Reflections on multivariate algorithmics and problem parameterization

Rolf Niedermeier

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 17-32
www.stacs-conf.org

REFLECTIONS ON MULTIVARIATE ALGORITHMICS AND PROBLEM PARAMETERIZATION

ROLF NIEDERMEIER

Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany
E-mail address: rolf.niedermeier@uni-jena.de

Abstract. Research on parameterized algorithmics for NP-hard problems has steadily grown over the last years. We survey and discuss how parameterized complexity analysis naturally develops into the field of multivariate algorithmics. Correspondingly, we describe how to perform a systematic investigation and exploitation of the "parameter space" of computationally hard problems.

Algorithms and Complexity; Parameterized Algorithmics; Coping with Computational Intractability; Fixed-Parameter Tractability

1. Introduction

NP-hardness is an every-day obstacle for practical computing. Since there is no hope for polynomial-time algorithms for NP-hard problems, it is pragmatic to accept exponential-time behavior of solving algorithms. Clearly, an exponential growth of the running time is bad, but maybe affordable, if the combinatorial explosion is modest and/or can be confined to certain problem parameters. This line of research has been pioneered by Downey and Fellows' monograph "Parameterized Complexity" [24] (see [32, 57] for two more recent monographs). The number of investigations in this direction has steadily grown over the recent years. A core question herein is what actually "a" or "the" parameter of a computational problem is. The simple answer is that there are many reasonable possibilities to "parameterize a problem". In this survey, we review some aspects of this "art" of problem parameterization.[1] Moreover, we discuss corresponding research on multivariate algorithmics, the natural sequel of parameterized algorithmics when expanding to multidimensional parameter spaces.

We start with an example. The NP-complete problem Possible Winner for k-Approval is a standard problem in the context of voting systems. In the *k-approval* protocol, for a given set of candidates, each voter can assign a score of 1 to k of these candidates and the rest of the candidates receive score 0. In other words, each voter may linearly order the candidates; the "first" k candidates in this order score 1 and the remaining ones score 0. A winner of an election (where the input is a collection of votes) is a candidate who achieves the maximum total score. By simple counting this voting protocol can be

[1] In previous work [56, 57], we discussed the "art" of parameterizing problems in a less systematic way.

evaluated in linear time. In real-world applications, however, a voter may only provide a partial order of the candidates: The input of Possible Winner for k-Approval is a set of partial orders on a set of candidates and a distinguished candidate d, and the question is whether there exists an extension for each partial order into a linear one such that d wins under the k-approval protocol. Possible Winner for k-Approval is NP-complete already in case of only two input votes when k is part of the input [10]. Moreover, for an unbounded number of votes Possible Winner for 2-Approval is NP-complete [7]. Hence, Possible Winner for k-Approval parameterized by the number v of votes as well as parameterized by k remains intractable. In contrast, the problem turns out to be fixed-parameter tractable when parameterized by the combined parameter (v, k) [6], that is, it can be solved in $f(v, k) \cdot \text{poly}$ time for some computable function f only depending on v and k (see Section 2 for more on underlying notions). In summary, this implies that to better understand and cope with the computational complexity of Possible Winner for k-Approval, we should investigate its parameterized (in)tractability with respect to various parameters and combinations thereof. Parameter combinations—this is what multivariate complexity analysis refers to—may be unavoidable to get fast algorithms for relevant special cases. In case of Possible Winner for k-Approval such an important special case is a small number of votes[2] together with a small value of k. Various problem parameters often come up very naturally. For instance, besides v and k, a further parameter here is the number c of candidates. Using integer linear programming, one can show that Possible Winner for k-Approval is fixed-parameter tractable with respect to the parameter c [10].

Idealistically speaking, multivariate algorithmics aims at a holistic approach to determine the "computational nature" of each NP-hard problem. To this end, one wants to find out which problem-specific parameters influence the problem's complexity in which quantitative way. Clearly, also combinations of several single parameters should be investigated. Some parameterizations may yield hardness even in case of constant values, some may yield polynomial-time solvability in case of constant values, and in the best case some may allow for fixed-parameter tractability results.[3] Hence, the identification of "reasonable" problem parameters is an important issue in multivariate algorithmics. In what follows, we describe and survey systematic ways to find interesting problem parameters to be exploited in algorithm design. This is part of the general effort to better understand and cope with computational intractability, culminating in the multivariate approach to computational complexity analysis.

2. A Primer on Parameterized and Multivariate Algorithmics

Consider the following two NP-hard problems from algorithmic graph theory. Given an undirected graph, compute a minimum-cardinality set of vertices that either cover all graph edges (this is Vertex Cover) or dominate all graph vertices (this is Dominating Set). Herein, an edge e is *covered* by a vertex v if v is one of the two endpoints of e, and a vertex v is *dominated* by a vertex u if u and v are connected by an edge. By definition, every vertex dominates itself. The NP-hardness of both problems makes the search for

[2]There are realistic voting scenarios where the number of candidates is large and the number of voters is small. For instance, this is the case when a small committee decides about many applicants.

