Recognizing Global Occurrence of Local Properties

Yair Caro
School of Education, Dept. of Mathematics
University of Haifa-ORANIM, Tivon 36006, Israel
and
Raphael Yuster
Department of Mathematics
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel-Aviv University, Tel-Aviv, Israel

Abstract

Let $\mathcal{P}$ be a graph property. For $k \geq 1$, a graph G has property P_k iff every induced k-vertex subgraph of G has $\mathcal{P}$. For a graph G we denote by $N_{P_k}(G)$ the number of induced k-vertex subgraphs of G having $\mathcal{P}$. A property is called *spanning* if it does not hold for graphs that contain isolated vertices. A property is called *connected* if it does not hold for graphs with more than one connected component. Many familiar graph properties are spanning or connected. We also define the notion of *simple* properties which also applies to many well-known monotone graph properties. A property $\mathcal{P}$ is recursive if one can determine if a graph G on n vertices has $\mathcal{P}$ in time $O(f_P(n))$ where $f_P(n)$ is some recursive function of n. We consider only recursive properties. Our main results are the following.

- If $\mathcal{P}$ is spanning and $k \geq 1$ is fixed, deciding whether a graph $G = (V, E)$ has P_k can be done in $O(V + E)$ time.
- If $\mathcal{P}$ is spanning, $f_P(n) = O(2^{n^3})$ and $k = O((\log n/\log\log n)^{1/3})$, deciding whether G has P_k can be done in polynomial time. Furthermore, if $\mathcal{P}$ is a monotone-increasing simple property with $f_P(n) = O(2^{n^2})$ (Hamiltonicity, perfect-matching and s-connectivity are just a few examples of such properties) and $k = O(\sqrt{\log n/\log\log n})$, deciding whether G has P_k can be done in polynomial time.
- If $k \geq 1$ and $d \geq 1$ are fixed, and $\mathcal{P}$ is either a connected property (Hamiltonicity is an example of such a property) or a monotone-decreasing infinitely-simple property (perfect-matching of independent vertices and Hamiltonian hole are examples of such properties) computing $N_{P_k}(G)$ for graphs G with $\Delta(G) \leq d$ can be done in linear time.
- If $\mathcal{P}$ is an NP-Hard monotone property and $\epsilon > 0$ is fixed, Then $P_{\lfloor n^\epsilon \rfloor}$ is also NP-Hard. The monotonicity is required as there are NP-Hard properties where P_k is easy when $k < n$.

1 Introduction

All graphs considered here are finite, undirected and simple. The graph-theoretical terminology used is compliant with that of [6]. A graph property $\mathcal{P}$ is a subset of the family of all graphs. We usually identify a property with a graph-theoretical statement that applies to all graphs in $\mathcal{P}$, and to no graph outside of $\mathcal{P}$. For example the Hamiltonicity property is the set of all Hamiltonian graphs. We also use the term "G has $\mathcal{P}$" or "$\mathcal{P}$ holds for G" as a synonym to $G \in \mathcal{P}$. The *complement property* of $\mathcal{P}$, denoted by $\mathcal{P}^c$, is $\{G^c \mid G \in P\}$ where G^c is the graph complement of G. Thus the property of containing a triangle is the complement property of containing an independent set of size 3. $\mathcal{P}$ is called monotone-decreasing if whenever $G \in \mathcal{P}$ then also $H \in \mathcal{P}$ for every spanning subgraph of G. $\mathcal{P}$ is monotone-increasing if $\mathcal{P}^c$ is monotone-decreasing. $\mathcal{P}$ is *spanning* if it does not hold for graphs with isolated vertices (hence it cannot be monotone-decreasing). $\mathcal{P}$ is *connected* if every graph having $\mathcal{P}$ is connected.

We associate with $\mathcal{P}$ a decision problem Π_P which, given as input a graph G, answers whether $G \in \mathcal{P}$. We call $\mathcal{P}$ NP-Hard (NP-Complete) if Π_P is NP-Hard (NP-Complete). Note that $\mathcal{P}$ and $\mathcal{P}^c$ are polynomial-time equivalent. It is an easy set-theoretical argument that there exist properties which are undecidable, and even monotone properties which are undecidable. In the sequel we consider only decidable (recursive) properties, thus the running time of Π_P is bounded by some recursive function $f_P(n)$. In what follows n will denote the size of the input graph and m the number of its edges.

Let $k = k(n)$ be an integer valued function (we assume that $k(n)$ is computable in $O(p(n))$ time for some polynomial p, and also that k(n)=o(n)). With every property $\mathcal{P}$ we associate a property P_k, where a graph G on n vertices has P_k iff every induced subgraph of G on k vertices has $\mathcal{P}$. The associated decision problem is Π_{P_k}. We also denote by $N_{P_k}(G)$ the number of induced k-subgraphs of G having $\mathcal{P}$. Thus G has P_k iff $N_{P_k}(G) = \binom{n}{k}$.

