Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT•

ACADEMIC PRESS

Journal of Computer and System Sciences 67 (2003) 654–685

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

The Turing way to parameterized complexity

Marco Cesati

Department of Computer Science, Systems, and Industrial Engineering, University of Rome "Tor Vergata", via del Politecnico 1, I-00133 Rome, Italy

Received 28 September 2001; revised 2 April 2002

Abstract

We propose a general proof technique based on the Turing machine halting problem that allows us to establish membership results for the classes W[1], W[2], and W[P]. Using this technique, we prove that Perfect Code belongs to W[1], Steiner Tree belongs to W[2], and α-Balanced Separator, Maximal Irredundant Set, and Bounded DFA Intersection belong to W[P].
© 2003 Elsevier Science (USA). All rights reserved.

Keywords: Parameterized complexity; W-class membership; Turing machine; Halting problem

1. Introduction

Parameterized Complexity [10] has been introduced by Downey and Fellows during the last 10 years. It is a powerful framework in which to address the different "parameterized behavior" of many computational problems. Almost all natural problems have instances consisting of at least two logical items; many NP-complete problems admit "efficient" algorithms for small values of one item (the *parameter*).

A parameterized problem is said to be *fixed parameter tractable* if it admits a solving algorithm whose running time on instance (x, k) is bounded by $f(k) \cdot |x|^{\alpha}$, where f is an arbitrary function and α is a constant not depending on the parameter k. The class of fixed parameter tractable problems is denoted by FPT.

In order to characterize those problems that do not seem to admit a fixed parameter-efficient algorithm, Downey and Fellows defined a *parameterized reduction* and a hierarchy of classes $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots$ including likely fixed parameter intractable problems. Each W-class is the closure under parameterized reductions with respect to a kernel problem, which is usually formulated in terms of special mixed-type boolean circuits in which the number of input lines set to true is bounded by a function of the parameter.

E-mail address: cesati@uniroma2.it.

Several natural parameterized problems have been proved to be complete for the first two levels W[1] and W[2]. Although W[1]-complete problems are not fixed parameter tractable (unless W[1] = FPT, which is very unlikely), they appear to be easier than W[2]-complete problems. Essentially, candidate solutions for W[1]-complete problems (like Independent Set [7,10]) can be verified using constant-depth boolean circuits having just one level of gates with unbounded fan-in, while solutions for W[2]-complete problems (like Dominating Set [8,10]) require two levels of large gates. Many, but not all, parameterized problems can be verified by using general boolean circuits in which the number of input lines set to true is bounded by a function of the parameter: such problems belong to the class W[P].

In this paper we establish some membership results for the classes W[1], W[2], and W[P] through parameterized reductions to special versions of the Turing machine halting problem.

In general, it seems fruitful to think in terms of Turing machine computations when trying to establish membership in some W-class. In order to prove that a parameterized problem A belongs to a W-class, we always pick (more or less randomly) a problem B already in the class and we try to show that A reduces to B. However, it can be very difficult to devise such a reduction, because problems A and B can be very different; thus, even if the reduction exists, it may be very complicated. This is quite different from what we do in classical computational complexity, where usually we just show that a suitable model of Turing machine (for instance, a polynomial-time nondeterministic Turing machine) is able to decide an instance of our problem.

In our approach, in order to prove that problem A belongs to a W-class, we try to devise a reduction from A to some version of the Turing machine halting problem. In particular, in this paper we show that Perfect Code belongs to W[1], that Steiner Tree belongs to W[2], and that α-Balanced Separator, Maximal Irredundant Set, and Bounded DFA Intersection belong to W[P].

Apparently, the technique adopted in these proofs is the usual way to show membership in a W-class: a reduction from our candidate problem to a "target" problem already placed in the W-class. However, we recognize an important difference: the target problem is formulated in terms of a "general-purpose" Turing machine computation. For example, to show membership of Perfect Code we devise a single-tape nondeterministic Turing machine that guesses and verifies a candidate perfect code with a "short" computation. This is exactly what we would have done to show that a problem belongs to NP: devise a nondeterministic Turing machine that guesses and verifies a candidate solution in polynomial time.

The main advantage of the technique is that a Turing machine is a very general model of computation: it is such an opaque object that the reduction is mostly straightforward. One just has to design a suitable algorithm that solves the candidate problem without worrying about the inner, cumbersome details of the target problem. Another advantage is that it affords a natural way to arrive at certificates of size of the order of k^2 or k^3, when problems whose more natural certificates have size k may require ones of the larger size for their classification in the W-hierarchy.

