Pareto Complexity of Two-Parameter FPT Problems: A Case Study for Partial Vertex Cover

Peter Damaschke
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract

We describe a framework for expressing the complexity of algorithms for FPT problems with two separate parameters k, m and with exponential time bounds $O^*(x^k y^m)$ where $x, y > 1$ are constant bases. An optimal combination of bases x, y can be chosen depending on the ratio m/k. We demonstrate the framework on a graph problem: finding a vertex cover of size k that leaves at most m edges uncovered. We state the best branching rules we could find so far, for all ranges of m/k. Special attention is paid to the extremal cases where either m/k or k/m are close to 0.

1 Introduction

Parameterized computational problems may have multiple parameters. The discussion in this paper will focus on an examplary problem that we call Vertex Cover with Missed Edges (VCME): Given a graph G and integers k, m, find a set S of at most k vertices and at most m edges such that every edge either belongs to S or has a vertex in S.

We became interested in VCME (and its generalization to hitting sets in hypergraphs) by certain error-resilient inference problems: Every vertex is an unnegated boolean variable, and every edge is a clause, here with two variables. (In the hitting set problem with missed hyperedges, clauses can be larger.) We want to satisfy all clauses by a small number k of true variables. Clauses represent observations and the variables are possible alternative "explanations" of them. A few clauses, say at most m, may erroneously get into the formula (noisy data) and need not be satisfied, but it is unknown which clauses are the spurious ones. Thus we first ask whether there exists a solution with at most k true variables that violates at most m clauses. In the enumeration version of the problem we would like to get all solutions with parameter values bounded by some k and m.

VCME differs from weighted covering problems with one parameter: We may assign weights to vertices and edges and seek a minimum-weight covering. Once we have an algorithm for the weighted version, we can get solutions with several ratios m/k by adjusting the weights. However, since the two parameters can have a very different meaning (see above), we want to have them under control separately and explicitly. The "weighting approach" can easily miss solutions with the prescribed parameter values. By considering the parameters separately we may also develop algorithms optimized in efficiency for specific ranges of k, m. VCME also differs from the Partial Vertex Cover problem where parameters k, t are given and the goal is to cover at least t edges by k vertices [6, 8]. While VCME and Partial Vertex Cover are equivalent as optimization problems (since $m+t$ is the edge number of the input graph), we are interested in cases where the number of uncovered edges is small. In general, complexity results for one parameterization do not say much about other parameterizations.

Algorithms for single-parameter problems that are NP-hard but fixed-parameter tractable (FPT) usually have time bounds $O^*(x^k)$ where $x > 1$ is a constant base, and the O^*-notation suppresses polynomial factors. Here we do not consider cases where the parameterized part of the time bound is a superexponential function. In order not to complicate the discussion we also leave out polynomial factors; in our examples one can always easily see that they are moderate. Any two $O^*(x^k)$ algorithms can be directly compared, and the one with smaller base is superior, at least from sufficiently large k on. In the case of two (or more) parameters the situation becomes more complicated, even if polynomial factors are neglected. Time bounds will be of the form $O^*(x^k y^m)$ with constants $x, y > 1$. Moreover, we may combine several available algorithms with different pairs of constant bases (x, y), and either one can be the best for different ranges of parameters k, m. The O^* complexity of such a compound algorithm is then appropriately described as a set of optimal points (x, y) in the quadrant $x, y > 1$ of the (x, y)-plane. (Here, *optimal* is meant with respect to the given algorithm, but there might exist faster algorithms for the problem in question.)

In Section 2 we propose a framework in which we view the complexity of two-parameter problems with time bounds $O^*(x^k y^m)$. In Section 3 we illustrate the notion and provide some basic two-parameter algorithms for VCME, focussing on the standard technique of bounded search trees. They cover the whole range of ratios m/k. This is only a starting point for refined and improved algorithms, but in the case of large ratio m/k we argue that we have probably the best possible branching algorithm for VCME. Section 4 points out some questions for further research. The main suggestion is to study other natural two-parameter problems in this framework.

2 Pareto Complexity of Two-Parameter Problems

For a general introduction to FPT algorithms and parameterized complexity we refer to [5, 9]. We list a number of definitions and facts for two-parameter problems.

