Proc. IWPEC-04, Vol. 3162 in LNCS, pp. 162-173, Springer, 2004

A Structural View on Parameterizing Problems: Distance from Triviality*

Jiong Guo, Falk Hüffner, and Rolf Niedermeier

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13,
D-72076 Tübingen, Fed. Rep. of Germany
{guo,hueffner,niedermr}@informatik.uni-tuebingen.de

Abstract. Based on a series of known and new examples, we propose the generalized setting of "distance from triviality" measurement as a reasonable and prospective way of determining useful structural problem parameters in analyzing computationally hard problems. The underlying idea is to consider tractable special cases of generally hard problems and to introduce parameters that measure the distance from these special cases. In this paper we present several case studies of distance from triviality parameterizations (concerning Clique, Power Dominating Set, Set Cover, and Longest Common Subsequence) that exhibit the versatility of this approach to develop important new views for computational complexity analysis.

1 Introduction

Vertex Cover is one of the NP-complete problems that stood at the cradle of parameterized algorithm design and analysis [11]. Given an undirected graph with n vertices and a nonnegative integer k, the question is whether we can find a set of at most k graph vertices such that each graph edge has at least one of its endpoints in this set. The currently best fixed-parameter algorithms exactly solve Vertex Cover in $O(1.3^k + kn)$ time [8,22]; that is, Vertex Cover is fixed-parameter tractable when parameterized by k. A different way of parameterizing Vertex Cover is to consider the structure of the input graph. If the given graph allows for a tree decomposition [26,4] of width w, then it is well-known that Vertex Cover can be solved in $O(2^w \cdot n)$ time [30] independent of the size k of the cover set we are searching for. Hence, Vertex Cover is also fixed-parameter tractable when parameterized by w. As a rule, most problems can be parameterized in various reasonable ways.[1] The example Vertex Cover exhibits two fundamentally different ways of parameterization—"parameterizing by size" (i.e., the size of the vertex cover) and "parameterizing by structure" (i.e., the treewidth of the underlying graph). In this paper we propose to take

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

[1] For instance, Fellows [13] discusses how to parameterize the Max Leaf Spanning Tree problem in at least five different ways.

a broader, generalized view on parameterizing problems by structure, leading to a generic framework of new research questions in parameterized complexity analysis.

The leitmotif of parameterized complexity theory [11] is to gain a better understanding of problem hardness through a refined complexity analysis that uses a two-dimensional view on problems by means of parameterization. A natural way to do this is as follows. Consider a problem such as Vertex Cover and find out what efficiently solvable special cases there are known. For instance, Vertex Cover is trivially solvable on trees. Now, for example, consider the parameter d defined as the number of edges that have to be deleted from a graph to transform it into a tree. In this sense parameter d measures the "distance from triviality" and one may ask whether Vertex Cover is fixed-parameter tractable when parameterized by d. In this simple example the answer is clearly "yes" because such a graph has treewidth bounded by $d + 1$ [3] and, thus, Vertex Cover can be solved using the tree decomposition approach [30]. But in other cases this "distance from triviality" approach to parameterization often leads to interesting new research questions: For instance, in a recent work Hoffmann and Okamoto [19] describe a fixed-parameter algorithm for the Traveling Salesman Problem in the two-dimensional Euclidean plane based on the following distance from triviality parameterization: Consider a set of n points in the Euclidean plane. Determine their convex hull. If all points lie on the hull, then this gives the shortest tour. Otherwise, Hoffmann and Okamoto show that the problem is solvable in $O(k! \cdot k \cdot n)$ time where k denotes the number of points inside the convex hull. Thus, the distance from triviality here is the number k of inner points.

In this paper we extend the distance from triviality concept to a broader setting and we discuss further examples for the fruitfulness of this new parameterization methodology. We present further recent examples from the literature concerning Satisfiability [28] and Graph Coloring [7] that fit into our framework. In addition, we provide four new fixed-parameter tractability results using this framework for Clique, Power Dominating Set, Set Cover, and Longest Common Subsequence. Given all these case studies, we hope to convince the reader that, in a sense, "parameterizing away from triviality" yields a generic framework for an extended parameterized complexity analysis to better understand computational (in)tractability. Further aspects of our scenario and its prospects for future research are discussed in the concluding section.

