A Survey on Fixed-Parameter Tractability and Related Concepts in Computational Complexity

Abstract

Fixed-parameter tractability (FPT) is a crucial concept in computational complexity, offering a framework for efficiently solving NP-hard problems by isolating complexity into specific parameters. This survey explores FPT's role in algorithm design, particularly in dynamic graph algorithms and optimization problems. It examines kernelization, a preprocessing technique that reduces problem size while preserving solvability, and its integration with approximation strategies. The survey also delves into treewidth, a measure of graph tree-likeness, highlighting its significance in designing efficient graph algorithms. The W-hierarchy is discussed as a classification system for parameterized problems, with emphasis on W[1]-hardness and its implications for algorithmic tractability. The Exponential Time Hypothesis (ETH) is explored for its role in establishing complexity lower bounds, influencing kernelization, and guiding the development of parameterized reductions. Parameterized reduction is presented as a method for transforming one parameterized problem into another, aiding in complexity classification. The survey concludes with a discussion on the challenges and future directions in FPT research, including the development of algorithms for W[1]-hard problems, exploration of new parameterizations, and refinement of kernelization techniques. By synthesizing these aspects, the survey underscores FPT's pivotal role in advancing computational complexity understanding and developing innovative algorithmic solutions.

1 Introduction

1.1 Significance of Fixed-Parameter Tractability (FPT)

Fixed-Parameter Tractability (FPT) is a crucial concept in computational complexity, providing a framework for tackling NP-hard problems by isolating complexity into specific parameters. This allows for the development of efficient algorithms for fixed parameters, yielding solutions to otherwise intractable problems [1]. FPT's significance is particularly evident in dynamic graph algorithms, facilitating efficient solutions for critical problems like Vertex Cover and Cluster Vertex Deletion, which are essential for optimizing network structures [2].

FPT extends its utility to parameterized complexity, aiding in the resolution of complex optimization issues, such as the Path Set Packing problem [3]. Moreover, it enhances our understanding of structural properties in problems like the Firefighting problem, which models fire spread on graphs and necessitates strategic vertex defense [4].

The intersection of FPT with graph theory is notable, particularly in problems parameterized by treewidth, such as the List Edge Chromatic Number and List Total Chromatic Number, both of which are fixed-parameter tractable [5]. This is exemplified in the Graph Isomorphism problem, where FPT methods have been employed to devise tractable solutions when parameterized by treewidth [6]. Additionally, the exploration of flat foldability through parameters like ply and treewidth underscores FPT's adaptability across diverse problem settings [7].

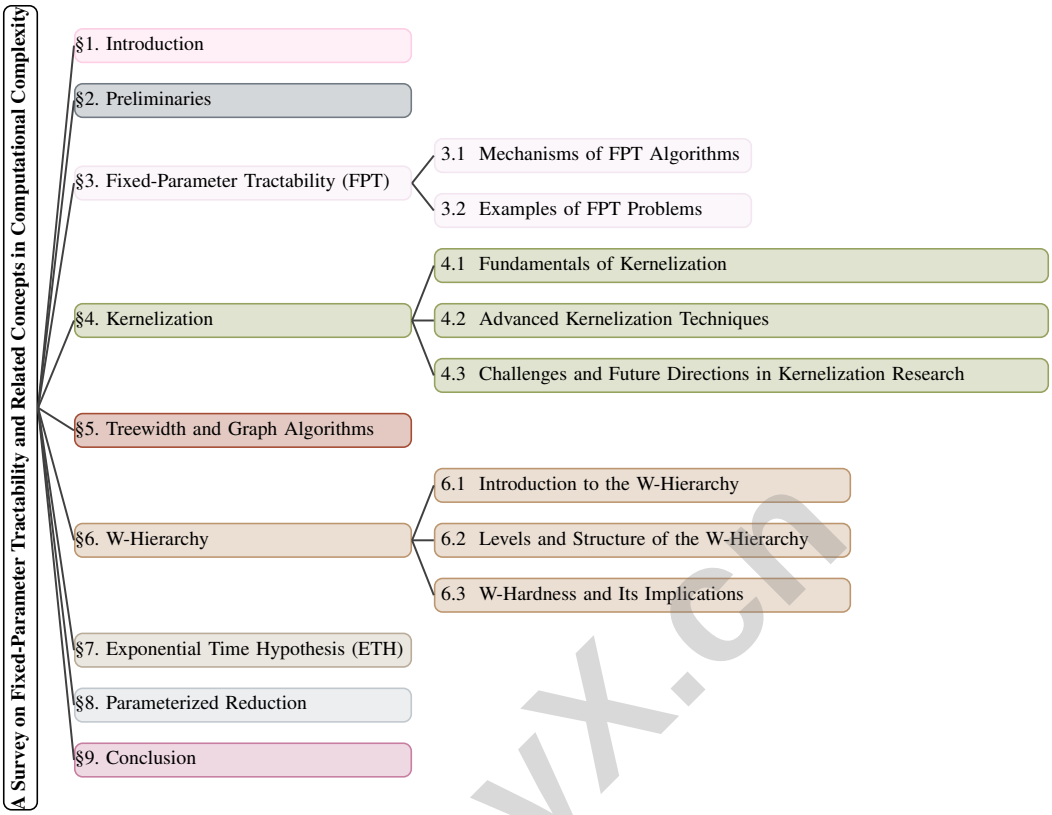


Figure 1: chapter structure

Furthermore, FPT is integral to preprocessing techniques such as kernelization, which enhance algorithm performance by reducing problem size while preserving solvability [8]. The application of FPT in parameterized enumeration problems drives the development of new methods for achieving parameter-efficient enumeration algorithms, further highlighting its importance in computational complexity.

1.2 Structure of the Survey

This survey is structured to provide a thorough exploration of fixed-parameter tractability (FPT) and its related concepts in computational complexity. It comprises several key sections, each dedicated to specific aspects of the topic.

The survey begins with an **Introduction**, introducing FPT and its significance in computational complexity and NP-hard problem-solving, setting the stage for subsequent discussions.

Section 2, Preliminaries, defines and explains core concepts relevant to the survey, including FPT, kernelization, treewidth, the W-hierarchy, the Exponential Time Hypothesis (ETH), parameterized reduction, computational complexity, parameterized algorithms, and graph algorithms. This foundational section prepares readers for more in-depth analyses in later sections.

Section 3, Fixed-Parameter Tractability (FPT), delves into FPT, emphasizing its critical role in addressing challenging computational problems. It discusses methodologies such as kernelization and search trees essential for developing efficient algorithms within the FPT framework. Recent advancements in parameterized complexity, including new structural parameters that improve FPT approximation schemes for specific graph problems, are highlighted, broadening FPT's applicability and efficacy in practical scenarios [9, 10, 11, 12].

