University of Reading

Parameterized algorithms of fundamental NP-hard problems: a survey

Article

Published Version

Creative Commons: Attribution 4.0 (CC-BY)

Open Access

Li, W. ORCID: https://orcid.org/0000-0001-6121-588X, Ding, Y., Yang, Y., Sherratt, R. S. ORCID: https://orcid.org/0000-0001-7899-4445, Park, J. H. and Wang, J. (2020) Parameterized algorithms of fundamental NP-hard problems: a survey. Human-centric Computing and Information Sciences, 10 (1). 29. ISSN 2192-1962 doi: 10.1186/s13673-020-00226-w Available at https://centaur.reading.ac.uk/91587/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

To link to this article DOI: http://dx.doi.org/10.1186/s13673-020-00226-w

Publisher: Springer Berlin Heidelberg

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

University of Reading

CentAUR

Central Archive at the University of Reading

Reading's research outputs online

Li *et al. Hum. Cent. Comput. Inf. Sci.* (2020) 10:29
https://doi.org/10.1186/s13673-020-00226-w

Human-centric Computing and Information Sciences

RESEARCH Open Access

Parameterized algorithms of fundamental NP-hard problems: a survey

Wenjun Li[1], Yang Ding[1], Yongjie Yang[2], R. Simon Sherratt[3], Jong Hyuk Park[4] and Jin Wang[1*]

*Correspondence:
jinwang@csust.edu.cn
[1] Department of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China
Full list of author information is available at the end of the article

Abstract

Parameterized computation theory has developed rapidly over the last two decades. In theoretical computer science, it has attracted considerable attention for its theoretical value and significant guidance in many practical applications. We give an overview on parameterized algorithms for some fundamental NP-hard problems, including MaxSAT, Maximum Internal Spanning Trees, Maximum Internal Out-Branching, Planar (Connected) Dominating Set, Feedback Vertex Set, Hyperplane Cover, Vertex Cover, Packing and Matching problems. All of these problems have been widely applied in various areas, such as Internet of Things, Wireless Sensor Networks, Artificial Intelligence, Bioinformatics, Big Data, and so on. In this paper, we are focused on the algorithms' main idea and algorithmic techniques, and omit the details of them.

Keywords: NP-hard, Parameterized complexity, FPT, W[*t*]-hard, AI, IoT

Introduction

In the modern society, computers play an important role in solving optimization problems, most of which are shown to be computationally hard in theory. On the one hand, computers accelerate the developments of various practical fields, such as artificial intelligence [1–6], bioinformatics [7, 8], big data [9–13], smart grid [14, 15], wireless sensor networks [16–19], internet of things [20–25], vehicular network [26–29], cloud computing [30–34], computer vision [35–37], security [38–43] and so on. On the other hand, all these applications promote improvement of computer science and technology.

Over the last decade, many new and powerful techniques have attracted a great attention in computer science, and have been successfully used in solving optimal problems in a body of applications. There are a great deal of computational problems derived from practical applications which can be modeled as combinatorial optimization problems. Acquiring optimal solutions of them became a crucial task for relevant engineers or researchers. Unfortunately, some of them turned out to be computationally hard to be solved when the input size of these problems are considerably large. Due to the huge amount of information and data to be processed, the existing computers often fall into an awkward situation of "powerlessness" when solving many practical computing problems. In other words, it is unlikely for them to solve these problems in acceptable time.

Springer Open

© The Author(s) 2020. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

As to the problems hard to be solved, we do not only want to get exact solutions, but also need some systematic theories as guidance, so as to avoid detours in the process of finding solutions. The complexity theory provides the support to some extent. The guiding role of this theory lies in that it provides a systematic and strict theoretical framework for the classification of difficult problems. It gives the formal definition of NP-hard problems. Based on which, there is a conclusion that if a computational problem is NP-hard, unless P = NP, then finding an polynomial optimal algorithm for it is impossible. In a sense, if P ≠ NP, an NP-hard problem is actual un-computational when the size of instance is large.

In order to cope with NP-hard problems, certain effective algorithmic frameworks have been proposed, among which are approximation algorithms, heuristic algorithms, and randomized algorithms. A common disadvantage of these approaches is that they cannot always guarantee optimal solutions. Yet, this may be essentially the case because assuming P≠NP any exact algorithm for an NP-hard problem runs in exponential time. Exponential time algorithms are usually extremely time-consuming when the input size is considerably large, but this is not always the case for relatively small instances with the help of modern computers. Let us look at the following example.

Assume that Q is an NP-hard problem and A is an $O(n^c \cdot 2^n)$-time optimal algorithm of Q, where n is the input size of instances of Q and c is a constant. If $n = 100$, the running time $O(n^c \cdot 2^n)$ is usually un-acceptable for the current computing resources. For a decision problem Q' with a parameter k, the output of any algorithm for Q' is "yes" or "no". Assume there is an algorithm A' which can give the right answer in $O(n^{c'} \cdot 3^k)$. At first glance, the time complexity of A' is exponential, and the base is larger than that of A. But the exponent k in $O(n^{c'} \cdot 3^k)$ maybe far less than the exponent n in $O(n^c \cdot 2^n)$. In that situation ,the running time $O(n^{c'} \cdot 3^k)$ is acceptable. From the theory perspective, when given a fixed parameter as k, then the time complexity of A' is polynomial. Based on this observation, Downey and Fellows proposed the parameterized computation and complexity theory about two decades ago, which became a branch of computational complexity of theory rapidly. It focuses on algorithms and complexity of the NP-complete problems with different parameters, and is considered to be a new method dealing with NP-hard, or NP-complete problems.

In the paper, we will give an overview about the parameterized algorithms and complexity for fundamental NP-hard problems, accurately fixed-parameter tractable (FPT) problems, which are derived from the modern industrial applications. We will introduce the applications and definitions of them first. And then, we will illustrate the kernelization and/or FPT algorithms. More preciously, we show the main idea of the algorithms, the algorithmic techniques used and time complexity.

Preliminaries

In this part, we introduce several concepts about Parameterized Complexity first. And then, we show some fundamental or frequently-used parameterized algorithms techniques. We do not give the formal definitions of them, but present the main idea or the framework of them.

Parameterized complexity

Parameterized complexity deals with problems whose instances are 2-tuples (I, k) where I and k are respectively referred to as the main part and the parameter of the problem. The major goal of parameterized complexity is to investigate how the parameters influence the tractability of the problems, and classifies the problems into corresponding complexity classes. In a formal way, a parameterized problem Q^* is a language $Q^* \subseteq \sum^* \times \sum^*$, where $\sum$ is an alphabet. The complexity theory of parameterized computation classifies NP-complete problems with different hardness. If A^* can output right answer for each instance of Q^* in time $O(f(k) \cdot n^{O(1)})$ [44], where f is a monotone non-decreasing function of parameter k and n is the cardinality of the problem, then we say Q^* is a fixed-parameter tractable (shorted by FPT) problem, and the algorithm A^* is an FPT algorithm. Furthermore, if Q^* is an FPT problem, then it has a kernel with size bounded by $g(k)$, where $g(k)$ is a function only related to k. However, some of the NP-complete problems are not FPT. The theory of Parameterized Computation further define these problems as W[t], W[P] or XP problems (see the textbook of Downey and Fellows [44] for further details).

As a commonly recognized framework to handle NP-hard problems, parameterized complexity has been successfully used in a wide range of areas (see, e.g., [45–51]). Many FPT-algorithms have been derived for a large number of classical problems including, for example, Feedback Vertex Set, Vertex Cover, Hyperplane Cover, Maximum Internal Spanning Tree, Planer Dominating Set, Edge Dominating Set, etc. On the other hand, some NP-complete problems turned out to be W[t]-hard, $t \geq 1$. For instance, the fundamental parameterized NP-complete problems Weighted 3-SAT, Clique, and Independent Set are W[1]-hard, and Dominating Set, Weighted CNF-SAT and Set Cover are W[2]-hard.

Algorithmic techniques of parameterized algorithms

In this subsection, we show some algorithmic techniques for FPT algorithms and kernelization. It should be remarked that kernelization, usually consisting of a set of reduction rules, is a pre-procedure to solve an FPT problem, which not only can decrease the size of instances, but also can make the reduced instances holding some special properties. These special properties are usually to be the key points of developing efficient FPT algorithms. For this reason, kernelization can be seen as one kind of algorithmic technique of parameterized algorithm.

Kernelization:

- *Crown decomposition*: Crown decomposition has been proposed in the early days of birth of parameterized computation theory, and it has been successfully applied in the kernelization of the Vertex Cover Problem. Subsequently, this technology is also used in the kernelization of many other problems, such as P_2 Packing, Hitting Set, Co-path\cycle Packing, and so on. A crown in a graph $G = (V, E)$ is a structure holding special property, which can be defined as a triple (I, H, R), where I, H, R are three disjoints set, $V = I \cup H \cup R$, I is a nonempty independent set, H is the neighbors set of I and $R = V \backslash I \backslash H$. Furthermore, there is a matching M between

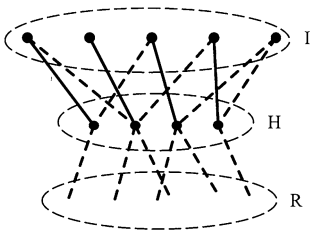

Fig. 1 A crown with 4 head vertices

I and H such that $|M| = |H|$. The Fig. 1 shows a crown, where the cardinality of H is 4. Usually, if a crown is found, deletion or replacement operations could be done in kernelization for many problems. Therefore, how to find the crowns is an important step during the kernelization.

- *Linear programming*: Linear programming (LP) is a powerful tool to solve optimal computation problems. In parameterized computation, we usually consider combinatorial problems. Fortunately, many of them can be modeled by Integer Linear Programming (ILP), in the instances of which, variables are with integer value, constraints are linear inequalities, and cost function is linear. But these problems can not be handled by Linear Programming, where the value of variables are not restricted to integer. Since finding the optimal solution of an ILP instance is NP-hard and that of a LP instance is polynomial-time solvable, transforming an ILP instance to a LP instance is a reasonable way in kernelization. This method has been successfully applied in the kernelization for Vertex Cover.
- *Local reduction*: Local reduction is the most widely used kernelization technique. The local reduction rules are based on observation of local structure properties of the problem, which are different as to different problems. The local reduction technique has two characteristics: The one is the local property of the rules. That is, the object considered by the rule is not the whole input. The other one is the particularity of the rules. Due to the special property of the problem, there is no universal reduction rule. And the rules are developed according to specific problems.