[3]For input size n and parameter value k, a running time of $O(n^k)$ would mean polynomial-time solvable for constant values of k whereas a running time of say $O(2^k n)$ would mean fixed-parameter tractability with respect to the parameter k, see Section 2 for more on this.

polynomial-time solving algorithms hopeless. How fast can we solve these two minimization problems in an exact way? Trying all possibilities, for an n-vertex graph in case of both problems we end up with an algorithm running in basically 2^n steps (times a polynomial), being infeasible for already small values of n. However, what happens if we only search for a size-at-most-k solution set? Trying all size-k subsets of the n-vertex set as solution candidates gives a straightforward algorithm running in $O(n^{k+2})$ steps. This is superior to the 2^n-steps algorithm for sufficiently small values of k, but again turns infeasible already for moderate k-values. Can we still do better? Yes, we can—but seemingly only for Vertex Cover. Whereas we do not know any notably more efficient way to solve Dominating Set [24, 20], in case of Vertex Cover a simple observation suffices to obtain a 2^k-step (times a polynomial) algorithm: Just pick any edge and branch the search for a size-k solution into the two possibilities of taking one of the two endpoints of this edge. One of them has to be in an optimal solution! Recurse (branching into two subcases) to find size-$(k-1)$ solutions for the remaining graphs where the already chosen vertex is deleted. In this way, one can achieve a search tree of size 2^k, leading to the stated running time. In summary, there is a simple 2^k-algorithm for Vertex Cover whereas there is only an $n^{O(k)}$-algorithm for Dominating Set. Clearly, this makes a huge difference in practical computing, although both algorithms can be put into the coarse category of "polynomial time for constant values of k". This categorization ignores that in the one case k influences the degree of the polynomial and in the other it does not—the categorization is too coarse-grained; a richer modelling is needed. This is the key contribution parameterized complexity analysis makes.

To better understand the different behavior of Vertex Cover and Dominating Set concerning their solvability in dependence on the *parameter* k (solution size) historically was one of the starting points of parameterized complexity analysis [24, 32, 57]. Roughly speaking, it deals with a "function battle", namely the typical question whether an $n^{O(k)}$-algorithm can be replaced by a significantly more efficient $f(k)$-algorithm where f is a computable function exclusively depending on k; in more general terms, this is the question for the fixed-parameter tractability (fpt) of a computationally hard problem. Vertex Cover is fpt, Dominating Set, classified as W[1]-hard (more precisely, W[2]-complete) by parameterized complexity theory, is very unlikely to be fpt. Intuitively speaking, a parameterized problem being classified as W[1]-hard with respect to parameter k means that it is as least as hard as computing a k-vertex clique in a graph. There seems to be no hope for doing this in $f(k) \cdot n^{O(1)}$ time for a computable function f.

More formally, parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [24, 32, 57]. One dimension is the input size n (as in classical complexity theory), and the other one is the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. This means that when solving a problem that is fpt, the combinatorial explosion can be confined to the parameter. There are numerous algorithmic techniques for the design of fixed-parameter algorithms, including data reduction and kernelization [11, 41], color-coding [3] and chromatic coding [2], iterative compression [58, 40], depth-bounded search trees, dynamic programming, and several more [44, 60]. Downey and Fellows [24] developed a parameterized theory of computational complexity to show fixed-parameter intractability. The basic complexity class for fixed-parameter intractability is called W[1] and there is good reason to believe that W[1]-hard problems are not fpt [24, 32, 57]. Indeed, there is a whole complexity hierarchy FPT $\subseteq$

W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ XP, where XP denotes the class of parameterized problems that can be solved in polynomial time in case of constant parameter values. See Chen and Meng [22] for a recent survey on parameterized hardness and completeness. Indeed, the typical expectation for a parameterized problem is that it either is in FPT or is W[1]-hard but in XP or already is NP-hard for some constant parameter value.

In retrospective, the one-dimensional NP-hardness theory [34] and its limitations to offer a more fine-grained description of the complexity of exactly solving NP-hard problems led to the two-dimensional framework of parameterized complexity analysis. Developing further into multivariate algorithmics, the number of corresponding research challenges grows, on the one hand, by identifying meaningful different parameterizations of a single problem, and, on the other hand, by studying the combinations of single parameters and their impact on problem complexity. Indeed, multivariation is the continuing revolution of parameterized algorithmics, lifting the two-dimensional framework to a multidimensional one [27].

3. Ways to Parameter Identification

From the very beginning of parameterized complexity analysis the "standard parameterization" of a problem referred to the cost of the solution (such as the size of a vertex set covering all edges of a graph, see Vertex Cover). For graph-modelled problems, "structural" parameters such as treewidth (measuring the treelikeness of graphs) also have played a prominent role for a long time. As we try to make clear in the following, structural problem parameterization is an enormously rich field. It provides a key to better understand the "nature" of computational intractability. The ultimate goal is to quantitatively classify how parameters influence problem complexity. The more we know about these interactions, the more likely it becomes to master computational intractability.

Structural parameterization, in a very broad sense, is the major issue of this section. However, there is also more to say about parameterization by "solution quality" (solution cost herein being one aspect), which is discussed in the first subsection. This is followed by several subsections which can be interpreted as various aspects of structural parameterization. It is important to realize that it may often happen that different parameterization strategies eventually lead to the same parameter. Indeed, also the proposed strategies may overlap in various ways. Still, however, each of the subsequent subsections shall provide a fresh view on parameter identification.