Many classical problems in graph theory involve the requirement that all k-subgraphs of a given graph have a certain property. Among the most famous ones are the Turán type problems, in which one requires that no k-vertex subgraph of G contains a copy of some fixed graph on k vertices, see e.g [5]. The *k-Independent Set* problem [11] in which one needs to determine if a graph has no independent set on k vertices is equivalent to the requirement that every k-vertex subgraph contains an edge. The Ramsey Numbers $R(H, K_n)$ are another obvious example [12]. Many other problems can be formulated equivalently be defining an appropriate property $\mathcal{P}$ and asking whether a given graph has P_k. Hence, the motivation in investigating P_k for various graph properties cannot be overestimated. For other sources which deal with graph-theoretical aspects of P_k the reader is referred to [2, 7, 8, 1, 14, 13]. Our main concern in this paper is to estimate the complexity of Π_{P_k} and the complexity of computing $N_{P_k}(G)$. Some graph-theoretical aspects of P_k (i.e. the structure possessed by graphs having P_k for various properties) was studied in [7], [8] and [1]. These papers

contain other relevant references as well. Approximating the number of labeled copies of k-vertex graphs in a large graph is discussed in [9].

Clearly, if k is a constant, $N_{P_k}(G)$ can be computed in $O(n^k)$ time. We simply check whether $\mathcal{P}$ holds for each induced k-subgraph of G. Each such check is done in constant time since $\mathcal{P}$ is recursive. This fact, in particular, means that Π_{P_k} can be solved in $O(n^k)$ time. It turns out that for spanning properties we can solve Π_{P_k} much faster.

Theorem 1.1 *Let $\mathcal{P}$ be a spanning graph property, and let $k = k(n)$. Then Π_{P_k} can be solved in $O(f_P(k)k^{4k^3}n + m)$ time.*

As an immediate consequence of Theorem 1.1 we have that when k is constant, Π_{P_k} can be solved in (linear) $O(n+m)$ time. Furthermore, even when k is $O((\log n/\log\log n)^{1/3})$ and $f_P(n) = O(2^{n^3})$ we have that Π_{P_k} can be solved in polynomial time. Note that many well known NP-Hard properties have simple algorithms which give $f_P(n) = O(2^{n^3})$. Numerous examples appear in [11].

The result in Theorem 1.1 is based on an algorithm for the induced subgraph isomorphism problem. In this algorithm we need to test whether a graph G on n vertices and maximum degree d contains an induced copy of some subgraph H on at most h vertices. When H is connected and the requirement that the copy be induced is not needed, this algorithm is a part of the folklore and can be implemented in $O(d^{2h^2}n)$ time. However, if we insist that the copy of H be induced, or that H may not be connected, the algorithm is more complex and we present it in section 2. The running time we obtain for it is $O(d^{3h^3}n)$. Some other special cases of the fixed subgraph isomorphism algorithm are presented in [3], [4] and [15].

In many interesting cases we are able to implement a faster version of the subgraph isomorphism algorithm. In order to identify these cases we need the following definitions. A graph H with no isolated vertices is called $(\mathcal{P},k)$-*extremal* if H has $h \leq k$ vertices, and the k-vertex graph H' obtained from H by adding $k-h$ isolated vertices does not have $\mathcal{P}$, but whenever we delete any edge from H', the resulting graph has $\mathcal{P}$. A monotone-decreasing property $\mathcal{P}$ is called *simple* if for every $k > 1$, every $(\mathcal{P},k)$-extremal graph is connected. A simple property is called k-*simple* if every $(\mathcal{P},k)$-extremal graph has more than $k/2$ vertices. If $\mathcal{P}$ is k-simple for all $k > 1$, we call $\mathcal{P}$ *infinitely-simple*. For example, the property of planarity is a simple property since it is monotone-decreasing, and every extremal graph w.r.t. planarity is connected. Also note that planarity is 9-simple since the smallest non-planar graph has 5 vertices. It is not 10-simple since K_5 is non-planar, and is $(\mathcal{P},10)$-extremal, but $5 \leq 10/2$. We call a monotone-increasing property simple (r-simple, infinitely-simple) if its complement property (which is monotone decreasing) is simple (r-simple, infinitely-simple, respectively). Identifying simple properties is an interesting graph-theoretical problem on its own right. It is also useful from the algorithmic perspective:

Theorem 1.2 *Let $\mathcal{P}$ be a monotone increasing simple property, and assume that $\mathcal{P}$ is a spanning property. Let $k = k(n)$. Then Π_{P_k} can be solved in $O(f_P(k)k^{3k^2}n + m)$ time.*

Theorem 1.3 *Let $\mathcal{P}$ be a monotone-decreasing k-simple property, and let d be a fixed integer. Let G be a graph with $\Delta(G) \leq d$. Then $N_{P_k}(G)$ can be computed in (linear) $O(n)$ time.*

A result similar to Theorem 1.3 applies also to connected properties.