Algorithmic techniques of FPT algorithms:

- *Bounded Search Trees*: This method is originated from the general idea of backtracking and is one of most widely used techniques for parameterized algorithms design. The main idea of this technique is to enumerate all the possible cases for the problem, so that it makes a series of possible decisions. For each decision, there will be a corresponding sub-problem, in which the parameter is decreased by some constant. Such an procedure will be repeated until the parameter is equal to or smaller than 0. Furthermore, each subproblems are solved one by one. The key point of Bounded Search Trees method is how to make the instances with some property so that some of the branches can be omitted. For this reason, applying some reduction rules before/or during branching are usually an effective way to get an FPT algorithm, by Bounded Search Trees method, with low time complexity.

- *Iterative compression*: Normally, iterative compression is novel parameterized algorithmic technique. It is used to design FPT algorithms for minimization problems, the task of which is to find the minimum solution. The algorithms based on this technique employing the "compression" iteratively. In the procedure of compression, the algorithm tries to find a smaller solution or outputs that the solution is exactly the smallest one. For this technique, the greatest feature is that the compression procedure can take the advantage of the intermediate solutions.
- *Randomized method*: Randomized method has been applied to the FPT algorithms for Feedback Vertex Set, Longest paths, P_2 Packing and d-clustering problems. Color Coding is a non-trivial randomized method. Its main idea is coloring vertices or edges of graph with a set of colors randomly. And the color is a mark that distinguish different vertices or edges.
- *Treewidth*: Treewidth is a relative new technique to design parameterized algorithms. Treewidth of a graph measures how much it is close to a tree. For a graph of a problem, if the treewidth is small, then the problem will be effectively solved by constructing a tree decomposition and using dynamic programming. If the tree width is large, then appropriately handling the obstacle structure will be a common way. This technique has shown great advantages in designing subexponential algorithm for some problems on planar graphs.
- *Others*: There are many other FPT algorithmic techniques, including Dynamic Programming, Linear Programming and Sun flower Lemma, etc.

Parameterized algorithms for FPT problems

In this section, we will give a survey on existed FPT algorithms and kernelization for some fundamental NP-complete problems in industrial applications, including Satisfiability, Spanning Trees and Out-branching, Planar Dominating Set, Feedback Vertex Set, Covering, Matching and Packing (see Table 1 for details). In this paper, we just consider the parameterized version of these optimal problems. For the convenience, the word "parameterized" is omitted where no ambiguity exist. And as to the running time $O(f(k)n^{O(1)})$ for some algorithm, we use $O^*(f(k))$ to replace it in the whole paper.

Satisfiability problems

Satisfiability (SAT) problem is a fundamental NP-hard problem and has wide applications in artificial intelligence, combinatorial optimization, expert systems, and database systems [52–56]. The formal definition of SAT is: Given a CNF formula F, decide if an assignment that satisfy all the clause in F exists. A CNF formula consists of some clauses, which are joined by AND. In each clause, there some literals joined by OR. And the literal could be positive (the same name as the corresponding variable) or negative (negation of the positive literal). Equation (1) shows a CNF formula that consists of six variables $(g_1, g_2, \ldots, g_6)$ and three clauses. It is known that the SAT problem is NP-hard even in many special cases.

$$F = (g_1 \vee g_2 \vee g_3) \wedge (\bar{g_1} \vee g_4) \wedge (g_1 \vee g_5 \vee g_6) \wedge (\bar{g_2} \vee g_5 \vee \bar{g_6}) \tag{1}$$

Table 1 A summary of parameterized algorithms for the problems discussed in this paper

Categories	Problems	Kernelization	FPT algorithms
Satisfiability	MaxSat		√
	$(n, 3)$-MaxSAT		√
Spanning Trees and Out-Branching	MIST	√	√
	MIOB		√
Dominating Set	Planar DS	√	√
	Planar CDS	√	√
	EDS	√	√
Feedback Vertex Set	FVS	√	√
	Planar FVS	√	
Covering	Vertex Cover	√	√
	Hyperplane cover	√	√
Packing and matching	P_2 Packing	√	√
	r-D Maching		√
	WrDM		√
	r-Set packing		√
	WrSP		√
	EDTP	√	

Table 2 Research history of FPT algorithms for MaxSAT

Year	Time complexity	References
1997	$O^*(1.618^k)$	Raman and Mahajan [57]
1999	$O^*(1.3995^k)$	Niedermeier and Rossmanith [60]
1999	$O^*(1.380278^k)$	Bansal et al. [61]
2004	$O^*(1.370^k)$	Chen et al. [62]
2012	$O^*(1.358^k)$	Ivan et al. [63]

When the target of the SAT problem is not to find an assignment satisfy all the clauses in F, but to find an assignment satisfying as many clauses as possible. The SAT problem becomes the MaxSAT problem. When the literal number in each clause is restricted to two, the MaxSAT problem is shorted by Max-2-SAT. This problem is also NP-hard even if their at most three clauses contain the same variable in F [57]. It has been proved that the (s, t)-MaxSAT problem is polynomial-time solvable when $t = 2$ [58]. The parameterized version of MaxSAT is as follows.

Definition 1 *MaxSAT*: Given a CNF formula F and an integer parameter k, is there an assignment which satisfies at least k clauses in F?

For the parameterized MaxSAT problem, there is a long line of researches (see Table 2 for details). Almost all the extant FPT algorithms consist of two parts. The one is the part that contains some reduction rules which can decrease the size of the formula and make the given CNF formula having some special properties. For example, if there is a $(q, 1)$-literal g in F, then the $q + 1$ clauses containing literal g or $\bar{g}$ can be reduce to q clauses, where a (n_1, n_2)-literal g is a literal such that g and $\bar{g}$ occurs n_1 and n_2 times respectively, and the details of the rule are omitted here. After this rule

applied, formula F can be reduced to F' as the Eq. 2. In the following, we lists several simple reduction rules.

For a variable g in formula F,

- If there only positive literals or negative literals of g in F, then assign $g = 1$ or $g = 0$.
- If there exists a clause containing literal g and literal $\bar{g}$, then delete this clause and $k = k - 1$.
- If there is a clause containing two or more literal x, then remove all the literals g from this clause except one.
- If the number of clause $\{\bar{g}\}$ in F is no less than that of clauses containing literal g, then assign $g = 0$.
- If g is a $(q, 1)$-literal and the clauses containing g are (gC_1), gC_2, ..., gC_q, $\bar{g}C'$, then replace these $q + 1$ clauses by the q clauses C_1C', C_2C', ..., C_qC', where C_i is the combination of several literals by 'OR', $1 \leq i \leq q$.

The other one is the branching part, in which the algorithms branch on variables whose degree are not too small, where the degree of some variable is the number of corresponding literals (including positive and negative) in the given formula. For example, the degree of variable g_1 in the formula F is 3, since the literal g_1 appears twice and the literal $\bar{g_1}$ appears once in F. Because of the reduction rules mentioned above, there exist no variable with degree smaller than 4.

$$F' = (g_2 \vee g_3 \vee g_4) \wedge (g_4 \vee g_5 \vee g_6) \wedge (\bar{g_2} \vee g_5 \vee \bar{g_6}) \tag{2}$$

Raman and Mahajan considered parameterized algorithm for the MaxSAT problem [57]. Their algorithm is depend on the framework proposed by Downey and Fellows [59]. Firstly, the clauses are divided into two kinds of clauses: (1) long clauses, each clause contains k or more literals; and (2) short clauses. For the given CNF formula F, if there are no less than k long clauses, then output 'Yes'. Otherwise, after some trivial preprocess, branch on some variable whose corresponding positive and negative literal appears in the formula at least once. The time complexity of this simple algorithm based on Bounded Search Trees method is $O^*(2^k)$. Through a minor modification, they developed an algorithm in time $O^*(1.618^k)$.

For obtaining a more efficient algorithm for the MaxSAT problem, Niedermeier and Rossmanith [60] used two kinds of rules: transformation rule and splitting rule. The former one is the rule replacing a formula by another formula, and the later is replacing a formula by several other formulas. After these rules applied, the Davis-Putnam-procedure is used to get the solution of the problem. The algorithm's running time is $O^*(1.3995^k)$.

Later, Bansal et al. proposed an algorithm for MaxSAT in time $O^*(1.380278^k)$ [61]. By using some simple rules, they got the solution of MaxSAT problem equivalent to the original formula, such as elimination of (1, 1)-literals and replacement of almost common clauses. At the same time, it combines the rules of some branches in davisputnam-type branching algorithm, according to the above two methods, two general algorithms are proposed.

Chen et al. by using the created low literal subroutine to enforce the invariant of the formula containing text that rarely appears, thus making the overall algorithm work better. In addition, they introduced a branch rule, which uniformly and systematically captures several branches and makes them more efficient, they greatly improved the bound $O^*(1.370^k)$ proved [62].

Ten years later, Ivan et al. presented an algorithm was based on standard splitting techniques with running time of $O^*(1.358^k)$ to examine if at least k clauses in an input formula are satisfied parameterized MaxSAT. They propose a easy way that improves the upper bound for MaxSAT [63].

(n, 3)-MaxSAT is a restricted version of MaxSAT in which each variable occurs no more than three times. The original bound of the (n, 3)-MaxSAT is the result $O^*(1.3247^k)$ [62]. For the first time, Ivan and Golovnev [63] treated the problem as a separate problem and got a bound $O^*(1.2721^k)$. New simplification rules are proposed on some traditional rules and introduced Create-Low-Literal subroutine. On the basis of the former, they proposed some new simplification and branching rules, and simplification rules can decrease k at least by one. Simplifying an example of restricted Max-SAT to an example of minimum set cover. Direct branching can give a good number of branching for a variable of high degree immediately. The main bottleneck is that the formula only contains variables of low degree. Branch processing is carried out through the "resolution-like" rule.

Based on two new resolution rules for the (n, 3)-MaxSAT problem, Xu et al. got an improved upper bound $O^*(1.194^k)$ of this problem [64]. After exhaustively applied of there rules, the given CNF formula will become linear, which is the key point of their contribution. And this property makes their branch and search strategy more efficient. As a consequence, for the MaxSAT problem, it significantly improved running times bound. Later, Li et al., through introducing some new rules, proposed an improved FPT algorithm bounded by $O^*(1.175^k)$ [65].

