Tree Containment Above Minimum Degree is FPT*

Fedor V. Fomin† fedor.fomin@uib.no
Petr A. Golovach† petr.golovach@uib.no
Danil Sagunov‡ danilka.pro@gmail.com
Kirill Simonov§ kirillsimonov@gmail.com

Abstract

According to the classic Chvátal's Lemma from 1977, a graph of minimum degree $\delta(G)$ contains every tree on $\delta(G)+1$ vertices. Our main result is the following algorithmic "extension" of Chvátal's Lemma: For any n-vertex graph G, integer k, and a tree T on at most $\delta(G)+k$ vertices, deciding whether G contains a subgraph isomorphic to T, can be done in time $f(k)\cdot n^{\mathcal{O}(1)}$ for some function f of k only.

The proof of our main result is based on an interplay between extremal graph theory and parameterized algorithms.

1 Introduction

In the Tree Containment problem we are given an n-vertex graph G and a tree T. The task is to identify whether G has a subgraph isomorphic to T.[1] For the very special case of T being an n-vertex path, solving Tree Containment is equivalent to deciding whether G contains a Hamiltonian path and thus is NP-complete. Our work on Tree Containment is strongly motivated by the recent advances in algorithmic "extensions" of the classic theorems of extremal combinatorics [16, 17, 26].

For example, the classic theorem of Dirac states that every 2-connected graph contains a cycle (and thus a path) of length at least $\min\{2\delta(G), n\}$, where $\delta(G)$ is the minimum degree of G. In [16], we gave an FPT algorithm for parameterization "above Dirac's bound"—an algorithm that for any $k \geq 1$, decides whether a connected G contains a path of length at least $2\delta(G)+k$ in time $f(k)\cdot n^{\mathcal{O}(1)}$ for some function f of k only.

The question of how to impose conditions on vertex degrees of the host graph G to guarantee that it contains a certain tree T as a subgraph is fundamental in extremal graph theory. However, compared with path and cycle containments, tree containment is much more challenging. For example, the theorem of Erdős and Gallai [11] from 1959 asserts that every graph of average degree $> d$ contains a cycle with at least $d+1$ vertices. Similarly, Erdős and Sós [9] conjectured in 1963 that every graph with average degree $> d$ contains any tree on $d+1$ vertices. This conjecture remains open.

The starting point of our algorithmic study of Tree Containment is the following cute result first published by Chvátal.

Lemma 1.1. (Chvátal's Lemma [6]) *If G is a graph of minimum degree $\delta(G)$, then G contains every tree on $\delta(G)+1$ vertices.*

From the combinatorial point of view, the result of Lemma 1.1 is tight: a δ-regular graph does not contain a star of degree $\delta(G)+1$. The proof of Lemma 1.1 is constructive and it yields a polynomial time algorithm

*The full version of the paper can be accessed at https://arxiv.org/abs/2310.09678

†Department of Informatics, University of Bergen, Norway.
‡St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia.
§Hasso Plattner Institute, University of Potsdam, Germany.

[1] Let us remark that in computational biology the name tree containment is used for a different problem of deciding whether a phylogenetic network displays a phylogenetic tree over the same set of labeled leaves.

computing a subtree in G isomorphic to a tree T on $\delta(G)+1$ vertices. Whether Lemma 1.1 is tight from the algorithmic point of view, that is, whether it is possible to decide in polynomial time if a tree on $\delta(G)+k$ vertices, for some fixed constant $k>1$, is in G, was open prior to our work. Our main result is the following "algorithmic extension" of Chvátal's Lemma.

THEOREM 1.1. *For any n-vertex graph G, integer k, and a tree T on at most $\delta(G)+k$ vertices, there is a randomized algorithm deciding with probability at least $\frac{1}{2}$ whether G contains a subgraph isomorphic to T in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$. The algorithm is with one-sided error and reports no false positives.*

In other words, TREE CONTAINMENT ABOVE MINIMUM DEGREE admits a randomized FPT algorithm. We state Theorem 1.1 for the decision variant of the problem. However, the proof of the theorem is constructive and if it exists, the corresponding subgraph isomorphism can also be constructed in the same running time.

It is useful to compare and contrast Theorem 1.1 and the algorithm "above Dirac" from [16] that decides whether a connected graph contains a path of length at least $2\delta(G)+k$ in time $f(k)\cdot n^{\mathcal{O}(1)}$. On the one hand, the statement of Theorem 1.1 holds for any tree, not only paths. On the other hand, the "combinatorial threshold" in the "above Dirac" algorithm is $2\delta(G)$ and in Theorem 1.1 it is $\delta(G)$. While in the statement of Chvátal's Lemma the value $\delta(G)$ cannot be replaced by $(1+\varepsilon)\delta(G)$ for $\varepsilon>0$, it is not clear a priory that the threshold $\delta(G)$ in Theorem 1.1 cannot be increased. Our next theorem rules out this option.

THEOREM 1.2. *For any $\varepsilon>0$,* TREE CONTAINMENT *is* NP*-complete when restricted to instances (G,T) with $|V(T)|\leq(1+\varepsilon)\delta(G)$.*

Related work TREE CONTAINMENT plays an important role both in graph theory and in graph algorithms.

Extremal Graph Theory. According to Maya Stein [42]: *"One of the most intriguing open questions in the area is to determine degree conditions a graph G has to satisfy to ensure it contains a fixed tree T, or more generally, all trees of a fixed size."* While the conjecture of Erdős and Sós [9] about the average degree remains open, various other conditions have been suggested that might ensure the appearance of all trees or forests of some fixed size [3, 5, 28, 31, 34, 10]. We refer to the survey of Stein [42] for a comprehensive overview of the area.

Our work is also closely related to *stability theorems* in extremal combinatorics. Informally, a stability theorem establishes that an "almost nice" extremal structure can always be obtained by slightly modifying a "nice structure". For example, coming back to Dirac's and Erdős-Gallai theorems, there is a significant amount of literature devoted to sharper versions of these classic results [20, 19, 36, 33, 32, 43]. The typical statement of such results is that when we weaken the condition on the minimum vertex degree or an average degree of a graph, the graph contains a long cycle (or path) unless it possesses a very specific structure. To prove Theorem 1.1, we have to establish several stability variants of Chvátal's Lemma.

