Polynomial Delay Algorithm for Listing Minimal Edge Dominating sets in Graphs

Mamadou Moustapha Kanté[1], Vincent Limouzy[1], Arnaud Mary[2], Lhouari Nourine[1], and Takeaki Uno[3]

[1] Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
[2] Université de Lyon, Université Lyon 1, UMR CNRS 5558, LBBE, INRIA Erable, France
[3] National Institute of Informatics, Japan

Abstract. It was proved independently and with different techniques in [Golovach et al. - ICALP 2013] and [Kanté et al. - ISAAC 2012] that there exists an incremental output polynomial algorithm for the enumeration of the minimal edge dominating sets in graphs, *i.e.*, minimal dominating sets in line graphs. We provide the first polynomial delay and polynomial space algorithm for the problem. We propose a new technique to enlarge the applicability of Berge's algorithm that is based on skipping hard parts of the enumeration by introducing a new search strategy. The new search strategy is given by a strong use of the structure of line graphs, and we hope is of great interest and could be used to get new output polynomial algorithms.

1 Introduction

The Minimum Dominating Set problem is a classic and well-studied graph optimization problem. A *dominating set* in a graph G is a subset D of its set of vertices such that each vertex is either in D or has a neighbor in D. Computing a minimum dominating set has numerous applications in many areas, *e.g.*, networks, graph theory (see for instance the book [10]). The Minimum Edge Dominating Set problem is a classic well-studied variant of the Minimum Dominating Set problem [10]. An edge dominating set is a subset F of the edge set such that each edge is in F or is adjacent to an edge in F. In this paper, we are interested in an output polynomial algorithm for listing without duplications the (inclusion-wise) minimal edge dominating sets of a graph. An output polynomial algorithm is an algorithm whose running time is bounded by a polynomial depending on the sum of the sizes of the input and output. The enumeration of minimal or maximal subsets of vertices satisfying some property in a (hyper)graph is a central area in graph algorithms and for several properties output polynomial algorithms have been proposed *e.g.* [1,4,6,7,16,19], while for others it was proven that no output polynomial time algorithm exists unless P=NP [15,16].

The existence of an output polynomial algorithm for the enumeration of minimal dominating sets of graphs (Dom-Enum problem) is a widely open question and is closely related to the well-known TRANS-ENUM problem in hypergraphs which asks for an output polynomial algorithm for the enumeration of all minimal transversals in hypergraphs. A *transversal* (or a *hitting-set*) in a hypergraph is a subset of its vertex set that intersects every of its hyper-edges. This is a long-standing open problem (see for instance [5]) and is well-studied due to its applications in several areas [5,6,9]. Up to now only few tractable cases are known (see [12] for some examples). It is easy to see that the minimal dominating sets of a graph are the minimal transversals of its *closed neighbourhoods*[4], and then, as a particular case, it seems that the DOM-ENUM problem is more tractable than the TRANS-ENUM problem, but some of the authors have very recently proved in [12] that the TRANS-ENUM problem can be polynomially reduced to the DOM-ENUM problem. They also investigate the enumeration of minimal dominating sets in the perspective of graph theory and exhibit several new tractable cases, split graphs [12], undirected path graphs [11], interval and permutation graphs [13] and chordal P_6-free graphs [12]. In particular, they prove that the enumeration of minimal edge dominating sets can be done in (incremental) output polynomial (a result obtained independently by Golovach et al. [8]). Many enumeration problems admit an output polynomial algorithm but no polynomial delay algorithm [**?**, Section 2.3], and so a natural question is whether one can enumerate with polynomial delay all the minimal edge dominating sets, and we answer positively to this question in this paper.

Further, some of the aforementioned tractable cases of the TRANS-ENUM problem are based on Berge's algorithm [2]. Berge's algorithm consists in ordering the hyper-edges $E_1, \dots, E_m$ of a given hypergraph $\mathcal{H}$ and computes incrementally the minimal transversals of $\{E_1, \dots, E_i\}$ from the minimal transversals of $\{E_1, \dots, E_{i-1}\}$. The algorithm is not output polynomial when there is a possibility that the intermediate steps have huge output sizes compared to the output solution. Indeed, it is proved in [21] that there exist hypergraphs for which Berge's algorithm is not output polynomial for any ordering. But, even though the applicability of Berge's algorithm seems to be limited, for several cases, *e.g.*, bounded tree-width graphs, planar graphs and more generally k-degenerate graphs, one can prove that Berge's algorithm is output polynomial. On the other hand, compared to the other algorithms (for instance Khachiyan's algorithm), Berge's algorithm has a great potential to produce polynomial delay polynomial space algorithm since the algorithm admits a depth-first search on the solution space so that it does not require to store the solutions already found in the memory. In this sense, expanding the applicability of Berge's algorithm is quite important for more understanding the TRANS-ENUM problem.

In this paper, we propose a new way of expanding the applicability of Berge's algorithm. One of the disadvantage of Berge's algorithm is a huge computation time on the intermediate steps. Our first idea is to identify intermediate steps which produce intermediate solutions that can be extended to an output solution, and "skip" the other costly intermediate steps. But the search cost for finding

[4] The *closed neighborhood* of a vertex v in a graph is the set containing v and all its neighbors.

neighboring solutions becomes hard; we have to spend much time and use much space for finding each solution in the next intermediate step, from the solution of the current step. Our idea is to introduce another enumeration scheme to cope with this difficulty, to enable polynomial delay enumeration of the solutions to the next intermediate step. We apply this idea to the minimal edge dominating set enumeration problem, and obtain the first polynomial delay polynomial space algorithm for this problem.

2 Preliminaries

If A and B are two sets, $A \backslash B$ denotes the set $\{x \in A \mid x \notin B\}$. The power-set of a set V is denoted by 2^V. The size of a set A is denoted by $|A|$. A graph $G = (V, E)$ is a pair of vertex set V and edge set $E \subseteq V \times V$. We only deal with finite and simple graphs. An edge between x and y in a graph is denoted by xy (equivalently yx) and sometimes it will be convenient to see an edge xy as the set $\{x, y\}$, but this will be clear from the context. A hypergraph is a pair $(V, \mathcal{F} \subseteq 2^V)$ with V called its vertex set and $\mathcal{F}$ its set of hyper-edges.

Let $G := (V, E)$ be a graph. For a vertex x, we denote by $\widetilde{N}(x)$ the set of edges incident to x, and $N(x)$ denotes the set of vertices adjacent to x. For every edge $e := xy$, we denote by $N[e]$ the set of edges adjacent to e, *i.e.* $N[e] := \widetilde{N}(x) \cup \widetilde{N}(y)$. A subset D of E is called an *edge dominating set* if for every edge e of G, we have $N[e] \cap D \neq \emptyset$, and D is *minimal* if no proper subset of it is an edge dominating set.