In order to obtain a lower upper bound for the (n, 3)-MaxSAT problem, Belova et al. proposed a very simple algorithm. And for the running time analysis, no case analysis is needed. They achieved an algorithm with running time of $O^*(1.175^k)$. Furthermore, they found that the upper bound for (n, 3)-MaxSAT in terms of k could be improved if that of the problem in terms of n is improved.

Spanning Trees and Out-Branching problems

Finding Spanning Trees in a given graph has widely applications in communication network design. The most fundamental one aims to construct a spanning tree with minimum edge-weight for a given edge-weighted undirected problem, which can find a solution by the Prim- or Kruskal-algorithm in polynomial time. But there are many other NP-hard combinatorial optimization problems about finding spanning, which are NP-hard. Finding spanning paths is also a classical problem in computational geometry, which is motivated by design of VLSI, the movement of heavy machinery and other applications. In this part, we will focused on Maximum Internal Spanning Tree (MIST) and Maximum Internal Out-Branching (MIOB).

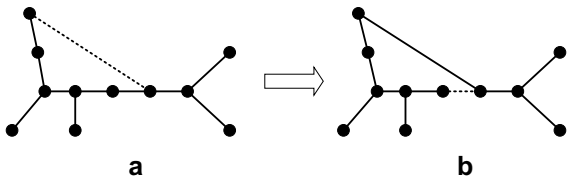

Fig. 2 An example of swapping operation exchanging one edge

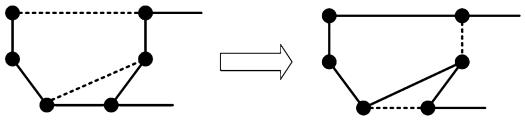

Fig. 3 An example of swapping operation exchanging 3 edges

Definition 2 *Maximum Internal Spanning Tree*: Given a simple undirected connected graph G and an integer parameter k, is there a spanning tree of G with at least k internal nodes?

For this problem, Prieto et al. [66] developed an $O(k^2)$-vertices kernelization algorithm and an $O^*(k^{2.5k})$-time FPT algorithm. They used a simplified set of rules to modify any spanning tree of the graph to a spanning tree so that there are no edge between any two leaves in graph. This rule either produces an independent set with large size, or generates a tree with many internal vertices. For the former case, the graph is likely to contain a crown structure, and the Crown Decomposition will be applied. For the later case, a small kernel for the problem could be obtained directly.

Fernau et al. obtained a branching algorithm with time complexity of $O^*(k^{2.1364k})$ for the problem in cubic graphs [67]. This paper focuses on the case of degree-bounded. The main innovation is that they analyze the algorithm by Measure and Conquer method.

Fomin et al. based on a min-max characterization of hypergraphs containing hypertrees in [68], firstly constructed a spanning tree by depth-first search, and then proposed a kernelization algorithm with $3k$-vertices linear kernel. Based on the kernel, an $O^*(8^k)$ -time FPT algorithm could be obtained.

On the basis of the algorithm proposed by Fomin et al. [68], Li et al. [69] changed the structure of spanning tree by using deeper local search method. They use edge-swapping operation deeply, which makes that some internal nodes in spanning tree T were not neighbors of leaf nodes in graph G. Edge-swapping operation is to replace some edge in spanning tree T by some edge in G but not in T. The aim of this operation is to make the constructed spanning tree with more internal nodes. Figure 2 shows an example of edge-swapping operation, which exchanges one edge in an edge-swapping operation. After this operation executed, the number of internal nodes is increased by one. The authors' main contribution is to exchange five edges in an edge-swapping operation. Figure 3 shows an example of edge-swapping operation, which exchanges threes edges in an edge-swapping operation. Furthermore, they provide an effective method to analyze the structure property of the spanning tree. Their kernelization algorithm can construct a spanning tree that there a certain number of internal nodes do not adjacent to the leaves.

Due to this structure property, a kernelization algorithm with a size of $2k$ was obtained finally, which results in an $O^*(4^k)$-time algorithm directly.

Now we show the parameterized algorithms for MIOB, a generalization of MIST. The following definition is for the parameterized version of MIOB.

Definition 3 *Maximum Internal Out-Branching*: Given a directed graph G and a non-negative integer k, is there an out-branching with at least k internal nodes?

Gutin et al. [70] proposed a kernelization algorithm for the problem which results in a kernel of size $O(k^2)$. This kernel directly leads to a trivial $O^*(2^{O(k \log k)})$-time FPT-algorithm. Cohen et al. [71] investigated the problem restricted symmetric digraphs. In particular, based on complex color coding and an unbalanced partitioning technique, a random algorithm with time complexity $O^*(49.4^k)$ was obtained in [71]. Along with this randomized algorithm, a deterministic algorithm running in $O^*(55.8^k)$ time was also proposed in [71]. Later, Fomin et al. [72] used the Sharp Separation Theorem to craft an FPT-algorithm running in $O^*(16^{k+O(k)})$-time. This result has been further improved to $O^*(4^k)$ by Zehavi et al. [73].

Dominating Set problems

(Connected) Minimum Dominating Set problem has wide applications in a variety of fields, such as resource allocation, power network, wireless sensor network and so on. A vertex u dominates all the vertices in $N[u]$, where $N[u]$ consists of all the neighbors of u and u itself. A dominating set of a graph G is a subset of vertices such that every vertex in the graph is dominated by at least one vertex in the subset. A connected dominating set is a dominating set which induces a connected subgraph. The (CONNECTED) DOMINATING SET problem is formally defined as follows.

Definition 4 *(Connected) Dominating Set*: Given a graph G and an integer parameter k, is there a (connected) dominating set of size at most k in G?

In general, the (CONNECTED) DOMINATING SET problem is W[2]-hard. However, the problem becomes FPT when restricted to certain special graph classes such as planar graphs. In this section, we discuss this special case and use PDS to denote it.

The FPT algorithms of PDS have attracted great attention (see Table 3). And various methods or techniques have been used to the algorithms' designing and analysis.

In a graph, a vertex could dominate itself and its neighbors. Therefore, for any vertex v in G, if it has been chosen to be one vertex in an optimal solution, then itself and all of its neighbors will be dominated. In this case, the vertex v could be removed from the given graph. However, the neighbors will be not allowed. Since each neighbor of v is a candidate for an optimal dominating set. Based on this simple observation, Alber et al. [74] consider a more general problem, named Annotated Dominating Set. In this problem, the vertices that are in the solution are colored by white, and the other vertices are colored by black. More specifically, the black vertices are un-dominated, while the white vertices are dominated. The task of Annotated Dominating Set is to seek out at most k vertices dominating all the black vertices. They proposed some reduction rules firstly

Table 3 Research history of FPT algorithms for PDS

Year	Time complexity	References
2002	$O^*(8^k)$	Alber et al. [74]
2002	$O^*(2^{27\sqrt{k}})$	Kanj et al. [75]
2004	$O^*(2^{15.13\sqrt{k}})$	Alber et al. [76]
2005	$O^*(4^{6\sqrt{34}\sqrt{k}})$	Alber et al. [77]

and then used Bounded Search Trees method to deal with the reduced graph. The reduction rules make the reduced given graph holding some special properties. For example, there is no edge between white vertices and no pendant white vertex in reduced graph. Of course, there are some other relatively more complicated properties. Through deeply analysis of all these properties and combining with the classical Euler formula for planar graphs, the authors got a critical theorem that there is a black vertex with degree no larger than 7. Based on this theorem, Bounded Search Trees method can results in a trivial $O^*(8^k)$-time FPT algorithms for the problem. Since MDS is a special case of the Annotated Dominating Set problem (no white vertices in the given graph), the above mentioned algorithm can solve MDS directly.

Later, Alber et al. [77] proposed the first sublinear exponential FPT algorithm for PDS, which enhanced previous algorithms dramatically. This great improvement is due to two facts: (1) if a planar graph G has a dominating set with size of k, then the tree-width of G is no more than $6\sqrt{34}\sqrt{k}+8$; and (2) if a w-width tree decomposition of G is constructed, then a minimum dominating set can be found in time $O(4^w)$. The algorithm constructs a treewidth decomposition for the given graph firstly, and then find a minimum dominating set on the tree-width decomposition. Since the treewidth decomposition can be constructed in time $O(\sqrt{k}n)$, the total time complexity of the whole algorithm is $O^*(4^{6\sqrt{34}\sqrt{k}})$.

Following the work in [77], Kanj et al. [75] improved the FPT algorithm depending on some novel observations. Firstly, they found that Baker's algorithm, through modification, not only can be used to solve the PDS problem but also can deal with the Planar Red-Black-White Dominating Set, which is a variety of PDS. Secondly, they can find the separator of large components with size of $15k$, which is much smaller than the separator with size of $51k$ established in [77]. Combining the two new rules with the divide-and-conquer approach immediately led to a significant improved FPT algorithm with time $O^*(2^{27\sqrt{k}})$.

Instead of using the tree decomposition process mentioned above, Fomin and Thilikos in [77] introduced branch decomposition for PDS. Similar to relationship between the treewidth of tree decomposition and the domination number of a planar graph, there is also a relationship between branchwidth and the size of dominating set of a planar graph. More specifically, when a planar graph G admits a k-dominating set, the branch-width will be no more than $6.37\sqrt{k}$. Furthermore, constructing such a branch decomposition is polynomial. Combined with the linear kernel ($335k$) of the problem proposed by Alber et al. [76], they get an $O^*(2^{15.13\sqrt{k}})$-time FPT algorithm.

For the kernelization of the PDS problem, Alber et al. [76] gave the first kernelization algorithm with kernel size of $335k$. The reduction rules in the kernelization procedure

are respecting to local structures of the given graph. Even though there is no global reduction rule, they can get a linear kernel for the problem because the given graph is planar and the planarity make the region decomposition becoming a useful tool during the kernel size analysis. Later, Chen et al. [78] introduced some improved and new reduction rules for the kernelization problem. In order to get some new and useful structure properties, they color the vertices of the graph. Finally, they improved the kernel size to $67k$. For the kernelization of the PCDS problem, Lokshtanov et al. [79] provided an algorithm with kernel size of ck, where $c = 3968187$. Although the constant c is huge, it is a linear kernel for PCDS. This result is based on method reduce-or-refine, which consists of two operations: reduce and refine. The kernelization algorithm, firstly, computes a region decomposition from an approximated connected dominating set. And then, it checks the structure of each region. If there are two many copies of a particular structure in the region, then removing some vertex is safe for the problem. It is the reduce operation, which can reduce the graph. Otherwise, it turns to pay attention to a smaller region which contains many vertices and no particular structure mentioned above. This is the refine operation. The process is repeated until each region looks simple and is easy to deal with. Later, based on some new observations, Gu et al. [80] proposed a set of similar reduction rules for connected dominating sets. Combining with refined kernel analysis, they improved the kernel to $413k$. Later, using similar method, Luo et al. [81] obtained a smaller kernel of $130k$ by new reductions and careful kernel analysis.