All Turing machines discussed in this paper have just one head on each tape. The halting problem asks whether the Turing machine accepts in a number of steps bounded by a parameter. Nondeterministic Turing machines having a fixed number of tapes, or a number of tapes bounded by a parameter, are used to show membership in W[1]. The same halting problem for nondeterministic Turing machines having a unbounded number of tapes is used to show membership in W[2]. The halting problem for nondeterministic single-tape Turing machines where the number of steps is unbounded (yet specified in the instance, of course) and the number of nondeterministic choices performed by the machine is bounded by a parameter is used to show membership in W[P].

The paper is organized as follows. Section 2 introduces the necessary preliminaries. We define the general technique for the W[1] class in Section 3; in particular, we prove in Section 3.1 that Perfect Code belongs to W[1]. In Section 4 we introduce the multi-tape version of the Turing machine halting problem, and we prove that it is W[2]-complete. Using this problem as target of a suitable parameterized reduction we can show in Section 4.1 that Steiner Tree belongs to W[2]. In Section 5 we define the W[P]-complete version of the Turing machine halting problem. Using this, we can establish membership in W[P] for the problems α-Balanced Separator, Maximal Irredundant Set, and Bounded DFA Intersection in Sections 5.1–5.3. Finally, in Section 6 we draw the conclusions.

2. Parameterized computational complexity

A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a fixed alphabet. For convenience, we can always assume that $L \subseteq \Sigma^* \times \mathbb{N}$. We say that a parameterized problem is L (*uniformly*) *fixed parameter tractable* if it admits a solving algorithm whose running time on instance (x, k) is bounded by $f(k) \cdot |x|^\alpha$, where f is an arbitrary function and α is a constant not depending on the parameter k. The class of fixed parameter tractable problems is denoted by FPT.

A parameterized problem A is (*uniformly many*: 1) *reducible* to a parameterized problem B if there is an algorithm Φ which transforms an instance (x, k) of A into an instance (x', k') of B, and such that:

1. $(x, k) \in A$ if and only if $(x', k') \in B$;
2. k' depends only on k (i.e., there exists a function g such that $k' = g(k)$);
3. the running time of Φ is bounded by $f(k) \cdot |x|^\alpha$, for some arbitrary function f and constant α.

Let us consider now boolean circuits having two kinds of gates: *small gates* with bounded fan-in (usually, at most two), and *large gates* with unrestricted fan-in. The *depth* of a circuit C is defined to be the maximum number of small and large gates on an input–output path in C. The *weft* of C is the maximum number of large gates on an input–output path in C. The *weight* of a boolean vector x is the number of 1's in the vector. A *decision circuit* is a boolean circuit having just one output line. A decision circuit is said to *accept* every input vector that forces the value 1 on the output line.

Let F be a family of decision circuits, possibly having several circuits with a given number of input lines. Let $L_F = \{(C, k) : C \in F$ accepts an input vector of weight $k\}$. A parameterized problem L belongs to W[t] if there is a constant h such that L reduces to the parameterized problem $L_{F(t,h)}$, where $F(t, h)$ is the family of decision circuits of weft at most t and depth at most h. Thus, there is a hierarchy of W classes:

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

From these definitions, if a parameterized problem A reduces to B, and $B \in \mathrm{W[t]}$, then $A \in \mathrm{W[t]}$ as well.

Many parameterized problems have been proved to be complete for W[1] and W[2], that is, the first two levels of the W hierarchy. As an example, Independent Set is W[1]-complete [7]. Informally, W[1]-membership for Independent Set means that there exists an FPT algorithm that, given a graph G and a parameter k, produces a constant-depth decision circuit C having just one large gate on every input–output path, and such that C accepts a input vector x of weight k if and only if x encodes an independent set of G. W[1]-hardness implies that all parameterized problems in W[1] reduce to Independent Set; since there are many problems in W[1] for which no FPT algorithm is known, it is very unlikely that Independent Set is fixed parameter tractable.

The class W[P] contains the parameterized problems that reduce to L_C, where C is the family of decision circuits of any weft and depth. Thus, $\mathrm{W[t]} \subseteq \mathrm{W[P]}$, for every $t \geqslant 1$. While no W[t] class with $t \geqslant 3$ seems to be well populated, there are several W[P]-complete problems.

