Enumerating Maximal Induced Subgraphs

Yixin Cao
Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

Abstract

Given a graph G, the maximal induced subgraphs problem asks to enumerate all maximal induced subgraphs of G that belong to a certain hereditary graph class. While its optimization version, known as the minimum vertex deletion problem in literature, has been intensively studied, enumeration algorithms were only known for a few simple graph classes, e.g., independent sets, cliques, and forests, until very recently [Conte and Uno, STOC 2019]. There is also a connected variation of this problem, where one is concerned with only those induced subgraphs that are connected. We introduce two new approaches, which enable us to develop algorithms that solve both variations for a number of important graph classes. A general technique that has been proven very powerful in enumeration algorithms is to build a solution map, i.e., a multiple digraph on all the solutions of the problem, and the key of this approach is to make the solution map strongly connected, so that a simple traversal of the solution map solves the problem. First, we introduce retaliation-free paths to certify strong connectedness of the solution map we build. Second, generalizing the idea of Cohen, Kimelfeld, and Sagiv [JCSS 2008], we introduce an apparently very restricted version of the maximal (connected) induced subgraphs problem, and show that it is equivalent to the original problem in terms of solvability in incremental polynomial time. Moreover, we give reductions between the two variations, so that it suffices to solve one of the variations for each class we study. Our work also leads to direct and simpler proofs of several important known results.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases enumeration algorithm, hereditary graph class, maximal induced subgraph

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.31

Related Version *Full Version*: `http://arxiv.org/abs/2004.09885`

Funding Supported in part by the Hong Kong Research Grants Council (RGC) under grant 15221420 and the National Natural Science Foundation of China (NSFC) under grant 61972330.

1 Introduction

A vertex deletion problem asks to transform a given graph into a graph in a certain graph class $\mathcal{P}$ by deleting vertices. Many classic optimization problems belong to the family of vertex deletion problems, and their algorithms and complexity have been intensively studied. More often than not, the graph class is *hereditary*, i.e., closed under taking induced subgraphs. Examples include complete graphs, edgeless graphs, acyclic graphs, bipartite graphs, planar graphs, and perfect graphs. For a hereditary graph class, this problem is either NP-hard or trivial [40]. An equivalent formulation of a vertex deletion problem toward $\mathcal{P}$ is to ask for a maximum induced subgraph that belongs to $\mathcal{P}$. A plethora of algorithms, including approximation [42, 14, 12, 51, 36, 1], exact [30, 29], and parameterized [10, 15, 14, 12, 36, 1, 2], have been proposed for both formulations.

Yet another approach toward this problem is to enumerate, or generate or list, inclusion-wise *maximal* induced subgraphs that belong to the graph class $\mathcal{P}$, hence called the *maximal induced $\mathcal{P}$ subgraphs* problem. Trivially, one can always use an enumeration algorithm to solve the optimization version of the same problem. A classical example of nontrivial use is to color a graph by enumerating its maximal independent sets [38, 27, 9, 6]. Indeed, the enumeration of maximal independent sets and the enumeration of maximal cliques are practically the same, and both are well-studied classic problems [3, 17, 43]. They were also used in finding connected vertex covers and edge dominating sets [28]. In general, the maximal induced $\mathcal{P}$ subgraphs problem is motivated by the intrinsic difficulty in formulating a combinatorial optimization problem: there are always factors that are ill characterized or even omitted in the formulation [26, 44].[1] This is particularly the case for vertex deletion problems, of which the primary applications are the processing of noisy data, where modifications to a graph are meant to exclude outliers or to fix noise.

Motivations of maximal induced $\mathcal{P}$ subgraphs problems also come from database theory [18], where one is usually concerned with only induced $\mathcal{P}$ subgraphs that are connected. Of any hereditary graph class $\mathcal{P}$, we can define a subclass by allowing only connected graphs in $\mathcal{P}$. With very few exceptions, this naturally defined class is not hereditary, and hence this variation, called the *maximal connected induced $\mathcal{P}$ subgraphs* problem, poses different challenges. Indeed, as we will see, it is somewhat more difficult than the original one.

Let n denote the number of vertices in the input graph, and by a solution we mean a maximal vertex set that induces a subgraph in $\mathcal{P}$. Since there might be an exponential number (on n) of *solutions*, care needs to be taken when we talk about the running time of an enumeration algorithm. For example, in a graph consisting of disjoint triangles, there are $3^{n/3}$ maximal independent sets. Johnson et al. [37] defined three complexity classes for enumeration algorithms, namely, polynomial total time (polynomial in n and the total number of solutions), incremental polynomial time (for all s, the time to output the first s solutions is polynomial in n and s), and polynomial delay (the time to output the first solution and the time between two consecutive solutions are both polynomial in n). Both maximal independent sets and maximal cliques can be enumerated with polynomial delay, and so are maximal bicliques (complete bipartite graphs) [23, 33] and maximal forests [45]. See the survey [48] and the recent results [21, 19].

2 Our contributions

Our algorithms are summarized below.

