Electronic Colloquium on Computational Complexity, Report No. 106 (2007)

On Parameterized Approximability[*]

Yijia Chen[†] Martin Grohe[‡] Magdalena Grüber[§]

Abstract. Combining classical approximability questions with parameterized complexity, we introduce a theory of *parameterized approximability*. The main intention of this theory is to deal with the efficient approximation of small cost solutions for optimisation problems.

Key words. Fixed-parameter tractability, approximation algorithms, hardness of approximation.

1 Introduction

Fixed-parameter tractability and approximability are two complementary approaches to dealing with intractability: Approximability relaxes the goal of finding exact or optimal solutions, but usually insists on polynomial time algorithms, whereas fixed-parameter tractable (fpt) algorithms are exact, but may have a super-polynomial running time that is controlled by a parameter associated with the problem instances in such a way that for small parameter values the running time can still be considered efficient. Obviously, the two approaches can be combined, which is what we do in this paper. Optimisation problems are often parameterized by the cost of the solution that is to be found. That is, together with an instance we are given a parameter k, and the goal is to find a solution of size at least k (for maximisation problems) or at most k (for minimisation problems). For small k, an fpt algorithm with a running time like $O(2^k \cdot n)$ can be quite efficient. For some problems, for example minimum vertex cover, such an algorithm exists, but for many other problems it does not, under plausible complexity theoretic assumptions. For such problems, can we at least find small solutions that approximately have the desired cost k? A slightly different, but closely related question can be asked starting from approximability: Suppose we have a problem that is hard to approximate. Can we at least approximate it efficiently for instances for which the optimum is small? The classical theory of inapproximability does not seem to help answering this question, because usually the hardness proofs require fairly large solutions.

Let us illustrate this with an example: The maximum clique problem is known to be hard to approximate — unless ZPP = NP not approximable with ratio $n^{1-\varepsilon}$ for any $\varepsilon > 0$ [16] — and, most likely, not fixed-parameter tractable — the problem is W[1]-complete [10], and unless the exponential time hypothesis fails, it is not even solvable in time $n^{o(k)}$ [6, 7]. Here, and in the following, k denotes the size of the desired clique and n the number of vertices of the input graph. Now we ask: Is there an fpt algorithm that, given a graph $\mathscr{G}$ and a $k \in \mathbb{N}$, finds a clique of size $k/2$ in $\mathscr{G}$ provided $\mathscr{G}$

[*]A preliminary version of this paper appeared in the *Proceedings of the Second International Workshop on Parameterized and Exact Computation (IWPEC '06)*, volume 4169 of Lecture Notes in Computer Science, 109–120, 2006

[†]BASICS, Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030, China. `yijia.chen@cs.sjtu.edu.cn`

[‡]Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin, Germany. `grohe@informatik.hu-berlin.de`

[§]Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin, Germany. `grueber@informatik.hu-berlin.de`, phone: 00493020933088, fax: 00493020933081

ISSN 1433-8092

has a clique of size at least k. (If $\mathscr{G}$ does not have a clique of size k, the algorithm may still find a clique of size at least $k/2$, or it may reject the input.) We would call such an algorithm an *fpt approximation algorithm with approximation ratio* 2 for the clique problem. If no such algorithm exists, we may still ask if there is an algorithm that finds a clique of size $\sqrt{k}$ or even $\log k$, provided the input graph $\mathscr{G}$ has a clique of size k. As a matter of fact, it would be interesting to have an fpt approximation algorithm with approximation ratio ρ for any function ρ on the positive integers such that $k/\rho(k)$ is unbounded (for technical reasons, we also require ρ to be computable and $k/\rho(k)$ to be nondecreasing). If such an algorithm existed, then the maximum clique problem would be *fpt approximable*. It is an open problem whether the clique problem is fpt approximable; unfortunately the strong known inapproximability results for the clique problem do not shed any light on this question. Note that when we go beyond a constant approximation ratio we express the ratio as a function of the cost of the solution rather than the size of the instance, as it is usually done in the theory of approximation algorithms. This is reasonable because in our parameterized setting we are mainly interested in solutions that are very small compared to the size of the instance.

Our main contribution is a framework for studying such questions. We define fpt approximability for maximisation and minimisation problems and show that our notions are fairly robust. We also consider a decision version of the fpt approximability problem that we call *fpt cost approximability*, where instead of computing a solution of cost approximately k, an algorithm only has to decide if such a solution exists. We observe that a few known results yield fpt approximation algorithms: Oum and Seymour [18] showed that the problem of finding a clique decomposition of minimum width is fpt approximable. Based on results due to Seymour [20], Even et al. [13] showed that the directed feedback vertex set problem is fpt approximable. Then it follows from a result due to Reed et al. [19] that the linear programming dual of the feedback vertex set problem, the vertex disjoint cycle problem, is fpt cost approximable. This is interesting because the standard parameterization of this maximisation problem is W[1]-hard.

The classes of the fundamental W-hierarchy of parameterized complexity theory are defined as closures of so called weighted satisfiability problems under fpt reductions. We prove that for all levels of the W-hierarchy, natural optimisation versions of the defining weighted satisfiability problems are not fpt approximable, not even fpt cost approximable, unless the corresponding level of the hierarchy collapses to FPT, the class of fixed-parameter tractable problems. Furthermore, we prove that the short halting problem, which is known to be W[1]-complete for single tape machines and W[2]-complete in general, is not fpt cost approximable unless $\mathrm{W}[2] = \mathrm{FPT}$.

As a final result, we show that every parameterized problem in NP is both equivalent to the standard parameterization of an optimisation problem that is fpt approximable and equivalent to the standard parameterization of an optimisation problem that is not fpt cost approximable; in other words: every parameterized complexity class above FPT that contains a problem in NP contains problems that are approximable and problems that are inapproximable.

Independently, Cai and Huang [4] and Downey, Fellows and McCartin [12] introduced similar frameworks of parameterized approximability.

2 Preliminaries

$\mathbb{N}$ denotes the natural numbers (positive integers), $\mathbb{R}$ the real numbers, and $\mathbb{R}_{\geq 1}$ the real numbers greater than or equal to 1. We recall a few basic definitions on optimisation problems and parameterized complexity. For further background, we refer the reader to [2] and [11, 15].

2.1 Optimisation Problems

In this paper we consider NP-optimisation problems O over a finite alphabet Σ consisting of triples $(\mathrm{sol}_O, \mathrm{cost}_O, \mathrm{goal}_O)$ where

1. sol_O is a function that associates to any input instance $x \in \Sigma^*$ the set of feasible solutions of x such that the relation $\{(x,y) \mid x \in \Sigma^* \text{ and } y \in \mathrm{sol}_O(x)\}$ is polynomially balanced and decidable in polynomial time;

2. cost_O is the measure function and is defined on the class $\{(x,y) \mid x \in \Sigma^*, \text{ and } y \in \mathrm{sol}_O(x)\}$; the values of cost_O are positive natural numbers and cost_O is polynomial time computable;

3. $\mathrm{goal}_O \in \{\max, \min\}$

The objective of an optimisation problem O is to find an optimal solution z for a given instance x, that is a solution z with $\mathrm{cost}_O(x,z) = \mathrm{opt}_O(x) := \mathrm{goal}_O\{\mathrm{cost}_O(x,y) \mid y \in \mathrm{sol}_O(x)\}$. If O is clear from the context, we omit the subscript and just write opt, sol, cost and goal.