Section 4, Kernelization, examines kernelization as a preprocessing technique that effectively reduces the size of computational problems while maintaining solvability. It introduces the concept of lossy kernelization, which combines traditional kernelization with approximation algorithms and heuristics, transforming an instance (I, k) into a smaller instance (I, k) that retains essential properties

for approximate solutions. The identification of c-essential vertices in vertex-subset problems is also discussed, focusing search space reduction for fixed-parameter tractable algorithms on critical components of optimal solutions [13, 14].

Section 5, Treewidth and Graph Algorithms, focuses on treewidth's significance in graph algorithms and its role in designing efficient algorithms for graph problems. Practical applications of treewidth are showcased through various algorithms, including edge-cut width as an alternative to treewidth for NP-hard graph problems, demonstrating improved algorithmic properties. Advancements in parameterized algorithms for treewidth have led to more efficient computation methods, while preprocessing techniques significantly reduce graph instance sizes, enhancing the feasibility of treewidth calculations in practice [15, 16, 17, 18].

Section 6, W-Hierarchy, introduces the W-hierarchy and its role in classifying parameterized problems by complexity. This section explores the hierarchy of problems in parameterized algorithmics, examining levels such as kernelization and Turing kernelization, with examples like the $(_{A,B}$)-Recognition problem and solution discovery variants such as Vertex Cover and Independent Set. The complexity and kernelization properties of these problems, particularly under specific assumptions in complexity theory, illustrate the nuanced landscape of parameterized complexity [19, 20, 21, 22].

Section 7, Exponential Time Hypothesis (ETH), provides an in-depth analysis of ETH, exploring its significance in computational complexity and the limitations it imposes on algorithm efficiency for various NP-hard problems. This section highlights recent advancements in parameterized algorithms that challenge the traditional (2^n) enumeration barrier, offering insights into ETH's influence on developing faster algorithms for determining graph parameters such as irredundance numbers and complexities associated with solution discovery in graph theory [19, 23].

Section 8, Parameterized Reduction, offers a comprehensive overview of parameterized reduction techniques, detailing their role in transforming one parameterized problem into another. This section emphasizes the significance of parameterized reduction within kernelization, particularly for problems like Vertex Cover, discussing its implications for optimizing algorithm performance and search space reduction in fixed-parameter tractable algorithms [13, 24]. Examples of parameterized reductions are provided to illustrate its importance in proving hardness results.

Finally, **Section 9, Conclusion**, summarizes the key points discussed throughout the survey, providing an overview of the current landscape of research in fixed-parameter tractability (FPT) and related fields. It emphasizes significant advancements such as the fixed-parameter tractability of the MaxSat problem parameterized by the matching number of clauses, the development of work-efficient parallel algorithms for FPT problems, and the establishment of fixed-parameter tractability for the Subset Feedback Vertex Set problem. Key open problems and potential future research directions are identified to enhance understanding and application of FPT methodologies [10, 25, 12].The following sections are organized as shown in Figure 1.

2 Preliminaries

2.1 Core Concepts and Their Interconnections

In computational complexity, Fixed-Parameter Tractability (FPT) plays a central role in efficiently addressing NP-hard problems by focusing on specific parameters. This is often complemented by kernelization, a preprocessing method that reduces problem size while preserving solvability, particularly in parameterized counting problems [26]. The synergy between kernelization and approximation is exemplified by their interplay in approximation algorithms [27].

Treewidth, a measure of a graph's resemblance to a tree, is vital for algorithmic efficiency, especially in NP-hard scenarios where bounded treewidth allows polynomial-time solutions as per Courcelle's Theorem [28]. The interaction between treewidth and structural parameters like treedepth supports the development of polynomial kernels for problems such as F-Deletion [29]. Modular decomposition has further led to modular-treewidth, bridging treewidth and clique-width to manage dense graph structures [30].

The W-hierarchy categorizes parameterized problems by complexity, guided by the Exponential Time Hypothesis (ETH), which posits that certain problems cannot be solved faster than exponential time, establishing crucial complexity bounds [31]. ETH informs parameterized reductions, vital for

transforming parameterized problems and aiding in classification and hardness proofs within the W-hierarchy.

Parameterized reduction is crucial in graph algorithm complexities, especially under connectivity constraints. Problems like Odd Cycle Transversal and Multiway Cut require strategic vertex selection for desired graph separations, leveraging treewidth and kernelization to manage computational demands [32]. The development of polynomial-size approximate Turing kernels for problems such as Independent Set and Vertex Cover marks significant progress in approximating challenging problems [33].

The interconnectedness of FPT, kernelization, treewidth, the W-hierarchy, ETH, and parameterized reduction forms a robust framework for tackling computational complexity. This framework enhances understanding of problem structures by analyzing kernelization complexity in solution discovery problems like Vertex Cover and Independent Set, while exploring diverse solutions through advanced parameters like clique-width. It streamlines search spaces for FPT algorithms by identifying essential vertices in optimal solutions, fostering innovative algorithmic solutions across multiple domains, demonstrating the powerful interplay among foundational concepts in theoretical computer science [19, 13, 20].

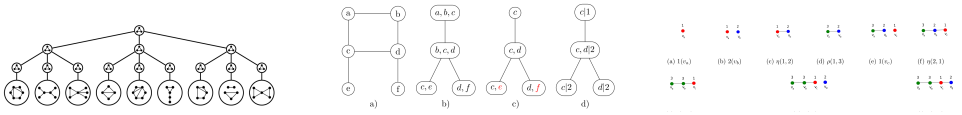
3 Fixed-Parameter Tractability (FPT)

Fixed-Parameter Tractability (FPT) is pivotal in addressing computational complexity by leveraging specific parameters to manage problem tractability. This section explores the mechanisms underpinning FPT algorithms, emphasizing the diverse techniques employed to streamline complexity and enhance problem-solving efficiency. These mechanisms will be further elucidated through various algorithmic approaches and techniques in the subsequent subsection.

3.1 Mechanisms of FPT Algorithms

FPT algorithms effectively tackle computationally challenging problems by utilizing parameters to simplify problem-solving processes. These algorithms employ advanced techniques such as dynamic programming, kernelization, and parameterized reductions. Dynamic programming on graph decompositions, like tree decompositions, exploits structural graph properties for efficient problem resolution, as seen in the Generalized Feedback Vertex Set problem [34]. Kernelization reduces problem instances to smaller, equivalent forms, exemplified by lossy kernelization in the Connected Vertex Cover problem [35]. Parameterized reductions streamline complex instances, enhancing computational efficiency, as demonstrated in polynomial kernelizations for the Subset Sum problem [36].