▶ **Theorem 1.** *The maximal induced $\mathcal{P}$ subgraphs problem and its connected variation can be solved with polynomial delay when $\mathcal{P}$ is one of the following graph classes: interval graphs, trivially perfect graphs, split graphs, complete split graphs, pseudo-split graphs, threshold graphs, cluster graphs, complete bipartite graphs, complete p-partite graphs, and d-degree-bounded graphs.*

▶ **Theorem 2.** *The maximal induced $\mathcal{P}$ subgraphs problem and its connected variation can be solved in incremental polynomial time when $\mathcal{P}$ is one of the following graph classes: wheel-free graphs, unit interval graphs, block graphs, 3-leaf powers, basic 4-leaf powers, and any graph class that can be characterized by a finite set of forbidden induced subgraphs.*

Since a connected cluster graph is a clique, the result for the maximal connected induced cluster subgraphs problem is already known. Also, for a graph class that can be characterized by a finite set of forbidden induced subgraphs, algorithms for the maximal induced $\mathcal{P}$ subgraphs problem, but not its connected variation, can be derived from Eiter and Gottlob [24] (see the discussion in Section 2.6).

[1] After all, how many times do you click "I'm Feeling Lucky" on Google.com?

2.1 Solution maps and retaliation-free paths

In a seminal work, Schwikowski and Speckenmeyer [45] proposed an algorithm for enumerating maximal induced forests, as well as its directed variation, enumerating maximal induced directed acyclic subgraphs. They introduced a successor function, which, given a solution, returns a nonempty multi-set of other solutions of G. In so doing they *implicitly* built a multiple digraph $\mathcal{M}(G)$ whose nodes are the solutions, and there are arc(s) from node S_1 to node S_2 if S_2 is one of the successors of S_1. The key properties of $\mathcal{M}(G)$ are (1) the successor function can be calculated in polynomial time; and (2) $\mathcal{M}(G)$ is strongly connected. As a result, traversing $\mathcal{M}(G)$ from any node, we can visit all the solutions in time polynomial in n and linear on the number of solutions. With a standard bookkeeping mechanism, we can easily implement it with polynomial delay. We call the multiple digraph on the solutions of an enumeration problem its *solution map.* Gély et al. [33] later rediscovered this idea, and used it to re-analyze enumeration algorithms for maximal cliques and maximal bicliques. Solution maps are actually very general and powerful. Conte et al. [21, 19] built solution maps to solve the maximal (connected) induced $\mathcal{P}$ subgraphs problem for several important graph classes. In particular, they solved the connected variation for forests, and directed acyclic graphs. (They also designed algorithms for enumerating maximal subgraphs, a direction that we will not pursue in the present paper, and other non-graphic problems.)

To solve an enumeration problem with a solution map consists in defining the successor function and proving that the implied solution map $\mathcal{M}$ is strongly connected. All the mentioned algorithms follow the same general scheme, although the details are quite problem specific. For convenience, we may assume that the solutions are subsets of some ground set U; for the maximal (connected) induced $\mathcal{P}$ subgraphs problem, U is the vertex set of the input graph.

Successors of a solution S: For each $v \in U \setminus S$, define a sub-instance restricted to $S \cup \{v\}$, and find a set of solutions of this sub-instance. The union of these $|U \setminus S|$ sets makes the successors of S.

Strong connectedness: For each solution S^*, define a specific metric and show that every other solution S has a successor "closer" to S^* than S with respect to this metric.

Since a solution S' can be a successor of S by virtue of two different elements in $U \setminus S$, the solution map is a multiple digraph. The simplest metric is arguably the lexicographical order. If we number the elements such that those in S^* are the smallest, then S^* is the smallest among all solutions, and hence it suffices to show that each solution S has a lexicographically smaller successor. The existence of a path from S to S^* in $\mathcal{M}$, hence the strong connectedness of $\mathcal{M}$, will then follow from the finiteness of the ground set U. Other metrics have also been used in literature [45, 18, 21, 19]

The requirement that every other solution has a *direct* successor that is closer to S^*, with respect to the metric decided by S^*, is not always easy to achieve. For example, such successor functions have been claimed for unit interval graphs and interval graphs, but both turned out to be incorrect. For the connectedness of a solution map, after all, we are only concerned with the reachability: whether a solution can reach S^* instead of how long it takes to do so. (Indeed, in "the most ideal case," the solution map can be a simple directed cycle, in which the distance of a pair of solutions can be the number of solutions minus one.)

We introduce retaliation-free paths to certify strong connectedness of a solution map. Suppose that we are using the lexicographic metric. In our framework, it is possible that all the successors of a solution S are lexicographically larger than S. Thus, the strong connectedness of our solution maps is not readily guaranteed, and we proceed as follows. Let s be the smallest element in $S^* \setminus S$. If any successor of S contains $[1..s]$, then it is lexicographically smaller than S and we are done. Otherwise, we look for a solution S' that contains $[1..s]$ and is reachable from $S_0 = S$ by a nontrivial path $S_0 S_1 \cdots S'$. It is better to view this path as a transforming procedure. In the first step, we choose a successor S_1 of S containing s; we call s the *gainer*, and elements in $[1..s] \setminus S_1$ the *victims* of s. We then try to add the victims of s back, with the guarantee that none of them unseats s, an action which we call *retaliation.* A victim r of s may become a gainer in a later step, hence introducing further victims. In this case, we put other victims of s on hold, and try to restore the victims of r. Now we need to make sure both r and s are safe. Thus, we consider the gainer–victim relationship transitive. Such a path of solutions in $\mathcal{M}$ is called *retaliation-free*; see, e.g., Figure 1. A retaliation-free path terminates only when it reaches S^*. Therefore, it suffices to show that it does terminate. (After we reach a solution containing $[1..s]$, the element s is no longer a gainer, and may be removed as a victim for adding a later element. The path from S to S^* we produced as such can have a length super-polynomial in n.)

