial $f \not\equiv 0$ of degree n has up to n roots. Next we proceed from t to $t+1$. Decompose $f(\boldsymbol{x}) =: \sum_{i=0}^{\ell} f_i(x_1, \ldots, x_t) \cdot x_{t+1}^i$ where $f_\ell \not\equiv 0$. Note that $\deg(f_\ell) + \ell \leqslant n$. We write

$$\mathbb{P}(f(\boldsymbol{r}) = 0) \leqslant \mathbb{P}(f_\ell(r_1, \ldots, r_t) = 0) + \mathbb{P}(f(\boldsymbol{r}) = 0 \mid f_\ell(r_1, \ldots, r_t) \neq 0).$$

The first term is at most $\deg(f_\ell) / |\mathbb{F}|$ by induction hypothesis. The second term is at most $\ell / |\mathbb{F}|$ because under any additional condition $r_1, \ldots, r_t$ compatible with $f_\ell(r_1, \ldots, r_t) \neq 0$, the univariate polynomial $g(x_{t+1}) := f(r_1, \ldots, r_t, x_{t+1}) \not\equiv 0$ has at most ℓ roots. Summing the two terms up, we conclude

$$\mathbb{P}(f(\boldsymbol{r}) = 0) \leqslant \frac{\deg(f_\ell) + \ell}{|\mathbb{F}|} \leqslant \frac{n}{|\mathbb{F}|}. \qquad \square$$

The probability dichotomy suggests a randomised algorithm that decides whether a k-PM exists.

Algorithm Random-Probe

- Fix a finite field $\mathbb{F}$ that is large enough, say $|\mathbb{F}| \geqslant 2n$.
- Instantiate each variable x_{uv} by a uniform random $r_{uv} \in \mathbb{F}$. It reduces $\det A$ to an implicit univariate polynomial $g(y) := \sum_{d=0}^{n} c_d(\boldsymbol{r})\, y^d$. Our goal is to extract $c_k(\boldsymbol{r})$.
- Take any $n+1$ distinct elements $a_1, \ldots, a_{n+1} \in \mathbb{F}$. For each $i = 1 \ldots n+1$, instantiate variable y by a_i and then evaluate $\det A$ numerically, which gives us $g(a_i)$.
- Use Lagrange interpolation to recover the coefficients $c_0(\boldsymbol{r}), \ldots, c_n(\boldsymbol{r})$ of the polynomial $g(y)$ from its evaluations $g(a_1), \ldots, g(a_{n+1})$. If $c_k(\boldsymbol{r}) \neq 0$ then answer "yes"; otherwise answer "no".

We remark in passing that the computation can be parallelised via a shallow circuit that evaluates the determinant [2].

As discussed earlier, if there is no k-PM then the algorithm always says "no". On the other hand, if there is a k-PM then with probability at least $1 - n/|\mathbb{F}| \geqslant 1/2$ we have $c_k(\boldsymbol{r}) \neq 0$ and hence the algorithm says "yes". By independent repetitions, the error probability can be made arbitrarily small.

Using Random-Probe as a blackbox, we may recover a solution by processing the red edges sequentially. For each $e \in R$, we ask the algorithm if $G - e$ admits a k-PM. If yes, then e is not necessary for a solution and we purge it from G. Else we add e to our solution, remove its two vertices and their incident edges from G, and decrease k by one. We stop when k drops to 0. Find a blue perfect matching in the remaining graph and piece it together with the reds that we collected so far. They would complete a k-PM.

5.3 The Struggle Against Cancellation

Before moving on, we reflect on the role of randomness. Assume $c_k(\boldsymbol{x}) \not\equiv 0$ is over a large field. All we need is a point—be it random or not—that avoids all the roots. The Schwartz-Zippel lemma assures us that the *majority* of the space will do. Randomly picking a point is a lazy yet effective strategy. But can we hunt for the point deterministically? To this end we must work beyond Schwartz-Zippel and understand the root distribution of $c_k(\boldsymbol{x})$.

Here is a first attempt. Recall that $c_k(\boldsymbol{x})$ is multilinear. It is not hard to show that the evaluations on a box $\{a, b\}^E$ (for any distinct $a, b \in \mathbb{F}$) uniquely determine a multilinear polynomial. So the evaluations cannot be all zero, for otherwise we would have $c_k(\boldsymbol{x}) \equiv 0$. Therefore, we are guaranteed to find the point within $\{a, b\}^E$.

Can we narrow the search space even further? Let us concretely assume $\operatorname{char} \mathbb{F} > n!$ and take a closer look at $c_k(\boldsymbol{x})$. It is a summation of all the (signed) products corresponding to k-PMs. Ideally, if all signs are positive, then the point $\mathbf{1} := (1, \ldots, 1)$ is certainly not a root. More generally, provided unequal occurrences of positive and negative signs in the summands, the point $\mathbf{1}$ remains valid. The worst scenario is when positive and negative occurrences are balanced, so any point without fluctuation (e.g. $\mathbf{1}$) suffers from cancellation.

What if we perturb the point $\mathbf{1}$ at coordinate uv? This will bias the contribution of every k-PM that contains edge uv, thus seemingly unbalance the sum. However, it might happen that half of these k-PMs occur positively whereas the other half occur negatively, so that cancellation remains a problem.

It turns out that we cannot handle the issue if we treat $\operatorname{sign}(\pi) \in \{-1, +1\}$ as a blackbox. At the end of this chapter, we will explain how to leverage the definition of $\operatorname{sign}(\pi)$ to our advantage and battle against cancellation for planar graphs G. It is open, however, whether the same can be done for denser graphs.

Our struggle sharply contrasts the simplicity and elegance of a random point where cancellation "just doesn't happen" even without knowing $\operatorname{sign}(\pi)$. The next section showcases the power of randomness once more.

5.4 A Fully-Parallel Version

We first present a powerful probabilistic lemma. Let $\mathcal{F} \subseteq 2^E$ be a non-empty family of edge subsets. For each edge $e \in E$, we assign a weight $w(e) \in \{1, \ldots, 2|E|\}$ uniformly and independently. The weight extends to every subset $F \subseteq E$ naturally: $w(F) := \sum_{e \in F} w(e)$.

The Isolation Lemma [20]. In this random experiment, the subset $F \in \mathcal{F}$ attaining the minimum weight is *unique* with probability at least $1/2$.

Proof. Condition on the weights of all edges but a particular $e \in E$. Partition $\mathcal{F}$ into two subfamilies

$$\mathcal{F}_1 := \{F \in \mathcal{F} : e \in F\}, \quad \mathcal{F}_0 := \{F \in \mathcal{F} : e \notin F\}$$

and define two thresholds

$$t_1 := \min_{F\in\mathcal{F}_1} w(F-e), \quad t_0 := \min_{F\in\mathcal{F}_0} w(F).$$

Now we sample the weight $w(e)$. It will "complete" the weights in family $\mathcal{F}_1$, but shall not affect the weights in family $\mathcal{F}_0$. Therefore,

- if $w(e)+t_1<t_0$ then every min-weight subset $F\in\mathcal{F}$ comes from $\mathcal{F}_1$, i.e. contains e;
- if $w(e)+t_1>t_0$ then every min-weight subset $F\in\mathcal{F}$ comes from $\mathcal{F}_0$, i.e. excludes e;
- if $w(e)+t_1=t_0$ then there could be ambiguity.

But the third event happens with probability at most $1/(2|E|)$. Since this holds under any condition, we may safely remove the conditioning. Then we apply a union bound over all $e\in E$ to conclude that, with probability at least $1/2$, every single edge is either included in *all* min-weight subset $F\in\mathcal{F}$ or excluded from *all* of them. In that case, the min-weight subset must be unique. $\square$

In our context, imagine $\mathcal{F}$ to be the collection of all k-PMs. Assign each edge $e\in E$ a uniform and independent weight $w(e)\in\{1,\ldots,2|E|\}$. Then by the isolation lemma, with decent chance there is a unique k-PM M^* of minimum weight $w(M^*)=:w^*$. It will help us avoid cancellation once we instantiate the variables $\boldsymbol{x}$ judiciously. We take $x_e:=2^{w(e)}$, so each k-PM M shall contribute

$$\left(\prod_{e\in M} x_e\right) y^k = \left(2^{\sum_{e\in M} w(e)}\right) y^k = 2^{w(M)}\, y^k$$

in the determinant. Hence $c_k=\sum_{M\in\mathcal{F}} \pm 2^{w(M)}$. Since $|2^{w(M)}/2^{w^*}|$ is some multiple of 2 for all $M\in\mathcal{F}\setminus\{M^*\}$, we must have $c_k=(2t+1)\,2^{w^*}$ for some $t\in\mathbb{Z}$. In particular it is non-zero. Essentially, the uniqueness "isolates" one term and protects it from cancellation. The algorithm is summarised below.

