Survey on graph algorithms parameterized by edge edit distances*

Christophe Crespelle

University of Bergen, Department of Informatics, N-5020 Bergen, Norway
christophe.crespelle@uib.no

Abstract. This survey is intended to list algorithmic results on graph problems that are parameterized by the edit distance of the input graph G to some graph class $\mathcal{G}$, i.e. the number of edges to be deleted and/or added to G in order to obtain a graph belonging to $\mathcal{G}$.

1 Introduction and necessary background

The goal of a survey is always to identify what has been done and what has not been done on a given topic. Here the topic we cover may seem quite precise and a bit restricted to the reader. The reason behind this is that the boundaries of this survey have been settled in order to determine to which extent one precise idea has been explored in the FPT framework. This idea is very simple. Special graph classes, such as chordal graphs or cographs for example, have much more properties than arbitrary graphs, which even stands for their definition. Because of these properties, they often admit more efficient solutions to algorithmic problems than those we can design for arbitrary graphs in general. This is in particular the case for many difficult (NP-hard) algorithmic problems which sometimes become polynomially solvable on special classes of graphs, such as coloring on chordal graphs for example. What about graphs that are not chordal but almost chordal, in the sense that removing and/or adding a few edges, say k, in the graph will make it chordal? Are they also easy to color, as chordal graphs are? One can legitimately expect that the difficulty to solve the coloring problem should depend on k: the smaller k, the more one should retrieve the good behaviour of chordal graphs which can be colored in polynomial time, and the larger k, the more one should retrieve the behaviour of arbitrary graphs which need exponential time to be colored (assuming $P \neq NP$). This is exactly the idea of parameterized complexity: separate the dependency on k in the complexity so that the complexity of coloring depends polynomially on the size of the input graph, as it does for chordal graphs, and depend exponentially on the distance from the input graph to chordal graphs. This is precisely with this idea of exploiting the proximity of graphs with special graph classes that we wrote this survey on algorithms parameterized by graph edit distances.

The main idea of parameterized algorithms is to bound the running time of algorithms by a function that does not only depend on the size of the input instance but also on some other parameters, which are relevant to measure the difficulty of solving the problem on the

* This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 749022.

given instance. Classically, this idea is used to design algorithm for difficult problems (NP-hard) whose complexity only depends polynomially on the size of the instance n, and super-polynomially on another parameter k. This means that the complexity of the algorithm is expressed as $O(f(k).n^{O(1)})$, where f is only required to be computable. When it is possible to design such an algorithm, the problem is said to be Fixed Parameter Tractable (FPT for short) with respect to parameter k. When the problem considered is NP-complete, the function f cannot be a polynomial, unless P=NP. Very often, f is an exponential or an even faster growing function. But in some cases, while staying larger than any polynomial, f can be smaller than any exponential (e.g. $f(n) = 2^{\sqrt{n}}$). We then say that f is subexponential and that the problem admits a subexponential parameterized algorithm.

One special and very interesting technique for designing FPT algorithms is kernelization. A problem parameterized by k admits a kernel if there exists a polynomial time (with regard to n) algorithm that transforms the instance (I, k) into an instance (I', k') such that:

- (I', k') is a YES instance iff (I, k) is a YES instance, and
- the size of I' is bounded by $f(k)$, with f a computable function, and
- $k' \leq k$.

When no further restriction is imposed on f, admiting a kernel is equivalent to be FPT. But in general, the corresponding function f for an FPT problem grows exponentially or faster. A special case of interest is when f is a polynomial. Then, this gives a very efficient reduction approach that exchanges the original instance for another one which is equivalent and whose size is bounded by a relatively small function depending on k only.

This survey intends to list most of the FPT algorithms where the input is a graph G and the parameter k considered is the *editing distance* from G to some graph class $\mathcal{G}$. The edit distance, $dist(G, \mathcal{G})$ of a graph G to a graph class $\mathcal{G}$ is defined as the minimum number of adjacencies to be changed in G so that the resulting graph is in $\mathcal{G}$. Formally, if $G = (V, E)$, then $dist(G, \mathcal{G}) = min\{|F| \mid G' = (V, E\Delta F) \in \mathcal{G}\}$, where Δ denotes the symetric difference on sets. Here, two operations are used to transform the graph G into a graph belonging to class $\mathcal{G}$: addition of one edge and deletion of one edge. There are many other operations that one can consider, leading to different notion of distance. One of the other popular kind of modification is vertex deletion, which we do not consider here. This survey considers only the notion of distance resulting from adding and/or removing edges to the graph, without changing its vertex set. Nevertheless, we also consider the two particular cases where only additions of edges are allowed (completion distances) and where only deletions of edges are allowed (deletion distances).

In the following, we report results on algorithms parameterized by the edit (or completion or deletion) distance to some class of graphs. The vast majority of such results in the litterature (Section 2) are algorithms where the edit distance k is at the same time the parameter and the quantity to compute, or more exactly to be decided whether it is less than k. Only few results deviate from this framework and present algorithms where the proximity to some class $\mathcal{C}$ is used to solve problems that are different from computing the distance to $\mathcal{C}$, see Section 3. Nevertheless, with regard to the motivation of this survey, they are the most interesting problems.