2.2 Parameterized Problems

We represent decision problems over a finite alphabet Σ as sets $Q \subseteq \Sigma^*$ of strings. Let us briefly recall the basic definitions of parameterized problems that we will need:

1. A *parameterization* of Σ^* is a polynomial time computable mapping $\kappa : \Sigma^* \to \mathbb{N}$.

2. A *parameterized decision problem* is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ and a parameterization κ of Σ^*.

3. An algorithm $\mathbb{A}$ with input alphabet Σ is an *fpt algorithm with respect to* κ if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for every instance $x \in \Sigma^*$ the running time of $\mathbb{A}$ on this input x is at most $f(\kappa(x)) \cdot |x|^{O(1)}$.

4. A parameterized decision problem (Q, κ) is *fixed-parameter tractable* if there is an fpt algorithm with respect to κ that decides Q. FPT denotes the class of all fixed-parameter tractable decision problems. In parameterized complexity theory the analogue to polynomial time reductions are *fpt reductions*.

5. A parameterized decision problem (Q, κ) belongs to the class XP if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm that decides if $x \in Q$ for a given $x \in \Sigma^*$ in at most $O(|x|^{f(\kappa(x))})$ steps.

An important class of parameterized problems is the class of weighted satisfiability problems. We look at weighted satisfiability problems for propositional formulas and circuits: A formula α is k-satisfiable if there exists a satisfying assignment that sets exactly k many variables to TRUE (this assignment has weight k). A circuit γ is k-satisfiable if there is a possibility of setting exactly k many input nodes to TRUE and getting the value TRUE at the output node (γ is satisfied by an input tuple of weight k). We are interested in special classes of propositional formulas, $\Gamma_{t,d}$ and $\Delta_{t,d}$, defined inductively for $t \geq 0$, $d \geq 1$ as follows:

$$\begin{aligned}
\Gamma_{0,d} &:= \{\lambda_1 \wedge \ldots \wedge \lambda_c \mid c \in [d], \lambda_1, \ldots, \lambda_c \text{ literals}\},\\
\Delta_{0,d} &:= \{\lambda_1 \vee \ldots \vee \lambda_c \mid c \in [d], \lambda_1, \ldots, \lambda_c \text{ literals}\},\\
\Gamma_{t+1,d} &:= \{\bigwedge_{i \in I} \delta_i \mid I \text{ finite and nonempty, } \delta_i \in \Delta_{t,d} \text{ for all } i \in I\},\\
\Delta_{t+1,d} &:= \{\bigvee_{i \in I} \gamma_i \mid I \text{ finite and nonempty, } \gamma_i \in \Gamma_{t,d} \text{ for all } i \in I\}.
\end{aligned}$$

For a class Γ of propositional formulas or Boolean circuits, the parameterized weighted satisfiability problem for Γ is

p-WSAT(Γ)	
Input:	$\gamma \in \Gamma$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether γ is k-satisfiable.

The problems p-WSAT$(\Gamma_{t,d})$ with $t, d \geq 1$ are used to define the classes W$[t]$ of the W-*hierarchy*: A parameterized problem (Q, κ) belongs to the class W$[t]$ if there is a $d \geq 1$ such that (Q, κ) is fpt reducible to p-WSAT$(\Gamma_{t,d})$. In the same way, the weighted satisfiability problem p-WSAT(CIRC) for the class CIRC of all Boolean circuits defines the parameterized complexity class W[P]. It holds

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W[P]} \subseteq \text{XP}$$

where FPT is known to be strictly contained in XP and all other inclusions are believed to be strict as well.

3 Parameterized Approximability

Definition 1. Let O be an NP-optimisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function. Let $\mathbb{A}$ be an algorithm that expects inputs $(x, k) \in \Sigma^* \times \mathbb{N}$.

1. $\mathbb{A}$ is a *parameterized approximation algorithm* for O with *approximation ratio* ρ if for every input $(x, k) \in \Sigma^* \times \mathbb{N}$ with $\text{sol}(x) \neq \emptyset$ that satisfies

$$\begin{cases} \text{opt}(x) \geq k & \text{if goal} = \max, \\ \text{opt}(x) \leq k & \text{if goal} = \min, \end{cases} \tag{$\star$}$$

   $\mathbb{A}$ computes a $y \in \text{sol}(x)$ such that

$$\begin{cases} \text{cost}(x, y) \geq \dfrac{k}{\rho(k)} & \text{if goal} = \max, \\ \text{cost}(x, y) \leq k \cdot \rho(k) & \text{if goal} = \min. \end{cases} \tag{$\star\star$}$$

   For inputs $(x, k) \in \Sigma^* \times \mathbb{N}$ not satisfying condition ($\star$), the output of $\mathbb{A}$ can be arbitrary.

2. $\mathbb{A}$ is an *fpt approximation algorithm* for O with *approximation ratio* ρ if it is a parameterized approximation algorithm for O with approximation ratio ρ and an fpt algorithm with respect to

the parameterization $(x,k) \mapsto k$ of its input space (that is, the running time of $\mathbb{A}$ is $f(k) \cdot |x|^{O(1)}$ for some computable function f).

$\mathbb{A}$ is a *constant fpt approximation algorithm* for O if there is a constant $c \geq 1$ such that $\mathbb{A}$ is an fpt approximation algorithm for O with approximation ratio $k \mapsto c$ (the constant function with value c).

3. The problem O is *fpt approximable with approximation ratio ρ* if there is an fpt approximation algorithm for O with approximation ratio ρ. The problem O is *fpt approximable* if it is fpt approximable with approximation ratio ρ for some computable function $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ such that

$$\begin{cases} \dfrac{k}{\rho(k)} \text{ is unbounded and nondecreasing} & \text{if } O \text{ is a maximisation problem,} \\ k \cdot \rho(k) \text{ is nondecreasing} & \text{if } O \text{ is a minimisation problem.} \end{cases}$$

O is *constant fpt approximable* if there is a constant fpt approximation algorithm for O.

Remark 2. Since it is decidable by an fpt algorithm whether an output y is an element of $\mathrm{sol}(x)$ that satisfies ($\star\star$), we can assume that an fpt approximation algorithm always (that is, even if the input does not satisfy ($\star$)) either outputs a $y \in \mathrm{sol}(x)$ that satisfies ($\star\star$) or outputs a default value, say "reject". Let us call an fpt approximation algorithm that has this property *normalised*.

Remark 3. We have decided to let the approximation ratio ρ be a function of the parameter k, because this is what we are interested in here. One could easily extend the definition to approximation ratios ρ depending on the input size as well, or even to arbitrary functions $\rho : \Sigma^* \times \mathbb{N} \to \mathbb{R}$. A technical condition that should be imposed then is that ρ be computable by an fpt algorithm with respect to the parameterization $(x,k) \mapsto k$.

