Supplementary material for:

Similarity as tractable transformation

by Moritz Müller, Iris van Rooij, Todd Wareham

In this supplementary material, we provide proofs of the complexity-theoretic statements appearing in our text. First we briefly recall some concepts and notations from parameterized complexity theory (Section 1) and review the problems we are concerned with in our text (Section 2). We then verify statements made in our text about the classical (Section 3) and parameterized (Section 4) (in)tractability of these problems.

1 Preliminaries from parameterized complexity theory

We assume that the reader is familiar with basic notions from classical complexity theory but review some basics of parameterized complexity that are relevant for our purpose. For details the reader is referred to the sources mentioned in the text.

In parameterized complexity theory, instances of problems come along with a *parameterization*. This is a function κ computable in polynomial time which associates with every instance $x \in \{0,1\}^*$ its *parameter* $\kappa(x)$. Thus *parameterized problems* (Q, κ) are considered as pairs of classical problems $Q \subseteq \{0,1\}^*$ and parameterizations κ. A parameterized problem (Q, κ) is *fixed-parameter tractable* if and only if it is decidable whether or not $x \in Q$ in time

$$f(\kappa(x)) \cdot |x|^{O(1)}.$$

Here $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary computable function. An algorithm with such a running time for a parameterized problem is called an *fpt algorithm*. The class of all fixed-parameter tractable parameterized problems is denoted by FPT. For example, a running time $2^{\kappa(x)} \cdot |x|$ which is exponential only in the parameter but linear in the instance size $|x|$ is considered feasible. So, intuitively, we are interested in instances where the parameter is "small" compared to the instance size $|x|$.

Relative to this notion of tractability a theory of parameterized intractability has been developed. This has led to a variety of parameterized classes classifying seemingly intractable problems. The most important of these constitute the so-called *W-hierarchy*

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

It is widely believed that the W-hierarchy is strict. These classes of the W-hierarchy can be defined in terms of parameterized problems on Boolean formulas. We view Boolean formulas as special Boolean circuits and write them in a linear fashion as is usual. For a set of Boolean formulas Γ, consider the following parameterized weighted satisfiability problem:

p-WSAT(Γ)	
Input:	$\alpha \in \Gamma$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Does α have a satisfying assignment of weight k?

Here the *weight* of an assignment is the number of variables set to 1 (“true”). The W-hierarchy can be defined by the those problems for different Γ — namely, for W[t], formulas of weft t and constant depth are used. The *weft* of a circuit is the maximal number of nested “large” gates of the circuit, where a gate is large if it has fan-in more than two. For simplicity and as it is sufficient for our purposes here, we give the definition for $t = 2$:

Definition 1 W[2] is the class of those parameterized problems which are for some $c \in \mathbb{N}$ fpt-reducible to p-WSAT($\Gamma_{2,c}$). A formula is in $\Gamma_{2,c}$ if and only if it is of the form

$$\bigwedge_i \bigvee_j (\lambda_{ij1} \wedge \ldots \wedge \lambda_{ijc})$$

where the λ’s are literals (variables or negated variables) and i, j range over arbitrary finite index sets, e.g. $\Gamma_{2,1}$ is the set of Boolean formulas in conjunctive normal form.

Definition 2 W[P] is the class of those parameterized problems fpt-reducible to p-WSAT(CIRC), where CIRC denotes the class of all Boolean circuits.

For W[P] the following machine characterization is known:

Theorem 3 *A parameterized problem (Q, κ) is in* W[P] *if and only if there is an fpt-time nondeterministic Turing machine deciding Q which for some computable function $g : \{0,1\}^* \to \mathbb{N}$ on input x performs at most $g(\kappa(x)) \cdot \log |x|$ nondeterministic steps.*

The above employ the following notion of reduction.

Definition 4 An *fpt reduction* from (Q, κ) to (Q', κ') is an fpt-computable function $R : \{0,1\}^* \to \{0,1\}^*$ such that for some computable function $g : \mathbb{N} \to \mathbb{N}$,

- $x \in Q$ if and only if $R(x) \in Q'$, and
- $\kappa'(R(x)) \leq g(\kappa(x))$.