There some researchers are interested in kernelization of the Connected Dominating Set (CDS) problem in special graphs. For instance, Misra et al. [82] considered the graphs with no little cycles, and proved that, when the given graph contains cycles of length smaller than 7, then there is no polynomial kernel for CDS (assuming the Polynomial Hierarchy does not collapse to the third level). Furthermore, they showed a kernelization algorithm for CDS on graphs with no cycle of length smaller than 7. They colored the vertices by red, white or black, developed some simple reduction rules, and got an $O(k^3)$-vertex kernel finally. Based on refined kernel analysis, Li et al. [83] simplified the algorithm and obtained a small kernel ($O(k^2)$) for this problem.

The Dominating Set problems mentioned above are respected to vertex-domination. There is another kind of dominating problems: Edge dominating Set, which is respected to the edge-domination.

Definition 5 *Edge Dominating Set* (*EDS*): Given a graph $G = (V, E)$ and an integer parameter k, is there a subset $E^* \subseteq E$ of at most k edges such that $\{u, v\} \cap V(E^*) \neq \emptyset$ for all edges $\{u, v\} \in E$.

From the definition of EDS, we know that if $E^* \subseteq E$ is an edge dominating set, then $V(E^*)$ is a Vertex Cover of G. Furthermore, if $|E^*| \leq k$, then there exists a vertex cover of G with size no more than $2k$. Thus, enumerating all the minimal vertex covers is reasonable in the algorithms for EDS and Partial Vertex Cover is introduced. Fernau et al. [84] found that if any vertex in $G \backslash C$ is with degree no more than 1, where C is vertex subset of some vertex cover, then finding an edge dominating set of G is polynomial. Thus, it is reasonable for the algorithm to branch on the vertices in $G \backslash C$ with degree no less than 2. Based on these two facts, Fernau proposed an $O^*(2.6181^k)$

-time FPT algorithm for EDS. In 2009, Fomin et al. [68] gave a similar algorithm, but picked the vertices with degree larger than 2 to branch. And then, they used path decomposition to handle the rest vertices in $G \backslash C$. Finally, they got an $O^*(2.4181^k)$-time FPT algorithm for EDS. In 2011, Xiao et al. developed an $O^*(2.3147^k)$ due to a technique which can deal with remaining graphs with maximum degree 3 in [85]. Furthermore, they showed EDS has a kernel with $2k^2 + 2k$ vertices and $O(k^3)$ edges, which improved the previous results [84] [86]. Through refinement of kernelization algorithm in [85] and a bit different analysis, Hagerup [87] got a smaller kernel $(\max(1/2k^2 + 7/2k, 6k))$ for this problem.

Feedback Vertex Set problem

Feedback Vertex Set (FVS) problem is crucial in dealing with deadlock in operating systems. Each deadlock situation has a corresponding directional loop in the graph derived from the operating system. The solution to deadlock is to abort some blocked processes, and a feedback vertex set corresponds to the processes that should be aborted [88]. In addition, the FVS problem is often used in designing VLSI chip [89].

Definition 6 *Feedback Vertex Set (FVS)*: Given a graph $G = (V, E)$ and an integer parameter k, can we remove at most k vertices so that the resulting graph does not contain any cycles?

Raman et al. proposed an $O^*(max\{12^k, (4 \log k)^k\})$ algorithm for FVS problem [90], improved the previous $O^*((2k + 1)^k)$-time algorithm. They proved that, for an undirected n-vertices graph with degree no less than three, if there is a feedback vertex set with size no more than $c\sqrt{n}$, then the length of any cycle in the graph is at most 12 (c is a constant).

Kanj et al. proposed an $O^*((2 \log k + 2 \log \log k + 18)^k)$ [91]-time algorithm for FVS. The algorithm applies an operation called short-cut repeatedly on the graph G, which led to no vertex with degree 1 and the degree-2 vertices only exist in the cycle with length of 3 in the graph. On the reduced graph, a quasi-shortest cycle C with length l can be found in time $O(n^2)$, which is close to the shortest cycle. When the length l of C satisfied certain conditions, then branch on C and update k, F and G accordingly. Later, Dehne et al. [92] Proposed an $O^*(2^{O(k)})$-time algorithm for FVS, which uses iterative compression technique. They used some reduction rules to simplify problem instances. These rules are applied so that all the vertices in the graph have degree at least three.

Chen et al. proposed two algorithms with $O^*(5^k)$ respectively for the FVS problem on un-weighted graph [93]. Their method also used iterative compression technology, which differs from other papers in that they proposed a new recursive process and introduced an subproblem (Disjoint FVS, see Fig. 4). They found a implicit parameter (the number of trees in the solution), which is the key point that could improve the previous algorithms. Cao et al. [94] provided an $O^*(3.83^k)$-time algorithm for FVS. The framework of it is almost the same as that of [93]. But, they found that Disjoint FVS is polynomial-time solvable when the degree of each vertex in the graph is no more than 3.

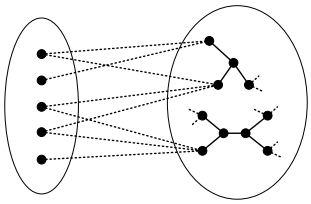

Fig. 4 An example of Disjoint FVS

Kociumaka et al. proposed a new deterministic $O^*(3.619^k)$-time algorithm for FVS [95]. The main idea of their algorithm is similar to that of the algorithm proposed by Cao et al. Their contribution for the improvement of the algorithm is due to a new and simple reduction rule. Furthermore, they developed a set of more complicated branching rules, which can achieve an algorithm with upper bound $O^*(3.592^k)$.

Recently, Cao designed smart $O^*(8^k)$-time algorithm for FVS [96]. His algorithm is a naive greedy branching algorithm, which always branches on an undecided vertex with the largest degree. The algorithm analysis is based on the following two facts: the one is that none of the vertices' degree in the algorithm increases; the other is that there is no vertex in the graph with degree ≤ 2.

For kernelization of the FVS problem, Burrage et al. proposed a kernel of size $O(k^{11})$ for FVS on undirected graphs [97]. Their algorithm calculates the specific structure map of problem instances in polynomial time and defines some polynomial-time data reduction rules for these specific structures. Bodlaender et al. considered the FVS problem on un-weighted undirected graphs and gave a kernel of size $O(k^3)$ [98]. Their algorithm is divided into two steps. In the first step, they used existed approximation algorithms (with ratio of 2) to find a feedback vertex set. The second step consists of several simple reduction rules. Based on Sun Flower Lemma, Thomasse [99] proposed a simple and novel $4k^2$-vertex kernelization algorithm for FVS.

For FVS on planar graphs, Bodlaender et al. proposed kernelization algorithm with linear kernel with size of $112k$ [100]. They proposed a novel notion, the bases of induced subgraphs, and used it to testify if the reduction rules they used is right or not. It is worthy to mention that all the rules in kernelization are local, and which are applied only on specific subgraphs. Abu-khzam et al. improved this kernel to $97k$ [101]. Later, Xiao et al. obtained a kernel with size of $29k$ for the planar FVS problem [102]. The improvement is mainly due to a different kernel analysis based on the properties of planar graphs. Bonamy et al. proposed a kernel of size $14k$ for FVS on planar graphs [103]. They proposed some new reduction rules to simplify the problem instances. The most important contribution they made was the application of region decomposition technology in kernel size analysis. Later, they revised the kernel size to $13k$ in the journal version [104].

Covering problems

Covering problems are generally such problems where we aim to find a subset of given elements to hit all or a partial of some specific structures. These problems have significant applications in the fields of process analysis, computational biology, network

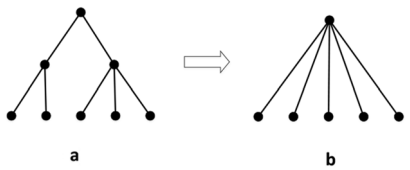

Fig. 5 An example of local reduction for Vertex Cover

Table 4 Research history of FPT algorithms for Vertex Cover

Year	Time complexity	References
1993	$O^*(2^k)$	Buss et al. [105]
1998	$O^*(1.324718^k)$	Balasubramanian et al. [106]
1999	$O^*(1.31951^k)$	Downey et al. [107]
1999	$O^*(1.129175^k)$	Niedermeier et al. [108]
2001	$O^*(1.2852^k)$	Chen et al. [109]
2006	$O^*(1.2738^k)$	Chen et al. [110]

address selection and circuit design. In this part, we will overview the parameterized algorithms for Vertex Cover and Hyperplane Cover. A vertex cover in a graph is a subset of vertices such that every edge in the graph has at least one of its endpoints in the subset.

Definition 7 *Vertex Cover*: Given a graph $G = (V, E)$ and an integer parameter k, does G have a vertex cover of size at most k?

Buss proposed a trivial $O^*(2^k)$-time algorithms in [105]. Later, almost all the FPT algorithms for Vertex Cover are based on some reduction rules and Bounded Search Trees method. Figure 5 shows a reduction rule, which handles the degree-2 vertex in the case that the two neighbors of it are not adjacent. There are a lot of improvements focusing on improving the exponent 2^k (see Table 4 of the research history of parameterized algorithms for Vertex Cover).

Balasubramanian et al., depending on Bounded Search Trees method, gave an $O^*(1.324718^k)$-time algorithm [106]. Later, Downey et al. slightly improved it to $O^*(1.31951^k)$ [107]. Niedermeier et al. developed an $O^*(1.129175^k)$-time FPT algorithm for Vertex Cover. They proposed a general technique for polynomial factors in which the complexity of the algorithm above can be removed from the dominating term. Niedermeier and Rossmanith got a new upper bound of $O^*(1.29175^k)$ [108].

Chen et al. [109], based on several simple new reduction rules, proposed an $O^*(1.2852^k)$-time algorithm for Vertex Cover. They used kernelization as a preprocesses for the FPT algorithm. According to a theorem of Nemhauser and Trotter [111], if some instance admits a vertex cover with size no more than k, then there are at most only $2k$ vertices needed to be concerned. The second is a simple new technique for dealing with vertices with degree of 2, which avoids complicated case-by-case analysis. At the same time, they developed a new technology of "iteration branch" to maintain the special effective of the graph.