Geometric definitions:

A point (x_0, y_0) in the (x, y)-plane is said to be *dominated* by point (x_1, y_1) if $x_0 \leq x_1$ and $y_0 \leq y_1$. Consider any algorithm for a problem with input parameters k, m, and let B be the set of points (x, y) such that $O^*(x^k y^m)$ is a valid upper bound for its time complexity. Trivially, if $(x, y) \in B$ then every point that dominates (x, y) is also in B.

For any specific problem instance with parameters k and m we want to minimize the algorithm's time bound, that is, choose some $(x, y) \in B$ that minimizes $x^k y^m$. (We can disregard the polynomial factor of the time bound, as it is fixed for a given instance.) Since $x^k y^m = (x y^{m/k})^k$, all parameter pairs (k, m) with the same ratio m/k have the same optimal points $(x, y) \in B$. (We say "points" because the optimal one is not necessarily unique.) We can characterize the optimal points geometrically. Define $X = \log x$ and $Y = \log y$, with any fixed logarithm base. Points in the $x, y > 1$ quadrant of the (x, y)-plane and in the $X, Y > 0$ quadrant of the (X, Y)-plane correspond to each other in the obvious sense. For convenience we do not distinguish them notationally if the reference is clear from context. In particular, B is also considered a set of points in the positive quadrant of the (X, Y)-plane. Let the X- and Y-axis be directed to the right and upwards, respectively. Since $\log(x^k y^m) = kX + mY$, an optimal point for parameter ratio m/k is any point in the (X, Y)-plane where a sweep line with slope $-k/m$ moving upwards meets B first. This gives rise to the following notions in the (X, Y)-plane which we also call the *logarithmic plane*.

Any non-vertical line splits the (X, Y)-plane obviously in a lower and an upper halfplane. A *lower tangent* to B is a line T with negative slope such that T goes through some point of B, and no point of B is in the lower halfplane of T. The *lower convex hull* of B is the intersection of all upper halfplanes of lower tangents to B. The *Pareto curve* of B is the boundary of the lower convex hull of B. It is easy to see that Pareto curves are graphs of monotone decreasing convex functions (with X and Y as the independent and dependent variable, or vice versa). The *maximum of two or more Pareto curves* is defined as the boundary of the intersection of the corresponding lower convex hulls. Equivalently, if we interpret Pareto curves as functions, we take their argument-wise maximum. The *minimum of two or more Pareto curves* is defined similarly, by first taking the union of the lower convex hulls, but after that we build the lower convex hull again, as the union of convex regions is in general not convex. Finally, if B describes

a set of upper bounds of a two-parameter FPT algorithm, we simply speak of the *Pareto curve of the algorithm*, with respect to these known time bounds. Any collection of algorithms for the same two-parameter problem, tuned to several ranges of the ratio m/k, can be merged into a new algorithm whose Pareto curve is the minimum of Pareto curves of the single algorithms. Thus we can formulate that, given a problem, we aim at an algorithm whose Pareto curve is minimal.

Recurrences:

For a problem with parameters k and m, we call the subproblems with $k = 0$ and $m = 0$, resepctively, the *marginal problems*. For example, in the case of VCME, $m = 0$ is the Vertex Cover problem, and $k = 0$ is the trivial problem of checking whether at most m edges are left over. The two-parameter problem cannot be easier than its marginal problems, that is, bases x, y in a bound $O^*(x^k y^m)$ are not smaller than the corresponding bases from the algorithms used for the marginal problems.

We can analyze search tree algorithms for two-parameter problems in a similar way as in the one-parameter case, though with some additional difficulties. The *branching vector* of a branching rule is now a vector of ordered pairs of numbers indicating the reduction of parameters k, m in the different branches. Let $T(k, m)$ be the maximum size, i.e., number of leaves, of a search tree for a problem instance with parameter values k, m. A branching vector with entries $(k_1, m_1), \ldots, (k_s, m_s)$ yields the recurrence $T(k, m) \leq \sum_{i=1}^{s} T(k - k_i, m - m_i)$, where we set $T(0, 0) := 1$. If k or m is already smaller than some k_i or m_i, respectively, the branching rule is no longer applicable. But then we can, in the remaining instance, immediately reduce this constantly bounded parameter to 0 in all possible ways. Even naive exhaustive search costs only polynomial time, since the k_i and m_i are constant. (For example, in VCME this simply means to search for a vertex cover of constantly bounded size.) Afterwards we are left with the marginal problem for $k = 0$ or $m = 0$.

