On Parameterized Approximability

Yijia Chen[1], Martin Grohe[2], and Magdalena Grüber[2]

[1] BASICS, Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030, China. yijia.chen@cs.sjtu.edu.cn
[2] Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin, Germany. grohe@informatik.hu-berlin.de, grueber@informatik.hu-berlin.de

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.

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 $\epsilon > 0$ [15] — and, most likely, not fixed-parameter tractable — the problem is W[1]-complete [9], and unless the exponential time hypothesis fails, it is not even solvable in time $n^{o(k)}$ [6]. 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 $\mathcal{G}$ and a $k \in \mathbb{N}$, finds a clique of size

$k/2$ in $\mathcal{G}$ provided $\mathcal{G}$ has a clique of size at least k. (If $\mathcal{G}$ 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 $\mathcal{G}$ 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 [17] showed that the problem of finding a clique decomposition of minimum width is fpt approximable. Based on results due to Seymour [19], Even et al. [12] showed that the directed feedback vertex set problem is fpt approximable. Then it follows from a result due to Reed et al. [18] 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 W[2] = 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 param-

eterized 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 [11] introduced similar frameworks of parameterized approximability.

Due to space limitations, proofs are omitted here, but can be found in the full version of this paper.

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 [10, 14].

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 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:

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

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* $\mathrm{sol}(x) \neq \emptyset$ *that satisfies*

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

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

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

   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 $(x,k) \mapsto k$.

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 4. *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 $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.*

For maximisation problems, our definition of fpt approximation algorithm does not coincide with the analogue of Proposition 4(2). Yet we do have an analogue of the implication (1) $\Rightarrow$ (2) of Proposition 4 for maximisation problems:

Proposition 5. *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 $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.

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). An example of such a problem is the maximum independent set problem on planar graphs. Then an algorithm $\mathbb{B}$ as in Proposition 5 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.

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 $\text{opt}(x) \geq k$ (if goal = max) or $\text{opt}(x) \leq k$ (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 $\text{opt}(x) \geq k$ to err if k is close to the optimum. The following definition makes this precise:

Definition 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.*

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 $(x, k) \in \Sigma^* \times \mathbb{N}$:*

- *If $k \leq \dfrac{\text{opt}(x)}{\rho(\text{opt}(x))}$, then $\mathbb{A}$ accepts (x, k).*
- *If $k > \text{opt}(x)$, 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 7. *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 $(Q', \kappa') \in \text{FPT}$ with $Q' \subseteq \Sigma^* \times \mathbb{N}$ and with $\kappa' : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ defined by $\kappa'(x, k) := k$ that "approximates" 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*.

Proposition 8. *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.

The following two propositions show that for cost approximability, we obtain a full analogue of Proposition 4 for maximisation problems and that the notion of fpt approximability is strictly stronger than that of fpt cost approximability:

Proposition 9. *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 exist a computable function g and an algorithm $\mathbb{B}$ that on input $x \in \Sigma^*$ computes an $\ell \in \mathbb{N}$ with $\mathrm{opt}(x) \geq \ell \geq \frac{\mathrm{opt}(x)}{\rho(\mathrm{opt}(x))}$ in time $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$.*

Proposition 10. *Assume that* NP $\cap$ co-NP $\neq P$. *Then there exists an NP-optimisation problem that is fpt cost approximable but not fpt approximable.*

3.2 Examples

Example 11. Min-Clique-Width is the problem of computing a decomposition of minimum clique-width for a given graph where Clique-width [7] 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. Given a decomposition of clique-width k (also called k-expression), many hard graph problems are solvable in polynomial time for graphs of bounded clique-width. Fellows et al. [13] recently proved that deciding whether the clique-width of $\mathcal{G}$ is at most k is NP-hard and that the minimisation problem Min-Clique-Width cannot be absolutely approximated in polynomial time unless P = NP.

Oum and Seymour [17] defined the notion of rank-width to investigate clique-width and showed that $\mathrm{rwd}(\mathcal{G}) \leq \mathrm{cwd}(\mathcal{G}) \leq 2^{\mathrm{rwd}(\mathcal{G})+1} - 1$ for the clique-width $\mathrm{cwd}(\mathcal{G})$ and the rank-width $\mathrm{rwd}(\mathcal{G})$ of a given simple, undirected and finite graph $\mathcal{G}$. In [16], Oum presents two algorithms to compute rank-decompositions approximately: For a graph $\mathcal{G} = (V, E)$ 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 12. 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 $\mathcal{G} = (V, E)$ and $k \in \mathbb{N}$. *Parameter:* k. *Problem:* Decide whether there is a set $S \subseteq V$ with $\lvert S\rvert \leq k$ such that $\mathcal{G} \setminus S$ 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 $\mathcal{G} = (V, E)$, where x_v is a variable for each vertex $v \in V$:

$$\begin{aligned} &\text{Minimise } \sum_{v \in V} x_v \\ &\text{subject to } \sum_{v \in C} x_v \geq 1 \text{ for every cycle } C \text{ in } \mathcal{G}, \\ &\qquad x_v \in \{0, 1\} \text{ for every vertex } v \in V. \end{aligned} \tag{1}$$

We denote the minimum size of a feedback vertex set in a directed graph $\mathcal{G}$ by $\tau(\mathcal{G})$ 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 $\tau^*(\mathcal{G})$, where (1) without the integrality constraints can be solved in polynomial time (see e.g. [12]). Clearly we have $\tau^*(\mathcal{G}) \leq \tau(\mathcal{G})$ and Seymour [19] 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 $O(\log \tau^* \cdot \log \log \tau^*)$ [12]. Using Proposition 4 we conclude that Min-Directed-Feedback-Vertex-Set is fpt approximable.

