Elimination Graphs*

Yuli Ye and Allan Borodin

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
{y3ye,bor}@cs.toronto.edu

Abstract. In this paper, we study graphs with inductive neighborhood properties. Let $\mathcal{P}$ be a graph property, a graph $G = (V, E)$ with n vertices is said to have an inductive neighborhood property with respect to $\mathcal{P}$ if there is an ordering of vertices $v_1, \ldots, v_n$ such that the property $\mathcal{P}$ holds on the induced subgraph $G[N(v_i) \cap V_i]$, where $N(v_i)$ is the neighborhood of v_i and $V_i = \{v_i, \ldots, v_n\}$. It turns out if we take $\mathcal{P}$ as a graph with maximum independent set size no greater than k, then this definition gives a natural generalization of both chordal graphs and $(k+1)$-claw-free graphs. We refer to such graphs as inductive k-independent graphs. We study properties of such families of graphs, and we show that several natural classes of graphs are inductive k-independent for small k. In particular, any intersection graph of translates of a convex object in a two dimensional plane is an inductive 3-independent graph; furthermore, any planar graph is an inductive 3-independent graph. For any fixed constant k, we develop simple, polynomial time approximation algorithms for inductive k-independent graphs with respect to several well-studied NP-complete problems. Our generalized formulation unifies and extends several previously known results.

1 Introduction

The main challenge of studying many graph theoretical problems is the size and variety of the graph itself. Large graphs arising from real applications can easily have millions of vertices, and for graphs with n vertices, there are $2^{\binom{n}{2}}$ different simple graphs. When facing graph optimization problems that are hard to solve or approximate, a common approach is to study restricted families of graphs; this often involves giving a specific graph parameter to restrict the graphs that are allowed. For any parameter, we give three criteria (GTR):

1. **Generality:** Be as rich and as general as possible, possibly contains interesting known families with relatively "small" parameters.

2. **Tractability:** For input instances with "small" parameters, it admits simple and relatively efficient algorithms which produce optimal or good approximation solutions for some hard problems.

* This research is supported by the Natural Sciences and Engineering Research Council of Canada and the University of Toronto, Department of Computer Science. A preliminary version of this paper appears in ICALP 2009, Rhodes, Greece.

3. **Recognizability:** Allows efficient recognition algorithms at least for instances with "small" parameters; note that this is not always the case, for example, recognizing graphs with chromatic number three is NP-hard.

One natural way of restricting a graph is to look at local neighborhoods of a given graph. Let $G = (V, E)$ be a graph of n vertices and m edges. If $X \subseteq V$, the subgraph of G induced by X is denoted by $G[X]$. For a particular vertex $v_i \in V$, let $d(v_i)$ denote its degree and $N(v_i)$ denote the neighborhood of v_i, i.e., the set of neighbors of v_i, excluding v_i. Given an ordering of vertices $v_1, v_2, \ldots, v_n$, we use V_i to denote the set of vertices that appear after v_{i-1} in the ordering, *i.e.*, $V_i = \{v_i, \ldots, v_n\}$.

Definition 1. *Let $\mathcal{P}$ be a graph property. A graph has a* **universal neighborhood property** *with respect to $\mathcal{P}$ if for all vertices $v_1, v_2, \ldots, v_n$, $\mathcal{P}$ holds on $G[N(v_i)]$. The set of all graphs satisfying such a neighborhood property is denoted as $\hat{G}(\mathcal{P})$. A graph has an* **inductive neighborhood property** *with respect to $\mathcal{P}$ if there exists an ordering of vertices $v_1, v_2, \ldots, v_n$ such that for any v_i, $1 \leq i \leq n$, $\mathcal{P}$ holds on $G[N(v_i) \cap V_i]$. The set of all graphs satisfying such an inductive neighborhood property is denoted as $\tilde{G}(\mathcal{P})$. Such an ordering of vertices is called an elimination ordering with respect to the property $\mathcal{P}$.*

In fact, similar ordering-based concepts exist in the literature, especially with regard to graph coloring; for example, graph degeneracy defined in [32], and equivalently, degree inductiveness in [28]. Most relevant to this paper, Hochbaum [29] and Halldórsson [28] studied approximation algorithms for the maximum independent set problem, and they bounded the approximation ratios in terms of the degree inductiveness. In a recent paper [33], Kako et al. introduced a weighted measure of inductiveness in order to study the weighted case of the maximum independent set problem. Jamison and Mulder [31] also studied an extension of chordal graphs by considering the inductive neighborhood property defined by the vertex clique cover number. Similar concepts are recently explored by Kammer et al. in [34].

Note that if the property $\mathcal{P}$ is closed on induced subgraphs, then $\hat{G}(\mathcal{P})$ is a subfamily of $\tilde{G}(\mathcal{P})$, and both $\hat{G}(\mathcal{P})$ and $\tilde{G}(\mathcal{P})$ can be recognized in polynomial time provided that the property $\mathcal{P}$ can be tested in polynomial time. In this paper, we focus on graphs with their universal and inductive neighborhoods satisfying the following three graph properties:

1. $|V|_k$: the size of the vertex set is no more than k.
2. VCC_k: the size of the minimum (vertex) clique cover is no more than k.
3. IS_k: the size of the maximum independent set is no more than k.

Clearly $\hat{G}(\mathcal{P}) \subset \tilde{G}(\mathcal{P})$ for the above three properties we study. Since $|V|_k \Rightarrow VCC_k \Rightarrow IS_k$, we have

$$\hat{G}(|V|_k) \subseteq \hat{G}(VCC_k) \subseteq \hat{G}(IS_k).$$

Similarly, we have

$$\tilde{G}(|V|_k) \subseteq \tilde{G}(VCC_k) \subseteq \tilde{G}(IS_k).$$

It is not difficult to construct examples to show that all inclusions are proper. A clique of size $k+1$ is in $\tilde{G}(|V|_k)$ and $\tilde{G}(VCC_1)$, but is not in $\tilde{G}(|V|_{k-1})$. This separates $\tilde{G}(|V|_k)$ and $\tilde{G}(VCC_k)$. In order to separate $\tilde{G}(VCC_k)$ and $\tilde{G}(IS_k)$, we construct the

following example. Given two cycles of length $2k+1$, we connect every vertex in the first cycle to every vertex in the second cycle. It is not hard to see that the graph is in $\tilde{G}(VCC_{k+1})$ and $\tilde{G}(IS_k)$, but it is not in $\tilde{G}(VCC_k)$.

The graph class of interest in this paper is the class of $\tilde{G}(IS_k)$. First, we give a formal definition of $\tilde{G}(IS_k)$ and illustrate its connection to chordal graphs. We use $\alpha(G)$ to denote the size of a maximum independent set of G. A graph is **chordal** if it does not contain any induced cycle of size greater than three. An alternative characterization of chordal graphs is via a perfect elimination ordering.

Definition 2. *A* **perfect elimination ordering** *is an ordering of vertices $v_1, \ldots, v_n$ such that for any v_i, $1 \leq i \leq n$, $\alpha(G[N(v_i) \cap V_i]) = 1$.*

A natural extension to this perfect elimination ordering is to relax the size of the maximum independent set. Surprisingly, this extension seems to have only been relatively recently proposed in Akcoglu et al. [2] and not studied subsequently.

Definition 3. *A* **k-independence ordering** *is an ordering of vertices v_1, v_2, …, v_n such that for any v_i, $1 \leq i \leq n$, $\alpha(G[N(v_i) \cap V_i]) \leq k$. The minimum of such k over all orderings is called the* **inductive independence number** [1]*, which we denote as $\lambda(G)$.*

This extension of a perfect elimination ordering leads to a natural generalization of chordal graphs.

Definition 4. *A graph G is* **inductive k-independent** *if $\lambda(G) \leq k$, i.e., $G \in \tilde{G}(IS_k)$.*

For several natural classes of graphs, the inductive independence number is bounded by a small constant.

- **Chordal graphs:** since a chordal graph admits a perfect elimination ordering, chordal graphs are inductive 1-independent graphs. All sub-classes of chordal graphs are then clearly inductive 1-independent graphs; for example, interval graphs and trees.
- **Graphs with a bounded average degree on every induced subgraph:** it is not hard to see that $\lambda(G)$ is bounded above by the maximum average degree over all induced subgraphs of G, though this bound is usually not tight.
- **Claw-free graphs:** a graph is $(k+1)$-claw-free if every vertex has most k independent neighbors. If a graph is $(k+1)$-claw-free, then it is an inductive k-independent graph. For example, line graphs are inductive 2-independent graphs. Note that the converse is not always true since for example, a k-ary tree is not k-claw-free, but it is an inductive 1-independent graph.
- **Graphs with constant tree-width:** since the tree-width of G can be viewed as the smallest k such that G is a partial k-tree, it is not hard to see graphs with tree-width k are inductive k-independent graphs. It is in fact in the more restricted class of $\tilde{G}(|V|_k)$.
- **Intersection graphs of geometric objects:** disk graphs and unit disk graphs are inductive 5 and 3-independent graphs, respectively. Marathe et al. [39] show that simple heuristics achieve good approximations for various problems for unit disk graphs. We extend most of their results in this paper, and explain a connection between the unit disk graphs and planar graphs as observed in their paper.

[1] Akcoglu et al. [2] refer to this as the directed local independence number.

Akcoglu et al. [2] show that the (weighted) maximum indepedent set (MIS) problem has a simple k-approximation algorithm for any inductive k-independent graph. We call attention to two interesting examples.