For all the problems listed here, we are interested in knowing whether they are FPT or not (the problems considered are always NP-hard), what is the best parameterized complexity known for solving them, lower bounds on their complexity, whether they can be solved in subexponential parameterized time or not, whether they admit polynomial kernel or not and what is the smallest size of kernel known for them. We note that all the lower bounds and the impossibility results reported here are obtained using one of the three following complexity hypothesis:

– $P \neq NP$, or
– $coNP \not\subseteq NP/Poly$, or
– the Exponential Time Hypothesis (ETH for short)

2 Edge modification problems parameterized by number of modifications allowed

In this section, we report results on edge modification problems. These problems take as input a graph G and an integer k and ask whether G can be turned into a graph of a fixed given graph class $\mathcal{G}$ by performing at most k modifications. We consider three versions of such problems:

– editing (the more general version): both edge additions and edge deletions are allowed;
– completion : only edge additions are allowed; and
– deletion : only edge deletions are allowed.

In all the problems presented in this section, the parameter is the number k of modifications that are allowed. Moreover, it turns out that for all these problems, the target class of graph is hereditary. It means that for any graph $G \in \mathcal{G}$ in the class, all the induced subgraphs of G also belong to the class. Not all classes of graphs are hereditary, but most of the classes that are classically studied are. One possible reason for this is that heredity is a rather natural property to require from a graph class. Indeed, if belonging to the class is a characteristic of simplicity for a graph (in the sense that it satisfies a certain property, which is not necessarily satisfied by all graphs), then it is natural to ask that a subpart of a simple object is also simple. It is worth to note that all classes that are defined by forbidden induced subgraphs are hereditary, and that conversely, all hereditary classes of graphs can be defined by a (possibly infinite) set of forbidden induced subgraphs: those minimal graphs (for the subgraph ordering) that do not belong to the class.

This section is divided into four subsections. Section 2.1 presents both some general and particular results on classes of graphs defined by excluding a finite number of forbidden subgraphs. Section 2.2 deals with the particular case of classes excluding exactly one subgraph. Section 2.3 considers some classically studied graph classes, which may be defined either by excluding a finite number of subgraphs or an infinite family of them. Finally, Section 2.4 lists results on some other hereditary graph classes.

2.1 Finite $\mathcal{F}$-free classes

There is an early, very strong and very general result by Cai in 1996 [17] that impacts a lot the study of the tractability of edge modification problems into classes of graphs defined by a finite family of forbidden subgraphs: if $\mathcal{F}$ is finite, then the $\mathcal{F}$-free modification problem consisting in deleting at most i vertices, at most j edges, and adding at most k edges is FPT wrt. i, j, k. In particular, for the problems we are interested in here, it means that completion ($i = 0$ and $j = 0$), deletion ($i = 0$ and $k = 0$) and editing ($i = 0$ and try all couples j, k such that $j + k \leq l$) are all FPT wrt. the number of modifications allowed.

This has two immediate consequences for the domain:

1. the only hereditary classes for which the question of knowing whether the edge modification problems are FPT or not are the classes that are defined by an infinite family of forbidden graphs (see Section 2.3), and
2. for classes defined by a finite set of forbidden subgraphs, the question of interest are i) the possibility to have somehow small functions of k in the parameterized complexity and ii) the (non-)existence of polynomial kernels.

$\mathcal{F}$-free class	completion		deletion		edition	
$\{2K_2, C_5, K_3\}$ (chain)	KER k^2 [8, 30]	SUB $2^{\sqrt{k}\log k}$ [36, 30] NO $2^{k^{1/4}}$ [10]	KER k^2 [42, 30]	SUB $2^{\sqrt{k}\log k}$ [30]	KER k^2 [30]	SUB $2^{\sqrt{k}\log k}$ [30]
$\{2K_2, C_4, P_4\}$ * (threshold)	KER k^2 [30] KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [30, 31] NO $2^{k^{1/4}}$ [10]	KER k^2 [30] KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [30, 31] NO $2^{k^{1/4}}$ [10]	KER k^2 [30]	SUB $2^{\sqrt{k}\log k}$ [30]
$\{C_4, P_4\}$ (trivially perfect)	KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [31] NO $2^{k^{1/4}}$ [10]	KER k^7 [32]	FPT 2.45^k [55] NOSUB [31]	KER k^7 [32]	NOSUB [32]
$\{2K_2, C_4, P_4\}$ * (split)	KER k^2 [38] KER k^4 [42]	SUB $2^{\sqrt{k}\log k}$ [38]	KER k^2 [38] KER k^4 [42]	SUB $2^{\sqrt{k}\log k}$ [38]	-	-
$\{2K_2, C_4\}$ * (pseudosplit)	-	SUB $2^{\sqrt{k}\log k}$ [31]	-	SUB $2^{\sqrt{k}\log k}$ [31]	-	-
$\{claw, diamond\}$	-	-	KER k^{24} [29]	NOSUB [29]	-	-
$\{K_{1,s}, K_t\}_{s>1,t>2}$	-	-	KER [6]	-	-	-

Table 1. Parameterized complexity of some edge modification problems into $\mathcal{F}$-free graph classes, with $\mathcal{F}$ a finite family of graphs. For each kind of problem (completion, deletion or editing), the left column reports results on polynomial kernels and the right one reports results on the complexity of FPT algorithms. A star next to the name of the class marks autocomplemented classes, for which any result for one of the completion problem or deletion problem automatically gives the same result for the other problem.

Some such results are reported in Table 1 in particular for $\{claw, diamond\}$-free graphs and $\{K_{1,s}, K_t\}$-free graphs, with $s > 1, t > 2$. The results for the other classes in the table are also reported in the classic graph classes section.