Algorithm Random-Isolate

- Instantiate each variable x_e by $2^{w(e)}$ where $w:E\to\{1,\ldots,2|E|\}$ is sampled uniformly.
- Recover c_k by the "evaluation and interpolation" trick as before. If $c_k\neq 0$ then answer "yes"; otherwise answer "no".

Not only can we parallelise it as before, we may also recover the M^* in parallel. Recall $c_k=(2t+1)\,2^{w^*}$ contains only one odd factor. So by repeated division of 2 until irreducible, we may recover w^*. Now for each edge $e\in E$ in parallel, we ask if the minimum weight among all k-PMs in $G-e$ remains w^*. If yes then $e\notin M^*$; otherwise $e\in M^*$.

5.5 Algebraic Encoding for General Matching

In general graphs there is no bipartition, so we need a new way that relates permutations and perfect matchings. Here is the idea. Every permutation $\pi\in S_{2n}$ can be interpreted as a perfect matching

$$\{\pi(1)\pi(2),\ \pi(3)\pi(4),\ \ldots,\ \pi(2n-1)\pi(2n)\}$$

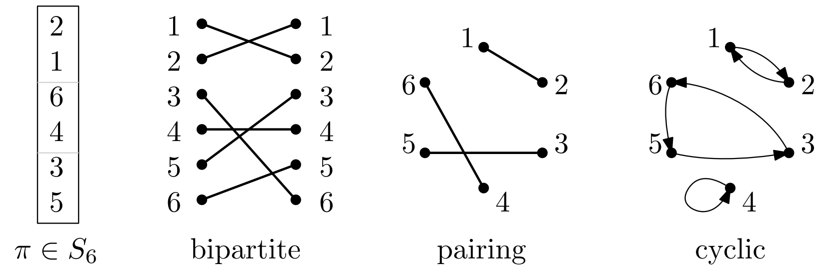
in the complete graph K_n. So the slots $2i-1$ and $2i$ form a "box"; the elements in which are paired. Conversely, however, every perfect matching is encoded by multiple permutations. The multiplicity is $2^n n!$, since there are $n!$ ways to permute the boxes, and then $(2!)^n=2^n$ ways to shuffle elements inside each box.

For convenience we need yet another interpretation of $\pi \in S_{2n}$. This time, it is identified with a directed cycle cover

$$\{1 \rightarrow \pi(1), \ldots, 2\,n \rightarrow \pi(2\,n)\}$$

of the bidirectional complete graph K_{2n} with self loops. Unlike above, this interpretation is bijective.

So far we have collected three interpretations, namely the *bipartite*, *pairing*, and *cyclic* interpretations. They are compared in the illustration below:



Now let us represent G by a $2n \times 2n$ skew-symmetric matrix A known as the *Tutte matrix*. The entries above diagonal are familiar:

$$\forall u \leqslant v, \quad A_{uv} := \begin{cases} x_{uv}\, y & uv \in R \\ x_{uv} & uv \in B \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{x} := (x_e)_{e \in E}$ and y are variables. The entries below diagonal are the negated mirror: $A_{uv} := -A_{vu}$ for all $u > v$. The reason will manifest itself soon. Alongside we invent a counterpart of the determinant, called the *Pfaffian* of A:

$$\operatorname{pf} A := \frac{1}{2^n\, n!} \sum_{\pi \in S_{2n}} \operatorname{sign}(\pi) \prod_{i=1}^{n} A_{\pi(2i-1),\pi(2i)}.$$

Again, the product reveals how π (as in the pairing interpretation) fits in G. If the product is zero, then some edge $\pi(2i-1)\pi(2i)$ is absent and thus π is not a perfect matching in G. On the other hand, if the product contributes non-zero monomial $\left(\prod_{i=1}^{n} x_{\pi(2i-1),\pi(2i)}\right) \cdot y^d$, then π is a d-PM in G. Different π's contribute distinct monomials, hence they cannot cancel each other.

From here everything follows our old recipe. The only difficulty is the evaluation of $\operatorname{pf} A$ at a chosen point. Thankfully there is a clean connection between $\operatorname{pf} A$ and $\det A$, so the evaluation can be delegated to Gaussian elimination.

To establish the connection, we (finally!) recall the definition of $\operatorname{sign}(\pi)$.

Definition 5.1. The sign of a permutation π is $(-1)^b$, where $b \pmod 2$ is determined in two equivalent ways:

(1) the number of crossings in the bipartite interpretation of π;

(2) the number of even cycles in the cyclic interpretation of π.

We leave the verification to the reader.

Lemma 5.2. Suppose we obtain π' by swapping two adjacent slots in permutation π. Then $\operatorname{sign}(\pi') = -\operatorname{sign}(\pi)$.

Proof. In the bipartite interpretation, rearranging two neighbouring nodes on the left shall change the number of crossings by exactly one. □

Lemma 5.3. Let A be any screw-symmetric matrix (including but not limiting to the one that we defined). Then all $2^n n!$ permutations $\pi \in S_{2n}$ that represent the same perfect matching M must contribute equally in pf A. So we may define

$$\varphi(M) := \operatorname{sign}(\pi) \prod_{i=1}^{n} A_{\pi(2i-1),\pi(2i)}$$

without ambiguity, and write

$$\operatorname{pf} A = \sum_M \varphi(M).$$

Proof. Take $\pi \in S_{2n}$. Observe that its contribution $\operatorname{sign}(\pi) \prod_{i=1}^{n} A_{\pi(2i-1),\pi(2i)}$ is invariant under (i) exchange of two adjacent boxes or (ii) swap of elements inside a box. For (i), $\operatorname{sign}(\pi)$ does not change by applying Lemma 5.2 four times; nor does the product. For (ii), $\operatorname{sign}(\pi)$ changes by Lemma 5.2, but some $A_{\pi(2i-1),\pi(2i)}$ turns into $A_{\pi(2i),\pi(2i-1)}$ and flips back the sign due to skew-symmetry.

Since all permutations $\pi \in S_{2n}$ that represent M can be converted into one another by a sequence of (i) and (ii), they must contribute equally to pf A. □

Theorem 5.4. (Cayley) For any screw-symmetric matrix A, $\det A = (\operatorname{pf} A)^2$.

Proof. By Lemma 5.3, we may expand

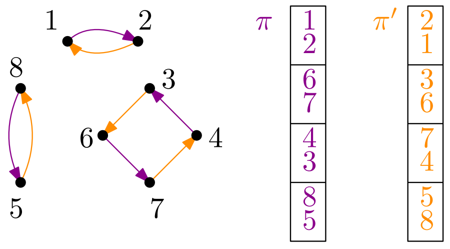
$$(\operatorname{pf} A)^2 = \sum_{M,M'} \varphi(M)\, \varphi(M').$$

Since M and M' are perfect matchings, $M \cup M'$ (where we allow multi-edges) is a cycle cover in which every cycle is $\{M, M'\}$-alternating (and in particular has even length). We orient each cycle by starting from its smallest vertex and go in the direction of the incident M edge. Hence, we have mapped (M, M') to a directed even cycle cover $\mathcal{C}$. Conversely, given a directed even cycle cover $\mathcal{C}$, we can recover the pair (M, M') by an apparent reverse procedure. With this bijection,

$$(\operatorname{pf} A)^2 = \sum_{\mathcal{C}} \operatorname{sign}(\mathcal{C}) \prod_{e \in \mathcal{C}} A_e$$

where $\operatorname{sign}(\mathcal{C}) \in \{-1, +1\}$ is defined so that $\operatorname{sign}(\mathcal{C}) \prod_{e \in \mathcal{C}} A_e = \varphi(M)\, \varphi(M')$.