- **The interval scheduling problem (ISP) and the (discrete) job interval scheduling problem (JISP):** For a given set of (weighted) intervals on the real line, the goal is trying to schedule a set of intervals of maximum size (or total weight in the weighted case) without any overlapping. There are simple algorithms that solve ISP optimally in both the unweighted and weighted cases. This is not a surprise since if we order intervals according to non-decreasing finishing times, then it is a perfect elimination ordering and the underlying intersection graph of ISP is a chordal graph. The job interval scheduling problem is an extension of ISP and has been extensively studied in the literature, for example, see [6][17][45]. In JISP, each interval belongs to a job. A job can be scheduled onto one and only one of its intervals. The objective is to find the maximum number (or total weight in the weighted case) of jobs that can be scheduled without conflicts. Using the same ordering of intervals by finishing time, it is easy to see that the intersection graph for the JISP problem is an inductive 2-independent graph.
- **The axis parallel rectangles problem:** This problem is studied by Berman and DasGupta in [12][13] and is motivated by applications to non-overlapping local alignment problems in computational molecular biology. The input is a set of axis parallel rectangles such that, for each axis, the projection of a rectangle on this axis does not enclose that of another. The goal is to select a subset of independent (non-overlapping projection on both axes) rectangles with maximum cardinality (or total weight in the weighted case). It is not hard to see that sorting the rectangles by (say) their rightmost x-coordinate yields a 3-independence ordering; hence the underlying graph is an inductive 3-independent graph. This also extends to D dimensions, where the underlying graph is an inductive $(2D-1)$-independent graph.

Both of the JISP and local alignment problems are MAX SNP-hard [41][45], although the current inapproximations are very weak. The existence of local ratio approximation algorithms in Bar-Noy et al. [5] and Berman and DasGupta in [12][13] for the above two problems was our initial motivation for investigating how the intersection graph structure underlies the success of those algorithms. In fact, such "elimination structure" occurs in many natural graph classes and extends in greater generality as we shall see.

However, not every graph has a small inductive independence number. For example, the complete bipartite graph $K_{n,n}$ has an inductive independence number n. Some examples of bipartite inductive k-independent graphs with small k are shown in Fig. 1.

The rest of the paper is organized as follows. In Section 2, we give some general bounds on the inductive independence number. In Section 3, we show several natural classes of inductive k-independent graphs. We study approximation algorithms for inductive k-independent graphs with respect to several well-studied NP-complete problems in Section 4 and we briefly discuss the class of $\tilde{G}(VCC_k)$ in Section 5. Section 6 concludes the paper and suggests a few open questions.

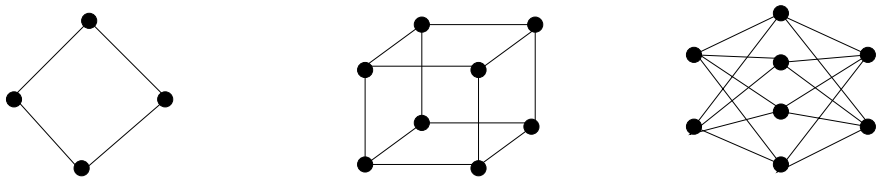


Fig. 1. Examples of inductive k-independent graphs for $k = 2, 3, 4$.

2 Properties of Inductive k-Independent Graphs

We have the following basic lemma.

Lemma 1. *Any induced subgraph of an inductive k-independent graph is an inductive k-independent graph.*

Proof. We prove this by contradiction. Suppose $G = (V, E)$ is an inductive k-independent graph but some induced subgraph G_1 of G is not an inductive k-independent graph, *i.e.*, for any vertex v in the vertex set of G_1, $\alpha(G_1[N(v)]) > k$. Let $v_1, \ldots, v_n$ be the k-independence ordering that G admits and $X \subseteq V$ be the vertex set that induces G_1. Let v_i be the first vertex in X that appears in the ordering and $G_2 = G[V_i]$. It is clear that the graph $G_1[N(v_i)]$ is an induced subgraph of $G_2[N(v_i)]$ since G_1 is an induced subgraph of G_2. Therefore, any independent set of $G_1[N(v_i)]$ is an independent set of $G_2[N(v_i)]$. Since $\alpha(G_1[N(v_i)]) > k$, we have $\alpha(G_2[N(v_i)]) > k$, which contradicts the fact that $v_1, \ldots, v_n$ is a k-independence ordering. Therefore any induced subgraph of G is also an inductive k-independent graph. □

Lemma 1 ensures that we can test if a graph is an inductive k-independent graph by repeatedly removing a vertex whose neighbors in the remaining graph have independent set size at most k, until there is no vertex remaining or for every vertex v in the remaining graph G, $\alpha(G[N(v)]) > k$. This k-elimination process, if successful (*i.e.*, no vertices remain at the end of the process), also constructs a k-independence ordering. Note that at each step, we can check for each vertex v to see if $\alpha(G[N(v)]) > k$ in $O(k^2 n^{k+1})$ time by enumerating all subsets of size $k+1$. Therefore this k-elimination process terminates in $O(k^2 n^{k+3})$ time. By the observation of Itai and Rodeh [30], and results in [22], we can improve the time complexities of a 2-elimination process to $O(n^{4.376})$, a 3-elimination process to $O(n^{5.334})$ and a 4-elimination process to $O(n^{6.220})$. The above algorithms only use linear space in the size of the graph. If we allow the algorithm to use $O(n^{k+2})$ space, we can further improve the time complexity of the k-elimination process.

Proposition 1. *An inductive k-independent graph can be recognized in $O(k^2 n^{k+2})$ time, and a k-independence ordering of an inductive k-independent graph can be constructed in $O(k^2 n^{k+2})$ time.*

Proof. Given a graph G we build a bipartite graph $G^* = (A, B)$ in the following way. We construct a subset-node (a node in A) for each of the subsets of size $k+1$ in G and a vertex-node (a node in B) for each vertex in G. We connect a vertex-node to a

subset-node with a red edge if the vertex in the vertex-node is adjacent to all vertices in the subset-node and the vertices in the subset form an independent set. We connect a vertex-node to a subset-node with a black edge if the vertex in the vertex-node is one of the vertices in the subset-node. Constructing such a graph G^* takes $O(k^2n^{k+2})$ time and $O(n^{k+2})$ space. Once G^* is constructed, we look for a vertex-node in B that is not incident to any red edge. The vertex in such a vertex-node is then the next vertex in the ordering. We then delete such a vertex-node in B and all its neighbors in A together with all incident (black and red) edges, and continue. If finally there is no node remaining in G^*, then we have constructed an inductive k-independent ordering, otherwise at some point of time, every vertex-node in B has at least one red edge and we conclude that G is not an inductive k-independent graph. □

Note that the maximum independent set problem is W[1]-complete when parameterized by the size of the independent set size k, see [19]. By a reduction from the maximum independent set problem, finding the inductive independence number is also complete for W[1], hence it is unlikely to have a fixed parameter tractable solution. But this does not exclude the possibility to improve the current complexity bound for a small fixed parameter k. It is interesting to note that recognizing a chordal graph, *i.e.*, an inductive 1-independent graph, can be done in linear time using LBFS or MCS, while our generic algorithm runs in time $O(n^3)$. It might be possible to improve on the bounds above by using different techniques. Note that we are mostly interested in inductive k-independent graphs with small k's, because in practice, we either know a-priori that a graph is a inductive k-independent graph with some small constant k, or we want to test whether or not it is the case. In many specific cases like JISP and non-overlapping local alignment, the complexity of computing a k-independence ordering can be reduced to $O(n \log n)$.

As mentioned in Section 1, the maximum average degree on every induced subgraph provides a trivial bound on the inductive independence number. Here we give bounds in terms of the number of vertices and edges in a graph.

Theorem 1. *A graph G with n vertices and m edges has inductive independence number no more than* $\min\{\lfloor \frac{n}{2} \rfloor, \lfloor \sqrt{m} \rfloor, \lfloor \frac{\sqrt{1+4[\binom{n}{2}-m]}+1}{2} \rfloor\}$.

Proof. If $\lambda(G)$ is the inductive independence number of G, then we can find an induced subgraph such that every vertex has at least $\lambda(G)$ independent neighbors and moreover, for any such vertex v, any of its neighbors, say u, must again have at least $\lambda(G)$ independent neighbors. Furthermore, these two independent neighbor sets of u and v are disjoint. Therefore, the total number of vertices is at least $2\lambda(G)$, the total number of edges is at least $\lambda(G)^2$ and the total number of missed edges is at least $2\binom{\lambda(G)}{2}$. Therefore, a graph G with n vertices and m edges has inductive independence number no more than

$$\min\{\lfloor \frac{n}{2} \rfloor, \lfloor \sqrt{m} \rfloor, \lfloor \frac{\sqrt{1+4[\binom{n}{2}-m]}+1}{2} \rfloor\}.$$

□

3 Natural Classes of Inductive k-Independent Graphs

In Section 1, we saw several known examples of inductive k-independent graphs. In this section, we show two more natural classes of graphs that fit our definition.

3.1 Translates of a Convex Object

Given a set of translates of a convex object in the two dimensional plane. We consider the intersection graph of those translates, *i.e.*, each translate is represented by a vertex; two vertices are adjacent if two associated translates are overlapping. One special case of such graphs are unit disk graphs, for which a robust PTAS for weighted maximum independent set is known [44]. In [23], Erlebach, Jansen and Seidel consider a more general case of geometric intersection graphs (including disk graphs) and give PTASs for weighted maximum independent set and weighted minimum vertex cover based on a sophisticated use of the *shifting strategy* [4]. The running time of both algorithms is $n^{O(1/\epsilon^2)}$ for achieving approximation ratio $1+\epsilon$. However, both algorithms require a geometric representation as the part of the input. Kim, Kostochka and Nakprasit [36] proved that any intersection graph of those translates with clique number k are $(3k-3)$-degenerate; i.e., $\tilde{G}(|V|_{3k-3})$ in our notation. Within that proof, they showed the neighborhood of the topmost object can be covered by three cliques. This immediately implies the following theorem.