There also exists an interesting general result stating the existence of polynomial kernels in the case of restricted inputs [6]: if the input graphs have bounded degree and if the graphs in $\mathcal{F}$ are connected, then the $\mathcal{F}$-free edge deletion problem admits a polynomial kernel.

2.2 *H*-free classes

In this section, we turn our attention to the special case where the number of forbidden subgraphs is exactly one. It is not clear whether this case is harder or easier than the case where the family contains several graphs. The first intuition is that forbidding only one subgraph should often yield harder problems as the target class has less structure than the case where several other graphs are forbidden. However, it is not true that the wider the target class, the harder the editing problems into this class. Take as an example the problem of editing a graph into an arbitrary graph (which is the wider class possible): it is obviously solvable in constant time (just output an empty set of modifications).

It turns out, that it took some time before it was exhibited an example of a graph H , on 7 vertices, such that both the H-free deletion and H-free editing problem were proven not to admit polynomial kernels unless $NP \subseteq coNP/poly$ [49]. Some other examples of such incompressibility results where shown very shortly after: [40] proved that the prolem does not admit a polynomial kernel for H being a cycle of at least 4 vertices or a path of at least 7 vertices. This result was improved until the complete characterisation of the existence of a kernel in the case where the target class excludes only one path or one cycle [19]: for H being a path or cycle, H-free edge deletion, completion and editing do not admit a polynomial kernel iff H has at least 4 edges.

As an example of this theorem, one can see in Table 2 that edge modification problems into the class of P_4-free graphs (also known as cographs) admit polynomial kernels, because a P_4 has only 3 edges, while modification problems into the class of C_4-free graphs do not, because a C_4 has 4 edges.

H-free class	completion		deletion		edition	
K_3	undefined		KER k^3 [14]	-	not meaningful	
K_4	undefined		k^4 [20]	NOSUB [5]	not meaningful	
P_3 (cluster)	polynomial		KER k^3 [39]	FPT 1.41^k [16] FPT 1.77^k [39] NOSUB [47]	KER $2k$ [28, 23] KER $4k$ [43] KER $6k$ [34] KER k^3 [39]	FPT 1.76^k [16] FPT 2.27^k [39] NOSUB [47]
P_4 (cograph)	k^3 [40]	NOSUB [31]	k^3 [40]	NOSUB [31]	k^3 [40]	NOSUB [32]
diamond	polynomial		k^3 [56, 25] k^4 [35]	NOSUB [56, 5]	k^8 [25]	NOSUB [5]
claw	-	NOSUB [5]	-	NOSUB [5]	-	NOSUB [5]
paw	-	NOSUB [5]	-	NOSUB [5]	-	NOSUB [5]
C_4	NOKER [40]	NOSUB [31]	NOKER [40]	NOSUB [31]	NOKER [40]	NOSUB [5]

Table 2. Parameterized complexity of edge modification problems into H-free graph classes, with H a graph on three or four vertices. Note that the complements of these graphs are not listed as both completion and deletion problems are considered.

A very general result on the existence of a kernel for edge modificaion problems into H-free graph classes was shown in [19]: when H is 3-connected, H-free edge deletion and editing admit no polynomial kernel iff H is not complete; and H-free edge completion admits no polynomial kernel iff H misses at least two edges. As a consequence, the question of the existence of a kernel is only open when H contains an articulation vertex or a disconnecting pair, which is the case of paths and cycles. Another very general result is about the existence of subexponential parameterized algorithms for edge modification problems into H-free graph classes [5]: when H has at least two edges (resp. non-edges), H-free edge deletion (resp. completion) is NP-complete and NOSUB; when H has at least three vertices, H-free edge editing is NP-complete and NOSUB.

Even more recently, such kind of results where extended to the question of the existence of parametrized subexponential approximation algorithms [11]: when H is 3-connected and has at least two non-edges, then there does not exist any poly(OPT)-approximation algorithm running in parameterized subexponential time (in OPT), unless ETH fails, for H-free edge deletion as well as H-free edge completion. Moreover, the same holds for H being a cycle on at least 4 vertices or a path on at least 5 vertices. With previous results, this solves all cases of paths and cycles except the cograph edge deletion problem, for which [11] suggests the existence of a parameterized subexponential approximation algorithm, because of the existence of a kernel to the problem [40].

On Table 2, one can see that the question of existence of a kernel for edge modification problems into H-free graphs has been answered for all graphs on 3 vertices and for almost all graphs on 4 vertices. Only the case of the claw ($K_{1,3}$) and the paw (triangle with a pending

vertex) are not solved, neither for completion, nor deletion nor editing. Also, it is not known whether there exist a subexponential parameterized algorithm for deletion into triangle free (K_3 free).

2.3 Classic classes

In this section, we report results on edge modification algorithms where the target class is one classically studied class of graphs, such as chordal graphs and cographs for example. These classes do not necessarily admit a characterisation by excluding a finite number of forbidden induced subgraphs, but some of them do and have therefore been already mentionned in the previous section (e.g. cographs). Nevertheless, they are also listed in this section in order to position the results obtained for them in the general picture of the classical graph classes hierarchy. All the FPT and kernelisation results for these classical classes of graphs are reported in Table 3. Most of the classes for which such results have been obtained fit into two families: subclasses of cographs (cographs, trivially perfect, threshold and cluster graphs) and subclasses of chordal graphs (chordal, strongly chordal, interval, proper interval, pseudo-split and split graphs). Only chain graphs (also called bipartite chain graphs) and 3-leaf powers do not fit in these two families.