Algorithms. TREE CONTAINMENT is the special case of SUBGRAPH ISOMORPHISM, where the guest graph T is a tree. Matula in [40] gave a polynomial time algorithm for TREE CONTAINMENT when the host graph G is also a tree. According to Matoušek and Thomas [39], TREE CONTAINMENT is NP-complete when all vertices of T but one are of degree ≤ 3 and G is a treewidth 2 graph and all vertices of G but one are of degree ≤ 3. The result of Matoušek and Thomas shows a sharp difference in the complexity of TREE CONTAINMENT and the LONGEST PATH, which is FPT parameterized by the treewidth of G. The exhaustive study of SUBGRAPH ISOMORPHISM by Marx and Pilipczuk [38] establishes several hardness results about TREE CONTAINMENT for different classes of graphs G and trees T. There is a broad literature in graph algorithms on a related problem of finding a spanning tree in a graph with specified properties, see e.g. [41, 21, 23].

The seminal work of Alon, Yuster, and Zwick on color coding [2] shows that TREE CONTAINMENT is FPT parameterized by the size of T. In other words, deciding whether G contains a tree T of size t could be done in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$. Let us remark that in the setting of Theorem 1.1, the color coding method provides an algorithm of running time $2^{\mathcal{O}(\delta(G)+k)}n^{\mathcal{O}(1)}$, which is not FPT in k.

Several results in the literature provide FPT algorithms for long paths, and cycles parameterized above some degree conditions. Our work is an extension of this line of research to more general subgraph

isomorphism problems. Fomin et al. [13] gave FPT algorithms for computing long cycles and paths above the degeneracy of a graph. The tractability of these problems was extended by the authors in [17] above the so-called Erdős-Gallai bound, which is above the average vertex degree of a graph. In [15, 16], we established that finding a cycle above Dirac's bound is FPT. In other words, we gave an algorithm of running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ deciding whether a 2-connected graph G contains a cycle of length at least $\min\{2\delta(G), n\} + k$. The ideas and methods used to prove all the above results about cycles and paths are quite different from those we use in the proof of Theorem 1.1.

From a more general perspective, Theorem 1.1 belongs to a rich subfield of Parameterized Complexity concerning parameterization above/below specified guarantees [1, 7, 22, 24, 35, 37, 30]. We refer to the recent survey of Gutin and Mnich [25] for an overview of this area. In particular, the parameterized complexity of finding an (s,t)-path above the distance between vertices s and t, the DETOUR problem, attracted significant attention recently [4, 14, 12, 27, 29].

Structure of the paper In this extended abstract, we give a high-level overview of the proof of our main result Theorem 1.1 and omit the proof of Theorem 1.2. The details and complete proofs are available in the full arXiv version [18]. In Section 2, we introduce the notation used throughout the paper and provide auxiliary results. Section 3 contains the overview of the proof of Theorem 1.1. We conclude in Section 4 by stating some open problems.

2 Definitions and preliminaries

For a positive integer t, we define $[t] = \{1, \ldots, t\}$.

We use standard graph-theoretic notation and refer to the textbook of Diestel [8] for non-defined notions. We consider only finite simple undirected graphs. We use $V(G)$ and $E(G)$ to denote the sets of vertices and edges of a graph G, respectively; n and m are used to denote the number of vertices and edges if this does not create confusion. A vertex v is a *non-neighbor* of u if $v \neq u$ and $uv \notin E(G)$. For a graph G and a subset $X \subseteq V(G)$ of vertices, we write $G[X]$ to denote the subgraph of G induced by X. We use $G - X$ to denote the graph obtained by deleting the vertices of X, that is, $G - X = G[V(G) \setminus X]$; we write $G - v$ instead of $G - \{v\}$ for a single element set. For a vertex v, $N_G(v) = \{u \in V(G) \mid vu \in E(G)\}$ is the *open neighborhood* of v and $N_G[v] = N_G(v) \cup \{v\}$ is the *closed neighborhood*. For a set of vertices X, $N_G(X) = \left(\bigcup_{v \in X} N_G(v)\right) \setminus X$ and $N_G[X] = \bigcup_{v \in X} N_G[v]$. We use $\deg_G(v) = |N_G(v)|$ to denote the *degree* of a vertex v; $\Delta(G) = \max_{v \in V(G)} \deg_G(v)$ is the *maximum degree* of G and $\delta(G) = \min_{v \in V(G)} \deg_G(v)$ is the *minimum degree*. In the above notation, we may omit subscripts denoting graphs if this does not create confusion. We write $P = v_1 - \cdots - v_k$ to denote a (simple) path in a graph G with k vertices $v_1, \ldots, v_k$ of length $k-1$; v_1 and v_k are the *end-vertices* of G and we say that P is an (v_1, v_k)-path. The *diameter* of G, denoted $\operatorname{diam}(G)$, is the maximum length of a shortest (u,v)-path in G over all $u, v \in V(G)$. Two vertices u and v compose a *diametral* pair if the distance between them, i.e. the length of the shortest path, is $\operatorname{diam}(G)$.

An *isomorphism* of a graph H into a graph G, a bijective mapping $\varphi\colon V(H) \to V(G)$ such that $uv \in E(H)$ for $u, v \in V(H)$ if and only if $\varphi(u)\varphi(v) \in E(G)$. A *subgraph isomorphism* of H into G is an injective mapping $\sigma\colon V(H) \to V(G)$ such that $uv \in E(H)$ for $u, v \in V(H)$ if and only if $\varphi(u)\varphi(v) \in E(G)$. In words, this means that G contains H as a subgraph. We use $\operatorname{Im}\sigma$ to denote $\sigma(V(H))$.

Throughout our paper, we use the following specific notions.