An *enumeration* algorithm for a search problem consists in listing completely all the solutions without duplications. When an enumeration algorithm always terminates in time polynomial in n and N where n is the input size and N is the output size, the algorithm is called *output polynomial*, and it is called *polynomial space* if it uses space bounded by a polynomial in n. The *delay* of an enumeration algorithm is the maximum computation time from the time that a solution is output to the time until the next solution, or the termination of the algorithm. The algorithm is called *polynomial delay* if its preprocessing time and the delay are both polynomial to the input size. It is worth noticing that such an algorithm has a running time bounded by the sum of the preprocessing time, and the delay multiplied by the number of solutions.

Let $\mathcal{H} := (V, \mathcal{F})$ be a hypergraph. The set of *private neighbors of a vertex* x w.r.t. $T \subseteq V$, denoted by $P_{\mathcal{H}}(x, T)$, is $\{E \in \mathcal{F} \mid E \cap T = \{x\}\}$. A subset T of V is called an *irredundant set* if $P_{\mathcal{H}}(x, T) \neq \emptyset$ for all $x \in T$. A *transversal* (or *hitting set*) of $\mathcal{H}$ is a subset of V that has a non-empty intersection with every hyper-edge of $\mathcal{H}$; it is *minimal* if it does not contain any other transversal as a proper subset. It is known that a transversal is minimal if and only if $P_{\mathcal{H}}(x, T) \neq \emptyset$ for all $x \in T$. The set of all minimal transversals of $\mathcal{H}$ is denoted by $tr(\mathcal{H})$.

For a graph $G := (V, E)$ and $E' \subseteq E$, we denote by $\mathcal{H}(E')$ the hypergraph $(E, \{N[e] \mid e \in E'\})$, and the *edge neighborhood hypergraph* of G is the hypergraph $\mathcal{H}(E)$. The following proposition is easy to obtain.

Proposition 1. *For any graph $G := (V, E)$, $T \subseteq E$ is an edge dominating set of G if and only if T is a transversal of $\mathcal{H}(E)$. Therefore, $T \subseteq E(G)$ is a minimal edge dominating set of G if and only if T is a minimal transversal of $\mathcal{H}(E)$.*

For a better readability we say that an edge f is a *private neighbor of an edge* e w.r.t. T in $\mathcal{H}(E')$, for $E' \subseteq E$, if $N[f] \in P_{\mathcal{H}(E')}(e, T)$, and by abuse of notation we will write $f \in P_{\mathcal{H}(E')}(e, T)$ instead of $N[f] \in P_{\mathcal{H}(E')}(e, T)$.

3 Berge's Algorithm and Basic Strategy

Our strategy for the enumeration is based on Berge's algorithm [2]. For a given hypergraph $\mathcal{H} := (V, \mathcal{F})$ with hyper-edges enumerated as $F_1, \ldots, F_m$, let $\mathcal{F}_j$ be $\{F_1, \ldots, F_j\}$ for each $1 \leqslant j \leqslant m$. Roughly, Berge's algorithm computes, for each $1 < j \leqslant m$, $tr(\mathcal{F}_j)$ from $tr(\mathcal{F}_{j-1})$. Although the algorithm is not polynomial space, there is a way to reduce the space complexity to polynomial. The algorithm follows a tree of height m rooted at $\varnothing$ and such that the nodes located on the i-th level correspond to the minimal transversals of $\mathcal{F}_i$. Thus, the leaves at level m correspond to the minimal transversals of the hypergraph. The tree can be described by the following parent-child relation. For $j \geqslant 1$ and $T \in tr(\mathcal{F}_j)$, we define the parent $Q'(T, j)$ of T as follows

$$Q'(T, j) := \begin{cases} T & \text{if } T \in tr(\mathcal{F}_{j-1}), \\ T \backslash \{v\} & \text{if } v \text{ is such that } P_{\mathcal{F}_j}(v, T) = \{F_j\}. \end{cases}$$

We can observe that $T \notin tr(\mathcal{F}_{j-1})$ if and only if $P_{\mathcal{F}_j}(v, T) = \{F_j\}$ holds for some $v \in T$, thus the parent is well defined and always in $tr(\mathcal{F}_{j-1})$ [14,18]. One can moreover compute the parent of any $T \in tr(\mathcal{F}_j)$ in time polynomial in $|V| + \sum_{F \in \mathcal{F}} |F|$. The tree induced by the parent-child relation spans all members of $\bigcup\limits_{1 \leqslant j \leqslant m} tr(\mathcal{F}_j)$. We can traverse this tree in a depth-first search manner from the root by recursively generating the children of the current visiting minimal transversal. Any child is obtained by adding at most one vertex, then the children can be listed in polynomial time. In this way, we can enumerate all the minimal transversals of a hypergraph with polynomial space.

Formally and generally, we consider the problem of enumerating all elements of a set $\mathcal{Z}$ that is a subset of an implicitly given set $\mathcal{X}$. Assume that we have a polynomial time computable parent function $P : \mathcal{X} \to \mathcal{X} \cup \{nil\}$. For each $X \in \mathcal{X}$, $P(X)$ is called the *parent* of X, and the elements Y such that $P(Y) = X$ are called *children* of X. The parent-child relation of P is *acyclic* if any $X \in \mathcal{X}$ is not a proper ancestor of itself, that is, it always holds that $X \neq P(P(\cdots P(X)) \cdots)$. We say that an acyclic parent-child relation is *irredundant* when any $X \in \mathcal{X}$ has a descendant in $\mathcal{Z}$, in the parent-child relation. The *depth* of an acyclic parent-child relation P is the size of the longest chain between nil and an element of $\mathcal{X}$. The following statements are well-known in the literature [1,20,17,14,18].

Proposition 2. *All elements in $\mathcal{Z}$ can be enumerated with polynomial space if there is a polynomial depth acyclic parent-child relation $P : \mathcal{X} \to \mathcal{X} \cup \{nil\}$ such that there is a polynomial space algorithm for enumerating all the children of each $X \in \mathcal{X} \cup \{nil\}$.*

Proposition 3. *All elements in $\mathcal{Z}$ can be enumerated with polynomial delay and polynomial space if there is a polynomial depth irredundant parent-child relation $P : \mathcal{X} \to \mathcal{X} \cup \{nil\}$ such that there is a polynomial delay polynomial space algorithm for enumerating all the children of each $X \in \mathcal{X} \cup \{nil\}$.*

With acyclic (resp., irredundant) parent-child relation $P : \mathcal{X} \to \mathcal{X} \cup \{nil\}$, the following algorithm enumerates all elements in $\mathcal{Z}$, with polynomial space (resp., with polynomial delay and polynomial space).

```
Algorithm ReverseSearch($X$)
 1. if $X \in \mathcal{Z}$ then output $X$
 2. for each candidate child $Y$, if $X = P(Y)$ then call ReverseSearch($Y$)
```

The call ReverseSearch(nil) enumerates all elements in $\mathcal{Z}$. Since the above parent-child relation for transversals Q' is acyclic, the algorithms proposed in [14,18] use polynomial space. However, the parent-child relation Q' is not irredundant and hence ReverseSearch(nil) does not guarantee a polynomial delay neither an output polynomiality. Indeed, we can expect that the size of $tr(\mathcal{F}_j)$ increases as the increase of j, and it can be observed in practice. However, $tr(\mathcal{F}_j)$ can be exponentially larger than $tr(\mathcal{F}_m)$, thus Berge's algorithm is not output polynomial [21]. Examples of irredundant parent-child relations can be found in the literature [1,20,17].