Advanced techniques, including parameterized streaming algorithms, allow for efficient data processing in a streaming manner, managing space based on additional parameters [3]. Turing kernelization exemplifies oracle-based approaches to enhance FPT algorithm efficiency [37]. Novel parameters such as sm-width facilitate FPT algorithm development for otherwise intractable problems [38]. These strategies underscore the significance of parameterization, structural insights, and innovative techniques in FPT algorithm development, improving our capacity to address complex computational challenges and bridging parameterized complexity with classic computational complexity [39, 19].



(a) Hierarchical structure of interconnected nodes and edges.[11]

(b) Graphs representing different conditional probability distributions.[40]

(c) Graphs with different node labels and edge types.[41]

Figure 2: Examples of Mechanisms of FPT Algorithms

As depicted in Figure 2, FPT algorithms are illustrated through various graphical representations, showcasing their adaptability and efficiency in addressing complex challenges [11, 40, 41].

AI-generated, for reference only.

3.2 Examples of FPT Problems

FPT problems exemplify the transformative potential of parameterization in managing computationally intensive tasks. The Collapsed k-Core problem, parameterized by the number of vertices to be removed, and the s-Club Cluster Edge Deletion problem, parameterized by the sum of s and the treewidth, illustrate effective parameterization in graph theory [42, 43]. The Traveling Salesperson Problem (TSP), when parameterized by budget constraints, demonstrates FPT tractability through polynomial kernelization [44].

In graph modification, the Vertex Planarization problem becomes tractable when parameterized by the number of vertices to be removed [45]. The Firefighting problem, parameterized by the size of a modulator, highlights structural parameterization's role in achieving tractability [4]. FPT algorithms enhance approximation ratios for problems like capacitated vertex cover, especially when parameterized by treewidth [46].

Explicit FPT algorithms, such as those for edge-deletion problems in interval graphs, demonstrate efficiency gains through specialized techniques [47]. Novel parameters, like the parameter in vertex deletion problems, facilitate advanced kernelization methods [48]. These examples highlight parameterization's role in transforming NP-hard problems into manageable forms, revealing the interplay between parameterization and approximation in algorithm development [49, 50, 51].

4 Kernelization

Kernelization is a fundamental technique in computational complexity, crucial for simplifying fixed-parameter tractable (FPT) problems by reducing instances to smaller, equivalent forms. This section explores kernelization's foundational principles, techniques, and its profound impact on computational challenges, followed by an examination of advanced methods that extend these foundations.

4.1 Fundamentals of Kernelization

Kernelization transforms problem instances into smaller, equivalent forms, producing a kernel whose size is bounded by a function of a chosen parameter, ensuring efficient problem resolution while maintaining essential properties [48]. Central to this process are polynomial-time compression algorithms that simplify problems by removing induced structures, thereby minimizing computational overhead [52].

Structural parameters like treewidth and feedback vertex sets guide these reductions, offering insights into problem complexity and enabling effective strategies. For instance, kernelization in flat folding leverages bounded parameters such as ply and treewidth to reduce complexity [7]. The synergy between kernelization and approximation strategies enhances data reduction efficiency, as demonstrated by polynomial-size -approximate kernels for problems like Connected Vertex Cover and Disjoint Cycle Packing, which improve both kernel sizes and approximation algorithm performance [36, 53, 54, 14].

4.2 Advanced Kernelization Techniques

Recent advancements in kernelization exploit structural parameters and innovative algorithms to refine problem preprocessing. Approximating treedepth modulators and applying protrusion replacement rules effectively reduce instances while preserving essential properties, emphasizing the role of structural parameters [55]. Utilizing flowers instead of sunflowers improves kernel sizes while maintaining necessary properties [56]. The Adaptive Optimization Algorithm (AOA) enhances kernelization efficiency by adapting search strategies to changing landscapes [8].

Linear kernels for H-topological-minor-free graphs, based on treewidth-bounding and finite integer index, represent a breakthrough by combining decomposition techniques with graph-specific preprocessing, yielding improved kernel size bounds [57, 54]. Detecting c-essential vertices enhances combinatorial efficiency by focusing on key structural components [13], complemented by FPT algorithms for maximum minimal blocking sets, which improve kernelization efficiency [58].

In planar graphs, reducing input graphs while preserving linkage properties for disjoint paths problems is critical for kernelization [59]. Polynomial-sized kernels for the Traveling Salesperson Problem

(TSP) and its generalizations leverage parameters like vertex cover number, demonstrating parameter-driven approaches' utility in kernelization [44]. The development of kernels cubic in the feedback vertex set size, rather than vertex cover size, marks a significant innovation in kernelization [60].

These advanced techniques highlight kernelization's dynamic nature, showcasing interactions between theoretical advancements—such as polynomial size -approximate kernels—and practical applications, including solution discovery problems and meta-theorems for bounded genus graphs [19, 61, 14]. By leveraging structural parameters, approximation strategies, and innovative algorithms, these advancements enhance preprocessing efficiency and broaden the scope of problems effectively addressed through kernelization.

4.3 Challenges and Future Directions in Kernelization Research

Kernelization research faces challenges, notably in developing efficient polynomial kernels for diverse parameterized problems. A significant limitation is the dependency on specific structural characteristics of input graphs, which can restrict kernelization techniques' generalizability across various graph classes. Methods relying on planarity or specific minor-free properties may not extend well to more complex structures, limiting applicability [62]. Additionally, the effectiveness of these methods can vary based on structural nuances, with not all instances benefiting equally from proposed reductions [44].

The reliance on randomized techniques in some kernelization methods introduces variability in performance and correctness, especially in edge cases, posing challenges for consistent application [48]. The pursuit of deterministic approaches that achieve similar efficiency without the unpredictability of randomized methods remains a critical area for future exploration.

The complexity of dynamic programming in certain parameterizations also presents implementation challenges, particularly for large graphs where computational resources may be strained [63]. Moreover, identifying lower bounds for Turing kernels in specific FPT problems highlights the need for refined frameworks that provide insights into kernelization limits [39].

Future research promises to address these challenges through various avenues. One potential direction involves exploring constant-factor approximations in c-essential detection to enhance search-space reduction and improve kernelization efficiency [13]. Further refinements of kernelization techniques and their applicability to related graph theory problems could expand the utility of these methods [59].

Developing frameworks that identify polynomial compressions and limitations for counting problems represents a significant contribution to parameterized complexity, offering potential pathways for future exploration [64]. Investigating the approximability of edge modification problems within a comprehensive theoretical framework could enrich the understanding of parameterized complexity and inform new kernelization strategies [27].

By tackling these challenges and exploring proposed future directions, researchers can significantly advance the field of kernelization, focusing on efficient preprocessing techniques for NP-hard problems. This progress will profoundly impact the broader area of parameterized complexity, potentially leading to groundbreaking methodologies for solving computationally intensive issues. Recent studies have investigated kernelization complexities for various solution discovery problems, such as Vertex Cover and Independent Set, while new frameworks for lossy kernelization have emerged that integrate approximation algorithms. Additionally, the exploration of strict polynomial kernelization has revealed limitations in certain parameterized problems, underscoring the intricate relationship between kernelization techniques and computational problem complexity [19, 50, 14].