DEFINITION 1. (MAXIMUM LEAF-DEGREE $\operatorname{ld}(T)$) *The* leaf-degree *of a vertex v in T is the number of leaves of T that are neighbors of v. The* maximum leaf-degree *of T, $\operatorname{ld}(T)$, is the maximum of the leaf-degrees over all vertices of T.*

DEFINITION 2. (NEIGHBOR DEFICIENCY $\nu_k(v)$) *For a graph G, an integer $k \geq 0$ and a vertex $v \in V(G)$ we define the* neighbor deficiency *of v as*

$$\nu_k(v) = \max\{(\delta(G) + k - 1) - \deg_G(v), 0\}.$$

If $\nu_k(v) = 0$, we say that v is non-deficient.

DEFINITION 3. (q-ESCAPE VERTEX) *For a graph G and integer q, a vertex v in G is an* q-escape vertex*, if $\deg_G(v) \geq \delta(G) + q$ or the maximum matching size between $N[v]$ and $V(G) \setminus N[v]$ in G is at least q.*

DEFINITION 4. (SIZE-q-SEPARABLE) *We say that a tree T is* size-q-separable *if there is an edge in T whose removal separates T into two subtrees consisting of at least q vertices each.*

Finally, we give here the following extension of Chvátal's Lemma (Lemma 1.1). While Chvátal's Lemma indeed provides the guarantee of $\delta(G)+1$ for TREE CONTAINMENT ABOVE MINIMUM DEGREE, throughout the proof of the main theorem, we often need a more general statement. Its proof repeats the original proof of Chvátal, and we provide it here for completeness.

PROPOSITION 2.1. ([6]) *Let G be a graph and let T be a tree on at most $\delta(G)+1$ vertices. Let $\sigma' : V(T') \to V(G)$ be a subgraph isomorphism mapping a connected subtree T' of T into G. Then there is a subgraph isomorphism $\sigma : V(T) \to V(G)$ mapping T into G such that σ is an extension of σ'.*

Proof. [Proof of Lemma 2.1] The proof is constructive. Let $T_0, T_1, \ldots, T_q$ be a sequence of trees such that $q = |V(T)| - |V(T')|$, $T_0 = T'$, $T_q = T$, and for each $i \in [q]$, $T_{i-1} = T_i - \ell_i$, where ℓ_i is a leaf of T_i that is not present in T'. One way to construct the sequence in reverse order is to start from $T_q := T$. Then, to obtain T_{i-1} from T_i for each consecutive i in $\{q, q-1, \ldots, 1\}$, we just delete a leaf of T_i that does not belong to T'. Since T' is a connected subtree of T_i, such a leaf always exists in T_i.

We then construct a series of subgraph isomorphisms $\sigma_0, \sigma_1, \ldots, \sigma_q$, such that for each $i \in \{0, \ldots, q\}$, σ_i is a subgraph isomorphism of T_i into G. We start from $\sigma_0 = \sigma'$. Then consecutively for each $i \in [q]$, σ_i is obtained by extending σ_{i-1} on ℓ_i. The leaf ℓ_i has only one neighbor s_i in T_i. Then the image of ℓ_i in σ_i should be the neighbor of $\sigma_{i-1}(s_i)$. Since $|\operatorname{Im}\sigma_{i-1}| \leq \delta(G)$, and $\sigma_{i-1}(s_i) \in \operatorname{Im}\sigma_{i-1}$, $\operatorname{Im}\sigma_{i-1}$ contains at most $\delta(G) - 1$ neighbors of $\sigma_{i-1}(s_i)$ in G. Then at least one neighbor of the image of s_i is not occupied by σ_{i-1}. We obtain σ_i from σ_{i-1} by extending mapping ℓ_i to such a neighbor.

The procedure produces the subgraph isomorphism σ_q of $T_q = T$ into G. ∎

3 Main ideas and structure of the proof of Theorem 1.1

In this overview, we will provide some intuition on how several various structural cases for G and T guarantee that G contains T. In several situations, we push the structural analysis to the limit. In the remaining cases, when the structural analyses (or stability theorems) cannot be pushed further, the obtained structural properties allow us to design algorithms. Such a WIN/WIN approach results is a randomized FPT algorithm running in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$. The order of presentation of the case analysis in the overview slightly disagrees with the order of the full version [18]. In the overview, we aimed to make the presentation more intuitive while in [18], we put parts using the same techniques and ideas closer to each other.

The natural way to proceed above the Chvátal's $\delta(G) + 1$ guarantee is to ask whether G contains an arbitrary T of size at most $\delta(G) + 2$. The answer to this question appears to be quite simple, yet it settles the starting point of our work. We state it as Proposition 3.1. While this result is not explicitly used in the proof of the main theorem, its proof exposes the key ideas we use to prove the main theorem.

PROPOSITION 3.1. *Every connected graph G contains every tree on at most $\min\{|V(G)|, \delta(G)+2\}$ vertices, unless G is $\delta(G)$-regular and T is isomorphic to the star graph $K_{1,\delta(G)+1}$ with $\delta(G)+1$ leaves.*

Proof. [Proof of Proposition 3.1] Note that the only case that has to be proved above Chvátal's Lemma is when $|V(T)| = \delta(G) + 2$.

We have that G is connected and has at least $\delta(G) + 2$ vertices. Let T be an arbitrary tree on exactly $\delta(G) + 2$ vertices. Clearly, if T has a vertex of degree $\delta(G) + 1$ but the maximum degree of G equals $\delta(G)$ then G does not contain T as a subgraph. This is equivalent to T being isomorphic to $K_{1,\delta(G)+1}$ and G being $\delta(G)$-regular. It is left to show that in any other case, G contains T.

Assume first that G has a vertex of degree at least $\delta(G) + 1$. Denote this vertex by u. Since T has at least two vertices, there is a vertex in T adjacent to a leaf. Denote this vertex by t and its adjacent leaf

by ℓ. We start constructing a mapping of T into G by mapping t to u. Since $T-\ell$ is a tree consisting of exactly $\delta(G)+1$ vertices, we apply Proposition 2.1 and extend mapping $t \to u$ to a subgraph isomorphism σ of $T-\ell$ into G. The size of $N_G[u]$ is at least $\delta(G)+2$. Hence, at least one vertex in $N_G[u]$ is not used by σ. We extend σ by mapping ℓ to this vertex. Then σ becomes a subgraph isomorphism of T into G. That is, if G has a vertex of degree at least $\delta(G)+1$ then G contains T.