2 Preliminaries and Previous Work

Preliminaries. Parameterized complexity theory [11] offers a two-dimensional framework for studying the computational complexity mostly of NP-hard problems. A *parameterized language* (problem) L is a subset $L \subseteq \Sigma^* \times \Sigma^*$ for some finite alphabet Σ. For $(x, k) \in L$, by convention, the second component denotes the *parameter*. The two dimensions of parameterized complexity analysis are constituted by the input size $n := |(x, k)|$ and the parameter value k (usually

a nonnegative integer). A parameterized language is *fixed-parameter tractable* if it can be determined in $f(k) \cdot n^{O(1)}$ time whether $(x, k) \in L$, where f is a computable function only depending on k. Since the parameter k represents some aspect(s) of the input or the solution, there usually are many meaningful ways to *parameterize* a problem. An important issue herein is whether a problem is fixed-parameter tractable with respect to a chosen parameter or not (i.e., W[1]-hard, see [11] for details), and, in case of fixed-parameter tractability, how small the usually exponential growth of the function f can be kept. Hence, investigating different parameterizations gives insight into what causes the computational (in)tractability of a problem and in which qualitative and quantitative sense this happens. Refer to [10,14,21] for recent surveys on parameterized complexity.

Previous Work. The aim of this paper is to stimulate research on the structural parameterization "distance from triviality." Clearly, one of the most sophisticated examples in this context is the notion of bounded treewidth developed by Robertson and Seymour [26]. Without going into details, we remark that the basic motivation for considering this concept can be derived from the fact that many NP-hard graph problems (such as Vertex Cover) become easy (linear-time solvable) on trees. Treewidth then measures how tree-like a graph is, and if this parameter is small, then many otherwise hard graph problems can be solved efficiently (see [4] for a survey). In this sense treewidth measures the distance from the triviality "tree" and problems such as Vertex Cover are fixed-parameter tractable with respect to this structural parameter [30].

Another prominent problem is Graph Coloring. Leizhen Cai [7] recently initiated a study of Graph Coloring which falls into our framework. For instance, considering split graphs (where Graph Coloring is solvable in polynomial time) he showed that Graph Coloring is fixed-parameter tractable with respect to parameter k on graphs that originate from split graphs when adding or deleting k edges. By way of contrast, it is W[1]-hard when deletion of k vertices leads to a split graph. Interestingly, the problem is much harder in case of bipartite graphs instead of split graphs: Graph Coloring becomes NP-complete for graphs that originate from bipartite graphs by adding three edges or if two vertex deletions are needed to make a graph bipartite. In summary, Cai states that "this new way of parameterizing problems adds a new dimension to the applicability of parameterized complexity theory" [7].[2]

Finally, to emphasize that not only graph problems fall into our framework we give an example with Satisfiability. It is easy to observe that a boolean formula in conjunctive normal form which has a matching between variables and clauses that matches all clauses is always satisfiable. For a formula F, considered as a set of m clauses over n variables, define the *deficiency* as $\delta(F) := m - n$. The maximum deficiency is $\delta^*(F) := \max_{F' \subseteq F} \delta(F')$. Szeider shows that the satisfiability of a formula F can be decided in $O(2^{\delta^*(F)} \cdot n^3)$ time [28]. Note that

[2] Juedes et al. [20] show that coloring an n-vertex-graph with $n - k$ colors is fixed-parameter tractable with respect to k. Clearly, $k = 0$ is trivial. This parameterization, however, is not a structural one.

a formula F with $\delta^*(F) = 0$ has a matching as described above. Again, $\delta^*(F)$ is a structural parameter measuring the distance from triviality in our sense.

