On Parameterized Enumeration

Henning Fernau

WSI-2001-21

Henning Fernau

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

E-Mail: `fernau@informatik.uni-tuebingen.de`
Telefon: (07071) 29-77565
Telefax: (07071) 29-5061

ISSN 0946-3852

On Parameterized Enumeration

Henning Fernau
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: fernau@informatik.uni-tuebingen.de

December 17, 2001

Abstract

We study several versions of parameterized enumeration. The idea is always to have an algorithm which outputs all solutions (in a certain sense) to a given problem instance. Such an algorithm will be analysed from the viewpoint of parameterized complexity. We show how to apply enumeration techniques in a number of examples. In particular, we give a fixed parameter algorithm for the reconfiguration of faulty chips when providing so-called shared and linked spares.

1 Introduction

In classical complexity theory, there are three main ways to build complexity classes or classify computational problems, respectively: decision problems, functional problems, and counting problems.

As a running example, let us consider the *vertex cover problem* (VC) for undirected graphs. This yields the following decision problem (already formulated with a glimpse on parameterized complexity):

Instance: A graph $G = (V, E)$
Parameter: positive integer k
Question: Is there a vertex cover $C \subseteq V$ with $|C| \leq k$, i.e., each edge from E is incident to at least one vertex from C?

A vertex cover with k vertices will also be called k-vertex cover. Alternatively, one could ask for an algorithm that in fact yields a vertex cover $C \subseteq V$ with $|C| \leq k$ instead of merely stating its existence. This would be the functional version of the problem. Finally, one could be interested in knowing how many different vertex covers $C \subseteq V$ with $|C| \leq k$ exist. This would be the natural counting version of VC.

Obviously, there is a fourth natural problem type, namely the functional version of the counting problem (which we will call *enumeration problem*

in the following):[1] Output *all* k-vertex covers of a given graph. A variant would be to output *all minimum* k-vertex covers of a given graph. This we call the *optima enumeration problem*.

In classical complexity theory, it does not make much sense to ask for such an algorithm for vertex cover, since only the size of the graph is measured within complexity considerations. This means that, generally speaking, for NP-hard problems, an exponential number of outputs is to be generated in the worst case. Even the counting problem is considerably hard. Notably, Goldberg, Spencer and Berque have published a low-exponential algorithm for counting vertex covers, see [11].

In contrast, the main idea of developing fixed parameter algorithms is to explicitly declare a part of the problem instance as a so-called parameter, expecting that this parameter tends to be small in practice, whereas the overall size of the instance might be huge. This means that one can afford (mildly) exponential behaviour of algorithms in terms of the parameter, as long as the overall running time is polynomial when considering the parameter as a fixed constant. More formally, a decision problem is called *fixed parameter tractable* if its running time is bounded by $f(k) \cdot n^{O(1)}$, where f is some arbitrary function, k is the parameter, and n is the size of the problem instance. For example, in the case of VC, algorithms running in time $O(c^k + kn)$ have been developed, where $c < 1.3$, see [17].

In this paper, we will focus on the following forms of parameterized enumeration:

- Generate *all* solutions,
- Generate *all optimal* solutions, and
- Generate *representative* solutions.

We will discuss all above-mentioned variants by means of examples in the following. This paper is intended to be a start-up of a theory of parameterized enumeration. The results obtained up to now are promising and we think that this issue deserves more in-depth study.