The remaining case is when all vertices of G are of degree $\delta(G)$ but T is not isomorphic to a star. Again, we take a leaf ℓ and its neighbor vertex t in T. Since T is not a star, there is a neighbor of t in T that is not a leaf. Denote this neighbor by x. Then x has at least one neighbor distinct from t, denote it by y.

We have a path on three vertices $t-x-y$ in the tree $T-\ell$ and we map this path into G as follows. Take an arbitrary vertex u in G. Since G is connected and $|V(G)| > \delta(G)+1 = |N_G[u]|$, there is at least one edge $vw \in E(G)$ such that $v \in N_G[u]$ and $w \in V(G) \setminus N_G[u]$. We have that $u-v-w$ is a path in G where u is not adjacent to w. We initiate the construction of a subgraph isomorphism of T into G by mapping t, x, y into u, v, w respectively. Then, by making use of Proposition 2.1, we extend this mapping into a subgraph isomorphism σ of $T-\ell$ into G.

In contrast to the previous case, the closed neighborhood of u, $N_G[u]$ is of size $\delta(G)+1$. But we ensured that σ uses at least one vertex outside $N_G[u]$ by mapping y to w initially. Hence, $N_G[u] \setminus \operatorname{Im}\sigma$ is not empty. Therefore, we can extend σ by mapping ℓ to an arbitrary vertex in $N_G[u] \setminus \operatorname{Im}\sigma$. The obtained mapping is a subgraph isomorphism of T into G. The proof of the proposition is complete. □

Let us highlight and discuss the key ideas of the proof above.

Idea I. *Saving space for mapping leaves starts from mapping their neighbors.*
In the proof of Proposition 3.1, we cut off a leaf ℓ of T and decide where it will be mapped later. Since the pruned tree has exactly $\delta(G)+1$ vertices, we can map it into G by Lemma 1.1. However, we have to make sure that the removed leaf can be mapped to a free vertex. To achieve that, we initially set the image of its neighbor t to a specific vertex in G. If G has a vertex of degree more than $\delta(G)$ then any such vertex will do as an image of t. This is because the isomorphism can occupy at most $\delta(G)$ of its neighbors, so one of them is always vacant to host ℓ. The other case occurs when G is regular and hence has no vertex of a large degree. This brings us to the next key idea.

Idea II. *Saving space for leaves is achieved by mapping outside specific sets.*
When we deal with the case of a regular graph G, we map t to an arbitrary vertex u of degree $\delta(G)$. Now a subgraph isomorphism of $T-\ell$ into G can occupy all $\delta(G)$ neighbors of u. If this happens, then the isomorphism uses no vertex outside $N_G[u]$ since already $\delta(G)+1$ vertices are occupied. Thus, if we force the subgraph isomorphism to use at least one vertex outside $N_G[u]$, then at least one vertex in $N_G[u]$ would be saved. This is exactly what we do in the proof of Proposition 3.1. We initially map a path of three vertices in T starting in t into a path of three vertices in G starting in u making sure that the last vertex of this path in G is not a neighbor of u. When this isomorphism is extended onto $T-\ell$, it automatically leaves at least one neighbor of u in G unused. This neighbor finally becomes the image of ℓ.

These two ideas bring to a polynomial time algorithm for finding trees of size $\delta(G)+2$ in a graph G. It appears that we can push the applicability of these ideas further. However, it does not come without additional effort. Let us provide some intuition on how the two ideas could be extended to the case when T is a tree on $\delta(G)+k$ vertices for $k \geq 3$.

3.1 Saving neighbors of a single vertex Let T be a tree on $\delta(G)+k$ vertices for $k \geq 3$. The first question arising when we try to adapt Proposition 3.1 to T is the following: If there is a vertex t in T with at least $k-1$ adjacent leaves, can we shave off $k-1$ leaves t and repeat the same arguments to map the shaved tree and save $k-1$ neighbors of the image of t? This would allow us to map the shaved leaves into the saved neighbors and obtain the subgraph isomorphism of T into G. According to the notion of the *maximum-leaf degree* (see Definition 1), the existence of a vertex t in T is equivalent to the condition $\mathrm{ld}(T) \geq k-1$.

Note that different possible images of t can require different numbers of saved neighbors. For example, if $\deg_G(u) \geq \delta(G)+k-1$, where u is the image of t, then saving neighbors is not required at all, as $k-1$ neighbors of u remain vacant after mapping the shaved tree of size $\delta(G)+1$. The less the degree of u is,

the more vertices are required to be mapped outside $N_G[u]$. We use $\nu_k(u)$, the *neighbor deficiency* of u, to denote the number of vertices to map outside $N_G[u]$ (see Definition 2).

In the proof of Proposition 3.1, we initially mapped a path of three vertices in order to map one vertex outside $N_G[u]$. We push this method further and show that mapping a path of length $3k$ (starting from t in T) initially can save $\nu_k(u)$ neighbors of some vertex u in G. This is achieved by mapping the path in T into a path in G starting in u that has roughly every third vertex outside $N_G[u]$. The choice of u depends drastically on the structure of G. The sufficient requirement for u is having enough vertices in $N_G[u]$ that have more than $k-1$ neighbors outside $N_G[u]$.

If such a choice of u is not possible, the regular "map every third outside" procedure is not possible. In this case, we show that the minimum vertex degree of graph $G - N_G[u]$ is at least k. Such a graph has a path of length at least k starting in an arbitrary vertex. Then we construct a path in G that starts in u, goes to some vertex v in $V(G) \setminus N_G[u]$ through an intermediate vertex in $N_G(u)$, and ends by following a path of length $k-2$ inside $G - N_G[u]$. This path is of length k and has its last $k-1$ vertices outside $N_G[u]$. It is easy to see that mapping the first $k+1$ vertices of the path in T into the constructed path in G saves $k-1 \geq \nu_k(u)$ neighbors of u.