In the following sections we provide new case studies for the applicability of the distance from triviality concept in various contexts. Clearly, it is conceivable that several other examples from the literature will fit as examples into our concept.[3] An important point, however, is that all of the parameterizations discussed here have nothing to do with the solution itself (i.e., the value to be determined or optimized). Our parameterizations are structural ones.

3 Case Study Clique

The Clique problem is defined as follows:

Input: A graph $G = (V, E)$ and a nonnegative integer s.
Question: Does G contain a *clique*, i.e., a complete subgraph, of size s?

Clique is W[1]-complete with respect to the natural parameter s [11]. It is also hard to approximate to an arbitrary constant factor. Here we exhibit fixed-parameter tractability with respect to the distance from a trivial case.

Our trivial case is the class of *cluster graphs*: graphs which are a disjoint union of cliques. Clique can be trivially solved in linear time on such graphs. We examine Clique on graphs which are "almost" cluster graphs, namely, on graphs which are cluster graphs with k edges added. From a general result on graph modification problems by Leizhen Cai [6] it follows that finding the added k edges is fixed-parameter tractable with respect to k. Improved algorithms for this problem (which is known as Cluster Deletion) were given by Gramm et al. [15,16], providing an algorithm running in $O(1.53^k + |V|^3)$ time.

It remains to show how to solve Clique for the "almost cluster graph" G after identifying the k added edges and the corresponding cluster graph G'. If the largest clique in G is not one which is already contained in G', then each of its vertices must have gained in degree by at least one compared to G'. This means it can only be formed by a subset of the up to $2k$ vertices "touched" by the added edges. Hence, we solve Clique for the subgraph of G induced by the up to $2k$ vertices which are endpoints of the added edges. This step can be done for example by using Robson's algorithm for Independent Set [27] on the complement graph in $O(1.22^{2k}) = O(1.49^k)$ time, which is dominated by the above time bound for the Cluster Deletion subproblem. The largest clique for G is simply the larger of the clique found this way and the largest clique in G'. We obtain the following theorem:

Theorem 1. Clique *for a graph $G = (V, E)$ which is a cluster graph with k edges added can be solved in $O(1.53^k + |V|^3)$ time.*

[3] For instance, Nishimura et al. [23] developed algorithms for recognizing general classes of graphs generated by a base graph class by adding at most k vertices. Their fixed-parameter tractability studies are closely related to our methodology.

4 Case Study Power Dominating Set

Domination in graphs is among the most important problems in combinatorial optimization. We consider here the Power Dominating Set problem [18], which is motivated from applications in electric networks. The task is to place monitoring devices (so-called *PMUs*) at vertices such that all edges and vertices are *observed*. The rules for observation are:

1. A PMU in a vertex v observes v and all incident edges and neighbors of v.
2. Any vertex that is incident to an observed edge is observed.
3. Any edge joining two observed vertices is observed.
4. If a vertex is incident to a total of $i > 1$ edges and if $i - 1$ of these edges are observed, then all i edges are observed. This rule is reminiscent of Kirchhoff's current law from electrical engineering.

We can now formulate the Power Dominating Set problem:

Input: A graph $G = (V, E)$ and a nonnegative integer k.
Question: Does G have a *power dominating set* of size at most k, that is, a subset $M \subseteq V$ of vertices such that by placing a PMU in every $v \in M$, all vertices in V are observed?

Power Dominating Set is NP-complete [18]. There is an algorithm known which solves Power Dominating Set in linear time on trees [18]. Since we use this algorithm as a building block for our result, we briefly sketch how it proceeds. This algorithm works bottom-up from the leaves and places a PMU in every vertex which has at least two unobserved children. Then it updates observation according to the four observation rules and prunes completely observed subtrees, since they no longer affect observability of other vertices.

Our goal is now to find an efficient algorithm for input graphs that are "nearly" trees. More precisely, we aim for a fixed-parameter algorithm for graphs which are trees with k edges added.

Note that a tree with k edges added has treewidth bounded by $k+1$ [3]. While Dominating Set is fixed-parameter tractable with respect to the parameter treewidth [2], no such result is currently known for Power Dominating Set. This motivates our subsequent result.