Remark 4. We assume nondecreasing behaviour of $k/\rho(k)$ and $k \cdot \rho(k)$ in the definition of fpt approximable as we expect that the nearer k is to the optimum the more difficult it should be to find an approximate solution for an input (x,k).

Furthermore, we demand unboundedness for maximisation problems because if $k/\rho(k)$ was bounded then a parameterized approximation algorithm would only have to output a constant size solution. For maximisation problems – in contrast to minimisation problems – this would be no useful approximation algorithm in most cases and almost all maximisation problems would be fpt approximable.

An alternative approach to parameterized approximability could be to parameterize optimisation problems by the optimum, with the goal of designing efficient approximation algorithms for instances with a small optimum. Interestingly, for minimisation problems, this yields exactly the same notion of parameterized approximability, as the following proposition shows.

Proposition 5. *Let O be an* NP-*minimisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k \cdot \rho(k)$ is nondecreasing. Then the following two statements are equivalent:*

1. *O has an fpt approximation algorithm with approximation ratio ρ.*

2. *There exists a computable function g and an algorithm $\mathbb{B}$ that on input $x \in \Sigma^*$ computes a solution $y \in \mathrm{sol}(x)$ such that $\mathrm{cost}(x,y) \leq \mathrm{opt}(x) \cdot \rho(\mathrm{opt}(x))$ in time $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.*

Proof. To prove the implication $(1) \Rightarrow (2)$, let $\mathbb{A}$ be a normalised (see Remark 2) fpt approximation algorithm for O with approximation ratio ρ. Let f be a nondecreasing computable function such that the running time of $\mathbb{A}$ on input (x,k) is bounded by $f(k) \cdot |x|^{O(1)}$. Let $\mathbb{B}$ be the algorithm that simulates $\mathbb{A}$ on inputs $(x,1),(x,2),\dots$ until $\mathbb{A}$ does not reject for the first time for some input (x,k) and then outputs the output y of $\mathbb{A}$ on input (x,k). Since $\mathbb{A}$ only rejects for inputs (x,i) with $i < \mathrm{opt}(x)$, we know that $k-1 < \mathrm{opt}(x)$ and hence $k \leq \mathrm{opt}(x)$. Thus the running time of $\mathbb{B}$ is bounded by

$$\underbrace{\sum_{i=1}^{\mathrm{opt}(x)} f(i)}_{=g(\mathrm{opt}(x))} \cdot |x|^{O(1)}.$$

Furthermore, by $(\star\star)$ we have

$$\mathrm{cost}(x,y) \leq k \cdot \rho(k) \leq \mathrm{opt}(x) \cdot \rho(\mathrm{opt}(x)).$$

To prove the implication $(2) \Rightarrow (1)$, let $\mathbb{B}$ be an algorithm that satisfies (2), and let c be a constant such that the running time of $\mathbb{B}$ is actually bounded by $g(\mathrm{opt}(x)) \cdot |x|^c$. Without loss of generality we may assume that g is nondecreasing. Let $\mathbb{A}$ be the algorithm that on input (x,k) simulates $\mathbb{B}$ for $g(k) \cdot |x|^c$ steps and outputs the output y of $\mathbb{B}$ if $\mathbb{B}$ stops in $g(k) \cdot |x|^c$ steps and rejects otherwise. Clearly, $\mathbb{A}$ is an fpt algorithm. Furthermore, if $\mathrm{opt}(x) \leq k$ then $\mathbb{B}$ stops in $g(k) \cdot |x|^c$ steps and its output y satisfies

$$\mathrm{cost}(x,y) \leq \mathrm{opt}(x) \cdot \rho(\mathrm{opt}(x)) \leq k \cdot \rho(k).$$

□

For maximisation problems, our definition of fpt approximation algorithm does not coincide with the analogue of Proposition 5(2). Yet we do have an analogue of the implication $(1) \Rightarrow (2)$ of Proposition 5 for maximisation problems:

Proposition 6. *Let O be an* NP*-maximisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded.*

Suppose that O has an fpt approximation algorithm with approximation ratio ρ. Then there exists a computable function g and an algorithm $\mathbb{B}$ that on input $x \in \Sigma^*$ computes a solution $y \in \mathrm{sol}(x)$ such that $\mathrm{cost}(x,y) \geq \frac{\mathrm{opt}(x)}{\rho(\mathrm{opt}(x))}$ in time $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.

Proof. Let $\mathbb{A}$ be a normalised fpt approximation algorithm for O with approximation ratio ρ. Let f be a nondecreasing function such that the running time of $\mathbb{A}$ on input (x,k) is bounded by $f(k) \cdot |x|^{O(1)}$. Let $\mathbb{B}$ be the algorithm that simulates $\mathbb{A}$ on inputs $(x,1),(x,2),\dots$ until $\mathbb{A}$ rejects for the first time for some input $(x,k+1)$ and then outputs the output y of $\mathbb{A}$ on input (x,k). Since $\mathbb{A}$ only rejects for inputs (x,i) with $i > \mathrm{opt}(x)$, we know that $k+1 > \mathrm{opt}(x)$ and hence $k \geq \mathrm{opt}(x)$. Furthermore, by $(\star\star)$ and as $k/\rho(k)$ is nondecreasing and unbounded we have

$$\mathrm{opt}(x) \geq \mathrm{cost}(x,y) \geq \frac{k}{\rho(k)} \geq \frac{\mathrm{opt}(x)}{\rho(\mathrm{opt}(x))}.$$

Because of $k/\rho(k)$ being nondecreasing and unbounded and $\mathrm{opt}(x) \geq k/\rho(k)$ we also know that k is bounded by $h(\mathrm{opt}(x))$ for some computable function h. Thus the running time of $\mathbb{B}$ is bounded by

$$\underbrace{\sum_{i=1}^{h(\mathrm{opt}(x))} f(i)}_{=g(\mathrm{opt}(x))} \cdot |x|^{O(1)}.$$

□

The problem with the converse direction is best illustrated for NP-optimisation problems where the optimal value is always large (say, of order $\Omega(|x|)$ for every instance x). Then an algorithm $\mathbb{B}$ as in Proposition 6 trivially exists even for $\rho = 1$, because all NP-optimisation problems can be solved exactly in exponential time. But this does not seem to help much for finding a solution of size approximately k for a given, small value of k in time $f(k) \cdot |x|^O(1)$ for some computable function f.

However, for maximisation problems with certain "self-reducibility" properties the existence of the two versions of approximation algorithms nevertheless coincides. The basic idea is as follows: Suppose that we can efficiently transform an instance x of a problem to a smaller instance x' such that all solutions of x' are also solutions of x and the optimum value of x' decreases not too much. Then we can repeatedly apply this reduction until we reach an instance where the size of the optimal value is bounded in terms of the parameter k; on this instance the two notions of approximability coincide.