Theorem 2. *[36] The intersection graph of translates of a convex object in the two dimensional plane is in* $\tilde{G}(VCC_3)$.

Since $\tilde{G}(VCC_3) \subseteq \tilde{G}(IS_3)$, the following corollary is immediate.

Corollary 1. *The intersection graph of translates of a convex object in the two dimensional plane is a inductive* 3*-independent graph.*

Please refer to the Appendix for a geometric alternative proof of Corollary 1. The bound in Corollary 1 is in fact tight; see Fig. 2.

Corollary 2. *The intersection graph of translates of a convex object in the two dimensional plane is a* 6*-claw free graph.*

Corollary 3. *The intersection graph of convex objects with different sizes (same shape and orientation) is an inductive* 5*-independent graph.*

Proofs for Corollaries 2 and 3 can be found in the Appendix. It follows immediately by Corollary 3 that disk graphs and unit disk graphs are inductive 5-independent and 3-independent graphs respectively. We conjecture that Corollary 1 extends to higher dimensions as follows:

Conjecture 1. The intersection graph of translates of a convex object in an D dimensional space is a inductive $(2D-1)$-independent graph.

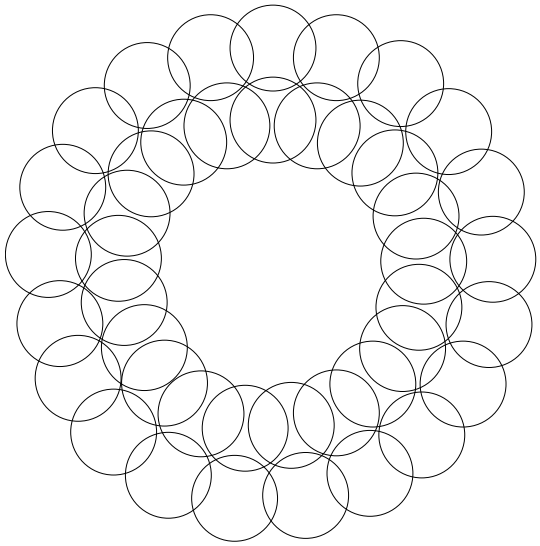


Fig. 2. An inductive 3-independent graph induced by the intersection of unit disks.

3.2 Planar Graphs

Planar graphs have been well-studied in the literature not only because of their numerous applications but also due to the existence of many non-trivial results. For many NP-complete problems, there exist PTASs when the graph is planar [4]. In this section, we present a nice property of planar graphs in terms of its inductive independence number. We first show a result for general genus $g \geq 1$.

Theorem 3. *A graph G with genus $g \geq 1$ has inductive independence number no more than $g + 4$.*

Proof. Let G be a graph with genus $g \geq 1$ and n be the number of vertices in G, then every induced subgraph of G has genus no more than g. We prove the theorem by induction on n. If $n < 12$ then by Theorem 1, we have $\lambda(G) \leq 5 \leq g + 4$. Now suppose the statement holds for $n < k$, we consider the case for $n = k$. If $k = 12$ then we have three cases:

1. If $\lambda(G) < 6$, then since $g \geq 1$, $\lambda(G) \leq g + 4$ clearly holds.
2. if $\lambda(G) = 6$, then it can be shown that G has to be $K_{6,6}$, which has $g = 4$, therefore $\lambda(G) \leq g + 4$.
3. If $\lambda(G) > 6$, this is impossible by Theorem 1.

If $k > 12$ then by Euler-Poincaré Theorem, the average degree of G is at most $6 + \frac{12(g-1)}{k} < g + 5$. That implies there exists a vertex v with degree $\leq g + 4$. By the inductive hypothesis, $\lambda(G[V - \{v\}]) \leq g + 4$, and thus $\lambda(G) \leq g + 4$. Therefore, a graph G with genus $g \geq 1$ has inductive independence number no more than $g+4$. □

The proof of Theorem 3 only relies on the average degree constraint from the Euler-Poincaré Theorem, and we believe that the following conjecture holds based on the fact that a graph with n vertices has the largest possible inductive independence number when it is a complete bipartite graph with equal bipartition.

Conjecture 2. The inductive independence number for a graph G with genus $g \geq 1$ is $O(\sqrt{g})$.

For planar graphs, since the average degree is always less than 6, any planar graph is an inductive 5-independent graph, but this is not tight.

Theorem 4. *Any planar graph is in* $\tilde{G}(VCC_3)$.

Proof. Let $\beta(G)$ denote the size of a minimum vertex clique cover of G. Let G^* be a minimum counter example, so for any vertex v in G^*, we have $\beta(G^*[N(v)]) > 3$. We look at a planar embedding of G^*, for a specific vertex v, depending on its degree, there are three cases:

1. If $d(v) = 4$, then since $\beta(G^*[N(v)]) > 3$, none of the neighbors are adjacent. Therefore, for each edge e incident to v, the face to its left has at least four edges as its boundary and so does the face to its right. Since each edge is counted twice, e contributes at most $\frac{1}{8}$ to the left face and $\frac{1}{8}$ to the right face; so the total face contribution of e is at most $\frac{1}{4}$. The edge contribution of e is $\frac{1}{2}$ due to double counting. Therefore, for such a vertex v, the total face contributions from all v's edges is at most 1 and the total edge contributions from all its edges is 2. We assume there are x such vertices.
2. If $d(v) = 5$, then since $\beta(G^*[N(v)]) > 3$, there are four cases:
   (a) If none of the neighbors are adjacent; see Fig. 3(a). Using a similar argument,

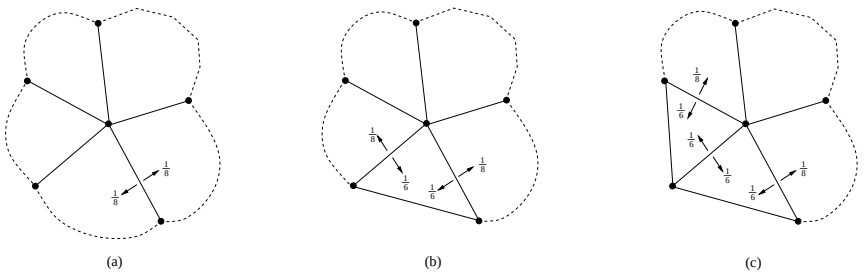


Fig. 3. Face contributions of edges for $d(v) = 5$.

we can show that for such a vertex v, the total face contribution from all its edges is at most $\frac{5}{4}$ and the total edge contributions from all its edges is $\frac{5}{2}$. We assume there are y_1 such vertices.

   (b) If exactly two neighbors are adjacent; see Fig. 3(b). Exactly one face associated with the vertex v is a triangle. We see that for such a vertex v, the total face contribution from all its edges is at most $\frac{4}{3}$ and the total edge contributions from all its edges is $\frac{5}{2}$. We assume there are y_2 such vertices.

(c) If there exists one neighbor adjacent to two other neighbors; see Fig. 3(c). Exactly two adjacent faces associated with the vertex v are triangles. We see that for such a vertex v, the total face contributions from all its edges is at most $\frac{17}{12}$ and the total edge contributions from all its edges is $\frac{5}{2}$. We assume there are y_3 such vertices.

(d) If there exists one neighbor adjacent to three or more other neighbors of v, it reduces to the case (c) above since these edges would still only create two triangles due to the fact that $\beta(G^*[N(v)]) > 3$.

3. If $d(v) > 5$, then we can conclude, for such a vertex v, the total face contributions from all its edges is at most $\frac{1}{3}d(v)$ and the total edge contributions from all its edges is $\frac{1}{2}d(v)$. We assume there are z such vertices, and we label them $v_1, \ldots, v_z$.

Now by summing up for all vertices, we get the total number of edges:

$$|E| = 2x + \frac{5}{2}(y_1 + y_2 + y_3) + \frac{1}{2}\sum_{i=1}^{z} d(v_i).$$

The total number of faces:

$$|F| \leq x + \frac{5}{4}y_1 + \frac{4}{3}y_2 + \frac{17}{12}y_3 + \frac{1}{3}\sum_{i=1}^{z} d(v_i).$$

The total number of vertices:

$$|V| = x + y_1 + y_2 + y_3 + z.$$

Therefore, the Euler characteristic:

$$|V| - |E| + |F| \leq z - \frac{1}{4}y_1 - \frac{1}{6}y_2 - \frac{1}{12}y_3 - \frac{1}{6}\sum_{i=1}^{z} d(v_i) \leq 0.$$

This contradicts the fact that G is planar. Therefore any planar graph is is in $\tilde{G}(VCC_3)$. □

Corollary 4. *Any planar graph is an inductive* 3*-independent graph.*

4 Algorithmic Aspects of Inductive k-Independent Graphs

In this section, we show several algorithmic results for inductive k-independent graphs. We do not explicitly state the time complexity for those algorithms (with the exception of the vertex cover problem) since they depend on the complexity for constructing a k-independence ordering. But by Proposition 1, all algorithms discussed here run in polynomial time when k is a fixed constant.

It is well known that many NP-complete problems can be solved optimally when restricted to chordal graphs [26][46]. We show that if we restrict the graph to be an inductive k-independent graph, we get simple approximation algorithms for several NP-complete problems. In fact, the structure of inductive k-independent graphs gives a unified treatment for many previous results. Note that for both minimization and maximization problems, we always consider approximation ratios to be greater or equal to one.

4.1 Weighted Maximum Independent Set

The maximum independent set problem is NP-complete for general graphs and for graphs with maximum degree Δ it is NP-hard to approximate within Δ^ϵ for some $\epsilon > 0$ even for the unweighted case [3], but polynomial time solvable for chordal graphs. Akcoglu et al. [2] show that the local ratio technique ([7][11]) achieves a k-approximation for the weighted maximum independent set on inductive k-independent graphs. The local ratio technique is usually described as a recursive algorithm. As in Berman and DasGupta [11], we view it as an iterative algorithm, modeled as stack algorithms in [14].