In recent years, the concept of treewidth has garnered significant attention due to its pivotal role in enhancing the efficiency of graph algorithms. As illustrated in Figure 3, the hierarchical structure of treewidth's influence can be categorized into three primary areas: understanding treewidth, its application in algorithm design, and the associated challenges and limitations. This figure not only delineates the definition and importance of treewidth but also underscores its utility across various algorithmic strategies and preprocessing techniques. Furthermore, it highlights the challenges encountered in approximation methods and unbounded scenarios, thereby providing a comprehensive overview of the current landscape in treewidth research. By integrating these insights, we can

better appreciate the multifaceted nature of treewidth and its implications for future algorithmic developments.

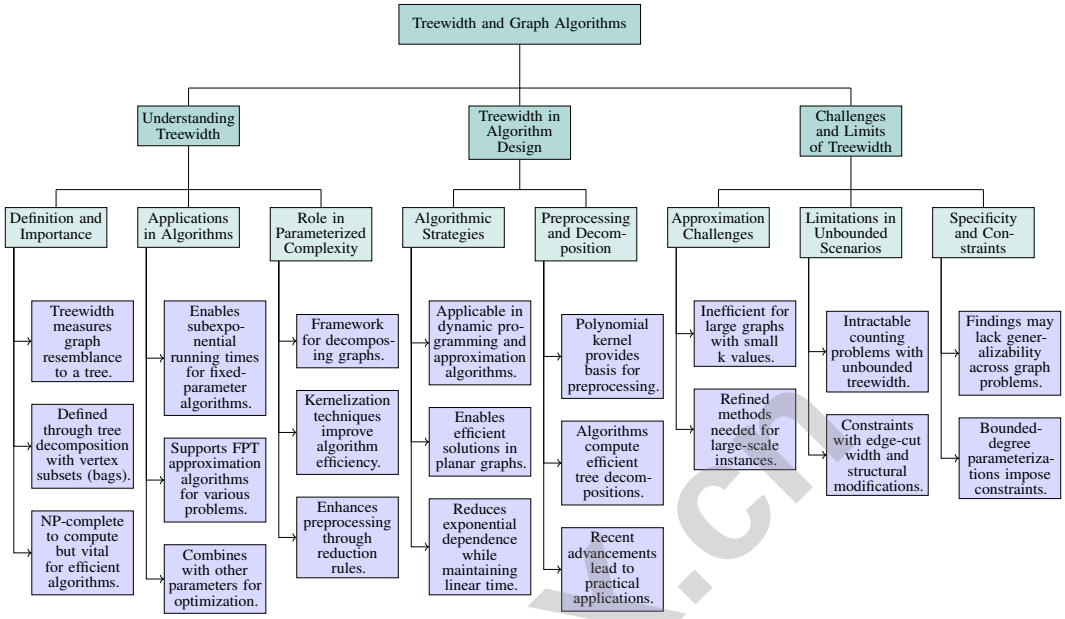


Figure 3: This figure illustrates the hierarchical structure of treewidth's role in graph algorithms, categorized into understanding treewidth, its application in algorithm design, and challenges and limitations. It highlights the definition and importance of treewidth, its utility in various algorithmic strategies, preprocessing techniques, and the challenges faced in approximation and unbounded scenarios.

5 Treewidth and Graph Algorithms

5.1 Understanding Treewidth

Treewidth is a pivotal concept in graph theory, measuring how closely a graph resembles a tree. It is defined through a tree decomposition, where each tree node corresponds to a vertex subset (or "bag") of the graph, and the width is the largest bag's size minus one. Computing treewidth is NP-complete, reflecting its complexity, yet it is vital for designing efficient algorithms for NP-hard problems, as graphs with bounded treewidth permit polynomial-time solutions [18].

In fixed-parameter algorithms, treewidth enables subexponential running times for complex problems. For instance, the k-arc Chinese Postman Problem (CPP) employs tree decompositions to develop fixed-parameter algorithms, illustrating treewidth's practical utility [65]. It also aids in efficiently computing independent and blocking sets by leveraging graph structural properties [58].

Treewidth is crucial in approximating graph parameters, contributing to FPT approximation algorithms for problems like capacitated vertex cover and vector dominating set [46]. Understanding the relationships between treewidth and other parameters, such as pathwidth and clique-width, enhances problem-solving approaches. In practice, treewidth often combines with other parameters to optimize algorithmic efficiency, as seen in s-club cluster problems, where it improves the resolution of larger instances through effective preprocessing [66, 16, 17, 18]. The construction of tree decompositions via reduction rules enhances preprocessing by identifying smaller equivalent instances efficiently.

Treewidth is indispensable in parameterized complexity, offering a framework for decomposing graphs and informing innovative algorithm design. Kernelization techniques, such as polynomial size -approximate kernels and search-space reduction strategies, improve preprocessing algorithms that reduce instance sizes, enhancing the efficiency of fixed-parameter tractable algorithms for NP-hard problems like Connected Vertex Cover and Subset Sum [14, 36, 13, 39, 67].

AI-generated, for reference only.

5.2 Treewidth in Algorithm Design

Treewidth is a critical parameter for designing efficient algorithms, particularly for computationally intensive problems. It is applicable across various strategies, including dynamic programming and approximation algorithms. In planar graphs, decomposition into bounded treewidth components enables efficient dynamic programming solutions, as demonstrated by subexponential parameterized algorithms [18].

Parameterized approximation algorithms further illustrate treewidth's utility. The Improved Treewidth Approximation Algorithm (ITAA) reduces reliance on parameter k and streamlines branching, showcasing treewidth's adaptability in approximation contexts [68]. This adaptability extends to algorithms minimizing exponential dependence on treewidth while maintaining linear time complexity concerning the number of vertices.

Preprocessing techniques are essential for leveraging treewidth in algorithm design. Treewidth's polynomial kernel, parameterized by feedback vertex set or vertex cover size, provides a theoretical basis for effective preprocessing strategies [18]. This simplifies problem space, facilitating more efficient algorithm design.

Algorithms computing tree decompositions are central to utilizing treewidth. An algorithm that computes tree decomposition of width $O(k)$ in $O(nk^2 \log k)$ time for n-vertex planar graphs with treewidth k exemplifies achievable efficiency gains. Integrating treewidth into algorithmic stages—tree decomposition enumeration, near-optimal decomposition construction, and simplifications for linear-time solutions—underscores its significance. Solving numerous NP-hard problems for graphs with small treewidth relies on obtaining reasonable tree decompositions, as optimal decompositions and exact treewidth calculations are NP-hard. Recent advancements include linear FPT algorithms mitigating exponential dependence on treewidth, leading to practical applications across combinatorial problems and addressing long-standing open questions [69, 16, 17, 15, 18].