One idea to avoid the lack of irredundancy is to certify the existence of minimal transversals in the descendants. Suppose that we choose some levels $1 = l_1, \dots, l_k = m$ of Berge's algorithm, and state that for any $T \in tr(\mathcal{F}_{l_j})$, we have at least one descendant in $tr(\mathcal{F}_{l_{j+1}})$. This implies that any transversal in $tr(\mathcal{F}_{l_j})$ has a descendant in $tr(\mathcal{F}_m)$, thus we can have an irredundant parent-child relation by looking only at these levels, and the enumeration can be polynomial delay and polynomial space.

We will use this idea to obtain a polynomial delay polynomial space algorithm to enumerate the minimal edge dominating sets, the levels are determined with respect to a *maximal matching*. From now we assume that we have a fixed graph $G := (V, E)$ and we will show how to enumerate all its minimal edge dominating sets. A subset of E is a *matching* if every two of its edges e and f are not adjacent. A matching is *maximal* if it is not included in any other matching. Let $\{b_1, \dots, b_k\}$ be a maximal matching of G, and let $b_i = x_i y_i$. For each $0 \leqslant i \leqslant k$, let $V_i := V \backslash \left(\bigcup_{i' > i} b_{i'} \right)$, and let $E_i := \{e \mid e \subseteq V_i\})$. Let $B_i := E_i \backslash E_{i-1}$ for $i > 1$. Note that any edge in E_1 is adjacent to b_1 and by definition B_i never includes an edge $b_j \neq b_i$. Without loss of generality, we here assume that we have taken a linear ordering $\leqslant$ on the edges of G so that: (1) for each $e \in B_i$ and each $f \in E_{i-1}$ we have $f < e$, (2) for each $e \in \widetilde{N}(x_i) \cap B_i$, each $f \in \widetilde{N}(y_i) \cap B_i$ we have $b_i < e < f$. Observe that with that ordering we have $e < f$ whenever $e \in B_i$ and $f \in B_j$ with $i < j$. We consider that Berge's algorithm on $\mathcal{H}(E)$ follows that ordering. In fact we will prove using Berge's algorithm that we can define an irredundant parent-child relation to enumerate $tr(\mathcal{H}(E_i))$ from $tr(\mathcal{H}(E_{i-1}))$.

Lemma 1. *Let $1 \leqslant i < k$. Any $T \in tr(\mathcal{H}(E_{i-1}))$ has at least one descendant in $tr(\mathcal{H}(E_i))$.*

Proof. If $T' \in tr(\mathcal{H}(E_i))$ satisfies $T' = T$, then T' is a descendant of T since the parent is never greater than the child. If $T \notin tr(\mathcal{H}(E_i))$, some edges X of B_i are not dominated by T, and consider $T' := T \cup \{b_i\}$. We observe that b_i is adjacent to all edges of B_i and the edges in X are private neighbors of b_i in T', thus T' is included in $tr(\mathcal{H}(E_i))$. Let us compute the ancestor of T' in $tr(\mathcal{H}(E_{i-1}))$ as follows: set $T" := T'$ and repeatedly compute the parent of $T"$ and set $T"$ to its parent, until reaching a minimal transversal in $tr(\mathcal{H}(E_{i-1}))$. In this process no vertex of T is removed since each vertex in T has a private neighbor in E_{i-1}. But, at some point b_i is removed from T' since it is the only one in T' which has a private neighbor in B_i. This means that T is an ancestor of T', and thus T always has a descendant in $tr(\mathcal{H}(E_i))$. □

For conciseness, we introduce a new parent-child relation for edge dominating set enumeration. For $T \in tr(\mathcal{H}(E_i))$, let $Q'_j(T, |E_i|)$ be the ancestor of T located on the j-th level of Berge's algorithm, i.e., $Q'_j(T, |E_i|) = Q'(Q'(\cdots(T, |E_i|), |E_i|-1), \cdots, j+1)$. Then, we define the *skip parent* $Q(T, i)$ of T by $Q'_{|E_{i-1}|}(T, |E_i|)$. T' is a *skip-child* of $T \in tr(\mathcal{H}(E_{i-1}))$ if and only if $T' \in tr(\mathcal{H}(E_i))$ and $Q(T', i) = T$. The set of skip-children of $T \in tr(\mathcal{H}(E_i))$ is denoted by $\mathcal{C}(T, i)$. From Propositions 2 and 3, and Lemma 1 we have the following proposition.

Proposition 4. *If we can list all skip-children of $T \in tr(\mathcal{H}(E_i))$, for each $1 \leqslant i \leqslant k$, with polynomial delay and polynomial space, then we can enumerate all minimal edge dominating sets with polynomial delay and polynomial space.*

But, as we will show in the next section, for a transversal T in $tr(\mathcal{H}(E_{i-1}))$, the problem of finding a transversal of $tr(\mathcal{H}(E_i))$ including T is NP-complete in general. In order to overcome this difficulty, we will identify a pattern, that we call an *H-pattern*, that makes the problem difficult. We will first show that one can enumerate with polynomial delay and polynomial space all the skip-children that include no edges from H-patterns, and then define a new parent-child relation that will allow to enumerate also with polynomial delay and polynomial space the other skip-children in a different way. In the following sections, we explain the methods for the enumeration.

4 Computing Skip-Children

Let T be in $tr(\mathcal{H}(E_{i-1}))$ and $T' \in tr(\mathcal{H}(E_i))$ a skip-child of T. First notice that every edge in $T' \backslash T$ can have a private neighbor only in B_i. Indeed every edge in

E_{i-1} is already dominated by T and an edge in $T'\backslash T$ is only used to dominate an edge in B_i. Moreover, an edge $e \neq b_i$ in $\widetilde{N}(x_i) \cap (T'\backslash T)$ (resp. in $\widetilde{N}(y_i) \cap (T'\backslash T)$) can only have private neighbors in $\widetilde{N}(x_i) \cap B_i$ (resp. $\widetilde{N}(y_i) \cap B_i$). And from the proof of Lemma 1 if $b_i \in T'\backslash T$ then $T'\backslash T = \{b_i\}$.

