A Polynomial-Time Algorithm for Coloring Perfect Graphs Based on Walk Counting

Amir Ali Ahmadi[*] Pravesh K. Kothari[†] Yukai Tang[*]

July 10, 2026

Abstract

We present a polynomial-time algorithm for optimally coloring perfect graphs that is based entirely on graph-theoretic operations. At its core, the algorithm decides whether a perfect graph contains a clique of a given size by iteratively counting walks in the graph with certain weights assigned to its edges and nonedges. These weights are initialized according to a uniform scheme and then updated in each iteration based on the walk counts from the previous iteration.

1 Introduction

In a simple graph $G(V, E)$, with vertex set V and edge set E, a *stable set* is a set of pairwise non-adjacent vertices and a *clique* is a set of pairwise adjacent vertices. A *coloring* of G is an assignment of colors to the vertices of G such that no two adjacent vertices share the same color. This can also be viewed as partitioning V into stable sets. We denote the size of a maximum stable set of G by $\alpha(G)$, the size of a maximum clique of G by $\omega(G)$, and the minimum number of colors needed to color G by $\chi(G)$.

In this paper, we are concerned with finding a maximum stable set, a maximum clique, and a minimum coloring of a *perfect graph*. A graph G is called *perfect* if for every induced subgraph[1] H of G, the chromatic number $\chi(H)$ is equal to the clique number $\omega(H)$. This is a highly studied family of graphs both in structural graph theory and in combinatorial optimization. One of the most significant results regarding perfect graphs is the strong perfect graph theorem conjectured by Berge [1] and proven many years later by Chudnovsky, Robertson, Seymour, and Thomas [2]. This theorem states that a graph G is perfect if and only if no induced subgraph of G is an odd cycle of length at least five or the complement of one. Among other implications, this theorem has led to a polynomial-time algorithm for recognizing perfect graphs [3].

It is well known that the problems of finding a maximum stable set, a maximum clique, and a minimum coloring of a general graph are all NP-hard [4]. In an influential paper, Grötschel, Lovász, and Schrijver proved that these problems can be solved in polynomial time for perfect graphs [5]. Their approach, however, relies on the ellipsoid method and is not regarded as combinatorial. Designing a combinatorial polynomial-time algorithm for finding a maximum stable set/clique or a minimum coloring of a perfect graph is a well-known open problem in graph theory; see, e.g., an interview with Lovász [6], an interview with Chudnovsky [7], [8, Chapter 9], [9, Section 12], [10, Section 1], or [11, Section 1]. Over the years, such algorithms have been designed for several subsets of perfect graphs, including interval graphs [12], chordal graphs [13], claw-free perfect graphs [14], bull-free perfect graphs [15], perfect graphs that do not have a balanced skew partition [16], and bounded degree perfect graphs with no prism or hole of length four as induced subgraphs [11].

[*]Princeton University, Operations Research and Financial Engineering. Partially supported by the Princeton AI Lab Seed Grant, the Princeton SEAS Innovation Grant, and a Research Gift in Mathematical Optimization. Email: aaa@princeton.edu, yt3846@princeton.edu

[†]Princeton University, Computer Science. Partially supported by the Princeton AI Lab Seed Grant and the Princeton SEAS Innovation Grant. Email: kothari@cs.princeton.edu

[1]A graph H is an *induced subgraph* of a graph G if $V(H) \subseteq V(G)$ and any two vertices of H are adjacent if and only if they are adjacent in G.

In this paper, we present a new polynomial-time algorithm for finding a maximum stable set, a maximum clique, and a minimum coloring in any perfect graph. At its core, our algorithm decides if a perfect graph contains a clique of a given size. This procedure relies solely on counting walks in the graph with certain weights assigned to its edges and nonedges. These weights are initially set in a uniform fashion, and then updated in each iteration using the walk counts from the previous iteration.

Despite the prominence of the open problem on finding a polynomial-time combinatorial algorithm to optimally color perfect graphs, there is surprisingly no universally accepted definition of a combinatorial algorithm [9, 10, 11, 17, 18, 19]. Roughly speaking, a 'combinatorial algorithm' should be an algorithm that can be described as a sequence of operations applied to vertices and edges of graphs. In the absence of a formal definition of the notion of a combinatorial algorithm, it is hard for us to genuinely claim that we have solved the open problem. We believe that walk counting is a basic combinatorial operation and hence our algorithm should be considered combinatorial. We hope that either our work will be accepted by the community as resolving this problem, or that it will lead to a formal definition of the notion of a combinatorial algorithm that excludes our algorithm. Either outcome would constitute meaningful progress on this longstanding question.

1.1 Organization and contributions of the paper

The remainder of this paper is organized as follows. Section 2 details a polynomial-time reduction from the problem of optimally coloring a perfect graph to that of deciding if a perfect graph has a clique of a given size. In Section 3, we prove that one can always give a certificate of nonexistence of a clique of a given size in a perfect graph in terms of an inequality involving the weights of closed walks of a certain length (Theorem 5). In Section 4, we present our walk counting algorithm for deciding if a perfect graph has a clique of a given size. The correctness of this algorithm is established by a Lyapunov function argument (Section 4.3). Finally, in Section 5, we show that our algorithm can be viewed as a combinatorial (approximate) implementation of a multiplicative weights update algorithm for solving a modified version of the semidefinite program associated with the Lovász theta function, or a combinatorial (approximate) implementation of the exponentiated gradient descent algorithm applied to the dual of this semidefinite program.

1.2 Notation

For a graph G, we denote its complement by $\bar{G}$. We denote the set of $n \times n$ real symmetric matrices by $\mathbb{S}^n$. A matrix $A \in \mathbb{S}^n$ is said to be positive semidefinite (denoted by $A \succeq 0$) if all its eigenvalues are nonnegative. We denote the identity matrix by I, the all-ones matrix by J, the maximum eigenvalue of a matrix $A \in \mathbb{S}^n$ by $\lambda_{\max}(A)$, its minimum eigenvalue by $\lambda_{\min}(A)$, and its trace by $\mathrm{Tr}(A)$.

For a vector $x \in \mathbb{R}^n$, we denote its transpose by x^T and its i-th entry by x_i. The i-th standard basis vector in $\mathbb{R}^n$ is denoted by e_i. We denote the probability simplex in $\mathbb{R}^n$ by Δ_n; any element of Δ_n is called a probability vector. We use the notation Y_{ij} to denote the symmetric matrix that has ones in entries (i, j) and (j, i), and zeros elsewhere. We denote the indicator function of a logical condition x by $\mathbf{1}(x)$; i.e., $\mathbf{1}(x) = 1$ if x is true and $\mathbf{1}(x) = 0$ otherwise.

2 Reducing minimum coloring of a perfect graph to clique decision

In this section, we give a polynomial-time combinatorial reduction from the problem of finding a minimum coloring of a perfect graph to that of deciding if the graph has a clique of a given size. While the main part of this reduction appears in slightly different forms in the literature [8, 9, 20], we give a self-contained presentation here with some details filled in for completeness and the benefit of the reader.

Suppose we have an oracle that, given a perfect graph $G(V, E)$ and an integer $k \in \{2, \ldots, |V|\}$, determines whether there is a clique of size k in G. In the analysis below, we will refer to this oracle as the clique decision oracle. The reduction is as follows. We first show how to find a maximum clique in a perfect graph using the clique decision oracle (Algorithm 1).[2] Then, we show how Algorithm 1 can

[2]This procedure works more generally on all graphs if given access to a clique decision oracle for an arbitrary graph.

be used to find a stable set that intersects a given collection of maximum cliques in a perfect graph (Algorithm 2). Finally, we show how Algorithm 1 and Algorithm 2 can be combined to color a perfect graph (Algorithm 3).

2.1 Searching for a maximum clique using the clique decision oracle

In this section, we show how to find a maximum clique in a perfect graph using the clique decision oracle. The procedure is outlined in Algorithm 1.

We begin by computing the clique number $\omega(G)$ by binary search over $k \in \{2, \ldots, |V|\}$ using the clique decision oracle. We then fix an ordering of the vertices of G and maintain a working graph H, initially equal to G. We process the vertices in the chosen order. For every vertex v, we use the clique decision oracle to decide whether the (still perfect) graph $H \setminus \{v\}$ contains a clique of size $\omega(G)$. If the answer is yes, then there exists a maximum clique of H that does not contain v, so v can be deleted safely. If the answer is no, then every clique of size $\omega(G)$ in H contains v, so v must be kept. Since a vertex is deleted only when the clique number remains $\omega(G)$, the current working graph always has clique number $\omega(G)$. At the end of the procedure, every remaining vertex belongs to every maximum clique of the final working graph H. Consequently, H itself is a maximum clique of G.