Later, Chen et al. [110] gave an $O^*(1.2738^k)$-time algorithm for vertex cover [110]. Different from the previous way, such as case-by-case branching rules, worst-case-scenario assumption, they introduced some new techniques, including vertex folding, amortized analysis. Based on which, their algorithms are simpler and more uniform. They briefly describe several new and previous technologies, such as general folding, local amortized analysis and tuples. They developed two types of graph operations: a generalization of folding operation and a special case of construction operation proposed by Ebengger et al. [112] These two operations allow them to delete some simple structures from the graph without branching.

For kernelization of vertex cover, there are two algorithms with kernel size of $2k$. They used matching and linear programming respectively. Faisal et al. [113] proposed an algorithm by Crown Decomposition. But the kernel size of it is $3k$. Through deep analysis of the crown decomposition for vertex cover, Li et al. improved the existing crown decomposition technique. The central idea of this method is, using deeper local search, try to find and delete all the crowns from the given graph. After the improved rule applied, the graph becomes a disjoint set of odd cycles and a subgraph having a perfect matching. It is easy to verify that all the crown structures have been removed from the given graph and a kernel with size of $2k$ exists [114].

Definition 8 *Hyperplane Cover*: Given n points in d-dimensional Euclidean space and an integer k, does there exist k hyperplanes covering all the points.

Langerman and Morin [115] proposed an $O^*(k^{dk+d})$-time algorithm by Bounded Search Trees method. Later, based on simple observation, Wang et al. [116] provided a similar algorithm in $O^*(k^{(d-1)k}/1.3^k)$ time. The improvement is mainly due to an interesting time complexity analysis. When $d = 2$, Hyperplane Cover problem is exactly the Line Cover problem. For the Line Cover problem, besides the algorithms mention above, Langerman and Morin proposed a $O^*((k/2.2)^{2k})$-time algorithm. Recently, by taking Szemeredi-Trotter-type incidence bounds and non-parameterized $O(2^n)$-algorithm for Curve Cover (a generalization of Line Cover), Afshani et al. [117] developed an FPT algorithm for Hyperplane Cover (in R^3) in time $O^*((Ck^2/logk^{1/5})^k)$ and an $O^*((Ck/logk)^k)$-time algorithm for Line Cover.

Packing and Matching problems

Packing and matching problems are classical problems in combinatorial optimization. Using parameterized algorithm to improve the results not only has great theoretical significance, but also has a significant impact on the actual industry, such as code optimization and scheduling. In practical applications, matching and packing problems are also classified, including 3-Set Packing, 3-D Matching, P_2-Packing and Edge Disjoint Triangle Packing.

Definition 9 P_2 *Packing*: Given a simple undirected graph G and an integer k, does there exists k P_2s in G (a P_2 is a path with length 2) such that no two P_2s share a common vertex?

Prieto and Sloper [118] used a set of reduction rules based on crown decomposition to obtain a kernel with size of $15k$ for the P_2-Packing problem. Later, some researcher followed this work, and used local search to find larger P_2-Packing (see Fig. 6).

Wang et al. [119] proposed two novel concepts: double crown decomposition and fat crown decomposition (see Figs. 7 and 8). Based on these two decompositions, they got a kernel for P_2 Packing with size of $7k$. In addition, on the basis of this new kernelization algorithm, they obtained an FPT algorithm in time $O^*(17.66^k)$. Through deep analysis of the relationship between the vertices not in P_2-packing and the vertices in P_2-packing, they realized that the number of vertices not in P_2-packing was reduced while the size of P_2-packing increased. Chen et al. [78] used the concept of combinatorial duality to transform the kernel of the packing problem and the covering problem. They proposed a kernelization algorithm for P_2-packing with kernel size of $6k$. Their algorithm first found a non-expandable P_2-packing and then optimized it according to the five rules they gave.

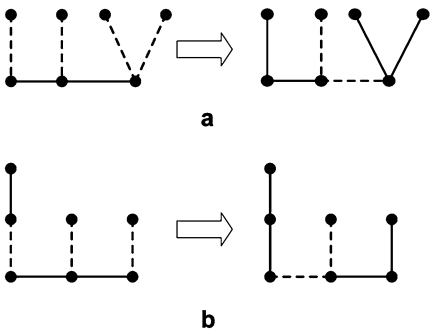

Fig. 6 Two examples of finding larger P_2-packing by local search

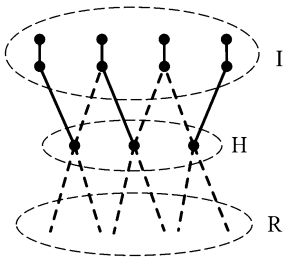

Fig. 7 An example of fat crown with 3 head vertices

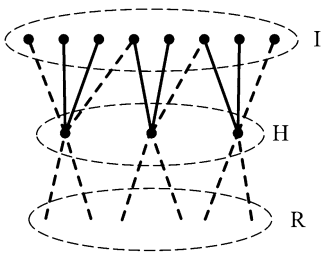

Fig. 8 An example of double crown with 3 head vertices

Li et al. [120] considered the kernelization algorithm for the P_2-Packing problem and obtained a $5k$ kernel by Deeper Local Search method. Figure 9 shows an example of the process of deeper local search. Based on previous algorithms, they proposed a more systematic and comprehensive local search method, which is a powerful tool to find more crowns.

Definition 10 *r-D Matching*: Given a set S of n r-tuples and an integer k, is there a set $S^* \subseteq S$ consists of no less than k many r-tuples such that there is no common element between any two r-tuples in S^*.

Definition 11 *Weighted r-D Matching (*wr dm*)*: Given a set S of n weighted r-tuples and an integer k, is there a set $S^* \subseteq S$ with weight value no less than k such that there is no common element between any two r-tuples in S^*.

Definition 12 *r-Set Packing*: Given a set S of n r-sets and an integer k , is there a set $S^* \subseteq S$ consists of no less k r-sets such that there is no common element between any two sets in S^*.

Definition 13 *Weighted r-Set Packing (*wr sp*)*: Given a set S of n weighted r-sets and an integer k, is there a set $S^* \subseteq S$ with weight value at least k so that there is no common element between any two sets in S^*.

When $r = 3$, the problems defined above are NP-hard. Chen et al. proposed an $O^*((5.7k)^k)$-time algorithm for 3-D Matching. They greatly improved the previous trivial algorithm with time complexity of $O((3k)!(3k)^{9k+1})$, which was proposed by Downey et al. [121]. They first designed a non-deterministic algorithm, and then used the deterministic simulation stage to eliminate the non-determinism of the algorithm. The method they used in the deterministic simulation phase was to calculate the number of guessed binary bits first, and then enumerate and check all the possibilities.

Fellows et al. proposed an FPT algorithm for r-D Matching and r-Set Packing. The running time of them are $O^*(2^{(c+1)rk})$ and $O^*(2^{O((c+1)rk})$ respectively [122], and the algorithm for r-Set Packing was obtained by adjusting the algorithm for r-D Matching. They built a common framework to discover and exploit problem kernels with small size. In this framework, they use the techniques color coding and dynamic programming to design PFT algorithms. The method they proposed is very flexible, and the smaller constants in the exponent can be achieved by adjusting the algorithm.

Chen et al. improved the deterministic and randomized algorithms for some packing and matching problems. Using the Divide-and-Conquer, they proposed a randomized algorithm with $O^*(4^k)$ [123]. In order to implement the improved deterministic

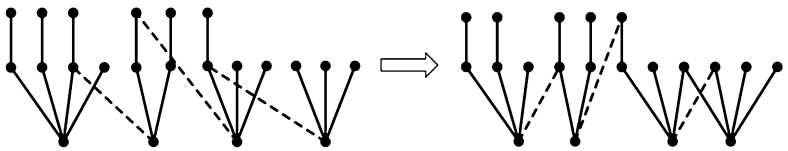

Fig. 9 An example of finding larger P_2-packing by Deeper Local Search

algorithm, they proposed a new (n, k)-family perfect hash function. Combining with color coding technology, they improved the deterministic algorithms for the following three problems. The first one is for k-path, which runs in $O^*(4^{k+o(k)})$-time; the second one is for 3-D Matching, the time complexity of which is $O^*(2.80^{3k})$; the last one is for 3-Set Packing, whose running time is $O^*(4^{3k+o(k)})$ [123]. They proposed a new method called iterative extension, which is based on the integration and improvement of known techniques including greedy localization, color coding, and dynamic programming. They gave a deterministic construction of scheme of the k-color coding and replaced the enumeration phase of the greedy location method with a more efficient enumeration phase using improved color coding and dynamic programming methods.

Later, Wang et al. considered w3sp. They, by using color coding technique, obtained an $O^*(10.6^{3k})$-time deterministic algorithm [124]. Furthermore, they pointed out that the bound could be improved to $O^*(7.56^{3k})$ by combining the random divide-and-conquer method.

Jia et al. proposed an $O^*((5.7k)^k)$-time algorithm for 3-Set Packing [125]. They first designed a non-deterministic algorithm and then used intelligent "deterministic techniques" to transform non-deterministic algorithms into effective deterministic algorithms for r-Set Packing problem. Based on the modified dynamic programming and color-coding, Wang et al. obtained an improved FPT algorithm for weighted r-Set Packing, which runs in time $O^*(12.8^{rk})$ [126]. They also proposed an FPT algorithm with running time $O^*(12.8^{(r-1)k})$ for wr DM.

Chen et al. proposed an $O^*(4^{(r-1)k+o(k)})$-time deterministic algorithm for wr DM [127]. Their algorithm further analyzed the structure of the problem, and combined (n, k)-universal set and Divide-and-Conquer method. Applying this algorithm to unweighted 3-D Matching problem, they obtained an $O^*(16^{k+o(k)})$-time deterministic algorithm. They also gave a deterministic FPT algorithm for wr SP which runs in time $O^*(2^{(2r-1)k+o(k)})$. Furthermore, this algorithm can solve the unweighted 3-Set Packing in $O^*(32^{k+o(k)})$ time.

For 3-Set Packing, Wang et al. [128] proposed a refined parameterized algorithm which runs in time $O^*(3.53^{3k})$. The improvement is mainly due to two sub-algorithms: one can deal with the instances in the case that there is one known element in each 3-set of the solution; the other can deal with the instances in the case that there are at least two known elements in each 3-set of the solution. Furthermore, they can invert one general instance into two instances mention above.