What exactly is $\operatorname{sign}(\mathcal{C})$? Recall the definitions of $\varphi(M)$ and $\varphi(M')$ allow us to use any convenient π, π' that represent M, M'. For our purpose, let us traverse $\mathcal{C}$ and fill π and π' top-down. The picture below gives an example where we traverse in order $1 \to 2 \to 1$, $3 \to 6 \to 7 \to 4 \to 3$ and then $5 \to 8 \to 5$.



This way, the products $\prod_{e\in\mathcal{C}} A_e$ and $(\prod_{i=1}^{n} A_{\pi(2i-1),\pi(2i)})\,(\prod_{i=1}^{n} A_{\pi'(2i-1),\pi'(2i)})$ will perfectly agree, hence $\operatorname{sign}(\mathcal{C}) = \operatorname{sign}(\pi)\operatorname{sign}(\pi')$. Denote by m the number of (even) cycles in $\mathcal{C}$. By repeated application of Lemma 5.2 (or just by comparing the number of crossings in the bipartite representations), we see $\operatorname{sign}(\pi') = \operatorname{sign}(\pi)\,(-1)^m$. Therefore $\operatorname{sign}(\mathcal{C}) = (-1)^m$. So if we reinterpret $\mathcal{C}$ as a permutation $\sigma \in S_{2n}$, then $\operatorname{sign}(\sigma) = \operatorname{sign}(\mathcal{C})$ coincide, and we may write

$$(\operatorname{pf} A)^2 = \sum_{\sigma\in S_{2n} \text{ without odd cycle}} \operatorname{sign}(\sigma) \prod_{i=1}^{2n} A_{i,\sigma(i)}.$$

It is essentially $\det A$, excluding those $\sigma \in S_{2n}$ with odd cycles. Fortunately, for any such $\sigma \in S_{2n}$, reversing an odd cycle that contains the smallest vertex gives us another such $\sigma' \in S_{2n}$ where $\operatorname{sign}(\sigma') = \operatorname{sign}(\sigma)$ by definition, and $\prod_{i=1}^{2n} A_{i,\sigma(i)} = -\prod_{i=1}^{2n} A_{i,\sigma'(i)}$ by skew-symmetry. The contributions of σ and σ' hence cancel each other, so we could safely ignore them altogether. Therefore $(\operatorname{pf} A)^2 = \det A$ indeed. $\square$

In our context, we stress that the coefficient of y^k in $\det A$ does *not* track the k-PMs. It is rather a mixture of all: contributions come from pairs of d-PMs and $(k-d)$-PMs, for every d. To correctly track the k-PMs, the algorithm should instead compute the coefficient $c_k(\boldsymbol{x})$ of y^k in $\operatorname{pf} A = \pm\sqrt{\det A}$.

5.6 Derandomisation for Planar Graphs

To conclude the chapter, we present a derandomisation of the approach when G is planar. Remember the discussion in Section 5.3: Our goal is to instantiate $\boldsymbol{x} = (x_e)_{e\in E}$ concretely so that the polynomial $c_k(\boldsymbol{x})$ does not evaluate to nil, as long as k-PM exists.

In a classical result, Kasteleyn [15] showed how to instantiate each x_e by either -1 or $+1$, so that the $\varphi(M)$'s have identical sign across all perfect matchings M. This together with Lemma 5.3 would imply that the evaluation $c_d(\boldsymbol{x})$ *counts* d-PMs for every d—far stronger than our goal!

We start by inspecting the proof of Theorem 5.4. The following are equivalent:

- $\varphi(M)$'s have identical sign across all perfect matchings $M \subseteq E$;
- $\varphi(M)\,\varphi(M')$ is positive for all perfect matchings $M, M' \subseteq E$;
- $\operatorname{sign}(\mathcal{C}) \prod_{e\in\mathcal{C}} A_e$ is positive for all directed even cycle cover $\mathcal{C}$ of G.
- for every even cycle $C = (v_1, \ldots, v_\ell)$ in G such that the graph $G[V \setminus C]$ admits an even cycle cover, the product $\prod_{i=1}^{\ell} A_{v_i,v_{i+1}}$ is negative.

Theorem 5.5. (Kasteleyn) Suppose G is planar. Then we can compute in linear time a $\{-1,+1\}$-assignment to each x_e that satisfies the last condition.

We will present a new proof of this result. As background knowledge, every maximal plane graph can be constructed “inside out” by inserting one vertex u in the outer face at a time. This is referred to a *canonical ordering* and can be computed in linear time [3].

Proof. Because the last condition is closed under edge removal, we may consider *maximal* planar graphs only. Draw G in a canonical ordering. When we insert vertex u, its neighbours $u_1, \ldots, u_t$ must form a path on the outer cycle. We orient the first edge uu_1 arbitrarily. Then for each $i = 2, \ldots, t$, we orient uu_i so that the face $uu_{i-1}u_i$ has an odd number of clockwise edges. Upon finish, all edges are oriented so that every face has an odd number of clockwise edges.

We claim that the property not only holds for faces: In fact, every cycle that encloses an even number of vertices has an odd number of clockwise edges.

Suppose C has length ℓ and encloses $\nu \in 2\mathbb{N}$ vertices. Then by easy application of Euler's formula, the interior of C contains exactly $f := \ell + 2\nu - 2$ faces and $m := \ell + 3\nu - 3$ edges. Suppose the faces have $m_1, \dots, m_f$ clockwise edges, respectively. Then C shall have $\sum_{i=1}^{f} m_i - m$ clockwise edges because each interior edge was counted by exactly one of its incident faces. Now that all m_i's are odd, the parity of this number agrees with $f - m = 1 - \nu$, which is odd.

Now we are almost done: For each edge $uv \in E$ that was oriented $u \to v$, we assign $x_{uv} := +1$ and $x_{vu} := -1$. For every even cycle $C = (v_1, \dots, v_\ell)$ such that $G[V \setminus C]$ admits an even cycle cover, C must enclose an even number of vertices by planarity, so it has an odd number of clockwise edges. As the product $\prod_{i=1}^{\ell} A_{v_i, v_{i+1}}$ "traverses" C either clockwise or counterclockwise, it shall encounter an odd number of -1's, giving a negative result. $\square$

The orientation idea extends to $K_{3,3}$-free graphs as well. These graphs can be decomposed into almost independent pieces of planar graphs and K_5. It was pointed out by Little [17] and Vazirani [25] that we may orient the pieces individually and then glue them together to form a Pfaffian orientation. Yuster [26] gave an alternative, more efficient method that employs a different decomposition.

However, when we look at general graphs, the condition "$\varphi(M)$'s have identical sign across all perfect matchings M" is too strong to ask for. The reason is that this would allow us to count perfect matchings, which is known to be #**P**-complete [24]. What we need is a refined version such as "the majority of φ's have the same sign across all d-PMs", but the realisation of such condition remains widely open.

6

Polytope Descriptions

A perfect matching $M \subseteq E$ can be encoded as a binary vector $\boldsymbol{x} \in \{0,1\}^E$ where

$$x_e := \begin{cases} 1 & e \in M \\ 0 & e \notin M \end{cases}$$

and $\sum_{e \ni v} x_e = 1$ for all $v \in V$. Conversely, any such vector can be interpreted as a perfect matching. Henceforth we will not differentiate the two representations.

These (exponentially many) vectors are not organised systematically. It is desirable to deal with a structural and continuous object, namely their convex hull, which is a polytope in $\mathbb{R}^E$. Edmonds [4] found a description of the polytope in terms of linear inequalities. This yields many combinatorial and algorithmic consequences, framed in the theory of linear programming.

Given such success, we are tempted to formulate Exact Perfect Matching in polytope language, too. We reveal the disappointing fact that the "exact k" constraint shatters every nice property that we know for the perfect matching polytope.

6.1 Polytopes and Linear Programs

Let us review some basic material from polytope theory; see [27] for a thorough treatment. A *polytope* is the convex hull of a finite point set $P \subseteq \mathbb{R}^d$, formally

$$\operatorname{conv}(P) := \left\{ \sum_{i=1}^{t} \lambda_i \boldsymbol{p}_i \ : \ t \in \mathbb{N},\ \boldsymbol{p}_1, \ldots, \boldsymbol{p}_t \in P,\ \lambda_1, \ldots, \lambda_t \geqslant 0 \text{ and } \sum_{i=1}^{t} \lambda_i = 1 \right\}.$$

A *hyperplane* h in $\mathbb{R}^d$ contains all points $\boldsymbol{x} \in \mathbb{R}^d$ satisfying some linear equation $\boldsymbol{a}^{\mathrm{T}} \boldsymbol{x} = b$. Geometrically it divides $\mathbb{R}^d$ into two halfspaces. For a point set $X \subseteq \mathbb{R}^d$ we denote $X \leqslant h$ if $\boldsymbol{a}^{\mathrm{T}} \boldsymbol{x} \leqslant b$ for all $\boldsymbol{x} \in X$, namely it lies entirely in the negative halfspace. The notation $X < h$ is defined likewise.