3.1. Parameterizations Related to Solution Quality

The Idea. The classical and most often used problem parameter is the cost of the solution sought after. If the solution cost is large, then it makes sense to study the dual parameter (the cost of the elements *not* in the solution set) or above guarantee parameterization (the guarantee is the minimum cost every solution must have and the parameter measures the distance from this lower bound). Solution quality, however, also may refer to quality of approximation as parameter, or the "radius" of the search area in local search (a standard method to design heuristic algorithms where the parameter k determines the size of a k-local neighborhood searched).

Examples. To find a size-k vertex cover in an n-vertex graph is solvable in $O(1.28^k + kn)$ time [21], that is, Vertex Cover is fixed-parameter tractable. In contrast, finding a size-k dominating set is W[1]-hard. In case of Vertex Cover, the dual parameterization leads to searching for a size-$(n-k')$ vertex cover, where k' is the number of vertices not contained in the vertex cover. This problem is W[1]-hard with respect to the parameter k' [24]. Indeed, this problem is equivalent to finding a size-k' independent set of vertices in a graph. This means that the corresponding problems Vertex Cover and Independent Set are dual to each other.

Above guarantee parameterization was pioneered by Mahajan and Raman [49] studying the Maximum Satisfiability problem, noting that in every boolean formula in conjunctive normal form one can satisfy at least half of all clauses. Hence, an obvious parameterization (leading to fixed-parameter tractability) is whether one can satisfy at least $\lceil m/2 \rceil + k$ clauses of a formula in conjunctive normal form. Herein, m denotes the total number of clauses and the parameter is k, measuring the distance to the guaranteed threshold $\lceil m/2 \rceil$. There is recent progress on new techniques and results in this direction [50, 1]. A long-standing open problem is to determine the parameterized complexity of finding a size-$(\lceil n/4 \rceil + k)$ independent set in an n-vertex planar graph, parameterized by k.

Marx [53] surveyed many facets of the relationship between approximation and parameterized complexity. For instance, he discussed the issue of ratio-$(1+\epsilon)$ approximation (that is, polynomial-time approximation schemes (PTAS's)) parameterized by the quality of approximation measure $1/\epsilon$. The central question here is whether the degree of the polynomial of the running time depends on the parameter $1/\epsilon$ or not.

Khuller et al. [45] presented a fixed-parameter tractability result for k-local search (parameterized by k) for the Minimum Vertex Feedback Edge Set problem. In contrast, Marx [54] provided W[1]-hardness results for k-local search for the Traveling Salesman problem. Very recently, fixed-parameter tractability results for k-local search for planar graph problems have been reported [31].

Discussion. Parameterization by solution quality becomes a colorful research topic when going beyond the simple parameter "solution size." Above guarantee parameterization and k-local search parameterization still seem to be at early development stages. The connections of parameterization to polynomial-time approximation and beyond still lack a deep and thorough investigation [53].

3.2. Parameterization by Distance from Triviality

The Idea. Identify polynomial-time solvable special cases of the NP-hard problem under study. A "distance from triviality"-parameter then shall measure how far the given instance is away from the trivial (that is, polynomial-time solvable) case.

Examples. A classical example for "distance from triviality"-parameterization are width concepts measuring the similarity of a graph compared to a tree. The point is that many graph problems that are NP-hard on general graphs become easily solvable when restricted to trees. The larger the respective width parameter is, the less treelike the considered graph is. For instance, Vertex Cover and Dominating Set both become fixed-parameter tractable with respect to the treewidth parameter; see Bodlaender and Koster [12] for a

survey. There are many more width parameters measuring the treelikeness of graphs, see Hliněný et al. [42] for a survey.

Besides measuring treewidth, alternatively one may also study the feedback vertex set number to measure the distance from a tree. Indeed, the feedback vertex set number of a graph is at least as big as its treewidth. Kratsch and Schweitzer [47] showed that the Graph Isomorphism problem is fixed-parameter tractable when parameterized by the feedback vertex set size; in contrast, this is open with respect to the parameter treewidth. A similar situation occurs when parameterizing the Bandwidth problem by the vertex cover number of the underlying graph [30].

Further examples for the "distance from triviality"-approach appear in the context of vertex-coloring of graphs [18, 51]. Here, for instance, coloring chordal graphs is polynomial-time solvable and the studied parameter measures how many edges to delete from a graph to make it chordal; this turned out to be fixed-parameter tractable [51]. Deineko et al. [23] and Hoffman and Okamoto [43] described geometric "distance from triviality"-parameters by measuring the number of points inside the convex hull of a point set. A general view on "distance from triviality"-parameterization appears in Guo et al. [39].

Discussion. Measuring distance from triviality is a very broad and flexible way to generate useful parameterizations of intractable problems. It helps to better analyze the transition from polynomial- to exponential-time solvability.

