ELSEVIER

Annals of Pure and Applied Logic 82 (1996) 1–15

ANNALS OF PURE AND APPLIED LOGIC

Sparse parameterized problems

Marco Cesati [a,*], Michael R. Fellows [b]

[a] *Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", via Salaria 113, 00198 Roma, Italy*

[b] *Department of Computer Science, University of Victoria, Victoria, British Columbia, V8W 3P6 Canada*

Received 11 May 1995
Communicated by A. Nerode

Abstract

Sparse languages play an important role in classical structural complexity theory. In this paper we introduce a natural definition of *sparse* problems for parameterized complexity theory. We prove an analog of Mahaney's theorem: there is no sparse parameterized problem which is hard for the tth level of the W hierarchy, unless the W hierarchy itself collapses up to level t. The main result is proved for the most general form of parametric many:1 reducibility, where the parameter functions are not assumed to be recursive. This provides one of the few instances in parameterized complexity theory of a full analog of a major classical theorem. The proof involves not only the standard technique of left sets, but also substantial circuit combinatorics to deal with the problem of small weft, and a diagonalization to cope with potentially nonrecursive parameter functions. The latter techniques are potentially of interest for further explorations of parameterized complexity analogs of classical structural results.

1. Introduction

Many natural computational problems have input that consists of a pair of items. Suppose that such a problem is NP-complete; what can be said about its complexity when the parameter is held fixed? In many practical applications, efficient algorithms for a small range of parameter values may be useful.

Informally, a parameterized problem is "fixed-parameter tractable" if, for any fixed value of the parameter, there exists an algorithm to solve the problem in time bounded by a polynomial of constant degree (that is, the degree of the polynomial does not depend on the parameter value).

Downey and Fellows have established in a series of recent papers [1, 2, 5–9] the framework of a completeness theory with which to address the apparent fixed-parameter intractability of many parameterized problems. In particular, they define a hierarchy

* Corresponding author. E-mail: cesati@dsi.uniroma1.it

SSDI 0168-0072(95)00069-0

of classes of parameterized problems (the "W hierarchy") and show that a variety of natural problems are complete for various levels of this hierarchy. The principal classes of the hierarchy are denoted

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

where FPT indicates the class of fixed-parameter tractable problems.

For example, the parameterized problem CLIQUE (given a graph G and an integer k, decide if G contains a clique of size k) is complete for W[1]. Therefore, if there is an algorithm which decides CLIQUE with running time bounded by $f(k)|G|^{\alpha}$ (where f is an arbitrary function and the constant α does not depend of k), then a similar algorithm exists for any problem in W[1], that is, all problems in W[1] would be fixed-parameter tractable. For another example, the k-time parameterized form of the HALTING PROBLEM that asks whether a nondeterministic Turing machine has a k-time-step accepting computation on an input x is also complete for W[1] and thus is fixed-parameter tractable if and only if CLIQUE is fixed-parameter tractable [9].

1.1. Structural aspects of the W hierarchy

It is easy to show that if P = NP, then the W hierarchy collapses (FPT = W[P]). Proving directly that a parameterized problem is not fixed-parameter tractable would therefore seem difficult.

A weak converse connection has been shown by Abrahamson, Downey and Fellows:

W[1] $\subseteq$ FPT if and only if 3SAT can be solved in time $p(n) \cdot 2^{o(v)}$, where p is a polynomial, n is the size of the boolean expression, and v is the number of variables.

The setting of parameterized complexity introduces substantial technical challenges for nearly every conjectured analog of classical structural results (including ordinary hardness and completeness results). An explanation of the difficulty of the subject is that while ordinary polynomial-time reducibilities are Σ_2 in arithmetical complexity, parameterized reducibilities are Σ_4 (Σ_3 in the case of strong reducibility) and indeed yield index sets that are Σ_4 complete (Σ_3 complete for strong reducibility)

For the most natural form of parameterized many:1 reducibility all of the following "obvious" possible analogs of classical structural results remain open.

- An analog of Ladner's theorem. (This has been established only for strong parameterized reducibility, where the functions of the parameters are required to be recursive [7].)
- Propagation either upwards or downwards of collapse in the W hierarchy.
- An analog of Savitch's theorem relating k-space deterministic and nondeterministic Turing machine computation.
- An analog of Toda's theorem. (Some partial results in this direction are described in [10].)

Furthermore, it is not at all clear if the above possible analogs of classical structural results are likely to be *true*, or how difficult it may be to prove or disprove them. Experience so far seems to indicate that new and more powerful techniques than in the classical case are often required for headway on analogous parameterized structural questions.

In this paper we define a natural notion of "sparse parameterized problems" and prove a full analog of Mahaney's theorem. A classical computational problem A is *sparse* if there exists a polynomial q such that for all $n > 0$ the cardinality of the set $\{x \in A : |x| \leqslant n\}$ is bounded by $q(n)$. The concept of sparseness has played an important role in classical structural complexity theory. In particular, we have the influential result due to Mahaney [12]:

There is no sparse NP-hard problem, unless P = NP.