Let us first consider the case that every edge in $T'\backslash T$ is adjacent to b_i. From our discussion above, when two edges in $T'\backslash T$ are incident to x_i (resp. y_i), they cannot have both private neighbors. Thus $T'\backslash T$ can include at most two such edges. Therefore, by choosing all combinations of one or two edges adjacent to b_i, adding them to T and then checking if the skip-parent of the resulting set is T, we can enumerate all the skip-children T' of T such that $T'\backslash T \subseteq B_i$ with polynomial delay and polynomial space.

We now consider the remaining case that an edge in $T'\backslash T$ is not adjacent to b_i. We call such a skip-child *extra*. We can see that at least one edge $f \neq b_i$ adjacent to b_i has to be included in T' to dominate b_i. Actually, since $b_i < e$ for any $e \in B_i\backslash\{b_i\}$, any extra skip-child of T is a descendant of some $T \cup \{f\}$ with $f \neq b_i$ incident to x_i or y_i in the original parent-child relation. So, without loss of generality, we will assume that such an edge $f \neq b_i$ is incident to x_i and is included in T. Hereafter, we suppose that $N(y_i) := \{z_1, \ldots, z_k\}$ and assume T' is an extra skip-child of T.

A vertex $z_h \in N(y_i) \cap V_i$ is *free* if it is not incident to an edge in T, and is *non-free* otherwise. A free vertex is said to be *isolated* if it is not incident to an edge in E_{i-1}. Clearly, if there is an isolated free vertex, then T has no extra skip-child. Thus, we assume that there is no isolated free vertex. Edges in $E_i\backslash B_i$ that are incident to some free vertices are called *border edges*. Observe that any border edge vz_h incident to a free vertex z_h is adjacent to an edge $vw \in T$ if $v \in V_{i-1}$. The set of border edges is denoted by $Bd(T, i)$. Note that no edge in $Bd(T, i)$ is incident to two free vertices, otherwise the edge is in E_{i-1} but not dominated by T, and then any border edge is incident to exactly one free vertex. We can see that an edge of B_i incident to y_i is not dominated by T if and only if it is incident to a free vertex, and any edge in $T'\backslash T$ that is not incident to x_i is a border edge. Then, for any border edge set $Z \subseteq Bd(T, i)$, $T \cup Z \in tr(\mathcal{H}(E_i))$ only if each free vertex has a border edge $e \in Z$ incident to it. Since any border edge is incident to exactly one free vertex, for any $Z \subseteq Bd(T, i)$ such that $T \cup Z$ is irredundant and for any edge $vz_h \in Z$ with free vertex z_h, $P_{\mathcal{H}(E_i)}(e, T \cup Z)$ is always $\{vz_h\}$. This implies that $T \cup Z$ is in $tr(\mathcal{H}(E_i))$ only if $Z \subseteq Bd(T, i)$ includes exactly one edge incident to each free vertex. We call such an edge set Z a *selection*. We observe that all border edges are dominated by Z. We have the following lemma which is straightforward to prove.

Lemma 2. *For any edge subset Z with $Z \cap T = \emptyset$, there holds $T \cup Z \in tr(\mathcal{H}(E_i))$ only if Z is a selection.*

An edge $e \in T$ is called *redundant* if all edges in $P_{\mathcal{H}(E_{i-1})}(e, T)$ are border edges and no edge y_iz_h is in $P_{\mathcal{H}(E_i)}(e, T)$.

Lemma 3. *If T has a redundant edge, then any selection Z does not satisfy $T \cup Z \in tr(\mathcal{H}(E_i))$.*

Proof. Let e be a redundant edge of T. Since any border edge f is incident to a free vertex z_h, any selection Z should contain one edge incident to z_h and then if f is incident to e, we have $f \notin P_{\mathcal{H}(E_i)}(e, T \cup Z)$. Since no edge $y_i z_h$ is in $P_{\mathcal{H}(E_i)}(e, T)$, there holds that $P_{\mathcal{H}(E_i)}(e, T \cup Z) = \emptyset$ for any selection Z. □

Let $X_T := \{e \in Bd(T, i) \mid \exists e' \in T \text{ and } P_{\mathcal{H}(E_i)}(e', T \cup \{e\}) \subseteq Bd(T, i)\}$. The addition of any edge $e \in X_T$ to T transforms an edge e' of T into a redundant one with respect to $T \cup \{e\}$, and thus by Lemma 3 for any $Z \subseteq Bd(T, i)$, $T \cup Z \in tr(\mathcal{H}(E_i))$ holds only if $Z \cap X_T = \emptyset$. Therefore, the following follows.

Lemma 4. *If a free vertex is not incident to an edge in $Bd(T, i) \backslash X_T$, then any $Z \subseteq Bd(T, i)$ does not satisfy $T \cup Z \in tr(\mathcal{H}(E_i))$.*

One can hope that we can characterize the selections Z not intersecting X_T such that $T \cup Z \in tr(\mathcal{H}(E_i))$ and be able to use it for listing the extra skip-children. Unfortunately, checking whether there is such a selection Z is NP -complete.

Theorem 1. *Given $T \in tr(\mathcal{H}(E_{i-1}))$, it is* NP*-complete to check whether there is a selection Z such that $Z \cap X_T = \emptyset$ and $T \cup Z \in tr(\mathcal{H}(E_i))$.*

In order to overcome this difficulty, we identify a pattern, that we call an *H-pattern*, that makes the problem difficult.

Definition 1 (H-Pattern). *A vertex set $\{z_\ell, v_\ell, z_j, v_j\}$ is an H-*pattern *if z_ℓ and z_j are free vertices, $v_\ell v_j$ is in T, and $v_\ell v_j$ has two non-border private neighbors in $E_{i-1} \backslash T$: one is adjacent to v_ℓ and the other to v_j. We also say that the edges $z_\ell v_\ell$, $z_j v_j$ and $v_\ell v_j$ induces an H-pattern.*

We will see that the NP-completeness comes from the presence of H-patterns. Indeed, for an H-pattern $\{z_\ell, v_\ell, z_j, v_j\}$, any private neighbor of $v_\ell v_j$ is adjacent to either $z_\ell v_\ell$ or to $z_j v_j$, thus we cannot add both to a selection Z since in that case $P_{\mathcal{H}(E_i)}(v_\ell v_j, T \cup Z)$ will be empty. Let H_T be the set of border edges included in an H-pattern. In the next two subsections we will see how to list selections including no edge from H_T, and those that do.