3.3. Parameterization Based on Data Analysis

The Idea. With the advent of algorithm engineering, it has become clear that algorithm design and analysis for practically relevant problems should be part of a development cycle. Implementation and experiments with a base algorithm combined with standard data analysis methods provide insights into the structure of the considered real-world data which may be quantified by parameters. Knowing these parameters and their typical values then can inspire new solving strategies based on multivariate complexity analysis.

Examples. A very simple data analysis in graph problems would be to check the maximum vertex degree of the input graph. Many graph problems can be solved faster when the maximum degree is bounded. For instance, Independent Set is fixed-parameter tractable on bounded-degree graphs (a straightforward depth-bounded search tree does) whereas it is W[1]-hard on general graphs.

Song et al. [61] described an approach for the alignment of a biopolymer sequence (such as an RNA or a protein) to a structure by representing both the sequence and the structure as graphs and solving some subgraph problem. Observing the fact that for real-world instances the structure graph has small treewidth, they designed practical fixed-parameter algorithms based on the parameter treewidth. Refer to Cai et al. [19] for a survey on parameterized complexity and biopolymer sequence comparison.

A second example deals with finding dense subgraphs (more precisely, some form of clique relaxations) in social networks [55]. Here, it was essential for speeding up the algorithm and making it practically competitive that there were only relatively few hubs (that is, high-degree vertices) in the real-world graph. The corresponding algorithm engineering exploited this low parameter value.

Discussion. Parameterization by data analysis goes hand in hand with algorithm engineering and a *data-driven* algorithm design process. It combines empirical findings (that is, small parameter values measured in the input data) with rigorous theory building (provable fixed-parameter tractability results). This line of investigation is still underdeveloped in parameterized and multivariate algorithmics but is a litmus test for the practical relevance and impact on applied computing.

3.4. Parameterizations Generated by Deconstructing Hardness Proofs

The Idea. Look at the (many-one) reductions used to show a problem's NP-hardness. Check whether certain quantities (that is, parameters) are assumed to be unbounded in order to make the reduction work. Parameterize by these quantities. It is important to note that this approach naturally extends to deconstructing W[1]-hardness proofs; here the goal is to find additional parameters to achieve fixed-parameter tractability results.

Examples. Recall our introductory example with Possible Winner for k-Approval. From the corresponding NP-hardness proofs it follows that this problem is NP-hard when either the number of votes v is a constant (but k is unbounded) or k is a constant (but v is unbounded) [7, 10], whereas it becomes fixed-parameter tractable when parameterized by both k and v [6].

A second example, where the deconstruction approach is also systematically explained, refers to the NP-hard Interval Constrained Coloring problem [46]. Looking at a known NP-hardness proof [4], one may identify several quantities being unbounded in the NP-hardness reduction; this was used to derive several fixed-parameter tractability results [46]. In contrast, a recent result showed that the quantity "number k of colors" alone is not useful as a parameter in the sense that the problem remains NP-hard when restricted to instances with only three colors [15]. Indeed, Interval Constrained Coloring offers a multitude of challenges for multivariate algorithmics, also see Subsection 4.3.

Discussion. Deconstructing intractability relies on the close study of the available hardness proofs for an intractable problem. This means to strive for a full understanding of the current state of knowledge about a problem's computational complexity. Having identified quantities whose unboundedness is essential for the hardness proofs then can trigger the search for either stronger hardness or fixed-parameter tractability results.

3.5. Parameterization by Dimension

The Idea. The dimensionality of a problem plays an important role in computational geometry and also in fields such as databases and query optimization (where the dimension number can be the number of attributes of a stored object). Hence, the dimension number and also the "range of values of each dimension" are important for assessing the computational complexity of multidimensional problems.

Examples. Cabello et al. [16] studied the problem to decide whether two n-point sets in d-dimensional space are congruent, a fundamental problem in geometric pattern matching. Brass and Knauer [13] conjectured that this problem is fixed-parameter tractable with respect to the parameter d. However, deciding whether a set is congruent to a subset of another set is shown to be W[1]-hard with respect to d [16]. An other example appears in the context of geometric clustering. Cabello et al. [17] showed that the Rectilinear 3-Center problem is fixed-parameter tractable with respect to the dimension of the input point set whereas Rectilinear k-Center for $k \geq 4$ and Euclidean k-Center for $k \geq 2$ are W[1]-hard with respect to the dimension parameter. See Giannopoulos et al. [35, 36] for more on the parameterized complexity of geometric problems.

The Closest String problem is of different "dimension nature". Here, one is given a set of k strings of same length and the task is to find a string which minimizes the maximum Hamming distance to the input strings. The two dimensions of this problem are string length (typically large) and number k of strings (typically small). It was shown that Closest String is fixed-parameter tractable with respect to the "dimension parameter" k [38], whereas fixed-parameter tractability with respect to the string length is straightforward in the case of constant-size input alphabets; also see Subsection 4.1.

Discussion. Incorporating dimension parameters into investigations is natural and the parameter values and ranges usually can easily be derived from the applications. The dimension alone, however, usually seems to be a "hard parameter" in terms of fixed-parameter tractability; so often the combination with further parameters might be unavoidable.

3.6. Parameterization by Averaging Out

The Idea. Assume that one is given a number of objects and a distance measure between them. In median or consensus problems, the goal is to find an object that minimizes the sum of distances to the given objects. Parameterize by the average distance to the goal object or the average distance between the input objects. In graph problems, the average vertex degree could for instance be an interesting parameter.