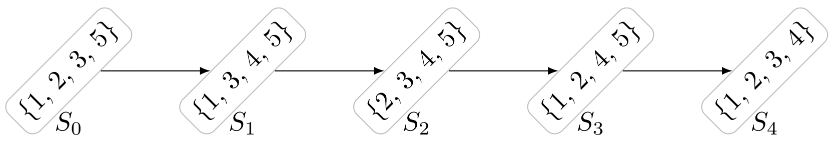


Figure 1 A path from S_0 to $S_4 = S^*$ in a solution map. Here $U = \{1, 2, 3, 4, 5\}$, and its elements are numbered such that those in S^* have the smallest numbers. In the first step ($S_0 \to S_1$), element 4 is the gainer and element 2 is the victim, which becomes the gainer of the second step ($S_1 \to S_2$), with victim 1. The last two steps have no victims.

▶ **Theorem 3** (Informal). *Let $\mathcal{M}$ be the solution map of an enumeration problem, and let S^* be a specific solution. Every retaliation-free path in $\mathcal{M}$ with respect to S^* terminates at S^*.*

We use Theorem 3 to develop enumeration algorithms for the maximal (connected) induced trivially perfect subgraphs problem and the maximal (connected) induced interval subgraphs problem, both running with polynomial delay. The two problems are paradigmatic for the use of this technique. The main structures we use for interval graphs are the clique paths, which are linear, while for trivially perfect graphs, we work on their generating forests, which are hierarchical.

A trivially perfect graph G can be represented as a forest F, called its generating forest, such that two vertices u and v are adjacent in G if and only if one of them is an ancestor of the other in F [49, 50]. It is simpler if we consider the connected variation, where generating forests are trees. For the purpose of transforming a solution S to another solution S^*, we transform a generating tree T of $G[S]$ to a generating tree T^* of $G[S^*]$. A main obstacle is that two vertices in $S \cap S^*$ may have different ancestor–descendant relationship in T and T^*; see, e.g., vertices 2 and 3 in Figure 2.

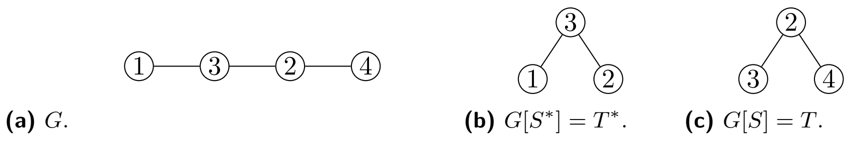
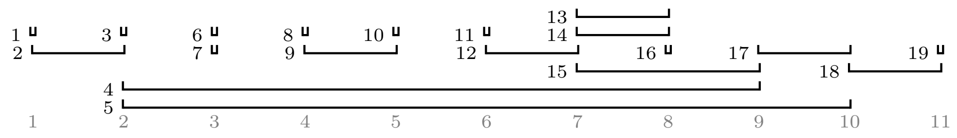


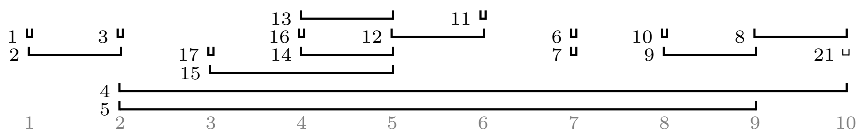
Figure 2 A graph G and its only two maximal connected induced trivially perfect subgraphs.

We add the vertices in $S^* \setminus S$ bottom-up, i.e., children before parents. Let v be the smallest vertex in $S^* \setminus S$ in the post-order of T^*; note that all the T^*-descendants of v are in S. It is easy to add v if v is a leaf of T^* (because all its T^*-ancestors have larger numbers than v) or if v has a single child v' in T^* (because v and v' are true twins in $G[S^*]$). The nontrivial case is when v has multiple T^*-children. Since we can only make a polynomial number of solutions to be successors of S, it is very likely that none of the successors keep all the T^*-children of v. We observe that as long as we keep two T^*-children of v, then the relationship between T^*-descendants of v and T^*-ancestors of v will be correctly maintained. We afford to lose other children of v and their descendants as victims of this step, because they can be added back easily: all the operations for adding these victims back are under v, hence isolated from other parts.