Let $\mathcal{C}$ be a class of parameterized problems. A parameterized problem (Q, κ) is said to be $\mathcal{C}$-*complete* or *complete for* $\mathcal{C}$ if and only if $(Q, \kappa) \in \mathcal{C}$ and (Q, κ) is $\mathcal{C}$-*hard*, i.e. every problem in $\mathcal{C}$ is fpt-reducible to (Q, κ).

Example 5 The following parameterized problem is complete for W[2]:

p-HITTING SET	
Input:	A hypergraph $\mathcal{H}$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Does $\mathcal{H}$ have a hitting set of k elements?

Here a *hypergraph* is a pair (V, E) for a nonempty set V of vertices and a set E of *hyperedges*, i.e. subsets of V. A *hitting set* of $\mathcal{H}$ is a subset $X \subseteq V$ intersecting every hyperedge, i.e. $X \cap e \neq \emptyset$ for all $e \in E$. Note that p-HITTING SET is a parameterized version of the following classical NP-complete problem:

HITTING SET
Input: A hypergraph $\mathcal{H}$ and $k \in \mathbb{N}$.
Problem: Does $\mathcal{H}$ have a hitting set of cardinality k?

2 Problems

2.1 Optimization problems

In our text we introduced the following problems:

REPRESENTATIONAL DISTORTION (version 1) [RD1]
Input: Two representations a and b and a set of basic transformations T.
Output: The length of a shortest sequence of basic transformations from T transforming a to b.

REPRESENTATIONAL DISTORTION (version 2) [RD2]
Input: Two representations a and b, a set of basic transformations $\mathcal{T}$, and a 'context' C.
Output: A number that equals the length of a shortest sequence of basic transformations from $T_C \subseteq \mathcal{T}$ transforming a to b, where T_C is a set of transformations that is 'most relevant' for 'context' C.

REPRESENTATIONAL DISTORTION (version 3) [RD3]
Input: Two representations a and b, a set of basic transformations $\mathcal{T}$, and a set of pairs of object representations X with $(a, b) \in X$.
Output: A number that equals the length of a shortest sequence of basic transformations from $T_X \subseteq \mathcal{T}$ transforming a to b, where T_X is a set of transformations in $\mathcal{T}$ that is most relevant for context X.

RELEVANCE
Input: A set of basic transformations $\mathcal{T}$, a set of pairs of object representations X, and integers s and w.
Output: A set of transformations $T_X \subseteq \mathcal{T}$ of minimum size such that for at least s pairs $(a, b) \in X$ there exists a sequence of transformations in T_X having length at most w that when applied to a yields b.

REPRESENTATIONAL DISTORTION (version 4) [RD4]
Input: Two representations a and b, a set of basic transformations $\mathcal{T}$, a set of pairs of object representations X with $(a, b) \in X$, and integers s and w.
Output: A number that equals the length of a shortest sequence of basic transformations from $T_X \subseteq \mathcal{T}$ transforming a to b where T_X is a solution of RELEVANCE$(\mathcal{T}, X, s, w)$.

All RD versions are to be understood in the following way: if a number as asked for in the output does not exist then an algorithm solving the problem has to report that, say by outputting a special symbol ∞. We adopt a similar convention for RELEVANCE. For example, an algorithm solving RD4 outputs ∞ on inputs $(a, b, \mathcal{T}, X, s, w)$ for which there is no solution of RELEVANCE$(\mathcal{T}, X, s, w)$.

2.2 Decision versions

To prove (classical or parameterized) hardness of the optimization problems defined above in Section 2.1, we consider the decision versions of these problems. To formulate such decision problems, we shall use the following mode of speech:

Definition 6 Let $a, b \in \{0,1\}^*$ and let T be a set transformations (i.e. of Boolean circuits). The *transformational distance from a to b with respect to T* is the minimal $k \in \mathbb{N}$ (if there is such a k) such that there are $C_1, \dots, C_k \in T$ for which $b = C_1 \circ \cdots \circ C_k(a)$. If there is no such k, the transformational distance from a to b with respect to T is ∞.

Definition 7 Let X be a set of pairs of binary strings and let T be a set of transformations. The *transformational diameter of X with respect to T* is the minimal $k \in \mathbb{N} \cup \{\infty\}$ such that for each pair $(a, b) \in X$ the transformational distance from a to b with respect to T is at most k.

Naturally, here it is to be understood that ∞ is bigger than any natural number.