The following definition will make this precise:

Definition 7. Let O be an NP-maximisation problem and $c \in \mathbb{N}$. O is *well-behaved for parameterized approximation (with constant c)* if the following holds:

(i) Given an instance $x \neq \varepsilon$ for O it is possible to construct a new instance x' for O in time polynomial in $|x|$ such that $|x'| < |x|$ and $\operatorname{opt}(x) \geq \operatorname{opt}(x') \geq \operatorname{opt}(x) - c$ (assuming here that the empty string ε is a possible input for every maximisation problem and that $\operatorname{opt}(\varepsilon) := 0$).

(ii) For every instance x it holds that a valid solution for the constructed instance x' is also a valid solution for x, i.e. $\operatorname{sol}(x') \subseteq \operatorname{sol}(x)$.

Example 8. MAX-CLIQUE and MAX-DIRECTED-VERTEX-DISJOINT-CYCLES are well-behaved.

Proposition 9. *Let O be an* NP-*maximisation problem over the alphabet Σ that is well-behaved for parameterized approximation with constant c, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded.*

Suppose that there exists a computable function g and an algorithm $\mathbb{B}$ that given an input $x \in \Sigma^*$ computes a solution $y \in \operatorname{sol}(x)$ such that $\operatorname{cost}(x,y) \geq \frac{\operatorname{opt}(x)}{\rho(\operatorname{opt}(x))}$ in time $g(\operatorname{opt}(x)) \cdot |x|^{O(1)}$. Then O has an fpt approximation algorithm with approximation ratio ρ.

Proof. Let $\mathbb{B}$ be an algorithm as in the assumption, and let d be a constant such that the running time of $\mathbb{B}$ is actually bounded by $g(\operatorname{opt}(x)) \cdot |x|^d$. Without loss of generality we may assume that g is nondecreasing. Let $h : \mathbb{N} \to \mathbb{N}$ be a function such that $h(k) - k \geq c$ for all $k \in \mathbb{N}$, so let us just choose $h(k) := k + c$.

Then define an algorithm $\mathbb{A}$ on inputs $(x,k) \in \Sigma^* \times \mathbb{N}$ as follows: Simulate $\mathbb{B}$ for $g(h(k)) \cdot |x|^d$ steps. If $\mathbb{B}$ stops then output the output y of $\mathbb{B}$. $\mathbb{B}$ may stop although $k < h(k) < \operatorname{opt}(x)$. Then it holds $\frac{k}{\rho(k)} \leq \frac{h(k)}{\rho(h(k))} \leq \frac{\operatorname{opt}(x)}{\rho(opt(x))} \leq \operatorname{cost}(x,y)$. Otherwise we have $h(k) \geq opt(x)$. For inputs (x,k) with $k \leq \operatorname{opt}(x)$ it then holds that $\frac{k}{\rho(k)} \leq \frac{\operatorname{opt}(x)}{\rho(opt(x))} \leq \operatorname{cost}(x,y)$ (and for inputs with $k > \operatorname{opt}(x)$ the output may be arbitrary). If $\mathbb{B}$ does not stop, it holds that $h(k) < \operatorname{opt}(x)$. Then we can use the well-behaved property of O to construct an instance (x',k) for O and start the next iteration with simulating $\mathbb{B}$ on (x',k) for $g(h(k)) \cdot |x'|^d$ steps. As $|x'| < |x|$, the number of iterations until $\mathbb{A}$ terminates is at most $|x|$. Furthermore $h(k) < \operatorname{opt}(x)$ implies $k < \operatorname{opt}(x')$ for every iteration step, so it is guaranteed that for inputs with $k \leq \operatorname{opt}(x)$ a solution y with $\operatorname{cost}(x,y) \geq \frac{k}{\rho(k)}$ is computed. The running time for each iteration is bounded by $g(h(k)) \cdot |x|^{O(1)}$. Therefore $\mathbb{A}$ has a running time of $f(k) \cdot |x|^{O(1)}$ for some computable function f. $\square$

3.1 Cost Approximability

Sometimes, instead of computing an optimal solution of an optimisation problem O, it can be sufficient to just compute the cost of an optimal solution (called *evaluation problem* in [2]). This is equivalent to solving the *standard decision problem* associated with O: Given an instance x and a natural number k, decide whether

$$\begin{cases} \mathrm{opt}(x) \geq k & \text{if } O \text{ is a maximisation problem,} \\ \mathrm{opt}(x) \leq k & \text{if } O \text{ is a minimisation problem.} \end{cases}$$

If we parameterize the standard decision problem by the input number k, we obtain the *standard parameterization* of O:

Input:	$x \in \Sigma^*$, $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether $\mathrm{opt}(x) \geq k$ (if goal $=$ max) or $\mathrm{opt}(x) \leq k$ (if goal $=$ min).

To simplify the notation, for the rest of this section we only consider maximisation problems. All definitions and results can easily be adapted to minimisation problems.

What if we only want to compute the cost of the optimal solution approximately, say, with ratio ρ? On the level of the decision problem, this means that we allow an algorithm that is supposed to decide if $\mathrm{opt}(x) \geq k$ to err if k is close to the optimum. The following definition makes this precise:

Definition 10. Let O be an NP-maximisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function.

Then a decision algorithm $\mathbb{A}$ is a *parameterized cost approximation algorithm* for O with *approximation ratio* ρ if it satisfies the following conditions for all inputs $(x,k) \in \Sigma^* \times \mathbb{N}$ with $\mathrm{sol}(x) \neq \emptyset$ (for x with $\mathrm{sol}(x) = \emptyset$ the algorithm $\mathbb{A}$ can be assumed to reject for all $k \in \mathbb{N}$):

- If $k \leq \dfrac{\mathrm{opt}(x)}{\rho(\mathrm{opt}(x))}$, then $\mathbb{A}$ accepts (x,k).
- If $k > \mathrm{opt}(x)$, then $\mathbb{A}$ rejects (x,k).

The notions of an *fpt cost approximation algorithm* and a *constant fpt cost approximation algorithm* and of a problem being *(constant) fpt cost approximable* are defined accordingly.

A parameterized cost approximation algorithm may be thought of as deciding a parameterized problem that approximates the standard parameterization of an optimisation problem. This is made precise in the following simple proposition:

Proposition 11. *Let O be an* NP-*maximisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. Then the following two statements are equivalent:*

1. *O has an fpt cost approximation algorithm with approximation ratio ρ.*
2. *There exists a parameterized problem $(Q', \kappa') \in$* FPT *with $Q' \subseteq \Sigma^* \times \mathbb{N}$ and with $\kappa' : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ defined by $\kappa'(x,k) := k$ that approximately decides the standard parameterization (Q_O, κ_O) of O with approximation ratio ρ: Given input $(x,k) \in \Sigma^* \times \mathbb{N}$, if $(x,k) \in Q_O$ then $(x, \lfloor k/\rho(k) \rfloor) \in Q'$ and if $(x,k) \notin Q_O$ then $(x,k) \notin Q'$.*

Mike Fellows (in a recent Dagstuhl Seminar) proposed a taxonomy of hard parameterized problems which is based on their approximability. In his terminology, the standard parameterization of an optimisation problem is *good* if it is fixed-parameter tractable; it is *bad* if it is not good, but constant fpt cost approximable; it is *ugly* if it is not bad, but fpt cost approximable; otherwise, it is *hideous*.