3. Membership in W[1]

In classical complexity theory, Cook's theorem states that the problem of deciding whether a CNF formula is satisfiable is NP-complete. The result is fundamental, because its proof consists of the simulation of a generic nondeterministic Turing machine computation by means of a CNF formula. In a sense, the theorem also provides strong evidence that $\mathrm{P} \neq \mathrm{NP}$, because a Turing machine is such a generic object that it does not seem reasonable that we should be able to predict its behavior without simulating all paths of the computation tree.

The analog of the Cook's theorem is a fundamental result of the parameterized complexity theory, because it gives evidence that W[1]-hard problems are likely not to be fixed parameter tractable. The theorem concerns the complexity of the following parameterized problem:

Short Nondeterministic Turing Machine Computation

Instance: A single-tape nondeterministic Turing machine M; a word x on the input alphabet of M; a positive integer k.

Parameter: k.

Question: Is there a computation of M on input x that reaches a final accepting state in at most k steps?

Theorem 1 (Analog of Cook's Theorem Cai et al. [4], Downey et al. [11]). Short Nondeterministic Turing Machine Computation *is* W[1]-*complete.*

There are several variations of the basic Short Nondeterministic Turing Machine Computation problem that remain W[1]-complete. For instance, the multi-tape version of the problem remains W[1]-complete if the number of tapes is bounded by a parameter [6], or even if it is unbounded but the transition table is full, that is, there is an applicable transition for every configuration of scanned symbols under the heads [6].

Thus, we can formulate the following suggestion:

Turing way to W[1]-membership: In order to show that a parameterized problem L belongs to W[1], devise a parameterized reduction from L to the Short Nondeterministic Turing Machine Computation problem (or to one of its W[1]-complete variations).

3.1. The Perfect Code problem

A *Perfect Code* in a graph $G = (V, E)$ is a subset of vertices V' such that for each vertex $v \in V$, the subset V' includes exactly one element among v and all vertices adjacent to v, that is, exactly one element in the *closed neighborhood* $N[v]$ of v. Formally:

Perfect Code
Instance: A graph $G = (V, E)$; a positive integer k.
Parameter: k.
Question: Does G have a k-element perfect code? A perfect code is a set of vertices $V' \subseteq V$ with the property that for each vertex $v \in V$ there is precisely one vertex in $N[v] \cap V'$.

Downey and Fellows proved that the Perfect Code problem is W[1]-hard by means of a reduction from Independent Set [8]. Although Perfect Code may be easily placed in W[2] (see [8]), till recently there was no evidence that it belongs to W[1]. Downey and Fellows conjectured that the problem could be of difficulty intermediate between W[1] and W[2], and thus not W[1]-complete ([10, pp. 277, 458]).

Eventually, however, the Perfect Code problem has been shown to belong to W[1] [5]. We report the proof here, because it represents a fundamental example of a W[1]-membership result established by means of the "Turing way."

Theorem 2. *The* Perfect Code *problem belongs to* W[1].

Proof. We devise a parameterized reduction from Perfect Code to Short Nondeterministic Turing Machine Computation.

Given a graph $G = (V, E)$ with n vertices and a positive integer k, let us construct a nondeterministic Turing machine $T = (\Sigma, Q, \Delta)$, where Σ includes the alphabet symbols

$$\Sigma = \{\square\} \cup \{\sigma_v : v \in V\} \cup \{s_i : i = 1, \dots, n\}$$

and Q contains the internal states

$$Q = \{q_A, q_R\} \cup \{q_i : i = 0, \dots, k\} \cup \{q_v^l, q_v^r : v \in V\} \cup \{q_j^s : j = 1, \dots, n\}$$

(Notice that both the alphabet size $|\Sigma|$ and the state set size $|Q|$ depend on n, hence the Turing machine behavior cannot be predicted with an FPT-algorithm unless W[1] = FPT [6].)

When the Turing machine starts, the internal state is q_0 and all tape cells contain the blank symbol ($\square$). The machine operates in three phases.

Phase 1: *guess k vertices.* The Turing machine nondeterministically chooses k vertices of G writing the corresponding symbols into the tape. This is achieved by including the following instructions in the transition table Δ:

$\langle \square, q_i, \sigma_v, q_{i+1}, +1 \rangle \in \Delta$
$(\forall i \in \{0, \dots, k-2\}, \forall v \in V)$ guess $k-1$ vertices moving to the right

$\langle \square, q_{k-1}, \sigma_v, q_k, 0 \rangle \in \Delta \quad (\forall v \in V)$ guess last vertex, do not move

(Each instruction specifies, in order, the symbol scanned under the head, the internal state of the machine, the new symbol written by the head, the new internal state, and the movement of the head: -1 for left, $+1$ for right, and 0 for no move.)