If $\mathcal{P} \leqslant h$ then we call $\mathcal{P} \cap h$ a *face* of $\mathcal{P}$ and say it is *supported* by h. The dimension of a face is the dimension of the smallest affine space containing it. Faces of dimension 0 (a point) are termed *vertices* and denoted $V(\mathcal{P})$. Faces of dimension 1 (a segment) are called *edges* and denoted $E(\mathcal{P})$. It is not hard to show $\mathcal{P} = \operatorname{conv}(V(\mathcal{P}))$; see for example [27].

Lemma 6.1. Let $\mathcal{P} = \operatorname{conv}(P)$ be a polytope and h be a hyperplane. Then $\mathcal{P} \leqslant h$ if and only if $P \leqslant h$. When it does, the supported face is $\mathcal{P} \cap h = \operatorname{conv}(P \cap h)$.

Proof. The "only if" is trivial, and the "if" follows from convexity of halfspaces. For the second part of the lemma, we have $\operatorname{conv}(P \cap h) \subseteq \mathcal{P} \cap h$ by convexity, so it remains to show the reverse inclusion.

Consider an arbitrary point $\boldsymbol{x} \in \mathcal{P} \cap h$. It is generated by a convex combination $\boldsymbol{x} = \sum_{i=1}^{t} \lambda_i \boldsymbol{p}_i$ for some $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_t \in P$. Recall that $P \leqslant h$ means $\boldsymbol{p}_i \leqslant h$ for all i. But all these points must be on h. Otherwise, say $\boldsymbol{p}_j < h$ then we derive $\boldsymbol{x} < h$, contradicting $\boldsymbol{x} \in h$. Therefore, $\boldsymbol{x}$ is a convex combination of points from $P \cap h$. □

A fundamental theorem says that every polytope can be realised as an intersection of halfspaces; vice versa, any bounded finite intersection of halfspaces gives a polytope. See Theorem 1.1 in [27]. This draws connection between polytopes and *linear programs* which concern with a linear system $A\boldsymbol{x} \leqslant \boldsymbol{b}$. Note that each row i in the system, $\boldsymbol{a}_i \boldsymbol{x} \leqslant b_i$, encodes a halfspace. So the feasible points form a polytope—as long as it is bounded.

6.2 The Perfect Matching Polytope

As we discussed in the chapter introduction, we are interested in $\mathcal{P}_G := \operatorname{conv}(P_G)$ where

$$P_G = \{\text{perfect matchings in } G\} = \left\{ \boldsymbol{x} \in \{0,1\}^E \ : \ \sum_{e \ni v} x_e = 1, \ \forall v \in V \right\}.$$

This is called the *perfect matching polytope* of G. It is not hard to see that $V(\mathcal{P}_G) = P_G$, meaning every perfect matching is a polytope vertex and vice versa.

When G is bipartite, its perfect matching polytope can be described by a very concise linear system. The proof employs an alternating cycle argument under the hood.

Theorem 6.2. The perfect matching polytope of a bipartite graph is precisely described by the linear system

$$\begin{aligned} \sum_{e \ni v} x_e &= 1 \quad (\forall v \in V) \\ \boldsymbol{x} &\geqslant \boldsymbol{0} \end{aligned}$$

Proof. Let $\mathcal{P}'_G$ be the polytope defined by the linear system. Clearly $P_G \subseteq \mathcal{P}'_G$ and thus $\mathcal{P}_G \subseteq \mathcal{P}'_G$ by convexity. To show the converse inclusion, it suffices to argue that all vertices of $\mathcal{P}'_G$ are perfect matchings in G.

Suppose to the contrary that some vertex $\boldsymbol{x} \in \mathcal{P}'_G$ is not a perfect matching. Then it contains a fractional entry $0 < x_{uv} < 1$ for some edge uv. By the first constraint in the system, there is an adjacent edge uv' such that $0 < x_{uv'} < 1$. Repeating the argument, these fractional edges propogate until closing a cycle C. Note that C is even since the graph is bipartite, so we can decompose $C =: M_1 \oplus M_2$ for perfect matchings M_1 and M_2. We adjust $\boldsymbol{x}$ by adding ε on each M_1 edge and subtracting ε on each M_2 edge; this gives us $\boldsymbol{x}_\varepsilon := \boldsymbol{x} + \varepsilon \cdot (M_1 - M_2)$. Note that the first constraint is always preserved, and with sufficiently small $|\varepsilon|$ the second constraint is also satisfied. This means $\boldsymbol{x}_\varepsilon \in \mathcal{P}'_G$ for small $|\varepsilon|$. But then $\boldsymbol{x} = \frac{1}{2}(\boldsymbol{x}_\varepsilon + \boldsymbol{x}_{-\varepsilon})$ is a convex combination of two points in $\mathcal{P}'_G$, so it cannot be a vertex. □

For general graphs, this linear system is no longer sufficient. Think about the triangle graph K_3. One can put $1/2$ on every edge to satisfy the system, yet in reality there is no perfect matching at all! Similar cheating is possible as long as the graph contains an odd cycle. In a celebrated result, Edmonds [4] found a complete linear system which we quote without proof. See also [18] for more background.

Theorem 6.3. (Edmonds) The perfect matching polytope of a graph is precisely described by the linear system

$$\begin{array}{rcll} \sum_{e \ni v} x_e & = & 1 & (\forall v \in V) \\ \sum_{e \in E[S]} x_e & = & \frac{|S|-1}{2} & (\forall S \subseteq V : |S| \text{ odd}) \\ \boldsymbol{x} & \geqslant & \boldsymbol{0} & \end{array}$$

6.3 Slicing the Polytope

Next we try to incorporate the "exact k" constraint. We are interested in $\mathcal{Q}_G := \mathrm{conv}(Q_G)$ where

$$Q_G = \{k\text{-PMs in } G\} = \left\{ \boldsymbol{x} \in P_G \;:\; \sum_{e \in R} x_e = k \right\}.$$

One may think of Q_G as the intersection of P_G with the hyperplane $h^* : \sum_{e \in R} x_e = k$. Is $\mathcal{Q}_G = \mathcal{P}_G \cap h^*$ as well? If this were true then we would automatically obtain a linear system for $\mathcal{Q}_G$ from Theorem 6.2 or 6.3.

Unfortunately the relation is very much false, even when the graph is bipartite! A crude explanation is that our proofs of Theorems 6.2 and 6.3 do not work any more, as applying an alternating cycle does not preserve the number of reds.

To gain a foothold, let us characterise the vertices of $\mathcal{P}_G \cap h^*$. Our first lemma below is geometric and does not use specific knowledge of the polytope. It says that "slicing" a polytope with a hyperplane shall turn its edges into vertices.

Lemma 6.4. Let $\mathcal{P} = \mathrm{conv}(P)$ be a polytope and h be a hyperplane. Then $V(\mathcal{P} \cap h) = (P \cap h) \cup \{f \cap h \;:\; f \in E(\mathcal{P}) \text{ and } f \cap h \text{ is a point}\}$.

Proof. To show the $\subseteq$ inclusion, assume that $\mathcal{P}$ is described by a linear system $A\boldsymbol{x} \leqslant \boldsymbol{b}$. Observe that every vertex $\boldsymbol{x}$ of $\mathcal{P} \cap h$ must lie on some face of $\mathcal{P}$. Suppose not, then we have $A\boldsymbol{x} < \boldsymbol{b}$, so $\mathcal{P}$ contains a sufficiently small ball $\mathcal{B}$ around $\boldsymbol{x}$. Thus $\mathcal{P} \cap h$ contains the low dimensional ball $\mathcal{B} \cap h$, which contradicts with $\boldsymbol{x}$ being a vertex.