Interval graphs are a very important and well-studied graph class [8, 32, 34, 35], and the interval vertex deletion problem receives a lot of attention [7, 14, 11, 2]. The maximal cliques of an interval graph can be arranged in a linear manner, called a clique path [32]. A natural idea is to take a clique path for $G[S^*]$ and number the vertices from left to right; see, e.g., Figure 3(a). Then for another solution S, we try to add the smallest vertex in $S^* \setminus S$. However, for solution S shown in Figure 3(b), there is no obvious way to add vertex 18 to S, without removing a smaller vertex. The reason is that intervals for vertices 6–17 are organized in very different ways in the two clique paths. For a reader familiar with the recognition of interval graphs, the issue is essentially the same as recognizing interval graphs by graph searches, scanning the vertices from one end to the other end [13]. Note that $\{8, 9, 10\}$ is a *module* (a subset of vertices that have the same neighborhood outside this subset) of $G[S^*]$ and switching the positions of vertices 8 and 10 gives another valid representation of $G[S^*]$. On the other hand, the set $\{6, \ldots, 17\}$ is not a module of $G[S^*]$, and its orientation is fixed from the right by 18. Since this fact can only be deduced after seeing vertex 18, it is not available if one scans from the left. The algorithm in [13] tries to detect and keep track of the orientation of such vertex sets via multiple-sweep searches. For our problem, we allow some vertices in $\{6, \ldots, 17\}$ to fall victims to vertex 18. Afterward, we restore the victims *from right to left*: we process them in a different direction because vertex 18 is the anchor of this set.



(a) S^*.



(b) S.

Figure 3 Two maximal connected induced interval subgraphs of some graph G (other vertices are immaterial and omitted). Gray numbers at the bottom indicate the indices of the maximal cliques.

2.2 Algorithms in incremental polynomial time

In each of the mentioned algorithms, the successor function has to be handcrafted. For example, our successor function for interval graphs does not work for unit interval graphs. Nor can the successor function of [21, 19] for chordal graphs be applied to interval graphs.[2] In the following we aim for general approaches that can be applied with little efforts.

There is nothing inherent in solution maps about polynomial delay, though all of the mentioned algorithms based on solution maps are of this type. The running time of such an algorithm is determined by the successor function, which is almost always determined by the number of successors a solution can have. If the number is polynomial in n, then the algorithm has polynomial delay. If we allow the number to be polynomial in both n and the number of solutions, then it needs polynomial total time. One "weakness" of the approach based on solution maps is the lack of a simple way to develop algorithms in incremental polynomial time, save those simple classes [18]. We obviate this concern with the following observation.

▶ **Theorem 4.** *For any hereditary graph class $\mathcal{P}$, the maximal (connected) induced $\mathcal{P}$ subgraphs problem can be solved in polynomial total time if and only if it can be solved in incremental polynomial time.*

For the maximal induced $\mathcal{P}$ subgraphs problem, the statement follows from the classical result of Bioch and Ibaraki [5]; see discussions in Section 2.6. As far as we can check, the statement for the connected variation was not previously known. (This explains why Cohen et al. [18] had to prove the statement separately.) Theorem 4 enables us to use solution maps transparently in developing algorithms for enumerating maximal (connected) induced $\mathcal{P}$ subgraphs in incremental polynomial time. There is strong evidence that a similar claim on general enumeration problems does not hold [46, 16].

2.3 Restricted versions

Lawler et al. [39] introduced the *input-restricted version* of an enumeration problem: there exists an element $x \in U$ such that $U \setminus \{x\}$ is a solution. This special problem arises naturally in several design paradigms for enumeration problems, e.g., the design of successor functions, and the reverse search technique [4]. Cohen et al. [18] proved that the maximal (connected) induced $\mathcal{P}$ subgraphs problem can be solved with polynomial delay when its input-restricted version can be solved in polynomial time. Moreover, the original problem can be solved in incremental polynomial time if and only if its input-restricted version can. They also proved a similar statement for polynomial total time, which is rendered redundant by Theorem 4. Thus, this ostensibly simpler version is the core of the maximal (connected) induced $\mathcal{P}$ subgraphs problem in terms of solvability in incremental polynomial time.

Since the appearance of [18], there have been attempts at extending its core idea. A "natural" way seems to be defining a restricted version that is equipped with a special set of more than one vertex; i.e., $G - Z$ is in $\mathcal{P}$ for some set Z of t vertices, where $t > 1$. However,

[2] An early version of [19] mistakenly claimed that their algorithm for the maximal induced chordal subgraphs problem also applies to interval graphs. Roughly speaking, what they did is to reconstruct a perfect elimination ordering of a desired solution S^* from a solution S. As long as the newly added vertex is simplicial, the resulting graph is chordal. But it is not always an interval graph. The tent graph (adding three edges to connect the even-number vertices of a 6-cycle) is a counterexample. In the solution map they built, the only successor of $V(G) \setminus \{v\}$, where v is any degree-2 vertex of G, is G itself, which is not an interval graph. It is quite nontrivial, if possible at all, to adapt this approach to interval graphs.

the extra $t-1$ vertices in the set Z turn out to be not helpful. Consider, for instance, $\mathcal{P}$ being the forests. For a graph G and any t, we can make a new graph H by introducing $t-1$ copies of disjoint triangles to G, and then H satisfies the new definition if and only if G is input-restricted. Therefore, this special version does not have any property that is not already in the input-restricted version, hence not easier to solve than the latter.