Phase 2: *check that the k vertices are "perfect"*. The Turing machine scans the guessed vertices and rejects as soon as it finds two vertices that violate one of the following conditions:

- For every pair of guessed vertices x and y, x and y are different.
- For every pair of guessed vertices x and y, x and y are not adjacent.
- For every pair of guessed vertices x and y, there is no vertex $z \in V$ that is adjacent to both x and y.

Moreover, in this phase each symbol σ_v is replaced by the symbol s_m, where m represents the size of the neighborhood $N[v]$ of v. This is achieved by including the following instructions in the transition table Δ:

$\langle \sigma_v, q_k, \sigma_v, q_v^l, -1 \rangle \in \Delta \quad (\forall v \in V)$ enter a loop for checking the vertex v under the head

$\langle \sigma_w, q_v^l, \sigma_w, q', -1 \rangle \in \Delta \quad (\forall v, w \in V)$ move to the left if the vertex w under the head satisfies the conditions with respect to v, reject otherwise

$$\text{where } q' = \begin{cases} q_R & \text{if } v = w \\ q_R & \text{if } (v,w) \in E \\ q_R & \text{if } \exists z \in V \text{ such that} \\ & (v,z) \in E \text{ and } (w,z) \in E \\ q_v^l & \text{otherwise.} \end{cases}$$

$\langle \square, q_v^l, \square, q_v^r, +1 \rangle \in \Delta \quad (\forall v \in V)$
$\langle \sigma_w, q_v^r, \sigma_w, q_w^r, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$ end of symbols, go back to the right up to vertex v

$\langle \sigma_v, q_v^r, s_m, q_k, -1 \rangle \in \Delta$
$(\forall v \in V)$, where $m = |N[v]|$ replace v with the symbol denoting the size of its closed neighborhood, then move to the left and enter the state q_k, thus restarting the loop with another vertex

$\langle \square, q_k, \square, q_k, +1 \rangle \in \Delta$ no more symbols to check, end of phase

Phase 3: *taking the sum*. The tape now contains exactly k symbols of $\{s_1, \dots, s_n\}$; each of them represents the neighborhood size of a guessed vertex. The Turing machine must accept if and only if the sum of all neighborhood sizes is equal to n. In fact, the checks in Phase 2 grant that no vertex in G belongs to the neighborhood of two different guessed vertices. In other words, the guessed vertices cover nonoverlapping subsets of V. Therefore, the sum cannot be greater than n; moreover, if the sum is equal to n, all vertices in G are dominated by the guessed k-element subset of V. The following instructions in Δ computes the sum:

$\langle s_i, q_k, s_i, q_i^s, +1 \rangle \in \Delta \quad (\forall i \in \{1, \dots, n\})$	initialize the internal state counter, and move to the right
$\langle s_j, q_i^s, s_j, q_t^s, +1 \rangle \in \Delta$ $(\forall i,j \in \{1, \dots, n\},\ i+j \leqslant n,\ t = i+j)$	add the size under the head to the internal state counter, and move to the right
$\langle \Box, q_n^s, \Box, q_A, 0 \rangle \in \Delta$ $\langle \Box, q_j^s, \Box, q_R, 0 \rangle \in \Delta \quad (\forall j \in \{1, \dots, n-1\})$	end of size symbols, accept if the internal state counter is equal to n, reject otherwise

It is straightforward to verify that the Turing machine T includes $(5/2)n^2 + (k + 7/2)n + 1$ instructions, that it can be derived in polynomial time in the size of G, and that it accepts in $k^2 + 4k + 2$ steps if and only if there exists a perfect code of size k in G. $\Box$

Downey and Fellows [8,10] showed that Perfect Code is equivalent to the Weighted Exact CNF Satisfiability problem, in which the instance is a boolean expression in conjunctive normal form, and the question is whether there exists a truth assignment of parameterized weight that makes exactly one literal in each clause true. An immediate consequence of Theorem 2 is that Weighted Exact CNF Satisfiability is W[1]-complete, hence it can be regarded as another analog of the Cook's theorem.

4. Membership in W[2]

Turing machines can be also useful in establishing membership results for the class W[2]. However, we need to identify a natural model of Turing machine whose parameterized computation problem is W[2]-complete.