Lemma 5. *If T has no redundant edge, then $T \cup Z \in tr(\mathcal{H}(E_i))$ holds for any selection $Z \subseteq Bd(T, i) \backslash (X_T \cup H_T)$.*

Proof. From the definition, $T \cup Z$ dominates all the edges in E_i and for each $e \in Z$ it holds that $P_{\mathcal{H}(E_i)}(e, T \cup Z) \neq \emptyset$. Since Z includes no edge from $H_T \cup X_T$, and T has no redundant edge, one easily checks from Lemmas 2, 3 and 4 by case analysis that any edge $e \in T$ has a private neighbor f that is adjacent to no border edge, or an edge $y_i z_h$ is adjacent to e and not to edges in $T \backslash \{e\}$. Thus, either $f \in P_{\mathcal{H}(E_i)}(e, T \cup Z)$ or $y_i z_h \in P_{\mathcal{H}(E_i)}(e, T \cup Z)$. These imply that $T \cup Z$ is in $tr(\mathcal{H}(E_i))$. □

4.1 Dealing with Redundancies

The lemmas above demonstrate how to construct transversals $T' \in tr(\mathcal{H}(E_i))$ from T, but some generated transversals may not be extra skip-children of T. This is because such T' can be also generated from other transversals in $tr(\mathcal{H}(E_{i-1}))$. Such redundancies happen for example when two edges f_1 and f_2 in T' have private neighbors only in B_i, but after the removal of either one from T', the other will have a private neighbor outside B_i. Assuming in this case that $f_1 \in T$ and $f_2 \in T' \backslash T$, it holds that T' can be generated from T or from $(T \backslash \{f_1\}) \cup \{f_2\}$. And since the number of selections Z such that $T \cup Z \in tr(\mathcal{H}(E_i))$ can be arbitrarily large, we need to avoid such redundancies.

To address this issue, we state the following lemmas to characterize the edges not to be added to selections Z such that $T \cup Z$ is an extra skip-child of T. We say that a border edge vz_ℓ is *preceding* if there is an edge vz_h in T satisfying $P_{\mathcal{H}(E_{i-1})}(vz_h, T) \subseteq N[vz_\ell]$ and $y_i z_\ell < y_i z_h$, and denote the set of preceding edges by X'_T. We also say that an edge $vz_h \in T$ is *fail* if $P_{\mathcal{H}(E_{i-1})}(vz_h, T) \subseteq Bd(T, i)$, $y_i z_h$ is in $P_{\mathcal{H}(E_i)}(vz_h, T)$, and no edge $wz_\ell \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$ satisfies $y_i z_h < y_i z_\ell$.

Lemma 6. *For any selection Z including a preceding edge, $T \cup Z$ is not an extra skip-child of T.*

Lemma 7. *If T has a fail edge, then $T \cup Z$ is not an extra skip-child of T for any selection Z.*

We are now able to characterize exactly those selections Z not intersecting H_T and such that $T \cup Z$ is an extra skip-child of T.

Lemma 8. *Suppose that T has neither redundant edges nor fail edges and any free vertex is incident to an edge in $Bd(T, i)$. Then, $T \cup Z$ with $T \cap Z = \emptyset$ is an extra skip-child of T including no edge of H_T if and only if Z is a selection including no edge of $X_T \cup X'_T \cup H_T$.*

As a corollary we have the following.

Proposition 1 *One can enumerate with polynomial delay and space all the extra skip-children of T that do not contain edges of H_T.*

Proof. If T has redundant edges or fail edges or has a free vertex not incident to an edge in $Bd(T, i) \backslash X_T$, then by Lemmas 3, 4 and 7 we can conclude that T has no extra skip-child. Since we can compute X_T in polynomial time and check in polynomial time whether an edge is redundant or is a fail edge, this step can be done in polynomial time. So, assume T has no redundant edges, no fail edges and every free vertex is incident to an edge in $Bd(T, i) \backslash X_T$. By Lemma 8 by removing all edges in $H_T \cup X_T \cup X'_T$, any selection Z is such that $T \cup Z$ is a skip-child of T. One easily checks that the enumeration of these selections can be performed by picking exactly one edge in each incident star. □

4.2 Dealing with the Presence of H-Patterns

As we saw in Theorem 1, it is hard to enumerate all extra skip-children having some edges in H-patterns from a given transversal $T \in tr(\mathcal{H}(E_{i-1}))$. Let us call these children *slide-children.* We approach this difficulty by introducing a new parent-child relation among slide-children, and enumerate them by traversing the forest induced by the new relation. In this way, we now do not follow the skip-parent skip-child relation for slide-children. However, the root of each tree in the induced forest is a transversal obtained with the skip-child skip-parent relation. Let us be more precise now. For two sets S and S' of edges we write $S <_{lex} S'$ if $\min(S \Delta S') \in S$, called *lexicographical ordering.*

Hereafter, we consider an extra skip-child $T' = T \cup Z$ of $T \in tr(\mathcal{H}(E_{i-1}))$ such that $T' \cap H_T \neq \emptyset$. Let $H^*(T') := \{v_h z_h, v_\ell z_\ell, v_h v_\ell\}$ be the lexicographically minimum H-pattern among all H-patterns of T that includes an edge of Z. Without loss of generality, we assume that $v_\ell z_\ell$ is in Z. Let uz_h be the edge in Z incident to z_h. Notice that such an edge exists because z_h is a free vertex. Then, we define the *slide-parent* $Q^*(T', i)$ of T' by $T' \cup \{v_h z_h\} \backslash \{uz_h, v_h v_\ell\}$.

Lemma 9. *The slide-parent of T' is well-defined and is a member of $tr(\mathcal{H}(E_i))$.*

Proof. Since z_h is a free vertex for T, Z includes exactly one edge incident to z_h, thus uz_h is uniquely determined, and thus the slide-parent is uniquely defined. Since uz_h is a border edge, either $u \notin V_{i-1}$ or u is incident to an edge of T. This together with that $v_h z_h$ and $v_\ell z_\ell$ dominate all edges in $N[v_h v_\ell]$ leads that $Q^*(T', i)$ dominates all edges in E_i.

By adding $v_h z_h$ to T', no edge in $T' \backslash \{uz_h, v_h v_\ell\}$ loses its private neighbor. The edge $v_h z_h$ is adjacent to no edge in $T' \backslash \{uz_h, v_h v_\ell\}$, and $v_h z_h \in P_{\mathcal{H}(E_i)}(v_h z_h, Q^*(T', i))$. These imply that $Q^*(T', i)$ is a member of $tr(\mathcal{H}(E_i))$. □

The slide-parent of T has less edges than T, thus the (slide-parent)-(slide-child) relationship is acyclic, and for each $T' \in tr(\mathcal{H}(E_i))$, there is an ancestor $T'' \in tr(\mathcal{H}(E_i))$ in the (slide-parent)-(slide-child) relation such that the skip-parent of T'' has no H-pattern. Similar to the depth-first search versions of Berge's algorithm [14,18], we will traverse the (slide-parent)-(slide-child) relation to enumerate all transversals including H-pattern edges. The following follows from the definition of slide-parent.

Proposition 5. *Any slide-child T' of T'' is obtained from T'' by adding two edges and remove one edge.*

The computation of the slide-parent of any $T' \in tr(\mathcal{H}(E_i))$ including edges of H-patterns can be easily done in polynomial time: compute its skip-parent T in polynomial time, choose $H^*(T)$ and then compute its slide-parent in polynomial time as described above. Proposition 5 shows that there are at most n^3 candidates for slide-children, thus the enumeration of slide-children can be done with polynomial delay and polynomial space.