The Stack Algorithm for packing problems

Push Phase:

while V is not empty
    select the next data item v according to some rule[2]
    decide to either push v onto the stack or discard it
end while

Pop Phase:

while the stack is not empty
    pop the data item v from the stack
    accept v if it is feasible with respect to current solution
    otherwise discard v
end while

When restricting to the MIS graph problem, "feasibility" simply means independence in the above algorithm. For some graph classes such as those defined by the JISP and axis paralell rectangles problems mentioned in section 1, the orderings satisfy the locally defined orderings in priority algorithms [15] and hence we obtain greedy (in the unweighted case) or greedy-like stack algorithms. For completeness, we include a proof of the weighted MIS result in this subsection. We study a natural generalization of the MIS problem in the next subsection from which the MIS result will follow.

Theorem 5. *[2] For all fixed constant $k \geq 1$, there is a polynomial time stack algorithm that achieves a k-approximation for the weighted maximum independent set if G is an inductive k-independent graph. In the unweighted case, the pop phase is not needed.*

Proof. The stack algorithm in this case always selects a vertex v according to the inductive k-independent ordering. It then calculates the updated weight of that vertex $\bar{w}(v)$ to be the weight of that vertex minus the updated total weight of its neighboring vertices in the current stack. If the updated weight is positive, we push that vertex onto the stack with its updated weight; otherwise, reject that vertex.

[2] These rules are often "local priority rules" as defined in [14] but here we only require that the rule is polynomial time computable.

Let $v_1, v_2, \ldots, v_n$ be an inductive k-independent ordering, A be the output of the stack algorithm, and O be an optimal solution. Let S be the set of vertices in the stack at the end of the push phase, and S_i be the content of the stack before v_i is being pushed onto the stack. We first prove the following claim:

Claim. The stack algorithm achieves at least the total weight of the stack.

For a given vertex $v_i \in A$, let $w(v_i)$ denote the weight of v_i and $\bar{w}(v_i)$ denote the updated weight of v_i in the stack, we then have

$$w(v_i) = \bar{w}(v_i) + \sum_{v_j \in S_i \cap N(v_i)} \bar{w}(v_j).$$

If we sum up for all $v_i \in A$, we have

$$\sum_{v_i \in A} w(v_i) = \sum_{v_i \in A} \bar{w}(v_i) + \sum_{v_i \in A} \sum_{v_j \in S_i \cap N(v_i)} \bar{w}(v_j) \geq \sum_{v_t \in S} \bar{w}(v_t).$$

The second inequality holds because for any $v_t \in S$, we either have $v_t \in S_i \cap N(v_i)$ for some $v_i \in A$ or we have $v_t \in A$.

Now we show that the optimal solution achieves at most k times the weight of the stack. For a given vertex $v_i \in O$, we have

$$w(v_i) \leq \bar{w}(v_i) + \sum_{v_j \in S_i \cap N(v_i)} \bar{w}(v_j).$$

If we sum up for all $v_i \in O$, we have

$$\sum_{v_i \in O} w(v_i) \leq \sum_{v_i \in O} \bar{w}(v_i) + \sum_{v_i \in O} \sum_{v_j \in S_i \cap N(v_i)} \bar{w}(v_j) \leq k \sum_{v_t \in S} \bar{w}(v_t).$$

The second inequality holds because when we sum up for all $v_i \in O$, each of the terms $\bar{w}(v_t)$ for any vertex $v_t \in S$ can at most appear k times, since the ordering $v_1, v_2, \ldots, v_n$ is an inductive k-independent ordering. Therefore, we have

$$\sum_{v_i \in O} w(v_i) \leq k \sum_{v_i \in A} w(v_i).$$

□

4.2 Weighted Maximum m-Colorable Subgraph

The interval scheduling problem is often extended to scheduling on m machines. For identical machines, the graph-theoretic formulation of this problem leads to the following natural generalization of the MIS problem. Given a graph $G = (V, E)$ with a positive weight $w(v)$ for each vertex v, an m-colorable subgraph of G is an induced subgraph $G[V']$ on a subset V' of V such that $G[V']$ is m-colorable. A maximum m-colorable subgraph is an m-colorable subgraph with maximum number (or total weight in the weighted case) of the vertices. This problem has also been referred to as the (weighted) maximum m-partite induced subgraph problem [1] in some other

graph theory literature. For chordal graphs, the unweighted case of the problem is polynomial time solvable for any fixed m, but NP-complete otherwise [46]. In general, by a result in [6], the existence of a r-approximation for weighted maximum independent set problem always implies (using the greedy algorithm that repeatedly takes an r-approximate weighted maximum independent set in the remaining graph) an approximation algorithm with ratio $\frac{(mr)^m}{(mr)^m-(mr-1)^m}$ for the weighted maximum m-colorable subgraph problem, and this ratio is less than $r+1-\frac{1}{m}$, however the running time of the above approach is $O(m(|E|+|V|))$. In a recent paper, Chakaravarthy and Roy [16] showed that for chordal graphs the problem admits a 2-approximation for the weighted case using a simpler and more efficient algorithm. Here we strengthen their approximation result.

Theorem 6. *For all fixed constant $k \geq 1$, there is a polynomial time algorithm that achieves a $(k+1-\frac{1}{m})$-approximation for weighted maximum m-colorable subgraph if G is an inductive k-independent graph. The algorithm runs in time $O(\min\{|E|\log m+|V|, |E|+m|V|\})$.*

We first describe the algorithm. We use almost the same idea for the weighted maximum independent set as proven in the previous subsection except that we now use a stack S_c for each color class c. At each step i, the algorithm considers the vertex v_i in the k-independence ordering, and computes the updated weight

$$\bar{w}(v_i) = w(v_i) - \sum_{v_j \in S_c \cap N(v_i)} \bar{w}(v_j)$$

for each color class c. If $\bar{w}(v_i)$ is non-positive for every color class, then reject v_i without coloring it. Otherwise, find a color class with the largest $\bar{w}(v_i)$ value and assign v_i with that color. We next prove the following permutation lemma.

Let M be an m by m square matrix, and $\sigma \in \Sigma$ be a permutation of $\{1, 2, \ldots, m\}$. and σ_i be the i^{th} element in the permutation.

Lemma 2. *There exists a permutation σ such that*

$$\sum_i M_{i\sigma_i} \leq \frac{1}{m} \sum_{i,j} M_{ij}.$$

Proof. Suppose otherwise, then for each permutation σ we have

$$\sum_i M_{i\sigma_i} > \frac{1}{m} \sum_{i,j} M_{ij}.$$

We sum up for all $\sigma \in \Sigma$, since in total we have $m!$ permutations, we have

$$\sum_{\sigma \in \Sigma} \sum_i M_{i\sigma_i} > m! \cdot \frac{1}{m} \sum_{i,j} M_{ij}.$$

Since each M_{ij} is counted exactly $(m-1)!$ times on the left hand side, we have

$$(m-1)! \sum_{i,j} M_{ij} > (m-1)! \sum_{i,j} M_{ij},$$

which is a contradiction. □

Let $c_1, c_2, \ldots, c_m$ be the color classes and $S_1, S_2, \ldots, S_m$ be the sets of vertices that have been put onto the stacks at the end of the push phase. Let S be the union of all the stacks. Now we prove the theorem.

Proof. It is not hard to see that the algorithm achieves at least the total weight $W = \sum_{v_t \in S} \bar{w}(v_t)$ of all stacks. The goal is to show that the weight of the optimal solution will be at most $(k+1-\frac{1}{m}) \cdot W$. Let A be the output of the algorithm and O be the optimal solution. For each given vertex v_i in O, let c_{oi} be its color class in O, and c_{si} be its color class in S if it is accepted into one of the stacks. Let S^i_{oi} be the content of the stack in color class c_{oi} when the algorithm considers v_i, then we have three cases:

1. If v_i is rejected during the push phase of the algorithm then we have
$$w(v_i) \leq \sum_{v_j \in S^i_{oi} \cap N(v_i)} \bar{w}(v_j).$$
In this case, we can view that $w(v_i)$ is charged to all $\bar{w}(v_j)$ with $v_j \in S^i_{oi} \cap N(v_i)$, each of which appears in the same color class c_{oi} and is charged at most k times coming from the same color class.
2. If v_i is accepted into the same color class during the push phase of the algorithm then we have
$$w(v_i) = \bar{w}(v_i) + \sum_{v_j \in S^i_{oi} \cap N(v_i)} \bar{w}(v_j).$$
In this case, we can view that $w(v_i)$ is charged to $\bar{w}(v_i)$ and all $\bar{w}(v_j)$ with $v_j \in S^i_{oi} \cap N(v_i)$. Note that they all appear in the same color class c_{oi}; $\bar{w}(v_i)$ is charged at most once and each $\bar{w}(v_j)$ is charged at most k times coming from the same color class.
3. If v_i is accepted into a different color class during the push phase of the algorithm then we have
$$w(v_i) \leq \bar{w}(v_i) + \sum_{v_j \in S^i_{oi} \cap N(v_i)} \bar{w}(v_j).$$
In this case, we can view that $w(v_i)$ is charged to $\bar{w}(v_i)$ and all $\bar{w}(v_j)$ with $v_j \in S^i_{oi} \cap N(v_i)$. Note that each $\bar{w}(v_j)$ appears in the same color class c_{oi} and is charged at most k times coming from the same color class. However $\bar{w}(v_i)$ in this case is in a different color class c_{si} and is charged at most once coming from a different color class.