This negative example does not suggest a dead end, and there is a natural generation that does work. We may view the input-restricted version as an enumeration problem by itself, and try to build a solution map to solve it. Then in designing the successor function, in the sub-instance restricted to $S \cup \{v\}$, aside from the vertex x we had, we are bestowed with another vertex v such that the removal of either of x and v leaves a subgraph in $\mathcal{P}$. We are thus inspired to define the *t-restricted version* of the maximal induced $\mathcal{P}$ subgraphs problem:

There exists a set Z of t vertices in G such that $G - z$ is in $\mathcal{P}$ for *every* $z \in Z$.

It is equivalent to that *every* induced subgraph of G that is not in $\mathcal{P}$ contains *all* vertices in Z. This requirement is far stronger than $G - Z$ being in $\mathcal{P}$, though equivalent when $t = 1$. We are able to show that even this far more restricted version is still equivalent to the original problem in terms of enumerability in incremental polynomial time. We remark that our proofs, based on solution maps, are significantly simpler than those on the input-restricted version [18], which can now be viewed as the 1-restricted version. With the benefit of hindsight, we can see that all the results of [18] can be easily interpreted using solution maps, with simpler proofs.

▶ **Theorem 5.** *Let $\mathcal{P}$ be a hereditary graph class. The maximal (connected) induced $\mathcal{P}$ subgraphs problem can be solved in incremental polynomial time if and only if there exists a positive integer t such that its t-restricted version can be solved in polynomial total time.*

The strong requirement stipulated in defining the t-restricted version makes it significantly easier than the original problem. All the algorithms in Theorem 2 are obtained by reducing these problems to their t-restricted versions. Of these results, we would like to draw special attention to those on wheel-free graphs, though this graph class in its own sense may seem to be less interesting compared to others we study. Lokshtanov [41] proved that the vertex deletion problem toward wheel-free graphs is W[2]-hard with respect to standard parameterization, hence very unlikely fixed-parameter tractable. This suggests that the complexity of the maximal induced $\mathcal{P}$ subgraphs problem can be quite different from its optimization counterpart.

Theorem 5 also implies, among others, the following result on graph classes that can be characterized by a finite set $\mathcal{F}$ of forbidden induced subgraphs. Indeed, with t being the maximum order of graphs in $\mathcal{F}$, the t-restricted version of the maximal (connected) induced $\mathcal{F}$-free subgraphs problem is trivial. Eiter and Gottlob [24] have proved this result for the maximal induced $\mathcal{F}$-free subgraphs problem, but their proof does not apply to the connected variation; see discussions in Section 2.6.

▶ **Corollary 6.** *Let $\mathcal{F}$ be a finite set of graphs. The maximal (connected) induced $\mathcal{F}$-free subgraphs problem can be solved in incremental polynomial time.*

We note that the vertex deletion problem to $\mathcal{F}$-free graphs for finite $\mathcal{F}$ has been well studied. In particular, they are fixed-parameter tractable [10] and admit constant-approximation [42].

2.4 Reductions between the two variations

Since we are dealing with both the maximal induced $\mathcal{P}$ subgraphs problem and its connected variation, it is really irksome if we have to develop two algorithms for each class. Although Cohen et al. [18] and Conte et al. [21, 19] dealt with both variations, they stopped at noting that with one of them solved, a slight modification would be able to solve the other. These modifications have to be done, however, case by case.

The main issue of the connected variation is that the set of connected graphs in a hereditary graph class $\mathcal{P}$, viewed as a graph class in its own regard, is mostly not hereditary. If there are two nonadjacent vertices in any connected graph G in $\mathcal{P}$, then the edgeless graph on two vertices is an induced subgraph of G, hence in $\mathcal{P}$. Therefore, the class of connected graphs in $\mathcal{P}$ remains hereditary only when $\mathcal{P}$ is the class of cluster graphs, in which the connected ones are the complete graphs, and the class of edgeless graphs. A connected edgeless graph has precisely one vertex, and this seems to be the only class on which the connected variation, which is trivial, is the easier between the two variations. For a hereditary graph class $\mathcal{P}$, and a graph G, let us use N_1 and N_2 to denote, respectively, the number of maximal induced $\mathcal{P}$ subgraphs and the number of maximal connected induced $\mathcal{P}$ subgraphs of G. It is not difficult to see that N_1/N_2 can be an exponential number on n, while N_2/N_1 can never be more than n. In the trivial case when $\mathcal{P}$ is edgeless, N_2 is precisely n, while N_1 can be $3^{n/3}$. For a nontrivial example, the graph consisting of $n/4$ disjoint 4-cycles has $2^{n/2}$ maximal induced forests, while only n maximal induced trees. This example applies to interval graphs, chordal graphs, and many others. Therefore, it is very unlikely we can use an enumeration algorithm for the maximal induced $\mathcal{P}$ subgraphs problem to solve its connected variation.