The following two propositions show that the notion of fpt approximability is strictly stronger than that of fpt cost approximability:

Proposition 12. *Let O be an* NP-*maximisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded.*

Suppose that O is fpt approximable with approximation ratio ρ. Then O is fpt cost approximable with approximation ratio ρ.

Proof. Choose an algorithm $\mathbb{B}$ according to Proposition 6, and let c be a constant such that the running time of $\mathbb{B}$ is bounded by $g(\mathrm{opt}(x)) \cdot |x|^c$ for some nondecreasing computable function g. Let $\mathbb{A}$ be the following algorithm: On input $(x,k) \in \Sigma^* \times \mathbb{N}$, it simulates $\mathbb{B}$ for $g(k) \cdot |x|^c$ steps. If $\mathbb{B}$ halts and produces an output $y \in \mathrm{sol}(x)$, then $\mathbb{A}$ accepts if $k \leq \mathrm{cost}(x,y)$ and rejects otherwise. If $\mathbb{B}$ does not halt in $g(k) \cdot |x|^c$ steps, then $k < \mathrm{opt}(x)$, and $\mathbb{A}$ accepts. □

Proposition 13. *Assume that* $\mathrm{NP} \cap \text{co-NP} \neq \mathrm{P}$. *Then there exists an* NP-*optimisation problem that is fpt cost approximable but not fpt approximable.*

Proof. Let Q be a problem defined over an alphabet Σ with $Q \in \mathrm{NP} \cap \text{co-NP}$ but $Q \notin \mathrm{P}$. As $Q \in \mathrm{NP}$ there is a polynomially balanced relation R_1 that is decidable in polynomial time such that $Q = \{x \in \Sigma^* \mid \exists y : (x,y) \in R_1\}$, so y "witnesses" $x \in Q$. Analogously, as $Q \in$ co-NP, there is a polynomially balanced and polynomial time decidable relation R_2 such that y is a witness for $x \notin Q$ if $(x,y) \in R_2$. We consider the following NP-optimisation problem O:

Input:	$x \in \Sigma^*$.
Solutions:	y where y is a witness for $x \in Q$ or for $x \notin Q$.
Cost:	1.
Goal:	min.

Since the cost of any solution is 1, O is trivially fpt cost approximable. Now assume O has an fpt approximation algorithm $\mathbb{A}$. Then given any input $x \in \Sigma^*$, $\mathbb{A}(x,1)$ outputs a witness for $x \in Q$ or for $x \notin Q$ in polynomial time and therefore decides Q. □

Considering cost approximability we obtain a full analogue of Proposition 5 also for maximisation problems:

Proposition 14. *Let O be an* NP-*maximisation problem over the alphabet Σ, and let $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. Then the following two statements are equivalent:*

1. *O has an fpt cost approximation algorithm with approximation ratio ρ.*
2. *There exists a computable function g and an algorithm $\mathbb{B}$ that on input $x \in \Sigma^*$ computes an $\ell \in \mathbb{N}$ such that $\mathrm{opt}(x) \geq \ell \geq \frac{\mathrm{opt}(x)}{\rho(\mathrm{opt}(x))}$ in time $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.*

Proof. The implication (1) ⇒ (2) is proved analogously to Proposition 6, and the backward implication is proved similarly to Proposition 12. □

3.2 Examples

Example 15. We first look at the problem MIN-CLIQUE-WIDTH of computing a decomposition of minimum clique-width for a given graph. Clique-width [8] is a graph parameter that is defined by a composition mechanism for vertex-labelled graphs and measures the complexity of a graph according to the difficulty of decomposing the graph into a kind of tree-structure. A decomposition of clique-width k is also called k-expression and given the k-expression, many hard graph problems are solvable in polynomial time for graphs of bounded clique-width. Fellows et al. [14] recently proved that deciding whether the clique-width of $\mathscr{G}$ is at most k is NP-hard and that the minimisation problem MIN-CLIQUE-WIDTH cannot be absolutely approximated in polynomial time unless $\mathrm{P} = \mathrm{NP}$.

Oum and Seymour [18] defined the notion of rank-width to investigate clique-width and showed that $\operatorname{rwd}(\mathscr{G}) \leq \operatorname{cwd}(\mathscr{G}) \leq 2^{\operatorname{rwd}(\mathscr{G})+1} - 1$ for the clique-width $\operatorname{cwd}(\mathscr{G})$ and the rank-width $\operatorname{rwd}(\mathscr{G})$ of a given simple, undirected and finite graph $\mathscr{G}$. In [17], Oum presents two algorithms to compute rank-decompositions approximately: For a graph $\mathscr{G} = (V,E)$ and $k \in \mathbb{N}$ the algorithms either output a rank-decomposition of width at most $f(k)$ with $f(k) = 3k+1$ or $f(k) = 24k$, respectively, or confirm that the rank-width is larger than k where the running time of these algorithms for fixed k is $O(|V|^4)$ for the first one and $O(|V|^3)$ for the second. Returning to clique-width there therefore exist algorithms that either output an $(2^{1+f(k)} - 1)$-expression or confirm that the clique-width is larger than k and that have the above running times for fixed k.

As both algorithms fulfil the properties of parameterized approximation algorithms with approximation ratio ρ defined by $\rho(k) := (2^{1+f(k)} - 1)/k$, we get that MIN-CLIQUE-WIDTH is fpt approximable.

Example 16. One of the major open problems in parameterized complexity is whether the following problem is fixed-parameter tractable.

p-DIRECTED-FEEDBACK-VERTEX-SET	
Input:	A directed graph $\mathscr{G} = (V,E)$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether there is a set $S \subseteq V$ with $|S| \leq k$ such that $\mathscr{G} \setminus S$ is acyclic.

Although still far from settling it, we note that the corresponding optimisation problem MIN-DIRECTED-FEEDBACK-VERTEX-SET is at least fpt approximable.

It is well-known that MIN-DIRECTED-FEEDBACK-VERTEX-SET can be described by the following integer linear program for a given directed graph $\mathscr{G} = (V,E)$, where x_v is a variable for each vertex $v \in V$:

$$\begin{array}{lll} \text{Minimise} & \sum\limits_{v\in V} x_v & \\ \text{subject to} & \sum\limits_{v\in C} x_v \geq 1 & \text{for every cycle } C \text{ in } \mathscr{G}, \\ & x_v \in \{0,1\} & \text{for every vertex } v \in V. \end{array} \tag{1}$$

We denote the minimum size of a feedback vertex set in a directed graph $\mathscr{G}$ by $\tau(\mathscr{G})$ and the size of a *fractional feedback vertex set* $(x_v)_{v\in V}$ with $0 \leq x_v \leq 1$ for every $v \in V$ by $\tau^*(\mathscr{G})$, where (1) without the integrality constraints can be solved in polynomial time (see e.g. [13]). Clearly we have $\tau^*(\mathscr{G}) \leq \tau(\mathscr{G})$ and Seymour [20] proved that the integrality gap of the feedback vertex set problem can be at most $O(\log \tau^* \cdot \log\log \tau^*)$. This proof can be modified to obtain a polynomial time approximation algorithm for MIN-DIRECTED-FEEDBACK-VERTEX-SET with an approximation ratio of $O(\log \tau^* \cdot \log\log \tau^*)$ [13]. Using Proposition 5 we conclude that MIN-DIRECTED-FEEDBACK-VERTEX-SET is fpt approximable.