We get solutions of the form $T(k, m) \leq x^k y^m$ with constant bases $x, y > 1$ from the characteristic equation $x^0 y^0 = \sum_{i=1}^{s} x^{-k_i} y^{-m_i}$, which may be multiplied by $x^{k_i} y^{m_j}$ with the largest k_i and m_j, so that all exponents become nonnegative. Any pair of real numbers $x, y > 1$ that satisfies the characteristic equation yields an upper bound $x^k y^m$ on the search tree size. Formally this is proved by bottom-up induction in the search tree, however the argument is fairly simple: $x^k y^m$ satisfies the recurrence as long as the branching rule applies, and when k or m is already down to a constant then the search tree for the marginal problem is no larger than our y^m or x^k, respectively, where m or k denotes the residual value at this stage. In conclusion, these solution pairs determine the Pareto curve of the branching rule. Accordingly, the Pareto curve of a search tree algorithm working with several branching rules is the maximum of the Pareto curves

of the branching rules involved.

An interesting side question is whether the solutions of recurrences of branching rules always form a convex curve in the logarithmic plane. However, if not, then the Pareto curve is just the lower convex hull of the solution curve, that is, nonconvex parts are replaced with lower tangents.

Subgraphs and branching rules:

By *subgraph* we mean a graph formed by a subset of vertices and edges, not necessarily an induced subgraph. Given any graph $G = (V, E)$, the subgraph *spanned* by a set $F \subseteq E$ of edges is the graph consisting of F and the set $V(F)$ of all vertices incident with F, but without the additional edges that might exist inside $V(F)$.

An FPT algorithm for VCME (or a similar problem) in the bounded search tree paradigm consists of branching rules, each working on a set F of edges. A branching rules decides in every branch which vertices and edges of the subgraph spanned by F shall be added to the solution. We also say that these vertices and edges are *selected*. One has to prove that every graph contains some of the specified subgraphs, i.e., some of the branching rules is always applicable, or that the graph is simple in the sense that the problem can be solved in polynomial time. We call a branching rule on F *exhaustive* if, for every valid solution S to VCME restricted to the subgraph spanned by F, the rule has a branch that selects only vertices and edges from S. Search tree algorithms with exhaustive branching rules is somehow the simplest type of FPT algorithms. Clearly, exhaustiveness guarantees that a rule cannot miss possibles solutions to VCME on the entire graph G. On the other hand, exhaustiveness is not necessary for correctness. The simplest example is the following degree-1 rule on a single edge, $F = \{uv\}$: If u is a degree-1 vertex and v its only neighbor, either select vertex u or select edge uv. This rule misses solutions containing only u, but u can always be replaced with v without increasing k or m. The additional information that u has no further neighbors outside $V(F)$ allows us to prove that u is never needed in an optimal solution.

3 Vertex Cover with Missed Edges

In this section we give a first Pareto curve for VCME. We focus on the standard paradigm of bounded search trees with local branching rules on carefully chosen subgraphs. This does not exclude the later consideration of other techniques, and later improvements of our search tree algorithms are possible as well.

First of all, we cannot expect bounds where $y = 1$, that is, FPT algorithms with parameter k only. This is because VCME and Partial Vertex Cover are equivalent as optimization problems, and Partial Vertex Cover with parameter k is $W[1]$-complete [6]. In view of this

fact it is pleasing that we can achieve $O^*(x^k y^m)$ time, with some constant x, for any $y > 1$. In fact, a trivial algorithm is to branch on single edges: Either select the edge or one of its vertices. The characteristic equation $xy = x + 2y$ yields $x = 2y/(y-1) \approx 2/\delta$ for $y = 1 + \delta$ with an arbitrarily small $\delta > 0$. The more interesting question is : What is the *best* x for any given (small) δ? In the following we propose a search tree algorithm with exhaustive branching rules, which is probably already the best algorithm of this type; see details below.

