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 that enables efficient algorithm development for NP-hard problems by focusing on specific parameters. This approach allows for the creation of algorithms that maintain efficiency for fixed parameters, providing solutions to otherwise intractable problems [1]. FPT is particularly significant in dynamic graph algorithms, facilitating efficient resolutions for issues like Vertex Cover and Cluster Vertex Deletion, which are essential for network optimization [2].

FPT's utility extends to parameterized complexity, aiding in solving complex optimization challenges such as the Path Set Packing problem, thereby addressing computationally intensive tasks [3]. Furthermore, FPT techniques have enhanced our understanding of problem structural properties, as evidenced by studies on the Firefighting problem, which involves strategically defending vertices against fire spread on graphs [4].

In graph theory, FPT intersects with problems parameterized by treewidth, including 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 led to tractable solutions when parameterized by treewidth [6]. Additionally, studies on flat foldability through parameters like ply and treewidth showcase FPT's adaptability across various problem contexts [7].

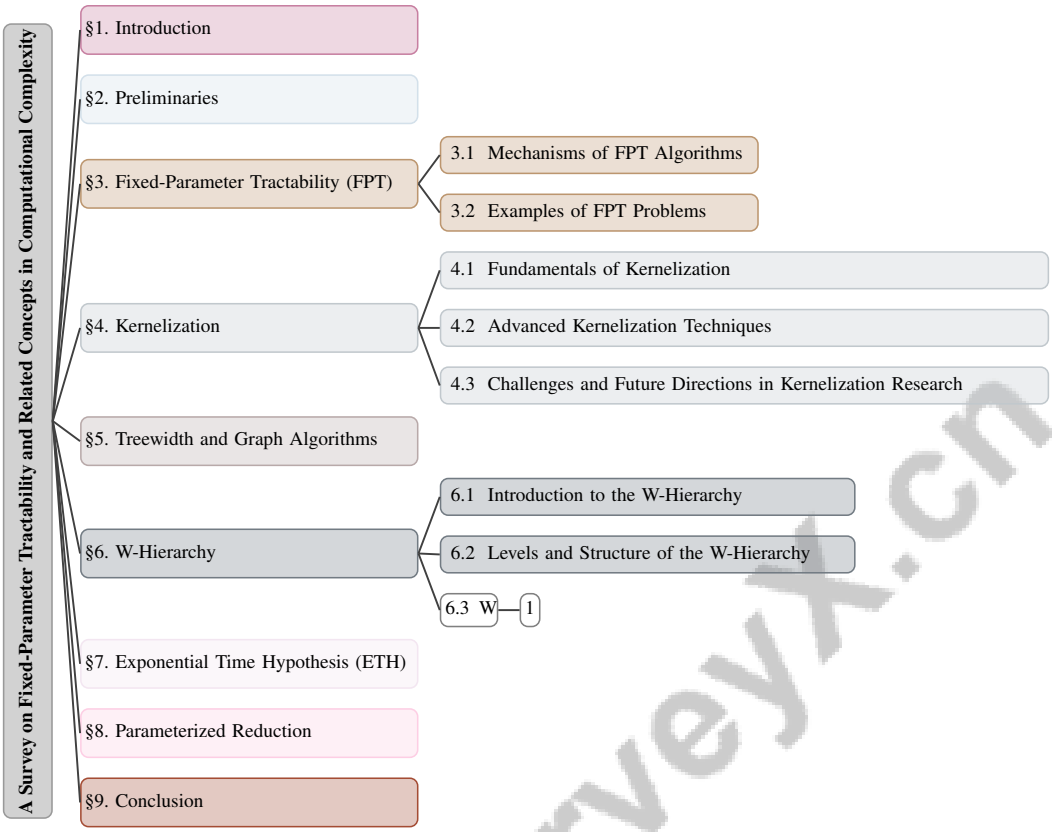

Figure 1: chapter structure

Moreover, FPT is integral to preprocessing techniques such as kernelization, which enhance algorithm performance by reducing problem size while maintaining solvability [8]. The application of FPT in parameterized enumeration problems necessitates innovative methods for achieving parameter-efficient enumeration algorithms, further emphasizing its importance in computational complexity.

1.2 Structure of the Survey

This survey is systematically organized to provide an in-depth exploration of fixed-parameter tractability (FPT) and its related concepts within computational complexity.

The paper begins with an **Introduction**, introducing FPT and highlighting its significance in addressing NP-hard problems, thereby setting the foundation 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 the reader for deeper analyses in later sections.

Section 3, Fixed-Parameter Tractability (FPT), discusses the importance of FPT in tackling computationally hard problems. It examines the mechanics of FPT algorithms and specific problem classes that benefit from FPT solutions. Notably, it explores obstruction sets in characterizing FPT problems, illustrating how parameters like elimination distance, treewidth, and modulator size can lead to efficient algorithms for various graph challenges, including Vertex Cover, Feedback Vertex Set, and d-Cut problems. By providing concrete examples and theoretical insights, this section elucidates the relationship between FPT algorithm design and kernelization [9, 10, 11].

Section 4, Kernelization, investigates kernelization as a preprocessing technique that reduces problem size while preserving solvability. This section explores kernelization applications in parameterized complexity, focusing on its role in efficiently preprocessing NP-hard problems. Specific problems such as Vertex Cover, Independent Set, and Knapsack are examined to illustrate established kernelization

results, emphasizing both theoretical frameworks and practical implications. Various kernelization approaches, including strict polynomial kernels and -approximate kernels, are discussed alongside their respective successes and limitations [12, 13, 14, 15, 16].

Section 5, Treewidth and Graph Algorithms, centers on treewidth's significance in graph algorithms and efficient algorithm design for graph problems. It illustrates practical applications through algorithms like edge-cut width, which serves as an effective alternative to treewidth for NP-hard graph problems. Unlike tree-cut width, edge-cut width can be computed using a fixed-parameter algorithm, proving effective in solving previously intractable problems. Recent advancements have improved computational efficiency, with a new algorithm achieving a runtime of $2^{O(k^2)}n^{O(1)}$ for producing tree decompositions, marking a significant enhancement in treewidth-related problem-solving approaches [17, 18, 19].

In **Section 6, W-Hierarchy**, the W-hierarchy is introduced as a framework for classifying parameterized problems based on computational complexity. This hierarchy builds upon Parameterized Complexity, distinguishing efficiently solvable problems from those likely to be intractable, thus providing insights into structural parameters influencing computational challenges [20, 21, 1]. Different levels of the hierarchy are discussed, along with examples of problems at each level.