Example 17. The linear programming dual of MIN-DIRECTED-FEEDBACK-VERTEX-SET is the optimisation problem MAX-DIRECTED-VERTEX-DISJOINT-CYCLES, whose standard parameterization is the following problem:

p-DIRECTED-VERTEX-DISJOINT-CYCLES	
Input:	A directed graph $\mathscr{G}$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether there are k vertex-disjoint cycles in $\mathscr{G}$.

It is implicit in [21] that p-DIRECTED-VERTEX-DISJOINT-CYCLES is W[1]-hard. For the maximum number $\nu(\mathscr{G})$ of vertex-disjoint cycles and for the minimum size $\tau(\mathscr{G})$ ($\tau^*(\mathscr{G})$) of a (fractional) feedback vertex set in a given directed graph $\mathscr{G}$ it holds that $\nu(\mathscr{G}) \leq \tau^*(\mathscr{G}) \leq \tau(\mathscr{G})$. Furthermore, there is an upper bound of $\tau(\mathscr{G})$, in terms of $\nu(\mathscr{G})$ *only* as Reed et al. [19] proved the existence of a computable function $f : \mathbb{N} \cup \{0\} \to \mathbb{N}$, such that

$$\tau(\mathscr{G}) \leq f(\nu(\mathscr{G})) \tag{2}$$

for any directed graph $\mathscr{G}$. (The function f constructed in [19] is very large, a multiply iterated exponential, where the number of iterations is also a multiply iterated exponential; the best known lower bound is $f(x) \geq O(x \cdot \log x)$ for any $x \in \mathbb{N}$, a result attributed to Alon in [19].)

Together with the above inequalities, we can derive a very simple fpt cost approximation algorithm for MAX-DIRECTED-VERTEX-DISJOINT-CYCLES: Let f be the function with property (2). Without loss of generality, we can assume f is increasing and time-constructible. Now let $\iota_f : \mathbb{N} \to \mathbb{N}$ be defined by $\iota_f(n) := \min\{i \in \mathbb{N} \mid f(i) \geq n\}$. Then ι_f is nondecreasing and unbounded, $\iota_f(n)$ is computable in time polynomial in n and $\iota_f(f(k)) \leq k$ for every $k \in \mathbb{N}$. Therefore we conclude

$$\iota_f(\nu(\mathscr{G})) \leq \iota_f(\lceil \tau^*(\mathscr{G}) \rceil) \leq \iota_f(\tau(\mathscr{G})) \leq \iota_f(f(\nu(\mathscr{G}))) \leq \nu(\mathscr{G}).$$

Thus the algorithm that given an input $(\mathscr{G}, k)$ computes $\tau^*(\mathscr{G})$ in time polynomial in the size of $\mathscr{G}$ and accepts if $k \leq \iota_f(\lceil \tau^*(\mathscr{G}) \rceil)$ and rejects otherwise is an fpt cost approximation algorithm with approximation ratio ρ where $\rho(k) = k/\iota_f(k)$.

4 Inapproximability Results

Under assumptions from parameterized complexity theory, the following theorem states the non-approximability of weighted satisfiability optimisation problems for the above defined classes of propositional formulas:

Theorem 18. MIN-WSAT$(\Gamma_{t,d})$ *with $t \geq 2$ and $d \geq 1$ is not fpt cost approximable unless* $\mathrm{W}[t] = \mathrm{FPT}$, *where the optimisation problem* MIN-WSAT$(\Gamma_{t,d})$ *is defined as follows:*

Input:	*A propositional formula $\alpha \in \Gamma_{t,d}$.*
Solutions:	*All satisfying assignments for α.*
Cost:	$\max\{1,$ *weight of a satisfying assignment*$\}$.
Goal:	*min.*

Proof. Let $t \geq 2$ and $d \geq 1$. The main idea is as follows: we assume the existence of an fpt cost approximation algorithm for MIN-WSAT$(\Gamma_{t,d})$ and show that we can then solve every instance of p-WSAT$(\Gamma_{t,d})$ exactly by constructing (in polynomial time) a new formula β^* that is either k^*-satisfiable or not satisfiable at all and k^*-satisfiable if and only if the original formula is k-satisfiable (for a number $k^* \in \mathbb{N}$ dependent on k). Since p-WSAT$(\Gamma_{t,d})$ is W$[t]$-hard this would imply that W$[t]$ = FPT.

Let (α, k) be an instance of p-WSAT$(\Gamma_{t,d})$ with $\alpha \in \Gamma_{t,d}$ and $k \in \mathbb{N}$. To decide if α is k-satisfiable with a given parameterized approximation algorithm for MIN-WSAT$(\Gamma_{t,d})$ we have to assure that the new formula β^* is still contained in the class $\Gamma_{t,d}$. Therefore, as a first step we use an idea appearing in [15] (proof of lemma 7.6.) to create a propositional formula $\beta \in \Gamma_{t,d}$ having the following properties

(i) β is k^*-satisfiable with $k^* := 2k - 1$ if and only if α is k-satisfiable,

(ii) if t is even, then the new formula β is positive (β contains no negation symbols) and negative otherwise (β is in negation normal form and a negation symbol is in front of every variable).

Let $X_1, \ldots, X_n$ be the variables of the formula α. For even t we have to express a negative literal $\neg X_\ell$ with $1 \leq \ell \leq n$ positively (the case for odd t works similarly). To do this we fix an arbitrary order of the variables in α and introduce new variables $X_{i,j}$ and $Y_{i,i',j}$ for $1 \leq i, i' \leq n$ and $1 \leq j \leq k$ which indicate the following:

$X_{i,j}$: the jth variable set to TRUE is X_i,

$Y_{i,i',j}$: the jth variable set to TRUE is X_i and the $(j+1)$th is $X_{i'}$.

Then we can replace $\neg X_\ell$ with a formula saying that X_ℓ is either strictly before the first or after the last variable set to TRUE, or strictly between two successive variables set to TRUE:

$$\bigvee_{\ell < i \leq n} X_{i,1} \vee \bigvee_{1 \leq i < \ell} X_{i,k} \vee \bigvee_{1 \leq j \leq (k-1)} \bigvee_{1 \leq i < \ell < i' \leq n} Y_{i,i',j}.$$

Additionally we have to add a positive formula in $\Gamma_{t,d}$ assuring the desired behaviour of the new variables $X_{i,j}$ and $Y_{i,i',j}$ for $1 \leq i, i' \leq n$ and $1 \leq j \leq k$ but we omit the details here. As a result we get a positive formula $\beta \in \Gamma_{t,d}$ that is k^*-satisfiable if and only if α is k-satisfiable.