Theorem 1.4 *Let $\mathcal{P}$ be a connected property, and let k and d be fixed integers. Let G be a graph with $\Delta(G) \leq d$. Then $N_{P_k}(G)$ can be computed in (linear) $O(n)$ time.*

In order to demonstrate the applicability of the above mentioned theorems, let us consider some well-known graph properties. The following properties are spanning, simple, and have $f_P(n) = O(n^n)$. The first three are also connected properties.

1. Hamiltonicity (containing a Hamiltonian cycle).
2. Containing a Hamiltonian path.
3. Being s-connected.
4. Containing a perfect matching ($f_P(n) = O(n^{2.5})$ in this case).

Thus for all these properties we have that if $k = O(\sqrt{\log n / \log\log n})$ then Π_{P_k} can be solved in polynomial time. The following properties are monotone-decreasing, simple, and have $f_P(n) = O(n^n)$.

1. The complements of all the properties mentioned in the list above. They are all infinitely-simple.
2. Planarity is 9-simple.
3. The property of having maximum degree r is $2r + 3$ simple.
4. The property of not containing a fixed connected graph H on h vertices as a subgraph is $2h - 1$ simple.

The proof that a property is simple can be easy or hard, depending on the property. In section 3 we present the proofs of r-simplicity for all of the above properties, and prove Theorems 1.2, 1.3 and 1.4. We also give an example of a monotone graph property which is non-simple (but still with $f_P(n) = O(2^{n^3})$) in which case we can use Theorem 1.1.

In section 4 we consider the hardness of P_k. It is not difficult to show that if $\mathcal{P}$ is a monotone NP-Hard problem and $\epsilon > 0$ is fixed, then $P_{\lfloor n^\epsilon \rfloor}$ is NP-Hard as well. Hence we can verify if all subgraphs of order $\sqrt{\log n / \log\log n}$ are Hamiltonian in polynomial time, but we cannot verify this property in polynomial time (unless P=NP) if the subgraphs are of order, say, $n^{0.01}$. (This problem may even be outside of NP). We show that the monotonicity requirement is essential as there are NP-Hard properties for which P_k is easy for all $k \leq n - 1$. We also show that there are easy properties for which $P_{\lfloor n^\epsilon \rfloor}$ is NP-Hard.

2 Detecting P_k for spanning properties

As mentioned in the introduction, the basic ingredient that we require is an algorithm for the (induced) subgraph isomorphism problem. The following lemma describes such an algorithm.

Lemma 2.1 *Let G be a graph on n vertices, and let $d = \Delta(G)$ denote its maximal degree. Let H be a graph on h vertices. Finding whether H is isomorphic to some (induced) h-vertex subgraph of G can be done in $O(d^{3h^3}n)$ time. Furthermore, if H is connected we may list all the subgraphs of G which are isomorphic to H in $O(d^{2h^2}n)$ time.*

Proof We prove the lemma for the case where an induced copy of H is searched for. The case where a not-necessarily induced copy of H is searched for is simpler (and faster). Let $H_1, \ldots, H_r$ denote the distinct connected components of H, and let h_i be the number of vertices of H_i. Let p_i be the multiplicity of H_i. (E.g. if H is the union of two vertex-disjoint triangles we have $H_1 = K_3$, $r = 1$, $h_1 = 3$ and $p_1 = 2$). Constructing $H_1, \ldots, H_r$ can be done in $O(h^2)$ time, and finding $p_1, \ldots, p_r$ can be done in $O(r^2 h!) = O(h^h)$ time using a naive graph isomorphism algorithm.
Assume for simplicity that $V = \{1, \ldots, n\}$. For each $j = 1, \ldots, n$ and each $i = 1, \ldots, r$ let L_{ij} denote a list whose elements are all the h_i subsets of $\{j, \ldots, n\}$ that induce a copy of H_i, and that contain j. L_{ij} is computed as follows. We perform a Breadth-First Search beginning at j, and whose depth is $h_i - 1$. Let X_{ij} be the set of all vertices discovered by this search and that are greater than $j - 1$. Clearly, $|X_{ij}| < d^{h_i}$, and X_{ij} can be computed in $O(d^{h_i})$ time, and, since H_i is connected, every member of L_{ij} is a subset of X_{ij}. We consider all h_i subsets of X_{ij} that contain j. There are at most $\binom{d^{h_i}}{h_i - 1}$ such subsets. For each such subset, we test whether the induced subgraph of G on this subset is isomorphic to H_i. If this is the case we add the subset to L_{ij}. Thus, L_{ij} is constructed in $O(d^{h_i} + \binom{d^{h_i}}{h_i - 1} h_i^2 h_i!) = O(d^{2h_i^2})$. All L_{ij}'s are constructed in $O(nd^{2h^2})$ time. With no additional cost we can also compute the size l_{ij} of L_{ij}, and $l_i = \sum_{j=1}^{n} l_{ij}$. Note also that $l_{ij} \leq d^{h_i^2}$, and since this bound does not depend on j, every vertex j appears in at most $d^{2h_i^2}$ subsets of h_i vertices that induce a copy of H_i. Thus every vertex appears in at most d^{2h^2} subsets that induce some subgraph isomorphic to a connected component of H.
Consider a subset S that appears in some L_{ij}. Let $N(S)$ be the set of all vertices of G that are neighbors of some vertex of S. Clearly, $|N(S)| \leq dh_i$, and $S \subset N(S)$ (since H_i is connected and S induces H_i). We claim that if $l_i > (dh)d^{2h^2}$ for all $i = 1, \ldots, r$, then G contains an induced copy of H. We can construct such a copy using the following greedy procedure: We first pick some subset S of L_{1j} for some j such that L_{1j} is not empty (there exists such a j since $l_1 > 0$). For each $v \in N(S)$ we nullify all the lists $L_{vj'}$ for all $j' = 1, \ldots, r$. This is done since the subgraphs induced by the subsets in these lists either intersect with S or have an edge connecting them to S. Hence they are no longer allowed. Note that the number of subsets that have been nullified is at most $(dh_1)d^{2h^2}$. Suppose we have already constructed an induced subgraph containing all required