However, this technique is not applicable when (i) T has no path of length at least $3k$ starting in t *or* (ii) the minimum degree of G is in $\mathcal{O}(k^2)$. In each of these cases, containment of T in G is not guaranteed and requires an algorithmic approach. We handle each case separately. The second case is easier than the first: It guarantees that the size of T is bounded by $\mathcal{O}(k^2)$. Since Tree Containment can be solved in $2^{\mathcal{O}(|V(T)|)} \cdot n^{\mathcal{O}(1)}$ running time using the color-coding technique of Alon, Yuster and Zwick [2], we achieve a time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ algorithm for the case $\delta(G) = \mathcal{O}(k^2)$. We highlight this idea for further use in the overview.

Idea III. *Color-coding applies for graphs of minimum degree bounded by $k^{\mathcal{O}(1)}$.*

It remains to discuss case (i) when T has no path of length at least $3k$ starting in t. The algorithm for this case also involves color-coding. The initial step towards the algorithm here is to note that T has a bounded diameter and so does every subtree of T. For the previous cases, we constructed a path-to-path isomorphism that hits enough vertices in $V(G) \setminus N_G[u]$, where u is the fixed[2] image of t. In this case, T has no path of enough length starting in u. However, we know that if T is isomorphic to a subgraph of G, then the subgraph isomorphism of T into G occupies at least $\nu_k(u)$ vertices outside $N_G[u]$. Thus, solving the problem is equivalent to finding an isomorphism of a connected subtree of T containing t into G that maps t to u and also maps at least $\nu_k(u)$ vertices of T to vertices in G that are outside $N_G[u]$.

We aim to find such a subtree of minimum size, the existence of such a tree is equivalent to the containment of T in G. By minimality, the leaves of such a subtree are necessarily mapped to either t or into $V(G) \setminus N_G[u]$. Then by the minimality, this tree should have at most $\nu_k(u)+1$ leaves, so the size of any minimal subtree is at most $\operatorname{diam}(T) \cdot (\nu_k(u)+1) \leq k \cdot \operatorname{diam}(T)$. The obtained bound allows us to use color-coding to find the required subtree in G. We color G with $k \cdot \operatorname{diam}(T)$ colors uniformly at random and use the coloring to find a subgraph of G that is (a) isomorphic to the required subtree of T, (b) the isomorphism maps t to u, and (c) the isomorphism occupies $\nu_k(u)$ vertices in $V(G) \setminus N_G[u]$. This is done via dynamic programming in time $2^{\mathcal{O}(k \cdot \operatorname{diam}(T))} \cdot n^{\mathcal{O}(1)}$. As there is no vertex at a distance at least $3k$ from t in T, we have that $\operatorname{diam}(T) = \mathcal{O}(k)$. Hence the overall running time of the algorithm is $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$, similar to the case (ii).

This algorithm is important not only for the particular case $\operatorname{ld}(T) \geq k-1$—we shall recall it once again in the overview in a slightly more general setting for the problem which we call Annotated Hitting Subtree Containment. In this problem, it is allowed to have an arbitrary starting mapping from T into G and have multiple sets in G that are required to be hit by the isomorphism. Our algorithm for this problem works in time $2^{k^{\mathcal{O}(1)} \cdot \operatorname{diam}(T)} \cdot n^{\mathcal{O}(1)}$ if the total number of vertices to hit by the isomorphism is bounded by $k^{\mathcal{O}(1)}$.

[2]The choice of the vertex u in G is fixed in the outer loop of the algorithm. Thus, the algorithm considers every possible vertex in G as a candidate for the image of t.

The above arguments bring us to an algorithm for Tree Containment running in time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$, where $k = |V(T)| - \delta(G)$, for the special case $\mathrm{ld}(T) \geq k-1$.

3.2 Filtering out yes-instances We still have to consider the case of the maximum-leaf degree $\mathrm{ld}(T) < k-1$. As in the case $\mathrm{ld}(T) \geq k-1$, we first filter out some structural properties of G and T that guarantee G containing T. Taking the success of the complement case into account, a good candidate for being filtered out first is the case of T having less than $k-1$ leaves (that actually stops us from applying Idea II by shaving leaves).

In this case, T can be covered by at most $k-3$ (the number of leaves minus one) paths starting in the same arbitrary leaf of T and ending in pairwise distinct leaves of T. Each of these paths consists of at most $\mathrm{diam}(T)+1$ vertices. Since these paths share at least one common vertex, we obtain $|V(T)| \leq 1 + (k-3) \cdot \mathrm{diam}(T) < k \cdot \mathrm{diam}(T)$. It follows that $\mathrm{diam}(T)$ is at least $\delta(G)/k$. As we can handle small values of $\delta(G)$ by making use of Idea III, we need to look only at trees of large diameter.

Up to this point, we still have not introduced any mechanism of constructing isomorphisms of T into G other than mapping outside a neighborhood of a *specific* vertex. To avoid keeping track of this, we discover that it is possible to save neighbors of *all* vertices of G simultaneously. Moreover, path-to-path isomorphisms are very suitable for this purpose. This is the next idea in the proof of the main result

Idea IV. *Shortest paths are very good for saving neighbors.*
Assume that G has a shortest (s,t)-path that consists of at least $k+2$ vertices. On the one hand, each vertex of G that does not belong to this path cannot have more than three neighbors belonging to this path (otherwise we can make the (s,t)-path shorter). On the other hand, each vertex of the (s,t)-path cannot have more than two neighbors inside the path for the same reason. Then every vertex in G has at least $k-1$ non-neighbors in the path. Hence, if we map a path of T into the (s,t)-path initially, we can extend this mapping to a subgraph isomorphism of T into G without any additional work following Idea II. Due to that for each $u \in V(G)$, we have at least $k-1 \geq \nu_k(u)$ non-neighbors in the isomorphism initially.