If we sum up for all $v_i \in O$, we have

$$\sum_{v_i \in O} w(v_i) \leq \sum_{v_i \in S \cap O \wedge c_{oi} \neq c_{si}} \bar{w}(v_i) + k \sum_{i=1}^{m} \sum_{v_t \in S_i} \bar{w}(v_t).$$

The inequality holds when we sum up for all $v_i \in O$, since the number of charges coming from the same color class can be at most k; the number of charges coming from a different color class can be at most one, which only appears when v_i is accepted into a stack of a different color class (comparing to the optimal) during the push phase of

the algorithm. Therefore we have the extra term $\sum_{v_i \in S \cap O \wedge c_{oi} \neq c_{si}} \bar{w}(v_i)$. Now if we can permute the color classes of the optimal solution so that for any $v_i \in S \cap O$, $c_{oi} = c_{si}$, then the term $\sum_{v_i \in S \cap O \wedge c_{oi} \neq c_{si}} \bar{w}(v_i)$ disappears and we achieve a k approximation. But it might be the case that no matter how we permute the color classes of the optimal solution, we always have some $v_i \in S \cap O$ with $c_{oi} \neq c_{si}$. We construct the weight matrix M in the following way. An *assignment* $c_i \rightarrow c_j$ is to assign the color class c_i of O to the color class c_j of S. A vertex is *misplaced* with respect to this assignment $c_i \rightarrow c_j$ if it is in $S \cap O$ and its color class is c_i in O, but is not c_j in S. We then let M_{ij} be total updated weight of misplaced vertices with respect to the assignment $c_i \rightarrow c_j$. Note that the total weight of the matrix is $(m-1)\sum_{v_i \in S \cap O} \bar{w}(v_i)$, and applying Lemma 2, there exists a permutation of the color class in O such that

$$\sum_{v_i \in S \cap O \wedge c_{oi} \neq c_{si}} \bar{w}(v_i) \leq \frac{m-1}{m} \sum_{v_i \in S \cap O} \bar{w}(v_i) \leq \frac{m-1}{m} \sum_{v_i \in A} w(v_i).$$

Therefore, we have

$$\sum_{v_i \in O} w(v_i) \leq \frac{m-1}{m} \sum_{v_i \in A} w(v_i) + k \sum_{i=1}^{m} \sum_{v_t \in S_i} \bar{w}(v_t) \leq (k + 1 - \frac{1}{m}) \sum_{v_i \in A} w(v_i).$$

Given a k-independence ordering, the running time of the above algorithm is dominated by the push phase and can be bounded by $O(\min\{|E| \log m + |V|, |E| + m|V|\})$. The first quantity is obtained as follows: for each vertex, we maintain a priority queue of its updated weights for all the color classes. An update occurs for each edge in the graph and the cost of such an update is $O(\log m)$. Therefore the running time is bounded by $O(|E| \log m + |V|)$. For the second quantity, at each step, we basically calculate the updated weighted of that vertex for all color classes, and then find the best color class to push that vertex onto the stack. Calculating the update weighted for all vertices costs time $O(|E|)$, and finding the best color class for each vertex costs time $O(m)$. Therefore the running time is bounded by $O(|E|+m|V|)$. □

4.3 Minimum Vertex Coloring

Minimum vertex coloring is a well-studied NP-hard problem, and is not approximable within $n^{1-\epsilon}$ for any fixed $\epsilon > 0$, unless ZPP=NP [24]. For chordal graphs, a greedy algorithm on the reverse ordering of any perfect elimination ordering gives an optimal coloring. For inductive k-independent graphs, the same greedy algorithm achieves a k-approximation.

Theorem 7. *For all fixed constant $k \geq 1$, there is a polynomial time algorithm that achieves a k-approximation for the minimal vertex coloring if G is an inductive k-independent graph.*

Proof. The algorithm just takes the reverse of a k-independence ordering, and assigns the minimal color number to each vertex. Let $v_1, v_2, \ldots, v_n$ be a k-independence ordering, and $V_i = \{v_i, \ldots, v_n\}$. We prove by induction that the algorithm achieves k-approximation for $G[V_i]$ for all i from n to 1. The base case is clear, since when

$i = n$, $G[V_n]$ is just a single vertex. Now we assume the statement holds for $i > t$, *i.e.*, the number of colors c_i used in the algorithm for $G[V_i]$ satisfies

$$c_i \leq k \cdot \chi(G[V_i]).$$

Now we consider $i = t$. There are three cases:

1. If $c_t = c_{t+1}$, then the statement holds trivially since

$$c_t = c_{t+1} \leq k \cdot \chi(G[V_{t+1}]) \leq k \cdot \chi(G[V_t]).$$

2. If $\chi(G[V_t]) = \chi(G[V_{t+1}]) + 1$, then the statement also holds trivially since

$$c_t \leq c_{t+1} + 1 \leq k \cdot \chi(G[V_{t+1}]) + 1 \leq k(\chi(G[V_{t+1}]) + 1) = k \cdot \chi(G[V_t]).$$

3. The only remaining case is when $c_t = c_{t+1} + 1$, and $\chi(G[V_t]) = \chi(G[V_{t+1}])$. Suppose $c_t > k \cdot \chi(G[V_t])$. Since we have to increase the color number in the algorithm, there exist c_{t+1} neighbors of v_t, each having a different color. These c_{t+1} neighbors together with v_t must be grouped into $\chi(G[V_t])$ color classes in the optimal coloring. Therefore at least one color class in the optimal coloring will have at least $\frac{c_{t+1}+1}{\chi(G[V_t])}$ vertices from the set $N(v_t) \cap V_t$. Since

$$\frac{c_{t+1}+1}{\chi(G[V_t])} = \frac{c_t}{\chi(G[V_t])} > k,$$

we have one color class containing more than k vertices from $N(v_t) \cap V_t$. This contradicts the fact that $v_1, v_2, \ldots, v_n$ is an inductive k-independent ordering.

This completes the induction. Therefore the algorithm achieves k-approximation for the minimal vertex coloring if G is an inductive k-independent graph. □

4.4 Minimum Vertex Cover

Minimum vertex cover is one of the most celebrated problems for approximation algorithms, because there exist simple 2-approximation algorithms, yet for general graphs no known algorithm[3] can achieve approximation ratio $2 - \epsilon$ for any fixed $\epsilon > 0$. In this section, we show a $(2 - \frac{1}{k})$-approximation algorithm for minimum vertex cover on inductive k-independent graphs. The algorithm shares the same spirit of the result of Bar-Yehuda and Even [8] for the $\frac{5}{3}$ approximation of vertex cover for planar graphs, and can be viewed as a generalization of that algorithm. Baker's PTAS algorithm [4] for minimum vertex cover on planar graphs depends on a planar embedding and would not be considered as a simple combinatorial algorithm.

Theorem 8. *There is a simple polynomial time algorithm that achieves a $(2 - \frac{1}{k})$-approximation for minimum vertex cover if G is an inductive k-independent graph. Furthermore, for triangle free, inductive k-independent graphs with $k > 1$, the algorithm is a greedy algorithm (in the sense of [15]).*

[3] In fact, modulo unique games conjecture, no polynomial-time algorithm can achieve that ratio; see [35].

Proof. If $k = 1$ then it is a chordal graph, and hence minimum vertex cover can be done optimally. Now assume $k > 1$; if the inductive k-independent graph G contains triangles, then since finding a triangle can be done in polynomial time, we can find a triangle, select all three vertices and delete all incident edges, hence reduce the problem to a smaller inductive k-independent graph. We can do this because covering three edges of the triangle requires at least two nodes, so this elimination can only improve the approximation ratio. Therefore we can assume, at a certain point of time, the remaining inductive k-independent graph G^* does not contain triangles. Let v be a vertex with the smallest degree in G^* and $N'(v)$ be the set of vertices, excluding v, adjacent to vertices in $N(v)$. Note that since there are no triangles, $N(v)$ is an independent set and $d(v) \leq k$. We first prove the following claim:

Claim. The size of the maximum matching between $N(v)$ and $N'(v)$ is at least $d(v)-1$.

Suppose the size of the maximum matching between $N(v)$ and $N'(v)$ is less than $d(v) - 1$, then there must exist some vertex $x \in N(v)$, which is not in the maximum matching, such that $d(x) < d(v) - 1 + 1 = d(v)$. This contradicts the fact that v has the minimum degree.

Therefore, we select a $d(v) - 1$ matching M between $N(v)$ and $N'(v)$, and let u be the vertex in $N(v)$ which is not selected in M, then $M \cup \{uv\}$ is a matching of size $d(v)$, depicted by the thick edges in Fig. 4. The algorithm then selects both end-

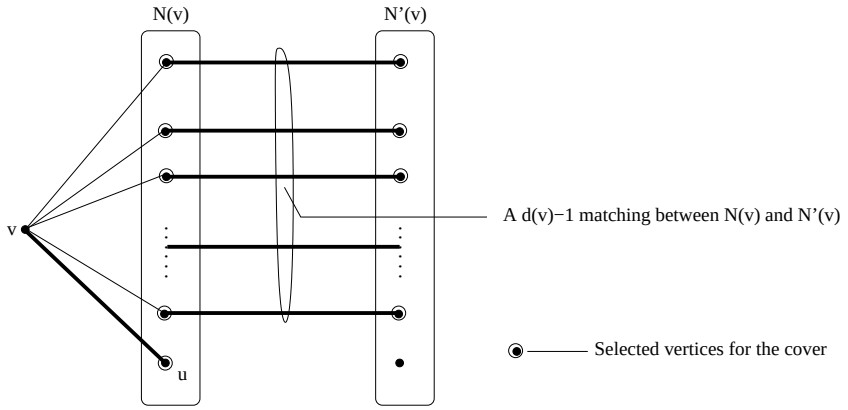


Fig. 4. A large local matching in $G[\{v\} \cup N(v) \cup N'(v)]$.

vertices of the matching M, plus u. This set C of $2d(v) - 1$ vertices covers $M \cup \{uv\}$, which requires at least $d(v)$ vertices to cover. At the same time, this covering C is also maximum in the sense that for any other covering C', where C' is a subset of the vertex set of matching $M \cup \{uv\}$, the set of edges covered by C' is always a subset of the set of edges covered by C. Therefore by taking C the algorithm can always do better locally. Since $d(v) \leq k$, the approximation ratio is at most $2 - \frac{1}{k}$. □

The running time of this algorithm is dominated by the time to remove all triangles which can be done in $n \times n$ matrix multiplication time $O(n^\omega) \approx O(n^{2.376})$; or in $O(nm)$

time for sparse graphs. We can further improve the ratio to $2 - \frac{2}{k+1}$ using a result of Hochbaum [29], which uses Nemhauser and Trotter's decomposition scheme [43]. This yields a $\frac{3}{2}$ approximation for planar graphs.