The other direction is promising. Two simple observations help here. First, if we can add vertices to a graph without making new forbidden induced subgraphs, then it does not change the number of maximal induced $\mathcal{P}$ subgraphs, though each of them gets more vertices. Second, if the extra vertices make all the maximal induced $\mathcal{P}$ subgraphs connected, then by enumerating all maximal connected induced $\mathcal{P}$ subgraphs of the new graph, we obtain effortlessly all maximal induced $\mathcal{P}$ subgraphs of the original graph. This idea works when $\mathcal{P}$ is closed under adding universal vertices, examples of which include interval graphs, chordal graphs, etc. A less trivial reduction can be devised for several other graph classes, e.g., wheel-free graphs and triangle-free graphs. These reductions focus us on the connected variation of the problems only, instead of working on both.

Via Theorem 5, we are able to establish a more interesting connection. Although the condition in the statement looks rather technical, it is satisfied by many natural graph classes: apart from those in Theorem 2, another notable example is planar graphs. As for the t-restricted version of both variations, there are only a polynomial number of solutions that are not a proper superset of Z, and they are easy to handle. Therefore, the focus is on those solutions containing all vertices in Z. We observe that under the connectivity condition, in every maximal induced $\mathcal{P}$ subgraph of G that contains all vertices in Z, all the vertices in Z are always in the same component. As a result, there is a one-to-one mapping between such solutions for these two variations.

▶ **Theorem 7.** *Let c be a constant. Let $\mathcal{F}$ be a set of graphs such that every graph in $\mathcal{F}$ of order c or above is biconnected. The maximal induced $\mathcal{F}$-free subgraphs problem can be solved in incremental polynomial time if and only if its connected variation can be solved in incremental polynomial time.*

2.5 Characterizing the easy classes

We try to better understand maximal (connected) induced $\mathcal{P}$ subgraphs problems that can be solved with polynomial delay by considering its input-restricted version. We say that a graph class $\mathcal{P}$ has the *CKS property*, after the initials of the authors of [18], if the input-restricted version of the maximal (connected) induced $\mathcal{P}$ subgraphs problem can be solved in time polynomial only in n. Since it is straightforward to see that the class of edgeless graphs (independent sets), the class of complete graphs (cliques), and the class of complete bipartite graphs (bicliques) have the CKS property, the polynomial-delay algorithms for them can be immediately explained by this general result. On the other hand, the polynomial-delay algorithms of [45] and [21, 19], and our polynomial-delay algorithms for the maximal (connected) induced interval subgraphs problem and for the maximal (connected) induced trivially perfect subgraphs problem cannot be derived from this observation. As we will see, none of these graph classes has the CKS property.

It turns out that star forests, forests in which every tree is a star, play a crucial role in characterizing graph classes with the CKS property. Therefore, to find those graph classes with the CKS property, it suffices to consider those forbidding both a star forest and the complement of a star forest.

▶ **Theorem 8.** *Let $\mathcal{F}$ be a nonempty set of graphs. If the class of $\mathcal{F}$-free graphs has the* CKS *property, then $\mathcal{F}$ contains at least one star forest and the complement of at least one star forest.*

Unfortunately, this condition is not sufficient, and many graph classes satisfy this condition but do not have the CKS property. Moreover, it is possible that a class $\mathcal{P}$ of graphs has the CKS property but a proper subclass of $\mathcal{P}$ does not. This fact makes a full characterization of graph classes with the CKS property more difficult.

2.6 Related work

Let us put our work into context. The aforementioned characterization of a hereditary graph class by a set $\mathcal{F}$ of forbidden induced subgraphs provides another perspective to view the maximal induced $\mathcal{P}$ subgraphs problem, but not its connected variation in general. A subset S of vertices is a solution if and only if $V(G) \setminus S$ intersects all *forbidden sets* of G, i.e., a minimal set X of vertices such that $G[X] \in \mathcal{F}$. If we list all the forbidden sets of G as a set system, then the problem becomes enumerating *minimal* vertex sets that intersect each of the forbidden sets. This brings us to the well-studied problem of enumerating minimal hitting sets of a set system. A set system is also known as a *hypergraph*, and the problem is called *hypergraph transversal* in literature. Some important general results on enumeration of maximal induced $\mathcal{P}$ subgraphs, e.g., the algorithm for graph classes characterized by a finite number of forbidden induced subgraphs, were first proved in this setting [24]. Reducing to the hypergraph transversal problem was also a common approach used by many earlier heuristic enumeration algorithms, for maximal independent sets and for maximal forests. A graph may have an exponential number of simple cycles, and thus the maximal induced forests problem and its associated hypergraph transversal problem have significantly different input sizes: the latter, including the number of elements and the number of forbidden sets, may be exponential on the former. This can happen for all graph classes $\mathcal{P}$ that have an infinite number of forbidden induced subgraphs. Deciding whether a graph belongs to such a class $\mathcal{P}$ may be NP-hard, and hence the maximal induced $\mathcal{P}$ subgraphs problem cannot be solved in polynomial total time, though it is still possible for the corresponding hypergraph transversal problem, with all the forbidden sets given. Thus, we have to exclude from our study those graph classes that cannot be recognized in polynomial time. For a graph class that can be recognized in polynomial time, the maximal (connected) induced $\mathcal{P}$ subgraphs problem is in EnumP, the counterpart of NP for enumeration [16]. (One may consider the oracle model, where whether a graph is in the class is answered by an oracle in $O(1)$ time, but this paper will not take this direction.)