Proposition 1 *Consider any search tree algorithm with exhaustive branching rules for VCME, such that for every fixed $y = 1+\delta$ ($\delta > 0$) the algorithm runs in $O^*(x^k y^m)$ time, with some constant x. Then every rule is, without loss of generality, of the following form: For a set F of f edges, either select all edges of F, or select any one of the g vertices in $V(F)$. (Any other rules make x worse.) Moreover, for $y = 1 + \delta$, $\delta \to 0$, base x behaves as $x \approx g/f\delta$. Clearly, if several, but finitely many branching rules are involved, then we have $x \approx g/f\delta$ for the largest g/f.*

Proof. Consider any branching rule that works on a set F of f edges. Let $g = |V(F)|$. The characteristic equation of the rule can be written as $x^g y^f = x^g p(y) + q(x, y)$, where p is a polynomial of degree strictly smaller than f, and q is a polynomial where x appears only with exponents strictly smaller than g. Since the rule must not enforce that some of the vertices in $V(F)$ be selected, it must possess at least one branch where only edges are selected. On the other hand, the rule must not enforce that only edges be selected, hence it must possess at least one branch where also vertices are selected. It follows that neither p nor q is identical to zero.

Due to the assumptions, the characteristic equation must have solutions with $y = 1 + \delta$ for all (especially, for arbitrarily small) $\delta > 0$. For ease of presentation we use in the following some asymptotic arguments and neglect lower-order terms: For small $\delta > 0$ we may replace terms y^a with $1+a\delta$, neglecting higher-order powers of δ. Then our characteristic equation becomes $x^g(1 + f\delta) \approx x^g p(1 + \delta) + q(x, 1 + \delta)$. Note that $p(1)$ is the sum of coefficients of p. If $p(1) > 1$ then obviously the right-hand side is too large, and no solution (x, y) can exist for small $\delta > 0$. Since the coefficients of p are positive integers, the only remaining possibility is $p(1) = 1$. Hence $p(y) = y^{f-h} = (1+\delta)^{f-h}$ for some integer $h > 0$, corresponding to a unique branch where a set $H \subseteq F$ of h edges is selected. (All other branches have to select also vertices.) Thus, our characteristic equation asymptotically simplifies to $x^g h\delta \approx q(x, 1 + \delta)$. Since $q(x, 1) > 0$ for any $x > 0$, and q as a polynomial is a continuous function, we can even write $x^g h\delta \approx q(x, 1)$. Since $\lim_{\delta \to 0} x = \infty$, only the term with highest degree in $q(x, 1)$ determines the asymptotic behaviour of x as a function of δ. Denote this dominating term by ax^{g-c}, with some integers $a > 0$ and $c > 0$. Hence $x^g h\delta \approx ax^{g-c}$. This finally allows us to express x explicitly as $x \approx \sqrt[c]{a/h\delta}$.

For exhaustive rules we claim that necessarily $c = 1$. To see this, recall that some branch selects H, and all other branches have to select also vertices. Consider any $v \in V(H)$. Since some solution exists that includes v as the only vertex from $V(F)$, some branch must select v and no other vertices (but perhaps some edges along with v), showing that $c = 1$. In particular we get $x \approx a/h\delta$.

On the other hand, the branching rule where one branch selects H and, for every $v \in V(H)$, some branch selects only v, is already an exhaustive branching rule on H. Further branches would only increase a and thus x, and selecting edges along with a vertex would also increase the right-hand side of the characteristic equation and thus x. Hence we can restrict the possible branching rules to this form, and redefine $F := H$, $f := h$, $g := a$. Finally note that $x \approx g/f\delta$. □

For an algorithm of the form as in Proposition 1 we would like to utilize edge sets F with g/f as small as possible. One obvious idea is to choose a set F of three edges incident to one vertex. Since VCME is straightforwardly solvable in polynomial time if all vertices have degree at most 2, this yields an $O^*((4/3\delta)^k(1+\delta)^m)$ time algorithm. However, we can get considerably better x:

Theorem 2 *For every fixed positive integer c, VCME is solvable in time $O^*(((1+1/c)/\delta)^k(1+\delta)^m)$, for all sufficiently small $\delta > 0$.*

Proof. Fix any positive integer c. Find a cycle C in the input graph, which is possible in polynomial time. If C has at most $c+1$ vertices, let F denote its edge set, and apply the branching rule on F as specified in Proposition 1. Using the earlier denotations we have $g/f = 1$ in this case. If C has at least $c+2$ vertices, let F be the edge set of a subpath of C with $c+1$ vertices and c edges. Again, apply the branching rule on F as specified in Proposition 1, now with $g/f = 1+1/c$. As soon as all cycles are destroyed, the remaining graph is a forest, and VCME can be solved straightforwardly in polynomial time, by dynamic programming bottom-up in the trees. (We omit the tedious details.) □