Given the above, decision versions of RD1 and RELEVANCE read as follows:

SIMILARITY
Input: $(a, b) \in (\{0,1\}^*)^2$, a set of transformations T and $k \in \mathbb{N}$.
Problem: Is the transformational distance from a to b with respect to T at most k?

D-RELEVANCE
Input: $X \subseteq (\{0,1\}^*)^2$, a set of transformations $\mathcal{T}$ and $t, s, w \in \mathbb{N}$.
Problem: Is there a subset $T_C \subseteq \mathcal{T}$ with $|T_C| \leq t$ and a subset $X' \subseteq X$ with $|X'| \geq s$ such that the transformational diameter of X' with respect to T_C is at most w?

Remark 8 Clearly the problem SIMILARITY is a subproblem of RELEVANCE in the sense that instances of SIMILARITY correspond to instances of D-RELEVANCE with $s = |X| = 1$ and $t = |\mathcal{T}|$.

3 Classical intractability

In the section "Representational Distortion is Intractable" we claimed that well-defined RD models, i.e. RD1, RELEVANCE, and RD4, are classically intractable. This claim rests on the following result:

Theorem 9 *SIMILARITY and D-RELEVANCE are NP-hard.*

Proof: By Remark **??** it suffices to give a polynomial-time reduction from HITTING SET to SIMILARITY. Let an instance $(\mathcal{H}, k)$ of HITTING SET be given. Say $\mathcal{H} = (V, E)$ contains m hyperedges $E = \{e_1, \dots, e_m\}$. For a vertex $v \in V$ let C_v be the circuit with m input nodes and m output nodes which computes on a string $a = (a_1, \dots, a_m) \in \{0,1\}^m$ the string $a' = (a'_1, \dots, a'_m)$ such that for all $i \in [m]$

$$a'_i = \begin{cases} 1 & \text{if } v \in e_i \\ a_i & \text{else} \end{cases} .$$

Then $\mathcal{H}$ has a hitting set of size at most k if and only if the instance $((a,b), T, k)$ of SIMILARITY with $a := \underbrace{0 \cdots 0}_{m \text{ times}}, b := \underbrace{1 \cdots 1}_{m \text{ times}}, T := \{C_v \mid v \in V\}$ is a "yes" instance. □

For later use we prove the following:

Theorem 10 *The following problem is* PSPACE-*hard:*

∞-SIMILARITY	
Input:	$(a,b) \in (\{0,1\}^*)^2$ and a set of transformations T.
Problem:	Is the transformational distance of a to b with respect to T finite?

To prove this result, we will use the following well-known connection between Boolean circuits and Turing machines.

Lemma 11 *Let $t : \mathbb{N} \to \mathbb{N}$ with $t(n) \geq n$ for all $n \in \mathbb{N}$. Further let $\mathbb{A}$ be a Turing machine running in time t. Then for each $n \in \mathbb{N}$ there is a circuit C_n with n input nodes such that $\mathbb{A}(a) = C_n(a)$ for all $a \in \{0,1\}^n$. Furthermore C_n can be computed from n in time polynomial in $t(n)$.*

Proof of Theorem **??***:* Let $\mathbb{A}$ be a Turing machine solving a PSPACE-complete problem $P \subseteq \{0,1\}^*$ in space $n^{O(1)}$. Without loss of generality we can assume that there is exactly one accepting configuration of $\mathbb{A}$ on inputs of length n. We reduce P to ∞-SIMILARITY. Let an instance x of P be given, say of length n.

Configurations of $\mathbb{A}$ on inputs of length n can be encoded by binary strings of length ℓ_n for ℓ_n polynomially bounded in n. Let $\mathbb{A}'$ be a polynomial-time algorithm which given a encoding of a configuration of $\mathbb{A}$ computes its successor configuration, i.e. the configuration reached by $\mathbb{A}$ from the given configuration in one step. Apply Lemma **??** to $\mathbb{A}'$ to get in polynomial time a circuit C_n with ℓ_n inputs and outputs computing on a (or rather, the encoding of a) configuration of $\mathbb{A}$ on inputs of length n the successor configuration (and a dummy string not encoding a configuration in case the input is not of the required form).