Definition 14 *Edge Disjoint Triangle Packing (* EDTP*)*: Given a graph $G = (V, E)$ and a non-negative integer k, the question is whether G contain k many edge disjoint triangles.

Mathieson et al. [129] proposed a kernelization algorithm resulting in a $4k$-kernel for the EDTP problem. The algorithm use a modified Crown Decomposition method, in which the matching is between an independent set and a set of edge (not vertices). Figure 10 shows an example of such an modified crown. Yang [130] proposed an algorithm with a kernel size of $3.5k$ for the EDTP problem. He first found a maximum triangle packing with kernelization technology, containing no more than $3k$ vertices, then bound the size of the rest of the graph by using reduction rules. Lin et al. [131] further studied

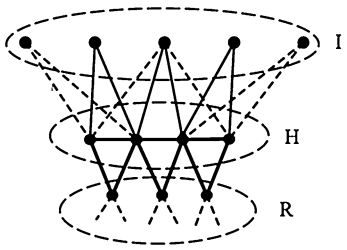

Fig. 10 An example of modified crown for EDTP

the edge disjoint triangle packing problem and obtained a $(3+\varepsilon)k$ kernel. They used Divide-and-Conquer techniques in the algorithm, so that some parts of the graph could be solved independently, which improving the analysis process. They proposed a property called p-property, where p is a non-negative integer constant.

Conclusion

Aiming at providing a reference for researchers interested in parameterized algorithms or engineers in practical applications, we surveyed the parameterized complexity of numerous fundamental NP-hard problems, including MaxSAT, Spanning Trees, Dominating Set, Feedback Vertex Set, Covering, Matching, Packing problems, etc. These problems have significant applications in a wide range of industrial fields. The problems discussed in the paper are selected due to their influence, representativeness, and significance.

Acknowledgements
The authors would like to thank the reviewers for their valuable comments which improve the quality of this paper dramatically.

Authors' contributions
Everyone in the author list has participated in the writing of this article, reviewed and revised the article reasonably. All authors read and approved the final manuscript.

Funding
This work was supported by National Natural Science Foundation of China (Nos. 61872048, 61772454, 61811530332, 61702491, 61702557).

Competing interests
No competing interests.

Author details
[1] Department of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China. [2] Saarland University, Saarbrücken, Germany. [3] School of Systems Engineering, University of Reading, Reading, UK. [4] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, South Korea.

Received: 7 January 2020 Accepted: 15 April 2020
Published online: 02 July 2020

References

1. Ghrabat MJJ, Ma G, Maolood IY, Alresheedi SS, Abduljabbar ZA (2019) An effective image retrieval based on optimized genetic algorithm utilized a novel svm-based convolutional neural network classifier. Hum Centr Comput Inform Sci 9:31. https://doi.org/10.1186/s13673-019-0191-8
2. Darwish A, Hassanien AE, Das S (2020) A survey of swarm and evolutionary computing approaches for deep learning. Artif Intell Rev 53(3):1767–1812. https://doi.org/10.1007/s10462-019-09719-2
3. Chen Y, Wang J, Liu S, Chen X, Xiiong J, Xie J, Yang K (2019) The multi-scale fast correlation filtering tracking algorithm based on a features fusion model. Concurr Comput Pract Exp. https://doi.org/10.1002/cpe.5533

4. Yu F, Liu L, Xiao L, Li K, Cai S (2019) A robust and xed-time zeroing neural dynamics for computing time-variant nonlinear equation using a novel nonlinear activation function. Neurocomputing 350:108–116. https://doi.org/10.1016/j.neucom.2019.03.053
5. Rostami SMH, Sangaiah AK, Wang J, Liu X (2019) Obstacle avoidance of mobile robots using modified artificial potential field algorithm. EURASIP J Wireless Comm Netw 2019:70. https://doi.org/10.1186/s13638-019-1396-2
6. Danial SN, Smith J, Veitch B, Khan FI (2019) On the realization of the recognition-primed decision model for artificial agents. Hum Centr Comput Inform Sci 9:36. https://doi.org/10.1186/s13673-019-0197-2
7. Cohen J (2005) Computer science and bioinformatics. Commun ACM 48(3):72–78. https://doi.org/10.1145/1047671.1047672
8. Toro-Dominguez D, Villatoro-Garca JA, Martorell-Marugan J, Roman-Montoya Y, Alarcon-Riquelme ME, Carmona-Saez P (2020) A survey of gene expression meta-analysis: methods and applications. Briengs Bioinf. https://doi.org/10.1093/bib/bbaa019
9. Naimi AI, Westreich DJ (2014) Big data: a revolution that will transform how we live, work, and think. Oxford University Press, Oxford
10. Liao Z, Zhang R, He S, Zeng D, Wang J, Kim H (2019) Deep learning-based data storage for low latency in data center networks. IEEE Access 7:26411–26417. https://doi.org/10.1109/ACCESS.2019.2901742
11. Wang J, Gu X, Liu W, Sangaiah AK, Kim H (2019) An empower Hamilton loop based data collection algorithm with mobile agent for WSNs. Hum Centr Comput Inform Sci 9:18. https://doi.org/10.1186/s13673-019-0179-4
12. Xiang L, Shen X, Qin J, Hao W (2018) Discrete multi-graph hashing for large-scale visual search. Neural Process Lett. https://doi.org/10.1007/s11063-018-9892-7
13. Wang J, Yang Y, Wang T, Sherratt RS, Zhang J (2020) Big data service architecture: a survey. J Internet Technol 21(2):393–405. https://doi.org/10.3966/160792642020032102008
14. Gungor VC, Lu B, Hancke GP (2010) Opportunities and challenges of wireless sensor networks in smart grid. IEEE Trans Ind Electr 57(10):3557–3564. https://doi.org/10.1109/TIE.2009.2039455
15. Tang Q, Wang K, Song Y, Li F, Park JH (2019) Waiting time minimized charging and discharging strategy based on mobile edge computing supported by software defined network. IEEE Intern Things J. https://doi.org/10.1109/JIOT.2019.2957124
16. Yick J, Mukherjee B, Ghosal D (2008) Wireless sensor network survey. Comput Netw 52(12):2292–2330. https://doi.org/10.1016/j.comnet.2008.04.002
17. Wang J, Gao Y, Wang K, Sangaiah AK, Lim S-J (2019) An affinity propagation-based self-adaptive clustering method for wireless sensor networks. Sensors 19(11):2579. https://doi.org/10.3390/s19112579
18. He S, Xie K, Xie K, Xu C, Wang J (2019) Interference-aware multisource transmission in multiradio and multichannel wireless network. IEEE Syst J 13(3):2507–2518. https://doi.org/10.1109/JSYST.2019.2910409
19. Wang W, Deng Z, Wang J (2019) Enhancing sensor network security with improved internal hardware design. Sensors 19(8):1752. https://doi.org/10.3390/s19081752
20. Nieto A, Rios R (2019) Cybersecurity profiles based on human-centric IoT devices. Hum Centr Comput Inform Sci. https://doi.org/10.1186/s13673-019-0200-y
21. Li W, Chen Z, Gao X, Liu W, Wang J (2019) Multimodel framework for indoor localization under mobile edge computing environment. IEEE Intern Things J 6(3):4844–4853. https://doi.org/10.1109/JIOT.2018.2872133
22. Jo D, Kim GJ (2019) Iot + AR: pervasive and augmented environments for "digi-log" shopping experience. Hum Centr Comput Inform Sci 9:1. https://doi.org/10.1186/s13673-018-0162-5
23. Li W, Xu H, Li H, Yang Y, Sharma PK, Wang J, Singh S (2019) Complexity and algorithms for superposed data uploading problem in networks with smart devices. IEEE Intern Things J. https://doi.org/10.1109/JIOT.2019.2949352
24. Luo Y, Li W, Qiu S (2020) Anomaly detection based latency-aware energy consumption optimization for iot dataflow services. Sensors 20(1):122. https://doi.org/10.3390/s20010122
25. Wang J, Gao Y, Zhou C, Sherratt RS, Wang L (2020) Optimal coverage multi-path scheduling scheme with multiple mobile sinks for WSNs. Comput Mater Continua 62(2):695–711. https://doi.org/10.32604/cmc.2020.08674
26. Bilal SM, Bernardos CJ, Guerrero C (2013) Position-based routing in vehicular networks: a survey. J Netw Comput Appl 36(2):685–697. https://doi.org/10.1016/j.jnca.2012.12.023
27. Cao D, Zheng B, Ji B, Lei Z, Feng C (2018) A robust distance-based relay selection for message dissemination in vehicular network. Wireless Netw. https://doi.org/10.1007/s11276-018-1863-4
28. Cao D, Liu Y, Ma X, Wang J, Ji B, Feng C, Si J (2019) A relay-node selection on curve road in vehicular networks. IEEE Access 7:12714–12728. https://doi.org/10.1109/ACCESS.2019.2892979
29. Gao K, Huang S, Han F, Li S, Wu W, Du R (2020) An integrated algorithm for intersection queue length estimation based on IoT in a mixed trac scenario. Appl Sci 10(6):2078. https://doi.org/10.3390/app10062078
30. Alresheedi SS, Lu S, Elaziz MEA, Ewees AA (2019) Improved multiobjective salp swarm optimization for virtual machine placement in cloud computing. Hum Centr Comput Inform Sci 9:15. https://doi.org/10.1186/s13673-019-0174-9
31. He S, Xie K, Zhou X, Semong T, Wang J (2019) Multi-source reliable multicast routing with qos constraints of nfv in edge computing. Electronics 8:10. https://doi.org/10.3390/electronics8101106
32. Tang Q, Changa L, Yang K, Wang K, Wanga J, KumarSharma P (2020) Task number maximization offloading strategy seamlessly adapted to UAV scenario. Comput Commun 151:19–30. https://doi.org/10.1016/j.comcom.2019.12.018
33. Gu K, Wu N, Yin B, Jia W (2019) Secure data query framework for cloud and fog computing. IEEE Trans Netw Serv Manag. https://doi.org/10.1109/TNSM.2019.2941869
34. Gu K, Wu N, Yin B, Jia W (2019) Secure data sequence query framework based on multiple fogs. IEEE Trans Emerg Top Comput. https://doi.org/10.1109/TETC.2019.2943524
35. Loce RP, Bernal EA, Wu W, Bala R (2013) Computer vision in roadway transportation systems: a survey. J Electr Imag 22(4):041121. https://doi.org/10.1117/1.JEI.22.4.041121
36. Zhang J, Xie Z, Sun J, Zou X, Wang J (2020) A cascaded r-cnn with multiscale attention and imbalanced samples for traffic sign detection. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2972338