The definition of "sparseness" can be naturally extended to parameterized problems: informally, a parameterized problem L is *sparse* if and only if there is a constant α such that, for any fixed value of the parameter k, the corresponding language $\{x : (x,k) \in L\}$ is (classically) sparse and the bounding polynomial has degree α.

In the next section we will prove that

For any $t \geqslant 1$, there is no sparse parameterized problem which is hard for the tth level of the W hierarchy, unless the W hierarchy collapses up to level t.

Our proof is based on substantial modifications and extensions of methods introduced in the classical setting. In [13], Ogiwara and Watanabe generalized Mahaney's Theorem to bounded-Turing reducibility; then Homer and Longpré [11] simplified the proof due to Ogiwara and Watanabe. We use a technique derived from a restriction of the proof of Homer and Longprè to the case of polynomial-time many:1 reducibility (this restriction is due to R. Gavaldà and is reported in [3]). To adapt this technique to the setting of parameterized complexity we must deal with two sources of difficulty. A first problem arises from the definition of the W-classes, which essentially limits us to very restricted computational power, especially in the case of $W[1]$. We address this with several combinatorial tricks in the design of bounded-depth circuits. A second problem arises from the fact that the parameter functions might not be recursive. We handle this by incorporating the left sets algorithm inside of a diagonalization against all possible values of the relevant parameter function $f(k)$.

1.2. The formal framework

Let us give the basic formal definitions of parameterized complexity theory.

A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is a fixed alphabet. For convenience, we consider that a parameterized problem L is a subset of $\Sigma^* \times \mathbb{N}$. For a parameterized problem L and $k \in \mathbb{N}$ we write L_k to denote the associated fixed-parameter problem $\{x : (x,k) \in L\}$ (namely, a *slice*).

A parameterized problem L is (uniformly) *fixed-parameter tractable* if there is a constant α and an algorithm Φ such that Φ decides if $(x,k) \in L$ in time $f(k)|x|^{\alpha}$ where $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary function. We designate the class of fixed-parameter tractable problems FPT.

Let L_1, L_2 be parameterized problems. We say that L_1 is (uniformly many:1) *reducible* to L_2 if there is a constant α and an algorithm Φ which transforms (x,k) into $(x', g(k))$ in time $f(k)|x|^{\alpha}$, where $f, g : \mathbb{N} \to \mathbb{N}$ are arbitrary functions, so that $(x,k) \in L_1$ if and only if $(x', g(k)) \in L_2$.

The reduction is said to be *strong* if the function f is recursive.

A logical circuit is of *mixed type* if it has gates of the two kinds:

1. *Small gates*: *not* gates, *and* gates and *or* gates with bounded fan-in (we will assume fan-in 1 for *not* gates and fan-in 2 for *or* gates and *and* gates).
2. *Large gates*: *And* gates and *Or* gates with unrestricted fan-in.

The *depth* of a circuit C is the maximum number of gates (small or large) on an input–output path of C. The *weft* of a circuit C is the maximum number of large gates on an input–output path of C. A family of decision circuits F has *bounded depth* (resp. *bounded weft*) if there is a constant K such that every circuit in the family F has depth (resp. weft) at most K.

The *weight* of a boolean vector x is the number of 1's in the vector.

Let F be a family of decision circuits (possibly having many different circuits with a given number of inputs). We associate to F the *parameterized circuit problem*

$$L_F = \{ (C,k) : C \in F \text{ and } C \text{ accepts an input vector of weight } k \}.$$

A parameterized problem L belongs to W[t] if there exists a constant h such that L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t,h)$ of mixed type decision circuits of weft at most t and depth at most h.

A parameterized problem L belongs to W[P] if L reduces to the circuit problem L_F, where F is the set of all circuits (no restrictions).

This gives us the hierarchy of parameterized complexity classes

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

for which there are many natural hard or complete problems [9, 5, 4].

To refer to the components of a length n vector u we will write $u = (u[1], \ldots, u[n])$.

2. Sparse parameterized problems

A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is said to be *sparse* if there exist an arbitrary function $f : \mathbb{N} \to \mathbb{N}$ and a constant α such that for all $n > 0$ and $k \geqslant 0$,

$$|\{ (x,k) \in L : |x| \leqslant n \}| \leqslant f(k) n^{\alpha}.$$

In other words, a parameterized problem L is sparse if and only if any slice L_k is (classically) sparse and the degree of the bounding polynomial does not depend of k.

2.1. Witness problem

In the following, let us consider a fixed class W[t].

Let $L \in$ W[t]. By definition, there exist two constants h, α, two functions $f, g : \mathbb{N} \to \mathbb{N}$ and a deterministic Turing machine M such that, for every $(x,k) \in \Sigma^* \times \mathbb{N}$, the machine M on input (x,k) computes (C,k') with running time bounded by $f(k)|x|^\alpha$, where $k' = g(k)$ and C is a depth h, weft t circuit such that C accepts an input vector of weight k' if and only if $(x,k) \in L$. We will say that M is a *witness* Turing machine for L. Observe that the function f could be not recursive.