The idea above is the basis for an alternative mechanism that allows us to deal with G for the case $\mathrm{diam}(G) \geq k+1$ automatically and the additional work is required only to deal with the case $\mathrm{diam}(G) \leq k$. We turn this upper bound on the diameter of G into our advantage. In fact, we are able to take any set S of size $o(\delta(G))$ and construct a path of size $\mathcal{O}(k \cdot |S|)$ that traverses all vertices of S. We do this iteratively by connecting consecutive vertices in S by a shortest path in G. Since we do not want such segments to intersect, we have to remove the prefix of the path from G before finding the shortest path to the next vertex of S. If during this process the diameter of G becomes even greater than $2k$, we make good use of it (following Idea IV with additional arguments) and construct a path that contains enough non-neighbors of each vertex of G.

Now we discuss finding the appropriate set S in G. This set is required to contain at least $\nu_k(u)$ non-neighbors for every vertex $u \in V(G)$. If the size of the graph is slightly above $\delta(G)$, that is, $|V(G)| \leq (1+\epsilon) \cdot \delta(G)$, the existence of a small S is hardly possible. For this reason, we separately deal with his case for arbitrary $\epsilon \leq \frac{1}{4k}$. We show that if $\delta(G) = \Omega(k^2)$, $|V(G)| \leq (1+\epsilon) \cdot \delta(G)$ and $\mathrm{ld}(T) < k$, then G contains T as a subgraph. The proof is based on extending a partial isomorphism of T into G using unoccupied vertices. This is always possible to do because the vertices outside the partial isomorphism in G should have many neighbors inside it. It grants many options for an outer vertex to be inserted between the vertices of the isomorphism, and at least one option will always suffice for extension.

By achieving the lower-bound $|V(G)| \geq (1+\epsilon) \cdot \delta(G)$, we are able to construct the set S of bounded size. We use the probabilistic method here. For simplicity, we slightly increase the lower bound for the number of vertices up to $|V(G)| \geq (1+\epsilon) \cdot \delta(G) + k^{\Omega(1)} \cdot \log \delta(G)$. (The increase is achieved by lowering the value of ϵ.) Each vertex u in G with $\nu_k(u) > 0$ has at least $|V(G)| - \delta(G) > \epsilon \cdot \delta(G)$ non-neighbors in G. For a fixed $u \in V(G)$, a random choice of a vertex in $V(G)$ gives a non-neighbor of u with probability at least $\frac{\epsilon}{1+\epsilon}$. Then the expected value of deficient vertices $u \in V(G)$ that are not neighbors of a random vertex of G is at least $\frac{\epsilon}{1+\epsilon} \cdot |V(G)|$, so there should be a single vertex in G that is non-neighbor to at least $\frac{\epsilon}{1+\epsilon}$ ratio of all deficient vertices. We find such vertex in G and put it inside S. With a careful analysis, we show that repeating this step for $\mathcal{O}(\log \delta(G)/\epsilon)$ times results in a set S having at least one non-neighbor for each

deficient vertex of G. We have to repeat the whole process $k-2$ times and obtain the required set S of size $\mathcal{O}(k \cdot \log \delta(G)/\epsilon)$. Note that the additive part of the lower bound is required for maintaining the probability $\frac{\epsilon}{1+\epsilon}$ during the consecutive choices of distinct vertices into S. The value of ϵ we use is of the form $\frac{1}{k^{\Theta(1)}}$, so the final bound on the size of S is $k^{\mathcal{O}(1)} \cdot \log \delta(G)$.

The vertices of S are finally tied into a path in G of length at most $k^{\mathcal{O}(1)} \cdot \log \delta(G)$ as discussed above, and a path of sufficient length in T is mapped to the path in G. The discussion sums up into filtering out trees of diameter $k^{\Omega(1)} \cdot \log \delta(G)$, and the gap between this bound and the desirable $k^{\mathcal{O}(1)}$ still remains.

3.3 Shaving off leaves from distinct neighbors The results discussed above leave us with a tree T satisfying $\mathrm{ld}(T) < k-1$ and $\mathrm{diam}(T) \leq k^{\mathcal{O}(1)} \cdot \log \delta(G)$. The number of leaves in T, in this case, is at least $k^{\Omega(1)}$ (following the $\delta(G)/\mathrm{diam}(T)$ lower bound discussed before). Then we can choose at most $k-1$ vertices in T, such that in total they have at least $k-1$ adjacent leaves. We denote this set by W. The strategy we want to implement in this case is to use Idea II and to construct a mapping of a connected subtree of T containing the vertices of W into G such that at least $\nu_k(w)$ non-neighbors of each $w \in W$ are occupied by this mapping. The formal proof encapsulates several methods of constructing a mapping that hits non-neighbors of W by exploiting the structure of T and G. This is the final step of filtering out yes-instances before applying the last (and the most involved) of the algorithms we use to prove the main theorem.

The first obstacle encountered here is that the mapping we require to construct should map all vertices of W (as we should know their respective images in G in order to collect non-neighbors). To our advantage, the choice of W can vary (while $N_T(W)$ has at least $k-1$ leaves) and we will make this choice depending on the structure of T.

The initial step here filters out the case of T with $\mathrm{diam}(T) \geq \Omega(k^4)$. First, W is chosen in a way that it contains two vertices on distance exactly $\mathrm{diam}(T)-2$. Then we consider the minimum spanning tree T_W of W in T. Its leaves form a subset of T, so the number of leaves in T_W is at most $k-1$. Since T_W has a large diameter, there exist long paths in T_W consisting of degree-2 vertices, which we refer to as *trivial path*. We further shrink T_W by contracting edges of each long trivial path down until its length reaches $2k$. The shrank tree T'_W has at most $\mathcal{O}(k^2)$ vertices and we initialize the mapping with an arbitrary subgraph isomorphism of T'_W into G.