The picture revealed by Table 3 is easy to catch at the first sight. For chain graphs, 3-leaf powers and all subclasses of cographs, polynomial kernels are known for all the three kinds of edge modification problems. The question of the existence of a subexponential parameterized algorithm has also been answered for all these classes and for all three problems, except for 3-leaf powers.

On the other hand, the kernelisation results for subclasses of chordal graphs appear to be much more sparse. Kernelisation algorithms are known only for split deletion and completion (the class is autocomplemented), but not for editing, for proper interval completion and for chordal completion. In all the other cases, the existence of a polynomial kernel is an open question. In particular, for interval completion the question is very appealing because of the existence of a kernel for the subclass of proper interval graph and for the super class of chordal graphs. We also note that the existence of a polynomial kernel for the editing problem is known for none of these subclasses of chordal graphs. Finding such a kernel for one them or proving that there is no is a question of high interest. Regarding subexponential parameterized algorithms, the situation is very similar to the one for polynomial kernels, with the difference that subexponential parameterized algorithms are known for pseudo-split deletion and completion (the class is auto complemented) as well as for interval completion.

Among the classes of graphs listed in Table 3, one received a particular attention: cluster graphs. The reason is that cluster graph modification problems, more precisely deletion and edition, are closely related to the question of community detection, which is central in the domain of complex networks. It is striking to see that despite the simplicity of the structure of cluster graphs (they are disjoint union of cliques), both the editing and deletion problems remain NP-complete. Completion is trivially polynomial: simply turn each connected component into a clique. Therefore, many variants of the problem of cluster editing have been considered in the litterature. They are not listed in Table 3 and we report them below. [37] gave a subexponential algorithm $O(2^{\sqrt{pk}} + n^{O(1)})$ for custer editing in the case where the number of clusters p is given in the problem and is taken as parameter as well. [47] gives a polynomial kernel $O(dt)$ for cluster editing when parameterized both by number of clusters d and maximum number of modifs t touching one vertex. The cluster editing problem also admits a $2k$ kernel in its integraly weighted version [23] and a FPT algorithm in $O(1.82^k)$ time [15]. [35] study a cluster editing problem where a vertex (or an edge) is allowed to participate to at most s clusters, where s is part of the

graph class	completion		deletion		edition	
chain	KER k^2 [8, 30]	SUB $2^{\sqrt{k}\log k}$ [36, 30] NO $2^{k^{1/4}}$ [10]	KER k^2 [42, 30]	SUB $2^{\sqrt{k}\log k}$ [30]	KER k^2 [30]	SUB $2^{\sqrt{k}\log k}$ [30]
3-leaf power	KER k^3 [7]	-	KER k^3 [7]	-	KER k^3 [7]	-
cluster	polynomial		KER k^3 [39]	FPT 1.41^k [16] FPT 1.77^k [39] NOSUB [47]	KER $2k$ [28, 23] KER $4k$ [43] KER $6k$ [34] KER k^3 [39]	FPT 1.76^k [16] FPT 2.27^k [39] NOSUB [47]
threshold *	KER k^2 [30] KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [30, 31] NO $2^{k^{1/4}}$ [10]	KER k^2 [30] KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [30, 31] NO $2^{k^{1/4}}$ [10]	KER k^2 [30]	SUB $2^{\sqrt{k}\log k}$ [30]
trivially perfect	KER k^3 [42]	SUB $2^{\sqrt{k}\log k}$ [31] NO $2^{k^{1/4}}$ [10]	KER k^7 [32]	FPT 2.45^k [55] NOSUB [31]	KER k^7 [32]	NOSUB [32]
cograph *	KER k^3 [40]	FPT 2.56^k [55] NOSUB [31, 32]	KER k^3 [40]	FPT 2.56^k [55] NOSUB [31, 32]	KER k^3 [40]	FPT 4.61^k [50] NOSUB [32]
split *	KER k^2 [38] KER k^4 [42]	SUB $2^{\sqrt{k}\log k}$ [38]	KER k^2 [38] KER k^4 [42]	SUB $2^{\sqrt{k}\log k}$ [38]	-	-
pseudosplit *	-	SUB $2^{\sqrt{k}\log k}$ [31]	-	SUB $2^{\sqrt{k}\log k}$ [31]	-	-
proper interval	KER k^3 [8]	SUB $2^{2/3\log k}$ [9] FPT 4^k [51] FPT 16^k [45] NO $2^{k^{1/4}}$ [10]	-	-	-	-
interval	-	SUB $2^{\sqrt{k}\log k}$ [12] FPT 6^k [21, 22] FPT $4^{k\log k}$ [57] NO $2^{k^{1/4}}$ [10]	-	FPT $2^{k\log k}$ [22]	-	-
strongly chordal	-	FPT 64^k [45]	-	-	-	-
chordal	KER k^3 [45]	SUB $2^{\sqrt{k}\log k}$ [36] FPT 4^k [17] FPT 16^k [45] FPT $2^{k\log k}$ [24] NO $2^{\sqrt{k}}$ [26]	-	FPT [53] FPT $2^{k\log k}$ [24]	-	FPT $2^{k\log k}$ [24]