Let us define the *witness problem for L (with respect to M)*:

Instance: A binary string v, and a word x over the alphabet Σ of M.

Parameter: An integer k.

Question: Is it the case that $(x,k) \in L$, and moreover $|v| = n$, weight$(v) = k'$, and $C(v)$ accepts, where (C,k') is the output of $M(x,k)$ and n is the number of input lines of C?

The Witness Problem (for some language L and witness machine M) encodes the pairs $\langle v,(x,k)\rangle$, where (x,k) is a yes-instance of L and v is a binary vector which witnesses about the membership of (x,k) in L. The following lemma shows that we can efficiently check if a binary string v is the witness for a particular instance (x,k); this is quite similar to the characterization of NP in terms of problems with easy-to-check candidate solutions.

Lemma 1. *Let $L \in$ W[t], let M be a witness Turing machine for L and let W be the witness problem for L (with respect to M). Then $W \in$ FPT.*

Proof. Let (v, x, k) be an instance of W; the following algorithm decides if $(v,x,k) \in W$: simulate $M(x,k)$ and compute (C,k'); check if $|v|$ is equal to the number of input lines of C, if weight$(v) = k'$, and if $C(v)$ accepts. The algorithm accepts if and only if all tests are passed. The running time of the algorithm is bounded by $f'(k)(|v|+|x|)^{\alpha'}$, for some function f' and constant α'. In fact, the M's computation has at most $f(k)|x|^\alpha$ steps; therefore the size of the description of the circuit C is bounded by $f(k)|x|^\alpha$, and the above tests can be performed in polynomial time in the size of the description of C, where the degree of the polynomial does not depend of k. □

2.2. Maximal witness vector

Let $u, v \in \{0,1\}^*$, and let $\tilde{u}$ (respectively, $\tilde{v}$) be the integer having u (respectively, v) as binary representation. We will write $u \prec v$ if and only if either $|u| < |v|$, or $|u| = |v|$ and $\tilde{u} \leqslant \tilde{v}$ (that is, u precedes v in the lexicographical order). It is easy to verify that the relation $\prec$ is reflexive and transitive.

Let $L \in \mathrm{W}[t]$, let M be a witness Turing machine for L, and let W be the witness problem for L (with respect to M). Let us define the 'projection' of W:

$$W_{(x,k)} = \{ v : (v,x,k) \in W \}.$$

For every (x,k), $W_{(x,k)}$ is finite, because there are at most 2^n binary strings of length n (where n is the number of input lines of the circuit C computed by $M(x,k)$). Observe that if $(x,k) \notin L$, then $W_{(x,k)}$ is the empty set. For every $(x,k) \in L$, define the *maximal witness vector* $v_{\max}^{x,k}$ as the maximal element (in the above defined lexicographical ordering) in $W_{(x,k)}$.

2.3. *Left problem*

Let us define now the *left problem for L (with respect to M)*:
Instance: A binary string v; a word x over the alphabet Σ of M.
Parameter: An integer k.
Question: Is it the case that $(x,k) \in L$ and moreover $v \prec v_{\max}^{x,k}$?

We will denote the left problem for L (with respect to M) by $\Pi^{L,M}$; informally, $\Pi^{L,M}$ contains, for any $(x,k) \in L$, all the binary strings v which precede $v_{\max}^{x,k}$ in the lexicographical ordering (that is, at the *left* of $v_{\max}^{x,k}$). The following theorem is an important brick of our proof.

Theorem 1. *Let $L \in \mathrm{W}[t]$, and let M be a witness Turing machine for L. Then $\Pi^{L,M} \in \mathrm{W}[t]$.*

Proof. We have to show that there exist functions $f, g : \mathbb{N} \to \mathbb{N}$, constants h', α and a Turing machine M' such that, for every (v,x,k), $M'(v,x,k)$ computes (in at most $f(k)(|v| + |x|)^{\alpha}$ steps) a circuit $\overline{C}$ of depth at most h' and weft at most t, and an integer $\overline{k} = g(k)$, such that $(v,x,k) \in \Pi^{L,M}$ if and only if there is an input vector z of weight $\overline{k}$ such that $\overline{C}(z)$ accepts.

Consider the deterministic Turing machine M' which implements the following algorithm:

input (v,x,k)
simulate $M(x,k)$ and get (C,k')
compute the number n of input lines of C
if $(|v| < n)$ **then print** (C,k'); **stop**
if $(|v| > n)$ **then print** (C_N,k'); **stop**
{ C_N is a trivial circuit which rejects any input }
{ here $|v| = n$ }
construct a weft t circuit C' which simulates C and
  accepts a weight k'' input vector if and only if $v \prec v_{\max}^{x,k}$
  and there exists a weight k' vector u such that $C(u) = 1$
print (C',k'');
stop

As is often the case in parameterized complexity, we are compelled to make separate arguments for the cases of $t \geqslant 2$ and $t = 1$.

Case A: *weft* $t \geqslant 2$. In this case, the circuit C' has the same input lines of C; this is its informal description:

> C' simulates C and performs the logical *and* between C's output and the carry line of the sum of the 2-complement w of $0v$(with respect to $n+1$ bits) and the input lines.