37. Zhang J, Wang W, Lu C, Wang J, Sangaiah AK (2019) Lightweight deep network for traffic sign classification. Ann Telecommun. https://doi.org/10.1007/s12243-019-00731-9
38. Imran M, Durad MH, Khan FA, Derhab A (2019) Reducing the effects of dos attacks in software defined networks using parallel flow installation. Hum Centr Comput Inform Sci 9:16. https://doi.org/10.1186/s13673-019-0176-7
39. Xiang L, Guo G, Yu J, Sheng V, Yang P (2020) A convolutional neural network-based linguistic steganalysis for synonym substitution steganography. Math Biosci Eng 17:1041–1058. https://doi.org/10.3934/mbe.2020055
40. Zhang P, Wang J (2019) On enhancing network dynamic adaptability for compressive sensing in wsns. IEEE Trans Comm 67(12):8450–8459. https://doi.org/10.1109/TCOMM.2019.2938950
41. Yu F, Liu L, He B, Huang Y, Shi C, Cai S, Song Y, Du S, Wan Q (2019) Analysis and FPGA realization of a novel 5D hyperchaotic four-wing memristive system, active control synchronization, and secure communication application. Complexity 2019:4047957. https://doi.org/10.1155/2019/4047957
42. Yuan C, Xia Z, Sun X, Wu QJ (2019) Deep residual network with adaptive learning framework for fingerprint liveness detection. IEEE Trans Cogn Dev Syst. https://doi.org/10.1109/TCDS.2019.2920364
43. Zhang J, Zhong S, Wang T, Chao H-C, Wang J (2020) Blockchain-based systems and applications: a survey. J Intern Technol 21(1):1–14. https://doi.org/10.3966/160792642020012101001
44. Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Springer, London. https://doi.org/10.1007/978-1-4471-5559-1
45. Xu C, Li W, Yang Y, Chen J, Wang J (2019) Resolution and domination: An improved exact maxsat algorithm. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pp. 1191–1197. https://doi.org/10.24963/ijcai.2019/166
46. Yang Y, Guo J (2018) Parameterized complexity of voter control in multi-peaked elections. Theory Comput Syst 62(8):1798–1825. https://doi.org/10.1007/s00224-018-9843-8
47. Ganian R, Kanj IA, Ordyniak S, Szeider S (2018) Parameterized algorithms for the matrix completion problem. In: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, pp. 1642–1651. http://proceedings.mlr.press/v80/ganian18a.html
48. Grohe M (2001) The parameterized complexity of database queries. In: Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA, pp. 82–92. https://doi.org/10.1145/375551.375564
49. Yang Y, Wang J (2018) Parameterized complexity of multi-winner determination: More effort towards fixed-parameter tractability. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018, pp. 2142–2144. http://dl.acm.org/citation.cfm?id=3238099
50. Chen Y, Goebel R, Lin G, Su B, Xu Y, Zhang A (2019) An improved approximation algorithm for the minimum 3-path partition problem. J Comb Optim 38(1):150–164. https://doi.org/10.1007/s10878-018-00372-z
51. Yang Y, Shrestha YR, Li W, Guo J (2018) On the kernelization of split graph problems. Theor. Comput. Sci. 734:72–82. https://doi.org/10.1016/j.tcs.2017.09.023
52. Braunstein A, Mézard M, Zecchina R (2005) Survey propagation: an algorithm for satisfiability. Random Struct Algor 27(2):201–226. https://doi.org/10.1002/rsa.20057
53. Battiti R, Protasi M (1997) Reactive search, a history-sensitive heuristic for MAX-SAT. ACM J Exp Algor 2:2. https://doi.org/10.1145/264216.264220
54. Gallaire H, Minker J, Nicolas J-M (1989) Logic and databases: a deductive approach. In: Readings in Artificial Intelligence and Databases, pp. 231–247. New York: Elsevier
55. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44(4):279–303. https://doi.org/10.1007/BF02241270
56. Nguyen TA, Perkins WA, Laffey TJ, Pecora D (1985) Checking an expert systems knowledge base for consistency and completeness. In: Joshi AK (ed.) Proceedings of the 9th International Joint Conference on Artificial Intelligence, pp. 375–378
57. Raman V, Ravikumar B, Rao SS (1998) A simplified NP-complete MAXSAT problem. Inform Process Lett 65(1):1–6. https://doi.org/10.1016/S0020-0190(97)00223-8
58. Davis M, Putnam H (1960) A computing procedure for quantification theory. J ACM 7(3):201–215. https://doi.org/10.1145/321033.321034
59. Downey RG, Fellows MR (1995) Fixed-parameter tractability and completeness I: basic results. SIAM J Comput 24(4):873–921. https://doi.org/10.1137/S0097539792228228
60. Niedermeier R, Rossmanith P (1999) New upper bounds for MaxSat. In: ICALP, pp. 575–584. https://doi.org/10.1007/3-540-48523-6_54
61. Bansal N, Raman V (1999) Upper bounds for MaxSAT: Further improved. In: ISAAC, pp. 247–258. https://doi.org/10.1007/3-540-46632-0_26
62. Chen J, Kanj IA (2004) Improved exact algorithms for Max-Sat. Discr Appl Math 142(1–3):17–27. https://doi.org/10.1016/j.dam.2003.03.002
63. Bliznets I, Golovnev A (2012) A new algorithm for parameterized MAX-SAT. In: International Symposium on Parameterized and Exact Computation Springer, pp. 37–48. https://doi.org/10.1007/978-3-642-33293-7_6
64. Xu C, Chen J, Wang J (2019) Resolution and linear CNF formulas: Improved (n, 3)-maxsat algorithms. Theor Comput Sci 774:113–123. https://doi.org/10.1016/j.tcs.2016.08.008
65. Li W, Xu C, Wang J, Yang Y (2017) An improved branching algorithm for $(n, 3)$-MaxSAT based on refined observations. In: International Conference on Combinatorial Optimization and Applications, pp. 94–108. https://doi.org/10.1007/978-3-319-71147-8_7
66. Prieto-Rodriguez E, Sloper C (2005) Reducing to independent set structure - the case of k-internal spanning tree. Nordic J Comput 12(3):308–318. https://doi.org/10.5555/1145884.1145890
67. Binkele-Raible D, Fernau H, Gaspers S, Liedloff M (2013) Exact and parameterized algorithms for max internal spanning tree. Algorithmica 65(1):95–128. https://doi.org/10.1007/s00453-011-9575-5

68. Fomin FV, Gaspers S, Saurabh S, Stepanov AA (2009) On two techniques of combining branching and treewidth. Algorithmica 54(2):181–207. https://doi.org/10.1007/s00453-007-9133-3
69. Li W, Cao Y, Chen J, Wang J (2017) Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree. Inf Comput 252:187–200. https://doi.org/10.1016/j.ic.2016.11.003
70. Gutin GZ, Razgon I, Kim EJ (2009) Minimum leaf out-branching and related problems. Theor Comput Sci 410(45):4571–4579. https://doi.org/10.1016/j.tcs.2009.03.036
71. Cohen N, Fomin FV, Gutin GZ, Kim EJ, Saurabh S, Yeo A (2010) Algorithm for finding *k*-vertex out-trees and its application to *k*-internal out-branching problem. J Comput Syst Sci 76(7):650–662. https://doi.org/10.1016/j.jcss.2010.01.001
72. Fomin FV, Grandoni F, Lokshtanov D, Saurabh S (2012) Sharp separation and applications to exact and parameterized algorithms. Algorithmica 63(3):692–706. https://doi.org/10.1007/s00453-011-9555-9
73. Zehavi M (2013) Algorithms for *k*-internal out-branching. In: 8th International Symposium on Parameterized and Exact Computation, pp. 361–373. https://doi.org/10.1007/978-3-319-03898-8_30
74. Alber J, Bodlaender HL, Fernau H, Kloks T, Niedermeier R (2002) Fixed parameter algorithms for DOMINATING SET and related problems on planar graphs. Algorithmica 33(4):461–493. https://doi.org/10.1007/s00453-001-0116-5
75. Kanj IA, Perkovic L (2002) Improved parameterized algorithms for planar dominating set. In: 27th International Symposium on Mathematical Foundations of Computer Science, pp. 399–410. https://doi.org/10.1007/3-540-45687-2_33
76. Alber J, Fellows MR, Niedermeier R (2004) Polynomial-time data reduction for dominating set. J ACM 51(3):363–384. https://doi.org/10.1145/990308.990309
77. Alber J, Fan H, Fellows MR, Fernau H, Niedermeier R, Rosamond FA, Stege U (2005) A refined search tree technique for dominating set on planar graphs. J Comput Syst Sci 71(4):385–405. https://doi.org/10.1016/j.jcss.2004.03.007
78. Chen J, Fernau H, Shaw P, Wang J, Yang Z (2012) Kernels for packing and covering problems - (extended abstract). In: Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, pp. 199–211. https://doi.org/10.1007/978-3-642-29700-7_19
79. Lokshtanov D, Mnich M, Saurabh S (2009) Linear kernel for planar connected dominating set. In: Proceedings of Theory and Applications of Models of Computation. Lecture Notes in Computer Science, vol. 5532, pp. 281–290. https://doi.org/10.1007/978-3-642-02017-9_31
80. Gu Q, Imani N (2010) Connectivity is not a limit for kernelization: Planar connected dominating set. In: Latin American Symposium on Theoretical Informatics, pp. 26–37. https://doi.org/10.1007/978-3-642-12200-2_4
81. Luo W, Wang J, Feng Q, Guo J, Chen J (2011) An improved kernel for planar connected dominating set. In: Proceedings of Theory Applications of Models of Computation-conference. Lecture Notes in Computer Science, vol. 6648, pp. 70–81. https://doi.org/10.1007/978-3-642-20877-5_8
82. Misra N, Philip G, Raman V, Saurabh S (2014) The kernelization complexity of connected domination in graphs with (no) small cycles. Algorithmica 68(2):504–530. https://doi.org/10.1007/s00453-012-9681-z
83. Li W, Feng Q, Chen J, Hu S (2017) Improved kernel results for some FPT problems based on simple observations. Theor Comput Sci 657:20–27. https://doi.org/10.1016/j.tcs.2016.06.012
84. Fernau H (2006) Edge dominating set: Efficient enumeration-based exact algorithms. In: International Workshop on Parameterized and Exact Computation, pp. 142–153. https://doi.org/10.1007/11847250_13
85. Xiao M, Kloks T, Poon S (2013) New parameterized algorithms for the edge dominating set problem. Theor Comput Sci 511:147–158. https://doi.org/10.1016/j.tcs.2012.06.022
86. Rodríguez EP (2005) Systematic kernelization in FPT algorithm design. PhD thesis, The University of Newcastle
87. Hagerup T (2012) Kernels for edge dominating set: Simpler or smaller. In: Rovan B, Sassone V, Widmayer P (eds) 37th International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 7464, pp. 491–502 (2012). https://doi.org/10.1007/978-3-642-32589-2_44
88. Silberschatz A, Galvin PB, Gagne G (2005) Operating system concepts. Wiley, New York
89. Festa P, Pardalos PM, Resende MGC (2009) Feedback set problems. In: Floudas CA, Pardalos PM (eds.) Encyclopedia of Optimization, Second Edition, pp. 1005–1016. https://doi.org/10.1007/978-0-387-74759-0_178
90. Raman V, Saurabh S, Subramanian CR (2002) Faster fixed parameter tractable algorithms for undirected feedback vertex set. In: 13th International Symposium on Algorithms and Computation, pp. 241–248. https://doi.org/10.1007/3-540-36136-7_22
91. Kanj IA, Pelsmajer MJ, Schaefer M (2004) Parameterized algorithms for feedback vertex set. In: International Workshop on Parameterized and Exact Computation, pp. 235–247. https://doi.org/10.1007/978-3-540-28639-4_21
92. Dehne FKHA, Fellows MR, Langston MA, Rosamond FA, Stevens K (2007) An $o(2^{O(k)})n^3$) FPT algorithm for the undirected feedback vertex set problem. Theory Comput Syst 41(3):479–492. https://doi.org/10.1007/s00224-007-1345-z
93. Chen J, Fomin FV, Liu Y, Lu S, Villanger Y (2007) Improved algorithms for the feedback vertex set problems. In: 10th International Workshop on Algorithms and Data Structures, pp. 422–433. https://doi.org/10.1007/978-3-540-73951-7_37
94. Cao Y, Chen J, Liu Y (2015) On feedback vertex set: new measure and new structures. Algorithmica 73(1):63–86. https://doi.org/10.1007/s00453-014-9904-6
95. Kociumaka T, Pilipczuk M (2014) Faster deterministic feedback vertex set. Inform Process Lett 114(10):556–560. https://doi.org/10.1016/j.ipl.2014.05.001
96. Cao Y (2018) A naive algorithm for feedback vertex set. In: 1st Symposium on Simplicity in Algorithms, pp. 1–119. https://doi.org/10.4230/OASIcs.SOSA.2018.1
97. Burrage K, Estivill-Castro V, Fellows MR, Langston MA, Mac S, Rosamond FA (2006) The undirected feedback vertex set problem has a poly(*k*) kernel. In: International Workshop on Parameterized and Exact Computation, pp. 192–202. https://doi.org/10.1007/11847250_18
98. Bodlaender HL, van Dijk TC (2010) A cubic kernel for feedback vertex set and loop cutset. Theory Comput Syst 46(3):566–597. https://doi.org/10.1007/s00224-009-9234-2
99. Thomassé S (2010) A $4k^2$ kernel for feedback vertex set. ACM Trans Algor 6(2):1–8. https://doi.org/10.1145/1721837.1721848
100. Bodlaender HL, Penninkx E (2008) A linear kernel for planar feedback vertex set. In: Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings, pp. 160–171. https://doi.org/10.1007/978-3-540-79723-4_16