Theorem 2 is the best we could accomplish, in the sense that search tree algorithms based on exhaustive branching rules and with $x \leq 1/\delta$ are unlikely to exist: Since the largest g/f determines the constant factor in $x = \Theta(1/\delta)$, we would need a collection of "dense" subgraphs with $g/f < 1$ such that VCME is polynomial-time solvable on graphs that are free of such subgraphs. However, since $g > f$ if F forms a forest, every subgraph F with $g/f \leq 1$ must include a cycle. On the other hand, by subviding the edges of arbitrary graphs we can avoid any fixed-length cycles, but still the graphs remain arbitrarily complicated, leading to the conjecture that VCME remains NP-complete there. This suggests that probably any

VCME algorithm with $x \leq 1/\delta$ for $y = 1 + \delta$ must use non-exhaustive branching rules or other, more global FPT techniques. We leave it as an open question whether such an algorithm exists.

We need to fix c in Theorem 2, as we can take the maximal $x \approx g/f\delta$ only from a finite collection of branching rules. However, we may choose the optimal c depending on δ as follows. Inspecting the algorithm from Theorem 2, observe that we used branching rules with $x = by^b/(y^b - 1)$, $b \leq c$, and $x = (c+1)y^c/(y^c - 1)$. We can approximate x by $(1 + b\delta)/\delta \leq (1+c\delta)/\delta$ and $(1+1/c)(1+c\delta)/\delta$, respectively. The bound on x is minimized if $1/c + c\delta$ is minimized, that is, $c \approx 1/\sqrt{\delta}$, rounded to an integer.

Corollary 3 *VCME is solvable in* $O^*(((1 + 2\sqrt{\delta} + \delta)/\delta)^k(1 + \delta)^m)$ *time, for all sufficiently small* $\delta > 0$. □

This concludes our discussion of the case when y is close to 1. For increasing y and decreasing x we can afford some branching on the choice of edges while we must make the choice of vertices more efficient. The best algorithm that we found until now for the range $y \leq 1.6$ is stated in:

Theorem 4 *VCME is solvable in* $O^*(x^k y^m)$ *time, for all* (x, y) *that fulfill the equation* $x^3y^4 = 2x^2y^4 + (x + y)^3$.

Proof. Let F be a subgraph with vertices r, s, t, u, v and edges rs, st, tu, tv. First we show that every graph contains such a subgraph, or VCME is solvable in polynomial time. In all connected components that are not just paths or cycles, there exists a vertex t with at least three neighbors s, u, v. Assume that, for every vertex t with three neighbors s, u, v, none of s, u, v has further neighbors outside $\{t, s, u, v\}$. Now, if t has degree exactly 3, then $\{t, s, u, v\}$ forms already a connected component. Consider t with degree 4 or larger. If some edge connects two of t's neighbors, then obviously a (non-induced) subgraph isomorphic to F exists. Otherwise, the connected component of t is merely a star, that is, all edges therein are incident with t. Thus we have shown that "F-free" graphs have only trivial connected components where VCME is easy to solve.

On F we branch as follows. Select s or t or st, and in the last case select, for each of the other three edges independently, either the edge or the other vertex distinct from s, t. The characteristic equation is easy to establish. □

The table shows some points on the Pareto curve:

y	1.02	1.05	1.10	1.15	1.20	1.30	1.40	1.50	1.60
x	63.98	26.49	14.00	9.84	7.77	5.70	4.67	4.06	3.66

For $y > 1.6$ it becomes advantageous to slide to "more vertex-efficient" branching. We found that the following rule is superior to Theorem 4 for larger y:

Theorem 5 *VCME is solvable in $O^*(x^k y^m)$ time, for all (x, y) that fulfill the equation $x^3y^3 = x^2y^3 + (x + y)^3$.*

Proof. Let F be a set of three edges incident to some vertex t. Graphs without such subgraphs are trivial. If F exists, then either we select t, or we select, for each of the other three edges independently, either the edge or the other vertex distinct from t. □

The table shows some points on the Pareto curve:

y	1.70	1.80	1.90	2.00	2.50	3.00	3.50	4.00	5.00	6.00
x	3.34	3.10	2.92	2.77	2.33	2.12	1.98	1.90	1.79	1.73

Note that the rule in Theorem 5 converges for growing y to the simple $x \approx 1.47$ rule for Vertex Cover, having with the characteristic equation $x^3 = x^2 + 1$: take a vertex of degree at least 3 or all its neighbors. In particular, x is limited from below by 1.47, for any y.