Claim. *The circuit C' has weft t and bounded depth.*

Let $a[1],\ldots,a[n]$ be the input lines of C (and C'); let $a[n+1]$ be a lines with enforced value 0 (the input lines should encode a nonnegative value in the 2-complement $(n+1)$-bits binary representation); and let $b[n+1],\ldots,b[1]$ be the lines which encode w (actually, the value of w is hard-wired into C', and therefore the circuit can be simplified). We are only interested into the carry line of the $(n+1)$-bits sum of w and the C''s input lines:

$$\text{carry line} = \bigvee_{i=1}^{n} \left[a[i] \wedge b[i] \wedge \bigwedge_{k=i+1}^{n+1} (a[k] \vee b[k]) \right].$$

Moreover, C''s output is the output of a small *and* gate between the C's output and the carry line. Therefore, if h is a bound on the depth of C, the depth of C' is bounded by $h' = \max\{h+1, 4\}$. However, the weft of C' still remains t, since we assumed that $t \geqslant 2$. Observe that the subcircuit which computes the carry line has weft 2, and therefore we cannot apply this method to the class W[1] as well.

Claim. *Let $(\overline{C}, \overline{k})$ be the output of $M'(v,x,k)$; then $(v,x,k) \in \Pi^{L,M}$ if and only if there exists a boolean vector z of weight $\overline{k}$ such that $\overline{C}(z)$ accepts.*

Observe that in any case $\overline{k} = k'$, where k' is the integer computed by $M(x,k)$.

Let us suppose that there exists an input vector z of weight k' such that $\overline{C}(z)$ accepts. Therefore it is not the case that $|v| > n$ (if $\overline{C} = C_N$, then $\overline{C}$ always rejects).

Case 1: $|v| < n$. In this case, $\overline{C} = C$, and therefore (by definition of M) $(x,k) \in L$. Moreover, $v \prec v^{x,k}_{\max}$, because $|v| < n = |v^{x,k}_{\max}|$. Thus $(v,x,k) \in \Pi^{L,M}$.

Case 2: $|v| = n$. In this case, $\overline{C} = C'$. By construction of C', $C(z)$ accepts, and therefore there exists a weight k' input vector accepted by C. Thus $(x,k) \in L$ (by definition of M), and $z \prec v^{x,k}_{\max}$ (by definition of $v^{x,k}_{\max}$). Moreover, since the carry line must be 1, then $z+w$ has an overflow. It is easy to verify that $z+w$ has an overflow if and only if $z - v \geqslant 0$, that is, the value encoded by z is not smaller than the value encoded by v. Therefore $v \prec z$ (because $|z| = n = |v|$), and thus $v \prec v^{x,k}_{\max}$. Therefore $(v,x,k) \in \Pi^{L,M}$.

By converse, suppose that $(v,x,k) \in \Pi^{L,M}$. Therefore $(x,k) \in L$ and $v \prec v^{x,k}_{\max}$. Observe that $|v| \leqslant |v^{x,k}_{\max}| = n$, and therefore $\overline{C}$ cannot be C_N. By definitions, both

$C(v^{x,k}_{\max})$ and $C'(v^{x,k}_{\max})$ accept, and $\text{weight}(v^{x,k}_{\max}) = k'$; thus $\overline{C}(v^{x,k}_{\max})$ accepts, both in the case $|v| < n$ and in the case $|v| = n$. Therefore there exists a weight k' input vector accepted by $\overline{C}$. The claim follows.

Case B: *weft* $t = 1$. By the fact established in [9] that INDEPENDENT SET is complete for W[1], we may assume that the circuit C produced by M corresponds directly to a product-of-sums expression E of the input variables $a[1],\ldots,a[n]$ where: (1) each clause (sum) has size 2, and (2) each variable occurring in E is negated. The expression E can thus be written:

$$E = \bigwedge_{c=1}^{m} (\neg a[s_1(c)] \vee \neg a[s_2(c)])$$

where s_i is a function $s_i : \{1,\ldots,m\} \to \{1,\ldots,n\}$, for $i = 1,2$.

We begin with the following observations.

Observation 1. *Under the assumption that* $|v| = |u| = n$, *we can have* $v \prec u$ *and* $v \neq u$ *if and only if there exists* λ, $1 \leqslant \lambda \leqslant n$, *such that*:

(1) $v[i] = u[i]$ *for* $i = 1,\ldots,\lambda - 1$,

(2) $v[\lambda] = 0$,

(3) $u[\lambda] = 1$.

Suppose the weight of v is l. Let $\{q_i\}$ be the set of l indices for which $v[q_i] = 1$. (For convenience, assume $q_1 < q_2 < \cdots < q_l$.) Assume now that u represents a weight k' input vector of the circuit C; therefore we may further observe that the λ promised above must satisfy $\lambda < q_{k'}$.

Observation 2. *We can have* $v \prec v^{x,k}_{\max}$ *and* $v \neq v^{x,k}_{\max}$ *if and only if there exists* u *of weight* k' *such that* $C(u) = 1$, $v \neq u$ *and* $v \prec u$.