Algorithm 1: Finding a maximum clique using a clique decision oracle

Input: A perfect graph $G(V, E)$
Output: A maximum clique of G

Compute $\omega(G)$ using binary search and the clique decision oracle;
$H \leftarrow G$;
Fix any ordering $v_1, \ldots, v_{|V|}$ of V;
for $i = 1$ **to** $|V|$ **do**
  if $H \setminus \{v_i\}$ *contains a clique of size* $\omega(G)$ **then**
    $H \leftarrow H \setminus \{v_i\}$;
  end
end
return the vertex set of H

2.2 Finding a stable set that intersects a given set of maximum cliques

In this section, we show how to find a stable set in a perfect graph G that intersects a given collection of maximum cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of G. This is done as given in Algorithm 2.

Algorithm 2: Finding a stable set that intersects every maximum clique in a given set

Input: A perfect graph $G = (V, E)$, a collection of maximum cliques $\mathcal{C} = \{C_1, \ldots, C_m\}$ of G
Output: A stable set S with $S \cap C_i \neq \emptyset$ for all i

foreach $v \in V$ **do**
  $w(v) \leftarrow |\{\, i \mid v \in C_i \}|$;
end
Construct an auxiliary graph $G' = (V', E')$ as follows:
  V': for every vertex $v \in V$, delete v and, if $w(v) > 0$, add $w(v)$ copies $\{v^{(1)}, \ldots, v^{(w(v))}\}$ to V';
  E': for every edge $uv \in E$, make every copy of u in V' adjacent to every copy of v in V';
$S' \leftarrow$ Algorithm 1$(\bar{G'})$;
$S \leftarrow \{\, v \in V \mid \exists j$ such that $v^{(j)} \in S' \,\}$;
return S

The proof of correctness of Algorithm 2 is given in Proposition 3 and relies on the two lemmas below.

Lemma 1. *Let G be a perfect graph. Then there exists a stable set S in G such that S intersects every maximum clique of G.*

Proof. Suppose for the sake of contradiction that no stable set in G intersects every maximum clique. Let $S_1, S_2, \ldots, S_{\chi(G)}$ be the color classes of a minimum coloring of G. Since S_1 is a stable set and, by assumption, does not intersect every maximum clique, there exists a maximum clique K of G such that $K \cap S_1 = \emptyset$. It follows that $K \subseteq G \setminus S_1$, and hence $\omega(G \setminus S_1) \geq \omega(G)$. Meanwhile, the remaining color classes $S_2, \ldots, S_{\chi(G)}$ form a valid coloring of $G \setminus S_1$. Therefore, $\chi(G \setminus S_1) \leq \chi(G) - 1$.

Since G is perfect, every induced subgraph of G is perfect; in particular, $G \setminus S_1$ is perfect. Thus $\chi(G \setminus S_1) = \omega(G \setminus S_1)$. Combining this with the inequalities above yields

$$\omega(G) \leq \omega(G \setminus S_1) = \chi(G \setminus S_1) \leq \chi(G) - 1,$$

which contradicts the perfectness of G. □

Definition 1 (False twin copy)**.** *Let $G = (V, E)$ be a graph, and let $v \in V$. A vertex v' added to G is called a* false twin copy *of v if v' is adjacent to exactly the neighbors of v and is not adjacent to v.*

Lemma 2. *For any perfect graph G, the auxiliary graph G' constructed in Algorithm 2 is still perfect.*

The proof of Lemma 2 follows immediately from Theorem 1 in [21].

Proposition 3. *Let $G = (V, E)$ be a perfect graph, and let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ be a collection of maximum cliques in G. Then Algorithm 2 returns a stable set S in G that intersects every maximum clique in $\mathcal{C}$.*

Proof. We first prove that the output S of Algorithm 2 is a stable set in G. By Lemma 2, the auxiliary graph G' constructed in Algorithm 2 is perfect. Hence, its complement $\bar{G}'$ is also perfect. Therefore, by the guarantee of Algorithm 1, the set S' in line 5 of Algorithm 2 is a maximum clique of $\bar{G}'$, i.e., a maximum stable set of G'. By the construction of the auxiliary graph G', the set S in line 6 of Algorithm 2 has the following equivalent description:

$$S = \{\, v \in V : \text{some false twin copy of } v \text{ belongs to } S' \,\}.$$

We claim that S is a stable set in G. Suppose for the sake of contradiction that $u, v \in S$ are adjacent in G. Since $u, v \in S$, there exist copies $u^{(a)}$ of u and $v^{(b)}$ of v that both belong to S'. However, since u, v are adjacent in G, by the construction of G', $u^{(a)}$ is adjacent to $v^{(b)}$. This contradicts the fact that S' is a stable set in G'.

It remains to show that S intersects every clique in $\mathcal{C}$. Suppose for the sake of contradiction that there exists some $C_j \in \mathcal{C}$ such that $S \cap C_j = \emptyset$. Recall that the weight $w(v)$ in Algorithm 2 is defined as the number of cliques in $\mathcal{C}$ that contain v. Therefore, $\sum_{v \in S} w(v) = \sum_{i=1}^{m} |S \cap C_i|$. Since S is a stable set and each C_i is a clique, S intersects every C_i at at most one vertex. Together with the fact that $S \cap C_j = \emptyset$, it follows that $\sum_{v \in S} w(v) \leq m - 1$. Since S' contains only copies of vertices in S and there are exactly $w(v)$ copies of each v, we have

$$|S'| \leq \sum_{v \in S} w(v) \leq m - 1.$$

From Lemma 1, there exists a stable set T in G that intersects every C_i. We construct a stable set T' in G' by taking all $w(v)$ false twin copies of v for each vertex $v \in T$. Since false twin copies of the same vertex are pairwise nonadjacent and the vertices of T are pairwise nonadjacent in G, T' is a stable set in G'. Moreover,

$$|T'| = \sum_{v \in T} w(v) = m > |S'|.$$

This contradicts the fact that S' is a maximum stable set in G', which shows that S must intersect every clique in $\mathcal{C}$.

□

Algorithm 3: Optimally coloring a perfect graph

Input: A perfect graph $G(V, E)$
Output: A minimum coloring of G
while G *not empty* **do**
    Call Algorithm 1 to get a maximum clique C in G
    Initialize the collection of maximum cliques $\mathcal{C}$ in G: $\mathcal{C} \leftarrow \{C\}$
    while *True* **do**
        Call Algorithm 2 to get a stable set S that intersects every maximum clique in $\mathcal{C}$
        if $\omega(G) > \omega(G \setminus S)$ **then**
            Break
        end
        Call Algorithm 1 to get a maximum clique C' in $G \setminus S$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{C'\}$
    end
    Color S with a new color
    $G \leftarrow G \setminus S$
end

2.3 Reducing coloring a perfect graph to clique decision

Algorithms 1 and 2 together lead to a reduction from coloring a perfect graph to deciding whether the graph contains a clique of a given size. Algorithm 3 implements this reduction.

We now prove the following proposition.

Proposition 4. *For every perfect graph $G(V, E)$, Algorithm 3 gives an optimal coloring of G.*[3] *Moreover, during any execution of the inner loop, the number of iterations is at most $|V|$.*

Proof. We first prove that Algorithm 3 gives an optimal coloring of G. Fix one outer iteration of Algorithm 3, and let H denote the current graph at the start of that outer iteration. Let S be the stable set produced when the inner loop terminates. When the inner loop terminates, $\omega(H \setminus S) < \omega(H)$. We claim that S intersects every maximum clique of H. Indeed, if there is a maximum clique Q of H disjoint from S, then $Q \subseteq H \setminus S$, and hence $\omega(H \setminus S) \geq |Q| = \omega(H)$, contradicting the strict inequality above.

Since S is a stable set, it contains at most one vertex from any clique of H. Together with the fact that S intersects every maximum clique of H, every maximum clique of H loses exactly one vertex when S is deleted. It follows that $\omega(H \setminus S) = \omega(H) - 1$. We also know that $H \setminus S$ is an induced subgraph of the perfect graph H, so it is also a perfect graph. Consequently,

$$\chi(H \setminus S) = \omega(H \setminus S) = \omega(H) - 1 = \chi(H) - 1.$$

Thus each outer iteration removes one stable set and decreases the chromatic number of the remaining perfect graph by exactly one. Starting from G, the algorithm therefore performs exactly $\chi(G)$ outer iterations and uses exactly $\chi(G)$ colors.