As a first step we present a simple algorithm with quadratic running time for the case of one single added edge.

Lemma 1. Power Dominating Set *for a graph $G = (V, E)$ with $n := |V|$ which is a tree with one edge added can be solved in $O(n^2)$ time.*

Proof. Graph G contains exactly one cycle and a collection of trees T_i touching the cycle at their roots.

We use the above mentioned linear time algorithm to find an optimal solution for each T_i. When it reaches the root r_i, several cases are possible:

– The root r_i needs to be in M, and we can remove it. This breaks the cycle, and we can solve the remaining instance in linear time.

- The root r_i is not in M, but already observed. Then all children of r_i in T_i except for at most one are observed, or we would need to take r_i into M. Then, we can remove T_i except for r_i and except for the unobserved child, if it exists. The root r_i remains as an observed degree-2 or degree-3 vertex on the cycle.
- The root r_i still needs to be observed. This is only possible if it has exactly one child in T_i which is unobserved since otherwise r_i either would be in M, or be observed. As in the previous case, we keep r_i and the unobserved child, and the rest of T_i can again be removed.

If after these data reductions two observed vertices are adjacent on the cycle, their connecting edge becomes observed, and we can break the cycle. Otherwise, we call it a *reduced cycle*.

At least one vertex on the reduced cycle has to be added to M. We simply try each vertex. After each choice, the rest of the cycle decomposes into a tree after pruning observed edges, and can be handled in linear time. From all possible initial choices, we keep the one leading to a minimal M, which then is an optimal choice for the initial problem. Since there are $O(n)$ vertices and edges on the cycle, this takes $O(n^2)$ time. □

We note without proof that Lemma 1 can be improved to linear time by examining a fixed-size segment of the cycle. In each possible case we can determine at least one vertex in the segment which has to be taken into M.

Lemma 1 is applicable whenever each vertex is part of at most one cycle. We now generalize this and Haynes et al.'s [18] result.

Theorem 2. Power Dominating Set *for a graph which is a tree with k edges added is fixed-parameter tractable with respect to k.*

Proof. We first treat all trees which are attached in single points to cycles as in the proof of Lemma 1. What remains are degree-2 vertices, degree-3 vertices with a degree-1 neighbor, and other vertices of degree 3 or greater, the *joints*. For a vertex v, let $\deg v$ denote its degree, that is, the number of its adjacent vertices. We branch into several cases for each joint:

- The joint v is in M. We can prune it and its incident edges.
- The joint v is not in M. Note that the only effect v can still have is that a neighbor of v becomes observed from application of observation rule 4 ("Kirchhoff's current law") applied to v. We branch further into $\deg v \cdot (\deg v - 1)$ cases for each pair (w_1, w_2) of neighbors of v with $w_1 \neq w_2$. In each branch, we omit the edges between v and all neighbors of v except w_1 and w_2. Clearly any solution of such an instance provides a solution for the unmodified instance. Furthermore, it is not too hard to show that if the unmodified instance has a solution of size s, then on at least one branch we will also find a solution of size s. To see this, consider a solution M for the unmodified problem. Vertex v is observed; this can only be either because a neighbor w_1 of v was put into M, or because there is a neighbor w_1 of v

such that the edge $\{w_1, v\}$ became observed from observation rule 4. Furthermore, as mentioned, there can be at most one vertex w_2 which becomes observed by observation rule 4 applied to v. Then M is also a valid solution for the branch corresponding to the pair (w_1, w_2).