It arises the question how small we can make x, for instances with small m/k. Since VCME is a proper generalization of Vertex Cover, a smooth transition would be desirable: Given a known $O^*(b^k)$ time algorithm for Vertex Cover, we would like to have an $O^*(x^k y^m)$ algorithm for VCME where x tends to b if m/k goes to 0. We do not have a method that takes an arbitrary Vertex Cover algorithm as a black box and generates such an algorithm for VCME. We state the existence of such a method as an open problem. However, for search tree algorithms that use only exhaustive (and also certain non-exhaustive) branching rules we can achieve the desired behaviour, as we explain below.

First, we can transform every exhaustive branching rule for Vertex Cover, working on an edge set F, in a generic way into a branching rule for VCME: Every branch of the original rule is preserved, i.e., the specified vertices are selected, and for every edge in F we add a branch where only this edge is selected. Correctness is evident. For some non-exhaustive rules we can proceed in the same way, but correctness must be proved separately. As an example consider again the degree-1 rule: If u is a degree-1 vertex and v its only neighbor, we can erase u and put v in the vertex cover. The only role of u in a vertex cover would be to cover the edge uv, hence u can always be replaced with v. The corresponding rule for VCME selects either v or uv. This is still correct, as we discussed earlier.

For the analysis, consider any branching rule in the given Vertex Cover algorithm, working on a set F of f edges. Let $x^d = p(x)$ be its

characteristic equation, where p is a polynomial of degree smaller than d, with nonnegative coefficients and, in particular, with a positive coefficient of x^0. Let b be the branching number of the rule, that is, $b^d = p(b)$. In the following we use some known basic properties of such equations (we refer to, e.g., [3]): $x^d = p(x)$ has a unique positive root b, and it holds $x^d < p(x)$ for $x < b$, and $x^d > p(x)$ for $x > b$. It also follows that $db^{d-1} - p'(b) > 0$, since if $x^d - p(x)$ had derivative zero at $x = b$ then b would be a root with multiplicity 2 or higher, and a slight perturbation of the coefficients would yield another polynomial of the considered type with more than one positive root, which contradicts the mentioned properties. The branching rule for VCME obtained by the scheme above has the characteristic equation $x^d y = p(x)y + fx^d$. Since $fx^d > 0$, any solution (x, y) must satisfy $x > b$, thus $x^d > p(x)$. It follows $y = fx^d/(x^d - p(x))$. For $x = b + \delta$ with small δ we obviously get $y \approx fb^d/(db^{d-1} - p'(b))\delta$. That is, we can bring x as close to b as desired, still keeping a finite base y for each δ, although with $y \to \infty$ for $x \to b$. This reasoning holds in particular for the branching rule with the largest b that determined the $O^*(b^k)$ complexity of the Vertex Cover algorithm we started from. Since y goes to infinity, for all x sufficiently close to b the Pareto curves of all other rules are below the Pareto curve of this rule with maximum b. Hence this rule also dominates the complexity of the VCME algorithm, for δ sufficiently small.

Now we apply this machinery to the Vertex Cover algorithm of [1]. It has been subsequently improved several times, but its base 1.325 is already fairly close to the currently best branching number 1.2738 for Vertex Cover [4]. Moreover, we can easily apply our transformation and analysis.