It remains to bound the number of iterations of every inner loop. Fix again one outer iteration, and let $H(V_H, E_H)$ be the graph considered during that outer iteration. At inner iteration t, denote by $\mathcal{C}_t = \{C_1, \ldots, C_t\}$ the collection of maximum cliques accumulated so far. Define the affine subspace

$$L^t \coloneqq \{x \in \mathbb{R}^{|V_H|} : x(C_i) = 1 \text{ for all } i = 1, \ldots, t\},$$

where $x(C_i) \coloneqq \sum_{v \in C_i} x_v$. Since L^{t+1} is obtained from L^t by adding the constraint $x(C_{t+1}) = 1$, we have $L^{t+1} \subseteq L^t$. Let S_t be the stable set returned by Algorithm 2 when the current clique collection is $\mathcal{C}_t$. Write $\mathbf{1}_{S_t} \in \mathbb{R}^{|V_H|}$ for the characteristic vector of S_t, with $(\mathbf{1}_{S_t})_v = 1$ for $v \in S_t$ and $(\mathbf{1}_{S_t})_v = 0$ otherwise. By Proposition 3, S_t intersects every clique in $\mathcal{C}_t$. Thus, we have $\mathbf{1}_{S_t} \in L^t$. If the algorithm

[3] It follows that the number of iterations of the outer loop is $\chi(G)$.

proceeds to the next inner iteration, then it has found a new maximum clique C_{t+1} in $H \setminus S_t$. Since S_t and C_{t+1} are disjoint, we have $\mathbf{1}_{S_t} \notin L^{t+1}$. This indicates that $L^{t+1} \subsetneq L^t$. Thus, the dimension of the affine subspace decreases by at least one at each inner iteration. We know that the affine subspaces are in $\mathbb{R}^{|V_H|}$, hence the inner loop can iterate at most $|V_H| \leq |V|$ times. □

It follows from the analysis above that if the clique decision oracle runs in polynomial time, then Algorithm 3 runs in polynomial time.

3 Certifying nonexistence of a clique of a given size in a perfect graph

In the previous section, we showed that optimal coloring of perfect graphs can be reduced, in polynomial time and via combinatorial operations, to deciding whether the graph has a clique of a given size. In this section, we provide a certificate for the "no" answer to this question in terms of counting closed walks in the graph with certain weights assigned to its edges and nonedges. This certificate will play a central role in the design of our main algorithm in the next section.

We begin by introducing notation for walks in weighted graphs. If G is a weighted graph, we write $A_G \in \mathbb{S}^n$ for its weighted adjacency matrix, where $(A_G)_{ij}$ denotes the weight of the edge ij. The diagonal entries of A_G denote the weight of self loops. We now define the total weight of walks in such a graph.

Definition 2 (Total weight of walks)**.** *Let T be a positive integer and G be a weighted graph with weighted adjacency matrix $A_G \in \mathbb{S}^n$. We define $\psi_{T,A_G}(i,j)$ to be the total weight of length T walks in G from vertex i to vertex j:*

$$\psi_{T,A_G}(i,j) = \sum_{(v_1,\ldots,v_{T+1}):v_1=i,v_{T+1}=j} \prod_{m=1}^{T} (A_G)v_m, v_{m+1}.$$

Here the sum is over all $(T+1)$-tuples $(v_1, v_2, \ldots, v_{T+1})$ of vertices with $v_1 = i$ and $v_{T+1} = j$. We denote the total weight of (closed) walks of length T from vertex i to itself by $\psi_{T,A_G}(i) \coloneqq \psi_{T,A_G}(i,i)$.

In our algorithm, we will assign weights to edges and nonedges of a perfect graph $G(V,E)$ according to a probability vector $p \in \Delta_{|\bar{E}|+1}$ to get a weighted graph G_p. More precisely, the weighted adjacency matrix is given by

$$A_{G_p} = \sum_{ij \in \bar{E}} -p_{ij} Y_{ij} + p_0 J + I, \tag{1}$$

where the matrices Y_{ij}, J, I are as introduced in Section 1.2, and $p = (p_0, p_{ij})$ for $ij \in \bar{E}$ is a probability vector defined over the nonedges of G and a special index 0. Figure 1 illustrates an example of a perfect graph G and the resulting weighted graph G_p when p is the uniform distribution.

The following theorem provides a certificate of nonexistence of a clique of a given size in a perfect graph G by counting the total weight of closed walks of a certain length in G_p for some probability vector p.

Theorem 5 (Certificate of nonexistence of a clique of size k)**.** *Let $G(V,E)$ be a perfect graph with $|V| = n$, $k \in \{2,\ldots,n\}$, and fix $\hat{\epsilon} \leq \frac{1}{(k+1)\left((k-1)n(n-1)+2\right)}$. Then for any $\hat{T} \geq \frac{1}{\hat{\epsilon}} \ln(\frac{2n}{\hat{\epsilon}})$, the following two statements are equivalent:*

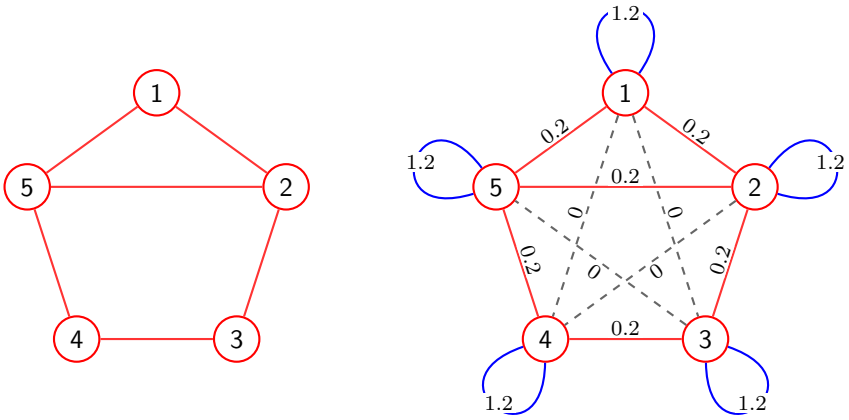
(i) G has no clique of size k,

(ii) there exists a probability vector $p \in \Delta_{|\bar{E}|+1}$, with $p = (p_0, p_{ij})$ for $ij \in \bar{E}$, such that the walk count ratio of the weighted graph with adjacency matrix $A_{G_p} = \sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J + I$ satisfies

$$\max_{i \in S} \frac{\psi_{2\hat{T}+1,A_{G_p}}(i)}{\psi_{2\hat{T},A_{G_p}}(i)} < (1-\hat{\epsilon})(p_0 k + 1),$$

where

$$S = \left\{ i \in [n] \mid \psi_{2\hat{T},A_{G_p}}(i) > 0 \right\}.$$



Perfect Graph G **Weighted Graph** G_p

Figure 1: An example of a perfect graph G and its associated weighted graph G_p when p is the uniform distribution.

Throughout the paper, we refer to the quantity $\max_{i\in S} \frac{\psi_{2\hat{T}+1,A_{G_p}}(i)}{\psi_{2\hat{T},A_{G_p}}(i)}$ as the *walk count ratio*. This quantity depends on the graph G, the probability vector p, and the integer $\hat{T}$. We defer the proof of Theorem 5 until after some auxiliary lemmas. The first lemma recalls the connection between walk counting and matrix multiplication. We omit the proof of this standard fact.

Lemma 6. *Let G be a weighted graph on n vertices with weighted adjacency matrix A_G. Then for any $i, j \in [n]$ and any integer $t \geq 1$, we have*

$$\psi_{t,A_G}(i,j) = (A_G^t)_{ij}.$$

The next lemma shows that when the weighted adjacency matrix is nonzero and positive semidefinite, its largest eigenvalue is well approximated by the walk count ratio for sufficiently large $\hat{T}$. The proof of this lemma appears in Appendix 6.1.

Lemma 7. *Let G be a weighted graph with a nonzero positive semidefinite weighted adjacency matrix A_G, and let ψ be defined as in Definition 2. Then for any $\epsilon \in (0,1]$ and $\hat{T} \geq \frac{1}{\epsilon}\ln(\frac{2n}{\epsilon})$, we have*

$$(1-\epsilon)\lambda_{\max}(A_G) \leq \max_{i\in S} \frac{\psi_{2\hat{T}+1,A_G}(i)}{\psi_{2\hat{T},A_G}(i)} \leq \lambda_{\max}(A_G),$$

where

$$S = \left\{ i \in [n] \mid \psi_{2\hat{T},A_G}(i) > 0 \right\}.$$

Our final lemma shows that the matrix A_{G_p} defined in (1) always satisfies the assumption of Lemma 7. The proof of this lemma can be found in Appendix 6.2.

Lemma 8. *For any graph $G(V,E)$, let $A_{G_p} = \sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J + I$, where $p = (p_0, p_{ij})$ for $ij \in \bar{E}$ is a probability vector. Then A_{G_p} is nonzero and positive semidefinite.*

We are now ready to prove Theorem 5.

Proof of Theorem 5. Fix $k \in \{2,\ldots,n\}$, $\hat{\epsilon} \leq \frac{1}{(k+1)\left((k-1)n(n-1)+2\right)}$, and $\hat{T} \geq \frac{1}{\hat{\epsilon}}\ln(\frac{2n}{\hat{\epsilon}})$.