Theorem 9. (Hochbaum) *Let G be a weighted graph with n vertices and m edges. If it takes only s steps to color the vertices of G in c colors then it takes only $s + O(nm \log n)$ steps to find a cover whose weight is at most $2 - \frac{2}{c}$ times the weight of an optimal cover.*

Lemma 3. *For a triangle-free, inductive k-independent graph, a simple greedy algorithm can color its vertices with $k + 1$ colors.*

Proof. The greedy algorithm colors vertices on the reverse of a k-independence ordering. Since whenever the algorithm colors a vertex v, at most k neighbors of v are already colored, the algorithm uses at most $k + 1$ colors. □

Theorem 10. *For $k > 2$, there is a polynomial time algorithm that achieves a $(2 - \frac{2}{k+1})$-approximation for minimum vertex cover if G is an inductive k-independent graph.*

Proof. Note that if we remove all triangles from an inductive k-independent graph, the remaining graph is $(k + 1)$-colorable by Lemma 3. It then follows immediately by Theorem 9 that there is a polynomial time algorithm that achieves a $(2 - \frac{2}{k+1})$-approximation for minimum vertex cover if G is an inductive k-independent graph. The running time is $O(nm \log n)$. □

Theorem 11. *For $k > 2$, there is a polynomial time algorithm that achieves a $(2 - \frac{2}{k+1})$-approximation for weighted minimum vertex cover if G is an inductive k-independent graph.*

Proof. As observed by Zimny [47], this follows directly from the Local Ratio vertex cover algorithm of Bar Yehuda and Even [9]. Namely, for any triangle in the graph, the local ratio algorithm removes the vertex with minimum weight and reduces the weights of the other two vertices by that weight. The algorithm keeps doing that until the graph is triangle free. Then it applies Theorem 9 to get $(2 - \frac{2}{k+1})$-approximation. □

5 Graph Class of $\tilde{G}(VCC_k)$

The graph class $\tilde{G}(VCC_k)$ is a subclass of $\tilde{G}(IS_k)$. All natural examples of $\tilde{G}(IS_k)$ studied in this paper are in $\tilde{G}(VCC_k)$ for the same corresponding value k. It is an interesting question whether there is a natural graph class which is in $\tilde{G}(IS_k)$ but is not in $\tilde{G}(VCC_k)$ for the same k.

For $\tilde{G}(VCC_k)$, the weighted maximum clique and minimum vertex clique cover problem can be approximated with the ratio k if an elimination ordering with respect to the k-vertex-clique-cover is given. However, constructing such an elimination ordering, and hence testing membership in $\tilde{G}(VCC_k)$ for $k > 2$, is NP-hard.

The graph class $\tilde{G}(VCC_2)$ can be recognized in polynomial time, and it contains several interesting classes such as translates of a rectangle, JISP graphs and circular-arc graphs. Here, we give an optimal algorithm for weighted maximum clique and a 2-approximation algorithm for minimum vertex clique cover.

Theorem 12. *Given a graph in $\tilde{G}(VCC_2)$, there is a polynomial time algorithm that solves weighted maximum clique.*

Proof. Let $v_1, v_2, \ldots, v_n$ be an elimination ordering with respect to the 2-vertex-clique-cover. For each v_i, let $G_i = G[(N(v_i) \cup \{v_i\}) \cap V_i]$. Since G_i has vertex clique cover size 2, the complement of G_i is a bipartite graph. Since a weighted maximum independent set in a bipartite graph can be determined in polynomial time [25][27], weighted maximum clique in G_i can be computed in polynomial time. We compute weighted maximum clique for each G_i, and the largest one is weighted maximum clique for G. This is because for any weighted maximum clique C of G, there exist some vertex v_j in C that appears first in the ordering. Hence, when we compute the weighted maximum clique for G_j, we catch the weighted maximum clique C of G. □

Theorem 13. *Given a graph in $\tilde{G}(VCC_2)$, there is a polynomial time 2-approximation algorithm for minimum vertex clique cover.*

Proof. Let $v_1, v_2, \ldots, v_n$ be an elimination ordering with respect to the 2-vertex-clique-cover. We construct an independent set S by repeatedly taking a vertex according to this elimination ordering and removing all its neighbors. For each $v_i \in S$, let $G_i = G[(N(v_i) \cup \{v_i\}) \cap V_i]$. Since G_i has vertex clique cover size 2, we take both cliques to cover v_i, resulting in a clique cover of size $2|S|$. Since S is an independent set, the optimal clique cover has size at least $|S|$. Therefore, the algorithm achieves approximation ratio 2. □

6 Conclusion and Open Questions

We considered a generalization of chordal graphs due to Akcoglu et al. [2] based on a specific type of elimination ordering. We showed that several natural classes of graphs have a small inductive independence number, and give a unified approach for several optimization problems when such structure is present. Since the notion of independence naturally extends to hypergraphs, our results also extend to hypergraphs.

There are many open questions. The first and perhaps the most immediate issue is to improve the time complexity to recognize an inductive k-independent graph for small constants k. As we already mentioned, the current time complexity is unsatisfactory, and one can expect it to be improved.

Our second question is related to the intersection of graph classes. It is known [38] that the intersection of asteroidal triple-free graphs and chordal graphs is exactly interval graphs. One immediate question is what is the intersection of asteroidal triple-free graphs [18] and inductive k-independent graphs for $k \geq 2$? Do they present any interesting properties?

Our third question is on the algorithmic aspects of inductive k-independent graphs. We have studied the weighted maximum m-colorable subgraph, minimum vertex cover, and minimum vertex coloring problems. What can be said about other basic graph problems? Several other problems can be solved optimally in polynomial time for chordal graphs such as maximum clique and minimum clique cover. Can we $O(k)$-approximate such problems for inductive k-independent graphs? A particularly interesting problem is minimum independent dominating set. The unweighted case can be solved optimally in polynomial time for chordal graphs, but the weighted case

is NP-complete even for chordal graphs. Can we $O(k)$-approximate the independent dominating set problem for inductive k-independent graphs?

Last but not the least, we think there is a correspondence between algorithm paradigms and problem structures. We have seen multiple evidences for simple algorithms based on local decisions. The most notable one is the matroid [21] and greedoid [37] structures in correspondence to greedy algorithms, which have been studied extensively in the literature. As illustrated in this paper, various problems solved (or approximated) by the local ratio technique seem to be connected with the graphs having small sequential independence number. One other interesting class is d-claw-free graphs. In [10], Berman gives a $\frac{d}{2}$ approximation for maximum weight independent set in d-claw-free graphs. We also note for a given graph G, the weighted maximum matching problem can be viewed as the weighted maximum independent set problem for the line graph of G, which is 3-claw-free. Edmonds's weighted matching algorithm [20] has been extended to the weighted maximum independent set problem for 3-claw-free graphs [40][42]. These algorithms are local search based algorithms.

Acknowledgment We thank Derek Corneil for many helpful discussions on various graph classes. We also thank Bhaskar DasGupta, who pointed out a misstatement in an early version of the paper and Yuval Emek for providing us with an interesting planar graph example (icosahedron) with minimum degree five.

We also would like to thank anonymous referees who pointed out the existence of the previous literature [36] on intersection graphs of translates and provided many constructive comments which improved the paper.

References

1. L. Addario-Berry, W.S. Kennedy, A.D. King, Z. Li, and B. Reed. Finding the maximum-weight induced k-partite subgraph of an i-triangulated graph. *Discrete Applied Mathematics,* **158-7**, 765-770, 2010.
2. K. Akcoglu, J. Aspnes, B. DasGupta, and M-Y.Kao. Opportunity Cost Algorithms for Combinatorial Auctions. *Applied Optimization 74: Computational Methods in Decision-Making, Economics and Finance,* 455-479, 2002.
3. N. Alon, U. Feige, A. Wigderson and D. Zuckerman. Derandomized graph products. *Computational Complexity,* **5**, 60-75, 1995.
4. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery,* **41-1**, 153-180, 1994.
5. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber. A unified pproach to approximating resource allocation and scheduling. *JACM,* **48-5**, 1069-1090, 2001.
6. A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing,* **31-2**, 331-352, 2001.
7. R. Bar-Yehuda, A. Bendel, A. Freund, and D. Rawitz. Local ratio: A unied framework for approxmation algorithms in memoriam: Shimon Even 1935-2004. *Computing Surveys,* **36**, 422-463, 2004.
8. R. Bar-Yehuda and S. Even. On approximating a vertex cover for planar graphs. *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing,* 303-309, 1982.
9. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics,* **25**, 27-46, 1985.