Section 7, Exponential Time Hypothesis (ETH), explores ETH and its implications for problem complexity. It explains how ETH establishes lower bounds for complexity, particularly concerning kernelization techniques, approximation algorithms, and the inapproximability of problems like Treewidth. The interplay between kernelization lower bounds and approximation hardness in parameterized complexity is highlighted through examples such as Longest Path, Set Cover, and Treewidth [9, 12, 22, 15].

Section 8, Parameterized Reduction, provides an overview of parameterized reduction techniques, emphasizing kernelization's role in optimizing problem instances. This section discusses various reduction rules, the significance of essential vertices in minimizing search space for fixed-parameter tractable algorithms, and recent advancements and open problems in the field, highlighting the ongoing relevance of parameterized reduction in addressing complex computational challenges [13, 23, 24, 25, 26].

Finally, **Section 9, Conclusion**, summarizes the key points discussed throughout the survey, providing a comprehensive overview of the current research landscape in fixed-parameter tractability (FPT) and its related areas. It emphasizes recent advancements, such as the fixed-parameter tractability of the Subset Feedback Vertex Set problem and the development of work-efficient parallel algorithms. Significant results regarding the fixed-parameter tractability of the MaxSat problem parameterized by matching numbers and the implications of lossy kernelization frameworks are also discussed, along with promising future research directions and unresolved challenges that could enhance the understanding and application of FPT methodologies [27, 28, 29, 15].The following sections are organized as shown in Figure 1.

2 Preliminaries

2.1 Core Concepts and Their Interconnections

In computational complexity, a network of interrelated core concepts is essential for addressing challenging problems. Fixed-Parameter Tractability (FPT) is central, enabling efficient handling of NP-hard problems by focusing on specific parameters. This is complemented by kernelization, a preprocessing technique that reduces problem size while maintaining solvability, particularly in parameterized counting problems, where it compresses instances into manageable forms [30]. The synergy between kernelization and approximation strategies is evident, as demonstrated in their connection with approximation algorithms [31].

Treewidth, a measure of a graph's tree-like structure, is crucial for developing efficient algorithms. Its significance is highlighted in NP-hard problems where bounded treewidth allows polynomial-time solutions, as per Courcelle's Theorem [32]. The interplay between treewidth and other structural parameters, such as treedepth, facilitates the creation of polynomial kernels for problems like F-Deletion, enhancing tractability [33]. Modular decomposition has introduced modular-treewidth, an

intermediate parameter between treewidth and clique-width, improving the management of dense structures [34].

The W-hierarchy classifies parameterized problems by complexity, providing insights into computational challenges. This hierarchy is informed by the Exponential Time Hypothesis (ETH), which posits that certain problems cannot be solved faster than exponential time, setting critical lower bounds for complexity [35]. ETH also guides the development of parameterized reductions, crucial for transforming one parameterized problem into another, aiding classification and hardness proofs within the W-hierarchy.

Parameterized reduction is vital for addressing the complexity of graph algorithms with connectivity constraints. Problems like Odd Cycle Transversal and Multiway Cut require strategic vertex selection for effective graph separations, leveraging concepts such as treewidth and kernelization to manage computational demands [36]. Advances in polynomial-size approximate Turing kernels for graph problems, such as Independent Set and Vertex Cover, mark significant progress in approximating previously challenging problems [37].

The interconnectedness of core concepts—FPT, kernelization, treewidth, the W-hierarchy, ETH, and parameterized reduction—constitutes a robust framework for addressing computational complexity. This framework enhances our understanding of problem structures and is essential for developing innovative algorithmic solutions across various domains. It emphasizes the interplay between foundational concepts like kernelization complexity and graph parameters, such as clique-width, which are crucial for solving problems like Vertex Cover and Independent Set. This framework underscores its potential to advance theoretical computer science and enhance practical applications across multiple fields by facilitating the exploration of diverse solutions with minimal additional computational cost [12, 20].

3 Fixed-Parameter Tractability (FPT)

3.1 Mechanisms of FPT Algorithms

Fixed-Parameter Tractable (FPT) algorithms are pivotal for solving computationally challenging problems by exploiting specific parameters like clique-width and treewidth to enhance efficiency. Recent FPT advancements focus on work-efficient parallel algorithms and novel structural parameters, leading to faster exact and approximation algorithms for graph problems such as Vertex Cover and Maximum Matching. Techniques such as kernelization and modular decomposition are crucial for improving algorithmic efficiency and resolving NP-hard problems in polynomial time under certain parameterizations [11, 10, 29, 38]. These algorithms employ dynamic programming, kernelization, and parameterized reductions to manage complexity effectively.

Dynamic programming on graph decompositions, particularly tree decompositions, leverages graph structure for efficient problem-solving, as seen in the Generalized Feedback Vertex Set problem on bounded treewidth graphs [39]. This technique also identifies minimal separators of specified sizes, demonstrating its versatility [40]. The analysis of 1-safe Petri nets further illustrates the use of parameters like treewidth, benefit depth, and vertex cover number to assess complexity [41].

Kernelization is another key mechanism, reducing problem instances to smaller, equivalent forms. Lossy Kernelization, as applied to the Connected Vertex Cover problem, exemplifies this by ensuring polynomially bounded output sizes [42]. Polynomial-time compression techniques further highlight kernelization intricacies [43].

Parameterized streaming algorithms introduce a new dimension to FPT, enabling efficient data processing in a streaming context with space management based on additional parameters [3]. This approach is advantageous for large-scale data processing without requiring the entire dataset in memory.

Advanced techniques like Turing kernelization decompose input graphs into smaller components, query an oracle for information, and reduce the problem to a smaller equivalent instance based on the oracle's responses, enhancing FPT algorithm efficiency [44].

Novel parameters, such as sm-width, facilitate FPT algorithm development for problems otherwise intractable with clique-width [45]. This innovation underscores parameterization's importance in broadening FPT algorithm applicability. The complexity classification of the Tutte polynomial

evaluation based on treewidth, pathwidth, and cutwidth further exemplifies structural parameters' utility in refining problem spaces for FPT solutions [36].

These strategies highlight parameterization, structural insights, and innovative algorithmic techniques' significance in FPT algorithm development. These advanced methods enhance our ability to tackle intricate challenges, such as vertex subset problems and parameterized complexity frameworks, while establishing a comprehensive understanding of inherent structures and parameters—like the number of tokens and discovery budgets—that influence problem complexity [12, 46, 1].