(ii) $\Rightarrow$ (i). Suppose there is a probability vector $p = (p_0, p_{ij})$ for $ij \in \bar{E}$, such that

$$\max_{i\in S} \frac{\psi_{2\hat{T}+1,A_{G_p}}(i)}{\psi_{2\hat{T},A_{G_p}}(i)} < (1-\hat{\epsilon})(p_0 k + 1).$$

By Lemma 8, we know A_{G_p} is nonzero and positive semidefinite. Hence Lemma 7 applies and gives

$$(1-\hat{\epsilon})\lambda_{\max}(A_{G_p}) \leq \max_{i\in S} \frac{\psi_{2\hat{T}+1,A_{G_p}}(i)}{\psi_{2\hat{T},A_{G_p}}(i)} < (1-\hat{\epsilon})(p_0 k+1).$$

Since $1-\hat{\epsilon}>0$, it follows that

$$\lambda_{\max}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right) = \lambda_{\max}(A_{G_p}) - 1 < p_0 k.$$

Suppose for the sake of contradiction that G contains a clique C of size k. Consider the $|C|\times|C|$ matrix

$$\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right)_C,$$

which is the principal submatrix of $\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J$ indexed by the vertices in C. Since C is a clique, no pair of vertices in C is a nonedge of G. Hence all terms $-p_{ij}Y_{ij}$ vanish in this principal submatrix, and so

$$\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right)_C = p_0 J_C.$$

Thus,

$$\lambda_{\max}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right) \geq \lambda_{\max}(p_0 J_C) = p_0 k,$$

where the inequality follows, for example, from the Rayleigh quotient inequality. This contradicts the previous strict inequality. Therefore, G has no clique of size k.

(i) ⇒ (ii): Suppose G has no clique of size k. Since G is perfect, it follows that it is $(k-1)$-colorable. Pick any $(k-1)$-coloring $\mathcal{C}$ of G. Let $w_0 = \frac{1}{k-1}$ and $w_{ij} = \mathbf{1}(\mathcal{C}(i)=\mathcal{C}(j))$. Let

$$p_0 = \frac{w_0}{\sum_{ij\in\bar{E}} w_{ij} + w_0},\ p_{ij} = \frac{w_{ij}}{\sum_{ij\in\bar{E}} w_{ij} + w_0}.$$

Since vertices of the same color are nonadjacent, p_{ij} is nonzero only when $ij\in\bar{E}$. We first claim that

$$\lambda_{\max}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right) \leq p_0(k-1). \tag{2}$$

This is equivalent to showing that

$$\lambda_{\max}\left(\sum_{ij\in\bar{E}} -w_{ij}Y_{ij} + \frac{1}{k-1}J\right) \leq 1,$$

or that for every $x\in\mathbb{R}^n$,

$$\frac{1}{k-1}\left(\sum_{i=1}^n x_i\right)^2 \leq \sum_{i=1}^n x_i^2 + \sum_{ij\in\bar{E}} 2w_{ij}x_i x_j. \tag{3}$$

Let $I_{\mathcal{C},1}, I_{\mathcal{C},2}, \ldots, I_{\mathcal{C},k-1}$ denote the color classes of the coloring $\mathcal{C}$. The right-hand side of (3) can be rewritten as

$$\begin{aligned}\sum_{i=1}^n x_i^2 + 2\sum_{ij\in\bar{E}} \mathbf{1}(\mathcal{C}(i)=\mathcal{C}(j))x_i x_j &= \sum_{l=1}^{k-1}\sum_{i\in I_{\mathcal{C},l}} x_i^2 + \sum_{l=1}^{k-1}\sum_{\substack{i<j\\ i,j\in I_{\mathcal{C},l}}} 2x_i x_j \\ &= \sum_{l=1}^{k-1}\left(\sum_{i\in I_{\mathcal{C},l}} x_i\right)^2.\end{aligned}$$

Applying the Cauchy-Schwarz inequality to the left-hand side of (3), we have

$$\frac{1}{k-1}\left(\sum_{i=1}^{n} x_i\right)^2 = \frac{1}{k-1}\left(\sum_{l=1}^{k-1}\sum_{i\in I_{\mathcal{C},l}} x_i\right)^2 \leq \sum_{l=1}^{k-1}\left(\sum_{i\in I_{\mathcal{C},l}} x_i\right)^2.$$

Hence, the inequality (2) is proven. Using this inequality, we now verify that $p \coloneqq (p_0, p_{ij})$ for $ij \in \bar{E}$ satisfies condition (ii) of the theorem. Since $w_{ij} \in \{0,1\}$, we have

$$p_0 = \frac{\frac{1}{k-1}}{\sum_{ij\in\bar{E}} w_{ij} + \frac{1}{k-1}} \geq \frac{\frac{1}{k-1}}{\frac{n(n-1)}{2} + \frac{1}{k-1}} = \frac{2}{(k-1)n(n-1)+2}.$$

Recalling that $\hat{\epsilon} \leq \frac{1}{(k+1)\left((k-1)n(n-1)+2\right)}$, it follows that

$$(p_0 k + 1)\hat{\epsilon} \leq \frac{1}{(k-1)n(n-1)+2} < p_0,$$

and therefore

$$p_0(k-1) + 1 < (1-\hat{\epsilon})(p_0 k + 1). \tag{4}$$

We now observe that

$$\begin{aligned}
\max_{i\in S} \frac{\psi_{2\hat{T}+1,A_{G_p}}(i)}{\psi_{2\hat{T},A_{G_p}}(i)} &\leq \lambda_{\max}(A_{G_p}) \\
&= \lambda_{\max}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij} + p_0 J\right) + 1 \\
&\leq p_0(k-1) + 1 \\
&< (1-\hat{\epsilon})(p_0 k + 1),
\end{aligned}$$

where the first inequality follows from Lemma 7, the second from (2), and the third from (4). This completes the proof.[4]

□

4 Deciding if a perfect graph has a clique of a given size

Following the reduction established in Section 2, the problem of optimally coloring a perfect graph G reduces to deciding if G has a clique of a given size $k \in \{2, \ldots, |V|\}$. In this section, we present a combinatorial algorithm (Algorithm 5) for this decision problem. Following Theorem 5, the algorithm attempts to construct a probability vector that certifies the nonexistence of a k-clique. To do so, it iteratively updates a probability vector based on closed walk counts in G with certain weights assigned to its edges and nonedges. We begin by detailing the walk counting procedure (Algorithm 4). We then present the main algorithm (Algorithm 5) and establish its correctness and polynomial-time complexity via a Lyapunov function argument.

4.1 The walk counting algorithm

The walk counting algorithm (Algorithm 4) takes as input a weighted adjacency matrix $A \in \mathbb{Q}^{n\times n}$ and a positive integer $\hat{T}$. It outputs a scalar β, which we call the walk count ratio, and a vector $x \in \mathbb{Q}^n$, which we call the walk count vector. The expression for the walk count ratio is the same as Theorem 5:

$$\beta = \max_{i\in S} \frac{\psi_{2\hat{T}+1,A}(i)}{\psi_{2\hat{T},A}(i)},$$

[4]Note from the proof that the implication $(ii) \Rightarrow (i)$ in Theorem 5 holds for arbitrary graphs, while the implication $(i) \Rightarrow (ii)$ holds for graphs G that satisfy $\chi(G) = \omega(G)$.

where ψ is as in Definition 2 and $S = \{i \in [n] \mid \psi_{2\hat{T},A}(i) > 0\}$. Let i_0 be an index that achieves this maximum. The walk count vector collects the total weight of walks of length $\hat{T}$ from i_0 to all the other vertices; *i.e.,* it is defined entrywise as

$$x_j = \psi_{\hat{T},A}(i_0, j), \quad \forall j \in [n].$$

Algorithm 4: Walk counting

Input: A weighted adjacency matrix $A \in \mathbb{Q}^{n\times n}$, a positive integer $\hat{T} \in \mathbb{Z}$
Output: A scalar $\beta \in \mathbb{Q}$ and a vector $x \in \mathbb{Q}^n$
Initialize: $\psi_{1,A}(i,j) \leftarrow A_{ij}$ for all $i, j \in [n]$
for $t \leftarrow 1$ **to** $2\hat{T}$ **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** n **do**
      $\psi_{t+1,A}(i,j) \leftarrow \sum_{i_1 \in [n]} A_{i_1,j} \cdot \psi_{t,A}(i, i_1)$
    end
  end
end
$S \leftarrow \left\{i \in [n] \mid \psi_{2\hat{T},A}(i) > 0\right\}$
$\beta \leftarrow \max_{i\in S} \frac{\psi_{2\hat{T}+1,A}(i)}{\psi_{2\hat{T},A}(i)}$
$i_0 \leftarrow \arg\max_{i\in S} \frac{\psi_{2\hat{T}+1,A}(i)}{\psi_{2\hat{T},A}(i)}$ `// pick any one if the maximizer is not unique`
for $j \leftarrow 1$ **to** n **do**
  $x_j \leftarrow \psi_{\hat{T},A}(i_0, j)$
end
Output: (β, x)

Let us argue that the walk counting algorithm correctly returns the walk count ratio β and the walk count vector x by showing that the nested loops in this algorithm correctly compute $\psi_{m,A}(i,j)$, the total weight of walks of length m from vertex i to vertex j. For $m = 1$, the algorithm initializes $\psi_{1,A}(i,j) = A_{ij}$, which is consistent with the definition. For $m \geq 1$, the total weight of walks satisfies the recurrence:

$$\begin{aligned}\psi_{m+1,A}(i,j) &= \sum_{k\in[n]} A_{k,j} \sum_{\substack{\text{length } m \text{ walk} \\ \ell:\ell(1)=i,\, \ell(m+1)=k}} \prod_{r=1}^{m} A_{\ell(r),\ell(r+1)} \\ &= \sum_{k\in[n]} A_{k,j} \cdot \psi_{m,A}(i,k).\end{aligned}$$

This recurrence is precisely the update rule in the innermost loop of Algorithm 4. The time complexity of Algorithm 4 is $O(n^3\hat{T})$.