Example 13. 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 $\mathcal{G}$ and $k \in \mathbb{N}$. *Parameter:* k. *Problem:* Decide whether there are k vertex-disjoint cycles in $\mathcal{G}$.

It is implicit in [20] that *p*-Directed-Vertex-Disjoint-Cycles is W[1]-hard. For the maximum number $\nu(\mathcal{G})$ of vertex-disjoint cycles and for the minimum size $\tau(\mathcal{G})$ ($\tau^*(\mathcal{G})$) of a (fractional) feedback vertex set in a given directed graph $\mathcal{G}$ it holds that $\nu(\mathcal{G}) \leq \tau^*(\mathcal{G}) \leq \tau(\mathcal{G})$. Furthermore, there is an upper bound of $\tau(\mathcal{G})$, in terms of $\nu(\mathcal{G})$ *only* as Reed et al. [18] proved the existence of a computable function $f : \mathbb{N} \cup \{0\} \to \mathbb{N}$, such that

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

for any directed graph $\mathcal{G}$. (The function f constructed in [18] 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 [18].)

Together with the above inequalities, we can derive a very simple fpt cost approximation algorithm for MAX-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

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

Thus the algorithm that given an input $(\mathcal{G}, k)$ computes $\tau^*(\mathcal{G})$ in time polynomial in the size of $\mathcal{G}$ and accepts if $k \leq \iota_f(\lceil \tau^*(\mathcal{G}) \rceil)$ 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 14. MIN-WSAT($\Gamma_{t,d}$) *with* $t \geq 2$ *and* $d \geq 1$ *is not fpt cost approximable unless* W[t] = 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:* $\max\{1, \textit{weight of a satisfying assignment}\}$.
> *Goal: min.*

Similarly as above we define the problem MIN-WSAT(CIRC) and maximisation versions of weighted satisfiability problems and get the following results:

Theorem 15. MIN-WSAT(CIRC) *is not fpt cost approximable unless* W[P] = FPT.

Theorem 16.

(1) MAX-WSAT($\Gamma_{t,d}$) *with* $t \geq 2$ *and* $d \geq 1$ *is not fpt cost approximable unless* W[t] = FPT.
(2) MAX-WSAT(CIRC) *is not fpt cost approximable unless* W[P] = FPT.

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, which is W[2]-complete [3] (W[1]-complete [5], respectively).

Theorem 17. Min-Short-NTM-Halt *is not fpt cost approximable unless* W[2] = FPT.

Theorem 18. Min-Short-NSTM-Halt *is not fpt cost approximable unless* W[1] = FPT.

Recall the definition of the standard parameterization of an optimisation problem from page 6. 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 19. *Let* (Q, κ) *be a parameterized problem not in* FPT *such that* $Q \in$ 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.*

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, 8], 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), pages 70–122, 1998.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer-Verlag, 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. These proceedings.
5. M. Cesati and M. Di Ianni. Computation models for parameterized complexity. *Mathematical Logic Quarterly*, 43:179–202, 1997.
6. J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 150–160, 2004.
7. 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, 4th International Workshop*, Proceedings, volume 532 of Lecture Notes in Computer Science, pages 253–268, 1991.
8. I. Dinur. The PCP theorem by gap amplification. *Proceedings of STOC 2006*, 38th ACM Symposium on Theory of Computing, Seattle, Washington, USA, to appear.
9. R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Journal of Theoretical Computer Science*, volume 141, pages 109–131, 1995.
10. R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
11. R.G. Downey, M.R. Fellows, and C. McCartin. Parameterized Approximation Algorithms. These proceedings.
12. 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.
13. M. Fellows, F. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard. *Proceedings of STOC 2006*, 38th ACM Symposium on Theory of Computing, Seattle, Washington, USA, to appear.
14. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
15. J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Electronic Colloquium on Computational Complexity*, Report TR97-038, 1997.
16. S. Oum. Approximating rank-width and clique-width quickly. *31th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2005*, volume 3787 of Lecture Notes in Computer Science, pages 49–58, 2005.
17. S. Oum and P. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory*, Series B, to appear.
18. B. Reed, N. Robertson, P. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.
19. P. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.
20. A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In G. D. Battista and U. Zwick editors, *Proceedings of 11th Annual European Symposium on Algorithms, ESA'03*, volume 2832 of Lecture Notes in Computer Science, pages 482–493, 2003.