In each of the less than $(\deg v)^2$ branches we can eliminate the joint. If we branch for all joints in parallel, we end up with an instance where every connected component is a tree or a cycle with attached degree-1 vertices, which can be solved in linear time. The number of cases to distinguish is $\prod_{v \text{ is joint}} (\deg v)^2$. Since there are at most $2k$ joints, each of maximum degree k, the total running time is roughly bounded by $O(n \cdot 2^{4k \log k})$, confirming fixed-parameter tractability. □

5 Case Study Tree-like Weighted Set Cover

Set Cover is one of the most prominent NP-complete problems. Given a base set $S = \{s_1, s_2, \ldots, s_n\}$ and a collection C of subsets of S, $C = \{c_1, c_2, \ldots, c_m\}$, $c_i \subseteq S$ for $1 \leq i \leq m$, and $\bigcup_{1 \leq i \leq m} c_i = S$, the task is to find a subset C' of C with minimal cardinality which covers all elements in S, i.e., $\bigcup_{c \in C'} c = S$. Assigning weights to the subsets and minimizing the total weight of the collection C' instead of its cardinality, one naturally obtains the Weighted Set Cover problem. We call C' the *minimum set cover* of S resp. the *minimum weight set cover*. We define the *occurrence* of an element $s \in S$ in C as the number of subsets in C which contain s. Set Cover remains NP-complete even if the occurrence of each element is bounded by 2 [24].

Definition 1 (Tree-like subset collection).
Given a base set $S = \{s_1, s_2, \ldots, s_n\}$ and a collection C of subsets of S, $C = \{c_1, c_2, \ldots, c_m\}$. We say that C is a tree-like *subset collection of S if we can organize the subsets in C in an unrooted tree T such that every subset one-to-one corresponds to a node of T and, for each element $s_j \in S$, $1 \leq j \leq n$, all nodes in T corresponding to the subsets containing s_j induce a subtree of T.*

We call T the underlying *subset tree* and the property of T that, for each $s \in S$, the nodes containing s induce a subtree of T, is called the "*consistency property*" of T. Observe that the consistency property is also of central importance in Robertson and Seymour's famous notion of tree decompositions of graphs [26,4]. By results of Tarjan and Yannakakis [29], we can test whether a subset collection is a tree-like subset collection and, if so, we can construct a subset tree for it in linear time. Therefore, in the following we always assume that the subset collection is given in form of a subset tree. For convenience, we denote the nodes of the subset tree by their corresponding subsets.

Here, we consider the Tree-like Weighted Set Cover (TWSC) problem with bounded occurrence which is defined as follows:

Tree-like Weighted Set Cover with bounded occurrence:
Input: Given a base set $S = \{s_1, s_2, \ldots, s_n\}$ and a tree-like collection C

of subsets of S, $C = \{c_1, c_2, \ldots, c_m\}$. Each element of S can be in at most d subsets for a fixed $d \geq 1$. Each subset in C has a positive real weight $w(c_i) > 0$ for $1 \leq i \leq m$. The weight of a subset collection is the sum of the weights of all subsets in it.
Task: Find $C' \subseteq C$ with minimum weight which covers all elements in S, i.e., $\bigcup_{c \in C'} c = S$.

TWSC with bounded occurrence $d \geq 3$ is NP-complete even if the underlying subset tree is a star [17]. However, it can be solved in $O(m^2 n)$ time if the underlying subset tree is a path [17]. Now our goal is, based on the "trivial" path-like case, to give a fixed-parameter algorithm for TWSC with bounded occurrence where the number of leaves of the subset tree functions as the distance parameter from the path-like case.

The fixed-parameter algorithm. Given a subset tree T with k leaves, the following observations are easy to prove.

Observation 1. The maximum degree of the nodes of T is upperbounded by k.
Observation 2. The number of tree nodes with more than 2 neighbors is upperbounded by k.
Observation 3. For each $c \in C$, the number of subsets $c' \in C$ which share some elements with c, i.e., $c \cap c' \neq \emptyset$, is upperbounded by $d \cdot k$. These subsets and c induce a subtree of T.