With the observation, $\boldsymbol{x} = f \cap h$ for some face f of $\mathcal{P}$. But $\dim(f) - 1 \leqslant \dim(f \cap h) \leqslant \dim(f)$, so $\dim(f) \in \{0, 1\}$. Therefore, f is either a vertex or an edge of $\mathcal{P}$.

For the $\supseteq$ inclusion, we distinguish two cases:

- Let $\boldsymbol{x} \in P \cap h$. In particular it is a vertex of $\mathcal{P}$, hence supported by some hyperplane h'. The same h' supports $\boldsymbol{x}$ as a vertex in $\mathcal{P} \cap h$ as well.
- Let $f \in E(\mathcal{P})$ such that $f \cap h$ is a point. Assume that the edge f is supported by h'. Then $(\mathcal{P} \cap h) \cap h' = (\mathcal{P} \cap h') \cap h = f \cap h$, so the latter is a vertex of $\mathcal{P} \cap h$ supported by h'. □

Corollary 6.5. $V(\mathcal{P}_G \cap h^*) = Q_G \cup \{f \cap h^* \;:\; f \in E(\mathcal{P}_G) \text{ and } f \cap h^* \text{ is a point}\}$. □

Our second lemma characterises $E(\mathcal{P}_G)$.

Lemma 6.6. Let M and M' be perfect matchings in G. Then $\mathrm{conv}\{M, M'\}$, the segment connecting M and M', is an edge of $\mathcal{P}_G$ if and only if $M \oplus M'$ contains a single cycle.

Proof. First assume that $M \oplus M'$ contains a single cycle C. Consider the hyperplane $h: \sum_{e \in M \cup M'} x_e = n$. Observe that $\sum_{e \in M \cup M'} x_e \leqslant n$ for all perfect matchings $\boldsymbol{x} \in P_G$, namely $P_G \leqslant h$. To attain the equality, $\boldsymbol{x}$ must pick all edges in $M \cap M'$ and exactly half of the edges from C. Therefore $P_G \cap h = \{M, M'\}$. By Lemma 6.1 we conclude that $\mathcal{P}_G \cap h = \mathrm{conv}\{M, M'\}$ is a 1-dimensional face, that is an edge.

Next assume that $M \oplus M'$ contains at least two cycles C and D. Let us construct two perfect matchings $N := M \oplus C$ and $N' := M' \oplus C$. Observe that M, M', N, N' are distinct. Moreover,

$$\frac{1}{2}M + \frac{1}{2}M' = \frac{1}{2}N + \frac{1}{2}N',$$

meaning $\mathrm{conv}\{M, M'\} \cap \mathrm{conv}\{N, N'\} \neq \emptyset$. So $\mathrm{conv}\{M, M'\}$ cannot be an edge. The last point can be argued more formally: Suppose to contrary that $\mathcal{P}_G \cap h = \mathrm{conv}\{M, M'\}$ for some supporting hyperplane h. Then $N < h$ and $N' < h$, so is $\mathrm{conv}\{N, N'\}$. This implies $\mathrm{conv}\{M, M'\} \cap \mathrm{conv}\{N, N'\} = \emptyset$, a contradiction. □

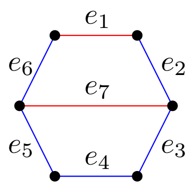
Remark. The lemma reveals the interesting fact that walking along an edge of the polytope $\mathcal{P}_G$ corresponds to applying an alternating cycle. This somehow unifies the difficulties we are facing in both polyhedral and combinatorial approaches.

Using Corollary 6.5 and Lemma 6.6, we can generate all the vertices of $\mathcal{P}_G \cap h^*$ as follows. First, every k-PM is a vertex. Second, for every k_1-PM M_1 and k_2-PM M_2 such that $k_1 < k < k_2$ and $M_1 \oplus M_2$ is a single cycle, their mixture

$$\frac{k_2 - k}{k_2 - k_1} M_1 + \frac{k - k_1}{k_2 - k_1} M_2$$

gives a vertex.

The second case generates plenty of fractional vertices. The figure below draws a concrete example. There are three perfect matchings $(1,0,1,0,1,0,0)$, $(0,1,0,1,0,1,0)$ and $(1,0,0,1,0,0,1)$ with 1, 0 and 2 reds, respectively. For $k = 1$, the sliced polytope $\mathcal{P}_G \cap h^*$ has one integral vertex $(1,0,1,0,1,0,0)$ and one fractional vertex $(1/2, 1/2, 0, 1, 0, 1/2, 1/2)$.



For most graphs—including bipartite graphs—$\mathcal{P}_G \cap h^*$ may contain far more fractional vertices than integral ones. Therefore, the linear system

$$\begin{aligned}
\sum_{e \ni v} x_e &= 1 && (\forall v \in V)\\
\sum_{e \in E[S]} x_e &= \frac{|S| - 1}{2} && (\forall S \subseteq V : |S| \text{ odd})\\
\sum_{e \in R} x_e &= k\\
\boldsymbol{x} &\geqslant \mathbf{0}
\end{aligned}$$

is far from complete to describe $\mathcal{Q}_G$.

6.4 Exponential Extension Complexity

We have had an impression that $\mathcal{Q}_G$ is tricky to describe in terms of linear systems. It was actually formalised in [13] that $\mathcal{Q}_G$ has exponential extension complexity. Here "extension" means a higher-dimensional polytope whose projection onto $\mathbb{R}^E$ equals $\mathcal{Q}_G$. The result says that every possible extension (in particular, $\mathcal{Q}_G$ itself) requires exponentially large linear system to describe.

But we should not be too demoralised. After all, the perfect matching polytope $\mathcal{P}_G$ (for general G) has exponential extension complexity, too [23]. The important thing is that the linear system given by Theorem 6.3 is "good enough" to yield efficient algorithms via *primal-dual methods* [4]. Speaking from high level, we transform the linear program (with exponentially many constraints) to its dual program (with exponentially many variables). The algorithm improves a dual solution gradually, while maintaining that all but polynomially many variables are zero so that we could store them in memory. Eventually we reach an optimal dual solution. Via the duality theorem of linear programming [19], this corresponds to an optimal primal solution.

There is yet another way to circumvent the exponential extension complexity. It utilises an optimisation procedure called the *ellipsoid method*. The idea is pretty simple. Initially we encircle the polytope by a large enough ellipsoid—an affinely transformed ball—centred at $\boldsymbol{p}$. Assume that an oracle can report whether $\boldsymbol{p}$ is in the polytope and, if not, return a hyperplane h through $\boldsymbol{p}$ such that the polytope is strictly on one side. In the former case we are done. In the latter case, h divides the ellipsoid in two halves. We shrink the ellipsoid (in volume) to encircle the half that contains the polytope. Repeating like this, the ellipsoid eventually becomes so small that its centre $\boldsymbol{p}$ must be inside the polytope, *unless* the polytope is empty. The efficiency of the ellipsoid method solely depends on the assumed oracle. For the perfect matching polytope $\mathcal{P}_G$, it is possible to design an efficient oracle based on Theorem 6.3 and other ideas; see Lovász and Plummer [18].

In that regard, deriving a good (though exponentially large) family of inequalities for $\mathcal{Q}_G$ is an important and educational step towards an algorithm. What does "good" mean? This is just a vague term, but let us give an example that surely does not qualify: For each perfect matching M of $i < k$ reds, we explicitly add a constraint $\sum_{e\in M} x_e \leqslant n-1$.

It is somewhat disappointing that we do not even know a partial family of inequalities that can exlude *fractional vertices* in $\mathcal{P}_G \cap h^*$ from $\mathcal{Q}_G$. So in the next section we go on a different route, which turns out to reveal information about $\mathcal{Q}_G$ indirectly.

6.5 Polytope for Extendable Sets

Let us switch to the Max Weight Extendable k-Set problem (see Section 2.4) and model it in polytope language. This time we are interested in $\mathcal{S}_G := \operatorname{conv}(S_G)$ where

$$\begin{aligned} S_G &:= \left\{(F, M) \;:\; F \in \binom{M}{k} \text{ and } M \text{ is a perfect matching in } G\right\} \\ &= \left\{(\boldsymbol{y}, \boldsymbol{x}) \in \{0,1\}^E \times P_G \;:\; \boldsymbol{y} \leqslant \boldsymbol{x} \text{ and } \sum_{e\in E} y_e = k\right\}. \end{aligned}$$

The goal is to maximise the linear objective $w(\boldsymbol{y}) := \sum_{e\in E} w(e)\, y_e$ over S_G, or equivalently over the polytope $\mathcal{S}_G$. Let w^*_{int} be the maximum objective value.