Let `start` be the starting configuration of $\mathbb{A}$ on x and let `acc` be the (unique) accepting configuration of $\mathbb{A}$ on inputs of length n. Then $x \in P$ if and only if $((\texttt{start}, \texttt{acc}), \{C\})$ is a "yes" instance of ∞-SIMILARITY. □

A proof of the PSPACE-completeness of ∞-SIMILARITY will appear in the journal version of our paper.

4 Parameterized complexity analyses

We now prove Results 1 to 10 on the parameterized (in)tractability of RD-models as stated in section "Results and Discussion".

4.1 Result 1: RD1 is fp-intractable for parameter set $\{t, \ell_1\}$

It suffices to show that the following parameterized (decision) problem is not fixed-parameter tractable unless P = PSPACE:

p-$\{t, \ell_1\}$-SIMILARITY	
Input:	$(a, b) \in \{0, 1\}^*$, a set of transformations T and $k \in \mathbb{N}$.
Parameter:	$\lvert T\rvert + \max\{\lvert a\rvert, \lvert b\rvert\}$.
Problem:	Is the transformational distance from a to b with respect to T at most k?

This easily follows from the proof of Theorem **??**: first note that an algorithm solving p-$\{t, \ell_1\}$-SIMILARITY also solves ∞-SIMILARITY. Note also that we constructed a polynomial-time reduction from a PSPACE-complete P to instances of ∞-SIMILARITY with $|T| = 1$, i.e. instances of the form $(a, b, \{C\})$. Let C_a and C_b be circuits computing a from 0 and 1 from b respectively. On other strings these circuits compute a dummy string (recall the proof of Theorem **??**). Then $(0, 1, \{C, C_a, C_b\})$ is an equivalent instance of ∞-SIMILARITY that, when viewed as an instance of p-$\{t, \ell_1\}$-SIMILARITY, has constant parameter 4. An fpt algorithm for p-$\{t, \ell_1\}$-SIMILARITY could therefore be used to solve P in polynomial time. Hence there is no such algorithm unless P = PSPACE.[1]

4.2 Result 2: RD1 is fp-intractable for parameter set $\{k, \ell_1\}$

It suffices to show that the following parameterized (decision) problem is not fixed-parameter tractable unless W[P] = FPT:

p-$\{k, \ell_1\}$-SIMILARITY	
Input:	$(a, b) \in \{0, 1\}^*$, a set of transformations T and $k \in \mathbb{N}$.
Parameter:	$k + \max\{\lvert a\rvert, \lvert b\rvert\}$.
Problem:	Is the transformational distance from a to b with respect to T at most k?

Theorem 12 *p-$\{k, \ell_1\}$-SIMILARITY is* W[P]*-hard.*

Proof: Let $(Q, \kappa) \in$ W[P]. We have to find a fpt reduction from (Q, κ) to p-$\{k, \ell_1\}$-SIMILARITY. By the machine characterization in Theorem **??**, there is a nondeterministic Turing machine $\mathbb{A}$ and computable functions $f, g : \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that $\mathbb{A}$ on input $x \in \{0, 1\}^*$ decides whether or not $x \in Q$ in time at most $f(\kappa(x)) \cdot |x|^c$ and makes at most $g(\kappa(x)) \cdot \lceil \log |x| \rceil$ nondeterministic guesses. We can assume that a nondeterministic step consists in nondeterministically writing 0 or 1 on a tape. For technical reasons we assume without loss of generality that $\mathbb{A}$ on an input x makes *exactly* $g(\kappa(x)) \cdot \lceil \log |x| \rceil$ guesses and that it stops immediately after its last guess.

[1]This result also holds in the even stronger case when we consider so-called xp algorithms, i.e. algorithms with running times bounded by $f(\kappa(x)) \cdot |x|^{g(\kappa(x))}$ for arbitrary computable functions f and g.

Let an input $x \in \{0,1\}^*$ be given, and set $k := g(\kappa(x))$. Divide the computation of $\mathbb{A}$ on x into k stages, each with exactly $\lceil \log |x| \rceil$ guesses. Each stage is a computation mapping a configuration and a string in $\{0,1\}^{\lceil \log |x| \rceil}$ (encoding the $\lceil \log |x| \rceil$ guesses) to another configuration. We want a circuit doing this computation. Clearly we intend to apply Lemma **??**. However, a technical difficulty is that our bound on the running time of $\mathbb{A}$ on x is not of the form $t(|x|)$ but rather of the more general form $t(x)$.