As illustrated in Figure 2, which depicts the core mechanisms of Fixed-Parameter Tractable (FPT) algorithms, various key techniques, structural parameters, and advanced methodologies enhance algorithm efficiency for complex problems. The accompanying figure showcases images representing different graph structures, each highlighting unique aspects of these algorithms. The first image presents a hierarchical structure of interconnected nodes and edges, emphasizing foundational relationships within an FPT context. The second image showcases graphs representing various conditional probability distributions, demonstrating how FPT algorithms manage complex probabilistic relationships. The third image displays graphs with diverse node labels and edge types, underscoring the adaptability of FPT algorithms in handling varied configurations. Collectively, these visual examples illustrate the intricate mechanisms through which FPT algorithms operate, emphasizing their capacity to efficiently solve complex problems by leveraging specific structural properties of graphs [11, 47, 38].

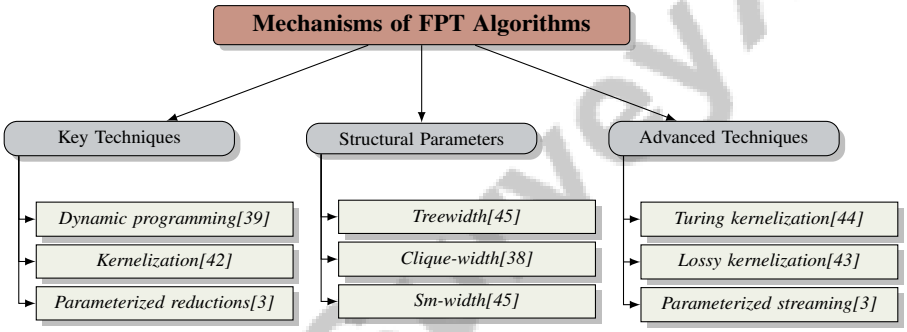

Figure 2: This figure illustrates the core mechanisms of Fixed-Parameter Tractable (FPT) algorithms, highlighting key techniques, structural parameters, and advanced methodologies that enhance algorithm efficiency for complex problems.

3.2 Examples of FPT Problems

Fixed-Parameter Tractable (FPT) problems showcase the transformative potential of parameterization in managing computationally intensive tasks. The Collapsed k-Core problem exemplifies this by identifying a vertex set whose removal results in a k-core of bounded size, demonstrating parameterization's effectiveness in graph theory [48].

The s-Club Cluster Edge Deletion problem, which modifies a graph to ensure each connected component is an s-club, is FPT when parameterized by the sum of s and the graph's treewidth, highlighting the utility of combining multiple parameters for tractability [49].

The Traveling Salesperson Problem (TSP), a classical NP-hard problem, demonstrates fixed-parameter tractability when parameterized by the budget constraint on the total walk weight, allowing efficient solutions via polynomial kernelization [50].

In graph modification, the Vertex Planarization problem, which involves removing the minimum vertices to render a graph planar, becomes tractable when parameterized by the number of vertices to be removed, enabling efficient algorithms [51].

The Firefighting problem, modeling strategic vertex defense to prevent fire spread, is FPT when parameterized by a modulator's size to specific graph classes, illustrating structural parameterization's role in achieving tractability for complex optimization problems [4].

In approximation contexts, FPT algorithms significantly enhance approximation ratios for problems like the capacitated vertex cover and vector dominating set, particularly when parameterized by

treewidth, underscoring FPT algorithms' role in improving solution quality for challenging problems [52].

Explicit FPT algorithms for edge-deletion problems in interval graphs, parameterized by treewidth, demonstrate potential for faster solutions compared to general theorems like Courcelle's, highlighting efficiency gains through specialized FPT techniques [53].

The introduction of novel parameters, such as parameter in vertex deletion problems, facilitates advanced kernelization methods that improve existing techniques. This parameterization supports randomized polynomial kernelization strategies, showcasing FPT approaches' adaptability in tackling complex problems [54].

These examples collectively illustrate parameterization's significant role in computational complexity, transforming NP-hard problems into more manageable forms. This approach facilitates efficient algorithm development and establishes a structured framework for analyzing various problems' tractability, enabling the resolution of instances otherwise deemed intractable. By integrating techniques like kernelization and approximation, researchers can uncover new problem-solving pathways, enhancing our understanding of parameterized complexity's boundaries and capabilities [13, 55, 56, 57, 58].

4 Kernelization

Kernelization serves as a cornerstone in computational complexity, particularly for fixed-parameter tractable (FPT) problems, by simplifying problem instances into smaller, equivalent forms. This section delves into the core principles and advanced techniques of kernelization, highlighting its role in enhancing computational efficiency across diverse challenges.

4.1 Fundamentals of Kernelization

Method Name	Reduction Techniques	Structural Parameters	Integration Strategies
RPK-VC[54]	Reduction Rules	Maximum Matching	Approximation Methods
PFAFF[7]	Tree Decomposition	Treewidth	Fixed-parameter Tractable

Table 1: Overview of kernelization methods, highlighting reduction techniques, structural parameters, and integration strategies. The table compares the RPK-VC and PFAFF methods, detailing their approach to problem reduction and parameterized algorithm design.

Kernelization transforms problem instances into reduced forms, preserving solvability while improving efficiency in FPT problems through the creation of a kernel—a simplified instance bounded by a function of a chosen parameter [54]. Central to this process are polynomial-time compression algorithms, which reduce problem complexity by focusing on structural parameters like treewidth and feedback vertex sets [43]. These parameters inform reduction strategies, as seen in flat folding studies that utilize bounded parameters such as ply and treewidth [7]. The integration of kernelization with approximation strategies further enhances data reduction, as evidenced in studies on Vertex Cover and Independent Set, which explore polynomial kernels for weighted problems and introduce lossy kernelization frameworks [12, 13, 14, 15, 54]. Table 1 provides a comprehensive comparison of key kernelization methods, illustrating how different reduction techniques, structural parameters, and integration strategies are employed in the field.

4.2 Advanced Kernelization Techniques

Recent advancements have refined kernelization through innovative methods exploiting structural parameters. Techniques like approximating treedepth modulators and applying protrusion replacement rules demonstrate the importance of understanding graph structures for effective kernelization [59]. The use of flowers instead of sunflowers in algorithms significantly improves kernel sizes [60], while the Adaptive Optimization Algorithm (AOA) exemplifies dynamic strategy modification for enhanced efficiency [8].