Throughout we assume that the underlying graph G is bipartite. Our starting point of study is a natural linear system satisfied by $\mathcal{S}_G$:

$$\begin{aligned} \sum_{e\ni v} x_e &= 1 \quad (\forall v\in V) \\ \sum_{e\in E} y_e &= k \\ \mathbf{0} \leqslant \boldsymbol{y} &\leqslant \boldsymbol{x} \end{aligned}$$

Definition 6.7. We call $(\boldsymbol{y},\boldsymbol{x})$ a *solution* if it satisfies the system above. It is an *optimum* if it maximises $w(\boldsymbol{y})$ among all solutions; denote its objective value as w^*.

Note that a solution might well be outside $\mathcal{S}_G$, thus $w^* \geqslant w^*_{\text{int}}$. But it turns out that $w^* \leqslant 2\, w^*_{\text{int}}$. That is, a solution can take advantage of no more than a factor of two! In the literature, the exact ratio is called the *integrality gap* of the system.

We will do some preparations before presenting a proof. For $F\subseteq E$, let $w_i(F)$ be the weight of its i^{th} heaviest edge, and let $W_k(F) := \sum_{i=1}^k w_i(F)$.

Lemma 6.8.

- $W_a(F) \leqslant W_b(F) \leqslant \frac{b}{a} W_a(F)$ for all $F\subseteq E$ and $1\leqslant a\leqslant b$.
- $W_b(F\uplus F') = \max_{0\leqslant a\leqslant b} [W_a(F) + W_{b-a}(F')]$.

Proof. For $a\leqslant b$, we have $W_b(F) = W_a(F) + \sum_{i=a+1}^b w_i(F) \geqslant W_a(F)$, hence the first inequality. The average weight of the top b edges is at most the average weight of the top a edges, that is $\frac{W_b(F)}{b} \leqslant \frac{W_a(F)}{a}$, hence the second inequality. To see the final claim, observe that $W_b(F\uplus F') \geqslant W_a(F) + W_{b-a}(F')$ for all $0\leqslant a\leqslant b$ since the right hand side accumulates exactly b edges in $F\uplus F'$. On the other hand, the top b edges in $F\uplus F'$ must be contributed by the top edges in F and F', so the equality is attained when a is the number contributed by F. □

Lemma 6.9. With $\boldsymbol{x}$ fixed, a solution $(\boldsymbol{y},\boldsymbol{x})$ is optimum iff $\boldsymbol{y}$ prioritises filling heavier edges e up to their upper limits x_e. □

Definition 6.10. A solution $(\boldsymbol{y},\boldsymbol{x})$ is *confined* if there is a cycle C such that $x_e\in\{0,1\}$ for all $e\notin C$, and $x_e + x_{e'} = 1$ for all consecutive edges $e, e'\in C$.

Lemma 6.11. There exists a confined optimum.

Proof. Take an arbitrary optimum $(\boldsymbol{y},\boldsymbol{x})$ and suppose it is not confined. So there exists a fractional edge e (namely $0 < x_e < 1$) and such imperfection must propagate along some path until enclosing an even cycle C. There is also a fractional edge outside C, and by a similar reasoning we can enclose another fractional even cycle $C' \neq C$. Note that C and C' may intersect. We will slightly perturb the solution on each of the two cycles.

Let $\varepsilon\in\mathbb{R}$ be a small number. Decompose $C =: F_0 \cup F_1$ as a union of two matchings. We add an ε to every $x_e : e\in F_0$ and subtract an ε from every $x_e : e\in F_1$. Moreover, each y_e moves with x_e if they were equal initially; and stays unchanged otherwise. Clearly the first and the third constraints in the system are preserved for small $|\varepsilon|$. However, $\sum_{e\in E} y_e$ might change linearly in ε, say by $\gamma\varepsilon$ and thus breaks the second constraint.

The idea is to use the other cycle C' to balance the effect. We apply an ε' modification on C' similarly. Suppose that $\sum_{e\in E} y_e$ changes by $\gamma'\varepsilon'$. Choose $\varepsilon, \varepsilon'$ so that $\gamma\varepsilon + \gamma'\varepsilon' = 0$, which ensures that the second constraint holds. Note that at least one of $\varepsilon, \varepsilon'$ remains a free variable, say ε. For different signs of ε, the objective value $w(\boldsymbol{y})$ shall move in different directions (or stay the same). However, by the optimality of $(\boldsymbol{y}, \boldsymbol{x})$ it has to stay the same for all ε. Therefore, by picking ε appropriately we can either eliminate a fractional entry in $\boldsymbol{x}$, or extend the region where $\boldsymbol{x}$ and $\boldsymbol{y}$ agree. So we are guaranteed to get a confined solution by repeating the argument. □

Theorem 6.12. $w^* \leqslant 2\, w^*_{\text{int}}$.

Proof. Take an optimum $(\boldsymbol{y}, \boldsymbol{x})$ confined in C via Lemma 6.11. Decompose C into a union of two matchings $F_0 \cup F_1$. Let $F_2 := \{e \notin C : x_e = 1\}$ be the matching outside C. Note that

$$x_e = \begin{cases} \lambda & e \in F_0 \\ 1-\lambda & e \in F_1 \\ 1 & e \in F_2 \\ 0 & \text{otherwise} \end{cases}$$

for some constant $\lambda \in [0,1]$.

For $i = 0, 1, 2$ we denote $k_i := \sum_{e\in F_i} y_e$, thus $k_0 + k_1 + k_2 = k$. By Lemma 6.9, the $\boldsymbol{y}$ must fill heavier edges up to their limits, hence

$$\begin{aligned} w^* &= \sum_{e\in F_0} w(e)\, y_e + \sum_{e\in F_1} w(e)\, y_e + \sum_{e\in F_2} w(e)\, y_e \\ &= \lambda \cdot W_{k_0/\lambda}(F_0) + (1-\lambda) \cdot W_{k_1/(1-\lambda)}(F_1) + W_{k_2}(F_2) \\ &\leqslant W_{k_0}(F_0) + W_{k_1}(F_1) + W_{k_2}(F_2) \end{aligned}$$

where the last line follows from Lemma 6.8. Without loss of generality we assume $W_{k_0}(F_0) \leqslant W_{k_1}(F_1)$, so

$$w^* \leqslant 2\, W_{k_1}(F_1) + W_{k_2}(F_2).$$

Consider the perfect matching $F_1 \uplus F_2$. Using Lemma 6.8, we see

$$\begin{aligned} w^*_{\text{int}} &\geqslant W_k(F_1 \uplus F_2) \\ &\geqslant W_{k_1}(F_1) + W_{k_0+k_2}(F_2) \\ &\geqslant W_{k_1}(F_1) + W_{k_2}(F_2). \end{aligned}$$

Therefore $w^* \leqslant 2\, w^*_{\text{int}}$ as desired. □

Getting back to the study of polytope $\mathcal{S}_G$, our next result establishes a close connection between $\mathcal{S}_G$ and $\mathcal{Q}_G$.

Theorem 6.13. Let G' be obtained from G by reduction (3)⇒(1) in Section 2.4. Then some face of $\mathcal{S}_{G'}$ is an extension of $\mathcal{Q}_G$.

Proof. Let w be the weight function produced by the reduction. Recall that every i-PM M in G maps to a pair $(F', M') \in \mathcal{S}_{G'}$ such that

$$w(F') = \begin{cases} 3\,r + i & i \leqslant k, \\ 3\,r - i + 2\,k & i > k. \end{cases}$$

Consider the hyperplane $h: \sum_{e\in E} w(e) = 3r + k$. Note that $S_{G'} \leqslant h$, and (F', M') is on h iff the preimage M is a k-PM. It remains to realise that

- M' is a projection of (F', M') onto the second half of coordinates; and
- M is a projection of M' onto the edges that appear in G.