Why do we think that parameterized enumeration is important? There are a number of possible applications of such a theory, mainly dealing with the further processing of data. For example, Gramm and Niedermeier [12] developed a fixed parameter algorithm for the so-called minimum quartet inconsistency problem (MQI) which is important for constructing evolutionary trees in biology. An evolutionary tree is a rooted binary tree whose leaves

[1] We are aware of the fact that Valiant [19] introduced the term "enumeration" in complexity theory in the sense of "counting". In this sense, it was also used in parameterized complexity [5]. Since the term "enumeration" was used in recursion theory in the sense of generating all elements of a certain (possibly infinite) set, we think that our notation is appropriate nonetheless.

are bijectively labelled by taxa from a set S. A quartet is an evolutionary tree with four leaves. A problem instance of MQI consists of an n-element set of taxa S and $\left(\begin{array}{c} n \\ 4 \end{array} \right)$ quartets such that, to each four-element subset S' of S, there is exactly one quartet whose leaves are labelled with taxa from S'. The aim is to construct an evolutionary tree T whose leaves are bijectively labelled by taxa from S such that the number of sub-trees of T with four leaves which are different from the input quartet with the same leaf labels is bounded by a given error bound, the parameter k of the problem. In this application, it is interesting for the human expert to see and check *all* reconstructed evolutionary trees (satisfying the given error bound) in order to choose the tree variants which appear to him to be the most reasonable choice, given his additional background knowledge on the subject. In fact, Gramm and Niedermeier already showed how to enumerate all such minimum solutions in time $O(4^k p(n))$.

Of course, the enumerated solutions could also be the basis of further computations, even as a kind of heuristic estimate. For example, some researchers interested in computing a k-dominating set of a graph heuristically assume that such a dominating set is included within a $2k$-vertex cover and use the known (comparatively fast) vertex cover algorithm (computing some $2k$-cover) in a preprocessing phase.[2] Of course, this heuristic could be improved by starting off from all (minimum) vertex covers.

Below, we will discuss an example from VLSI reconfiguration that shows the practical importance of knowing some representative of all kinds of uncomparable minimal solutions as a basis of further computations. Moreover, it is shown how these enumeration algorithms can be employed to solve practically relevant variants of decision problems in relation with VLSI reconfiguration.

2 Generating all solutions

Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized language. In the vertex cover example, L would consist of pairs $(c(G), k)$, where G is some graph having a k-vertex cover and c is some natural coding function.

Let $L_f \subseteq \Sigma^* \times \Sigma^* \times \mathbb{N}$ be the "corresponding" functional language. A tuple (σ, x, k) is in L_f iff $(x, k) \in L$ and σ is a "solution witness" for (x, k). In the vertex cover example, L_f consists of triples $(c'(V'), c(G), k)$, where G is a graph having the k-vertex cover V', and c' and c are some coding functions.

Let us assume in the following that the parameterized problem we are considering is related to an optimization problem such that the parameter bounds the entity to be optimized. Then, we say that k is optimal for the

[2] U. Stege, personal communication about a Swedish bioinformatics group

given optimization problem instance x if the size of the optimal solution to x matches k.

L_f is *[optimally] fixed parameter enumerable* iff there is an algorithm which, given $(x,k) \in L$ [where k is optimal for x] generates all $\sigma \in \Sigma^*$ with $(\sigma, x, k) \in L_f$ in time $f(k) \cdot |x|^{O(1)}$.

L_f is of *[optimal] fixed parameter size* iff

$$|\{(\sigma, x, k) \mid \sigma \in \Sigma^*[, k \text{ optimal for } x]\}| \leq f(k) \cdot |x|^{O(1)}.$$

From the discussion in the introduction, we get a first example:

Example 2.1 MQI is minimally fixed parameter enumerable.

The contraposition of the next lemma is important in the following.

Lemma 2.2 Let L_f be a functional parameterized language. If L_f is [optimally] fixed parameter enumerable, then L_f is of optimal fixed parameter size.

Theorem 2.3 VC is optimally fixed parameter enumerable in time $O(2^k k^2 + kn)$, where n is the number of vertices of the input graph and k is the parameter.

Proof. This can be shown by using Buss' kernelization (see [6]) and a search-tree technique. More precisely, we use the following two kernelization rules as long as possible:

- If v is a vertex with no neighbours, v can be removed from the graph, since v will not be part of any minimum vertex cover.
- If v is a vertex of degree greater than k, v must be in any vertex cover, since otherwise all neighbours would be in the cover, which is not feasible, because we are looking for vertex covers with at most k vertices. Hence, we can remove v from the graph.