101. Abu-Khzam FN, Khuzam MB (2012) An improved kernel for the undirected planar feedback vertex set problem. In: 7th International Symposium on Parameterized and Exact Computation, pp. 264–273. https://doi.org/10.1007/978-3-642-33293-7_25
102. Xiao M (2014) A new linear kernel for undirected planar feedback vertex set: Smaller and simpler. In: 10th International Conference on Algorithmic Aspects in Information and Management, pp. 288–298. https://doi.org/10.1007/978-3-319-07956-1_26
103. Bonamy M, Kowalik L (2014) A 14*k*-kernel for planar feedback vertex set via region decomposition. In: International Symposium on Parameterized and Exact Computation, pp. 97–109. https://doi.org/10.1007/978-3-319-13524-3_9
104. Bonamy M, Kowalik L (2016) A 13*k*-kernel for planar feedback vertex set via region decomposition. Theor Comput Sci 645:25–40. https://doi.org/10.1016/j.tcs.2016.05.031
105. Buss JF, Goldsmith J (1993) Nondeterminism within P. SIAM J Comput 22(3):560–572. https://doi.org/10.1137/0222038
106. Balasubramanian R, Fellows MR, Raman V (1998) An improved fixed-parameter algorithm for vertex cover. Inf Process Lett 65(3):163–168. https://doi.org/10.1016/S0020-0190(97)00213-5
107. Downey RG, Fellows MR (1999) Parameterized complexity. Springer, New York. https://doi.org/10.1007/978-1-4612-0515-9
108. Niedermeier R, Rossmanith P (1999) Upper bounds for vertex cover further improved. In: 16th annual symposium on theoretical aspects of computer science, pp. 561–570. https://doi.org/10.1007/3-540-49116-3_53
109. Chen J, Kanj IA, Jia W (2001) Vertex cover: further observations and further improvements. J Algor 41(2):280–301. https://doi.org/10.1006/jagm.2001.1186
110. Chen J, Kanj IA, Xia G (2006) Improved parameterized upper bounds for vertex cover. In: 31st International symposium on mathematical foundations of computer science, pp. 238–249. https://doi.org/10.1007/11821069_21
111. Nemhauser GL, Trotter LE Jr (1975) Vertex packings: structural properties and algorithms. Math Program 8(1):232–248. https://doi.org/10.1007/BF01580444
112. Ebengger C, Hammer P, de Werra D (1984) Pseudo-boolean functions and stability of graphs. Ann Discr Math 19:83–93. https://doi.org/10.1016/S0304-0208(08)72955-4
113. Abu-Khzam FN, Fellows MR, Langston MA, Suters WH (2007) Crown structures for vertex cover kernelization. Theory Comput Syst 41(3):411–430. https://doi.org/10.1007/s00224-007-1328-0
114. Li W, Zhu B (2018) A 2*k*-kernelization algorithm for vertex cover based on crown decomposition. Theor Comput Sci 739:80–85. https://doi.org/10.1016/j.tcs.2018.05.004
115. Grantson M, Levcopoulos C (2006) Covering a set of points with a minimum number of lines. In: 6th Italian Conference on Algorithms and Complexity, pp. 6–17. https://doi.org/10.1007/11758471_4
116. Wang J, Li W, Chen J (2010) A parameterized algorithm for the hyperplane-cover problem. Theor Comput Sci 411(44–46):4005–4009. https://doi.org/10.1016/j.tcs.2010.08.012
117. Afshani P, Berglin E, van Duijn I, Nielsen JS (2016) Applications of incidence bounds in point covering problems. In: 32nd International symposium on computational geometry, pp. 60–16015. https://doi.org/10.4230/LIPIcs.SoCG.2016.60
118. Prieto-Rodriguez E, Sloper C (2006) Looking at the stars. Theor Comput Sci 351(3):437–445. https://doi.org/10.1016/j.tcs.2005.10.009
119. Wang J, Ning D, Feng Q, Chen J (2010) An improved kernelization for p_2-packing. Inform Process Lett 110(5):188–192. https://doi.org/10.1016/j.ipl.2009.12.002
120. Li W, Ye J, Cao Y (2018) Kernelization for p_2-packing: A gerrymandering approach. In: International frontiers of algorithmics workshop, pp. 140–153. https://doi.org/10.1007/978-3-319-78455-7_11
121. Chen J, Friesen DK, Jia W, Kanj IA (2004) Using nondeterminism to design efficient deterministic algorithms. Algorithmica 40(2):83–97. https://doi.org/10.1007/s00453-004-1096-z
122. Fellows MR, Knauer C, Nishimura N, Ragde P, Rosamond FA, Stege U, Thilikos DM, Whitesides S (2008) Faster fixed-parameter tractable algorithms for matching and packing problems. Algorithmica 52(2):167–176. https://doi.org/10.1007/s00453-007-9146-y
123. Chen J, Lu S, Sze SH, Zhang F (2007) Improved algorithms for path, matching, and packing problems. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, pp. 298–307. https://doi.org/10.1145/1283383.1283415
124. Wang J, Feng Q (2008) Improved parameterized algorithms for weighted 3-set packing. In: 14th annual international conference on computing and combinatorics, pp. 130–139. https://doi.org/10.1007/978-3-540-69733-6_14
125. Jia W, Zhang C, Chen J (2004) An efficient parameterized algorithm for *m*-Set Packing. J Algor 50(1):106–117. https://doi.org/10.1016/j.jalgor.2003.07.001
126. Wang J, Liu Y (2008) Parameterized algorithms for weighted matching and packing problems. Discr Optim 5(4):748–754. https://doi.org/10.1016/j.disopt.2008.07.002
127. Chen J, Feng Q, Liu Y, Lu S, Wang J (2011) Improved deterministic algorithms for weighted matching and packing problems. Theor Comput Sci 412(23):2503–2512. https://doi.org/10.1016/j.tcs.2010.10.042
128. Wang J, Feng Q, Chen J (2011) An $\sigma^*(3.53^{3k})$-time parameterized algorithm for the 3-set packing problem. Theor Comput Sci 412(18):1745–1753. https://doi.org/10.1016/j.tcs.2010.12.048
129. Mathieson L, Prieto-Rodriguez E, Shaw P (2004) Packing edge disjoint triangles: a parameterized view. In: First international workshop on parameterized and exact computation, pp. 127–137. https://doi.org/10.1007/978-3-540-28639-4_12
130. Yang Y (2014) Towards optimal kernel for edge-disjoint triangle packing. Inform Process Lett 114(7):344–348. https://doi.org/10.1016/j.ipl.2014.02.003
131. Lin W, Xiao M (2019) A $(3+\epsilon)k$-vertex kernel for edge-disjoint triangle packing. Inform Process Lett 142:20–26. https://doi.org/10.1016/j.ipl.2018.10.006

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.