Observation 3. *If* v *has weight* k' *and* $C(v) = 1$ *then necessarily* $v \prec v^{x,k}_{\max}$.

We describe a boolean expression E' over the set of variables $V = V_1 \cup V_2$ where,

$$V_1 = \{ b[i,j] : 1 \leqslant i \leqslant k',\ 1 \leqslant j \leqslant n \},$$

$$V_2 = \{ d[i] : 1 \leqslant i \leqslant n+1 \}.$$

Informally, $d[i] = 1$ means that t (i.e., the minimum bit on where u and v differ) corresponds to the position i, and $b[i,j] = 1$ means that $q_i = j$ (however the $b[i,j]$'s are set to the proper values w.r.t. the q_i's only for all $j < t$).

The expression E' will be a product of sums of size at most two, and will thus correspond to a circuit of weft 1. We will use C' to denote the circuit corresponding to E'. Our construction will have the property that C' accepts an input vector of weight $k'' = k' + 1$ if and only if $v \prec v^{x,k}_{\max}$. This circuit C' will replace the circuit with

the same name in the algorithm shown in the case of weft $\geqslant 2$. Let us describe E:

$$E' = E_1 \wedge E_2 \wedge E_3 \wedge E_4 \wedge E_5 \wedge E_6 \wedge E_7 \wedge E_8$$

where

$$E_1 = \bigwedge_{i=1}^{k'} \bigwedge_{j=1}^{n-1} \bigwedge_{j'=j+1}^{n} \left(\neg b[i,j] \vee \neg b[i,j']\right)$$

(for each q_i, there is at most one j s.t. $q_i = j$)

$$E_2 = \bigwedge_{i=1}^{k'-1} \bigwedge_{i'=i+1}^{k'} \bigwedge_{j=1}^{n} \left(\neg b[i,j] \vee \neg b[i',j]\right)$$

(for each j, there is at most one q_i s.t. $q_i = j$)

$$E_3 = \bigwedge_{i=1}^{n} \bigwedge_{i'=i+1}^{n+1} \left(\neg d[i] \vee \neg d[i']\right)$$

(there is at most one position i s.t. $\lambda = i$)

$$E_4 = \bigwedge_{c=1}^{m} \bigwedge_{i=1}^{k'} \bigwedge_{i'=1}^{k'} \left(\neg b[i, s_1(c)] \vee \neg b[i', s_2(c)]\right)$$

(each *or* gate of C has output 1)

$$E_5 = \bigwedge_{r=2}^{n+1} \bigwedge_{i=1}^{k'} \bigwedge_{q=1}^{r-1} \bigwedge_{v[q]=0} (\neg d[r] \vee \neg b[i,q])$$

(for all $q < \lambda$, if $v[q] = 0$ then $q_i \neq q$)

$$E_6 = \bigwedge_{r=2}^{n+1} \bigwedge_{s=1}^{k'} \bigwedge_{q_s \leqslant r-1} (\neg d[r] \vee b[s, q_s])$$

(for all $q_s < \lambda$, $b[s, q_s] = 1$)

$$E_7 = \bigwedge_{r=1}^{n} \bigwedge_{v[r]=1} (\neg d[r])$$

(the bit λ ($\leqslant n$) of v must be set to 0)

$$E_8 = \bigwedge_{r=1}^{n} (\neg d[r] \vee b[i+1, r]) \text{ where } i = |\{ q_s : q_s \leqslant r, 1 \leqslant s \leqslant k' \}|$$

($b[i+1, \lambda] = 1$, where i denotes the last $q_i < \lambda$).

The correctness of the construction is sketched as follows. Suppose $v \prec v_{\max}^{x,k}$. We consider two cases: (1) $v = v_{\max}^{x,k}$, and (2) $v \neq v_{\max}^{x,k}$. What we must show is that there is an input vector y to C' of weight $k'+1$ such that $C'(y) = 1$.

Case (1): $v = v_{\max}^{x,k}$. The following assignment to the variables of V may be verified to yield the required input y for C'.

$$b[i,j] = \begin{cases} 1 & \text{if } j = q_i, \\ 0 & \text{otherwise,} \end{cases}$$

$$d[i] = \begin{cases} 1 & \text{if } i = n+1, \\ 0 & \text{otherwise.} \end{cases}$$

Case (2): $v \neq v_{\max}^{x,k}$. Considering Observation 1, let λ be the promised index, $\lambda < q_{k'}$, for which conditions (1)–(3) hold for $u = v_{\max}^{x,k}$. Let $q'_1 < q'_2 < \cdots < q'_{k'}$ be the indices such that $v_{\max}^{x,k}[q'_i] = 1$. The following assignment is easily verified to represent the requisite y for C'.

$$b[i,j] = \begin{cases} 1 & \text{if } j = q'_i, \\ 0 & \text{otherwise,} \end{cases}$$

$$d[i] = \begin{cases} 1 & \text{for } i = \lambda, \\ 0 & \text{otherwise.} \end{cases}$$

Conversely, suppose C' accepts a weight $k'+1$ input y, or equivalently, that the weight $k'+1$ truth assignment to V corresponding to y satisfies E'. Since E_1 and E_2 are satisfied, it must be the case that in the $k' \times n$ "array" of variables V_1, no more than one variable in each "row" and no more than one variable in each "column" is assigned the value 1. Since E_3 is satisfied, no more than one variable in V_2 is assigned the value 1. This forces exactly one variable in each row of V_1 to have the value 1, with no two of these having the same second index; it also forces exactly one variable in V_2 to have the value 1.