After having applied these kernelization rules exhaustively, we are left with a graph with at most k^2 vertices. Now, we can use the following simple search-tree algorithm `enumcover`$(G = (V, E), k, C)$ (similar to the algorithm which already appeared in [15] before the advent of parameterized complexity):
`enumcover`$(G = (V, E), k, C)$:
IF $k = 0$ and ($V = \emptyset$ or $E = \emptyset$) THEN output C.
IF $k > 0$ and ($V \neq \emptyset$ and $E \neq \emptyset$) THEN DO:

- Choose some edge $\{v_1, v_2\} \in E$.
- Recursively branch according to the following two cases:
  1. v_1 is contained in a cover: Call `enumcover`$(G - v_1, k-1, C \cup \{v_1\})$,

2. v_2 is contained in a cover: Call `enumcover`$(G - v_2, k-1, C \cup \{v_2\})$.

If G and k is the input instance of the VC optima enumeration problem (after having applied the kernelization rules), then `enumcover`$(G, k, \emptyset)$ solves the problem. The claimed time bounds are immediate: $O(kn)$ time is needed for the kernelization, and the computation necessary in each invocation of `enumcover` can be done in time proportional to the number of vertices in the remaining graph, so that $O(2^k k^2)$ time is spent for evaluating the search tree. □

Remark 2.4 Let us mention the following historical aside: Although the reduction rules listed above are generally attributed to a personal communication of Sam Buss, there is a reference of Evans [8] that predates the Buss reference considerably. Admittedly, Evans considers a special variant of vertex cover (namely, the CBVC problem discussed in detail in the next section) which arises in connection with VLSI reconfiguration, but the reduction rules are basically the same. Possibly even more interestingly, Haddad, Dahbura and Sharma presented a rather complete fixed parameter algorithm (and its analysis) for CBVC (including the discussion of kernelization and search-tree techniques) already in 1991 [13].

The following remark is interesting, since lower bounds are usually hard to obtain.

Remark 2.5 Essentially, there is no better minimum vertex cover enumeration algorithm than the one given in Theorem 2.3, since the graph

$$(\{1, \ldots, k\} \times \{1, 2\}, \{\{(i, 1), (i, 2)\} \mid 1 \leq i \leq k\})$$

has 2^k many different minimum vertex covers.

This simple example is interesting, since it shows, in addition, that there is no minimum vertex cover enumeration algorithm for planar vertex cover having running time of the form $c^{\sqrt{k}}n$, as it has been found for the decision problem in [2].

Remark 2.6 On the contrary, vertex cover is *not* of fixed parameter size, since the n-vertex graph with no edges has $\binom{n}{k}$ many different k-vertex covers. Lemma 2.2 shows that VC is hence not fixed parameter enumerable.

The previous considerations show that the two notions of parameterized enumerability defined above are really different. On the other hand, we can prove the following general relationship between both notions:

Lemma 2.7 If a minimization problem is fixed parameter enumerable, then it is optimally fixed parameter enumerable.

If a maximization problem (where the size of the parameter is naturally bounded by a polynomial of the size of the problem instance) is fixed parameter enumerable, then it is optimally fixed parameter enumerable.

Proof. We consider the case of minimization problems. Maximization problems are treated similarly. One simply starts the enumeration algorithm with parameter 1, 2 through k and checks, for each output solution, whether it is minimal; the minimality is checked by going through all solutions generated by invocations of the enumeration algorithm with smaller parameter values. If the enumeration problem can be solved in time $f(k) \cdot |x|^{O(1)}$ on a problem instance (x, k), then the minima enumeration problem is solvable in time

$$f(k) \cdot |x|^{O(1)} \cdot (\sum_{j=0}^{k-1} f(j) \cdot |x|^{O(1)}) \leq k(f(k))^2 \cdot |x|^{O(1)}.$$

□

The next remark shows that not all parameterized problems are optimally fixed parameter enumerable. Moreover, the given example proves again that the dominating set problem[3] appears to be harder than the vertex cover problem from a parameterized point of view, also see [6].