Now let $Z_1, \ldots, Z_{n^*}$ be the variables of the above constructed formula β. In the second step we want to create a propositional formula β^* that has the property of being either k^*-satisfiable or not satisfiable at all and for which holds:

$$\beta^* \text{ is } k^*\text{-satisfiable} \quad \Leftrightarrow \quad \beta \text{ is } k^*\text{-satisfiable} \quad \Leftrightarrow \quad \alpha \text{ is } k\text{-satisfiable}.$$

We use k^* copies of the variables $Z_1, \ldots, Z_{n^*}$ and arrange them in k^* columns $Z_{1,1}, \ldots, Z_{n^*,1}, \ldots,$ $Z_{1,k^*}, \ldots, Z_{n^*,k^*}$ to express the exactly k^*-satisfiability of the original formula β as having exactly one variable in each column and at most one in each row set to TRUE:

$$\begin{aligned}\beta^* \quad := \quad & \bigwedge_{1 \leq j \leq k^*} \bigvee_{1 \leq i \leq n^*} Z_{i,j} \\ \wedge \quad & \bigwedge_{1 \leq j \leq k^*} \bigwedge_{1 \leq i < i' \leq n^*} \neg Z_{i,j} \vee \neg Z_{i',j} \\ \wedge \quad & \bigwedge_{1 \leq i \leq n^*} \bigwedge_{1 \leq j < j' \leq k^*} \neg Z_{i,j} \vee \neg Z_{i,j'} \\ \wedge \quad & \beta',\end{aligned}$$

where $\beta' := \begin{cases} \beta \text{ with } Z_i \text{ replaced by } & \bigvee_{1 \le j \le k^*} Z_{i,j}, \quad \text{if } t \text{ is even,} \\ \beta \text{ with } Z_i \text{ replaced by } & \bigwedge_{1 \le j \le k^*} \neg Z_{i,j}, \quad \text{if } t \text{ is odd.} \end{cases}$

As the first three parts of β^* are in $\Gamma_{2,1}$ and $\beta' \in \Gamma_{t,d}$, it holds that $\beta^* \in \Gamma_{t,d}$. Given the existence of an fpt cost approximation algorithm $\mathbb{A}$ for MIN-WSAT($\Gamma_{t,d}$) with approximation ratio ρ where $\rho : \mathbb{N} \to \mathbb{R}_{\ge 1}$ is computable and $k \cdot \rho(k)$ is nondecreasing we could decide exactly if β^* is k^*-satisfiable or not as either $\mathbb{A}$ on input $(x, \lceil k^* \cdot \rho(k^*) \rceil)$ accepts or it rejects. Therefore we could also decide if α is k-satisfiable and this is only possible if $\mathrm{W}[t] = \mathrm{FPT}$. □

Similarly as above we define the problem MIN-WSAT(CIRC) to be

Input:	A Boolean circuit $\mathscr{C}$.
Solutions:	All satisfying input tuples.
Cost:	$\max\{1, \text{weight of a satisfying input tuple}\}$.
Goal:	min.

and using the same proof-ideas as before we get:

Theorem 19. MIN-WSAT(CIRC) *is not fpt cost approximable unless* $\mathrm{W[P]} = \mathrm{FPT}$.

The standard parameterization of MIN-WSAT($\Gamma_{t,d}$) and that of MIN-WSAT(CIRC) is to decide for an input (γ, k) if γ is at most k-satisfiable. This parameterized problem is fpt equivalent to the classical parameterized weighted satisfiability problem of deciding if a given propositional formula or Boolean circuit γ is exactly k-satisfiable.

For the maximisation variants MAX-WSAT($\Gamma_{t,d}$) and MAX-WSAT(CIRC) of weighted satisfiability, which have the same instances as the corresponding minimisation problems, but the goal is to find satisfying assignments of maximum weight, the inapproximability is much easier to establish. First consider the standard parameterization of these problems ("at least k-satisfiability"). As opposed to the minimisation problems, these are not fpt equivalent to the parameterized weighted satisfiability problem of deciding exact k-satisfiability. Furthermore, setting $k = 1$ an existing fpt cost approximation algorithm for one of these maximisation weighted satisfiability problems could decide the satisfiability of a given input in polynomial time: If an fpt cost approximation algorithm accepts the input $(\gamma, 1)$ then γ is satisfiable. Otherwise γ is not satisfiable at all or satisfiable and the fpt cost approximation algorithm nevertheless rejects as false rejecting answers are possible if the parameter value is close to the optimum. In the second case it holds $1 > \mathrm{opt}(\gamma)/\rho(\mathrm{opt}(\gamma))$. But as $k/\rho(k)$ is unbounded, there exists a constant c with $c/\rho(c) \ge 1$ and because of the nondecreasing behaviour of $k/\rho(k)$ we have $\mathrm{opt}(\gamma) \le c$. Then we can check for all values at most c if γ is c-satisfiable in time polynomial in the size of the input and this implies that we can distinguish satisfiable inputs from unsatisfiable ones in polynomial time. Therefore MAX-WSAT($\Gamma_{t,d}$) and MAX-WSAT(CIRC) are not fpt cost approximable unless $\mathrm{P} = \mathrm{NP}$.

We now look at the following two versions MIN-SHORT-NTM-HALT and MIN-SHORT-NSTM-HALT of optimising halting problems:

Input:	A nondeterministic Turing machine $\mathbb{M}$.
Solutions:	All accepting runs of $\mathbb{M}$ on the empty string.
Cost:	The number of steps in such an accepting run.
Goal:	min.

Input:	A nondeterministic single-tape Turing machine $\mathbb{M}$.
Solutions:	All accepting runs of $\mathbb{M}$ on the empty string.
Cost:	The number of steps in such an accepting run.
Goal:	min.

The corresponding parameterized problems p-SHORT-NTM-HALT and p-SHORT-NSTM-HALT are to decide for a given nondeterministic (single-tape) Turing machine $\mathbb{M}$ and a given parameter $k \in \mathbb{N}$ whether $\mathbb{M}$ accepts the empty string in at most k steps, and it was shown in [3], respectively [5] that the single-tape version is complete for W[1] and the other one complete for W[2].

Theorem 20. MIN-SHORT-NTM-HALT *is not fpt cost approximable unless* $\mathrm{W}[2] = \mathrm{FPT}$.

Proof. Without loss of generality the following proof only considers Turing machines that are either accepting or not halting at all. If a Turing machine $\mathbb{M}$ would halt and reject we can always define a new Turing machine with the same behaviour as $\mathbb{M}$ except that this new Turing machine starts an endless computation when reaching a rejecting state. Concerning the problem MIN-SHORT-NTM-HALT this causes no different result.

Assume there exists an fpt cost approximation algorithm $\mathbb{A}$ for MIN-SHORT-NTM-HALT with approximation ratio ρ where $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ is computable and $k \cdot \rho(k)$ is nondecreasing. We will show that using this parameterized approximation algorithm we can solve the problem p-SHORT-NTM-HALT exactly.