Table 2 provides a detailed overview of advanced kernelization techniques, showcasing the interplay between structural features, parameter-driven approaches, and their specific applications in planar and TSP graphs. Figure 3 illustrates the advanced kernelization techniques categorized into graph

Method Name	Structural Features	Parameter-Driven Approaches	Applicable Scenarios
KSP[59]	Treedepth Modulators	Treewidth, Feedback	Planar, Tsp
LKP[60]	Flower Lemma	Counting Arguments	Np-hard Problems
AOA[8]	Treedepth Modulators	Treewidth Feedback Vertex	Planar Tsp Graphs
KDP[61]	Protrusion Replacement	Treewidth	Planar Graphs
PK-TSP[50]	Treedepth Modulators	Vertex Cover Number	Planar And Tsp
FVS-VC[62]	Protrusion Replacement	Feedback Vertex Set	Planar Graphs

Table 2: This table presents a comparative analysis of advanced kernelization methods, detailing the structural features, parameter-driven approaches, and applicable scenarios for each method. The methods are categorized based on their utilization of graph structures, such as treedepth modulators and protrusion replacement, and their application to specific graph types, including planar and TSP graphs. This overview highlights the innovative methodologies that enhance kernelization efficiency and applicability in computational graph theory.

structural methods, parameter-driven approaches, and specific applications in planar and TSP graphs. Each category highlights key innovations and methodologies that enhance kernelization efficiency and applicability. Linear kernels for H-topological-minor-free graphs, leveraging treewidth-bounding and finite integer index, mark a breakthrough by combining decomposition techniques with novel preprocessing steps [63, 64]. The detection of c-essential vertices and the development of fixed-parameter tractable algorithms for maximum minimal blocking sets (mmbs) further enhance kernelization efficiency [23, 65]. In planar graphs, preserving linkage properties for disjoint paths problems is crucial [61], and polynomial-sized kernels for TSP highlight parameter-driven approaches [50]. Innovations in kernels cubic in the feedback vertex set size emphasize parameter-driven preprocessing [62]. These advancements illustrate the robust interplay between theoretical insights and practical applications, expanding computational feasibility and enriching our understanding of kernelization [12, 15, 66].

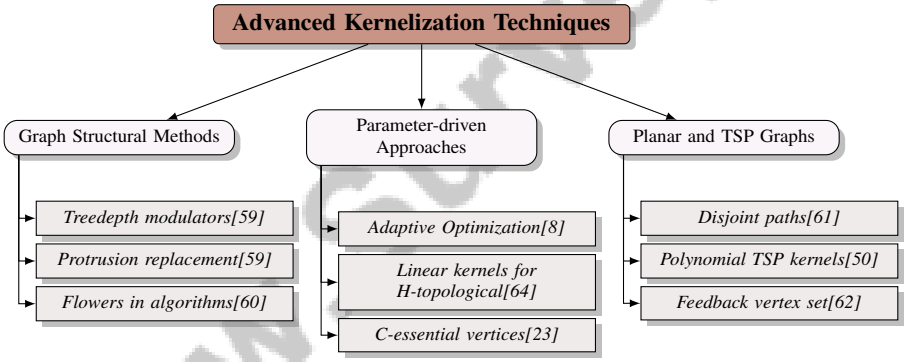

Figure 3: This figure illustrates the advanced kernelization techniques categorized into graph structural methods, parameter-driven approaches, and specific applications in planar and TSP graphs. Each category highlights key innovations and methodologies that enhance kernelization efficiency and applicability.

4.3 Challenges and Future Directions in Kernelization Research

Method Name	Structural Features	Performance Variability	Framework Development
PBK-PFD[67]	Planar Graphs	Randomized Approaches	Protrusion Techniques
RPK-VC[54]	Maximum Matching	Randomized Techniques Variability	Randomized Polynomial Kernelization
CED[23]	C-essential Vertices	Method's Effectiveness Varies	Refined Theoretical Frameworks
KDP[61]	Planar Graph Properties	Randomized Approaches Inconsistency	Polynomial Kernel Status
CK[68]	C-closure Parameter	Randomized Approaches Variability	C-Closure Kernelization

Table 3: This table presents a comparative analysis of various kernelization methods, highlighting their structural features, performance variability, and framework development strategies. The methods examined include PBK-PFD, RPK-VC, CED, KDP, and CK, each associated with specific graph properties and kernelization techniques. The table underscores the challenges and potential directions for future research in the field of kernelization.

Kernelization research encounters challenges in developing efficient polynomial kernels across diverse parameterized problems, often constrained by specific structural graph characteristics [67].

Table 3 provides a detailed comparison of different kernelization methods, emphasizing their structural features, performance variability, and the frameworks developed to address these challenges in kernelization research. The variability in performance, especially with randomized techniques, poses consistency challenges [54]. Dynamic programming complexities in certain parameterizations and identifying lower bounds for Turing kernels highlight the need for refined frameworks [69, 46]. Future research could explore constant-factor approximations in c-essential detection [23], refine kernelization techniques for related graph theory problems [61], and develop frameworks for polynomial compressions in counting problems [68]. Investigating edge modification problem approximability within a theoretical framework could further inform kernelization strategy development [31]. Addressing these challenges will advance kernelization techniques, enhancing computational solutions for complex problems like Vertex Cover and Knapsack, while elucidating the limitations and potential of polynomial-size kernels [12, 13, 24, 14, 15].

5 Treewidth and Graph Algorithms

5.1 Understanding Treewidth

Treewidth is a pivotal concept in graph theory, measuring how closely a graph approximates a tree structure. Defined through a tree decomposition, treewidth is determined by the size of the largest "bag" of vertices minus one. Although computing treewidth is NP-complete, it remains essential for developing efficient algorithms for NP-hard problems, as graphs with bounded treewidth often permit polynomial-time solutions [70]. Its significance is underscored in fixed-parameter algorithms, enabling subexponential running times for otherwise intractable problems, such as the k-arc Chinese Postman Problem (CPP), where tree decompositions aid in constructing fixed-parameter algorithms [71]. This utility stems from the structural properties of graphs, facilitating efficient computations of independent and blocking sets [65].

Treewidth also plays a crucial role in approximating graph parameters, supporting the development of fixed-parameter tractable (FPT) approximation algorithms for problems like capacitated vertex cover and vector dominating set, thus enhancing algorithmic performance [52]. Understanding its interplay with other graph parameters, such as pathwidth and clique-width, enriches problem-solving strategies. In practice, treewidth is often used alongside structural parameters like edge-cut width and vertex cover size, improving the efficiency of fixed-parameter algorithms for NP-hard problems [70, 17, 72, 18, 73]. For instance, in s-club cluster problems, treewidth facilitates handling larger instances, demonstrating adaptability across computational settings. Reduction rules in tree decompositions help identify smaller equivalent instances, enhancing preprocessing capabilities.