Remark 2.8 Dominating set is even *not* of optimal fixed parameter size, as the k-fold disjoint graph union of K_n shows. By Lemma 2.2, this problem is not optimally fixed parameter enumerable.

Up to now, we only considered minimization problems. Let us briefly consider one maximization problem in the parameterized setting, namely, the problem of finding a maximum *independent set*, i.e., a set of vertices I of a given graph such that no vertex in I is neighbour of another vertex in I, of size (at least) k.

Remark 2.9 We first consider the independent set problem restricted to planar graphs. It is quite easy to see that this problem is optimally fixed parameter enumerable. Namely, construct a 4-colouring of the given planar graph G (which exists due to the famous four-colour theorem for planar graphs); each of the four such-obtained monochromatic vertex sets is independent and the largest one contains at least $n/4$ vertices. Hence, if $k < n/4$, we can always answer "no"; otherwise, we know that $n \leq 4k$ and, hence, there are at most $f(k) = \left(\begin{array}{c} 4k \\ k \end{array} \right)$ many different independent sets of size k. Hence, enumerating all *maximum* independent sets would amount checking for at most $3kf(k)$ many vertex sets whether they are independent or not

[3] A *dominating set* of a graph is a subset of vertices such that every vertex is either a member of the dominating set or a neighbour of a member of the dominating set.

(the additional factor of $3k$ comes from the necessity of checking all possible extensions of a candidate set of size k).

As in Remark 2.6, one can see that planar independent set is *not* of fixed parameter size.

This example is illuminating in a further way:

As Eppstein has shown in [7], there is an algorithm for listing all maximum independent sets smaller than k in a (not necessarily planar) n-vertex graph in time $O(3^{4k-n}4^{n-3k})$. Moreover, he gave an example, namely the disjoint graph union of $4k-n$ triangles and $n-3k$ K_4's for proving that the derived time bound is tight if $n/4 \leq k \leq n/3$. It is tempting to conclude that the language describing all maximum independent sets is not of optimal fixed parameter size, even if we restrict the problem to planar graphs, but this conclusion is false, as explained above.

As can be seen similarly to Remark 2.8, the independent set problem on general graphs is not of optimal fixed parameter size.

Finally, we observe that Theorem 2.3 can be used in order to show fixed parameter tractability of the decision problem mentioned in the introduction:

Remark 2.10 The following decision problem is fixed parameter tractable: Given a graph G and parameters k and ℓ, is there a k-dominating set included in some minimum $\ell \cdot k$-vertex cover of G?

This can be seen by generating all minimum $\ell \cdot k$-vertex covers and then testing, for each k-element subset of such a cover, whether it forms a dominating set.

3 Generating all representative solutions

In the course of this section, we will mainly focus on parameterized minimization problems with two parameters, although the main ideas can be easily generalized to an arbitrary number of parameters. Similar notions can be coined for maximization problems, as well.

Let $L \subseteq \Sigma^* \times \mathbb{N}^2$ be a parameterized language with two parameters k_1, k_2 (stemming from a minimization problem). If $(\sigma, x, k_1, k_2) \in L_f$, where L_f is the functional problem corresponding to L as in the previous section, then (k_1, k_2) is called the *signature* of (σ, x) if $(\sigma, x, k_1', k_2') \in L_f$ and $(k_1', k_2') \leq (k_1, k_2)$ imply $(k_1', k_2') = (k_1, k_2)$, where we consider the partial order $(k_1', k_2') \leq (k_1, k_2)$ iff $k_1' \leq k_1$ and $k_2' \leq k_2$.

Lemma 3.1 If we consider k_1 and k_2 as fixed, then there are at most $\min\{k_1, k_2\} + 1$ pairwise uncomparable (minimal) signatures.