copies of $H_1, \ldots, H_t$ and nullified all the disallowed lists. We now need to find another induced copy of H_t (or the first copy of H_{t+1} in case we already found p_t copies of H_t). This can be done since we have only nullified at most $d(h_1p_1 + \ldots + h_tp_t)d^{2h^2} \leq (dh)d^{2h^2}$ subsets. Hence some L_{tj} (or $L_{(t+1)j}$) is not empty, and we can pick a subset from it which will be the desired induced copy. Since the computation of $N(S)$ requires only dh_i^2 time, this is dominated by the time needed to construct the L_{ij}'s which is $O(nd^{2h^2})$, which is the running time of the algorithm in this case.
We need to consider the case where for some i we have $l_i \leq dh(d^{2h^2})$. Let I denote the set of all these i's. Let $L_i = \cup_{j=1}^{n} L_{ij}$ for $i \in I$. (Constructing the L_i's is linear in the sizes of the lists, and incurs no additional overhead). We pick all possible p_i subsets from L_i for each $i \in I$. We check whether all these $\sum_{i \in I} p_i$ subsets are pairwise-disjoint and that there is no edge connecting them. The number of choices for these subsets is $O((dh(d^{2h^2}))^{\sum_{i \in I} p_i})$ and each check can be done in dh^2 time, which is dominated, overall, by $O(d^{3h^3})$. Clearly, G contains an induced copy of H iff one of these checks succeeds. For the rest of the induced copies, that correspond to the H_i where $i \notin I$, we can proceed as before, using the greedy procedure.
It is easy to see that when H is connected (hence $r = 1$ and $p_1 = 1$) it is sufficient to halt the algorithm whenever the first subgraph isomorphic to H is found. Hence the running time in this case is $O(d^{2h^2}n)$. Furthermore, L_{1j} contains at most d^{h^2} subsets. All the subsets from all the L_{1j}'s can therefore be listed in $O(d^{2h^2}n)$ time. □

Proof of Theorem 1.1 Let $\mathcal{P}$ be the spanning graph property, and $k = k(n)$. Let G be the input to Π_{P_k}. We first verify in $O(n+m)$ time whether $\delta(G) > n - k$, where $\delta(G)$ is the minimum degree of G. If this is not the case then P_k does not hold for G since some vertex has $k-1$ non-neighbors, which implies that G has a k-subgraph with an isolated vertex. Such a subgraph cannot have $\mathcal{P}$ since $\mathcal{P}$ is a spanning property.

Since we assume that $k = o(n)$ we are guaranteed that $m = \Theta(n^2)$, and we may construct, in $O(n+m)$ time, the graph G^c for which $\Delta(G^c) \leq k-2$ holds. Let $\mathcal{H}$ be the set of all k-vertex graphs that do not have $\mathcal{P}^c$. We can construct the set $\mathcal{H}$ in $O(f_P(k)2^{k^2})$ time since we may generate all $2^{\binom{k}{2}}$ (labeled) graphs on k vertices and check each graph, in $O(f_P(k))$ time, whether it has $\mathcal{P}^c$. Clearly, G has P_k iff G^c does not contain any graph from $\mathcal{H}$ as an induced subgraph. We use Lemma 2.1 to check, for each member of $\mathcal{H}$, in $O(k^{3k^3}n)$ time, whether it is an induced subgraph of G^c. The overall running time is therefore $O(f_P(k)2^{k^2} + 2^{k^2}k^{3k^3}n) = O(f_P(k)k^{4k^3}n + m)$. □

As mentioned in the introduction, an immediate corollary of Theorem 1.1 is that when k is constant P_k is linear-time solvable, and when $k = O((\log n/\log\log n)^{1/3})$ and $f_P(k) = O(2^{k^3})$, P_k is solvable in polynomial time. If all graphs in $\mathcal{H}$ turn out to be connected, we may even have $k = O((\log n/\log\log n)^{1/2})$. This may be the case for some spanning graph properties. Consider the property of being connected. This is a spanning property, and $\mathcal{H}$ is the set of all k-vertex graphs whose complements are non-connected. Such graphs must be connected, since every graph is either connected or its complement is. It follows that all elements of $\mathcal{H}$ are connected.