Examples. In the Consensus Patterns problem, for given strings $s_1, \ldots, s_k$ one wants to find a string s of some specified length such that each s_i, $1 \leq i \leq k$, contains a substring such that the average of the distances of s to these k substrings is minimized. Marx [52] showed that Consensus Patterns is fixed-parameter tractable with respect to this average distance parameter.

In the Consensus Clustering problem, one is given a set of n partitions $C_1, \ldots, C_n$ of a base set S. In other words, every partition of the base set is a clustering of S. The goal is to find a partition C of S that minimizes the sum $\sum_{i=1}^{n} d(C, C_i)$, where the distance function d measures how similar two clusters are by counting the "differently placed" elements of S. In contrast to Consensus Patterns, here the parameter "average distance between two input partitions" has been considered and led to fixed-parameter tractability [9]. Thus, the higher the degree of average similarity between input objects is, the faster one finds the desired median object.

Discussion. The average parameterization for Consensus Patterns directly relates to the solution quality whereas the one for Consensus Clustering relates to the structure of the input. In the latter case, the described example showed that one can deal with "outliers" having high distance to the other objects. Measuring the average distance between the input objects means to determine their degree of average similarity. This structural parameter value may be quickly computed in advance, making it easy to forecast the performance of the corresponding fixed-parameter algorithm.

4. Three Case Studies

In the preceding section, we focussed on various ways to single out various interesting problem parameterizations. In what follows, we put emphasis on the multivariate aspects of complexity analysis related to (combining) different parameterizations of one and the same problem. To this end, we study three NP-hard problems that nicely exhibit various relevant features of multivariate algorithmics.

4.1. Closest String

The NP-hard Closest String problem is to find a length-L string that minimizes the maximum Hamming distance to a given set of k length-L strings. The problem arises in computational biology (motif search in strings) and coding theory (minimum radius problem).

Known Results. What are natural parameterizations here? First, consider the number k of input strings. Using integer linear programming results, fixed-parameter tractability with respect to k can be derived [38]. This result is of theoretical interest only due to a huge combinatorial explosion. Second, concerning the parameter string length L, for strings over alphabet Σ we obviously only need to check all $|\Sigma|^L$ candidates for the closest string and choose a best one, hence fixed-parameter tractability with respect to L follows for constant-size alphabets. More precisely, Closest String is fixed-parameter tractable with respect to the combined parameter $(|\Sigma|, L)$. Finally, recall that the goal is to minimize the maximum distance d; thus, d is a natural parameter as well, being small (say values below 10) in biological applications. Closest String is also shown to be fixed-parameter tractable with respect to d by designing a search tree of size $(d+1)^d$ [38]. A further fixed-parameter algorithm with respect to the combined parameter $(|\Sigma|, d)$ has a combinatorial explosion of the form $(|\Sigma|-1)^d \cdot 2^{4d}$ [48], which has recently been improved to $(|\Sigma|-1)^d \cdot 2^{3.25d}$ [62]. For small alphabet size these results improve on the $(d+1)^d$-search tree algorithm. There are also several parameterized complexity results on the more general Closest Substring and further related problems [29, 37, 52, 48, 62].

Discussion. Closest String carries four obvious parameters, namely the number k of input strings, the string length L, the alphabet size $|\Sigma|$, and the solution distance d. A corresponding multivariate complexity analysis still faces several open questions with respect to making solving algorithms more practical. For instance, it would be interesting to see whether the (impractical) fixed-parameter tractability result for parameter k can be improved when adding further parameters. Moreover, it would be interesting to identify

further structural string parameters that help to gain faster algorithms, perhaps in combination with known parameterizations. This is of particular importance for the more general and harder Closest Substring problem.

Data analysis has indicated small d- and k-values in biological applications. Interesting polynomial-time solvable instances would help to find "distance from triviality"-parameters. Closest String remains NP-hard for binary alphabets [33]; a systematic intractability deconstruction appears desirable. Closest String has the obvious two dimensions k and L, where k is typically much smaller than L. Parameterization by "averaging out" is hopeless for Closest String since one can easily many-one reduce an arbitrary input instance to one with constant average Hamming distance between input strings: just add a sufficiently large number of identical strings. Altogether, the multivariate complexity nature of Closest String is in many aspects unexplored.

4.2. Kemeny Score

The Kemeny Score problem is to find a consensus ranking of a given set of votes (that is, permutations) over a given set of candidates. A *consensus ranking* is a permutation of the candidates that minimizes the sum of "inversions" between this ranking and the given votes. Kemeny Score plays an important role in rank aggregation and multi-agent systems; due to its many nice properties, it is considered to be one of the most important preference-based voting systems.