Hence, given a (codified) problem instance $x \in \Sigma^*$ and parameters k_1 and k_2, there are at most $\min\{k_1, k_2\} + 1$ elements in

$$\{(\sigma, x, k_1', k_2') \in L_f \mid (k_1', k_2') \leq (k_1, k_2) \wedge (k_1', k_2') \text{ is the signature of } (\sigma, x)\}$$

having different minimal signatures (k_1', k_2'). In some applications (as explained below), it is interesting to generate one *representative solution* for each minimal signature, given some problem instance. Due to the above lemma, there are at most $\min\{k_1, k_2\} + 1$ such representative solutions.

Detailed example: chip reconfiguration

Kuo and Fuchs [14] provide a fundamental study of the *spare allocation problem*. Put concisely, this "most widely used approach to reconfigurable VLSI" uses spare rows and columns to tolerate failures in rectangular arrays of identical computational elements, which may be as simple as memory cells or as complex as processor units. If a faulty cell is detected, the corresponding entire row or column is replaced by a spare one.

The array below sketches a concrete small example of a 7×9 array, where faults are indicated by question marks.

	1	2	3	4	5	6	7	8	9
1	?			?					?
2									
3	?								
4			?	?			?		
5									
6									
7				?					?

This array can be repaired, e.g., by using three spare rows (replacing rows number 1,4 and 7) and one spare column (replacing column number 1).

Equivalently, this reconfiguration problem can be formulated graph-theoretically as *Constraint Bipartite Vertex Cover (CBVC)* problem as follows: given a bipartite graph $G = (V_1, V_2, E)$ and *two* positive integers k_1 and k_2, are there two subsets $C_1 \subseteq V_1$ and $C_2 \subseteq V_2$ of sizes $|C_1| \leq k_1$ and $|C_2| \leq k_2$ such that each edge in E has at least one endpoint in $C_1 \cup C_2$?[4]

In [10], a fixed parameter algorithm running in time less than $O(1.4^{k_1+k_2}n)$ was developed for this decision problem.[5] In fact, by analyzing the decision procedure developed in that paper one easily derives:

[4] It was this problem where Evans proposed the Buss-like reduction as noted in Remark 2.4.

[5] A simpler algorithm for the CBVC problem with the additional restriction that only those bipartite covers are considered which also form a minimum vertex cover of the graph was established in [4].

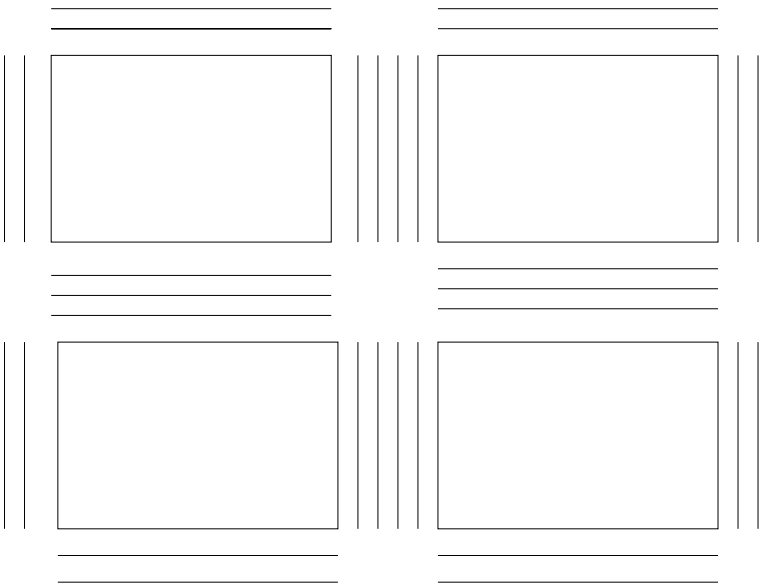

Figure 1: Sharing repair resources

Corollary 3.2 For the CBVC problem, generating one representative solution for each minimal signature can be done in time

$$O(1.3999^{k_1+k_2}k_1k_2 + (k_1+k_2)n),$$

where n is the number of vertices of the input graph and k_1 and k_2 are the two parameters.