3 Detecting P_k and computing $N_{P_k}(G)$ when P is simple or connected

The main purpose of this section is twofold. Our first goal is to show how to compute $N_{P_k}(G)$ whenever $\mathcal{P}$ is a simple property or a connected property, and whenever G has bounded degree. This is done by proving Theorems 1.3 and 1.4. We also show how Π_{P_k} can be solved in polynomial time even when $k = O(\sqrt{\log n / \log\log n})$ whenever $\mathcal{P}$ is a spanning simple property. This is done by proving Theorem 1.2. In the second part of this section we prove that many well-known and even some NP-Hard graph properties are simple or connected. We also give an example of a natural graph property which is non-simple.

The following lemma gives an indication as to why simple properties are easier to detect than others.

Lemma 3.1 *Let $\mathcal{P}$ be a monotone-decreasing property, and let $k \geq 1$. Let $\mathcal{H}$ be the set of all $(\mathcal{P}, k)$-extremal graphs. Then G has P_k iff it does not contain any member of $\mathcal{H}$ as a subgraph.*

Proof Suppose G contains a graph $H \in \mathcal{H}$ with h vertices. Thus, G also contains the k-vertex graph H' obtained from H by adding $k-h$ isolated vertices. Since H is $(\mathcal{P}, k)$-extremal, H' does not have $\mathcal{P}$. Let G' be a k-vertex induced subgraph of G containing H' as its subgraph. Since $\mathcal{P}$ is monotone-decreasing we have that G' also does not have $\mathcal{P}$. Thus, G does not have P_k. Now suppose that G does not have P_k. Let G' be a k-vertex induced subgraph of G that does not have $\mathcal{P}$. We delete edges from G' one by one until we obtain a subgraph H' of G' that is minimal in the sense that H' does not have $\mathcal{P}$ but any edge we delete from H' results in a graph which has $\mathcal{P}$. Ignoring the isolated vertices of H', if there are any, results in a $(\mathcal{P}, k)$-extremal graph H. □

Proof of Theorem 1.2, We use the same proof of Theorem 1.1, with one change. The set $\mathcal{H}$ is now defined as in Lemma 3.1, but with respect to the property $\mathcal{P}^c$ which is monotone-decreasing. That is, $\mathcal{H}$ is the set of all $(\mathcal{P}^c, k)$-extremal graphs. Note that given a graph H on $h < k$ vertices, we can verify (in $O(k^2 f_P(k))$ time) whether H is $(\mathcal{P}^c, k)$-extremal by adding to it $k-h$ isolated vertices and checking that the resulting graph H' does not have $\mathcal{P}^c$ and that whenever any edge is deleted from H' the resulting graph has $\mathcal{P}^c$. By Lemma 3.1 we have that the same proof of Theorem 1.1 still holds. Furthermore, since every graph in $\mathcal{H}$ is connected, the claimed running times follow from Lemma 2.1. □

The proof of Theorem 1.3 requires an additional idea which is stated in the following lemma.

Lemma 3.2 *Let k and d be fixed positive integers. Let $\mathcal{F} = \{S_1, \ldots, S_m\}$ be a family of subsets of $\{1, \ldots, n\}$, where $k/2 < |S_j| \leq k$, for $j = 1, \ldots, m$. Suppose that each $i = 1, \ldots, n$ belongs to at most d subsets. Then we can count how many k-subsets of $\{1, \ldots, n\}$ contain a member of $\mathcal{F}$ in $O(n)$ time.*

Proof We note first that $m \leq dn = O(n)$. Each S_j intersects at most $k(d-1)$ other members of $\mathcal{F}$. Hence if we put $I(j) = \{j' \mid j' < j\ ,\ S_j \cap S_{j'} \neq \emptyset\}$, we have $|I(j)| \leq k(d-1)$. All the $I(j)$'s can be constructed in $O(m) = O(n)$ time by scanning the S_j's sequentially and creating the lists $L(i) = \{j \mid i \in S_j\}$ and observing that $I(j) = \cup_{i \in S_j} L(i) \cap \{1, \ldots, j-1\}$. A subset $R \subset I(j)$ is called j-obsolete if $|\cup_{j' \in R} S_{j'} \cup S_j| \leq k$. Let $R(j)$ denote the set of all j-obsolete subsets of $I(j)$. For a j-obsolete subset R let $s(R) = |\cup_{j' \in R} S_{j'} \cup S_j|$. Since $I(j)$ has constant size, $R(j)$ can also be computed in constant time for each j, and all $R(j)$'s can be computed in $O(n)$ time. Let C_j denote the number of k-subsets of $\{1, \ldots, n\}$ that contain some subset $S_{j'}$ with $1 \leq j' \leq j$. We want to compute C_m. Note that $C_1 = \binom{n-|S_1|}{k-|S_1|}$. Assuming we have already computed C_{j-1}, we show how to compute C_j in constant time. We claim that

$$C_j = C_{j-1} + \binom{n-|S_j|}{k-|S_j|} + \sum_{R \in R(j)} (-1)^{|R|} \binom{n-s(R)}{k-s(R)}.$$