4.2 The main algorithm

In this section, we present our main algorithm (Algorithm 5), which given a perfect graph $G(V,E)$ and an integer $k \in \{2, \ldots, |V|\}$ determines whether G contains a clique of size k.[5]

In each iteration, the algorithm maintains a probability vector $p^t \in \Delta_{|\bar{E}|+1}$, which defines a weighted graph G_{p_t} with adjacency matrix $A_{G_{p_t}}$ as in (1). The algorithm then invokes the walk counting algorithm (Algorithm 4) on G_{p^t} to compute the walk count ratio β^t and the walk count vector x^t.

[5]As the proof of correctness of Algorithm 5 will demonstrate, this algorithm will output the correct decision more generally on graphs for which the clique number and the chromatic number coincide.

Algorithm 5: Deciding if a perfect graph G has a clique of size k

Input: Perfect graph $G(V,E)$ with $|V|= n$, integer $k \in \{2,\dots,n\}$

Output: "YES" or "NO"

Set parameters: $\epsilon = \frac{1}{2n}\frac{1}{n(n-1)(k-1)+2}$, $\hat{\epsilon} \in \left(0, \frac{1}{(k+1)\big((k-1)n(n-1)+2\big)}\right]$, integer $T > 4\ln(|\bar{E}|+1)n^2\Big((k-1)n(n-1)+2\Big)^2$, integer $\hat{T} \geq \frac{1}{\hat{\epsilon}}\ln(\frac{2n}{\hat{\epsilon}})$ `// any value of` $\hat{\epsilon}, T, \hat{T}$ `which results in` T `and` $\hat{T}$ `having polynomial magnitude is acceptable`

Initialize weights: $w_{ij}^1 \leftarrow 1,$ for $ij \in \bar{E}$, $w_0^1 \leftarrow 1$

for $t \leftarrow 1$ **to** T **do**

  Normalize the weights:

$$p_{ij}^t \leftarrow \frac{w_{ij}^t}{\sum_{ij\in\bar{E}} w_{ij}^t + w_0^t}, \quad p_0^t \leftarrow \frac{w_0^t}{\sum_{ij\in\bar{E}} w_{ij}^t + w_0^t}$$

  Construct a weighted graph G_{p^t} with weighted adjacency matrix $A_{G_{p^t}} = \sum_{ij\in\bar{E}} -p_{ij}^t Y_{ij} + p_0^t J + I$.

  $(\beta^t, x^t) \leftarrow$ Algorithm 4$(A_{G_{p^t}}, \hat{T})$ `// compute walk count ratio and walk counts`

  if $\beta^t < (1-\hat{\epsilon})(p_0^t k + 1)$ **then**

    return "NO"

  end

  Update weights based on walk counts:

$$w_{ij}^{t+1} \leftarrow w_{ij}^t\left(1 + \frac{\epsilon}{n}\frac{2x_i^t x_j^t}{\|x^t\|_2^2}\right),\ w_0^{t+1} \leftarrow w_0^t\left(1 - \frac{\epsilon}{n}\left(\frac{(\sum_{i\in[n]} x_i^t)^2}{\|x^t\|_2^2} - k\right)\right)$$

end

Return "YES"

If $\beta^t < (1-\hat{\epsilon})(p_0^t k + 1)$, Theorem 5 guarantees that there is no clique of size k in G; hence, the algorithm returns "NO". Otherwise, the weights are updated based on the walk counts in x^t, and the process repeats. A combinatorial interpretation of the quantities appearing in the weight update step is as follows. Let i_0^t denote the index maximizing the walk count ratio β^t. Recall that x_j^t represents the total weight of walks of length $\hat{T}$ from i_0^t to j in G_{p^t}. The quantities in the update rule have the following combinatorial interpretation:

- $x_i^t x_j^t$ is the total weight of walks of length $2\hat{T}$ from i to j passing through i_0^t at the midpoint.
- $\|x^t\|_2^2$ is the total weight of closed walks of length $2\hat{T}$ passing through i_0^t at the midpoint.
- $(\sum_{i\in[n]} x_i^t)^2$ is the total weight of all walks of length $2\hat{T}$ passing through i_0^t at the midpoint.

If the loop terminates after T iterations without a "NO" certificate, the algorithm returns "YES". We defer the proof of correctness of Algorithm 5 to the next section. We note that since the bounds on T and $\hat{T}$ are of polynomial magnitude, Algorithm 5 runs in polynomial time.

4.3 Proof of correctness of the main algorithm

In this section, we prove the correctness of Algorithm 5. The strategy is as follows: If there is a clique of size k in the perfect graph G, Theorem 5 implies that no iterate p^t can satisfy the termination condition (in line 9), so the algorithm will necessarily return 'YES'. If there is no clique of size k in G, we define a Lyapunov function $L : \Delta_{|\bar{E}|+1} \to \mathbb{R}$ to measure how far the current probability vector p^t is from the target probability vector constructed in the proof of Theorem 5. We further argue that this Lyapunov function is nonnegative, not too large at the first iterate, and decreases at least by a certain fixed amount in each iteration of the algorithm. From this analysis, it follows that the probability vectors produced by the algorithm must necessarily satisfy the termination condition within T steps, leading to the output 'NO' in polynomial time.

Let us now formalize the proof strategy. Suppose the perfect graph G has no clique of size k. By perfectness, we can pick a $(k-1)$-coloring of G, denoted by $\mathcal{C}$. Then, we let $w_0^\star = \frac{1}{k-1}, w_{ij}^\star = \mathbf{1}(\mathcal{C}(i) = \mathcal{C}(j))$, and

$$p_0^\star = \frac{w_0^\star}{\sum_{ij\in\bar{E}} w_{ij}^\star + w_0^\star},\ p_{ij}^\star = \frac{w_{ij}^\star}{\sum_{ij\in\bar{E}} w_{ij}^\star + w_0^\star}.$$

And we write $p^\star = (p_0^\star, p_{ij}^\star)$, for $ij \in \bar{E}$. From the proof of Theorem 5 (more specifically inequality (2)), we know that $p^\star$ satisfies

$$\lambda_{\max}\left(\sum_{ij\in\bar{E}} -p_{ij}^\star Y_{ij} + p_0^\star(J - kI)\right) \le -p_0^\star \le -\frac{2}{(k-1)n(n-1)+2}. \tag{5}$$

Consider the following Lyapunov function $L : \Delta_{|\bar{E}|+1} \to \mathbb{R}$ defined as

$$L(p) = D_{KL}(p^\star \| p) \coloneqq \sum_{ij\in\bar{E}} p_{ij}^\star \ln\frac{p_{ij}^\star}{p_{ij}} + p_0^\star \ln\frac{p_0^\star}{p_0}. \tag{6}$$

The choice of the Kullback-Leibler (KL) divergence as a Lyapunov function has been made previously; see, e.g., [22, 23]. Let $x^t \in \mathbb{R}^n$ be the walk count vector at iteration t of Algorithm 5, and let

$$r_{ij}^t = -\frac{2x_i^t x_j^t}{\|x^t\|_2^2},\ \text{for } ij \in \bar{E},\quad r_0^t = \frac{(\sum_{i\in[n]} x_i^t)^2}{\|x^t\|_2^2} - k.$$

Let $p_0^t, p_{ij}^t \in [0,1]$ be the scalars at iteration t of Algorithm 5. We define the vectors $p^t \in \Delta_{|\bar{E}|+1}$ and $r^t \in \mathbb{R}^{|\bar{E}|+1}$ as

$$\begin{aligned} p^t &= (p_0^t, p_{ij}^t), \\ r^t &= (r_0^t, r_{ij}^t). \end{aligned} \tag{7}$$

To prove that our Lyapunov function decreases along the iterations of Algorithm 5, we first establish two lemmas.