Remark 3.3 As in the case of VC, CBVC is not fixed parameter enumerable.

A more realistic scenario

As pointed out in, e.g., [13], there are several points due to which the problem formulated above is not a completely adequate model:

1. In the manufacturing process, the cost of repairing a chip by using vertical movements of the repair laser may be different from that of horizontal movements. This leads to a sort of weighted variant of CBVC.

2. As indicated in Fig. 1, a huge memory chip may be split into smaller blocks, each of them possibly having its own spare rows and columns. For reasons of economy, other designs are preferred in this case, e.g., each spare row depicted inbetween two memory blocks can be individually used to reconfigure either the block above or the block below it.

In other words, in such complex designs, spares may be *shared*. Moreover, there may be spare rows or columns which are *linked*, which means that such a spare can only be used to reconfigure *one* certain row or column in several blocks. Obviously, the idea is here to reduce the costs of chip repair.

We will address the problems mentioned above in the following.

Theorem 3.4 The weighted CBVC problem mentioned in point 1. above can be solved in time $O(1.3999^{k_1+k_2}k_1k_2+(k_1+k_2)n)$, where n is the number of vertices of the input graph and k_1 and k_2 are the two parameters.

Proof. According to Cor. 3.2, one representative solution per minimal signature can be produced in time $O(1.3999^{k_1+k_2}k_1k_2+(k_1+k_2)n)$. Among these, a cost-optimal solution can be found basically in time $O(k_1+k_2)$, since the "repair cost" only depends on the signature, and there are at most $\min\{k_1,k_2\}+1$ minimal signatures, see Lemma 3.1. □

Let us now consider the chip reconfiguration problem with memory blocks and shared spares.

Theorem 3.5 Given a chip board with n elementary cells which is split into k_3 blocks each of which has at most k_1 neighbouring spare rows and k_2 neighbouring spare columns, then a reconfiguration strategy can be found in time

$$O(k_3(1.3999^{k_1+k_2}k_1k_2+(k_1+k_2)n)+k_3(\min\{k_1,k_2\}+1)^{\sqrt{k_3}+1})$$

if it exists.

Proof. (Sketch) At first, we run the representative enumeration procedure from Cor. 3.2 for each block. Then, all possible combinations of signatures for all blocks are examined to see whether the decision problem is solvable. This second step can be implemented more efficiently by using dynamic programming techniques in a sweep-line fashion. From a graph-theoretic point of view, we exploit the fact that a grid graph (representing the local dependencies between the blocks on the chip) with k vertices has treewidth of at most $\sqrt{k}+1$, see [3]. □

In other words, the parameterized enumeration of representative solutions can be used in order to show that another (related) decision problem is fixed parameter tractable, considering k_1, k_2 and k_3 as parameters of the problem.

The third mentioned variation which is also incorporating linked spares seems to be harder, since knowing only one representative solution per signature is of not much help here. Even worse, also the generation of all minimum solutions (which can be done as in the case of vertex cover elaborated

above) would not help, since possibly non-optimal solutions (considered "locally" for each block) would be a better choice. For example, consider the following chip with two blocks each containing three rows:

	1	2	3	4	5	6	7	8	9
1	?			?					
2									
3		?							
4				?					
5	?			?			?		
6	?			?					

For each of the two blocks, we have one spare row and, furthermore, there are two linked spare columns. If we use the linked spare columns in order to repair columns number 1 and 4, the array can be repaired by using the remaining two spare rows for row number 3 and row number 5. Only considering the first block, this solution is not minimal, since its signature $(1, 2)$ is outperformed by taking, e.g., a spare row for row number 1 and one of the two linked spare columns for column number 2. However, then the second block would be not repairable with the remaining spares (one spare row and one spare column).