We do not have this problem if $|x| > f(\kappa(x))$, because then the running time of $\mathbb{A}$ on x is bounded by $|x|^{c+1}$. This is good enough: on an input x our reduction first computes $f(\kappa(x))$. If $|x| \le f(\kappa(x))$ it runs a decision procedure $\mathbb{A}_Q$ for Q on x (Note that $(Q,\kappa) \in$ W[P] implies that Q is decidable). The reduction then maps x to some fixed "yes" instance of p-$\{k, \ell_1\}$-SIMILARITY if $x \in Q$, and to some fixed "no" instance otherwise. We can assume that $\mathbb{A}_Q$ on x needs time at most $t(|x|)$ for some computable nondecreasing $t : \mathbb{N} \to \mathbb{N}$. Then $\mathbb{A}$ on x with $|x| \le f(\kappa(x))$ needs to simulate at most $t(f(\kappa(x)))$ steps and hence the time $\mathbb{A}$ needs on such x is effectively bounded in terms of $\kappa(x)$.

Consider now the case in which $|x| > f(\kappa(x))$. Choose an encoding of configurations of $\mathbb{A}$ on inputs of length $|x|$ such that any such configuration is encoded by a string of length ℓ such $\ell \ge |x|$ is polynomial in $|x|$. We define a deterministic algorithm $\mathbb{A}'$ such that $\mathbb{A}'$ gets two strings as input, a binary string $a = (a_1, \ldots, a_{\lceil \log |x| \rceil})$ of length $\lceil \log |x| \rceil$ and a string of length ℓ. Given such an input $\mathbb{A}'$ first checks if the second string encodes a configuration of $\mathbb{A}$. If this is not the case $\mathbb{A}$ stops and outputs some dummy string not encoding a configuration. Otherwise it simulates $\mathbb{A}$ starting from the given configuration using the string a as outcomes of guesses, i.e. for $i \in [\lceil \log |x| \rceil]$, $\mathbb{A}'$ writes a_i when $\mathbb{A}$ wants to make its ith guess (nondeterministically writing 0 or 1). Immediately after the $\lceil \log |x| \rceil$th guess of $\mathbb{A}$, $\mathbb{A}'$ stops the simulation and outputs (the encoding of) the current configuration of $\mathbb{A}$.

With $\mathbb{A}'$ as described above, we have no time bound on $\mathbb{A}'$ because we do not know what it does on inputs which do not encode a configuration actually encountered by $\mathbb{A}$ on x. To get around this, we enhance $\mathbb{A}$ with a clock such that $\mathbb{A}'$ simulates at most $(2^{|a|})^{c+1} \ge |x|^{c+1}$ steps of $\mathbb{A}$. Note that $\mathbb{A}'$ can compute this time bound from its input in polynomial time because $\ell \ge |x|$ and $|a| = \lceil \log |x| \rceil$, so $2^{|a|} \le 2|x|$. If the last simulated step is not a guess of $\mathbb{A}$, then $\mathbb{A}'$ stops and outputs the dummy string.

Clearly the running time of $\mathbb{A}'$ is polynomially bounded in x. We now apply Lemma **??** to get a circuit $C = C(\bar{Y}\bar{Z})$ with $|\bar{Y}| = \lceil \log |x| \rceil$ and $|\bar{Z}| = \ell$ and ℓ output gates that computes, when given values $a \in \{0,1\}^{\lceil \log |x| \rceil}$ for $\bar{Y}$ and $\texttt{conf} \in \{0,1\}^\ell$ for $\bar{Z}$, the output of $\mathbb{A}'$ on a and $\texttt{conf}$. We can compute C from x in polynomial time.