Table 3. Parameterized complexity of edge modification problems into classic hereditary graph classes. A star next to the name of the class marks autocomplemented classes, for which any result for one of the completion problem or deletion problem automatically gives the same result for the other problem.

input. The problem is shown $W[1]$-hard wrt k and FPT wrt (s,k). [35] also gives a $O(k^4)$ kernel for 1-edge-overlap-deletion (= diamond free editing) and a $O(k^3)$ kernel for 2-vertex-overlap-deletion. There is also a relaxation of the problem called s-plex cluster editing [41], where the clusters S are not clique but graphs of minimum degree $|S|-s$ ($s=1$ is the case of the clique). The problem is shown to admit a polynomial kernel parameterized by both k and s, and a FPT algorithm is given, running in time $O(\sqrt{s}^k)$. A variant of cluster editing for bipartite graphs aims at obtaining a union of complete bipartite graphs. It admits a $4k$ linear kernel and a FPT algorithm running in $O(3.24^k + |E|)$ [44].

2.4 Other target classes

In this section, we report results on edge modification problems into other graph classes, which are not defined by a finite family of forbidden subgraphs and that are not usually refered to in the hierarchy of classic graph classes.

There has been work done on the Flip-consensus tree problem, which is the problem of editing a bipartite graph into a bipartite graph with same partition that contains no P_5 starting from any top vertex: [27] proved the problem FPT with a $O(6^k n^2)$ algorithm, [13] gave a $O(4.42^k n)$ algorithm and [48] gave a $O(3.68^k n^3)$ algorithm and a $O(k^3)$ kernel.

[58] gives kernels for edge-deletion problems in order to remove cycles (not induced) of a given size equal to 4 or 5, or less than this size. The kernels given are quadratic for general graphs, for the $=4$ and ≤ 4 problems, and they are linear for planar graphs, for the $=4$, ≤ 4 and ≤ 5 problems.

For tournaments, which are directed complete graphs, both [33] and [46] independently give a subexponential algorithm in $O(2^{O(\sqrt{k})})$ time for Feddback Arc Set, which is equivalent to the DAG-deletion problem. There was previously a $O(2^{O(\sqrt{k}\log^2 k))}$ subexponential algorithm for the problem [4].

3 Other kind of parameterized problems

In this section, we report results on problems different from edge modification problems or that are parameterized by a quantity different from the distance to the target class.

3.1 Treewidth and Minimum Fill-in parameterized by the distance to split graphs

[52] shows that Treewidth and Minimum Fill-In are FPT for almost split graphs, more precisely for the so called split+ke graphs, which are these graphs that can be turned into a split graph by deleting at most k edges. The parameter used in the FPT algorithm is precisely the upperbound k on the distance to split graphs. [52] design an $O(2^{k\log k})$ algorithm for both problems. The technique involves the enumeration in FPT time of all minimal chordal completions of the graph.

This is an example of a FPT algorithm that is parameterized by the distance to some class of graphs (here split graphs) but where the aim of the problem is to compute something else than this distance. In the case of [52], the aim is either to compute the distance to another (very related) class of graphs, Minimum Fill-in, or another parameter that quantify the complexity of the graph, Treewidth. Note that the target class in the Minimum Fill-in problem is chordal graphs, which is a superclass of split graphs. In addition, note that computing the treewidth is closely related to chordal graph completion: the difference being that the quantity to minimize is not the number of edges added but instead the maximum clique size of the completed graph. Finally, it is worth noting that both Minimum Fill-in and Treewidth are polynomially

computable on split graphs. Then the result of [52] can be understood as taking advantage of the proximity of the input graph with a class $\mathcal{C}$ where these two parameters are easy to compute in order to compute it in FPT time parameterized by the distance to $\mathcal{C}$.

3.2 Boxicity

In [1, 2], the problem considered is to compute the boxicity of a graph, which is a graph parameter quantifying the difficulty of encoding a graph with a certain encoding scheme (namely as intersections of boxes in a d-dimensional space). This parameter is not related to edge modification problem. [1, 2] show a nice general result: for any family $\mathcal{F}$ of bounded boxicity, taking as parameter the number $k = k_1 + k_2$ of edge modifications in order to obtain a graph in $\mathcal{F}$, that is $G \in \mathcal{F} + k_1 e - k_2 e$, they design an $O(2^{k^2 \log k})$ FPT approximation algorithm (with a small additive constant) for computing the boxicity of G. This gives as a particular case, the same result using as a parameter edit distances such as interval edge completion, proper interval edge deletion, planar edge deletion. They are also able to retrieve results that are not parameterized by edge edit distances, such as the results in [3] that are parameterized by Minimum Vertex Cover, Feedback Vertex Set, Max Leaf Number (maximum number possible of leaves in a spanning tree of the graph).

Again, these results can be understood as using the proximity with a class $\mathcal{C}$ having a good behaviour with regard to the considered problem, here having bounded boxicity, in order to extend this nice behaviour in a parameterized way to graphs that almost belong to $\mathcal{C}$, taking as a parameter the edit distance to $\mathcal{C}$.

3.3 Coloring

As mentionned earlier, the results presented in Section 3 so far can be seen as extending the good behaviour of a class of graphs $\mathcal{C}$ with regard to some problem to the graphs that are almost in $\mathcal{C}$, via an algorithm parameterized by the distance to $\mathcal{C}$. The best example of this general approach is provided by the coloring problem on almost chordal graphs.