Lemma 9. *Let r^t be the vector defined in (7) at iteration t of Algorithm 5 with input $G(V,E)$ and $k \in \{2,\dots,n\}$. We have $\|r^t\|_\infty \leq n$.*

Proof. Note that for $ij \in \bar{E}$, we have

$$|r^t_{ij}| = \left|\frac{2x^t_i x^t_j}{\|x^t\|_2^2}\right| \leq \frac{(x^t_i)^2 + (x^t_j)^2}{\|x^t\|_2^2} \leq 1.$$

Furthermore,

$$r^t_0 = \frac{(\sum_{i\in[n]} x^t_i)^2}{\|x^t\|_2^2} - k \leq \frac{\|x^t\|_1^2}{\|x^t\|_2^2} - k \leq n - k,$$

and

$$r^t_0 = \frac{(\sum_{i\in[n]} x^t_i)^2}{\|x^t\|_2^2} - k \geq -k,$$

and hence $|r^t_0| \leq n$. Therefore, $\|r^t\|_\infty \leq n$. □

Lemma 10. *Let p^t and r^t be the vectors defined in (7) at iteration t of Algorithm 5 with input $G(V,E)$ and $k \in \{2,\dots,n\}$. We have*

$$\frac{p^t_m}{p^{t+1}_m} \leq \frac{\exp(-\frac{\epsilon}{n}\langle p^t, r^t\rangle)}{1 - \frac{\epsilon}{n} r^t_m},$$

for all $m \in \bar{E} \cup \{0\}$.

Proof. Define $\Phi^t = \sum_{ij\in\bar{E}} w^t_{ij} + w^t_0$. From the update rule of w^t and Lemma 9, we know that $\Phi^t > 0$. Observe that

$$\begin{aligned}
\Phi^{t+1} &= \sum_{k\in\bar{E}\cup\{0\}} w^{t+1}_k \\
&= \sum_{k\in\bar{E}\cup\{0\}} w^t_k \,(1 - \frac{\epsilon}{n} r^t_k) \\
&= \Phi^t - \frac{\epsilon}{n}\Phi^t \left\langle p^t,\, r^t \right\rangle \\
&= \Phi^t (1 - \frac{\epsilon}{n} \left\langle p^t,\, r^t \right\rangle) \\
&\leq \Phi^t \exp\Big(-\frac{\epsilon}{n} \left\langle p^t,\, r^t \right\rangle\Big),
\end{aligned}$$

where the last step follows from the facts that $1 + x \leq e^x$ for all $x \in \mathbb{R}$ and $\Phi_t > 0$. Since from Lemma 9 we know that $1 - \frac{\epsilon}{n} r^t_m > 0$ for all $m \in \bar{E} \cup \{0\}$, we have

$$\frac{p^t_m}{p^{t+1}_m} = \frac{w^t_m}{w^{t+1}_m} \cdot \frac{\Phi^{t+1}}{\Phi^t} = \frac{1}{1 - \frac{\epsilon}{n} r^t_m} \cdot \frac{\Phi^{t+1}}{\Phi^t} \leq \frac{\exp(-\frac{\epsilon}{n}\langle p^t, r^t\rangle)}{1 - \frac{\epsilon}{n} r^t_m}.$$

□

We can now bound the amount by which the proposed Lyapunov function decreases at every iteration of Algorithm 5.

Theorem 11 (Lyapunov function decrease)**.** *Let $G(V,E)$ be a perfect graph and $k \in \{2,\dots,n\}$. Let p^t be the vector defined in (7) at iteration t of Algorithm 5 with input $G(V,E)$ and k. If G has no clique of size k, then either the termination condition in Algorithm 5 is satisfied at iteration t, namely,*

$$\max_{i\in S^t} \frac{\psi_{2\hat{T}+1, A_{G_{p^t}}}(i)}{\psi_{2\hat{T}, A_{G_{p^t}}}(i)} < (1-\hat{\epsilon})(p^t_0 k + 1),$$

where $S^t := \{i \in [n] \mid \psi_{2\hat{T}, A_{G_{p^t}}}(i) > 0\}$, or the Lyapunov function L defined in (6) satisfies

$$L(p^{t+1}) - L(p^t) \leq -\frac{1}{4n^2((k-1)n(n-1)+2)^2}. \tag{8}$$

Proof. Fix an iteration t, and suppose that Algorithm 5 does not terminate with output "NO" at this iteration. We show that inequality (8) must hold. Recall that in Algorithm 5, we choose

$$\epsilon = \frac{1}{2n}\frac{1}{n(n-1)(k-1)+2} \leq \frac{1}{2}.$$

Together with Lemma 9, this implies that $\frac{\epsilon}{n} r_m^t \leq \frac{1}{2}$ for every index $m \in \bar{E} \cup \{0\}$. Since the vector w^1 of initial weights is at the all-ones vector, it follows that all coordinates of p^t and p^{t+1} are positive, so all logarithms below are well-defined. Using the definition of L in (6), we have

$$\begin{aligned}
L(p^{t+1}) - L(p^t) &= \sum_{ij\in\bar{E}} p_{ij}^\star \ln \frac{p_{ij}^\star}{p_{ij}^{t+1}} + p_0^\star \ln \frac{p_0^\star}{p_0^{t+1}} - \sum_{ij\in\bar{E}} p_{ij}^\star \ln \frac{p_{ij}^\star}{p_{ij}^t} - p_0^\star \ln \frac{p_0^\star}{p_0^t} \\
&= \sum_{ij\in\bar{E}} p_{ij}^\star \ln \frac{p_{ij}^t}{p_{ij}^{t+1}} + p_0^\star \ln \frac{p_0^t}{p_0^{t+1}} \\
&\leq \sum_{ij\in\bar{E}} p_{ij}^\star \left(-\frac{\epsilon}{n}\langle p^t, r^t\rangle - \ln(1 - \frac{\epsilon}{n} r_{ij}^t)\right) + p_0^\star \left(-\frac{\epsilon}{n}\langle p^t, r^t\rangle - \ln(1 - \frac{\epsilon}{n} r_0^t)\right) \\
&= -\frac{\epsilon}{n}\langle p^t, r^t\rangle - \sum_{ij\in\bar{E}} p_{ij}^\star \ln(1 - \frac{\epsilon}{n} r_{ij}^t) - p_0^\star \ln(1 - \frac{\epsilon}{n} r_0^t) \\
&\leq -\frac{\epsilon}{n}\langle p^t - p^\star, r^t\rangle + \sum_{ij\in\bar{E}} p_{ij}^\star \left(\frac{\epsilon}{n}\right)^2 (r_{ij}^t)^2 + p_0^\star \left(\frac{\epsilon}{n}\right)^2 (r_0^t)^2 \\
&\leq -\frac{\epsilon}{n}\langle p^t - p^\star, r^t\rangle + \epsilon^2,
\end{aligned}$$

where the first inequality follows from Lemma 10, the second inequality from the fact that for $x \leq 1/2$, $-\ln(1-x) \leq x + x^2$, and the last one from Lemma 9. We next bound $\langle p^t, r^t\rangle$ and $\langle p^\star, r^t\rangle$ separately. Recall that at iteration t of Algorithm 5, Algorithm 4 returns the walk count ratio β^t and the walk count vector x^t, which in view of Lemma 6 and their definitions satisfy

$$\begin{aligned}
\beta^t &= \max_{i\in S^t} \frac{\psi_{2\hat{T}+1, A_{G_{p^t}}}(i)}{\psi_{2\hat{T}, A_{G_{p^t}}}(i)} \\
&= \max_{i\in S^t} \frac{(A_{G_{p^t}}^{\hat{T}} e_i)^\mathsf{T} A_{G_{p^t}} (A_{G_{p^t}}^{\hat{T}} e_i)}{\|A_{G_{p^t}}^{\hat{T}} e_i\|_2^2} \\
&= \frac{(x^t)^\mathsf{T}(\sum_{ij\in\bar{E}} -p_{ij}^t Y_{ij} + p_0^t J + I)x^t}{\|x^t\|_2^2},
\end{aligned}$$

where $S^t = \left\{ i \in [n] \mid \psi_{2\hat{T}, A_{G_{p^t}}}(i) > 0 \right\}$. Then, by the definitions of p^t and r^t in (7), we have

$$\langle p^t, r^t\rangle = \frac{(x^t)^\mathsf{T}(\sum_{ij\in\bar{E}} -p_{ij}^t Y_{ij} + p_0^t(J - kI))x^t}{\|x^t\|_2^2} = \beta^t - p_0^t k - 1.$$

Since the termination condition of Algorithm 5 is not satisfied, this gives

$$\langle p^t, r^t\rangle = \beta^t - p_0^t k - 1 \geq -\hat{\epsilon}(p_0^t k + 1).$$

By the definitions of $p^\star$ and r^t, and by the Rayleigh quotient inequality, we have

$$\langle p^\star, r^t\rangle = \frac{(x^t)^\mathsf{T}(\sum_{ij\in\bar{E}} -p_{ij}^\star Y_{ij} + p_0^\star(J - kI))x^t}{\|x^t\|_2^2} \leq \lambda_{\max}(\sum_{ij\in\bar{E}} -p_{ij}^\star Y_{ij} + p_0^\star(J - kI)).$$