Let $\texttt{start}$ be the (code of the) starting configuration of $\mathbb{A}$ on x. We assume that $\mathbb{A}$ has exactly one accepting halting configuration (with code) $\texttt{acc}$ of length ℓ. Let $m := 2^{\lceil \log |x| \rceil}$. Then $m \le 2|x|$. For $i \in [m]$ let $bin(i)$ be the binary representation of i by a string in $\{0,1\}^{\lceil \log |x| \rceil}$. For $i \in [m]$ let

$$C\frac{\mathit{bin}(i)}{\bar{Y}} = C\frac{\mathit{bin}(i)}{\bar{Y}}(\bar{Z})$$

denote the circuit obtained from C by fixing the inputs $\bar{Y}$ to the values $bin(i)$. Then $\mathbb{A}$ accepts x if and only if there are $i_1, \ldots, i_k \in [m]$ such that

$$\texttt{acc} = C\frac{\mathit{bin}(i_1)}{\bar{Y}} \circ \cdots \circ C\frac{\mathit{bin}(i_k)}{\bar{Y}}(\texttt{start}).$$

Thus $x \in Q$ if and only if $((a, b), T, k) \in p\text{-}\{k, \ell_1\}$-SIMILARITY for

$$\begin{aligned} a &:= \texttt{start}, \\ b &:= \texttt{acc}, \\ T &:= \{C_{\frac{bin(1)}{Y}}, \dots, C_{\frac{bin(m)}{Y}}\}. \end{aligned}$$

By adding suitable circuits C_a and C_b, we can replace a by 0 and b by 1 as seen in the proof in Section 4.1. This completes the description of a fpt reduction from (Q, κ) to $p\text{-}\{k, \ell_1\}$-SIMILARITY. □

4.3 Result 3: RD1 is fp-tractable for parameter set $\{t, k\}$

It suffices to show that SIMILARITY is fixed-parameter tractable when parameterized by $\{t, k\}$. In the proof of Result 9 we will see that D-RELEVANCE is fixed-parameter tractable when parameterized by $\{m, w\}$. This is enough by Remark **??**.

4.4 Result 4: RD1 is fp-tractable for parameter set $\{\ell_2\}$

As in Section 4.3, by Remark **??** it suffices to show fixed-parameter tractability of D-RELEVANCE parameterized by ℓ_2. We will do so in the proof of Result 10.

4.5 Result 5: RD2 and RD3 are fp-tractable for parameter set $\{t_c, k\}$

As RD2 and RD3 are informal problems, we have to make these statements of fp-tractability precise. We treat the case for RD2 (RD3 is treated completely analogous). For this case, it suffices to make precise the following informal statement:

"RD2 where T_C is non-inferentially given is fp-tractable for parameter set $\{t_c, k\}$" $(*)$

To that end we assume that any explanation of "context" is given, e.g. as is done in RD3. Formally this means that we assume that there is a polynomial-time decidable subset of $\{0, 1\}^*$ whose members are called *contexts*.

Given the above, we can formalize and prove statement $(*)$ as follows:

Proposition 13 *For* all *functions f mapping pairs $(\mathcal{T}, C)$ of a finite set of Boolean circuits $\mathcal{T}$ and a context C to a subset of $\mathcal{T}$, the parameterized problem*

$p\text{-}\{t_c, k\}$-RD2(f)	
Input:	$(a, b) \in (\{0, 1\}^*)^2$, a set of Boolean circuits $\mathcal{T}$, a context C and $k \in \mathbb{N}$.
Parameter:	$\lvert f(\mathcal{T}, C)\rvert + k$.
Problem:	Is there a length $\leq k$ sequence of circuits from $f(\mathcal{T}, C)$ transforming a to b?

can be solved by an fpt algorithm with oracle access to f.

Proof: The proof is simple. Choose by Result 3 an algorithm $\mathbb{A}$ and a computable $g : \mathbb{N} \to \mathbb{N}$ such that $\mathbb{A}$ solves the problem

Input:	$(a, b) \in (\{0,1\}^*)^2$, a set Boolean circuits T, and $k \in \mathbb{N}$.
Problem:	Is there a length $\leq k$ sequence of circuits from T transforming a to b?

in time $g(|T| + k)$ times a polynomial in the input size.

Given an instance $((a, b), \mathcal{T}, C, k)$ of RD2(f), say of size n, call the oracle to get $f(\mathcal{T}, C)$. Then run $\mathbb{A}$ on the instance $((a, b), f(\mathcal{T}, C), k)$ of the above problem; this instance has some size $m \leq n$. $\mathbb{A}$ needs time at most $O(n) + g(|f(\mathcal{T}, C)| + k) \cdot m^{O(1)}$. We thus have an fpt algorithm for $\{t_c, k\}$-RD2(f). □

4.6 Result 6: RD2 and RD3 are fp-tractable for parameter set $\{\ell_2\}$

This can be shown analogously to Result 5, now employing Result 4 instead of Result 3.