Lemma 10. *For any $T' \in tr(\mathcal{H}(E_i))$, all its slide-children can be enumerated with polynomial delay and polynomial space.*

We can now summarize the steps of the algorithm.

1. All transversals in $tr(\mathcal{H}(E_1))$ can be enumerated with polynomial delay and polynomial space, since they include at most two edges from $N[b_1]$.
2. In Section 4 (second paragraph), we have explained how to enumerate all non-extra skip-children with polynomial delay and polynomial space.
3. By Proposition 1 all the extra skip-children not including any edges of H-patterns can be enumerated with polynomial delay and polynomial space.
4. By Lemma 10 all the extra skip-children including some edges from H-patterns can be enumerated with polynomial delay and space.
5. Therefore, by executing these three enumeration algorithms for each minimal transversal $T \in tr(\mathcal{H}(E_{i-1}))$, we can generate all the members in $tr(\mathcal{H}(E_i))$ with polynomial delay and polynomial space.

All these show that the conditions of Proposition 4 are satisfied. And thus we can state our main result.

Theorem 2. *All edge minimal dominating sets in a graph G can be enumerated with polynomial delay and polynomial space.*

The greatest delay is reached by the computation of the slide-children of a given $T \in tr(\mathcal{H}(E_i))$. We have $O(n^3)$ candidates and for each one we compute its slide-parent in time $O(m)$. Then the slide-children can be enumerated with delay $O(mn^3)$. Since the depth of the (skip-parent)-(skip-child) relation is the size of the maximal matching, it is bounded by n, and then the total delay is bounded by $O(n^6)$.

5 Conclusion

In this paper, we propose a polynomial delay polynomial space algorithm for listing all minimal edge dominating sets in a given graph. This improves drastically the previously known algorithms which were incremental output polynomial and use exponential space. We state furthermore that usual approaches with Berge's algorithm involves an NP-complete problem, and thus it is difficult with usual approaches of Berge's algorithm to produce an efficient algorithm. To cope with this difficulty, we introduce a new idea of "changing the traversal routes in the area of difficult solutions" (the notion of skip-children and the removal of edges involved in H-patterns). Based on this idea, we give a new traversal route on these difficult solutions, that is totally independent from Berge's traversal route (the (slide-parent)-(slide-child) relation). As a result, we are able to construct a polynomial delay polynomial space algorithm.

The idea of changing the traversal routes seems to be new and to be able to apply to many other kind of algorithms in enumeration area. Interesting future works are applications of this idea to other kind of enumeration algorithms, *e.g.* the one used by Lawler et al. for enumerating maximal subsets [16] or other algorithms for enumerating minimal transversals (see for instance [6]).

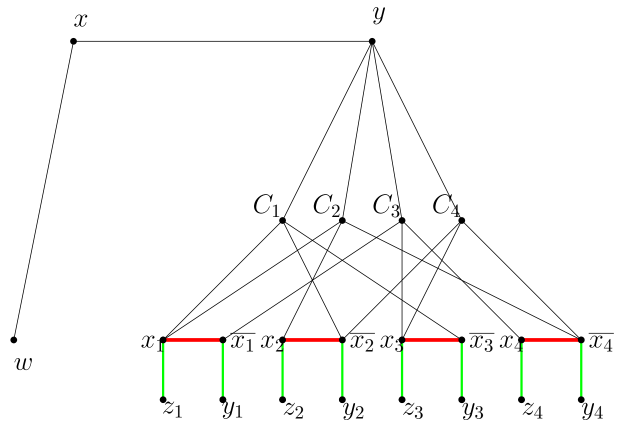
References

1. D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.
2. C. Berge. *Hypergraphs: Combinatorics of Finite Sets.* North-Holland, 1989.
3. A. Bondy and U.S.R. Murty. *Graph Theory.* Graduate Texts in Mathematics. Springer, 2008.
4. E. Boros, K. M. Elbassioni, and V. Gurvich. Transversal hypergraphs to perfect matchings in bipartite graphs: Characterization and generation algorithms. *Journal of Graph Theory*, 53(3):209–232, 2006.
5. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.
6. T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM J. Comput.*, 32(2):514–537, 2003.
7. F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger. Enumerating minimal subset feedback vertex sets. *Algorithmica*, 69(1):216–231, 2014.
8. P. A. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. An incremental polynomial time algorithm to enumerate all minimal edge dominating sets. In *ICALP (1)*, volume 7965 of *LNCS*, pages 485–496. Springer, 2013.
9. D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *PODS*, pages 209–216, 1997.
10. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*, volume 208 of *Pure and Applied Mathematics.* M. Dekker, 1998.
11. M. M. Kanté, V. Limouzy, A. Mary, and L.Nourine. On the neighbourhood helly of some graph classes and applications to the enumeration of minimal dominating sets. In *ISAAC*, pages 289–298, 2012.
12. M. M. Kanté, V. Limouzy, A. Mary, and L. Nourine. On the enumeration of minimal dominating sets and related notions. *SIAM J. Discrete Math.*, 28(4):1916–1929, 2014.
13. M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, and T. Uno. On the enumeration and counting of minimal dominating sets in interval and permutation graphs. In *ISAAC*, volume 8283 of *LNCS*, pages 339–349. Springer, 2013.
14. D. J. Kavvadias and E. C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *J. Graph Algorithms Appl.*, 9(2):239–264, 2005.
15. L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, V. Gurvich, and K. Makino. Generating cut conjunctions in graphs and related problems. *Algorithmica*, 51(3):239–263, 2008.
16. E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. *SIAM J. Comput.*, 9(3):558–565, 1980.
17. K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *SWAT*, volume 3111 of *LNCS*, pages 260–272. Springer, 2004.
18. K. Murakami and T.Uno. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Applied Mathematics*, 170(0):83 – 94, 2014.
19. B. Schwikowski and E. Speckenmeyer. On enumerating all minimal solutions of feedback problems. *Discrete Applied Mathematics*, 117(1-3):253–265, 2002.
20. A. Shioura, A. Tamura, and T. Uno. An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM J. Comput.*, 26(3):678–692, 1997.
21. K. Takata. A worst-case analysis of the sequential method to list the minimal hitting sets of a hypergraph. *SIAM J. Discrete Math.*, 21(4):936–946, 2007.