So $Q_G = \mathrm{proj}(S_{G'} \cap h)$ and, consequently,

$$\mathcal{Q}_G = \mathrm{conv}(Q_G) = \mathrm{proj}(\mathrm{conv}(S_{G'} \cap h)) = \mathrm{proj}(\mathcal{S}_{G'} \cap h)$$

is the projection of the face $\mathcal{S}_{G'} \cap h$. □

Corollary 6.14. $\mathcal{S}_G$ has exponential extension complexity. □

So our best bet is finding a good though exponential family of linear inequalities for $\mathcal{S}_G$. We managed to find a partial family. For every cycle C of length $2\ell > 2k$, we decompose it as a union of two matchings $F \cup F'$ and write ∂C for the edges with exactly one end in C. We constrain

$$\ell\left(\sum_{e\in F} y_e + \sum_{e\in\partial C} y_e\right) \leqslant k\left(\sum_{e\in F} x_e + (\ell-1)\sum_{e\in\partial C} x_e\right).$$

Note that for all vertices $(\boldsymbol{y}, \boldsymbol{x}) \in S_G$:

- If $\boldsymbol{x} \cap F = \emptyset$ then $\sum_{e\in F} y_e = \sum_{e\in F} x_e = 0$, so the inequality holds.
- If $\boldsymbol{x} \cap F \neq \emptyset$ then we distinguish two cases. If $\boldsymbol{x} \cap C = F$ then $\sum_{e\in F} x_e = \ell$; otherwise $\sum_{e\in\partial C} x_e \geqslant 1$. In either case the right hand side is at least $k\ell$, which is larger than the left hand side.

The rationale is to prevent the $\boldsymbol{y}$ "mixing" F and F', which is the main contributor of the factor 2 in Theorem 6.12. Our constraint is capable to rule out some, but not all, fractional vertices. It is an interesting problem to find a complete linear system that describes $\mathcal{S}_G$.

7

Exact Perfect f-Matching

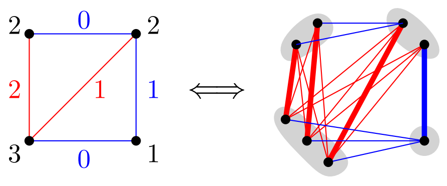
Here is a direct generalisation of perfect matchings. Given graph $G=(V,E)$ and function $f:V\to\mathbb{N}$, a *perfect f-matching* is an assignment $x:E\to\mathbb{N}$ such that $\sum_{u\in N(v)} x_{uv}=f(v)$ for all $v\in V$. One may understand x_{uv} as the times that we pick the edge uv. Note that the notion degenerates to perfect matchings when $f\equiv 1$. It is known that we can tell the existence of perfect f-matching in strongly polynomial time [1]; that is, the number of arithmetic operations depends on n but not on the amplitude of f.

With some edges being red and the other being blue, the Exact Perfect f-Matching problem asks if there is a perfect f-matching that uses exactly k reds (i.e. $\sum_{e\in R} x_e=k$). When f is polynomially bounded we show its equivalence to Exact Perfect Matching. In this case, we also provide an efficient dynamic program if G has small tree-width. However, little is known when f takes exponential values, which marks an intriguing complexity status of the problem.

7.1 The Complexity Status

Theorem 7.1. Exact Perfect f-Matching reduces to Exact Perfect Matching when the f-values are polynomially bounded in n.

Proof. Given $G=(V,R\uplus B)$ and f, we explode each vertex v into a set S_v of $f(v)$ copies; and for each edge uv, we add full connection between S_u and S_v using the colour of uv. The reader might have noticed that the resulting G' is a blow-up of G; see Definition 4.7.



Any perfect matching M' in G' naturally induces an assignment x on the edges of G:

$$x_{uv}:=\text{number of } M'\text{-edges between } S_u \text{ and } S_v.$$

Since every copy in S_v is matched by M' to some copy in S_u, $u\in N(v)$, we have $\sum_{u\in N(v)} x_{uv}=|S_v|=f(v)$. Therefore, x is a perfect f-matching in G; moreover, it uses the same number of reds as M'.

Conversely, any perfect f-matching x in G maps to a perfect matching in G'. To see this, recall $|S_v| = f(v) = \sum_{u\in N(v)} x_{uv}$. So for each $v \in V$ we can partition $S_v =: \biguplus_{u\in N(v)} S_v^u$ where $|S_v^u| := x_{uv}$. Then we perfectly and arbitrarily match the copies in S_v^u with the copies in S_u^v. This is feasible because S_v^u and S_u^v have identical size and are fully connected. The result is a perfect matching in G' that uses the same number of reds as f. □

Interestingly, if we allow exponential f-values then it is unclear whether a *randomised* polynomial-time algorithm exists. For one thing, determinants and Pfaffians enumerate permutations, which do not incorporate multiplicities. For another, although the isolation lemma extends to multisets, the weight of a multiset can become exponential and thus hinders algorithmic usage. Is the problem efficiently solvable? Or is it **NP**-complete? We leave them as open questions to the reader.

7.2 Tree Decompositions

Solving Exact Perfect f-Matching on trees is easy: The quota of the leaves must fully pass to their parents, and the leftover of parents must fully pass to the grandparents, and so on until we reach the root.

How about tree-like graphs? The solution becomes less trivial, but we will present a standard dynamic program that does the trick. When specialised to $f \equiv 1$ we recover a dynamic program for Exact Perfect Matching. A catch is that its running time depends exponentially on "tree-likeness", so for dense graphs it is prohibitively slow.

How do we quantify the closeness of a graph to a tree? A gauge can be derived from a notion called tree decompositions. It was popularised by Robertson and Seymour in their renowned work on graph minors [22].

Definition 7.2. A *tree decomposition* of graph $G = (V, E)$ is a rooted tree T where every node $a \in T$ claims a territory $V_a \subseteq V$. We require two properties:

(i) Every $\{u, v\} \in E$ lies in the territory of some node $a \in T$, meaning that $\{u, v\} \subseteq V_a$.
(ii) $V_b \cap V_c \subseteq V_a$ for all nodes $a \in T$ on the path between $b, c \in T$.

The second property says that if a vertex is in the territories of both b and c, then it should also be found in all territories "between" b and c. In other words, $\{a \in T : v \in V_a\}$ forms a subtree for every $v \in V$.

A trivial tree decomposition is to create a single node a that encompasses everything; namely $V_a = V$. But what if we limit the sizes of territories? When G is a tree then we may do the following:

- for each vertex $v \in V$ create a node a with $V_a := \{v\}$;
- for each edge $\{u, v\} \in E$ create a node a with $V_a := \{u, v\}$, and connect it to the nodes corresponding to u and v.

This clearly defines a tree decomposition where territories have sizes at most 2. When G contains a cycle, such construction fails and, in fact, territories of larger sizes are necessary. It motivates us to call $\max_{a\in T} |V_a| - 1$ the *complexity* of tree decomposition T. The *tree-width* of G is the smallest possible complexity over all tree decompositions of G.

We will not explain how one can determine the tree-width or even an optimal tree decomposition. It suffices to know that if G has tree-width t, then we can compute in $2^{O(t)} n$ time a tree decomposition T of G with $O(n)$ nodes and complexity at most $2t + 1$ [16].

Henceforth we will focus on how T helps solving Exact Perfect f-Matching. Let T_a be the subtree of T rooted at a. It covers territories $Z_a := \bigcup_{a'\in T_a} V_{a'}$.

Lemma 7.3. If node $a \in T$ has child $b \in T$, then $V_a \setminus V_b \subseteq Z_a \setminus Z_b$. In words, the vertices forsaken by b will never show up again in the territory of subtree T_b.

Proof. Let $v \in V_a \setminus V_b$. Obviously $v \in V_a \subseteq Z_a$, so it remains to argue $v \notin Z_b$. Suppose otherwise that $v \in V_c$ for some $c \in T_b$. By property (ii) of tree decomposition, we have $v \in V_b$ as well because b is on the path between a and c. This gives a contradiction. □

Lemma 7.4. If node $a \in T$ has children $b, c \in T$, then there is no edge between $Z_b \setminus V_a$ and $Z_c \setminus V_a$.