4.7 Result 7: RD4 is fp-intractable for parameter set $\{m\}$

By Result 1 and Remark **??**, D-RELEVANCE parameterized by $\{m\}$ is not fixed-parameter tractable unless P = PSPACE even when restricted to instances with $s = |X|$ and $t = |\mathcal{T}|$. An algorithm $\mathbb{A}$ solving RD4 can be used to solve D-RELEVANCE for these instances as follows: on input $(X, \mathcal{T}, |\mathcal{T}|, |X|, w)$ run $\mathbb{A}$ on $(0, 0, \mathcal{T}, X \cup \{(0,0)\}, |X \cup \{(0,0)\}|, w)$; then $\mathbb{A}$ answers 0 in case $(X, \mathcal{T}, |\mathcal{T}|, |X|, w)$ is a "yes"-instance of D-RELEVANCE and ∞ otherwise. It follows that RD4 parameterized by $\{m\}$ is not fixed-parameter tractable unless P = PSPACE.

4.8 Result 8: RD4 is fp-intractable for parameter set $\{t_c, w\}$

As in Section 4.7, it suffices to show that D-RELEVANCE parameterized by $\{t_c, w\}$ is unlikely to be fp-tractable. We can even show fp-intractability when the problem is restricted to instances with $w = 1$.

Theorem 14 *The following parameterized problem is* W[2]*-hard:*

p-t-RELEVANCE*	
Input:	$X \subseteq (\{0,1\}^*)^2$, a set of transformations T and $t \in \mathbb{N}$.
Parameter:	t.
Problem:	Is there a subset $T_C \subseteq T$ with $|T_C| \leq t$ such that the transformational diameter of X with respect to T_C is at most 1?

Proof: It suffices to give an fpt reduction from p-HITTING SET to p-t-RELEVANCE*. Let an instance $(\mathcal{H}, t)$ of p-HITTING SET be given. Without loss of generality we assume that all hyperedges are nonempty (otherwise $(\mathcal{H}, t)$ is a "no" instance). Say $\mathcal{H} = (V, E)$ has n nodes $V = \{v_1, \ldots, v_n\}$ and m hyperedges $E = \{e_1, \ldots, e_m\}$.

For $j \in [m]$ let $bin(j)$ be the binary representation of j by a string in $\{0,1\}^{\lceil \log m \rceil}$. Further let $X_1, \ldots, X_{\lceil \log m \rceil}$ be variables. For $a \in \{0,1\}^{\lceil \log m \rceil}$ let $\alpha_{\bar{X}=a}$ be a formula in these variables satisfied exactly by a. More precisely, for a binary string $a = a_1 \cdots a_{\lceil \log m \rceil}$ we set

$$\alpha_{\bar{X}=a} := \bigwedge_{i \in [\lceil \log m \rceil]} X_i^{a_i},$$

where for a variable X we write $X^0 := \neg X$ and $X^1 := X$. For each vertex $v \in V$ define

$$C_v := \bigvee_{j \in [m] \text{ such that } v \in e_j} \alpha_{\bar{X}=bin(j)}.$$

Thus C_v is satisfied by exactly the binary representations of indices of edges containing v. Hence $(\mathcal{H}, t) \in p$-HITTING SET if and only if (X, T, t) with

$$X := \{(bin(1), 1), \dots, (bin(m), 1)\} \text{ and } T := \{C_{v_1}, \dots, C_{v_n}\}$$

is a "yes" instance of p-t-RELEVANCE*. □

4.9 Result 9: RD4 is fp-tractable for parameter set $\{m, w\}$

It suffices to show that D-RELEVANCE is fixed-parameter tractable when parameterized by $\{m, w\}$. This is so because granted such an algorithm $\mathbb{A}$, we can solve RD4 as follows:

> On an instance $(a, b, \mathcal{T}, X, s, w)$ with $m = |\mathcal{T}|$ we first compute all minimal $T' \subseteq \mathcal{T}$ such that an s-element subset of X has transformational diameter at most w: this can be done by running $\mathbb{A}$ on $(T', X, |T'|, s, w)$ for all $T' \subseteq \mathcal{T}$. Note that there are only 2^m subsets of $\mathcal{T}$, and this is effectively bounded in the parameter.
>
> For each such T' we test if (a, b) has transformational distance at most w with respect to T', that is, if there is some RELEVANCE solution such that the corresponding $\geq s$-element subset of X can be assumed to contain (a, b). If this is not the case, we reject (output ∞); otherwise we compute the transformational distance of (a, b) with respect to T'. The test and the final distance computation can be done by solving instances (a, b, T', k) of SIMILARITY for $k = 1, \dots, w$. We can do this efficiently by Remark **??**.