Only at the expense of a considerable exponential blow-up, we can show the following fixed parameter tractability result:

Theorem 3.6 Given a chip board with n elementary cells which is split into k_3 blocks each of which has at most k_1 neighbouring spare rows and k_2 neighbouring spare columns and assuming that there are, furthermore, at most k_4 linked spare rows and k_5 linked spare columns on the whole board, then a reconfiguration strategy can be found in time

$$O(k_3((k_1+k_2+k_4+k_5)n+ \binom{k_3(k_1+k_4)}{k_4}\binom{k_3(k_2+k_5)}{k_5}[1.3999^{k_1+k_2}k_1k_2+(\min\{k_1,k_2\}+1)^{\sqrt{k_3}+1}]))$$

if it exists.

Proof. Such a board can be reconfigured as follows:

1. Kernelize each block assuming that there are at most $k_1 + k_4$ spare rows and at most $k_2 + k_5$ spare columns per block. The size of the problem kernel such obtained is $k_3(k_1 + k_4)(k_2 + k_5)$.

2. Consider all possible assignments of the k_4 linked spare rows to one of the $k_3(k_1 + k_4)$ possibly faulty rows and all assignments of linked spare columns to possibly faulty columns and apply the algorithm sketched in the proof of the preceding theorem to each of the remaining "boards". □

Of course, the algorithm obtained in the previous theorem is only manageable for very small values of k_3, k_4 and k_5. Again, one might think about a weighted variant of the last considered problem (which is again solvable by considering the signatures as detailed above), since a solution using one linked spare is probably to be preferred over a solution using $\approx \sqrt{k_3}$ many individual spares.

Remark 3.7 The example shown in this section proves that, from the point of view of applications, it might make perfect sense to consider problems with a certain number of parameters. Note that the philosophy behind the development of fixed parameter algorithms is that the involved parameters should be small in practice, and this is exactly what we expect for all five parameters occurring in Theorem 3.6.

4 Conclusions

We considered the problem of enumerating all solutions of a given problem from the parameterized point of view. We coined different notions of parameterized enumeration and gave several examples, mainly from graph theory, with motivations from chip fabrication. We have shown that kernelizations as well as search trees (which are the most prominent ways to devise fixed parameter decision algorithms) are very useful techniques also for parameterized enumeration.

Remarkably, lower bounds and non-membership can be shown for several examples of enumeration problems and enumeration classes. In contrast, in the classical area of decision problems, mostly only relativized assertions of this kind are obtainable. We showed that answers to enumeration problems can be used in solutions of decision problems, as well. In particular, we proved several more realistic scenarios of the chip reconfiguration problem considered in [10] to be fixed parameter tractable.

Note that we deliberately focussed on considering the complexity of enumeration problems to include the time to output the solutions. Another variant where the size of the output solutions was considered as an extra sort of parameter (in the sense of providing output sensitive algorithms) was discussed by Grohe.[6] In this spirit, several papers on graph algorithms appeared, too, see, e.g., [16] and the references therein. When thinking about enumeration as some sort of preprocessing step for another algorithm which investigates all the obtained solutions, considering the size or number of output solutions as additional parameter does not seem to be reasonable.

It would be also interesting to consider the parameterized complexity of enumerating all (optimal) solutions *without repetitions*, as discussed in [16].

[6] in a talk on parameterized complexity and databases given at the Dagstuhl Workshop on Parameterized Complexity in August, 2001

Of course, one could avoid repetitions by either examining all pairs of output solutions in a postprocessing phase (which would square the already exponential running time) or by additional bookkeeping (with tables of exponential size), but possibly better solutions can be found for concrete problems.

In [9], we showed that parameterized enumeration can be also used (in principle) for proving parameterized tractability for maximization problems (in the sense elaborated in [9]), thus providing another sort of application of enumeration problems.