[18] shows that given a modulator of the graph (i.e. a set of modifications to be performed in order to reach the target class), a graph G in $\mathcal{F} - ke$, where $\mathcal{F}$ is closed under edge contraction and admits a polynomial time algorithm to compute an optimal coloring, an optimal coloring of G can be found in FPT time. This implies, since the problem of finding a modulator of a chordal$-ke$ graph is FPT, that coloring a chordal$-ke$ graph is FPT, even without a modulator given a priori. A similar result is shown for $\mathcal{F}+ke$ if the chromatic polynomial can be computed in polynomial time on $\mathcal{F}$, which is not the case for chordal graphs. Finally, [18] shows that $split + k_1 e - k_2 e$ with $k_1 + k_2 \leq k$ can also be colored in FPT time.

In [54], these results are completed in the following manner: it is shown that coloring is FPT on chordal+ke graphs, that is graphs where there exist at most k edges whose deletion results in a chordal graph. Again, the modulator, i.e. the edges to be removed, are supposed given as part of the output. Nevertheless, this supposition is not a limitation since it was shown later that finding such a set of edges is FPT [53, 24]. The algorithm of [54] uses a dynamic programming scheme along the rooted clique tree of the underlying chordal graph, where the additional edges are considered. The clique tree has been previously transformed into a nice tree where each node is binary and corresponds to only one of the three following operations: join, loss of a vertex, gain of a vertex. Moreover, the graph is modified so that additional edges are pairwise non-incident. Two colorings of the children of one node can then be combined iff these two colorings don't assign the same color to the two vertices incident to one additional edge (called special vertex). This is encoded in each node by keeping the list of all possible colorings, described only by a set of couples encoding which color is assigned to each special vertex. Thus,

two colorings are compatible iff the set of couples describing them are disjoint. The main tool to get a FPT algorithm is to reduce the size of the list of colorings which is kept in each node. As we are only interested in checking whether there exists another set (from the other node) which is disjoint with one set of this node, and since all the sets have size at most $2k$, then it is possible to keep a list of representative sets of length at most $2^{4k}poly(n)$, with the property that the existence of such a disjoint couple of sets is equivalent to the same existence in the original list. In addition, these representative lists can be composed together to get the representative list of the parent of two nodes. Therefore the dynamic programming scheme along the tree, from the leaves to the root, is shown to be FPT in k.

4 Conclusion

We reviewed results on FPT algorithms parameterized by edge edit distance on graphs. It appears that the overwhelming majority of such algorithms precisely consider the problem of computing this edit distance, which stands both for the parameter and the quantity to compute. They answer the question "is the input graph at distance at most k from class $\mathcal{C}$?" To this regard, they can be thought of as recognition algorithms for the graphs almost in class $\mathcal{C}$. Most of the algorithms presented in this survey are of this type.

Nevertheless, there are also a few example where the parameter used is the edit distance to some class $\mathcal{C}$ of graphs but where the problem considered is independant from this edit distance. The idea exploited in such cases is to extend the good properties of class $\mathcal{C}$ with regard to the problem considered to the graphs that are not in $\mathcal{C}$ but almost, in a parameterized manner. It is also important to note that at the base of such algorithms lies the assumption that one does not only know that the graph is close to class $\mathcal{C}$ but also knows a modulator for it, that is a set of modifications that turn the input graph into a graph of class $\mathcal{C}$. This is precisely the goal of the algorithms refered to above as recognition algorithms for graphs almost in $\mathcal{C}$. Indeed, when such algorithms answer yes, they usually output a modulator. The reason for this is that they usually answer the yes/no question by trying to build a modulator of size k until they either succeed or come to a state of their execution that can stand for a proof that no such modulator exists. Therefore, these recognition algorithms are the corner stone of the general approach that tries to extend the good algorithmic properties of a class to graphs almost in this class.

Finally, this survey intentionally focussed on edge modifications and that it therefore excluded the algorithms parameterized by vertex modifications, such as the number of vertices to be deleted in order to obtain a graph in $\mathcal{C}$. Let us note that it seems that the question of solving problems parameterized by such vertex deletion distance to some (usually very simple) class of graphs has also received a great amount of attention.

References

1. Adiga, A., Babu, J., Chandran, L.S.: Parameterized and approximation algorithms for boxicity. CoRR abs/1201.5958 (2012), http://arxiv.org/abs/1201.5958
2. Adiga, A., Babu, J., Chandran, L.S.: Polynomial time and parameterized approximation algorithms for boxicity. In: Thilikos, D.M., Woeginger, G.J. (eds.) Parameterized and Exact Computation. pp. 135–146. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Adiga, A., Chitnis, R., Saurabh, S.: Parameterized algorithms for boxicity. In: Cheong, O., Chwa, K.Y., Park, K. (eds.) Algorithms and Computation. pp. 366–377. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
4. Alon, N., Lokshtanov, D., Saurabh, S.: Fast fast. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) Automata, Languages and Programming. pp. 49–58. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
5. Aravind, N., Sandeep, R., Sivadasan, N.: Dichotomy results on the hardness of h-free edge modification problems. SIAM Journal on Discrete Mathematics 31(1), 542–561 (2017)