Let $j_1 < j_2 < \cdots < j_{k'}$ be the second indices of variables in V_1 with value 1. Let u be the length n vector of weight k' with 1's in positions $j_1, \ldots, j_{k'}$. Let δ denote the unique index, $1 \leqslant \delta \leqslant n+1$, such that $d[\delta] = 1$.

Claim. $C(u) = 1$.

If $C(u) = 0$ then because of the structure of C, there must be a clause index c such that $u[s_1(c)] = 1$ and $u[s_2(c)] = 1$. But then E_4 would not be satisfied by y.

Thus we can conclude that $(x,k) \in L$. We now consider two cases.

Case 1: $\delta = n+1$. In this case, E_5 and E_6 insure that $u = v$. By Observation 3, we have $v \prec v_{\max}^{x,k}$.

Case 2: $1 \leqslant \delta \leqslant n$. In this case, E_5 and E_6 insure that $u[i] = v[i]$ for $1 \leqslant i \leqslant \delta - 1$. The clauses of E_7 and E_8 insure that δ is such that $v[\delta] = 0$ and $u[\delta] = 1$. Thus $v \prec u$ and $v \neq u$ by Observation 1. Since $C(u) = 1$ (by the above Claim) we have $v \prec v_{\max}^{x,k}$ by Observation 2.

Now the following claim concludes the proof, since the deterministic Turing machine M' testifies that $\Pi^{L,M} \in \mathrm{W}[t]$.

Claim. *$M'(v,x,k)$ has running time bounded by $f(k)(|v|+|x|)^{\alpha}$, for some f and α.*

Since M is a witness for a fixed parameter reduction, M's simulation on (x,k) can be completed in at most $\mathcal{O}(f_1(k)|x|^{\alpha_1})$ steps, for some function f_1 and constant α_1. The length of a description of C cannot exceed the running time of $M(x,k)$, and thus the number n of input lines of C is bounded by $f_1(k)|x|^{\alpha_1}$. Therefore, the tests on the length of v can be performed in $\mathcal{O}(|v|+f_1(k)|x|^{\alpha_1})$ steps; again, the output of a description of (C,k') (or (C_N,k')) requires $\mathcal{O}(f_1(k)|x|^{\alpha_1})$ steps.

Both in the case of weft $t \geqslant 2$ and of weft $t=1$, the subcircuits C' can be constructed in $\mathcal{O}(k'^2n^2)$ steps, and thus the claim holds. □

Observe that $(x,k) \in L$ if and only if $(0^n,x,k) \in \Pi^{L,M}$, and thus L trivially reduces to $\Pi^{L,M}$.

2.4. The main result

Theorem 2. *Let $L \in \mathrm{W}[t]$, and let M be a witness Turing machine for L. If S is sparse and $\Pi^{L,M}$ reduces to S, then $L \in \mathrm{FPT}$.*

Proof. Let W be the witness problem for L (with respect to M). To show that $L \in \mathrm{FPT}$, it is sufficient to show that there is an FPT algorithm that on input (x,k) will output some element $W_{(x,k)}$ or report that $W_{(x,k)}$ is empty. What the algorithm does is to perform a pruned search of all possible witnesses for (x,k), that is, to perform a pruned search of $W_{(x,k)}$.

By hypothesis, M is the witness Turing machine for L; let us assume that, for every (x,k), $M(x,k)$ computes (C,k') in at most $f_1(k)|x|^{\alpha_1}$ steps, where C is a depth h, weft t circuit and $k'=g_1(k)$.

By hypothesis, $\Pi^{L,M}$ reduces to S; therefore there exists a deterministic Turing machine T that for every (v,x,k) computes (y,k'') in at most $f_2(k)(|v|+|x|)^{\alpha_2}$ steps, where $k''=g_2(k)$ and $(v,x,k) \in \Pi^{L,M}$ if and only if $(y,k'') \in S$.