Treewidth is thus a cornerstone in parameterized complexity, enabling graph decomposition into simpler structures and guiding innovative algorithm development. Advanced preprocessing techniques, such as -approximate kernelization and essential vertex identification, are critical in enhancing solution efficiency for complex problems across various domains by reducing instances to manageable sizes and fostering fixed-parameter tractable algorithms [23, 15].

5.2 Treewidth in Algorithm Design

Treewidth is integral to designing efficient algorithms for computationally intensive problems, showcasing versatility in dynamic programming and approximation strategies. Its utility is pronounced in planar graphs, where decomposition into bounded treewidth components enables effective dynamic programming solutions, facilitating efficient resolution of complex problems through subexponential parameterized algorithms [70]. The Improved Treewidth Approximation Algorithm (ITAA) exemplifies treewidth's adaptability, minimizing dependence on parameter k and streamlining branching processes in approximation contexts [74]. This adaptability extends to algorithms that significantly reduce exponential dependence on treewidth while maintaining linear time complexity concerning vertex count, optimizing performance.

Preprocessing techniques are vital for leveraging treewidth in algorithm design. The existence of a polynomial kernel when parameterized by feedback vertex set size or vertex cover provides a theoretical basis for effective preprocessing strategies [70]. This insight simplifies problem spaces through targeted preprocessing, enabling more efficient algorithm design. Central to this process are algorithms that compute tree decompositions, such as those achieving width $O(k)$ in $O(nk^2 \log k)$

time for n-vertex planar graphs with treewidth k. These advancements underscore significant efficiency gains, allowing many problems expressible in linear extended monadic second order to be solved in linear time relative to vertex count, demonstrating practical benefits in computational graph theory [70, 75, 18]. The role of treewidth in algorithm design is further emphasized by organizing algorithms into stages, including tree decomposition enumeration, optimal decomposition construction, and simplifications leading to linear-time solutions.

5.3 Challenges and Limits of Treewidth

Despite its advantages, treewidth application in algorithm design faces challenges, especially for complex graph problems. A primary challenge is approximating treewidth, particularly in larger graphs where existing algorithms may be inefficient for small k values [74]. This inefficiency necessitates refined approximation methods for large-scale instances without performance loss. Unbounded treewidth scenarios present another limitation, where counting problem complexity becomes intractable, complicating efficient algorithm development or approximation [76]. This intractability poses a barrier to treewidth-based approaches for problems beyond bounded treewidth contexts.

Structural parameters like edge-cut width reveal further challenges. Edge-cut width is not closed under edge or vertex deletion, limiting its effectiveness in scenarios requiring structural modifications [17]. This necessitates careful assessment of graph structural properties when employing treewidth or related parameters in algorithm design. The specificity of findings, such as those in Grundy Coloring, highlights another challenge: potential lack of generalizability across different graph problems and parameters [77]. While treewidth is powerful for specific applications, it may not seamlessly extend to all graph theory areas, indicating a need for 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 [78]. This highlights the need for innovative approaches to transcend these limitations and enhance treewidth-based method efficiency.

Figure 4 illustrates the primary challenges and limitations associated with the application of treewidth in algorithm design, categorized into approximation challenges, structural parameter issues, and bounded-degree constraints. Each category highlights specific issues such as inefficiency in large-scale graphs, the specificity of structural parameters, and the limitations of dynamic programming under bounded-degree constraints.

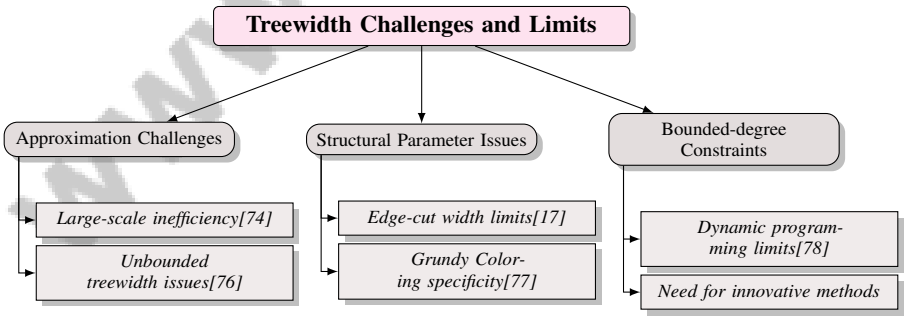

Figure 4: This figure illustrates the primary challenges and limitations associated with the application of treewidth in algorithm design, categorized into approximation challenges, structural parameter issues, and bounded-degree constraints. Each category highlights specific issues such as inefficiency in large-scale graphs, the specificity of structural parameters, and the limitations of dynamic programming under bounded-degree constraints.

6 W-Hierarchy

The W-hierarchy is a pivotal framework in parameterized complexity, delineating levels of computational difficulty for various problems. This section investigates the W-hierarchy's structure and

significance, offering insights into complexities within parameterized contexts and their implications for algorithmic strategies. As illustrated in Figure 5, the hierarchical structure of the W-hierarchy categorizes problems into levels based on their computational difficulty. This figure highlights key principles and connections to graph theory, emphasizing the significance of W[1]-hardness in algorithm development and complexity analysis. The following subsection introduces the W-hierarchy, highlighting its principles and role in classifying parameterized problems.