The negation of the rightmost summand in this expression counts the number of k-subsets of $\{1, \ldots, n\}$ that contain S_j and some $S_{j'}$ that intersects with S_j, where $j' < j$, by the inclusion-exclusion principle. However, since $|S_j| > k/2$, this is also the number of k-subsets of $\{1, \ldots, n\}$ that contain S_j and some $S_{j'}$ with $j' < j$. Hence in the above expression, $C_j - C_{j-1}$ is exactly the number of k-subsets that contain S_j and do not contain any $S_{j'}$ with $j' < j$. Thus, C_j is correctly computed. □

Proof of Theorem 1.3 Assume that the vertices of G are $\{1, \ldots, n\}$. We construct the set $\mathcal{H}$ of all $(\mathcal{P}, k)$-extremal graphs. This is done in constant time as k is constant. Note that $\mathcal{H}$ consists only of connected graphs with more than $k/2$ vertices. For each $H \in \mathcal{H}$, we construct a list, L_H, of all $|H|$-vertex subsets of G that induce a subgraph containing (as a spanning subgraph) a copy of H. By Lemma 2.1 each such list (and henceforth all lists) can be constructed in $O(n)$ time and contains $O(n)$ elements. We now create, in $O(n)$ time, the sets M_i, where M_i contains all the subsets in $\cup_{H \in \mathcal{H}} L_H$ that contain i as their lowest numbered element. The reason for creating the M_i's is the following: It may be the case that some x-subset of G appears in more than one list L_H, since the subgraph induced by it may contain more than one element of $\mathcal{H}$ as a spanning subgraph. By creating the M_i's and using the fact (which is shown in the proof of Lemma 2.1) that the number of elements in M_i is bounded by a constant which is at most $|\mathcal{H}|d^{2k^2} \leq d^{3k^2}$, we can make sure that we eliminate these multiplicities and that the M_i's are, indeed, sets. We now take the union of the M_i's and obtain $\mathcal{F} = \cup_{H \in \mathcal{H}} L_H$. $\mathcal{F}$ has the property that every induced k-subgraph of G that does not have $\mathcal{P}$ contains an element of $\mathcal{F}$. Furthermore, each subset in $\mathcal{F}$ has at most k elements and more than $k/2$ elements, and every vertex i of G appears in at most d^{3k^2} of the subsets. It therefore follows from Lemma 3.2 that $N_{P_k}(G)$ can be computed in $O(n)$ time. □

Proof of Theorem 1.4 The proof of this theorem is similar (and, in fact, simpler) than that of Theorem 1.3. In this case we construct the set $\mathcal{H}$ of all k-vertex graphs that have $\mathcal{P}$. As before,

this is done in constant time as k is constant, and $\mathcal{H}$ consists only of connected graphs having exactly k vertices. We construct the lists L_H and the sets M_i as before, in $O(n)$ time. Clearly, $N_{P_k}(G) = \sum_{i=1}^{n} |M_i|$. $\square$

It is time to show that simplicity of properties is not an artificial definition, but, in fact, a very practical one, as many well-known graph-theoretic properties are simple.

Theorem 3.3 *The following monotone-increasing properties are spanning and infinitely-simple, and have* $f_P(n) = o(2^{n^2})$.

1. *Containing a Hamiltonian cycle.*
2. *Containing a Hamiltonian path.*
3. *Being s-connected.*
4. *Containing a perfect matching.*

Proof The fact that these properties are spanning and that they can be recognized by simple exponential algorithms (some are even polynomial) is straightforward. We prove that they are infinitely-simple. By definition, we need to look at their complement properties. We consider a (P, k)-extremal graph H on $h \leq k$ vertices, and denote by H' the k-vertex graph obtained from H be adding $k - h$ isolated vertices.

We begin by showing that containing a Hamiltonian hole is an infinitely-simple property (a hole is a permutation of vertices where every two consecutive vertices are independent). If H' contains at least $k/2$ isolated vertices, the minimum degree of the complement of H' is at least $k/2$, and thus by Dirac's Theorem [6] H' contains a Hamiltonian hole, which is impossible. Thus, we have shown $h > k/2$. The same argument shows that H' (and therefore H) must have at least one connected component X with $x = |X| > k/2$. We must show that X is the *only* connected component of H. Assume, in contradiction, that $x < h$. By the minimality of H we know that if we add to X a set of $k - x$ isolated vertices, the resulting graph has a Hamiltonian hole. This hole is a permutation of k vertices, where x of them are from X. An X-segment in this permutation is a set of consecutive vertices of X. The length of a segment is one less than the number of elements it contains. The total length of the maximal X-segments (segments that cannot be extended are maximal) is therefore at least $x - (k - x) = 2x - k$. Let us therefore consider a set $S = \{S_1, \ldots, S_s\}$ of vertex-disjoint X-segments (not necessarily maximal) whose total length is exactly $2x - k$. (Hence there is a total of $2x - k + s$ vertices of X in these segments). Each segment has two endpoints. We pick one endpoint from each S_i, denoted by s_i, $i = 1, \ldots, s$. If we delete from each such S_i all vertices but s_i, we remain with a set Y of exactly $x - (2x - k) = k - x$ vertices of X. There are also $k - x$ vertices in $H' \setminus X$. Consider a $2(k - x)$ permutation alternating between vertices of Y and vertices of $H' \setminus X$. Now, replacing s_i in this permutation with the segment S_i, for $i = 1, \ldots, s$, we obtain a Hamiltonian hole in H', a contradiction.