Known Results. Kemeny Score is NP-hard already for four votes [25, 26], excluding hope for fixed-parameter tractability with respect to the parameter "number of votes". In contrast, the parameter "number of candidates" c trivially leads to fixed-parameter tractability by simply checking all possible $c!$ permutations that may constitute the consensus ranking. Using a more clever dynamic programming approach, the combinatorial explosion can be lowered to 2^c [8]. A different natural parameterization is to study what happens if the votes have high pairwise average similarity. More specifically, this means counting the number of inversions between each pair of votes and then taking the average over all pairs. Indeed, the problem is also fixed-parameter tractable with respect to this similarity value s, the best known algorithm currently incurring a combinatorial explosion of 4.83^s [59]. Further natural parameters are the sum of distances of the consensus ranking to input votes (that is, the Kemeny score) or the range of positions a candidate takes within a vote [8]. Other than for the pairwise distance parameter, where both the maximum and the average version lead to fixed-parameter tractability [8, 59], for the range parameter only the maximum version does whereas the problem becomes NP-hard already for an average range value of 2. [8]. Simjour [59] also studied the interesting parameter "Kemeny score divided by the number of candidates" and also showed fixed-parameter tractability in this case. There are more general problem versions that allow ties within the votes. Some fixed-parameter tractability results also have been achieved here [8, 9].

Discussion. Kemeny Score is an other example for a problem carrying numerous "obvious" parameters. Most known results, however, are with respect to two-dimensional complexity analysis (that is, parameterization by a single parameter), lacking the extension to a multivariate view.

First data analysis studies on ranking data [14] indicate the practical relevance of some of the above parameterizations. Average pairwise distance may be also considered as a

straightforward "distance from triviality"-measure since average distance 0 means that all input votes are equal. The same holds true for the range parameter. Again, known intractability deconstruction for Kemeny Score just refers to looking at the NP-hardness result of Dwork et al. [25, 26], implying hardness already for a constant number of votes. A more fine-grained intractability deconstruction is missing. Kemeny Score can be seen as a two-dimensional problem. One dimension is the number of votes and the other dimension is number of candidates; however, only the latter leads to fixed-parameter tractability. In this context, the novel concept of "partial kernelization" has been introduced [9]. To the best of our knowledge, Kemeny Score has been the first example for a systematic approach to average parameterization [8, 9]. As for Closest String, a multidimensional analysis of the computational complexity of Kemeny Score remains widely open.

4.3. Interval Constrained Coloring

In the NP-hard Interval Constrained Coloring problem [4, 5] (arising in automated mass spectrometry in biochemistry) one is given a set of m integer intervals in the range 1 to r and a set of m associated multisets of colors (specifying for each interval the colors to be used for its elements), and one asks whether there is a "consistent" coloring for all integer points from $\{1, \ldots, r\}$ that complies with the constraints specified by the color multisets.

Known Results. Interval Constrained Coloring remains NP-hard even in case of only three colors [15]. Deconstructing the original NP-hardness proof due to Althaus et al. [4] and taking into account the refined NP-hardness proof of Byrka et al. [15], the following interesting parameters have been identified [46]:

- interval range,
- number of intervals,
- maximum interval length,
- maximum cutwidth with respect to overlapping intervals,
- maximum pairwise interval overlap, and
- maximum number of different colors in the color multisets.

All these quantities are assumed to be unbounded in the NP-hardness reduction due to Althaus et al. [4]; this immediately calls for a parameterized investigation. Several fixed-parameter tractability results have been achieved for single parameters and parameter pairs, leaving numerous open questions [46]. For instance, the parameterized complexity with respect to the parameter "number of intervals" is open, whereas Interval Constrained Coloring is fixed-parameter tractable with respect to the parameter "interval length". Combining the parameters "number of colors" and "number of intervals" though, one achieves fixed-parameter tractability. In summary, many multidimensional parameterizations remain unstudied.

Discussion. The case of Interval Constrained Coloring gives a prime example for deconstruction of intractability and the existence of numerous relevant parameterizations. There are a few known fixed-parameter tractability results, several of them calling for improved algorithms. Checking "all" reasonable parameter combinations and constellations could easily make an interesting PhD thesis.

The biological data often contain only three colors; the corresponding NP-hardness result [15] shows that this alone is not a fruitful parameter—combination with other parameters is needed (such as the interval range [46]). Moreover, observations on biological data indicate a small number of lengthy intervals, motivating a further parameterization possibility. Instances with only two colors or cutwidth two are "trivial" in the sense that (nontrivial) polynomial-time algorithms have been developed to solve these instances [4, 46]. Unfortunately, in both cases a parameter value of three already yields NP-hardness. The two natural dimensions of the problem are given by the interval range and the number of intervals, both important parameters. Average parameterization has not been considered yet. In summary, Interval Constrained Coloring might serve as a "model problem" for studying many aspects of multivariate algorithmics.

5. Conclusion with Six Theses on Multivariate Algorithmics

We described a number of possibilities to derive meaningful "single" parameterizations. Typically, not every such parameter will allow for fixed-parameter tractability results. Assume that a problem is W[1]-hard with respect to a parameter k (or even NP-hard for constant values of k). Then this calls for studying whether the problem becomes tractable when adding a further parameter k', that is, asking the question whether the problem is fixed-parameter tractable with respect to the (combined) parameter (k, k'). Moreover, even if a problem is classified to be fixed-parameter tractable with respect to a parameter k, this still can be practically useless. Hence, introducing a second parameter may open the route to practical fixed-parameter algorithms. Altogether, in its full generality such a "problem processing" forms the heart of multivariate algorithmics.