10. P. Berman. A $\frac{d}{2}$ Approximation for Maximum Weight Independent Set in d-Claw Free Graphs. *Nord. J. Comput.,* **7-3**, 178-184, 2000
11. P. Berman and B. DasGupta. Improvements in throughout maximization for real-time scheduling. *STOC,* 680-687, 2000.
12. P. Berman and B. DasGupta. A simple approximation algorithm for nonoverlapping local alignments (Weighted Independent Sets of Axis Parallel Rectangles). *Biocomputing,* **1**, 129-138, 2002.
13. P. Berman, B. DasGupta and S. Muthukrishnan. Simple approximation algorithm for nonoverlapping local alignments. *SODA,* 677-678, 2002.
14. A. Borodin, D. Cashman and A. Magen. How well can primal-dual and local-ratio algorithms perform? *Lecture Notes in Computer Science,* **3580**, 943-955, 2005.
15. A. Borodin, M. Nielsen and C. Rackoff. (Incremental) priority algorithms. *SODA,* 752-761, 2002.
16. V. T. Chakaravarthy and S. Roy. Approximating maximum weight K-colorable subgraphs in chordal graphs. *Information Processing Letters,* to appear.
17. J. Chuzhoy, R. Ostrovsky and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research,* **31-4**, 730-738, 2006.
18. D. G. Corneil, S. Olariu and L. Stewart. Asteroidal triple-free graph. *SIAM J. Discrete Math.,* **10**, 399-430, 1997.
19. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. II. On completeness for W[1]. *Theoretical Computer Science,* **141-1-2**, 109131, 1995.
20. J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.,* **17**, 449-467, 1965.
21. J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming,* **1**, 127-136, 1971.
22. F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science,* **326-1-3**, 57-67, 2004.
23. T. Erlebach, K. Jansen, E. Seidel. Polynomial-Time Approximation Schemes for Geometric Intersection Graphs. *SIAM J. Comput.,* **34-6**, 1302-1323, 2005
24. U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. System Sci.,* **57**, 187-199, 1998.
25. C. M. H. de Figueiredo and F. Maffray. Optimizing bull-free perfect graphs. *SIAM J. Discrete Math.,* **18-2**, 226-240, 2004.
26. F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximun independent set of a chordal graph. *SIAM Journal on Computing,* **1-2**, 180-187, 1972.
27. M. Grötschel, L. Lovász and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica,* **1**, 169-197, 1981.
28. M. Halldórsson. Approximations of weighed independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications,* **4-1**, 1-16, 2000.
29. D. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Discrete Applied Mathematics,* **6**, 243-254, 1983.
30. A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal of Computing,* **7**, 413-423, 1978.
31. R. E. Jamison and H. M. Mulder. Tolerance intersection graphs on binary trees with constant tolerance 3. *Discrete Math.,* **215**, 115-131, 2000.
32. T. R. Jensen and B. Toft. Graph coloring problems. *Wiley-Interscience,* New York, ISBN 0-471-02865-7, 1995.
33. A. Kako, T. Ono, T. Hirata and M. Halldórsson. Approximation algorithms for the weighted independent set problem in sparse graphs. *Discrete Applied Mathematics,* **157-4**, 617-626, 2009.

34. F. Kammer, T. Tholey and H. Voepel. Approximation Algorithms for Intersection Graphs. *Technical Report,* **2009-6**, Institut für Informatik, Universität Augsburg.
35. S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *CCC,* 379-386, 2003.
36. S. J. Kim, A. Kostochka and K. Nakprasit. On the chromatic number of intersection graphs of convex sets in the plane. *The Electronic Journal of Combinatorics,* **11**, #R52, 2004.
37. B. Korte and L. Lovász. Mathematical structures underlying greedy algorithms. *Lecture Notes in Computer Science,* **117**, 205-209, 1981.
38. C. G. Lekkerkerker and J. C. Boland. Representation of a finite graph by a set of Intervals on the real line. *Fund. Math.,* **51**, 45-64, 1962.
39. M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi and D. J. Rosenkrantz. Simple Heuristics for Unit Disk Graphs. *Networks,* **25**, 59-68, 1995.
40. G. J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory,* **B-28**, 284-304, 1980.
41. H. Nagashima and K. Yamazaki. Hardness of approximation for non-overlapping local alignments. *Discrete Applied Mathematics,* **137-3**, 293-309, 2004.
42. D. Nakamura and A. Tamura. A revision of Minty's algorithm for finding a maximum weighted stable set of a claw-free graph. *Journal of the Operations Research Society of Japan,* **44-2**, 194-204, 2001.
43. G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Math. Programming,* **8**, 232-248, 1975.
44. T. Nieberg, J. Hurink, W. Kern. A Robust PTAS for Maximum Weight Independent Sets in Unit Disk Graphs. *WG,* 214-221, 2004.
45. F. C. R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling,* **2**, 215-227, 1999.
46. M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters,* **24-2**, 133-137, 1987.
47. B. Zimny. Personal communication, 2009.

7 Appendix

7.1 A Geometric Proof of Corollary 1

For completeness, we include a geometric, alternative proof of Corollary 1 here. We treat each translate as a closed convex set. For each translate S, we fixed an interior point to be the center of the object, denoted as c_S. Note that it does not have to be the geometric center of S, but it has to be identically located for each translate. The **conflicting region** $\psi(S, c_S)$ of S is then defined to be the set of points on the plane such that if we place the center of another translate S' at that point, $S \cap S' \neq \emptyset$; see Fig. 5. It is not hard to see that the conflicting region $\psi(S, c_S)$ is symmetric at the point c_S.

Lemma 4. *The conflicting region of a (convex) translate is convex.*

Proof. We prove this by contradiction. Let S be a translate with center c_S and $\psi(S, c_S)$ be its conflicting region. Suppose that $\psi(S, c_S)$ is not convex, then there exist three points c_{S_1}, c_{S_2} and c_{S_3} of the objects S_1, S_2 and S_3, such that c_{S_2} is on the line segment of $c_{S_1}c_{S_3}$, and $c_{S_1} \in \psi(S, c_S)$, $c_{S_3} \in \psi(S, c_S)$ but $c_{S_2} \notin \psi(S, c_S)$. Since $c_{S_1} \in \psi(S, c_S)$ and $c_{S_3} \in \psi(S, c_S)$, we know that there are two points p_1 and q_3 such that $p_1 \in S_1 \cap S$ and $q_3 \in S_3 \cap S$; see Fig. 6(a).

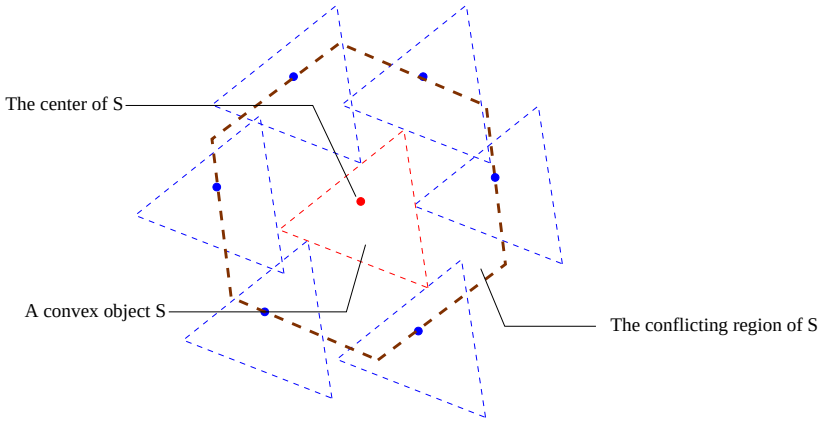


Fig. 5. A translate and its conflicting region.

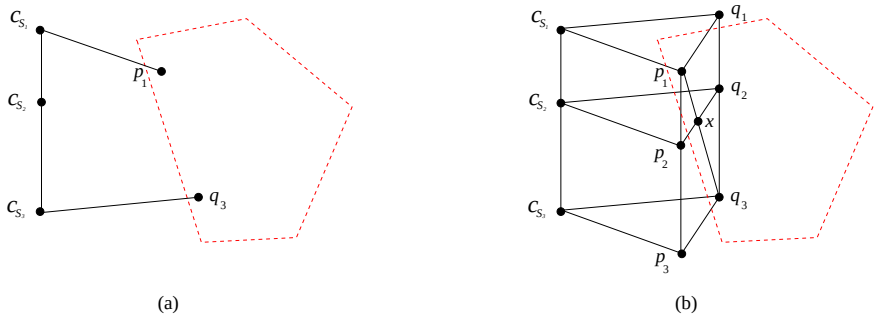


Fig. 6. The conflicting region is convex.