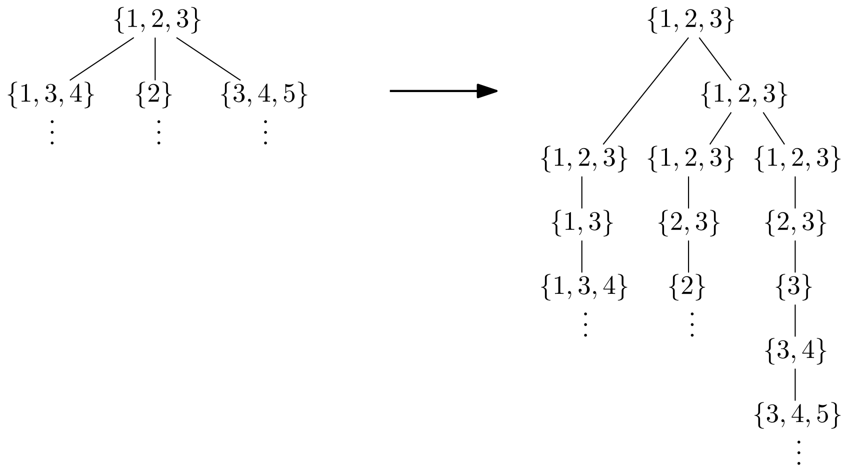
Proof. Suppose there is an edge $\{u, v\}$ between $u \in Z_b \setminus V_a$ and $v \in Z_c \setminus V_a$. So by property (i) we may find a node $d \in T$ with $\{u, v\} \subseteq V_d$. Observe that node a is between nodes b, d or nodes c, d (or both). The former case implies $u \in V_a$ by property (ii), and the latter case implies $v \in V_a$. Both reach a contradiction. □

The lemma essentially means that any two children of a can only interact within V_a. Hence, if we freeze the state inside V_a, the problem can be solved independently on the children.

To make a streamlined algorithm, though, we need to regulate the form of the tree. We require each node $a \in T$ to take one of the following four types:

leaf:	a has no child and $\lvert V_a \rvert = 1$;
introduce:	a has one child b and $V_b = V_a \cup \{v\}$ for some $v \notin V_a$;
remove:	a has one child b and $V_b = V_a \setminus \{v\}$ for some $v \in V_a$;
split:	a has two children b and c and $V_b = V_c = V_a$.

It is rather easy to transform an arbitrary tree decomposition to one that fits such form via telescoping:



Using the same idea, we may also assume that the root ρ satisfies $\lvert V_\rho \rvert = 1$. Note that the complexity of the tree decomposition is preserved, and the number of nodes grows by a factor of $2t$ at most. Furthermore, the transformation runs in $O(t^2 n)$ time.

7.3 A Dynamic Program

Now we are ready to describe the dynamic program for Exact Perfect f-Matching. Consider the following problem:

Input: node $a \in T$, function $g: V_a \to \mathbb{N}$, and integer $r \in \mathbb{N}$.
Output: a boolean value $\mathrm{out}(a, g, r)$, indicating whether $G[Z_a] - E[V_a]$ admits a perfect $\tilde{f}$-matching that uses r reds where

$$\tilde{f}(v) := \begin{cases} f(v) & v \in Z_a \backslash V_a, \\ g(v) & v \in V_a. \end{cases}$$

We have the following recursive relations.

- Suppose a is a leaf. Then $\mathrm{out}(a, g, r) = (g \equiv 0) \wedge (r = 0)$.
- Suppose a is an "introduce" node whose child b satisfies $V_b = V_a \cup \{v\}$. Our goal is to reduce to subproblems on $(b, *, *)$. Since $Z_b = Z_a$, the graph $G[Z_a] - E[V_a]$ is simply $G[Z_b] - E[V_b]$ plus the edges between v and V_a. So to bridge the gap it suffices to specify how we pick each of these edges. To this end, we iterate over all functions $h: V_a \cap N(v) \to \mathbb{N}$ such that $h \leqslant g$ and $\sum_{u \in V_a \cap N(v)} h(u) \leqslant f(v)$. It encodes the number of times that v matches to its neighbours in V_a. Define the residual $g': V_b \to \mathbb{N}$ by

$$g'(u) := \begin{cases} g(u) & u \in V_a \backslash N(v), \\ g(u) - h(u) & u \in V_a \cap N(v), \\ f(v) - \sum_{u \in V_a \cap N(v)} h(u) & u = v, \end{cases}$$

  and let $r' := r - \sum_{u \in V_a \cap N(v): uv \in R} h(u)$. Compute $\mathrm{out}(b, g', r')$ and take "$\vee$" over all these outputs.
- Suppose a is a "remove" node whose child b satisfies $V_b = V_a \backslash \{v\}$. Note that $Z_b = Z_a \backslash \{v\}$ by Lemma 7.3, so v does not have neighbours outside V_a in graph $G[Z_a]$. It is not hard to see that $\mathrm{out}(a, g, r) = [g(v) = 0] \wedge \mathrm{out}(b, g|V_b, r)$.
- Finally, suppose a is a "split" node whose children b, c satisfy $V_b = V_c = V_a$. Due to Lemma 7.4, we have $\mathrm{out}(a, g, r) = \bigvee_{r' + r'' = r} [\mathrm{out}(b, g, r') \wedge \mathrm{out}(c, g, r'')]$.

The dynamic program can be implemented in a bottom-up manner, filling the leaf entries at first and the root entry at last. Recall that the root ρ satisfies $|V_\rho| = 1$ and $Z_\rho = V$. Then $\mathrm{out}(\rho, f|V_\rho, k)$ exactly recovers the answer to Exact Perfect f-Matching.

7.4 Time Analysis

Observe that each entry $\mathrm{out}(a, g, r)$ is accessed twice: when it was filled and when it is queried by parent. So the running time is proportional to the number of entries. We have $O(tn)$ choices for node a, at most $[\max_{v \in V} f(v)]^{2t+1}$ choices for function g, and $k \leqslant n$ choices for integer r. Therefore, the running time is upper bounded by $O(tn^2) \cdot [\max_{v \in V} f(v)]^{2t+1}$.

In the special case of Exact Perfect Matching, we have $f \equiv 1$ and thus the running time is $O(tn) \cdot 2^{2t+1}$. It is interesting to contrast the case where f is exponential in n: the dynamic program is no longer efficient in any sense, yet we are not aware of any faster algorithm.

As a final remark, the algorithm can easily be extended to *find* a perfect f-matching if one exists. It adds essentially no overhead to the running time.

Bibliography

1 Richard P Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

2 Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.

3 Hubert De Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

4 Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

5 Nicolas El Maalouly. Exact matching: algorithms and related problems. *ArXiv preprint arXiv:2203.13899*, 2022.

6 Nicolas El Maalouly and Raphael Steiner. Exact matching in graphs of bounded independence number. *ArXiv preprint arXiv:2202.11988*, 2022.

7 Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf. Exact matching: correct parity and FPT parameterized by independence number. *ArXiv preprint arXiv:2207.09797*, 2022.

8 Nicolas El Maalouly and Lasse Wulf. Exact matching and the top-k perfect matching problem. 2022.

9 Hans-Florian Geerdes and Jácint Szabó. A unified proof for karzanov's exact matching theorem. *Quick proof QP-2011-02, Egerváry Research Group, Budapest*, 2011.

10 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.

11 Sebastian Haslebacher. Exact matching: related problems and characterization on restricted graph classes. Master's thesis, ETH Zürich, 2023.

12 Russell Impagliazzo and Avi Wigderson. P=BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229. 1997.

13 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.

14 Alexander V Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics*, 23(1):8–13, 1987.

15 Pieter Kasteleyn. Graph theory and crystal physics. *Graph theory and theoretical physics*, pages 43–110, 1967.

16 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192. IEEE, 2022.

17 Charles HC Little. An extension of kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics: Proceedings of the Second Australian Conference*, pages 63–72. Springer, 1974.

18 László Lovász and Michael D Plummer. *Matching theory*, volume 121. North-Holland mathematics studies, 1986.

19 Jiří Matoušek and Bernd Gärtner. *Understanding and using linear programming*, volume 1. Springer, 2007.

20 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354. 1987.

21 Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.

22 Neil Robertson and Paul D Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

23 Thomas Rothvoß. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)*, 64(6):1–19, 2017.

24 Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

25 Vijay V Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Information and computation*, 80(2):152–164, 1989.

26 Raphael Yuster. Almost exact matchings. *Algorithmica*, 63(1):39–50, 2012.

27 Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 2012.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

EXACT PERFECT MATCHING

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
WANG	YANHENG

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 18.09.2023	Yanheng Wang

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.