The proofs of infinite-simplicity of Hamiltonian path and perfect matching are analogous to the Hamiltonian cycle proof, and are left to the reader.

We now consider the s-connectivity property. Since H'^c is not s-connected, there are two vertices x, y of H' that do not have s vertex-disjoint paths connecting between them in H'^c. Assume H is not connected, and let X denote a connected-component of it that does not contain both x and y. Let (u, v) be an edge in X. By the minimality of H, we know that $L = H'^c \cup \{(u, v)\}$ is s-connected. There are, therefore, s vertex-disjoint paths $p_1, \ldots, p_s$ connecting x and y in L. We can assume that each p_i is an induced path (except for, maybe, the edge (x, y), if it exists). One of these paths, say p_1, contains (u, v). We claim that we cannot have $x \notin X$. To see this, let us denote by w the first vertex of p_1 (form the direction of x) that belongs to X. Assume $w \neq u$ (otherwise, $w \neq v$). Then the vertex preceding w in p_1 is connected to u (since they are in different connected components in H'), contradicting the fact that p_1 is an induced path. Likewise we cannot have $y \notin X$. Thus we have shown $x, y \in X$, a contradiction to our assumption that H is not connected. □

Theorem 3.4 *The following properties are monotone-decreasing, simple, and have $f_P(n) = o(2^{n^2})$.*

1. *The complements of all the properties mentioned in Theorem 3.3. They are all infinitely-simple.*
2. *Planarity is 9-simple.*
3. *The property of having maximum degree r is $2r + 3$ simple.*
4. *The property of not containing a fixed connected graph H on h vertices as a subgraph is $2h - 1$ simple.*

Proof It is straightforward that these properties are monotone-decreasing and can be solved by exhaustive search in $o(2^{n^2})$ time, and some of them even in polynomial time. As to the properties in the first item in the list, they are all infinitely-simple because they are complements of infinitely-simple properties, shown as such in Theorem 3.3. Planarity is shown to be 9-simple in the Introduction. The property of having maximum degree r is equivalent to the property of not containing the star with $r + 2$ vertices. It thus reduces to the property mentioned in the last item. Since every graph that contains a connected graph H also has a connected component containing H, this property is simple. Such a (P, k)-extremal graph must contain at least h vertices, so this property is $2h - 1$ simple. □

Theorems 1.3 and 3.4 have many interesting consequences. For example, given a graph G with, say, $\Delta(G) \leq 1000$, counting the number of induced 9-vertex subgraphs of G which are planar can be done in linear time. Note that the naive algorithm requires $O(n^9)$ time.

Finally, we give an example of a natural graph property which is non-simple. The property $\mathcal{P}$ of containing a factor into vertex-disjoint triangles. This is, in fact, a monotone-increasing property

(which is NP-Hard, but with $f_P(n) = O(2^{n^2})$). Consider $\mathcal{P}^c$ and consider, for $k \geq 2$, the graph H on $3k$ vertices which is the vertex-disjoint union of K_2 and $K_{1,3k-3}$ (recall that $K_{n,m}$ is the complete-bipartite graph having vertex classes of sizes m and n). H does not have k vertex-disjoint independent sets of size 3 each. However, whenever we delete any edge from H, the resulting graph does have k vertex-disjoint independent sets of size 3. Hence it is $(P^c, 3k)$-extremal. It is, however, a non-connected graph.

4 Hardness results for P_k decision problems

We begin this section by proving that if $\mathcal{P}$ is a hard monotone property, so is P_k for $k = \lfloor n^\epsilon \rfloor$. Although the proof is easy, it raises some interesting computational issues.

Theorem 4.1 *Let $\mathcal{P}$ be an NP-Hard monotone property, and $\epsilon > 0$, fixed. Then $P_{\lfloor n^\epsilon \rfloor}$ is also NP-Hard. If $\mathcal{P}$ belongs to NP, then $P_{\lfloor n^\epsilon \rfloor} \in \Pi_2^p$.*