Given a Turing machine $\mathbb{M}$ and an integer $k \in \mathbb{N}$ as input, we define a new Turing machine $\mathbb{M}'$ that simulates $\mathbb{M}$ on the empty string for at most k steps. If no accepting state of $\mathbb{M}$ is reached after k many steps then $\mathbb{M}'$ starts an endless computation. We can construct $\mathbb{M}'$ in fixed-parameter tractable time and it holds that the Turing machine $\mathbb{M}$ is accepting the empty string in at most k steps if and only if the Turing machine $\mathbb{M}'$ is accepting the empty string in at most k steps. Furthermore, $\mathbb{M}'$ either halts after at most k steps or it does not halt at all.

Therefore the parameterized approximation algorithm $\mathbb{A}$ on input $(\mathbb{M}', \lceil k \cdot \rho(k) \rceil)$ can solve the problem p-SHORT-NTM-HALT for $(\mathbb{M}', k)$ exactly, but having the exact solution for $(\mathbb{M}', k)$ directly gives us the solution for $(\mathbb{M}, k)$. As p-SHORT-NTM-HALT is W[2]-complete, it follows that such a parameterized approximation algorithm $\mathbb{A}$ can only exist if the classes FPT and W[2] are fpt equivalent. □

Using the same proof-ideas as in the preceding theorem the following can be shown:

Theorem 21. MIN-SHORT-NSTM-HALT *is not fpt cost approximable unless* $\mathrm{W}[1] = \mathrm{FPT}$.

Recall the definition of the standard parameterization of an optimisation problem from page 8. The previous results show that each level of the W-hierarchy contains natural complete problems that are not fpt cost approximable. Our final result shows that artificial approximable and inapproximable problems of any given complexity can be constructed (as long as it is in NP, because we are dealing with NP-optimisation problems).

Theorem 22. *Let* (Q, κ) *be a parameterized problem not in* FPT *such that* $Q \in \mathrm{NP}$.

(1) (Q, κ) *is fpt equivalent to the standard parameterization of an* NP-*optimisation problem that is fpt approximable with approximation ratio 2.*

(2) (Q, κ) *is fpt equivalent to the standard parameterization of an* NP-*optimisation problem that is not fpt cost approximable.*

Proof. Let Q be defined over the alphabet Σ (with empty string ε). Since $Q \in$ NP there is a polynomially balanced and polynomial-time decidable relation R such that $Q = \{x \in \Sigma^* \mid \exists y : (x,y) \in R\}$, so y is a witness for $x \in Q$. To prove (1), we define the NP-optimisation problem O to be

Input:	$x \in \Sigma^*$.
Solutions:	(y,i) where $y = \varepsilon$ and $i = \kappa(x)$ or, if $x \in Q$, y is a witness for $x \in Q$ and $i = \kappa(x)+1$.
Cost:	$\text{cost}(x,y,i) = i$.
Goal:	max.

It is easy to see that (Q,κ) is fpt equivalent to the standard parameterization (Q_O,κ_O) of O and the algorithm $\mathbb{A}$ that outputs $(\varepsilon,\kappa(x))$ for every input $x \in \Sigma^*$ is a constant fpt approximation algorithm for O with approximation ratio 2.

To prove (2), we define the NP-optimisation problem O to be

Input:	$x \in \Sigma^*$.
Solutions:	(y,i) if $x \in Q$ where y is a witness for $x \in Q$ and $i = \kappa(x)$.
Cost:	$\text{cost}(x,y,i) = i$.
Goal:	min.

Again, it is easy to see, that (Q,κ) is fpt equivalent to the standard parameterization (Q_O,κ_O) of O. Assume there exists an fpt cost approximation algorithm $\mathbb{A}$ for O with approximation ratio ρ where $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ is computable and $k \cdot \rho(k)$ is nondecreasing. For a given $x \in \Sigma^*$ we could then decide the parameterized problem (Q,κ) in fpt time, as either $\mathbb{A}$ on input $(x, \lceil \kappa(x) \cdot \rho(\kappa(x)) \rceil)$ accepts and $x \in Q$ or it rejects otherwise. As this is contradicting $(Q,\kappa) \notin$ FPT, the NP-optimisation problem O is not fpt cost approximable. ☐

5 Further Research

Next to finding additional natural examples for problems with existing fpt approximation algorithms, the main open problem at the moment is to specify the parameterized approximability properties of basic problems like MIN-DOMINATING-SET or the MAX-CLIQUE problem already mentioned as an introductory example. Non-approximability results in the classical framework were proved for the CLIQUE problem using the PCP-theorem [1, 9], so it might be necessary to obtain a parameterized version of the PCP-theorem to solve these questions.

References

[1] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, Berlin Heidelberg, 2003.

[3] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic*, 36:321–337, 1997.

[4] L. Cai and X. Huang. Fixed-parameter approximation: Conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108, Berlin Heidelberg, 2006. Springer.

[5] M. Cesati and M. D. Ianni. Computation models for parameterized complexity. *Mathematical Logic Quarterly*, 43:179–202, 1997.

[6] J. Chen, B. Chor, M. R. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 150–160, 2004.

[7] J. Chen, X. Huang, I. Kanj, and G. Xia. Linear fpt reductions and computational lower bounds. In *Proceedings of the 36th ACM Symposium on Theory of Computing, STOC 2004*, pages 212–221, 2004.

[8] B. Courcelle, J. Engelfriet, and G. Rozenberg. Context-free handle-rewriting hypergraph grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Graph-Grammars and their Application to Computer Science, Fourth International Workshop*, volume 532 of *Lecture Notes in Computer Science*, pages 253–268, 1991.

[9] I. Dinur. The pcp theorem by gap amplification. In *Proceedings of the 38th ACM Symposium on Theory of Computing, STOC 2006*, pages 241–250, 2006.

[10] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Journal of Theoretical Computer Science*, 141:109–131, 1995.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, New York, 1999.

[12] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation algorithms. In H. L. Bodlaender and M. A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129, Berlin Heidelberg, 2006. Springer.

[13] G. Even, J. S. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

[14] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard. In *Proceedings of the 38th ACM Symposium on Theory of Computing, STOC 2006*, pages 354–362, 2006.

[15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, Berlin Heidelberg, 2006.

[16] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Electronic Colloquium on Computational Complexity*, Report TR97-038, 1997.

[17] S. Oum. Approximating rank-width and clique-width quickly. In D. Kratsch, editor, *31th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2005*, volume 3787 of *Lecture Notes in Computer Science*, pages 49–58, Berlin Heidelberg, 2005. Springer.

[18] S. Oum and P. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[19] B. Reed, N. Robertson, P. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.

[20] P. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[21] A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In G. D. Battista and U. Zwick, editors, *Proceedings of the 11th Annual European Symposium on Algorithms, ESA '03*, volume 2832 of *Lecture Notes in Computer Science*, pages 482–493, 2003.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/