The rest of the work is to transform this isomorphism to the isomorphism of T_W into G by embedding vertices of G into trivial paths of the isomorphism image. We exploit the existence of a trivial path of length $\Omega(k^3)$ and embed a set S (containing $k-1$ non-neighbors for the image of each $w \in W$) into the image of this path. The technical work here is done using the arguments that we discussed already. If it cannot be done at some moment, then Idea IV helps us to construct a mapping in an alternative way.

The case of $\mathrm{diam}(T) \geq \Omega(k^4)$ is now dealt with. To proceed further, we cannot rely on path-to-path isomorphisms anymore since T has no very long paths. Then we focus more on the structure of G. This is when we put the notion of q-escape vertices into play (see Definition 3). We start with an isomorphism of T_W into G and show how a $k^{\Omega(1)}$-escape vertex helps in embedding the isomorphism with non-neighbors of W. This, however, requires a vertex of degree $k^{\Omega(1)}$ in T.

We partially resolve this issue by making additional assumptions of the structure of T. For this, we give an alternative (to exploiting an escape vertex) mechanism to extend a mapping of T_W with enough non-neighbors of W. If this mechanism fails, it produces a separator of G of size $k^{\Omega(1)}$. Finally, we use this separator to map a size-$k^{\mathcal{O}(1)}$-separable tree T (see Definition 4) inside G.

3.4 Solving remaining case algorithmically We move on to the remaining case of Tree Containment Above Minimum Degree: G contains no $k^{\Omega(1)}$-escape vertices and T is not size-$k^{\Omega(1)}$-separable, while $\delta(G) \geq k^{\Omega(1)}$. We prove that there is a randomized algorithm solving such cases in $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ running time. In the rest of the current part of the overview, we discuss the main parts of the proof of this result.

Following Idea II, the first part addresses hitting non-neighbors of images of vertices in W (the set of leaf-adjacent vertices in T) similar to the previous part of the overview. Before we used it to prove that G

contains T as a subgraph. That was achieved by occupying at least $\nu_k(w)$ non-neighbors of the image of w in G for each $w \in W$. But this is only a sufficient condition. In fact, not all vertices in W require so many saved neighbors. For example, if W has exactly $k-1$ vertices then we require exactly one vacant neighbor when it comes to mapping the leaf to the neighbor of the image of $w \in W$. Then if we have one vacant neighbor for w_1, two vacant neighbors for w_2, three vacant neighbors for w_3, and so on, the extension of the isomorphism is guaranteed to be possible.

The paragraph above suggests that a necessary condition should be discovered. We prove that if the mapping of W into G is known initially then the existence of a subgraph isomorphism of T into G respecting this mapping is equivalent to the existence of a mapping that hits specific sets in G. The details of this are quite complex, so we refer the reader to the full version [18]. It automatically provides a one-to-many reduction to Annotated Hitting Subtree Containment. Since the diameter of T is at most $k^{\mathcal{O}(1)}$, we obtain an algorithm that runs in $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ time and correctly decides the containment of T in G, provided the mapping of W into G is given.

This part alone does not provide any clue on how to choose the mapping of W into G. Trivial enumeration of all possible mappings gives $|V(G)|^{|W|} < |V(G)|^k$ possible options and yield only an XP-algortihm for Tree Containment parameterized by k. To resolve this issue, we guess mapping using random sampling. This leads to a polynomial-time randomized procedure that takes G, T, W and a single vertex $u \in V(G)$ as input and produces a mapping of W into G. If T is contained in G as a subgraph then with probability at least $2^{-k^{\mathcal{O}(1)}}$, this mapping is a restriction of some subgraph isomorphism of T into G. This is the only source of randomness in our algorithm which we do not know how to derandomize. Combining the results of two parts we obtain a (one-sided error) randomized algorithm for the specific case of Tree Containment with running time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$.

4 Conclusion

In our exploration of algorithmic extensions of classical combinatorial theorems, we have demonstrated that it is possible to determine, in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$, whether a graph G contains a tree T with at most $\delta(G)+k$ vertices as a subgraph. Our algorithm is a one-sided error Monte Carlo algorithm. This naturally raises two questions. First, can we develop a deterministic algorithm for this problem? Second, is there room for improvement in the running time? Can the problem be solved in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ or even in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?

Another question related to our work. Brandt, in his work [5], extended Chvátal's Lemma for forests.

Proposition 4.1. ([5]) *Let G be a graph, and F be a forest such that $|V(F)| \leq |V(G)|$ and $|E(F)| \leq \delta(G)$. Then, G contains F as a subgraph.*

Is the Forest Containment problem (for a given graph G and forest F, to decide whether G contains F) FPT when parameterized by $k = |E(F)| - \delta(G)$?

Acknowledgements The research leading to these results has been supported by the Research Council of Norway via the project BWCA (grant no. 314528) and DFG Research Group ADYN via grant DFG 411362735. The authors would like to express their deepest gratitude to Dimitrios Thilikos, who suggested that it would be interesting to investigate Chvátal's Lemma from an algorithmic viewpoint.

References

[1] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, *Solving MAX-r-SAT above a tight lower bound*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2010, pp. 511–517.
[2] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856.
[3] C. S. Babu and A. A. Diwan, *Degree conditions for forests in graphs*, Discrete Math., 301 (2005), pp. 228–231.
[4] I. Bezáková, R. Curticapean, H. Dell, and F. V. Fomin, *Finding detours is fixed-parameter tractable*, SIAM J. Discrete Math., 33 (2019), pp. 2326–2345.