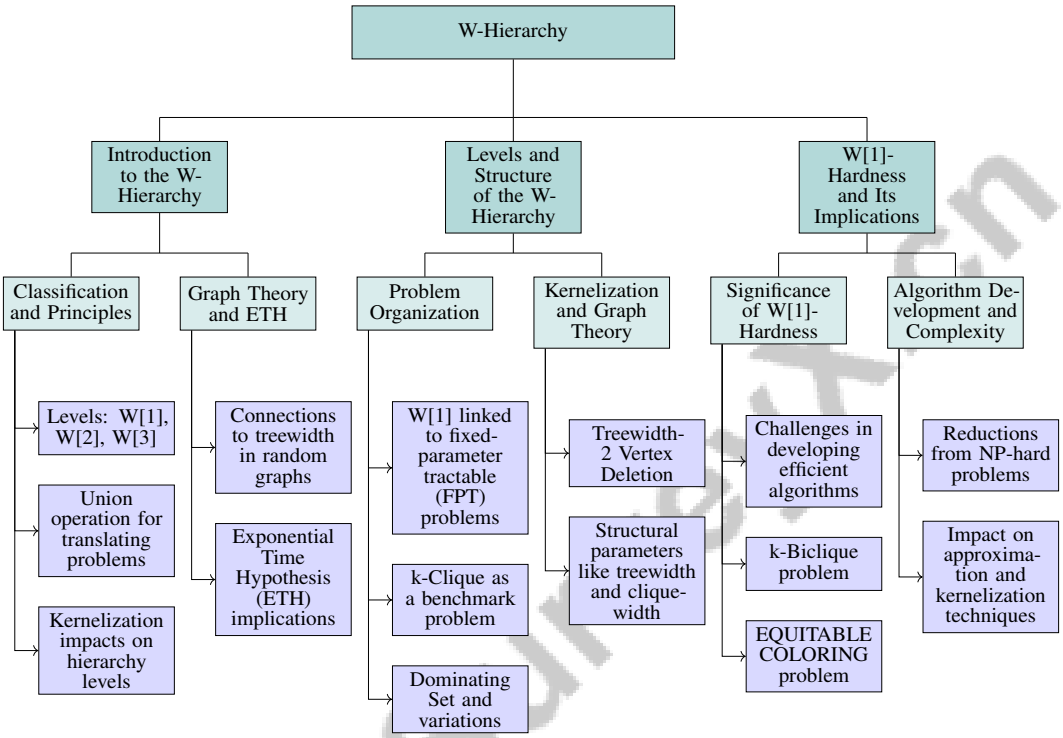

Figure 5: This figure illustrates the hierarchical structure of the W-hierarchy in parameterized complexity, categorizing problems into levels based on computational difficulty. It highlights key principles, connections to graph theory, and the significance of W[1]-hardness in algorithm development and complexity analysis.

6.1 Introduction to the W-Hierarchy

The W-hierarchy classifies problems by computational difficulty, akin to classical complexity classes. Levels such as W[1], W[2], and W[3] categorize problems based on complexity traits, enhancing understanding of parameterized complexity and strategies like kernelization and Turing kernelization [25, 79, 20, 80]. The 'union operation' is a significant innovation, translating problems between classical and W-classes, thus highlighting the W-hierarchy's versatility in capturing parameterized complexity nuances [81].

Kernelization profoundly impacts the W-hierarchy, with polynomial kernels influencing problem classification. The existence of polynomial kernels can lead to the collapse of certain W-hierarchy levels, illustrating the interplay between kernelization and complexity [82]. A new hierarchy of kernelizability distinguishes between problems admitting polynomial Turing kernels and those WK[1]-hard, refining complexity perspectives [35].

The W-hierarchy's relevance is underscored by connections to graph theory width measures like treewidth, especially in random graphs where width measures are critical for determining graph problem complexity [83]. The Exponential Time Hypothesis (ETH) further emphasizes the W-hierarchy's role in parameterized complexity, providing a framework for understanding algorithmic efficiency limits [7].

6.2 Levels and Structure of the W-Hierarchy

The W-hierarchy organizes problems by computational difficulty, with levels W[1], W[2], W[3], and beyond. W[1] is foundational, often linked to fixed-parameter tractable (FPT) problems. The k-Clique problem exemplifies this, serving as a benchmark for understanding graph-related problem hardness, fixed-parameter tractability, and kernelization [84, 85, 20, 38]. W[1] demarcates tractable and intractable parameterized problems, a critical reference for complexity analyses.

Progressing to W[2], W[3], and beyond, problems increase in complexity, often requiring sophisticated reductions for classification. The Dominating Set problem, classified as W[2]-complete, illustrates this complexity, seeking a vertex subset such that each graph vertex is included or adjacent to a subset vertex. Variations like the $[\sigma, \rho]$ Dominating Set problem offer insights into polynomial kernelization and tractability under specific conditions, particularly when parameterized by structural properties like treewidth or vertex cover number [86, 87, 88, 69].

The W-hierarchy's structure enriches connections to graph theory and parameterized complexity. Kernelization techniques for problems like Treewidth-2 Vertex Deletion build on foundational hierarchy work, offering explicit bounds and enhancing problem complexity understanding [89]. These advancements illustrate how W-hierarchy insights inform efficient algorithm development and preprocessing techniques.

The W-hierarchy provides a nuanced framework for understanding parameterized problem complexity. Parameterized complexity levels facilitate exploring computational challenges, while structural insights create a roadmap for investigating tractability limits. This includes identifying effective enumeration strategies and the interplay between structural parameters like treewidth and clique-width, significantly influencing algorithmic efficiency and complexity thresholds in NP-hard problems [90, 79].

6.3 W[1]-Hardness and Its Implications

W[1]-hardness is crucial in the W-hierarchy, representing problems conjectured intractable when parameterized by specific parameters. A problem is W[1]-hard if it is as challenging as the most difficult W[1] problems, implying it is unlikely to be fixed-parameter tractable (FPT) unless the Exponential Time Hypothesis (ETH) fails [1]. This classification serves as a critical demarcation in parameterized complexity, indicating problems for which efficient parameterized algorithms are not anticipated.

The implications of W[1]-hardness are significant for problem classification in the W-hierarchy. The k-Biclique problem is widely regarded as W[1]-hard, highlighting challenges in developing efficient algorithms [91]. Similarly, the EQUITABLE COLORING problem's complexity is underscored by its W[1]-hardness for various graph classes, established through reductions from the BIN-PACKING problem [92].

The complexity of addressing W[1]-hard problems is illustrated by the Firefighting problem, demonstrating W[1]-hardness for specific graph classes and emphasizing computational challenges [4]. Structural parameterizations in bounded-degree graphs provide reductions highlighting the impossibility of surpassing existing algorithms under the Strong Exponential Time Hypothesis (SETH) and ETH, reinforcing W[1]-hard problems' inherent complexity [78].

Classifying problems as W[1]-hard has significant ramifications for algorithm development and exploring parameterized complexity. Establishing W[1]-hardness typically requires reductions from established NP-hard problems, essential for defining tractability limits in computational complexity. This delineation is crucial for identifying problems effectively tackled using parameterized algorithms, particularly in developing new techniques for approximation and kernelization, vital for addressing NP-hard challenges. Understanding the interplay between parameterization, approximation, and inherent hardness is increasingly important [13, 23, 21, 56, 15]. These reductions guide innovative algorithmic strategies and efficiency limits.