5.3 Challenges and Limits of Treewidth

Applying treewidth in algorithm design presents challenges and limitations, especially for complex graph problems. A primary challenge is approximating treewidth, particularly in large graphs where existing algorithms may be inefficient for small k values [68]. This inefficiency necessitates refined approximation methods for large-scale instances without sacrificing performance.

A significant limitation arises with unbounded treewidth, where counting problems become intractable, complicating efficient algorithm or approximation method development [70]. This intractability limits treewidth-based approaches, especially for problems beyond bounded treewidth scenarios.

Exploring structural parameters like edge-cut width reveals further challenges. Edge-cut width's lack of closure under edge or vertex deletion restricts its effectiveness in scenarios requiring structural modifications [17]. This necessitates careful consideration of graph structural properties when employing treewidth or related parameters in algorithm design.

The specificity of findings, such as those related to Grundy Coloring, highlights another challenge: potential lack of generalizability across different graph problems and parameters [71]. While treewidth is powerful for specific applications, its seamless extension across graph theory is not guaranteed, necessitating further research into complementary parameters and techniques.

Constraints imposed by bounded-degree parameterizations indicate certain problems cannot be solved faster than existing dynamic programming algorithms under these constraints [72]. This emphasizes the need for innovative approaches to transcend these limitations and enhance treewidth-based method efficiency.

6 W-Hierarchy

The W-hierarchy is a pivotal framework in parameterized complexity, categorizing computational problems by their difficulty levels. This section delves into the W-hierarchy's structure and significance, offering insights into complexities within parameterized contexts and their implications for algorithmic strategies.

6.1 Introduction to the W-Hierarchy

The W-hierarchy classifies computational problems based on difficulty, considering input size and secondary measures like solution size or structural properties. This framework elucidates NP-hard problems' tractability, guiding efficient algorithm development tailored to specific structures such as treewidth, clique-width, and rank-width [73, 1, 74, 21]. Each hierarchy level represents a class of problems with similar complexity characteristics.

Advancements like the 'union operation' connect problems complete for classical complexity classes with those complete for W-classes, enhancing understanding of complexity class relationships [75]. Kernelization impacts W-hierarchy classification significantly; the presence or absence of polynomial kernels can collapse certain levels, illustrating kernelization's interplay with problem complexity [76]. A new kernelizability hierarchy distinguishes between problems with polynomial Turing kernels and those WK[1]-hard, refining the complexity landscape [31].

The W-hierarchy's relevance extends to graph theory width measures, such as treewidth, especially in random graphs where these measures critically influence graph problem complexity [77]. It also aids in understanding algorithmic limitations under the Exponential Time Hypothesis (ETH), providing a framework for assessing algorithmic efficiency limits [7].

6.2 Levels and Structure of the W-Hierarchy

The W-hierarchy stratifies parameterized complexity, categorizing problems by computational difficulty into levels like W[1], W[2], W[3], etc. W[1] is extensively studied, linked to parameterized complexity and kernelization techniques essential for solving various problems [78, 79, 21, 80]. Each level includes complete problems serving as benchmarks for complexity assessment.

W[1] is foundational, often linked to problems considered fixed-parameter tractable (FPT) under specific conditions. The k-Clique problem, determining a complete subgraph of size k, serves as a benchmark for graph-related problem complexity [81, 20, 22]. W[1] delineates the boundary between tractable and intractable parameterized problems, providing a critical reference for complexity analyses.

Ascending the W-hierarchy to W[2], W[3], and beyond, problem complexity escalates, requiring sophisticated kernelization techniques for effective classification. Some problems allow polynomial-size -approximate kernels, while others, like Longest Path and Set Cover, lack such kernels under complexity assumptions. This complexity increase necessitates innovative preprocessing algorithms to manage intricate parameterized problems effectively [50, 14, 19, 1, 21]. The Dominating Set problem exemplifies complexity challenges at higher W-hierarchy levels.

The W-hierarchy's structure is enriched by connections to other graph theory areas and parameterized complexity. For example, kernelization techniques for Treewidth-2 Vertex Deletion build on foundational hierarchy work, providing explicit bounds and enhancing problem complexity understanding [82]. These advancements illustrate how W-hierarchy insights inform efficient algorithm and preprocessing technique development.

The W-hierarchy offers a nuanced framework for understanding parameterized problem complexities. Its levels provide a lens for examining computational difficulties, while its structure delineates pathways for investigating tractability limits, emphasizing transitions between structural thresholds and algorithm efficiency tailored to specific input configurations [73, 10, 80, 83].

6.3 W-Hardness and Its Implications

W[1]-hardness is crucial within the W-hierarchy, indicating problems conjectured to be intractable when parameterized by specific parameters. A problem is W[1]-hard if it is as difficult as the most challenging W[1] problems, suggesting it is unlikely to be fixed-parameter tractable (FPT) unless the Exponential Time Hypothesis (ETH) fails [1]. This classification demarcates parameterized complexity, indicating problems where efficient parameterized algorithms are not expected.

W[1]-hardness has significant implications for W-hierarchy problem classification. The k-Biclique problem is recognized as W[1]-hard, illustrating algorithm development challenges [84]. The EQUITABLE COLORING problem's complexity is underscored by its W[1]-hardness across graph classes, established through reductions from BIN-PACKING [85].

Addressing W[1]-hard problems is challenging, exemplified by the Firefighting problem showing W[1]-hardness for specific graph classes, highlighting computational difficulties [4]. Structural parameterizations in bounded-degree graphs provide reductions illustrating the impossibility of surpassing existing algorithms under the Strong Exponential Time Hypothesis (SETH) and ETH, reinforcing W[1]-hard problems' inherent complexity [72].

Establishing W[1]-hardness involves reductions from well-known NP-hard problems, clarifying computational tractability limits. This process identifies problems unlikely solvable in polynomial time and aids in determining which complex problems can be effectively addressed using parameterized algorithms. Research examines parameterization and approximation interplay, revealing new techniques and exploration avenues, enriching hardness understanding in parameterized complexity [49, 74, 50]. These reductions are vital for understanding algorithmic efficiency limits and guiding innovative algorithmic strategy development.

W[1]-hardness plays a pivotal role in classifying parameterized problems, delineating tractability limits and influencing theoretical research and practical applications in computational complexity. This research resonates across theoretical computer science fields, particularly in parameterized complexity and efficient algorithm development for complex computational challenges. Recent advancements in diverse solution finding under graph parameters like cliquewidth demonstrate the potential for generating multiple diverse solutions with minimal additional computational cost, enriching the complexity landscape and enhancing algorithmic paradigms for parameterized enumeration and kernelization [50, 14, 80, 20, 51].