Proof, Note that we may assume that $\mathcal{P}$ is monotone-decreasing since $\mathcal{P}$ is NP-Hard iff $\mathcal{P}^c$ is NP-Hard, and P_k is NP-Hard iff P_k^c is NP-Hard. The following straightforward polynomial transformation from Π_P to $\Pi_{P_{\lfloor n^\epsilon \rfloor}}$ gives the first part of the theorem. Let G be an n vertex graph. We add to G a set of $N = \lceil n^{1/\epsilon} \rceil - n$ isolated vertices. Denote the obtained graph by G'. G' is constructed in polynomial time, has $N + n$ vertices, and $\lfloor (N+n)^\epsilon \rfloor = n$. Suppose G does not have $\mathcal{P}$. Since G is an n-vertex induced subgraph of G', we have that G' does not have P_n. Any n-vertex induced subgraph of G' is isomorphic to a spanning subgraph of G. Thus if G has $\mathcal{P}$, and since $\mathcal{P}$ is monotone decreasing, G' has P_n. For the second part of the theorem, note that when $\mathcal{P}$ belongs to NP, the complement problem of $\Pi_{P_{\lfloor n^\epsilon \rfloor}}$ can be solved using a non-deterministic Turing reduction to Π_P, hence $P_{\lfloor n^\epsilon \rfloor}$ belongs to the complexity class Π_2^p in the polynomial hierarchy. □

Theorem 4.1 raises some interesting questions. Suppose first that $\mathcal{P}$ is monotone and polynomial time solvable. What can be said about the hardness of $P_{\lfloor n^\epsilon \rfloor}$. It may be the case that the problem becomes difficult. Consider the property of being a non-isolated graph (i.e a graph that has at least one edge). Then $P_{\lfloor n^\epsilon \rfloor}$ is equivalent to the decision problem "does the graph have an independent set of size n^ϵ", which is NP-Complete.

We have shown that if $\mathcal{P}$ is the Hamiltonicity problem, P_k is polynomial whenever $k = O(\sqrt{\log n / \log \log n})$ and is NP-Hard whenever $k = \Omega(n^\epsilon)$. It would be interesting to narrow the gap between these two bounds.

Finally, we note that the monotonicity assumption in Theorem 4.1 is essential. Consider the property $\mathcal{P}$ of all graphs whose edge set is the edge-disjoint union of triangles. It was shown by [10] that decomposing a graph G into a fixed graph H (which has a component with at least three edges) is NP-Hard. In particular, the property $\mathcal{P}$ is NP-Complete. It is clearly non-monotone, as the degree of every vertex in every graph having the property must be even. Suppose G has P_{n-1}.

If G is the isolated graph, the answer is trivially yes. If G is a complete graph, the answer is yes iff $n-1$ has a Steiner triple-system, which is also easy to verify. Otherwise, G has vertices u, v, w such that (u, v) is an edge and (u, w) is not. The degree of u in $G \setminus \{w\}$ differs by one from the degree of u in $G \setminus \{v\}$. Thus u has odd degree in one of these graphs an hence G does not have P_{n-1}. A simple induction argument shows that G does not have P_k for $1 < k \leq n-1$ unless G is an isolated graph or a complete graph with $n-k$ having a Steiner triple-system.

5 Acknowledgment

The authors wish to thank Noga Alon and the anonymous referees for helpful suggestions.

References

[1] M. Aigner and E. Triesch, *Realizability and uniqueness in graphs*, Discrete Mathematics 136 (1994), 3-20.

[2] N. Alon, *Independence number of locally sparse graphs and a Ramsey type problem*, Random Structures and Algorithms, to appear.

[3] N. Alon, R. Yuster and U. Zwick, *Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs*, Proc. 26th ACM STOC, ACM Press (1994), 326-335.

[4] N. Alon, R. Yuster and U. Zwick, *Finding and counting given length cycles*, Proc. 2nd European Symposium on Algorithm, Utrecht, the Netherlands (1994) 354-364.

[5] B. Bollobás, *Extremal Graph Theory*, Academic Press, London, 1978.

[6] J. A. Bondy and U.S. R. Murty, *Graph Theory with Applications*, Macmillan Press, London, 1976.

[7] Y. Caro, *On graphs with prescribed subgraphs of order k and a theorem of Kelly and Merriell*, Czechoslovak Mathematical Journal 44 (1994), 623-629.

[8] Y. Caro, *On graphs determined by their k-subgraphs*, Ars Combinatoria, to appear.

[9] R.A. Duke, H. Lefmann and V. Rödl, *A fast approximation algorithm for computing the frequencies of subgraphs in a given graph*, SIAM J. Comp. 24 (1995), 598-620.

[10] D. Dor and M. Tarsi, *Graph decomposition is NPC - A complete proof of Holyer's conjecture*, Proc. 20th ACM STOC, ACM Press (1992), 252-263.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company. New York, 1979.

[12] R. L. Graham, B. L. Rothschild and J. H. Spencer, *Ramsey Theory*, Wiley, New York (second edition), 1990.

[13] L. Nebeský, *Some sufficient conditions for the existence of a 1-factor*, J. Graph Theory 2 (1978), 251-255.

[14] D. P. Sumner, *Graphs with 1-factors*, Proc. Amer. Math. Soc. 42 (1974), 8-12.

[15] G. Sundaram and S.S. Skiena, *Recognizing small subgraphs*, Networks (1995), to appear.