W[1]-hardness is crucial in classifying parameterized problems, delineating tractability boundaries by identifying problems unlikely to be solvable in polynomial time. This classification shapes theoretical research in computational complexity and significantly impacts practical applications, particularly in algorithm design and approximation techniques for NP-hard problems. As researchers explore new methodologies and frameworks, such as kernelization and diminishable problems,

understanding W[1]-hardness continues to evolve, influencing efficient algorithm development and exploring hardness results [13, 56]. Its implications extend across various domains, shaping the landscape of parameterized complexity and guiding the pursuit of efficient solutions for challenging computational problems.

7 Exponential Time Hypothesis (ETH)

7.1 Introduction to ETH and Its Role in Complexity Theory

The Exponential Time Hypothesis (ETH) posits that certain NP-complete problems cannot be solved in sub-exponential time, specifically $2^{o(n)}$ for instances of size n. This hypothesis is pivotal in computational complexity theory, providing a framework for understanding algorithmic efficiency limits, particularly within parameterized complexity. ETH facilitates the establishment of lower bounds on computational problems, offering insights into their inherent difficulty. It underpins proofs regarding the absence of polynomial-size kernels for specific problems and aids in demonstrating the NP-hardness of approximating significant computational tasks. Thus, ETH delineates efficient algorithm design boundaries and informs the development of preprocessing and approximation strategies [13, 23, 22, 9, 15].

ETH's influence extends to the classification of parameterized problems, serving as a benchmark for evaluating fixed-parameter tractable (FPT) solutions. By assuming ETH, researchers derive asymptotically optimal algorithms, such as the one for finding maximum minimal separators, which aligns with ETH's lower bounds by operating in time $2^{O(k)}n^{O(1)}$ [40]. Moreover, ETH impacts connectivity problem exploration, where improving constants in existing results could significantly affect the Strong Exponential Time Hypothesis (SETH), an extension refining our understanding of computational limits [93].

7.2 ETH and Lower Bounds in Parameterized Complexity

ETH is instrumental in parameterized complexity, underpinning the establishment of lower bounds for various computational problems. It suggests that certain NP-hard problems, such as METRIC DIMENSION and STRONG METRIC DIMENSION, cannot be solved in sub-double-exponential time on bounded diameter graphs unless ETH is disproven, enhancing our understanding of algorithmic solution limits [94, 95, 13]. ETH establishes a benchmark for assessing FPT algorithm feasibility by positing that certain NP-complete problems cannot be solved in sub-exponential time.

A significant implication of ETH is its influence on strict polynomial kernels. The relationship between ETH and kernelization is examined in diminishable parameterized problems, where strict polynomial kernels depend on ETH's validity [13]. This underscores ETH's role in delineating algorithmic efficiency boundaries and its impact on kernelization strategies.

ETH also informs lower bounds for specific parameterized problems, such as the Vertex Planarization problem, demonstrating that unless ETH fails, no algorithm can solve it on graphs of bounded treewidth in time better than $2^{o(w \log w)} \cdot n^{O(1)}$ [51]. This establishes critical lower bounds, emphasizing the problem's inherent complexity and the challenges of achieving efficient solutions.

Moreover, ETH is crucial in understanding the minimum degree barrier for kernelization, particularly regarding the Vertex Cover problem. Surpassing this barrier would necessitate a breakthrough in parameterized complexity understanding, emphasizing ETH's role in shaping kernelization research [24].

The defensive alliance problem exemplifies ETH's impact, being shown to be W[1]-hard when parameterized by treewidth, thus ruling out FPT algorithms under the assumption that W[1] is not equal to FPT [96]. This highlights ETH's importance in classifying problems within the W-hierarchy and understanding their computational limits.

Additionally, ETH provides insights into the complexity of feedback vertex set-related problems, establishing lower bounds that demonstrate these problems cannot be solved in time $2^{o(tw \log tw)} \cdot n^{O(1)}$ unless ETH fails [32]. This reinforces ETH's significance in setting theoretical limits for parameterized algorithms and guiding efficient solution development.

7.3 Challenges and Open Questions in ETH Research

ETH remains a critical and unresolved issue in computational complexity theory, raising profound questions about algorithmic efficiency limits. Recent research highlights challenges, such as whether exact algorithms for determining irredundance numbers in graphs can operate below the exponential time threshold of $\Omega(2^n)$. New findings indicate that approximating treewidth within certain bounds is NP-hard, necessitating time complexity of $2^{\Omega(n)}$, unless ETH is disproven. These developments underscore ETH's pivotal role in shaping our understanding of algorithmic limits and computational problem complexity [95, 22, 29, 15]. A primary challenge is verifying ETH itself, which remains a conjecture without definitive proof or disproof, leaving open questions about computational tractability boundaries and potential breakthroughs that could either strengthen or refute the hypothesis.

A critical inquiry area involves exploring ETH's implications for specific computational problems, particularly in parameterized complexity. Researchers are interested in determining whether ETH can establish tighter lower bounds for problems that have resisted efficient solutions, such as feedback vertex set-related problems and their parameterizations [32].

Another challenge lies in exploring the relationship between ETH and other complexity hypotheses, such as SETH. While ETH provides a foundational framework for understanding NP-complete problems, SETH offers a more granular perspective on specific problem complexities. The interplay between these hypotheses raises questions about potential unified theories that could provide deeper insights into computational hardness [93].

Developing new techniques for proving ETH-based lower bounds represents a significant research frontier. Current methods often rely on reductions from known hard problems, highlighting the need for innovative approaches that can extend these techniques to a broader range of problems. This includes exploring ETH alongside width measures, such as treewidth, to establish more refined complexity classifications [40].

Furthermore, applying ETH to emerging computational research areas, such as quantum computing and machine learning, presents both challenges and opportunities. The impact of ETH on algorithm evolution in computational complexity, especially concerning problems like Vertex Cover and its kernelization, remains unresolved and could significantly shape future algorithm design and analysis. This inquiry includes understanding minimum degree barriers for polynomial-time kernelization, exploring automated kernelization through AI-guided approaches, and assessing kernelization complexities of various solution discovery problems, underscoring the intricate interplay between theoretical frameworks and practical algorithmic advancements [12, 1, 24, 15, 97].

8 Parameterized Reduction

8.1 Concept and Importance of Parameterized Reduction

Parameterized reduction is a pivotal technique in computational complexity, particularly within the realm of parameterized complexity, facilitating the transformation of NP-hard problems into more tractable instances. This method is crucial for developing fixed-parameter tractable (FPT) algorithms and is often employed alongside kernelization and approximation strategies to tackle NP-hard challenges, thus probing the boundaries of computational feasibility [98, 13, 79, 56]. By transforming one parameterized problem into another while maintaining the parameterization, parameterized reduction simplifies problem structures, aiding in the classification of problems based on complexity and identifying those that are FPT or W[1]-hard.