Using the two inequalities above, we obtain

$$\begin{aligned}
L(p^{t+1}) - L(p^t) &\leq -\frac{\epsilon}{n}(\langle p^t, r^t\rangle - \langle p^\star, r^t\rangle) + \epsilon^2 \\
&\leq -\frac{\epsilon}{n}(-\hat{\epsilon}(p_0^t k + 1) - \lambda_{\max}(\sum_{ij\in\bar{E}} -p_{ij}^\star Y_{ij} + p_0^\star(J - kI))) + \epsilon^2 \\
&\leq -\frac{\epsilon}{n}(-\hat{\epsilon}(k+1) + \frac{2}{(k-1)n(n-1)+2}) + \epsilon^2 \\
&\leq -\frac{\epsilon}{n}\frac{1}{(k-1)n(n-1)+2} + \epsilon^2 \\
&= -\frac{1}{4n^2((k-1)n(n-1)+2)^2},
\end{aligned}$$

where the third inequality follows from (5) and the fact that $p_0^t \leq 1$, the fourth inequality from the fact that $\hat{\epsilon} \leq \frac{1}{(k+1)\left((k-1)n(n-1)+2\right)}$, and the equation from our choice of $\epsilon = \frac{1}{2n}\frac{1}{n(n-1)(k-1)+2}$. □

We are now ready to formalize the proof of correctness of Algorithm 5.

Theorem 12 (Correctness of Algorithm 5)**.** *Given a perfect graph $G(V, E)$ with $|V| = n$ and an integer $k \in \{2, \ldots, n\}$, Algorithm 5 correctly decides if there is a clique of size k in G.*

Proof. We split the proof into two cases.

Case 1: G has no clique of size k. First recall that $p^1 \in \Delta_{|\bar{E}|+1}$ is the uniform distribution. Thus, we can bound the initial value of the Lyapunov function defined in (6) as

$$L(p^1) = \ln(|\bar{E}|+1) + \sum_{ij\in\bar{E}} p_{ij}^\star \ln(p_{ij}^\star) + p_0^\star \ln(p_0^\star) \leq \ln(|\bar{E}|+1).$$

In each iteration t of Algorithm 5, either the termination condition is satisfied and the algorithm correctly outputs "NO", or, by Theorem 11, the Lyapunov function L satisfies

$$L(p^{t+1}) - L(p^t) \leq -\frac{1}{4n^2((k-1)n(n-1)+2)^2}.$$

Suppose for the sake of contradiction that Algorithm 5 does not return "NO" in the first T iterations. Then,

$$L(p^{T+1}) \leq L(p^1) - \frac{T}{4n^2((k-1)n(n-1)+2)^2}. \tag{9}$$

By our choice of T we have

$$\frac{T}{4n^2((k-1)n(n-1)+2)^2} > \ln(|\bar{E}|+1).$$

Recalling that $L(p^1) \leq \ln(|\bar{E}|+1)$, we observe that (9) implies $L(p^{T+1}) < 0$. This contradicts the fact that the KL divergence is nonnegative. Therefore, Algorithm 5 returns "NO" within the first T iterations.

Case 2: There is a clique of size k in G. Suppose for the sake of contradiction that Algorithm 5 returns "NO" at iteration t. This can only happen if the termination condition is satisfied; i.e.,

$$\beta^t < (1 - \hat{\epsilon})(p_0^t k + 1).$$

By Theorem 5, this implies that there is no clique of size k in G, contradicting the assumption. Thus, Algorithm 5 must necessarily return "YES". □

5 Connections with other optimization algorithms

In a seminal paper [24], Lovász introduced the *theta number* $\vartheta(G)$ of a graph $G(V, E)$ as the optimal value of the following semidefinite program (SDP):

$$\begin{aligned} \vartheta(G) \coloneqq \max_{X \in \mathbb{S}^n} \quad & \operatorname{Tr}(JX) \\ \text{subject to} \quad & \operatorname{Tr}(X) = 1, \\ & X \succeq 0, \\ & X_{ij} = 0, \quad \text{for } ij \in E. \end{aligned}$$

He further showed that for any graph G, the theta number of the complement graph $\bar{G}$ satisfies the following inequalities:

$$\omega(G) \leq \vartheta(\bar{G}) \leq \chi(G).$$

It follows that for perfect graphs, the above three parameters coincide. Therefore, *any* algorithm that computes the clique number of a perfect graph is, in effect, also solving the above semidefinite program on the complement graph. In this section, we argue that Algorithm 5 can be viewed as a combinatorial implementation of a multiplicative weights update algorithm for solving a modified version of the Lovász SDP, or a combinatorial implementation of the exponentiated gradient descent algorithm applied to the dual of this SDP. In fact, these algorithms were our starting point for the design of Algorithm 5. The modified version of the Lovász SDP that is of interest to us is due to Szegedy [25] and is formulated as follows:

$$\begin{aligned} \hat{\vartheta}(G) \coloneqq \max_{X \in \mathbb{S}^n} \quad & \operatorname{Tr}(JX) \\ \text{subject to} \quad & \operatorname{Tr}(X) = 1, \\ & X \succeq 0, \\ & X_{ij} \leq 0, \quad \text{for } ij \in E. \end{aligned} \tag{10}$$

It has been shown in [26] that $\hat{\vartheta}(\bar{G})$ satisfies the same inequalities as the theta number:

$$\omega(G) \leq \hat{\vartheta}(\bar{G}) \leq \chi(G).$$

The multiplicative weights update method is a general algorithmic framework with many applications; see [23] and the references therein. In particular, it can be used to approximately solve semidefinite programming feasibility problems. The algorithm takes an SDP feasibility problem and a quantity called the SDP "width" as inputs. For a specified accuracy parameter $\delta > 0$, it either outputs that the SDP is infeasible, or returns a δ-feasible point; i.e., a positive semidefinite matrix that violates every linear constraint by at most δ. In the latter case, the SDP may still be infeasible. The algorithm maintains a set of weights associated with the linear constraints, which are updated in each iteration according to an exponential update rule. Moreover, in each iteration, the algorithm calls an oracle that computes the largest eigenvalue and a corresponding eigenvector of a certain symmetric matrix.

In Algorithm 5, we replace the exponential update rule by its first-order Taylor approximation. We also approximate the eigenvalue/eigenvector oracle by the *power method*, which gives rise to the connection with walk counting. We show that the "width" parameter of the SDP in (10) can be taken to be n. Finally, we prove that despite the approximation errors made by the power method, the Taylor expansion, and the multiplicative weights update algorithm itself, we can still correctly decide if there is a clique of size k in a perfect graph.

Algorithm 5 can also be viewed as a combinatorial implementation of the exponentiated gradient descent algorithm applied to the dual of the modified Lovász SDP. By taking the dual of the SDP in (10) with $\bar{G}$ as input, one can invoke SDP strong duality to show that a perfect graph $G(V, E)$ has no clique of size k if and only if the following optimization problem has a negative optimal value:

$$\min_{p \in \Delta_{|\bar{E}|+1}} \quad \lambda_{\max}\left(p_0(J - kI) - \sum_{ij \in \bar{E}} p_{ij} Y_{ij} \right). \tag{11}$$

The exponentiated gradient descent algorithm is a method for minimizing convex functions over the probability simplex, which is precisely the form of problem (11). This algorithm can be seen as a specific instance of the so-called mirror descent method with negative entropy serving as the "mirror map"; see [27, 28] for details. The algorithm has an exponential update rule, a step size, and needs access to subgradients of the objective function. Algorithm 5 can be viewed as a combinatorial implementation of this algorithm where the step size is taken to be ϵ/n, the exponential update rule is Taylor expanded to first order, and the computation of the subgradient is approximated by the power method.

We note that the fact that Algorithm 5 has the above connections to approximate versions of first-order methods for semidefinite programming does not imply that it is non-combinatorial. Indeed, as we have shown in Sections 3–4, both the algorithm and its proof of correctness can be fully understood without any knowledge of semidefinite programming.

Acknowledgments.

The authors are grateful to Maria Chudnovsky and Cemil Dibek for their contributions to this project and many insightful discussions. We also thank Paul Seymour and Noga Alon for their invaluable feedback.

6 Appendix

6.1 Proof of Lemma 7

Proof. We first show that $S \neq \varnothing$. Let the eigenvalues of A_G be $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Since A_G is nonzero and positive semidefinite, $\lambda_1 = \lambda_{\max}(A_G) > 0$. By Lemma 6, we have

$$\sum_{i=1}^{n} \psi_{2\hat{T},A_G}(i) = \sum_{i=1}^{n} (A_G^{2\hat{T}})_{ii} = \operatorname{Tr}(A_G^{2\hat{T}}) = \sum_{i=1}^{n} \lambda_i^{2\hat{T}} > 0.$$

Therefore, there exists some $i \in [n]$ such that $\psi_{2\hat{T},A_G}(i) > 0$.