Moreover, S is sparse; thus let us assume that

$$|\{(y,k'') \in S : |y| \leqslant p\}| \leqslant f_3(k'')p^{\alpha_3}.$$

Let us define

$$q(n,m,k) = f_3(g_2(k))f_2(k)^{\alpha_3}(n+m)^{\alpha_2\alpha_3}.$$

It is easy to verify that $q(n,m,k)$ is a bound on the cardinality of the set

$$\{(y,k'') \in S : \exists v,x \text{ with } |v|=n,\ |x|=m \text{ such that } T(v,x,k)=(y,k'')\}.$$

Observe that in the general case the function $q(n,m,k)$ is not recursive, and therefore we will adopt a diagonalization technique. Our algorithm proceeds as follows:

input (x,k)
simulate $M(x,k)$ and get (C,k')

compute the number n of input lines of C
{ n is the length of all witnesses in $W_{(x,k)}$ }
$t := 1$
repeat
  for $l := 1$ **to** t **do**
    simulate *Pruned_Search*(l,x,k) for $t + l - 1$ steps
  enddo
  $t := t + 1$
until a subroutine *Pruned_Search* returns a set I
search exhaustively I to find the maximum witness
if $v_{\max}^{x,k}$ does exist **then print** $v_{\max}^{x,k}$
else print "$(x,k) \notin L$"
stop

where the subroutine *Pruned_Search* is the following:

procedure *Pruned_Search*
input (q,x,k)
partition the set of words $\{0^n,\ldots,1^n\}$
  into q intervals of almost equal size (± 1),
  and put the intervals in the set I
while there exist intervals of size > 1 in the set I **do**
  split each interval of I into two almost equal pieces (± 1)
  { assume now that I is an ordered set $\{t_1, t_2, \ldots, t_{2q}\}$ }
  for each i, let w_i be the minimal element in t_i and
    compute $(y_i, k'') = T(w_i, x, k)$
  { now drop intervals from I until at most q remain,
    making sure not to drop the interval containing $v_{\max}^{x,k}$
    if such an interval exists }
  while there exist i,j with $i < j$ and $y_i = y_j$ **do**
    drop the intervals $t_i,\ldots,t_{j-1}$
  enddo
enddo
return (I)

Claim. *The algorithm has running time bounded by* $f(k)\,|x|^{\alpha}$, *for some function* f *and constant* α.

We have to check that each phase of the algorithm can be performed with at most $f'(k)\,|x|^{\alpha'}$ steps, for some suitable functions f' and constants α'.

The simulation of $M(x,k)$ needs $\mathcal{O}(f_1(k)\,|x|^{\alpha_1})$ steps (by hypothesis); for the same reasons, $n \leqslant f_1(k)\,|x|^{\alpha_1}$.

The algorithm performs a diagonalization on the subroutine *Pruned_Search*(l,x,k) for $l = 1,2,\ldots$ Thus it executes one step of *Pruned_Search*$(1,x,k)$, then it executes two steps of *Pruned_Search*$(1,x,k)$ and one step of *Pruned_Search*$(2,x,k)$, and so on.

Let us consider the subroutine *Pruned_Search* with input $(q(n,|x|,k),x,k)$. It is easy to verify that the external while loop is iterated at most $\mathcal{O}(n)$ times. Observe that the partitioning step is easy since it is only a matter of finding where each interval begins and ends (and this is just arithmetic). Also, splitting the intervals and finding the minimal elements is easy, again just arithmetic. Let us suppose that in some iteration the set I has exactly $q(n,|x|,k)$ intervals. Recall that $q(n,|x|,k)$ is a bound on the number of different y_i's obtained from the $T(w_i,x,k)$'s; thus it is easy to verify that at least $q(n,|x|,k)$ intervals are dropped from I in the inner while loop, and the set I has at most $q(n,|x|,k)$ intervals when the next iteration of the external while loop starts. Therefore it is also easy to verify that the subroutine *Pruned_Search* on input $(q(n,|x|,k),x,k)$ returns a set of singletons I with running time at most $f''(k)\,|x|^{\alpha''}$.

Now observe that the diagonalization process ends no later than when the subroutine *Pruned_Search*$(q(n,|x|,k),x,k)$ returns the set I. The algorithm will have spent at this moment:

$$
\begin{array}{ll}
q(n,|x|,k) + f''(k)\,|x|^{\alpha''} - 1 & \text{steps on } \textit{Pruned\_Search}(1,x,k) \\
q(n,|x|,k) + f''(k)\,|x|^{\alpha''} - 2 & \text{steps on } \textit{Pruned\_Search}(2,x,k) \\
\quad\vdots & \\
f''(k)\,|x|^{\alpha''} & \text{steps on } \textit{Pruned\_Search}(q(n,|x|,k) \\
\quad\vdots & \\
1 & \text{step on } \textit{Pruned\_Search}(q(n,|x|,k)+f''(k)\,|x|^{\alpha''} \\
 & \quad -1,x,k).
\end{array}
$$

Thus the total number of steps is

$$\mathcal{O}((q(n,|x|,k) + f''(k)\,|x|^{\alpha''})^2).$$

Finally, in the exhaustive search each interval has size 1; moreover, there are at most $q(n,|x|,k)$ such intervals. By Lemma 1, we can check with an FPT procedure if any $w_i \in W_{(x,k)}$, and we can easily remember the maximum. The claim follows.

Claim. *If* $(x,k) \in L$, *then the algorithm outputs* $v_{\max}^{x,k}$; *otherwise it outputs* "$(x,k) \notin L$".