The systematic approach of parameterized reduction enables researchers to explore inter-problem relationships and complexities, crucial for deriving hardness results and understanding computational limits of specific problem classes. For instance, the subset feedback vertex set problem, which involves finding a set of at most k vertices intersecting all simple cycles through a specified subset S in an undirected graph G, exemplifies how parameterized reduction retains core computational challenges while simplifying instances [28].

Moreover, parameterized reduction is integral to developing parameterized enumeration algorithms (PEA), which use complexity classes to facilitate solution enumeration based on parameters [79]. This approach systematically explores solution spaces, offering insights into the structure and complexity

of parameterized problems, allowing PEA to efficiently enumerate solutions even in challenging scenarios.

8.2 Techniques and Methods in Parameterized Reduction

Parameterized reduction is foundational in parameterized complexity, providing techniques to transform problems while preserving computational characteristics. These reductions are essential for establishing the parameterized complexity of numerous problems, demonstrating hardness, and delineating fixed-parameter tractability (FPT) boundaries. Recent research highlights the role of structural parameters—such as modulator size, elimination distance, and treewidth—in developing efficient exact algorithms and FPT approximation schemes for graph problems. Advances in kernelization and essential vertex identification have refined search spaces and enhanced FPT algorithm efficiency, deepening our understanding of parameterized complexity [10, 13, 23, 29].

A key technique in parameterized reduction is problem kernels, aimed at reducing 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 yields a reduced instance retaining core challenges [42]. Kernelization simplifies instances, making them more amenable to efficient algorithmic solutions.

Turing reductions decompose problems into smaller subproblems solvable independently or with oracle queries, particularly effective in graph problems where complexity is distributed across subproblems parameterized by structural properties [44]. This facilitates granular exploration of problem complexity, enabling innovative algorithmic strategies.

Lossy kernelization expands parameterized reduction by allowing approximate solutions, trading exactness for efficiency. This is relevant when exact solutions are computationally prohibitive, as approximate solutions can suffice [42]. By relaxing exactness, lossy kernelization broadens parameterized reduction's applicability.

Integrating structural parameters like treewidth and feedback vertex sets into parameterized reduction techniques further aids in managing complexity. These parameters offer insights into graph structural properties, facilitating effective reductions and kernelization strategies. The study of flat folding illustrates this, reducing complexity by focusing on bounded parameters such as ply and treewidth [7].

9 Conclusion

9.1 Conclusion

The exploration of fixed-parameter tractability (FPT) within computational complexity reveals a landscape rich with both challenges and potential advances. A primary challenge lies in addressing the W[1]-hard problems, which demand innovative parameterizations and refined reduction techniques to achieve tractability. Future research endeavors could extend existing methodologies to encompass a wider range of graph classes and refine kernel sizes for specific problems. Enhancing algorithmic efficiency through advanced connectivity constraints and adapting methods for related issues, such as Connected -Treewidth Deletion, offers a promising avenue for exploration. The examination of problems like Collapsed k-Core, with a focus on larger k values and the potential for polynomial kernels, could yield significant insights. Moreover, parameterizing problems such as s-Club Cluster Edge Deletion by factors like k or the number of clusters may lead to the development of subexponential-time algorithms.

The parameterized complexity of maximum minimal blocking sets highlights the utility of FPT in tackling larger graph instances, suggesting that refining these algorithms could enhance their application to related graph theory challenges. The pursuit of polynomial kernels across various parameters and their applicability to diverse graph classes remains a pivotal area for further investigation. While theoretical results, such as those related to Vertex Planarization, are prevalent, there is a pressing need for practical algorithmic implementations. Future work should focus on developing practical algorithms for graph modification challenges and optimizing kernelization processes for broader applicability.

Additionally, exploring deterministic versions of randomized kernelization methods and their application to other NP-hard problems could lead to substantial advancements. Potential research avenues

include examining the fixed-parameter tractability of various list problems related to parameters like feedback vertex set size or vertex cover number. Further investigation into the structural properties of more complex graph classes may enhance the applicability of polynomial kernels in edge deletion problems. The potential of c-closure in other graph problems, along with improvements to kernelization techniques for a broader range of graph classes, represents a fertile ground for future research. Addressing these challenges and pursuing these directions will deepen our understanding of computational complexity and expand the horizons 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] Guilherme C. M. Gomes and Ignasi Sau. Finding cuts of bounded degree: complexity, fpt and exact algorithms, and kernelization, 2019.

[10] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

[25] Karl Bringmann, Danny Hermelin, Matthias Mnich, and Erik Jan van Leeuwen. Parameterized complexity dichotomy for steiner multicut, 2015.

[26] Sudeshna Kolay and Fahad Panolan. Parameterized algorithms for deletion to (r,l)-graphs, 2015.

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

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

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

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

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

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

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

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

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

[36] 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.

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

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

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

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

[41] M. Praveen and Kamal Lodaya. Parameterized complexity results for 1-safe petri nets, 2011.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[58] Robert Ganian, Mathis Rocton, and Daniel Unterberger. The parameterized complexity landscape of the unsplittable flow problem, 2024.

[59] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

[73] Yuri Faenza, Gonzalo Muñoz, and Sebastian Pokutta. New limits of treewidth-based tractability in optimization, 2019.

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

[75] Frank Kammer and Torsten Tholey. Approximate tree decompositions of planar graphs in linear time, 2016.

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

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

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

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

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

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

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

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

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

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

[86] Eva-Maria C. Hols and Stefan Kratsch. On kernelization for edge dominating set under structural parameters, 2019.

[87] Omid Amini, Fedor V. Fomin, and Saket Saurabh. Parameterized algorithms for partial cover problems, 2008.

[88] Pradeesha Ashok, Rajath Rao, and Avi Tomar. Polynomial kernels for generalized domination problems, 2022.

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

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

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

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

[93] 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.

[94] Florent Foucaud, Esther Galby, Liana Khazaliya, Shaohua Li, Fionn Mc Inerney, Roohani Sharma, and Prafullkumar Tale. Problems in np can admit double-exponential lower bounds when parameterized by treewidth or vertex cover, 2024.

[95] 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.

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

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

[98] Danny Hermelin, Moshe Kaspi, Christian Komusiewicz, and Barak Navon. Parameterized complexity of critical node cuts, 2015.

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.