Theorem 6 *VCME is solvable in $O^*((1.325 + \delta)^k (3.54/\delta)^m)$ time, for all sufficiently small $\delta > 0$.*

Proof. We only sketch the proof and refer to [1] for details. The Vertex Cover algorithm consists of twelve branching rules. One can easily check them one by one and verify that they are of the form described above and thus extendible to branching rules for VCME. As stated in [1], Rule 3 and 7 have the maximum branching number $b < 1.325$ in this algorithm. Rule 3 works on a subgraph with $f \leq 6$ edges (while rule 7 involves only 5 edges), which yields the characteristic equation $x^3 y = xy + y + 6x^3$ with the claimed solution. □

For more sophisticated algorithms using other techniques one has to check whether our assumptions on branching rules are still valid, or weaken the assumptions and enrich the rule transformation method. However, one may doubt that the use of more complicated Vertex Cover algorithms with somewhat improved b is of much practical value also for VCME: If x approaches b, then y goes up quickly, so that we need to relax the base x

anyway. That is, the smallest possible b and x does not seem to be the main concern in the VCME context. For very small ratios m/k we would in practice rather select m edges exhaustively on a kernel, and then simply apply any VERTEX COVER algorithm to the remaining instances. This type of algorithm does not enjoy constant bases y but is expected to be simpler and faster for very small m/k.

Related to this discussion, we finally give a simple quadratic kernel for VCME, generalizing straightforwardly the $O(k^2)$ kernelization for VERTEX COVER. Due to kernel results for VERTEX COVER we conjecture that VCME has actually an $O(k+m)$ size kernel.

Theorem 7 *A kernel for VCME with k^2+km+m edges can be computed in polynomial time.*

Proof. Any vertex of degree larger than $k+m$ must be selected, since otherwise we have to select more than k other vertices or more than m edges. After removal of the enforced vertices and all incident edges, there remains a graph of maximum degree $k+m$. Now k vertices can cover at most $k(k+m)$ edges, hence at most $k(k+m)+m$ edges remained, or there is no solution. □

4 Conclusions

We proposed the framework of Pareto complexity for FPT problems with two parameters k, m, where we want time complexities $O^*(x^k y^m)$ with constant bases x, y. As an illustration and starting point we gave some basic branching algorithms for VCME, the problem of finding vertex covers with k vertices and m missed edges. The algorithms are tailored to several ranges of m/k. We have not studied minor technical issues like convexity of their Pareto curves in the logarithmic plane (see Section 2). There are some natural questions for further research, besides the ones already brought up in the technical sections:

- Get improvements: Find VCME algorithms with lower Pareto curves. Better branching rules for some ranges of m/k are likely to exist.
- Despite our discussion of small m/k: Give algorithms for VCME with $x < 1.325$, possibly with x matching the best known base for VERTEX COVER algorithms.
- Study the Pareto complexity of the corresponding enumeration and counting problems: How many solutions with k vertices and m edges exist, etc.?
- Extend the approach to hitting sets in hypergraphs of fixed rank $3, 4, 5 \ldots$

- Study other problems. For instance, some variants of the *Cluster Editing* and *Cluster Vertex Deletion* problem naturally have two parameters: (a) number of edge deletions and edge insertions, (b) number of vertex deletions and edge edits, (c) number of vertex deletions and resulting cliques. (The latter problem was studied in [7], however not in the "Pareto framework".)

References

[1] R. Balasubramanian, M.R. Fellows, V. Raman. An improved fixed-parameter algorithm for vertex cover, *Info. Proc. Letters* 65 (1998), 163-168

[2] S. Böcker, S. Briesemeister, Q.B.A. Bui, A. Truß. Going weighted: Parameterized algorithms for cluster editing, *2nd COCOA 2008*, *LNCS* 5165, 1-12

[3] J. Chen, I.A. Kanj, G. Xia. A note on search trees, `TR05-006.pdf` on `facweb.cs.depaul.edu/research/TechReports/`

[4] J. Chen, I.A. Kanj, G. Xia. Simplicity is beauty: Improved upper bounds for vertex cover, `TR05-008.pdf` on `cdm.depaul.edu/research/Documents/TechnicalReports/2005/`

[5] R.G. Downey, M.R. Fellows. *Parameterized Complexity*, Springer, 1999

[6] J. Guo, R. Niedermeier, S. Wernicke. Parameterized complexity of generalized vertex cover problems, *9th WADS 2005*, *LNCS* 3608, 36-48

[7] F. Hüffner, C. Komusiewicz, H. Moser, R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion, *8th LATIN 2008*, *LNCS* 4957, 711-722, to appear in *Theory of Computing Systems*

[8] J. Kneis, A. Langer, P. Rossmanith. Improved upper bounds for partial vertex cover, *34th WG 2008*, *LNCS* 5344, 240-251

[9] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, Oxford Lecture Series in Mathematics and Its Applications, Oxford University Press 2006