Let us suppose that during the diagonalization some subroutine *Pruned_Search* returns a set I; then the algorithm will stop and give an answer based on the exhaustive search of the set I. Thus we have to show that any subroutine *Pruned_Search* never drops the interval containing $v_{\max}^{x,k}$ (if exists). Since eventually the algorithm executes entirely the subroutine *Pruned_Search*$(q(n,|x|,k),x,k)$ (provided that no other subroutine already returned), we have the guarantee that the algorithm ends with the correct answer.

Suppose that $(x,k) \in L$, consider the inner while loop of a subroutine *Pruned_Search*, and let $i < j$ and $y_i = y_j$. We have two cases:

Case 1: If $(y_i,k'') = (y_j,k'') \in S$, then $(w_j,x,k) \in \Pi^{L,M}$, and therefore $w_j \prec v_{\max}^{x,k}$.

Case 2: If $(y_i,k'') = (y_j,k'') \notin S$, then $(w_i,x,k) \notin \Pi^{L,M}$, and therefore $w_i \not\prec v_{\max}^{x,k}$, that is, $v_{\max}^{x,k} \prec w_i$ and $v_{\max}^{x,k} \neq w_i$.

In either case, $v_{\max}^{x,k}$ is in none of the intervals $t_i,\ldots,t_{j-1}$, so that each of these intervals can be safely deleted. We remark that the correctness of the dropping phase is not related to the value of the variable q.

By converse, suppose that $(x,k) \notin L$; then trivially the algorithm outputs "$(x,k) \notin L$", because in the exhaustive search no $w \in W_{(x,k)}$ can be found. The claim follows.

Corollary 1. *If there is a sparse problem which is hard for* W[t] ($t \geqslant 1$), *then* W[t] = FPT.

Proof. Let us consider a W[t]-complete language L. By definition, there exists a Turing machine M which is a witness for $L \in$ W[t]. By Theorem 1, $\Pi^{L,M} \in$ W[t], and therefore $\Pi^{L,M}$ reduces to S. By Theorem 2, $L \in$ FPT, and thus W[t] = FPT. □

3. Concluding remarks

In this paper we showed that for any $t \geqslant 1$, no sparse parameterized problem can be hard for W[t], unless an unlikely collapse occurs – an analog of Mahaney's Theorem. In order to understand the structure of parameterized complexity, it seems fruitful to further explore parameterized analogs of questions that have been resolved in the classical setting. It is clear that parameterization generally introduces significant additional difficulties. In most cases these difficulties appear to be quite substantial, and the questions remain open. Perhaps most notable in the regard are the present absence of an analog of Ladner's theorem for non-strong parameterized reducibilities, and of an analog of Savitch's theorem relating k-space deterministic and nondeterministic computation. It is conceivable, of course, that such analogs simply might not be true, which only makes these questions all the more interesting.

References

[1] K.A. Abrahamson, R.G. Downey and M.R. Fellows, Fixed-parameter intractability II (extended abstract), in: Proc. 10th Ann. Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 665 (Würzburg, Germany, February 1993) (Springer, Berlin) 374–385.

[2] K.A. Abrahamson, R.G. Downey and M.R. Fellows, Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues, Ann. Pure Appl. Logic 73 (3) (1995) 235–276.

[3] R.V. Book, Relativizations of the $P =?NP$ and other problems: Developments in structural complexity theory, SIAM Rev. 36 (1994) 157–175.

[4] R.G. Downey and M.R. Fellows, Parameterized complexity, Monograph in preparation.

[5] R.G. Downey and M.R. Fellows, Fixed-parameter intractability (extended abstract), in: Proc. 7th IEEE Conf. on Structure in Complexity Theory (Boston, June 1992). (IEEE Press, New York) 36–49.

[6] R.G. Downey and M.R. Fellows, Fixed-parameter tractability and completeness, Congressus Numerantium 87 (1992) 161–178.
[7] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy, in: S. Homer, K. Ambos-Spies and U. Schöning, eds., Complexity Theory: Current Research (Cambridge University Press, Cambridge, 1993) 191–226.
[8] R.G. Downey and M.R. Fellows, Parameterized computational feasibility, in: P. Clote and J. Remel, eds., Feasible Mathematics II (Birkhäuser, Boston, 1994) 219–244.
[9] R.G. Downey and M.R. Fellows, Fixed-parameter tractability and completeness I: Basic results, SIAM J. Comput. (1995) 873–921.
[10] R.G. Downey, M.R. Fellows and K.W. Regan, Parameterized circuit complexity and the W hierarchy, Theoret. Comput. Sci. Ser A, to appear.
[11] S. Homer and L. Longpré, On reductions of *NP* sets to sparse sets, in: Proc. 6th IEEE Conf. on Structure in Complexity Theory (1991) 79–88.
[12] S. Mahaney, Sparse complete sets for NP: solution of a conjecture by Berman and Hartmanis, J. Comput. System Sci. 25 (1982) 130–143.
[13] M. Ogiwara and O. Watanabe, On polynomial bounded truth-table reducibility of NP sets to sparse sets, SIAM J. Comput. 20 (1991) 471–483.