Let us prove the desired the upper bound. Fixing $i \in S$, Lemma 6 and the Rayleigh quotient inequality give

$$\frac{\psi_{2\hat{T}+1,A_G}(i)}{\psi_{2\hat{T},A_G}(i)} = \frac{(A_G^{2\hat{T}+1})_{ii}}{(A_G^{2\hat{T}})_{ii}} = \frac{(A_G^{\hat{T}}e_i)^\top A_G (A_G^{\hat{T}}e_i)}{(A_G^{\hat{T}}e_i)^\top (A_G^{\hat{T}}e_i)} \leq \lambda_{\max}(A_G).$$

We next prove the desired lower bound. If $i \notin S$, then $0 = \psi_{2\hat{T},A_G}(i) = (A_G^{2\hat{T}})_{ii} = (A_G^{\hat{T}}e_i)^\top (A_G^{\hat{T}}e_i)$, which implies $A_G^{\hat{T}}e_i = 0$. Therefore, $\psi_{2\hat{T}+1,A_G}(i) = (A_G^{2\hat{T}+1})_{ii} = (A_G^{\hat{T}}e_i)^\top A_G (A_G^{\hat{T}}e_i) = 0$. Therefore, we have

$$\max_{i\in S} \frac{\psi_{2\hat{T}+1,A_G}(i)}{\psi_{2\hat{T},A_G}(i)} \geq \frac{\sum_{i\in S} \psi_{2\hat{T},A_G}(i) \cdot \frac{\psi_{2\hat{T}+1,A_G}(i)}{\psi_{2\hat{T},A_G}(i)}}{\sum_{i\in S} \psi_{2\hat{T},A_G}(i)} = \frac{\sum_{i=1}^{n} \psi_{2\hat{T}+1,A_G}(i)}{\sum_{i=1}^{n} \psi_{2\hat{T},A_G}(i)} = \frac{\sum_{i=1}^{n} \lambda_i^{2\hat{T}+1}}{\sum_{i=1}^{n} \lambda_i^{2\hat{T}}}.$$

To bound the last quotient, let

$$\ell \coloneqq \left| \left\{ i \in [n] : \lambda_i > \lambda_1 \Big(1 - \frac{\epsilon}{2}\Big) \right\} \right|.$$

We can lower bound the numerator as

$$\sum_{i=1}^{n} \lambda_i^{2\hat{T}+1} \geq \sum_{i=1}^{\ell} \lambda_i^{2\hat{T}+1} \geq \lambda_1 \Big(1 - \frac{\epsilon}{2}\Big) \sum_{i=1}^{\ell} \lambda_i^{2\hat{T}}.$$

Since $\lambda_i \le \lambda_1(1-\epsilon/2)$ for $i > \ell$, we can upper bound the denominator as

$$\begin{aligned}\sum_{i=1}^{n} \lambda_i^{2\hat{T}} &= \sum_{i=1}^{\ell} \lambda_i^{2\hat{T}} + \sum_{i=\ell+1}^{n} \lambda_i^{2\hat{T}} \\ &\le \sum_{i=1}^{\ell} \lambda_i^{2\hat{T}} + (n-\ell)\lambda_1^{2\hat{T}} \left(1-\frac{\epsilon}{2}\right)^{2\hat{T}} \\ &\le \left(1 + n\left(1-\frac{\epsilon}{2}\right)^{2\hat{T}}\right) \sum_{i=1}^{\ell} \lambda_i^{2\hat{T}}.\end{aligned}$$

Using the two bounds above, we have

$$\begin{aligned}\max_{i\in S} \frac{\psi_{2\hat{T}+1,A_G}(i)}{\psi_{2\hat{T},A_G}(i)} &\ge \frac{1-\frac{\epsilon}{2}}{1+n(1-\frac{\epsilon}{2})^{2\hat{T}}}\lambda_1 \\ &\ge \frac{1-\frac{\epsilon}{2}}{1+\frac{\epsilon}{2}}\lambda_1 \\ &\ge (1-\varepsilon)\lambda_1 \\ &= (1-\epsilon)\lambda_{\max}(A_G),\end{aligned} \tag{12}$$

where the second inequality follows from the fact that $\hat{T} \ge \frac{1}{\epsilon}\ln(\frac{2n}{\epsilon})$ and $\ln(1-\frac{\epsilon}{2}) \le -\frac{\epsilon}{2}$, and the third from the fact that for $x \in [0,1]$, $\frac{1-x}{1+x} \ge 1-2x$.

□

6.2 Proof of Lemma 8

Proof. We first prove that for any $B, C \in \mathbb{S}^n$, the following inequality holds:

$$\lambda_{\min}(B+C) \ge \lambda_{\min}(B) + \lambda_{\min}(C).$$

To see this, let x be a unit eigenvector corresponding to $\lambda_{\min}(B+C)$. By the definition of the Rayleigh quotient, we have

$$\lambda_{\min}(B+C) = x^{\mathsf{T}}(B+C)x = x^{\mathsf{T}}Bx + x^{\mathsf{T}}Cx \ge \lambda_{\min}(B) + \lambda_{\min}(C).$$

Using this property, we lower bound $\lambda_{\min}(A_{G_p})$ as follows:

$$\begin{aligned}\lambda_{\min}(A_{G_p}) &\ge \lambda_{\min}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij}\right) + \lambda_{\min}(p_0 J) + 1 \\ &\ge \lambda_{\min}\left(\sum_{ij\in\bar{E}} -p_{ij}Y_{ij}\right) + 1 \\ &\ge 0,\end{aligned}$$

where the first inequality follows from the claim above. The second inequality holds because $p_0 \ge 0$ and $J \succeq 0$. The final inequality follows from the Gershgorin circle theorem. Since the diagonal entries of A_{G_p} are nonzero, A_{G_p} is nonzero. This concludes the proof. □

References

[1] Claude Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 10:114, 1961.

[2] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, pages 51–229, 2006.

[3] Maria Chudnovsky, Gérard Cornuéjols, Xinming Liu, Paul Seymour, and Kristina Vušković. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005.

[4] Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2009.

[5] Martin Grötschel, László Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. In *North-Holland mathematics studies*, volume 88, pages 325–356. Elsevier, 1984.

[6] Avi Wigderson and László Lovász. Science Lives: László Lovász. Interview by Avi Wigderson https://www.youtube.com/watch?v=ikOUHnrNaxA. YouTube (uploaded by Simons Foundation), April 2015. Accessed: 2025-08-24.

[7] Toufik Mansour. Interview with Maria Chudnovsky. *Enumerative Combinatorics and Applications*, 1(2), 2021. Interview #S3I4.

[8] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer Science & Business Media, 2012.

[9] Nicolas Trotignon. Perfect graphs: a survey. *arXiv preprint arXiv:1301.5149*, 2013.

[10] Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. On the maximum weight independent set problem in graphs without induced cycles of length at least five. *SIAM Journal on Discrete Mathematics*, 34(2):1472–1483, 2020.

[11] Tara Abrishami, Maria Chudnovsky, Cemil Dibek, and Kristina Vušković. Submodular functions and perfect graphs. *Mathematics of Operations Research*, 50(1):189–208, 2025.

[12] Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1):21–25, 1991.

[13] Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

[14] Wen-Lian Hsu. How to color claw-free perfect graphs. In *North-Holland Mathematics Studies*, volume 59, pages 189–197. Elsevier, 1981.

[15] Irena Penev. Coloring bull-free perfect graphs. *SIAM Journal on Discrete Mathematics*, 26(3):1281–1309, 2012.

[16] Maria Chudnovsky, Nicolas Trotignon, Théophile Trunck, and Kristina Vušković. Coloring perfect graphs with no balanced skew-partitions. *Journal of Combinatorial Theory, Series B*, 115:26–65, 2015.

[17] Friedrich Eisenbrand, Stefan Funke, Naveen Garg, and Jochen Könemann. A combinatorial algorithm for computing a maximum independent set in a t-perfect graph. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 12, pages 517–522, 2003.

[18] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[19] Satoru Iwata. A fully combinatorial algorithm for submodular function minimization. *Journal of Combinatorial Theory, Series B*, 84(2):203–212, 2002.

[20] Monique Laurent and Frank Vallentin. Semidefinite optimization. *Lecture Notes, available at https://www.mi.uni-koeln.de/opt/wp-content/uploads/2015/10/laurent_vallentin_sdo_2012_05.pdf*, 2012.

[21] László Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.

[22] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[23] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[24] László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

[25] Mario Szegedy. A note on the θ number of Lovász and the generalized Delsarte bound. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 36–39. IEEE, 1994.

[26] Igor Dukanovic and Franz Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. *Mathematical Programming*, 109(2):345–365, 2007.

[27] Arkadi Semenovič Nemirovski and David Borisovich Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, 1983.

[28] Sébastien Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.