7 Exponential Time Hypothesis (ETH)

7.1 Introduction to ETH and Its Role in Complexity Theory

The Exponential Time Hypothesis (ETH) is a cornerstone in computational complexity, positing that specific NP-complete problems cannot be solved in sub-exponential $2^{o(n)}$ time for input size n. This conjecture sets fundamental complexity bounds, influencing the evaluation of computational problem difficulty. ETH suggests that problems like treewidth computation or kernelization for certain parameterized problems are inefficient to solve unless NP problems become polynomially solvable, affecting the development of approximation algorithms and understanding polynomial kernel limitations in graph problems [86, 67, 14]. As a benchmark, ETH aids in deriving asymptotically optimal algorithms, such as finding maximum minimal separators in $2^{O(k)}n^{O(1)}$ time, aligning with ETH's constraints [79]. It also informs connectivity problem exploration within parameterized complexity, impacting the Strong Exponential Time Hypothesis (SETH) and extending our comprehension of computational limits [87].

7.2 ETH and Lower Bounds in Parameterized Complexity

ETH is crucial in establishing lower bounds for parameterized problems, providing a framework to understand the computational limits of algorithmic solutions. By asserting that certain NP-complete problems defy sub-exponential solutions, ETH sets a standard for evaluating fixed-parameter tractable (FPT) algorithms' efficiency, particularly in graph problems defined by structural properties like clique-width. Recent advances in parameterized complexity reveal how these parameters facilitate efficient algorithm design, clarifying FPT approaches' potential and constraints [9, 10, 11, 41]. ETH impacts the development of strict polynomial kernels in diminishable parameterized problems, where their existence hinges on ETH's validity [50]. It also informs lower bounds for specific problems like Vertex Planarization, establishing that no algorithm can solve it faster than $2^{o(w \log w)} \cdot n^{O(1)}$ time unless ETH fails, highlighting complexity challenges [45]. ETH's role in understanding the minimum degree barrier for kernelization, particularly in Vertex Cover, underscores its significance in kernelization research [24]. The defensive alliance problem illustrates ETH's impact, being W[1]-hard when parameterized by treewidth, excluding FPT algorithms under the assumption that W[1] differs from FPT [88]. Additionally, ETH provides insights into feedback vertex set problems, establishing that they cannot be solved in $2^{o(tw \log tw)} \cdot n^{O(1)}$ time unless ETH fails, reinforcing its role in setting theoretical limits for parameterized algorithms [28].

7.3 Challenges and Open Questions in ETH Research

ETH remains a pivotal yet unresolved element in computational complexity, particularly regarding algorithmic efficiency limits. It poses challenges in determining precise time complexities for problems like treewidth and irredundance numbers in graphs, with studies showing exponential time is necessary for exact solutions, such as approximating treewidth within a 1.00005 factor being NP-hard, requiring $2^{\Omega(n)}$ time unless ETH fails. Advances reveal some problems can be solved faster than the $O^*(2^n)$ barrier, enhancing understanding of ETH's algorithmic landscape [23, 86]. A major challenge is verifying ETH, as it lacks definitive proof or refutation, leaving questions about computational tractability boundaries and potential breakthroughs. Key inquiries involve ETH's implications for parameterized complexity, aiming to establish tighter lower bounds for resistant problems, such as feedback vertex set complexities [28]. Investigating ETH's relationship with complexity hypotheses like SETH raises questions about unified theories offering deeper insights into computational hardness [87]. Developing new techniques for ETH-based lower bounds is a research frontier, requiring innovative approaches beyond reductions from known hard problems, potentially involving width measures like treewidth [79]. Applying ETH to emerging areas like quantum computing and machine learning presents challenges and opportunities, with unresolved implications for parameterized complexity and kernelization, affecting problems like Vertex Cover and Independent Set. Understanding this relationship could lead to insights into polynomial-time kernelization feasibility and algorithmic solution efficiency in complex domains [24, 14, 89, 19, 1].

8 Parameterized Reduction

8.1 Concept and Importance of Parameterized Reduction

Parameterized reduction is a fundamental technique in parameterized complexity, enabling the transformation of one problem into another while maintaining the parameterization. This process is crucial for kernelization, which produces polynomial-size -approximate kernels that streamline the preprocessing of NP-hard problems by generating reduced instances that preserve essential problem properties. These kernels facilitate the search for approximate solutions, thereby enhancing algorithmic efficiency [51, 24, 50, 14]. Parameterized reduction aids in classifying problems by complexity, distinguishing fixed-parameter tractable (FPT) problems from those that are W[1]-hard or beyond.

The systematic transformation of problems through parameterized reduction allows researchers to explore inter-problem relationships and derive hardness results, which are vital for understanding computational limits. For instance, the subset feedback vertex set problem, which seeks a set of vertices intersecting all simple cycles through a specified subset in a graph, illustrates how parameterized reduction can simplify problem instances while preserving computational challenges [25].

Moreover, parameterized reduction is integral to developing parameterized enumeration algorithms (PEA), which leverage complexity classes to systematically enumerate solutions based on parameters [80]. This enables efficient exploration of solution spaces and provides insights into the structure and complexity of parameterized problems. By utilizing parameterized reduction, PEA can effectively enumerate solutions even in challenging computational scenarios.

8.2 Techniques and Methods in Parameterized Reduction

In parameterized complexity theory, parameterized reduction is pivotal for transforming problems while preserving core computational properties, often through kernelization. This aims to reduce the problem size by identifying critical components, such as c-essential vertices, crucial for constructing optimal solutions. These techniques facilitate the development of fixed-parameter tractable algorithms, enhancing the efficiency of solving combinatorial problems like Odd Cycle Transversal and Directed Feedback Vertex Set. Additionally, parameterized reduction supports efficient enumeration algorithms and reoptimization strategies, improving the classification and analysis of problem complexities relative to their parameters [13, 80, 51]. Such reductions are essential for establishing the parameterized complexity of problems, particularly in proving hardness results and exploring the boundaries of fixed-parameter tractability (FPT).

A key technique in parameterized reduction is the creation of problem kernels, which aim to reduce the problem size to a kernel bounded by a function of the parameter. This is exemplified in the Connected Vertex Cover problem, where polynomial-time compression techniques yield reduced instances that retain core computational challenges [35]. Kernelization simplifies problem instances, making them more amenable to efficient algorithmic solutions.

Turing reductions decompose a problem into smaller subproblems, solvable independently or with oracle queries, effectively distributing the complexity across subproblems parameterized by specific structural properties, especially in graph problems [37]. This enables a detailed exploration of problem complexity, fostering innovative algorithmic strategies.

Lossy kernelization expands the parameterized reduction toolkit by allowing approximate solutions that trade exactness for efficiency. This is particularly relevant when exact solutions are computationally prohibitive, providing sufficient utility through approximate solutions [35]. By relaxing exactness constraints, lossy kernelization broadens the applicability of parameterized reduction.