6. Aravind, N.R., Sandeep, R.B., Sivadasan, N.: On polynomial kernelization of $\mathcal{H}$-free edge deletion. Algorithmica 79(3), 654–666 (Nov 2017)
7. Bessy, S., Paul, C., Perez, A.: Polynomial kernels for 3-leaf power graph modification problems. Discrete Applied Mathematics 158(16), 1732 – 1744 (2010), http://www.sciencedirect.com/science/article/pii/S0166218X10002416
8. Bessy, S., Perez, A.: Polynomial kernels for proper interval completion and related problems. Information and Computation 231, 89 – 108 (2013), http://www.sciencedirect.com/science/article/pii/S0890540113000837, fundamentals of Computation Theory
9. Bliznets, I., Fomin, F., Pilipczuk, M., Pilipczuk, M.: A subexponential parameterized algorithm for proper interval completion. SIAM Journal on Discrete Mathematics 29(4), 1961–1987 (2015)
10. Bliznets, I., Cygan, M., Komosa, P., Mach, L., Pilipczuk, M.: Lower bounds for the parameterized complexity of Minimum Fill-In and other completion problems, pp. 1132–1151. ACM (2016)
11. Bliznets, I., Cygan, M., Komosa, P., Pilipczuk, M.: Hardness of approximation for h-free edge modification problems. ACM Trans. Comput. Theory 10(2), 9:1–9:32 (May 2018), http://doi.acm.org/10.1145/3196834
12. Bliznets, I., Fomin, F.V., Pilipczuk, M., Pilipczuk, M.: Subexponential parameterized algorithm for interval completion. In: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1116–1131. SODA '16, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2016)
13. Böcker, S., Bui, Q.B.A., Truss, A.: Improved fixed-parameter algorithms for minimum-flip consensus trees. ACM Trans. Algorithms 8(1), 7:1–7:17 (Jan 2012), http://doi.acm.org/10.1145/2071379.2071386
14. Brgmann, D., Komusiewicz, C., Moser, H.: On generating triangle-free graphs. Electronic Notes in Discrete Mathematics 32, 51 – 58 (2009), http://www.sciencedirect.com/science/article/pii/S1571065309000092, dIMAP Workshop on Algorithmic Graph Theory
15. Bcker, S., Briesemeister, S., Bui, Q., Truss, A.: Going weighted: Parameterized algorithms for cluster editing. Theoretical Computer Science 410(52), 5467 – 5480 (2009), http://www.sciencedirect.com/science/article/pii/S0304397509003521, combinatorial Optimization and Applications
16. Bcker, S., Damaschke, P.: Even faster parameterized cluster deletion and cluster editing. Information Processing Letters 111(14), 717 – 721 (2011), http://www.sciencedirect.com/science/article/pii/S0020019011001232
17. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters 58(4), 171 – 176 (1996), http://www.sciencedirect.com/science/article/pii/0020019096000506
18. Cai, L.: Parameterized complexity of vertex colouring. Discrete Applied Mathematics 127(3), 415 – 429 (2003), http://www.sciencedirect.com/science/article/pii/S0166218X02002421
19. Cai, L., Cai, Y.: Incompressibility of h-free edge modification problems. Algorithmica 71(3), 731–757 (Mar 2015)
20. Cai, Y.: Polynomial kernelisation of H-free edge modification problems. Mphil thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong (2012)
21. Cao, Y.: An efficient branching algorithm for interval completion. CoRR abs/1306.3181 (2013), http://arxiv.org/abs/1306.3181
22. Cao, Y.: Linear recognition of almost interval graphs. In: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1096–1115. SODA '16, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2016), http://dl.acm.org/citation.cfm?id=2884435.2884512
23. Cao, Y., Chen, J.: Cluster editing: Kernelization based on edge cuts. Algorithmica 64(1), 152–169 (Sep 2012)
24. Cao, Y., Marx, D.: Chordal editing is fixed-parameter tractable. Algorithmica 75(1), 118–137 (May 2016)
25. Cao, Y., Rai, A., Sandeep, R.B., Ye, J.: A polynomial kernel for diamond-free editing. CoRR abs/1803.03358 (2018), http://arxiv.org/abs/1803.03358
26. Cao, Y., Sandeep, R.B.: Minimum Fill-In: Inapproximability and Almost Tight Lower Bounds, pp. 875–880. ACM (2017)
27. Chen, D., Eulenstein, O., Fernandez-Baca, D., Sanderson, M.: Minimum-flip supertrees: complexity and algorithms. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(2), 165–173 (April 2006)
28. Chen, J., Meng, J.: A 2k kernel for the cluster editing problem. Journal of Computer and System Sciences 78(1), 211 – 220 (2012), http://www.sciencedirect.com/science/article/pii/S0022000011000468, jCSS Knowledge Representation and Reasoning
29. Cygan, M., Pilipczuk, M., Pilipczuk, M., van Leeuwen, E.J., Wrochna, M.: Polynomial kernelization for removing induced claws and diamonds. Theory of Computing Systems 60(4), 615–636 (May 2017)
30. Drange, P.G., Dregi, M.S., Lokshtanov, D., Sullivan, B.D.: On the threshold of intractability. In: Bansal, N., Finocchi, I. (eds.) Algorithms - ESA 2015. pp. 411–423. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
31. Drange, P.G., Fomin, F.V., Pilipczuk, M., Villanger, Y.: Exploring the subexponential complexity of completion problems. ACM Trans. Comput. Theory 7(4), 14:1–14:38 (Aug 2015), http://doi.acm.org/10.1145/2799640