A Omitted proofs

Proof (Proof of Theorem 1). We reduce the 3SAT problem to IMT. Let $\mathcal{C} = C_1, C_2, \cdots, C_m$ be the sets of clauses of a 3SAT instance over variables $x_1, ..., x_n$. We create an instance of IMT as follows:

- $V_{i-1} := \{x_j \mid j \leqslant n\} \sqcup \{\overline{x_j} \mid j \leqslant n\} \sqcup \{C_h \mid h \leqslant m\} \sqcup \{z_j \mid j \leqslant n\} \sqcup \{y_j \mid j \leqslant n\} \sqcup \{w\}$
- $E_{i-1} := \{x_j C_h \mid x_j \in C_h\} \sqcup \{\overline{x_j} C_h \mid \overline{x_j} \in C_h\} \sqcup \{x_j z_j\} \mid j \leqslant n\} \sqcup \{\overline{x_j} y_j \mid j \leqslant n\} \sqcup \{x_j \overline{x_j} \mid j \leqslant n\}$
- $V_i := V_{i-1} \sqcup \{x, y\}$
- $E_i := E_{i-1} \sqcup \{y C_h \mid h \leqslant m\} \sqcup \{xy, xw\}$
- $T := \{x_j \overline{x_j} \mid j \leqslant n\} \sqcup \{xw\}$



Notice that $X_T = \varnothing$, and then any selection Z is such that $Z \cap X_T = \varnothing$. Now we claim that the 3SAT instance is satisfiable if and only if there is an edge set $Z \subset Bd(T, i)$ with $Z \cap T = \varnothing$ such that $T \cup Z$ is included in $tr(\mathcal{H}(E_i))$.

Note that here we have $Bd(T, i) = \{x_j C_h \mid x_j \in C_h\} \cup \{\overline{x_j} C_h \mid \overline{x_j} \in C_h\}$. Notice now that a subset Z of $Bd(T, i)$ is such that $T \cup Z \notin tr(\mathcal{H}(E_i))$ if and only if Z contains an edge of $\widetilde{N}(x_i)$ and an edge of $\widetilde{N}(\overline{x_i})$ for some $i \leqslant n$.

Let $f : \{x_1, \cdots, x_n\} \to \{0, 1\}$ be an assignment to the variables which satisfies the 3SAT formula. Then consider the following subset Z of $Bd(T, i)$,

$$Z := \left(\bigcup_{x_j \mid f(x_j)=1} \widetilde{N}(x_j) \cap Bd(T, i) \right) \cup \left(\bigcup_{x_j \mid f(x_j)=0} \widetilde{N}(\overline{x_j}) \cap Bd(T, i) \right).$$

Clearly, since f satisfies the formula, Z is a selection in $Bd(T, i)$ since otherwise a clause would not be satisfied by f. Notice now that by construction, either $\widetilde{N}(x_j) \cap Z = \varnothing$ or $\widetilde{N}(\overline{x_j}) \cap Z = \varnothing$ for every $j \leqslant n$, and then there exists $Z' \subseteq Z$ such that $T \cup Z'$ is in $tr(\mathcal{H}(E_i))$.

Assume now that there exists a selection Z such that $T \cup Z \in tr(\mathcal{H}(E_i))$. Let $f : \{x_1, \cdots, x_n\} \to \{0,1\}$ be such that:

$$f(x_j) := \begin{cases} 1 & \text{if } \widetilde{N}(x_j) \cap Z \neq \varnothing \\ 0 & \text{if } \widetilde{N}(\overline{x_j}) \cap Z \neq \varnothing. \end{cases}$$

Notice first that f is well-defined. Indeed, assume that for some x_j, we have $\widetilde{N}(x_j) \cap Z \neq \varnothing$ and $\widetilde{N}(\overline{x_j}) \cap Z \neq \varnothing$. Then the private neighbor of the edge $x_j\overline{x_j}$ with respect to $T \cup Z$ would be empty, contradicting the fact that $T \cup Z \in tr(\mathcal{H}(E_i))$. Now since Z is a selection of $Bd(T,i)$, for every $h \leqslant m$, $Z \cap \widetilde{N}(C_h) \neq \varnothing$ and then there exists either $x_j \in C_h$ with $f(x_j) = 1$ or $\overline{x_j} \in C_h$ with $f(x_j) = 0$. Thus f satisfies all clauses. □

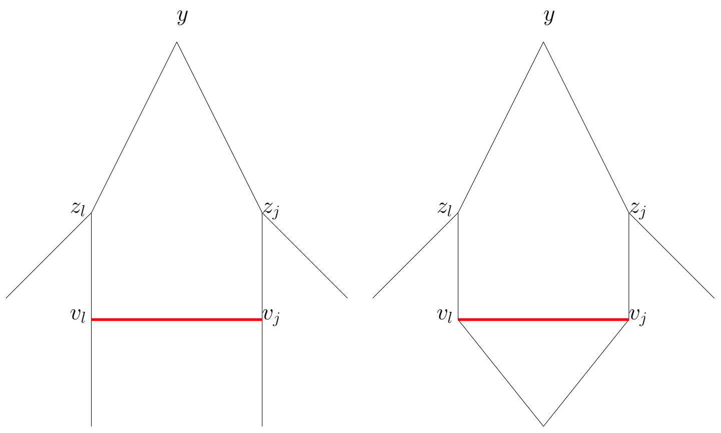


Fig. 1. Examples of H-pattern

Proof (Proof of Lemma 6). We can assume without loss of generality that $T \cup Z \in tr(\mathcal{H}(E_i))$, otherwise the statement holds. Suppose that there are several edges $v_1z_{h_1}, \cdots, v_1z_{h_p}$ in T that are adjacent to preceding edges included in Z, and among them let $v_jz_{h_j}$ be such that $y_iz_{h_j}$ is greater than any $y_iz_{h_\ell}$ for $1 \leqslant \ell \leqslant p$ and $\ell \neq j$, and let $v_jz_{\ell_j}$ be the preceding edge included in Z such that $P_{\mathcal{H}(E_{i-1})}(vz_{h_j}, T) \subseteq N[v_jz_{\ell_j}]$. Let t be the index of $y_iz_{h_j}$ in our ordering of the edges of E and let $\mathcal{F}_t$ be the first t edges of E in our ordering (which includes of course $y_iz_{h_j}$). Since $P_{\mathcal{H}(E_{i-1})}(vz_{h_j}, T) \subseteq N[v_jz_{\ell_j}]$ no edge of T but $v_jz_{h_j}$ is adjacent to $y_iz_{h_j}$ otherwise $P_{\mathcal{H}(E_i)}(v_jz_{h_j}, T \cup Z)$ would be empty, and then $P_{\mathcal{H}(E_i)}(v_jz_{h_j}, T \cup Z) = \{y_iz_{h_j}\}$. From the choice of $v_jz_{h_j}$ it follows that for any edge $e \in T$ that is adjacent to a neighbor z_h of y_i and such that $y_i z_h > y_i z_{h_j}$ there exists an edge $f \in E_{i-1} \cap P_{\mathcal{H}(E_i)}(e, Q'_t(T \cup Z, |E_i|))$. Thus, every edge in T has a private neighbor in $\mathcal{F}_t$ and hence $Q'_t(T \cup Z, |E_i|)$ includes all edges of T. But since $y_i z_{h_j}$ has index t and $P_{\mathcal{F}_t}(v_j z_{h_j}, T \cup Z) = P_{\mathcal{H}(E_i)}(v_j z_{h_j}, T \cup Z) = \{y_i z_{h_j}\}$, we can conclude that $Q'_{t-1}(T \cup Z, |E_i|)$ does not contain $v_j z_{h_j}$, and then the skip-parent of $T \cup Z$ does not include $v_j z_{h_j}$. Therefore, $T \cup Z$ is not an extra skip-child of T. □