For a subset $c \in C$, let $\deg c$ denote the degree of its corresponding node in T. The basic idea of the algorithm is to divide the tree-like instance into several "independent" path-like instances and to solve these instances separately by using the $O(m^2 n)$-time algorithm. Instances are independent if they share no common elements with each other. For each tree node c with $\deg c \geq 3$, we construct a set $H_c := \{c' \mid c \cap c' \neq \emptyset\}$. Note that $|H_c| \leq d \cdot k$ by Observation 3. To cover all elements of c, we have to add some subsets from H_c into the set cover. We delete the subsets added into the set cover and all their adjacent edges from T. Furthermore, we delete elements of S which are already covered from all remaining subsets. Observe that if c is in the set cover, we retain several subtrees of T after deleting c; otherwise, c is now an empty subset. By deleting all empty subsets, the subset tree T is divided into several subtrees. Due to the consistency property of T, all these resulting subtrees are independent. Since the possible combinations of the subsets from H_c which cover all elements of c are upperbounded by 2^{dk}, we can have up to 2^{dk} new instances by dividing T at c. By processing all nodes c with $\deg c \geq 3$ in the same way, there are $O(2^{dk^2})$ new instances each of which consists of $O(m)$ independent path-like instances. Then the minimum set cover for one of these new instances is the union of the optimal solutions for the path-like instances with the subsets already added into the set cover while processing the nodes with degree at least 3. In summary, we get the following theorem:

Theorem 3. *TWSC with occurrence bounded by d can be solved in $O(2^{dk^2} \cdot m^2 n)$ time, where k denotes the number of the leaves of the subset tree.*

Note that while results from [17] only cover cases with bounded subset size, we impose no such restriction here. However, here we need the bounded occurrence restriction.

6 Case Study Longest Common Subsequence

In this section we deal with the Longest Common Subsequence problem, an important problem in theoretical computer science and computational biology.

> Longest Common Subsequence (LCS):
> **Input:** Given a set of k strings $X_1, X_2, \ldots, X_k$ over an alphabet Σ and a positive integer m.
> **Question:** Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, \ldots, k$?

LCS is NP-complete even if $|\Sigma| = 2$. Concerning the parameterized complexity of LCS with unbounded alphabet size, Bodlaender et al. [5] showed that LCS is W[t]-hard for $t \geq 1$ with k as parameter, W[2]-hard with m as parameter, and W[1]-hard with k and m as parameters. With a fixed alphabet size, LCS is trivially fixed-parameter tractable with m as parameter, but W[1]-hard with k as parameter [25].

Let n denote the maximum length of the input strings and s_i^a denote the number of occurrences of letter $a \in \Sigma$ in X_i. We consider a new parameterization of LCS with k and $s := \max_{a \in \Sigma} \max_{1 \leq i \leq k} s_i^a$ as parameters. To begin with, we show that the case $s = 1$ of this parameterization, where every letter occurs in each string only once, is solvable in polynomial time.

Without loss of generality, we assume that all input strings have the same length n, $\Sigma = \{1, 2, \ldots, n\}$, and $X_1 = 1\,2\,3 \cdots n$. Then the strings $X_2, X_3, \ldots, X_k$ are permutations of X_1. We construct a directed graph G with $n \times k$ vertices; each vertex represents a position in a string. A directed edge is added from $v_{i,j}$ to $v_{i+1,l}$ for $1 \leq i < k$ iff the letters in position j of X_i and in position l of X_{i+1} are the same. It is easy to observe that a longest common subsequence one-to-one corresponds to a maximum set of directed paths in G which do not cross each other. Two paths P and P' cross each other if there is an edge $(v_{i,j}, v_{i+1,l})$ in P and an edge $(v_{i,j'}, v_{i+1,l'})$ in P' with $j \leq j'$ and $l \geq l'$. In order to find a maximum set of the noncrossing paths in G, we construct a "path-compatibility" graph $PC(G)$ from G: For each directed path in G, we create a vertex P_a in $PC(G)$ where a is the position in X_1 where the path starts. We add a directed edge (P_a, P_b) from P_a to P_b if P_a does not cross P_b and $a < b$. Thus, $PC(G)$ is an acyclic directed graph and a maximum set of noncrossing paths of G one-to-one corresponds to a longest path in $PC(G)$. By using depth-first search for each vertex in $PC(G)$ with in-degree of zero, we can easily find such a longest path.