In [6] it is shown that the natural variation of Short Nondeterministic Turing Machine Computation in which the machine has many read/write tapes is W[2]-hard. This is an interesting result by itself: in classical complexity theory almost all natural variations of the basic Turing machine model are equivalent, because they have NP-complete computation problems. Conversely, natural variations of the basic Turing machine model may have different parameterized complexities.

Formally, the multi-tape version of the problem is the following:

Short Multi-tape Nondeterministic Turing Machine Computation
Instance: A multi-tape nondeterministic Turing machine M; a word x on the input alphabet of M; a positive integer k.
Parameter: k.
Question: Is there a computation of M on input x that reaches a final accepting state in at most k steps?

We can now prove that this problem belongs to W[2]. The proof makes use of a characterization of the problems in W[2] established by Downey and Fellows [9]. Essentially they proved that the class W*[2], defined as W[2] but allowing decision circuits of parameterized depth instead of constant depth, coincides with W[2].

Theorem 3. *The* Short Multi-tape Nondeterministic Turing Machine Computation *problem belongs to* W[2].

Proof. Let $T = (\Sigma, Q, \Delta)$ be a nondeterministic Turing machine having m tapes, let $w \in \Sigma^*$ and let k be an integer. We shall show how to construct a circuit C with $k|\Delta| + 1$ input lines such that there exists a weight k input vector x such that $C(x) = 1$ if and only if there exists an accepting computation path of $T(w)$ having at most k steps.

Let us denote the input lines as $x[-1,-1]$ and $x[i,j]$ $(0 \leqslant i < k,\ 0 \leqslant j < |\Delta|)$. On any accepted input vector x, the first input line $x[-1,-1]$ is always 0. From now on, let us define $\underline{0}$ as $x[-1,-1]$, and $\underline{1}$ as $\neg x[-1,-1]$. The other input lines are partitioned in k blocks $x[i,\cdot]$; any block has $|\Delta|$ input lines and encodes the transition applied in the ith step of the computation path.

We shall consider several gates encoding various information about the computation path corresponding to x.

- $\tau_{\mathrm{o}}(i,l)$ $(0 \leqslant i < k,\ 0 \leqslant l < |Q|)$ identifies the "old" internal state of the ith transition of the computation path

$$\tau_{\mathrm{o}}(i,l) = \sum_{j \in J} x[i,j]$$

(where J is the subset of $\{0, \ldots, |\Delta| - 1\}$ containing the indices of the transitions whose old state has index l).
- $\tau_{\mathrm{n}}(i,l)$ $(0 \leqslant i < k,\ 0 \leqslant l < |Q|)$ identifies the "new" internal state of the ith transition of the computation path

$$\tau_{\mathrm{n}}(i,l) = \sum_{j \in J} x[i,j]$$

(where J is the subset of $\{0, \ldots |\Delta| - 1\}$ containing the indices of the transitions whose new state has index l).

- $\sigma_o(i,l,t)$ $(0 \leqslant i < k,\ 0 \leqslant l < |\Sigma|,\ 0 \leqslant t < m)$ identifies the "old" symbol scanned by the head on the tth tape of the ith transition of the computation path

$$\sigma_o(i,l,t) = \sum_{j \in J} x[i,j]$$

(where J is the subset of $\{0, \ldots |\Delta| - 1\}$ containing the indices of the transitions whose old symbol on tape t has index l).

- $\sigma_n(i,l,t)$ $(0 \leqslant i < k,\ 0 \leqslant l < |\Sigma|,\ 0 \leqslant t < m)$ identifies the "new" symbol scanned by the head on the tth tape of the ith transition of the computation path

$$\sigma_n(i,l,t) = \sum_{j \in J} x[i,j]$$

(where J is the subset of $\{0, \ldots, |\Delta| - 1\}$ containing the indices of the transitions whose new symbol on tape t has index l).

- $\mu(i,l,t)$ $(0 \leqslant i < k,\ 0 \leqslant t < m,\ l \in \{-1, 0, +1\})$ identifies the head move on the tth tape of the ith transition of the computation path

$$\mu(i,l,t) = \sum_{j \in J} x[i,j]$$

(where J is the subset of $\{0, \ldots, |\Delta| - 1\}$ containing the indices of the transitions whose head move on tape t is l).