Fellows et al. [28] proposed to study the "complexity ecology of parameters". For the ease of presentation restricting the discussion to graph problems, one may build "complexity matrices" where both rows and columns represent certain parameters such as treewidth, bandwidth, vertex cover number, domination number, and so on. The corresponding values deliver structural information about the input graph. Then, a matrix entry in row x and column y represents a question of the form "how hard is it to compute the quantity represented by column y when parameterized by the quantity represented by x?". For example, it is easy to see that the domination number can be computed by a fixed-parameter algorithm using the parameter vertex cover number. Obviously, there is no need to restrict such considerations to two-dimensional matrices, thus leading to a full-flavored multivariate algorithmics approach.

After all, a multivariate approach may open Pandora's box by generating a great number of questions regarding the influence and the interrelationship between parameters in terms of computational complexity. With the tools provided by parameterized and multivariate algorithmics, the arising questions yield worthwhile research challenges. Indeed, to better understand important phenomena of computational complexity, there seems to be no way to circumvent such a "massive analytical attack" on problem complexity. Opening Pandora's box, however, is not hopeless because multivariate algorithmics can already rely on numerous tools available from parameterized complexity analysis.

There is little point in finishing this paper with a list of open questions—basically every NP-hard problem still harbors numerous challenges in terms of multivariate algorithmics. Indeed, multivariation is a horn of plenty concerning practically relevant and theoretically

appealing opportunities for research. Instead, we conclude with six claims and conjectures concerning the future of (multivariate) algorithmics.

Thesis 1: Problem parameterization is a pervasive and ubiquitous tool in attacking intractable problems. A theory of computational complexity neglecting parameterized and multivariate analysis is incomplete.

Thesis 2: Multivariate algorithmics helps in gaining a more fine-grained view on polynomial-time solvable problems, also getting in close touch with adaptive algorithms.[4]

Thesis 3: Multivariate algorithmics can naturally incorporate approximation algorithms, relaxing the goal of exact to approximate solvability.

Thesis 4: Multivariate algorithmics is a "systems approach" to explore the nature of computational complexity. In particular, it promotes the development of meta-algorithms that first estimate various parameter values and then choose the appropriate algorithm to apply.

Thesis 5: Multivariate algorithmics helps to significantly increase the impact of Theoretical Computer Science on practical computing by providing more expressive statements about worst-case complexity.

Thesis 6: Multivariate algorithmics is an ideal theoretical match for algorithm engineering, both areas mutually benefiting from and complementing each other.

Acknowledgments. I am grateful to Nadja Betzler, Michael R. Fellows, Jiong Guo, Christian Komusiewicz, Dániel Marx, Hannes Moser, Johannes Uhlmann, and Mathias Weller for constructive and insightful feedback on earlier versions of this paper.

References

[1] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-r-SAT above a tight lower bound. In *Proc. 21st SODA*. ACM/SIAM, 2010.

[2] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *Proc. 36th ICALP*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009.

[3] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[4] E. Althaus, S. Canzar, K. Elbassioni, A. Karrenbauer, and J. Mestre. Approximating the interval constrained coloring problem. In *Proc. 11th SWAT*, volume 5124 of *LNCS*, pages 210–221. Springer, 2008.

[5] E. Althaus, S. Canzar, M. R. Emmett, A. Karrenbauer, A. G. Marshall, A. Meyer-Baese, and H. Zhang. Computing H/D-exchange speeds of single residues from data of peptic fragments. In *Proc. 23rd SAC '08*, pages 1273–1277. ACM, 2008.

[6] N. Betzler. On problem kernels for possible winner determination under the k-approval protocol. 2009.

[7] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. In *Proc. 34th MFCS*, volume 5734 of *LNCS*, pages 124–136. Springer, 2009.

[8] N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. Fixed-parameter algorithms for Kemeny scores. *Theor. Comput. Sci.*, 410(45):4454–4570, 2009.

[9] N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization and partial kernelization for computing medians. In *Proc. 9th LATIN*, LNCS. Springer, 2010.

[10] N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proc. 21st IJCAI*, pages 53–58, 2009.

[11] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009.

[4]For instance, an adaptive sorting algorithm takes advantage of existing order in the input, with its running time being a function of the disorder in the input.