[5] S. Brandt, *Subtrees and subforests of graphs*, J. Combin. Theory Ser. B, 61 (1994), pp. 63–70.
[6] V. Chvátal, *Tree-complete graph Ramsey numbers*, J. Graph Theory, 1 (1977), p. 93.
[7] R. Crowston, M. Jones, G. Muciaccia, G. Philip, A. Rai, and S. Saurabh, *Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound*, in IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), vol. 24 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2013, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 43–54.
[8] R. Diestel, *Graph theory*, vol. 173 of Graduate Texts in Mathematics, Springer-Verlag, Berlin, 5th ed., 2017.
[9] P. Erdős, *Extremal problems in graph theory*, in Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963), Publ. House Czech. Acad. Sci., Prague, 1964, pp. 29–36.
[10] P. Erdős, Z. Füredi, M. Loebl, and V. T. Sós, *Discrepancy of trees*, Studia Sci. Math. Hungar., 30 (1995), pp. 47–57.
[11] P. Erdős and T. Gallai, *On maximal paths and circuits of graphs*, Acta Math. Acad. Sci. Hungar, 10 (1959), pp. 337–356.
[12] F. V. Fomin, P. A. Golovach, W. Lochet, D. Sagunov, K. Simonov, and S. Saurabh, *Detours in directed graphs*, in 39th International Symposium on Theoretical Aspects of Computer Science (STACS), vol. 219 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 29:1–29:16.
[13] F. V. Fomin, P. A. Golovach, D. Lokshtanov, F. Panolan, S. Saurabh, and M. Zehavi, *Going far from degeneracy*, SIAM J. Discret. Math., 34 (2020), pp. 1587–1601.
[14] ———, *Multiplicative parameterization above a guarantee*, ACM Trans. Comput. Theory, 13 (2021), pp. 18:1–18:16.
[15] F. V. Fomin, P. A. Golovach, D. Sagunov, and K. Simonov, *Algorithmic extensions of Dirac's theorem*, CoRR, abs/2011.03619 (2020).
[16] ———, *Algorithmic extensions of Dirac's theorem*, in Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2022, pp. 406–416.
[17] ———, *Longest cycle above Erdős-Gallai bound*, in 30th Annual European Symposium on Algorithms (ESA), vol. 244 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 55:1–55:15.
[18] ———, *Tree containment above minimum degree is fpt*, CoRR, abs/2310.09678 (2023).
[19] Z. Füredi, A. Kostochka, R. Luo, and J. Verstraëte, *Stability in the Erdős-Gallai theorem on cycles and paths, II*, Discrete Math., 341 (2018), pp. 1253–1263.
[20] Z. Füredi, A. Kostochka, and J. Verstraëte, *Stability in the Erdős-Gallai theorems on cycles and paths*, J. Combin. Theory Ser. B, 121 (2016), pp. 197–228.
[21] M. Fürer and B. Raghavachari, *Approximating the minimum degree spanning tree to within one from the optimal degree*, in Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms (SODA), 1992, pp. 317–324.
[22] S. Garg and G. Philip, *Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2016, pp. 1152–1166.
[23] M. X. Goemans, *Minimum bounded degree spanning trees*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, 2006, pp. 273–282.
[24] G. Gutin, L. van Iersel, M. Mnich, and A. Yeo, *Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables*, J. Computer and System Sciences, 78 (2012), pp. 151–163.
[25] G. Z. Gutin and M. Mnich, *A survey on graph problems parameterized above and below guaranteed values*, CoRR, abs/2207.12278 (2022).
[26] J. Han and P. Keevash, *Finding perfect matchings in dense hypergraphs*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2020, pp. 2366–2377.
[27] M. Hatzel, K. Majewski, M. Pilipczuk, and M. Sokolowski, *Simpler and faster algorithms for detours in planar digraphs*, in 2023 Symposium on Simplicity in Algorithms (SOSA), SIAM, 2023, pp. 156–165.
[28] F. Havet, B. Reed, M. Stein, and D. R. Wood, *A variant of the Erdös-Sós conjecture*, J. Graph Theory, 94 (2020), pp. 131–158.
[29] A. Jacob, M. Wlodarczyk, and M. Zehavi, *Long directed detours: Reduction to 2-disjoint paths*, CoRR, abs/2301.06105 (2023).
[30] B. M. P. Jansen, L. Kozma, and J. Nederlof, *Hamiltonicity below Dirac's condition*, in Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), vol. 11789 of Lecture Notes in Computer Science, Springer, 2019, pp. 27–39.

[31] J. Komlós, G. N. Sárközy, and E. Szemerédi, *Proof of a packing conjecture of Bollobás*, Combin. Probab. Comput., 4 (1995), pp. 241–255.
[32] G. N. Kopylov, *Maximal paths and cycles in a graph.*, Dokl. Akad. Nauk SSSR, (1977), pp. 19–21.
[33] B. Li and B. Ning, *A strengthening of Erdős-Gallai Theorem and proof of Woodall's conjecture*, J. Combin. Theory Ser. B, 146 (2021), pp. 76–95.
[34] B. Lidický, H. Liu, and C. Palmer, *On the Turán number of forests*, Electron. J. Combin., 20 (2013), pp. Paper 62, 13.
[35] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *Faster parameterized algorithms using linear programming*, ACM Trans. Algorithms, 11 (2014), pp. 15:1–15:31.
[36] J. Ma and B. Ning, *Stability results on the circumference of a graph*, Combinatorica, 40 (2020), pp. 105–147.
[37] M. Mahajan, V. Raman, and S. Sikdar, *Parameterizing above or below guaranteed values*, J. Computer and System Sciences, 75 (2009), pp. 137–153.
[38] D. Marx and M. Pilipczuk, *Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask)*, CoRR, abs/1307.2187 (2013).
[39] J. Matoušek and R. Thomas, *On the complexity of finding iso- and other morphisms for partial k-trees*, Discret. Math., 108 (1992), pp. 343–364.
[40] D. W. Matula, *Subtree isomorphism in $O(n^{5/2})$*, in Annals of Discrete Mathematics, vol. 2, Elsevier, 1978, pp. 91–106.
[41] C. H. Papadimitriou and M. Yannakakis, *The complexity of restricted spanning tree problems*, Journal of the ACM, 29 (1982), pp. 285–309.
[42] M. Stein, *Tree containment and degree conditions*, in Discrete mathematics and applications, vol. 165 of Springer Optim. Appl., Springer, Cham, 2020, pp. 459–486.
[43] X. Zhu, E. Győri, Z. He, Z. Lv, N. Salia, and C. Xiao, *Stability version of dirac's theorem and its applications for generalized turán problems*, 2022.