The (minimal) hitting sets of a hypergraph H define another hypergraph H', and it is an easy exercise to verify that each set in H is a (minimal) hitting set of H'. The hypergraph transversal problem has a decision version: given a pair of hypergraphs on the same set of elements, decide whether one consists of exactly the minimal hitting sets of the other. Bioch and Ibaraki [5] showed that the hypergraph traversal problem can be solved in polynomial total time if and only if it can be solved in incremental polynomial time, by showing that this is the case precisely if the decision version can be solved in polynomial time. Note that under certain complexity assumptions, these two complexity classes for enumeration problems are not equal in general [46, 16].

The results of [5] were actually developed in the setting of monotone Boolean functions. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $x \leq y$ always implies $f(x) \leq f(y)$. The *identification* problem is to find all minimal true vectors *and* all maximal false vectors of f. It is easy to see that the maximal induced $\mathcal{P}$ subgraphs problem is a special case of it. Let G be a graph on n vertices, for any subset $X \subseteq V(G)$, we use x to denote the characteristic vector of X, i.e., an n-dimension Boolean vector with 1 at the ith position if and only if $v_i \in X$, and set $f(x) = 0$ if and only if $G[X] \in \mathcal{P}$. Then minimal forbidden sets and the vertex sets of maximal induced $\mathcal{P}$ subgraphs of G correspond to minimal true vectors and maximal false vectors of f, respectively. In this terminology, the maximal induced $\mathcal{P}$ subgraphs problem asks for maximal false vectors only, hence different in the output size. (Moreover, the identification problem is usually asked in the oracle model.) The decision version of the identification problem, known as dualization of monotone Boolean functions, is equivalent to the hypergraph traversal problem, and its incarnations can be found in database systems, artificial intelligence, and game theory. Fredman and Khachiyan [31] proved that dualization of monotone Boolean functions, hence hypergraph traversal, can be solved in quasi-polynomial time. It has been open for nearly four decades whether this problem can be solved in polynomial time; see the survey of Eiter et al. [25].

The main motivation of studying the connected variation is its practical applications in database theory, where several important problems can be modeled as enumerating maximal connected induced $\mathcal{P}$ subgraphs. We refer to Cohen et al. [18] and references therein for the background. Since this variation cannot be directly cast into hypergraphs or monotone Boolean functions, fewer results on it have been known in literature [18, 21, 19], and they had to be dealt with separately.

We are exclusively concerned with maximal (connected) induced $\mathcal{P}$ subgraphs. There is also research on enumerating all induced $\mathcal{P}$ subgraphs and enumerating all maximum induced $\mathcal{P}$ subgraphs. The first is usually very easy, if not completely trivial, while the latter has to be very hard: after all, finding a single maximum induced subgraph in a nontrivial and hereditary graph class is already NP-hard [40]. Yet another, probably more practical, approach is to list top k solutions, or solutions of costs at most (or at least) k. This is frequently studied in the framework of parameterized computation. There is work using the input size as the sole measure for the running time of enumeration algorithms, e.g., [47]. In summary, efforts toward a systematic understanding of enumeration are gaining momentum [22].

Our final remark is on the space usage. A naïve implementation of our main approaches needs to keep track of all the solutions, hence takes exponential space. How to reduce the space is the major open problem in this area, and positive results are few but growing. Using reverse search, Conte et al. [20] showed how to strengthen polynomial-delay algorithms of [18] to use only polynomial space. Conte et al. [21, 19] also managed to reduce some of their algorithms to polynomial space.

References

1 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. *ACM Transactions on Algorithms*, 15(1):11:1–11:28, 2019. doi:10.1145/3284356.

2 Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In Timothy M. Chan, editor, *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1711–1730. SIAM, 2019. doi:10.1137/1.9781611975482.103.

3 E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2(1):1–6, 1973. doi:10.1137/0202001.

4 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

5 Jan C. Bioch and Toshihide Ibaraki. Complexity of identification and dualization of positive boolean functions. *Information and Computation*, 123(1):50–63, 1995. doi:10.1006/inco.1995.1157.

6 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009. doi:10.1137/070683933.

7 Ivan Bliznets, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Largest chordal and interval subgraphs faster than 2^n. *Algorithmica*, 76(2):569–594, 2016. doi:10.1007/s00453-015-0054-2.

8 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

9 Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, 2004. doi:10.1016/j.orl.2004.03.002.

10 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

11 Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1115. SIAM, 2016. Full version available at arXiv:1403.1515. doi:10.1137/1.9781611974331.ch77.

12 Yixin Cao. Unit interval editing is fixed-parameter tractable. *Information and Computation*, 253:109–126, 2017. doi:10.1016/j.ic.2017.01.008.

13 Yixin Cao. Recognizing (unit) interval graphs by zigzag graph searches. In Hung Viet Le and Valerie King, editors, *Proceedings of the 4th SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 92–106. SIAM, 2021. doi:10.1137/1.9781611976496.11.

14 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.

15 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/s00453-015-0014-x.