Integrating structural parameters, such as treewidth and feedback vertex sets, into parameterized reduction techniques enhances complexity management. These parameters offer insights into graph structures, enabling more effective reductions and kernelization strategies. The study of flat folding, where complexity is reduced by focusing on bounded parameters like ply and treewidth, exemplifies this approach [7].

9 Conclusion

9.1 Challenges and Future Directions

Fixed-parameter tractability (FPT) continues to present both significant challenges and opportunities for advancement in the realm of computational complexity. One of the primary hurdles is developing parameterized algorithms for W[1]-hard problems, which necessitates innovative approaches, including new parameterizations and refined reduction techniques. Extending these methodologies to a broader spectrum of graph classes and optimizing kernel sizes for specific problems remains a critical research focus. Enhancing algorithmic efficiency through stronger connectivity constraints and adapting techniques for related challenges, such as Connected -Treewidth Deletion, are essential directions for further investigation.

Exploring the parameterized complexity of problems like the Collapsed k-Core and s-Club Cluster Edge Deletion offers additional potential, particularly when considering larger values of k and the development of subexponential-time algorithms. Recent progress in computing maximum minimal blocking sets highlights the importance of FPT in managing larger graph instances efficiently. Future research could refine these algorithms and assess their applicability to related graph theory challenges. Moreover, the exploration of polynomial kernels concerning other parameters and their utility across diverse graph classes is a crucial area for ongoing study.

While current research often yields theoretical insights, such as those pertaining to Vertex Planarization, translating these into practical algorithms remains a priority. Developing practical solutions for graph modification challenges and optimizing kernelization processes for broader applicability could greatly enhance the field. Investigating deterministic versions of randomized kernelization methods and their relevance to other NP-hard problems in parameterized complexity could also lead to substantial progress.

Further research directions include exploring the fixed-parameter tractability of additional list problems associated with various parameters, such as feedback vertex set size or vertex cover number. Additionally, examining the structural properties of more complex graph classes to improve the applicability of polynomial kernels in edge deletion problems presents a promising avenue for exploration. The potential applications of c-closure in various graph problems, alongside enhancements to kernelization techniques to accommodate a wider range of graphs, represent exciting prospects for future inquiry.

Addressing these challenges and pursuing these research directions will enable significant advancements in the field, deepening our understanding of computational complexity and expanding the applicability of fixed-parameter tractability.

References

[1] Akanksha Agrawal. Graph modification problems: Beyond the known boundaries. 2017.

[2] Leon Kellerhals, Tomohiro Koana, and Pascal Kunz. Vertex cover and feedback vertex set above and below structural guarantees, 2022.

[3] N. R. Aravind and Roopam Saxena. Parameterized complexity of path set packing, 2024.

[4] Bireswar Das, Murali Krishna Enduri, Neeldhara Misra, and I. Vinod Reddy. On structural parameterizations of firefighting, 2017.

[5] Kitty Meeks and Alexander Scott. The parameterised complexity of list problems on graphs of bounded treewidth, 2016.

[6] Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth, 2014.

[7] David Eppstein. A parameterized algorithm for flat folding, 2023.

[8] Stefan Kratsch and Magnus Wahlstrom. Preprocessing of min ones problems: A dichotomy, 2009.

[9] Tanmay Inamdar, Lawqueen Kanesh, Madhumita Kundu, M. S. Ramanujan, and Saket Saurabh. Fpt approximations for packing and covering problems parameterized by elimination distance and even less, 2023.

[10] Max Bannach, Malte Skambath, and Till Tantau. Towards work-efficient parallel parameterized algorithms, 2019.

[11] Michael R. Fellows and Bart M. P. Jansen. Fpt is characterized by useful obstruction sets, 2013.

[12] R. Crowston, G. Gutin, M. Jones, V. Raman, S. Saurabh, and A. Yeo. Fixed-parameter tractability of satisfying beyond the number of variables, 2012.

[13] Benjamin Merlin Bumpus, Bart M. P. Jansen, and Jari J. H. de Kroon. Search-space reduction via essential vertices, 2022.

[14] Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization, 2016.

[15] Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width, 2022.

[16] Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth, 2023.

[17] Cornelius Brand, Esra Ceylan, Christian Hatschka, Robert Ganian, and Viktoriia Korchemna. Edge-cut width: An algorithmically driven analogue of treewidth based on edge cuts, 2022.

[18] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization, 2013.

[19] Mario Grobler, Stephanie Maaz, Amer E. Mouawad, Naomi Nishimura, Vijayaragunathan Ramamoorthi, and Sebastian Siebertz. Kernelization complexity of solution discovery problems, 2024.

[20] Karolina Drabik and Tomáš Masařík. Finding diverse solutions parameterized by cliquewidth, 2024.

[21] Jouke Witteveen, Ralph Bottesch, and Leen Torenvliet. A hierarchy of polynomial kernels, 2019.

[22] Iyad Kanj, Christian Komusiewicz, Manuel Sorge, and Erik Jan van Leeuwen. Solving partition problems almost always requires pushing many vertices around, 2019.

[23] Ljiljana Brankovic, Henning Fernau, Joachim Kneis, and Dieter Kratsch Alexander Langer Mathieu Liedloff Daniel Raible Peter Rossmanith. Breaking the 2^n – $barrier for irredundance : A parameterized route to solving exact puzzles$, 2009.

[24] Michael R. Fellows, Lars Jaffke, Aliz Izabella Király, Frances A. Rosamond, and Mathias Weller. What is known about vertex cover kernelization?, 2019.

[25] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed parameter tractable, 2011.

[26] Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Kernelization of counting problems, 2023.

[27] Ivan Bliznets, Marek Cygan, Pawel Komosa, and Michal Pilipczuk. Hardness of approximation for h-free edge modification problems, 2018.

[28] Benjamin Bergougnoux, Édouard Bonnet, Nick Brettell, and O joung Kwon. Close relatives of feedback vertex set without single-exponential algorithms parameterized by treewidth, 2020.

[29] Bart M. P. Jansen and Astrid Pieterse. Polynomial kernels for hitting forbidden minors under structural parameterizations, 2018.

[30] Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by modular-treewidth, 2023.

[31] Danny Hermelin, Stefan Kratsch, Karolina Sołtys, Magnus Wahlström, and Xi Wu. Hierarchies of inefficient kernelizability, 2011.

[32] Isja Mannens and Jesper Nederlof. A fine-grained classification of the complexity of evaluating the tutte polynomial on integer points parameterized by treewidth and cutwidth, 2023.

[33] Eva-Maria C. Hols, Stefan Kratsch, and Astrid Pieterse. Approximate turing kernelization for problems parameterized by treewidth, 2020.

[34] Michael Lampis, Nikolaos Melissinos, and Manolis Vasilakis. Parameterized max min feedback vertex set, 2023.