- $\beta(i,l,t)$ $(0 \leqslant i < k,\ -k < l < k,\ 0 \leqslant t < m)$ encodes the head position on the tape t at the ith step (before applying the ith transition)

$$\beta(0,l,t) = \begin{cases} \underline{0} & \text{if } l \neq 0, \\ \underline{1} & \text{if } l = 0, \end{cases}$$

$$\begin{aligned} \beta(i,l,t) = {} & \beta(i-1,l,t) \cdot \mu(i-1,0,t) \\ & + \beta(i-1,l-1,t) \cdot \mu(i-1,+1,t) \\ & + \beta(i-1,l+1,t) \cdot \mu(i-1,-1,t) \end{aligned}$$

(where $i > 0$).

- $\sigma(i,l,s,t)$ $(0 \leqslant i < k,\ -k < l < k,\ 0 \leqslant s < |\Sigma|,\ 0 \leqslant t < m)$ encodes the content of the cells on tape t at step i:

$$\sigma(0,l,s,t) = \begin{cases} \underline{1} & \text{if } (t = 0) \wedge (0 \leqslant l < |w|) \wedge (s \text{ is the index of } w[l]), \\ \underline{1} & \text{if } (t = 0) \wedge (l < 0 \vee l \geqslant |w|) \wedge (s = 0), \\ \underline{0} & \text{otherwise} \end{cases}$$

(where the index of the blank symbol is 0, and $w[l]$ represents the lth character of the input word $w \in \Sigma^*$ on tape 0).

$$\begin{aligned} \sigma(i,l,s,t) = {} & \neg\beta(i-1,l,t) \cdot \sigma(i-1,l,s,t) \\ & + \beta(i-1,l,t) \cdot \sigma_n(i-1,l,t). \end{aligned}$$

The circuit C has a final *And* gate, corresponding to the following expression:

$$E = E_0 \cdot E_1 \cdot E_2 \cdot E_2 \cdot E_4.$$

The first term enforces the first input line to be always set to 0:

$$E_0 = \neg x[-1,-1].$$

The second term ensures that at most one input line in each block $x[i,\cdot]$ is set to 1:

$$E_1 = \prod_{i=0}^{k-1} \prod_{\substack{j,j'=0 \\ j \neq j'}}^{|\Delta|-1} (\neg x[i,j] + \neg x[i,j']).$$

The third term checks that the new internal state of the $(i-1)$th transition is equal to the old internal state of the ith transition:

$$E_2 = \prod_{i=1}^{k-1} \prod_{j=0}^{|Q|-1} (\neg \tau_{\mathrm{n}}(i-1,j) + \tau_{\mathrm{o}}(i,j)).$$

The fourth term checks that, for every transition and for every tape, the old symbol coincides with the symbol scanned under the corresponding head:

$$E_2 = \prod_{i=0}^{k-1} \prod_{t=0}^{m-1} \prod_{j=-k}^{k} \prod_{l=0}^{|\Sigma|-1} (\neg \beta(i,j,t) + \neg \sigma(i,j,l,t) + \sigma_{\mathrm{o}}(i,l,t)).$$

The fifth term ensures that the computation path starts from the internal state q_0 and ends with the accepting internal state q_A:

$$E_4 = \tau_{\mathrm{o}}(0,\bar{t}) \wedge \tau_{\mathrm{n}}(k-1,\bar{l}),$$

where $\bar{t}$ is the index corresponding to q_0 and $\bar{l}$ is the index corresponding to q_A.

It is easy to verify that the circuit C accepts a weight k input word x if and only if x encodes the k steps of a nondeterministic accepting computation path of $T(w)$, and that we can build the circuit C from an instance of Short Multi-tape Nondeterministic Turing Machine Computation with a parameterized reduction.

The above reduction shows that Short Multi-tape Nondeterministic Turing Machine Computation belongs to W*[2], which is the class of parameterized problems that reduce to a weft-2 circuit problem in which the circuit depth is bounded by a function of the parameter. In fact, every large gate having fan-in bounded by a function of the parameter can be replaced by a tree of small gates of depth bounded by a function of the parameter. The remaining large gates are either the sums whose input are the circuit's input lines, or the products in E_1, E_2, and E_2. Since Downey and Fellows [9] proved that $\mathrm{W}^*[2] = \mathrm{W}[2]$, the assertion follows. □

The previous theorem gives us another Turing machine model that can be used to establish membership results. We can thus formulate the following suggestion:

Turingway to W[2]-membership: In order to show that a parameterized problem L belongs to W[2], devise a parameterized reduction from L to the Short Multi-tape Nondeterministic Turing Machine Computation problem.