[12] H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comp. J.*, 51(3):255–269, 2008.
[13] P. Brass and C. Knauer. Testing the congruence of d-dimensional point sets. *Int. J. Comput. Geometry Appl.*, 12(1–2):115–124, 2002.
[14] R. Bredereck. Fixed-parameter algorithms for computing Kemeny scores—theory and practice. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2009.
[15] J. Byrka, A. Karrenbauer, and L. Sanità. The interval constrained 3-coloring problem. In *Proc. 9th LATIN*, LNCS. Springer, 2010.
[16] S. Cabello, P. Giannopoulos, and C. Knauer. On the parameterized complexity of d-dimensional point set pattern matching. *Inf. Process. Lett.*, 105(2):73–77, 2008.
[17] S. Cabello, P. Giannopoulos, C. Knauer, D. Marx, and G. Rote. Geometric clustering: fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Transactions on Algorithms*, 2009. To appear. Preliminary version at *SODA 2008*.
[18] L. Cai. Parameterized complexity of vertex colouring. *Discrete Appl. Math.*, 127(1):415–429, 2003.
[19] L. Cai, X. Huang, C. Liu, F. A. Rosamond, and Y. Song. Parameterized complexity and biopolymer sequence comparison. *Comp. J.*, 51(3):270–291, 2008.
[20] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inform. and Comput.*, 201(2):216–231, 2005.
[21] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for Vertex Cover. In *Proc. 31st MFCS*, volume 4162 of *LNCS*, pages 238–249. Springer, 2006.
[22] J. Chen and J. Meng. On parameterized intractability: Hardness and completeness. *Comp. J.*, 51(1):39–59, 2008.
[23] V. G. Deineko, M. Hoffmann, Y. Okamoto, and G. J. Woeginger. The traveling salesman problem with few inner points. *Oper. Res. Lett.*, 34(1):106–110, 2006.
[24] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
[25] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proc. 10th WWW*, pages 613–622, 2001.
[26] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation revisited, 2001. Manuscript.
[27] M. Fellows. Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In *Proc. IWOCA*, volume 5874 of *LNCS*, pages 2–10. Springer, 2009.
[28] M. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. Rosamond, and S. Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory Comput. Syst.*, 45:822–848, 2009.
[29] M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of motif search problems. *Combinatorica*, 26(2):141–167, 2006.
[30] M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *Proc. 19th ISAAC*, volume 5369 of *LNCS*, pages 294–305. Springer, 2008.
[31] M. R. Fellows, F. A. Rosamond, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? In *Proc. 21st IJCAI*, pages 486–491, 2009.
[32] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
[33] M. Frances and A. Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997.
[34] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[35] P. Giannopoulos, C. Knauer, and G. Rote. The parameterized complexity of some geometric problems in unbounded dimension. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 198–209. Springer, 2009.
[36] P. Giannopoulos, C. Knauer, and S. Whitesides. Parameterized complexity of geometric problems. *Comp. J.*, 51(3):372–384, 2008.
[37] J. Gramm, J. Guo, and R. Niedermeier. Parameterized intractability of distinguishing substring selection. *Theory Comput. Syst.*, 39(4):545–560, 2006.
[38] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for Closest String and related problems. *Algorithmica*, 37(1):25–42, 2003.
[39] J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 162–173. Springer, 2004.

[40] J. Guo, H. Moser, and R. Niedermeier. Iterative compression for exactly solving NP-hard minimization problems. In *Algorithmics of Large and Complex Networks*, volume 5515 of *LNCS*, pages 65–80. Springer, 2009.

[41] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[42] P. Hliněný, S. Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *Comp. J.*, 51(3):326–362, 2008.

[43] M. Hoffmann and Y. Okamoto. The minimum weight triangulation problem with few inner points. *Comput. Geom.*, 34(3):149–158, 2006.

[44] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *Comp. J.*, 51(1):7–25, 2008.

[45] S. Khuller, R. Bhatia, and R. Pless. On local search and placement of meters in networks. *SIAM J. Comput.*, 32(2):470–487, 2003.

[46] C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Deconstructing intractability—a case study for interval constrained coloring. In *Proc. 20th CPM*, volume 5577 of *LNCS*, pages 207–220. Springer, 2009.

[47] S. Kratsch and P. Schweitzer. Graph isomorphism parameterized by feedback vertex set number is fixed-parameter tractable. 2009.

[48] B. Ma and X. Sun. More efficient algorithms for closest string and substring problems. In *Proc. 12th RECOMB*, volume 4955 of *LNCS*, pages 396–409. Springer, 2008.

[49] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

[50] M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. *J. Comput. System Sci.*, 75(2):137–153, 2009.

[51] D. Marx. Parameterized coloring problems on chordal graphs. *Theor. Comput. Sci.*, 351(3):407–424, 2006.

[52] D. Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008.

[53] D. Marx. Parameterized complexity and approximation algorithms. *Comp. J.*, 51(1):60–78, 2008.

[54] D. Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Oper. Res. Lett.*, 36(1):31–36, 2008.

[55] H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. In *Proc. 8th SEA*, volume 5526 of *LNCS*, pages 233–244. Springer, 2009.

[56] R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th MFCS*, volume 3153 of *LNCS*, pages 84–103. Springer, 2004.

[57] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[58] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[59] N. Simjour. Improved parameterized algorithms for the Kemeny aggregation problem. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 312–323. Springer, 2009.

[60] C. Sloper and J. A. Telle. An overview of techniques for designing parameterized algorithms. *Comp. J.*, 51(1):122–136, 2008.

[61] Y. Song, C. Liu, X. Huang, R. L. Malmberg, Y. Xu, and L. Cai. Efficient parameterized algorithms for biopolymer structure-sequence alignment. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(4):423–432, 2006.

[62] L. Wang and B. Zhu. Efficient algorithms for the closest string and distinguishing string selection problems. In *Proc. 3rd FAW*, volume 5598 of *LNCS*, pages 261–270. Springer, 2009.