Concerning the running time to solve LCS on such an instance, we note that graphs G and $PC(G)$ can be constructed in $O(k \cdot n)$ and $O(k \cdot n^2)$ time, respectively. Finding a longest path in $PC(G)$ can be done in $O(n)$ time. Summarizing, the running time for solving LCS on these instances is $O(k \cdot n^2)$.

The fixed-parameter algorithm. Given strings $X_1, X_2, \ldots, X_k$, we construct a graph G with $n \cdot k$ vertices as described above. However, a vertex $v_{i,l}$ with $1 \leq i < k$ has to be connected to all vertices $v_{i+1,h}, 1 \leq h \leq n$, where X_{i+1} has the same letter in position h as X_i in position l. Graph G can be constructed in $O(k \cdot n^2)$ time. Then we construct the path-compatibility graph $PC(G)$ from G. Since the number of paths in G can be up to $\sum_{a \in \Sigma} \prod_{1 \leq i \leq k} s_i^a = O(n \cdot s^k)$, we have $O(n \cdot s^k)$ vertices in $PC(G)$. Each vertex $P_{j_1,j_2,\ldots,j_k}$ represents a path in G, and the indices $j_1, j_2, \ldots, j_k$ denote the positions in the k strings where this path passes. Furthermore, a directed edge from $P_{j_1,j_2,\ldots,j_k}$ to $P_{j'_1,j'_2,\ldots,j'_k}$ is added into $PC(G)$ if the corresponding paths do not cross each other, are vertex-disjoint, and $j_1 < j'_1$. It is easy to verify that by finding a longest path in this acyclic directed graph, we get a longest common subsequence of the input instance. The construction of the edges in $PC(G)$ takes $O(k)$ time per vertex pair, and we obtain an algorithm running in $O(k \cdot (n \cdot s^k)^2) = O(2^{2k \log s} \cdot k \cdot n^2)$ time. This yields the following result.

Theorem 4. Longest Common Subsequence *can be solved in* $O(2^{2k \log s} \cdot k \cdot n^2)$ *time, where* s *denotes the maximum occurrence number of a letter in an input string.*

7 Concluding Discussion

The art of parameterizing problems is of key importance to better understand and cope with computational intractability. In this work we proposed a natural way of parameterizing problems—the parameter measures some distance from triviality. The approach consists of two fundamental steps. Assume that we study a hard problem X.

1. Determine efficiently solvable special cases of X (e.g., in case of graph problems, the restriction to special graph classes)—the triviality.
2. Identify useful distance measures from the triviality (e.g., in case of trees and graphs the treewidth of a graph)—the (structural) parameter.

As to step 2, observe that various distance measures are possible such as edge deletions or vertex deletions in case of graphs. It is important, however, that we can efficiently determine the distance of a given input instance to the chosen triviality with respect to the parameter considered. For instance, it is "easy" to determine the distance of a graph to being acyclic with respect to edge deletion (this leads to the polynomial-time solvable Feedback Edge Set problem) whereas it is hard to determine the distance of a graph to being bipartite with respect to edge deletion (this leads to the NP-hard Graph Bipartization problem). However, in case we are explicitly given the "operations" that transform the given object into a trivial one (e.g., the edges to be deleted to make a graph bipartite), the question for the parameterized complexity of the underlying problem with respect to the distance parameter might still be of interest. In the new case studies presented in this paper the distance measurement for Longest

Common Subsequence was easy to determine whereas in the Clique case the measurement led to an NP-hard but fixed-parameter tractable problem.

We do not claim that all the parameterizations we considered generally lead to small parameter values. This was not the central point, which, by way of contrast, was to extend the range of feasible special cases of otherwise computationally hard problems. As pointed out by an anonymous referee, it would be interesting to study more drastic distance measures such as considering relative distances—for instance, what if 1 % of all edges may be edited in a graph.