32. Drange, P.G., Pilipczuk, M.: A polynomial kernel for trivially perfect editing. Algorithmica (Dec 2017)
33. Feige, U.: Faster fast(feedback arc set in tournaments). CoRR abs/0911.5094 (2009), http://arxiv.org/abs/0911.5094
34. Fellows, M., Langston, M., Rosamond, F., Shaw, P.: Efficient parameterized preprocessing for cluster editing. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) Fundamentals of Computation Theory. pp. 312–321. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
35. Fellows, M.R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Graph-based data clustering with overlaps. Discrete Optimization 8(1), 2 – 17 (2011), http://www.sciencedirect.com/science/article/pii/S1572528610000642, parameterized Complexity of Discrete Optimization
36. Fomin, F., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in. SIAM Journal on Computing 42(6), 2197–2216 (2013)
37. Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight bounds for parameterized complexity of cluster editing with a small number of clusters. Journal of Computer and System Sciences 80(7), 1430 – 1447 (2014), http://www.sciencedirect.com/science/article/pii/S0022000014000592
38. Ghosh, E., Kolay, S., Kumar, M., Misra, P., Panolan, F., Rai, A., Ramanujan, M.S.: Faster parameterized algorithms for deletion to split graphs. Algorithmica 71(4), 989–1006 (Apr 2015)
39. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. Theory of Computing Systems 38(4), 373–392 (Jul 2005)
40. Guillemot, S., Havet, F., Paul, C., Perez, A.: On the (non-)existence of polynomial kernels for p_l-free edge modification problems. Algorithmica 65(4), 900–926 (Apr 2013)
41. Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: A more relaxed model for graph-based data clustering: s-plex cluster editing. SIAM Journal on Discrete Mathematics 24(4), 1662–1683 (2010)
42. Guo, J.: Problem kernels for np-complete edge deletion problems: Split and related graphs. In: Tokuyama, T. (ed.) Algorithms and Computation. pp. 915–926. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
43. Guo, J.: A more effective linear kernelization for cluster editing. Theoretical Computer Science 410(8), 718 – 726 (2009), http://www.sciencedirect.com/science/article/pii/S0304397508007822
44. Guo, J., Hüffner, F., Komusiewicz, C., Zhang, Y.: Improved algorithms for bicluster editing. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) Theory and Applications of Models of Computation. pp. 445–456. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
45. Kaplan, H., Shamir, R., Tarjan, R.: Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. SIAM Journal on Computing 28(5), 1906–1922 (1999)
46. Karpinski, M., Schudy, W.: Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In: Cheong, O., Chwa, K.Y., Park, K. (eds.) Algorithms and Computation. pp. 3–14. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
47. Komusiewicz, C., Uhlmann, J.: Cluster editing with locally bounded modifications. Discrete Applied Mathematics 160(15), 2259 – 2270 (2012), http://www.sciencedirect.com/science/article/pii/S0166218X12002259
48. Komusiewicz, C., Uhlmann, J.: A cubic-vertex kernel for flip consensus tree. Algorithmica 68(1), 81–108 (Jan 2014)
49. Kratsch, S., Wahlstrm, M.: Two edge modification problems without polynomial kernels. Discrete Optimization 10(3), 193 – 199 (2013), http://www.sciencedirect.com/science/article/pii/S1572528613000030
50. Liu, Y., Wang, J., Guo, J., Chen, J.: Complexity and parameterized algorithms for cograph editing. Theoretical Computer Science 461, 45 – 54 (2012), http://www.sciencedirect.com/science/article/pii/S0304397511009595, 17th International Computing and Combinatorics Conference (COCOON 2011)
51. Liu, Y., Wang, J., Xu, C., Guo, J., Chen, J.: An effective branching strategy based on structural relationship among multiple forbidden induced subgraphs. Journal of Combinatorial Optimization 29(1), 257–275 (Jan 2015)
52. Mancini, F.: Minimum fill-in and treewidth of split+ke and split+kv graphs. Discrete Applied Mathematics 158(7), 747 – 754 (2010), http://www.sciencedirect.com/science/article/pii/S0166218X08005076, third Workshop on Graph Classes, Optimization, and Width Parameters Eugene, Oregon, USA, October 2007
53. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica 57(4), 747–768 (Aug 2010)
54. Marx, D.: Parameterized coloring problems on chordal graphs. Theoretical Computer Science 351(3), 407 – 424 (2006), http://www.sciencedirect.com/science/article/pii/S030439750500633X, parameterized and Exact Computation
55. Nastos, J., Gao, Y.: Bounded search tree algorithms for parametrized cograph deletion: Efficient branching rules by exploiting structures of special graph classes. Discrete Math., Alg. and Appl. 4 (2012)
56. Sandeep, R.B., Sivadasan, N.: Parameterized Lower Bound and Improved Kernel for Diamond-free Edge Deletion. In: Husfeldt, T., Kanj, I. (eds.) 10th International Symposium on Parameterized and Exact Computation (IPEC 2015). Leibniz International Proceedings in Informatics (LIPIcs), vol. 43, pp. 365–376. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2015), http://drops.dagstuhl.de/opus/volltexte/2015/5597

57. Villanger, Y., Heggernes, P., Paul, C., Telle, J.: Interval completion is fixed parameter tractable. SIAM Journal on Computing 38(5), 2007–2020 (2009)
58. Xia, G., Zhang, Y.: Kernelization for cycle transversal problems. Discrete Applied Mathematics 160(7), 1224 – 1231 (2012), http://www.sciencedirect.com/science/article/pii/S0166218X12000029