4.1. The Steiner Tree problem

As an example of application of the "Turing way" for W[2], consider the following problem:

Steiner Tree
Instance: A graph $G = (V, E)$; a set $S \subseteq V$; a positive integer k.
Parameter: k.
Question: Is there a set of vertices $T \subseteq V - S$ such that $|T| \leqslant k$ and $G[S \cup T]$ is connected?

This parameterized version of Steiner Tree is W[2]-hard [3]. (Notice however that the parameterized version in which k is unbounded and $|S|$ is a parameter is fixed-parameter tractable [12].)

Although no membership result for this problem was previously known, we can easily place it in W[2] by devising a multi-tape Turing machine that guesses a subset of vertices and checks whether it is a Steiner Tree:

Theorem 4. *The* Steiner Tree *problem belongs to* W[2].

Proof. We show a parameterized reduction from Steiner Tree to the Short Multi-tape Nondeterministic Turing Machine Computation problem on multi-tape machines. Since the latter problem is W[2]-complete, the reduction proves that Steiner Tree belongs to W[2].

Let us consider an instance $G = (V, E)$, $S \subseteq V$, $k \in \mathbb{N}$ of Steiner Tree. As first step of our reduction, let us replace every connected component of $G[S]$ with a single vertex. Formally, let us construct a new graph $G' = (V', E')$ where V' is obtained from V by replacing each connected component C of $G[S]$ with a vertex v_C, and

$$E' = \{(v, w) \in E : v, w \in V - S\} \cup \{(v, v_C) : \exists z \in C \text{ such that } (v, z) \in E\}.$$

Moreover, let us replace S with the subset S' of all new vertices v_C.

It is quite easy to verify that $[G = (V, E), S, k]$ is a yes-instance of Steiner Tree if and only if $[G' = (V', E'), S', k]$ is a yes-instance of Steiner Tree. In fact, $T \in V - S$, $|T| \leqslant k$, is a solution to the former problem if and only if $G[S \cup T]$ is connected, that is, if and only if $G'[S' \cup T]$ is connected, that is, if and only if $T \subseteq V' - S'$ is a solution to the latter problem. Since connected components can be determined in polynomial time in the size of the graph, this is a parameterized reduction.

The second step of our reduction consists of constructing a $(m + 1)$-tape nondeterministic Turing machine $M = (\Sigma, Q, \Delta)$, where $m = |S'|$,

$$\Sigma = \{\square, \#\} \cup \{v, v', v'' : v \in V' - S'\}$$

and

$$Q = \{q_A, q_R, q^a, q^b, q^e, q^f, q^g\} \cup \{q_i : 0 \leqslant i \leqslant k - 1\} \cup \{q_v^c, q_v^d : v \in V' - S'\}.$$

We fix an ordering for the subset S', and we associate each element of S' to a tape of M, starting from the second tape. The first tape will contain the elements of the subset T. Initially, all tapes contain blanks.

The Turing machine M accepts in $\mathcal{O}(k^2)$ steps if and only if $[G' = (V', E'), S', m]$ is a yes-instance. It operates in three phases: it guesses at most k vertices of $V' - S'$ representing the subset T; then, it checks that T "covers" S', that is, that each vertex of S' is adjacent to some vertex of T; finally, it checks that all vertices of T are connected together, optionally by means of vertices in S'. It is easy to verify that all checks are satisfied if and only if $G'[S' \cup T]$ is connected, that is, if and only if T is a solution of the problem.

Phase 1: *guess k vertices of G'*. The Turing machine nondeterministically chooses k vertices of G' writing the corresponding symbols into the tape. This is achieved by including the following instructions in the transition table Δ:

$$\left\langle \underbrace{\square, \dots, \square}_{m+1}, q_i, v, \underbrace{\square, \dots, \square}_{m}, q_{i+1}, +1, \underbrace{0, \dots, 0}_{m} \right\rangle \in \Delta$$

$$(\forall i \in \{0, \dots, k-2\},\ \forall v \in V' - S')$$

guess $k-1$ vertices and write them on the first tape, moving the corresponding head to the right; the heads on the other tapes do nothing

$$\left\langle \underbrace{\square, \dots, \square}_{m+1}, q_{k-1}, v, \underbrace{\square, \dots, \square}_{m}, q^a, \underbrace{0, \dots, 0}_{m+1} \right\rangle \in \Delta$$

$$(\forall v \in V' - S')$$

guess the last vertex, write it on the first tape, and leave the head on it

(Each instruction specifies, in order, the symbols scanned under the $m+1$ heads, the internal state of the machine, the $m+1$ new symbols written by the machine, the new internal state, and the movements of the $m+1$ heads: -1 for left, $+1$ for right, and 0 for no move.)