It is worth emphasizing that the tractable trivial case may refer to polynomial-time solvability as well as fixed-parameter tractability.[4] An example for the latter case is Dominating Set on planar graphs which is fixed-parameter tractable [1,2]. These results were extended to graphs of bounded genus [12,9], genus here measuring the distance from the "trivial case" (because settled) planar graphs. Moreover, the proposed framework might even be of interest in the approximation algorithms context where triviality then might mean good polynomial-time approximability (e.g., constant factor or approximation scheme). In summary, we strongly believe that distance from triviality parameterization leads to a wide range of prospective research opportunities.

References

1. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. 11
2. J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree technique for Dominating Set on planar graphs. In *Proc. 26th MFCS*, volume 2136 of *LNCS*, pages 111–122. Springer, 2001. 5, 11
3. H. L. Bodlaender. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Dept. of Computer Sci., Utrecht University, 1986. 2, 5
4. H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd MFCS*, volume 1295 of *LNCS*, pages 19–36. Springer, 1997. 1, 3, 7
5. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. T. Wareham. The parameterized complexity of the longest common subsequence problem. *Theoretical Computer Science*, 147:31–54, 1995. 9
6. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996. 4
7. L. Cai. Parameterized complexity of Vertex Colouring. *Discrete Applied Mathematics*, 127(1):415–429, 2003. 2, 3
8. J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further observations and further improvements. *J. Algorithms*, 41:280–301, 2001. 1
9. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded-genus and H-minor-free graphs. In *Proc. 15th SODA*, pages 830–839. SIAM, 2004. 11
10. R. G. Downey. Parameterized complexity for the skeptic. In *Proc. 18th IEEE Annual Conference on Computational Complexity*, pages 147–169, 2003. 3
11. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999. 1, 2, 3, 4

[4] The latter being of particular interest when attacking W[1]-hard problems.

12. J. Ellis, H. Fan, and M. R. Fellows. The Dominating Set problem is fixed parameter tractable for graphs of bounded genus. In *Proc. 8th SWAT*, volume 2368 of *LNCS*, pages 180–189. Springer, 2002. 11
13. M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th WG*, volume 2880 of *LNCS*, pages 1–12. Springer, 2003. 1
14. M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th WADS*, volume 2748 of *LNCS*, pages 505–520. Springer, 2003. 3
15. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In *Proc. 5th CIAC*, volume 2653 of *LNCS*, pages 108–119. Springer, 2003. To appear in *Theory of Computing Systems*. 4
16. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39(4):321–347, 2004. 4
17. J. Guo and R. Niedermeier. Exact algorithms and applications for Tree-like Weighted Set Cover. Manuscript, June 2004. 8, 9
18. T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning. Domination in graphs applied to electric power networks. *SIAM J. Discrete Math.*, 15(4):519–529, 2002. 5, 6
19. M. Hoffmann and Y. Okamoto. The traveling salesman problem with few inner points. In *Proc. 10th COCOON*, volume 3106 of *LNCS*. Springer, 2004. 2
20. D. Juedes, B. Chor, and M. R. Fellows. Linear kernels in linear time, or How to save k colors in $O(n^2)$ steps. In *Proc. 30th WG*, LNCS. Springer, 2004. To appear. 3
21. R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th MFCS*, LNCS. Springer, 2004. To appear. 3
22. R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for Weighted Vertex Cover. *J. Algorithms*, 47(2):63–77, 2003. 1
23. N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of Vertex Cover. In *Proc. 7th WADS*, volume 2125 of *LNCS*, pages 75–86. Springer, 2001. To appear in *Discrete Applied Mathematics*. 4
24. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43:425–440, 1991. 7
25. K. Pietrzak. On the parameterized complexity of the fixed alphabet Shortest Common Supersequence and Longest Common Subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003. 9
26. N. Robertson and P. D. Seymour. Graph minors. II: Algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986. 1, 3, 7
27. J. M. Robson. Algorithms for maximum independent sets. *J. Algorithms*, 7:425–440, 1986. 4
28. S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In *Proc. 9th COCOON*, volume 2697 of *LNCS*, pages 548–558. Springer, 2003. 2, 3
29. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984. 7
30. J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proc. 3rd WADS*, volume 709 of *LNCS*, pages 610–621. Springer, 1993. 1, 2, 3