We show that D-RELEVANCE is fixed-parameter tractable when parameterized by $\{m, w\}$ using the following "brute force" algorithm $\mathbb{A}$: On an instance $(X, \mathcal{T}, t, s, w)$ with $t \leq m$ and, say, $X = \{(a_1, b_1), \dots, (a_n, b_n)\}$ and $\mathcal{T} = \{C_1, \dots, C_m\}$, $\mathbb{A}$ tests for all t-tuples $(i_1, \dots, i_t) \in [m]^t$ the following: $\mathbb{A}$ makes a check for each $i \in [n]$, namely it checks if the transformational distance of a_i to b_i is at most w; that can be done by testing for each $w' \leq w$ and each tuple $(j_1, \dots, j_{w'}) \in [t]^{w'}$ if $b_i = C_{i_{j_1}} \circ \cdots \circ C_{i_{j_{w'}}}(a_i)$.

$\mathbb{A}$ accepts if for some t-tuple at least s of its checks are successful. Otherwise $\mathbb{A}$ rejects. In total, $\mathbb{A}$ has to perform for each of the m^t t-tuples n checks; a check can be done by at most t^w polynomial-time computations. This amounts to an fpt running time overall.

4.10 Result 10: RD4 is fp-tractable for parameter set $\{\ell_2\}$

By logic similar to that in Section 4.9, it suffices to show that D-RELEVANCE is fixed-parameter tractable when parameterized by $\{\ell_2\}$. To that end, it is sufficient to define a *kernelization*, that is, a polynomial-time reduction $\mathbb{K}$ of the parameterized problem to itself such that for all instances x, $|\mathbb{K}(x)|$ is effectively bounded in terms of the parameter. Composing $\mathbb{K}$ with an arbitrary decision procedure for D-RELEVANCE then defines the wanted fpt algorithm.

Our kernelization is defined as follows: Let $(\mathcal{T}, X, t, s, w)$ be an instance of D-RELEVANCE with parameter ℓ_2. There are at most 2^{ℓ_2+1} strings of length at most ℓ_2, so X has cardinality at most $2^{2\ell_2+2}$. We do not care about the precise bound, and simply say that X has size bounded by $f_1(\ell_2)$ for some computable f_1.

Choose a computable f_2 such that there are at most $f_2(\ell_2)$ functions from strings of some fixed length $\leq \ell_2$ to strings of some fixed length $\leq \ell_2$. Any $C \in \mathcal{T}$ computes such a function. For two given circuits $C, C' \in \mathcal{T}$, test if they compute the same function: evaluate them on all possible arguments; the number of arguments is at most 2^{ℓ_2} and each evaluation can be done in polynomial time; hence the test can be done in fpt time. If C and C' compute the same function delete one of the circuits from $\mathcal{T}$.

Continuing this way you end up with a new set of transformations $\mathcal{T}'$ of cardinality at most $f_2(\ell_2)$. Replace each $C \in \mathcal{T}'$ by an equivalent circuit of size exponential in the number of its input nodes, say, of size at most $f_3(\ell_2)$ for some computable f_3. Say, you end up with $\mathcal{T}''$. Then $\mathcal{T}''$ has size $O(f_2(\ell_2) \cdot f_3(\ell_2))$. The numbers t and s can be assumed to be bounded by $|\mathcal{T}''|$ and $|X|$. The number w can be assumed to be bounded by the number of sequences of *pairwise different* strings of length at most ℓ_2, something effectively bounded in ℓ_2. So we can choose a computable f_4 such that all numbers have encoding size at most $f_4(\ell_2)$. Our kernelization thus outputs $(\mathcal{T}'', X, t, s, w)$, an instance of size $O(f_2(\ell_2) \cdot f_3(\ell_2) + f_1(\ell_2) + f_4(\ell_2))$, completing the proof.