16 Florent Capelli and Yann Strozecki. Incremental delay enumeration: Space and time. *Discrete Applied Mathematics*, 268:179–190, 2019. doi:10.1016/j.dam.2018.06.038.

17 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985. doi:10.1137/0214017.

18 Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147–1159, 2008. doi:10.1016/j.jcss.2008.04.003.

19 Alessio Conte, Roberto Grossi, Andrea Marino, Takeaki Uno, and Luca Versari. Proximity search for maximal subgraph enumeration. *SIAM Journal on Computing*, 51(5):1580–1625, 2022. doi:10.1137/20m1375048.

20 Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. Listing maximal subgraphs satisfying strongly accessible properties. *SIAM Journal on Discrete Mathematics*, 33(2):587–613, 2019. doi:10.1137/17M1152206.

21 Alessio Conte and Takeaki Uno. New polynomial delay bounds for maximal subgraph enumeration by proximity search. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 1179–1190. ACM, 2019. doi:10.1145/3313276.3316402.

22 Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. A complexity theory for hard enumeration problems. *Discrete Applied Mathematics*, 268:191–209, 2019. doi:10.1016/j.dam.2019.02.025.

23 Vânia M. F. Dias, Celina M. H. de Figueiredo, and Jayme Luiz Szwarcfiter. Generating bicliques of a graph in lexicographic order. *Theoretical Computer Science*, 337(1-3):240–248, 2005. doi:10.1016/j.tcs.2005.01.014.

24 Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, 1995. doi:10.1137/S0097539793250299.

25 Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008. doi:10.1016/j.dam.2007.04.017.

26 David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. doi:10.1137/S0097539795290477.

27 David Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003. doi:10.7155/jgaa.00064.

28 Henning Fernau. Edge dominating set: Efficient enumeration-based exact algorithms. In Hans L. Bodlaender and Michael A. Langston, editors, *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 4169 of *LNCS*, pages 142–153. Springer, 2006. doi:10.1007/11847250_13.

29 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *Journal of the ACM*, 66(2):8:1–8:23, 2019. doi:10.1145/3284176.

30 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM Journal on Computing*, 44(1):54–87, 2015. doi:10.1137/140964801.

31 Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, 1996. doi:10.1006/jagm.1996.0062.

32 Delbert R. Fulkerson and Oliver A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965. doi:10.2140/pjm.1965.15.835.

33 Alain Gély, Lhouari Nourine, and Bachir Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009. doi:10.1016/j.dam.2008.10.010.

34 Wen-Lian Hsu. $O(m{\cdot}n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995. doi:10.1137/S0097539793260726.

35 Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999. doi:10.1137/S0097539792224814.

36 Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. In Philip N. Klein, editor, *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1399–1418. SIAM, 2017. doi:10.1137/1.9781611974782.91.

37 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988. doi:10.1016/0020-0190(88)90065-8.

38 Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5(3):66–67, 1976. doi:10.1016/0020-0190(76)90065-X.

39 Eugene L. Lawler, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9(3):558–565, 1980. doi:10.1137/0209042.

40 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

41 Daniel Lokshtanov. Wheel-free deletion is W[2]-hard. In Martin Grohe and Rolf Niedermeier, editors, *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 5018 of *LNCS*, pages 141–147, New York, 2008. Springer. doi:10.1007/978-3-540-79723-4_14.

42 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming (ICALP)*, volume 700 of *LNCS*, pages 40–51. Springer, 1993. doi:10.1007/3-540-56939-1_60.

43 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In Torben Hagerup and Jyrki Katajainen, editors, *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory, SWAT 2004*, volume 3111 of *LNCS*, pages 260–272. Springer, 2004. doi:10.1007/978-3-540-27810-8_23.

44 Andrea Marino. *Analysis and Enumeration: Algorithms for Biological Graphs*, volume 6 of *Atlantis Studies in Computing*. Atlantis, 2015. doi:10.2991/978-94-6239-097-3.

45 Benno Schwikowski and Ewald Speckenmeyer. On enumerating all minimal solutions of feedback problems. *Discrete Applied Mathematics*, 117(1-3):253–265, 2002. doi:10.1016/S0166-218X(00)00339-5.

46 Yann Strozecki. *Enumeration complexity and matroid decomposition*. PhD thesis, Université Paris Diderot – Paris 7, Paris, France, 2010.

47 Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006. doi:10.1016/j.tcs.2006.06.015.

48 Kunihiro Wasa. Enumeration of enumeration algorithms. arXiv:1605.05102, 2016.

49 E. S. Wolk. The comparability graph of a tree. *Proceedings of the American Mathematical Society*, 13:789–795, 1962. doi:10.1090/S0002-9939-1962-0172273-0.

50 Jing-Ho Yan, Jer-Jeong Chen, and Gerard Jennhwa Chang. Quasi-threshold graphs. *Discrete Applied Mathematics*, 69(3):247–255, 1996. doi:10.1016/0166-218X(96)00094-7.

51 Jie You, Jianxin Wang, and Yixin Cao. Approximate association via dissociation. *Discrete Applied Mathematics*, 219:202–209, 2017. doi:10.1016/j.dam.2016.11.007.