Proof (Proof of Lemma 7). Let vz_h be a fail edge of T and let Z be a selection. Suppose without loss of generality that $T \cup Z \in tr(\mathcal{H}(E_i))$. Since $P_{\mathcal{H}(E_{i-1})}(vz_h, T) \subseteq Bd(T, i)$ and each free vertex should be incident to an edge in Z we can conclude that $P_{\mathcal{H}(E_i)}(vz_h, T \cup Z) = \{y_i z_h\}$. Now let t be the index of $y_i z_h$ in the ordering of E and let $\mathcal{F}_t$ be the first t edges in this ordering. Assume also that $Q'_t(T \cup Z, |E_i|)$ contains all edges of T, otherwise $T \cup Z$ is not an extra skip-child of T. But, since $P_{\mathcal{F}_t}(v_j z_{h_j}, T \cup Z) = P_{\mathcal{H}(E_i)}(v_j z_{h_j}, T \cup Z) = \{y_i z_h\}$, we can conclude that $Q'_{t-1}(T \cup Z, |E_i|)$ does not contain $v_j z_h$, and then the skip-parent of $T \cup Z$ does not include $v_j z_{h_j}$. Thus, $T \cup Z$ is not an extra skip-child of T. □

Proof (Proof of Lemma 8). The only if part is clear from lemmas 2, 5, 6 and 7. Let us now prove the if part. Suppose that we have a selection Z including no edge in $X_T \cup X'_T \cup H_T$. By Lemmas 4 and 5 it holds that $T \cup Z \in tr(\mathcal{H}(E_i))$. Suppose now that $Q(T \cup Z, i) \neq T$. Then, let us consider the computation of $Q(T \cup Z, i)$ in E_i: we compute $Q'(T \cup Z, |E_i|)$, $Q'(Q'(T \cup Z, |E_i|), |E_i| - 1)$, and so on. Let F be the set of edges not in B_i incident to the neighbors of y_i in V_i, *i.e.* $F := \tilde{N}(N(y_i)) \backslash \tilde{N}(y_i)$. First notice that $T \backslash F = Q(T \cup Z, i) \backslash F$. So, T and $Q(T \cup Z, i)$ can differ only on edges in F. So, let vz_h be the first edge removed among $T \backslash Q(T \cup Z, i)$ in this operation sequence, *i.e.*, $Q'_{j+1}(T \cup Z, |E_i|)$ includes all edges in $T \backslash Q(T \cup Z, i)$, but $Q'_j(T \cup Z, |E_i|)$ does not include vz_h. Let $\mathcal{F}_j$ be the first j edges in our ordering. Notice that all edges in E_{i-1} are in $\mathcal{F}_j$. Then, we can see that $P_{\mathcal{H}(E_i)}(vz_h, T \cup Z) = \{y_i z_h\}$. This implies that any private neighbor in $P_{\mathcal{H}(E_{i-1})}(vz_h, T)$ is dominated by some edges in Z, and no edge in $(T \backslash \{vz_h\}) \cup Z$ is adjacent to z_h. We further see that if there is a private neighbor $uv \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$ that is not a border edge, then no border edge is incident to u. Indeed, u is not a free vertex and is necessarily in V_{i-1} and if there is a border edge uz_ℓ this edge should be in E_{i-1} and since it should be dominated by T and $vu \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$, there would exist an edge in T incident to z_ℓ contradicting that uz_ℓ is a border edge. Similarly if there is a non border $uz_h \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$, then no border edge is incident to u.

Suppose that all edges in $P_{\mathcal{H}(E_{i-1})}(vz_h, T)$ are border edges. Since vz_h is not a fail edge, there exists an edge $wz_\ell \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$ satisfying $y_i z_h < y_i z_\ell$. This implies that the edge $wz_\ell \in P_{\mathcal{F}_j}(vz_h, Q'_{j+1}(T \cup Z, |E_i|))$, and then vz_h should be in $Q'_j(T \cup Z, |E_i|)$, otherwise wz_ℓ would not be dominated by $Q'_j(T \cup Z, |E_i|)$, thus yielding a contradiction.

Suppose now that there is a non-border edge vu in $P_{\mathcal{H}(E_{i-1})}(vz_h, T)$. We note that u can be z_h so that $vu = vz_h$. Since vz_h is not in $Q'_j(T \cup Z, |E_i|)$, there should be a border edge in Z adjacent to vz_h. We can observe that

u is incident to no edge in $T \backslash \{vz_h\}$, thus any border edge adjacent to uv is incident to v. If $P_{\mathcal{H}(E_{i-1})}(vz_h, T) \subseteq \widetilde{N}(v)$, then since T has no preceding edge any border edge $vz_\ell \in Z$ satisfies that $y_i z_h < y_i z_\ell$. This implies that $Q'_{j+1}(T \cup Z, |E_i|)$ includes no such border edge, and uv is a private neighbor of vz_h in $P_{\mathcal{F}_j}(vz_h, Q'_{j+1}(T \cup Z, |E_i|))$. This implies that vz_h is included in $Q'_j(T \cup Z, |E_i|)$, yielding a contradiction. So, there is a non border edge $z_h w \in P_{\mathcal{H}(E_{i-1})}(vz_h, T)$. Let z_s and z_p be free vertices adjacent respectively to z_h and v and such that $z_h z_s$ and vz_p are in Z. If $z_s \neq z_p$, then $\{z_s, z_h, z_p, v\}$ would form an H. So there is at most one free vertex z_s such that vz_s and $z_h z_s$ are in Z. If such a z_s exists, then one of $z_h z_s$ and vz_s is not in Z. And then in this case either wz_h or vu is in $P_{\mathcal{F}_j}(vz_h, Q'_{j+1}(T \cup Z, |E_i|))$, contradicting that vz_h is not in $Q'_j(T \cup Z, |E_i|)$. If z_h is not adjacent to a border edge, then $wz_h \in P_{\mathcal{F}_j}(vz_h, Q'_{j+1}(T \cup Z, |E_i|))$, and then again vz_h would be in $Q'_j(T \cup Z, |E_i|)$.

From the discussion, we have that $T \backslash Q(T \cup Z, i) = \emptyset$. Since $Q(T \cup Z, i)$ and T are both minimal in $tr(\mathcal{H}(E_{i-1}))$, we have $Q(T \cup Z, i) = T$. □