[35] R. Krithika, Diptapriyo Majumdar, and Venkatesh Raman. Revisiting connected vertex cover: Fpt algorithms and lossy kernels, 2017.

[36] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems, 2015.

[37] Bart M. P. Jansen and Jari J. H. de Kroon. Fpt algorithms to compute the elimination distance to bipartite graphs and more, 2021.

[38] Sigve Hortemo Sæther and Jan Arne Telle. Between treewidth and clique-width, 2014.

[39] Weidong Luo. Frameworks for solving turing kernel lower bound problem and finding natural candidate problems in np-intermediate, 2016.

[40] Ernst Althaus and Sarah Ziegler. Optimal tree decompositions revisited: A simpler linear-time fpt algorithm, 2020.

[41] David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial fpt algorithms for some classes of bounded clique-width graphs, 2017.

[42] Junjie Luo, Hendrik Molter, and Ondrej Suchy. A parameterized complexity view on collapsing k-cores, 2018.

[43] Fabrizio Montecchiani, Giacomo Ortali, Tommaso Piselli, and Alessandra Tappini. On the parameterized complexity of the s-club cluster edge deletion problem, 2022.

[44] Václav Blažej, Pratibha Choudhary, Dušan Knop, Šimon Schierreich, Ondřej Suchý, and Tomáš Valla. On polynomial kernels for traveling salesperson problem and its generalizations, 2022.

[45] Marcin Pilipczuk. A tight lower bound for vertex planarization on graphs of bounded treewidth, 2015.

[46] Huairui Chu and Bingkai Lin. Fpt approximation using treewidth: Capacitated vertex cover, target set selection and vector dominating set, 2024.

[47] Toshiki Saitoh, Ryo Yoshinaka, and Hans L. Bodlaender. Fixed-treewidth-efficient algorithms for edge-deletion to intersection graph classes, 2021.

[48] Stefan Kratsch. A randomized polynomial kernelization for vertex cover with a smaller parameter, 2016.

[49] Andreas Emil Feldmann, Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020.

[50] Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier. Diminishable parameterized problems and strict polynomial kernelization, 2017.

[51] Hans-Joachim Böckenhauer, Elisabet Burjons, Martin Raszyk, and Peter Rossmanith. Reoptimization of parameterized problems, 2019.

[52] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, Erik Jan van Leeuwen, and Marcin Wrochna. Polynomial kernelization for removing induced claws and diamonds, 2015.

[53] Holger Dell and Dániel Marx. Kernelization of packing problems, 2018.

[54] An improved time-efficient approximate kernelization for connected treedepth deletion set.

[55] Jakub Gajarský, Petr Hliněný, Jan Obdržálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes, 2015.

[56] Stefan Fafianie and Stefan Kratsch. A shortcut to (sun)flowers: Kernels in logarithmic space or linear time, 2015.

[57] Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Linear kernels on graphs excluding topological minors, 2012.

[58] Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. Parameterized complexity of computing maximum minimal blocking and hitting sets, 2021.

[59] Michał Włodarczyk and Meirav Zehavi. Planar disjoint paths, treewidth, and kernels, 2023.

[60] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter, 2013.

[61] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization, 2013.

[62] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization, 2010.

[63] Dipayan Chakraborty, Florent Foucaud, Diptapriyo Majumdar, and Prafullkumar Tale. Structural parameterization of locating-dominating set and test cover, 2024.

[64] Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting $\mathbf{c}$-closure in kernelization algorithms for graph problems, 2022.

[65] Gregory Gutin, Mark Jones, and Bin Sheng. Parameterized complexity of the k-arc chinese postman problem, 2016.

[66] Mathieu Chapelle, Mathieu Liedloff, Ioan Todinca, and Yngve Villanger. Treewidth and pathwidth parameterized by vertex cover, 2013.

[67] Guilherme C. M. Gomes and Ignasi Sau. Finding cuts of bounded degree: complexity, fpt and exact algorithms, and kernelization, 2019.

[68] Mahdi Belbasi and Martin Fürer. An improvement of reed's treewidth approximation, 2022.

[69] Mahdi Belbasi, Martin Fürer, and Medha Kumar. Optimized 2-approximation of treewidth, 2024.

[70] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems, 2015.

[71] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth, 2022.

[72] Michael Lampis and Manolis Vasilakis. Structural parameterizations for two bounded degree problems revisited, 2024.

[73] Falko Hegerfeld and Stefan Kratsch. Towards exact structural thresholds for parameterized complexity, 2022.

[74] Tight lower bounds for problems parameterized by rank-width.

[75] Christoph Stockhusen and Till Tantau. Completeness results for parameterized space classes, 2013.

[76] Liang Ding, Abdul Samad, Xingran Xue, Xiuzhen Huang, and Liming Cai. Polynomial kernels collapse the w-hierarchy, 2013.

[77] Jakub Marecek. Some probabilistic results on width measures of graphs, 2009.

[78] Emmanuel Sam, Benjamin Bergougnoux, Petr A. Golovach, and Nello Blaser. Kernelization for finding lineal topologies (depth-first spanning trees) with many or few leaves, 2023.

[79] Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, and Tsuyoshi Yagita. Finding a maximum minimal separator: Graph classes and fixed-parameter tractability, 2020.

[80] Nadia Creignou, Arne Meier, Julian-Steffen Müller, Johannes Schmidt, and Heribert Vollmer. Paradigms for parameterized enumeration, 2013.

[81] Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Clique cover and graph separation: New incompressibility results, 2011.

[82] Jeroen L. G. Schols. Kernelization for treewidth-2 vertex deletion, 2022.

[83] Tomáš Gavenčiak, Dušan Knop, and Martin Koutecký. Integer programming in parameterized complexity: Three miniatures, 2018.

[84] Bingkai Lin. The parameterized complexity of k-biclique, 2019.

[85] Guilherme de C. M. Gomes, Carlos V. G. C. Lima, and Vinícius F. dos Santos. Parameterized complexity of equitable coloring, 2019.

[86] Édouard Bonnet. Treewidth inapproximability and tight eth lower bound, 2024.

[87] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time, 2011.

[88] Bernhard Bliem and Stefan Woltran. Defensive alliances in graphs of bounded treewidth, 2017.

[89] Pål Grønås Drange and Michał Pilipczuk. A polynomial kernel for trivially perfect editing, 2014.

Disclaimer:

SurveyX is an AI-powered system designed to automate the generation of surveys. While it aims to produce high-quality, coherent, and comprehensive surveys with accurate citations, the final output is derived from the AI's synthesis of pre-processed materials, which may contain limitations or inaccuracies. As such, the generated content should not be used for academic publication or formal submissions and must be independently reviewed and verified. The developers of SurveyX do not assume responsibility for any errors or consequences arising from the use of the generated surveys.