Finally, it would be interesting to see whether problems related to vertex cover are also fixed parameter enumerable. Here, the setting established in [18] might be helpful in order to prove enumerability results. More generally speaking, it would be interesting to develop enumeration techniques which are applicable not only to special situations. One idea would be to see whether (or in which cases) dynamic programming techniques typical for treewidth-based algorithms (see [3]) could be useful also for enumeration. This could be tricky, since new bookkeeping strategies need to be developped. In particular, such considerations might help answer the question whether or not the optima dominating set problem restricted to planar graphs is feasible or not, see [1] for the corresponding decision problem.

Acknowledgments: We thank M. R. Fellows and U. Stege for some discussions.

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed parameter algorithms for planar dominating set and related problems. In M. M. Halldórsson, editor, *7th Scandinavian Workshop on Algorithm Theory SWAT 2000*, volume 1851 of *LNCS*, pages 97–110, 2000. Long version to appear in Algorithmica.

[2] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speedup for planar graph problems. In F. Orejas, P. G. Spirakis, and J. v. Leeuwen, editors, *International Colloquium on Automata, Languages and Programming ICALP'01*, volume 2076 of *LNCS*, pages 261–272. Springer, 2001.

[3] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[4] J. Chen and I. A. Kanj. On constrained minimum vertex covers of bipartite graphs: Improved algorithms. In A. Brandstädt and V. B. Le, editors, *Graph-Theoretic Concepts in Computer Science WG'01*, volume 2204 of *LNCS*, pages 55–65. Springer, 2001.

[5] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108:23–52, 2001.

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

[7] D. Eppstein. Small maximal independent sets and faster exact graph coloring. In F. Dehne, J.-R. Sack, and R. Tamassia, editors, *Proc. 7th Workshop Algorithms and Data Structures WADS*, volume 2125 of *LNCS*, pages 462–470. Springer, 2001.

[8] R. C. Evans. Testing repairable RAMs and mostly good memories. In *Proceedings of the IEEE Int'l Test Conf.*, pages 49–55, 1981.

[9] H. Fernau. Parameterized maximization. Technical Report WSI–2001–22, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 2001.

[10] H. Fernau and R. Niedermeier. An efficient exact algorithm for constraint bipartite vertex cover. *Journal of Algorithms*, 38(2):374–410, 2001.

[11] M. K. Goldberg, T. H. Spencer, and D. A. Berque. A low-exponential algorithm for counting vertex covers. *Graph Theory, Combinatorics, Algorithms, and Applications*, 1:431–444, 1995.

[12] J. Gramm and R. Niedermeier. Quartet inconsistency is fixed parameter tractable. In A. Amir and G. M. Landau, editors, *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM 2001)*, volume 2089 of *LNCS*, pages 241–256. Springer, 2001.

[13] R. W. Haddad, A. T. Dahbura, and A. B. Sharma. Increased throughput for the testing and repair of RAMs with redundancy. *IEEE Transactions on Computers*, 40(2):154–166, Feb. 1991.

[14] S.-Y. Kuo and W. Fuchs. Efficient spare allocation for reconfigurable arrays. *IEEE Design and Test*, 4:24–31, Feb. 1987.

[15] K. Mehlhorn. *Graph algorithms and NP-completeness.* Heidelberg: Springer, 1984.

[16] S. Nakano. Efficient generation of triconnected plane triangulations. In J. Wang, editor, *Computing and Combinatorics, Proceedings COCOON 2001*, volume 2108 of *LNCS*, pages 131–141. Springer, 2001.

[17] R. Niedermeier and P. Rossmanith. Upper bounds for vertex cover further improved. In C. Meinel and S. Tison, editors, *Proceedings*

of the 16th Symposium on Theoretical Aspects of Computer Science (STACS'99), volume 1563 of *LNCS*, pages 561–570. Springer, 1999.

[18] N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. In F. Dehne, J.-R. Sack, and R. Tamassia, editors, *Proc. 7th Workshop Algorithms and Data Structures WADS*, volume 2125 of *LNCS*, pages 75–86. Springer, 2001.

[19] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal Comput.*, 8(3):410–421, 1979.