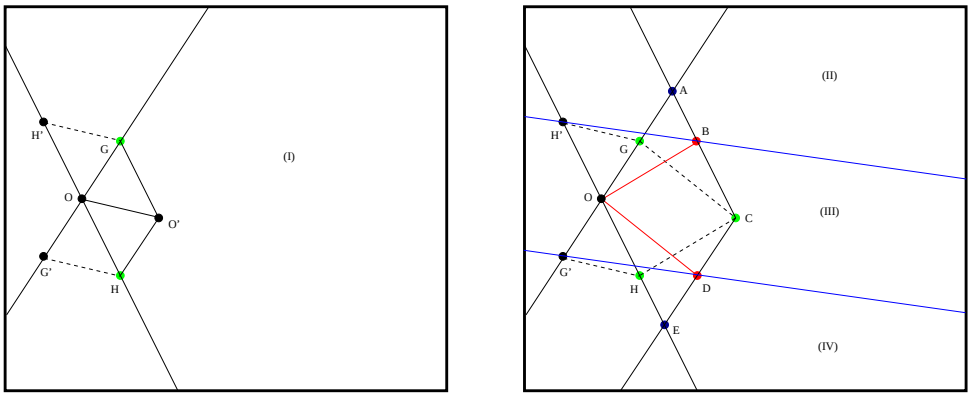
We draw lines $c_{S_2}p_2 \shortparallel c_{S_1}p_1$, $c_{S_3}p_3 \shortparallel c_{S_1}p_1$, $c_{S_1}q_1 \shortparallel c_{S_3}q_3$ and $c_{S_2}q_2 \shortparallel c_{S_3}q_3$, then p_1, p_3, q_3 and q_1 form a parallelogram with p_1q_3 being one of the diagonals; see Fig. 6(b). Since p_2 and q_2 are on the opposite sides of the parallelogram p_2q_2 intersects p_1q_3 at some point x. Since $p_2 \in S_2$, $q_2 \in S_2$ and S_2 is convex, we know that $x \in S_2$. Since $p_1 \in S$, $q_3 \in S$ and S is convex, we know that $x \in S$. Therefore $x \in S \cap S_2$, and hence $c_{S_2} \in \psi(S, c_S)$; which is a contradiction. □

Suppose we have a set of translates on a cartesian plane and we sort the centers by their x-coordinates. We claim that for any translate S with center c_S, the number of centers of non-overlapping translates with larger or equal x-coordinates, which lies in $\psi(S, c_S)$, is less than four. We can transform this problem into a simpler mathematical problem: given a set of n real vectors $X = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ in $\mathbf{R^2}$ with non-negative x-coordinates, and $D(X) = \{\boldsymbol{v}_i - \boldsymbol{v}_j | i \neq j\}$. Let $C(X)$ be the convex set

$$C(X) = \{\sum_{i=1}^{n} a_i \boldsymbol{v}_i | \sum_{i=1}^{n} |a_i| \leq 1, |a_i| \leq 1, \boldsymbol{v}_i \in X\}.$$

How large must n be to guarantee that the two sets $C(X)$ and $D(X)$ are not disjoint. It turns out $n = 4$ is sufficient; we have the following lemma:

Lemma 5. *If $n \geq 4$, then $C(X) \cap D(X) \neq \emptyset$.*



(a) (b)

Fig. 7. Cases when $n = 2$ and 3.

Proof. We present a geometric proof of the lemma. We assume that there are four points in X such that $C(X) \cap D(X) = \emptyset$, and try to derive a contradiction. Without loss of generality, we can assume that the first two points G and H have the maximum and minimum slope OG and OH respectively, see Fig. 7(a). Let G' and H' be the symmetric points of G and H with respect to O, and we draw $GO' \parallel OH$ and $HO' \parallel OG$; GO' and HO' meets at O'.

The third point therefore cannot lie to the left of OG or the right of OH. At the same time, since $\triangle GOO' \cong OG'H$ and $\triangle HOO' \cong OH'G$, both $\triangle GOO'$ and $\triangle HOO'$ cannot be feasible regions for the third point. To see this, suppose that the third point C lies in $\triangle GOO'$, then the vector $\boldsymbol{CG} \in C(X)$, which is a contradiction. Therefore the only feasible region is region (I). Let C be the third point, and we draw $OB \parallel HC$ and $CB \parallel HO$ so that OB and CB meet at B; we draw $OD \parallel GC$ and $CD \parallel GO$ so that OD and CD meet at D. We then draw lines $H'B$ and $G'D$, it is easy to see that $H'B \parallel OC \parallel G'D$. Let A be the intersection of CB and OG, E be the intersection of CD and OH; see Fig. 7(b).

Let P be the fourth point, so $X = \{\boldsymbol{OG}, \boldsymbol{OH}, \boldsymbol{OC}, \boldsymbol{OP}\}$, then depending the location of P, there are four cases:

1. If P lies in $\triangle ABO$ or $\triangle EDO$, without loss of generality, assume P lies in $\triangle ABO$. We draw lines $G'Q \parallel OB$ and $GQ \parallel AB$; GQ meets $G'Q$ at Q, then it is clear that $\triangle ABO \cong \triangle GQG'$. Since $\triangle GQG'$ completely lies in $C(X)$, it follows that $\boldsymbol{GP} \in C(X)$, which is a contradiction; see Fig. 8(a).
2. If P lies in the quadrilateral $OBCD$, then the quadrilateral $OBCD$ is congruent to the quadrilateral $CHOG$, therefore $\boldsymbol{PC} \in C(X)$, which is a contradiction; see Fig. 8(b).

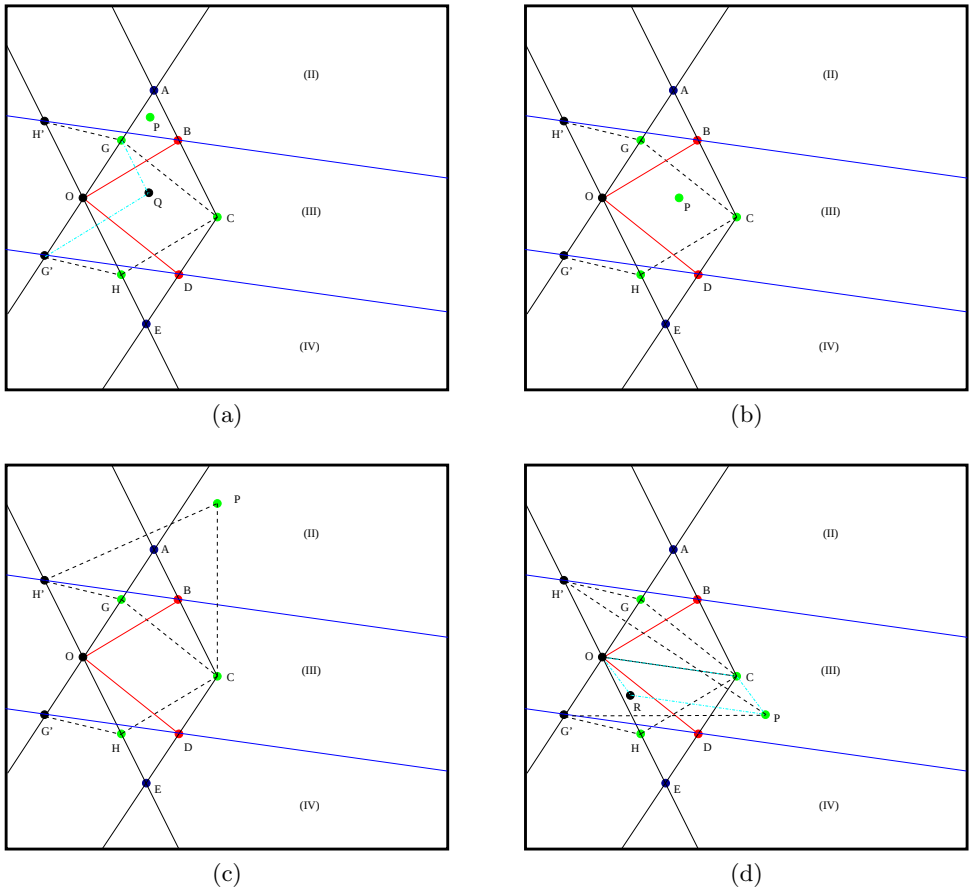


Fig. 8. Cases when $n = 4$.

3. If P lies in region (II) or (IV), without loss of generality, assume P in region (II), then $\boldsymbol{OB} \in C(X)$, and hence $\boldsymbol{HC} \in C(X)$, which is a contradiction; see Fig. 8(c).
4. If P lies in region (III), then we draw lines $PR \parallel CO$ and $OR \parallel CP$; PR meets OR at R. Since $PR \parallel CO$ and $|PR| = |CO|$, $\boldsymbol{OR} \in C(X)$, and hence $\boldsymbol{CP} \in C(X)$, which is a contradiction; see Fig. 8(d).

Therefore, if $n \geq 4$, then $C(X) \cap D(X) \neq \emptyset$; this completes the proof. □

We now prove Corollary 1.

Proof. We consider the intersection graph of the translates and show that there always exists a vertex which has at most 3 independent neighbors. We consider the leftmost translate S, breaking tie arbitrarily; and we set the center c_S of S to be the origin $(0, 0)$ of coordinate system. Then the centers of all translates have non-negative x-coordinates, furthermore, each of these centers defines a vector to the origin. Suppose there are at least four independent translates S_1, S_2, S_3, S_4 intersecting with S then we let $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ be the vectors representing the centers and $X = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4\}$.

Let

$$C(X) = \{\sum_{i=1}^{4} a_i \boldsymbol{v}_i | \sum_{i=1}^{4} |a_i| \leq 1, |a_i| \leq 1, \boldsymbol{v}_i \in X\},$$

then by Lemma 4, the conflicting region $\psi(S, c_S)$ of S contains $C(X)$. Let $D(X) = \{\boldsymbol{v}_i - \boldsymbol{v}_j | i \neq j\}$, by Lemma 5, $C(X) \cap D(X) \neq \emptyset$. Therefore $\psi(S, c_S) \cap D(X) \neq \emptyset$. Without loss of generality, we can assume $\boldsymbol{v}_1 - \boldsymbol{v}_2 \in \psi(S, c_S)$, which means that S_1 intersects with S_2, which is a contradiction. Therefore, in terms of the intersection graph of the translates, the leftmost vertex has at most 3 independent neighbors; hence the graph is inductive 3-independent. □

7.2 A Proof of Corollary 2

Proof. Considering any object, we draw a line passing through its center. Suppose it has at least six independent neighbors, then by rotating the line, we can force at least four centers to be on one side of the line (including those on the line). By Lemma 5, and a similar argument in the proof of Corollary 1, we can conclude that this is impossible. □

7.3 A Proof of Corollary 3

Proof. We order the objects by non-decreasing size. Consider the object of smallest size. The argument in Lemma 5 and Corollary 1 shows that there are at most 5 independent objects intersecting the smallest object. □