Notice that there could be some duplicated vertices, thus the subset T may have less than k vertices.

Phase 2: *check that T covers S'*. The Turing machine checks that every vertex in S' is adjacent to some vertex in T. Let us assume that $S' = \{s_1, \dots, s_m\}$. For each guessed vertex v, the machine writes a # symbol on every tape corresponding to a vertex in S' that is adjacent to v. It must do this with a running time depending only by $|T|$, and not by $|S'|$. Hence, the machine has to write several # symbols in the same step, exploiting the parallelism inherent to the multi-tape model of Turing machine.

We also need a "information hiding" trick: every time the machine writes a # symbol, it moves the writing head, so that in the next step the head will read a blank ($\square$). The trick is required in order to keep the transition table small: basically, we cannot insert in the transition table 2^m different instructions to take into account all different configurations of $\square$ and # symbols on the m tapes.

When all vertices in T have been considered, the Turing machine moves backward all heads on the m tapes corresponding to the element of S', and checks that all heads read a # symbol. If there is a head reading a blank, the machine hangs up. Observe that S' is an independent set in G', thus if some vertex of S' is not adjacent to any vertex of T, then $G'[S' \cup T]$ cannot be connected.

$$\left\langle v, \underbrace{\Box, \ldots, \Box}_{m}, q^a, v, \sigma_1, \ldots, \sigma_m, q^a, -1, \mu_1, \ldots, \mu_m \right\rangle \in \Delta$$

consider the vertex v under the head of the first tape; in parallel, write a # symbol on each tape corresponding to a vertex in S' that is adjacent to v; move all writing heads to the right, and the head on the first tape to the left

$$(\forall v \in V' - S'), \text{ where } \sigma_j = \begin{cases} \# & \text{if } s_j \in N[v] \\ \Box & \text{if } s_j \notin N[v] \end{cases}$$

$$\text{and } \mu_j = \begin{cases} +1 & \text{if } s_j \in N[v] \\ 0 & \text{if } s_j \notin N[v] \end{cases}$$

$$\left\langle \underbrace{\Box, \ldots, \Box}_{m+1}, q^a, \underbrace{\Box, \ldots, \Box}_{m+1}, q^a, 0, \underbrace{-1, \ldots, -1}_{m} \right\rangle \in \Delta$$

no more elements of T to consider: move to the left all heads on the tapes corresponding to the elements of S'

$$\left\langle \Box, \underbrace{\#, \ldots, \#}_{m}, q^a, \underbrace{\Box, \ldots, \Box}_{m+1}, q^b, +1, \underbrace{0, \ldots, 0}_{m} \right\rangle \in \Delta$$

proceed with the next phase only if all heads on the tapes corresponding to the elements of S' read a # symbol

Phase 3: *check that T is connected by means of S'*. The Turing machine checks whether all vertices of T are connected together, optionally using vertices of S'. Recall that S' is an independent set in G', so the Turing machine doesn't have to search for paths inside $G'[S']$.

The machine iteratively enlarges the set of vertices that are reachable from the first guessed vertex of T. It uses three different sets of symbols: $\{v : v \in V' - S'\}$ is used to identify the elements of T that have not been reached, $\{v' : v \in V' - S'\}$ is used to identify the elements that have been reached but not yet analyzed, and $\{v'' : v \in V' - S'\}$ is used to identify the elements that have been reached and analyzed.

Internal states q_v^c and q_v^d are used to search for the vertices reachable from v: the former state moves the head onto the first guessed vertex, and the latter state scans all vertices on the tape. Internal states q^e and q^f move the head toward the first guessed vertex before considering another reachable vertex not yet analyzed; the state q^f indicates that at least one unreached vertex exists. Finally, internal state q^g looks for a reachable vertex that has not been analyzed.

From now on, the Turing machine uses the first tape only. To save space, in the following instructions we shall omit the symbols under the heads of the other tapes (always $\Box$'s), as well as their moves (always 0's).

$$\langle v, q^b, v'', q_v^c, -1 \rangle \in \Delta \quad (\forall v \in V' - S')$$

mark the rightmost vertex v as being analyzed and record it in the internal state

$$\langle \sigma, q_v^c, \sigma, q_v^c, -1 \rangle \in \Delta$$
$$(\forall v \in V' - S', \forall \sigma \in \{w, w', w'' : w \in V' - S'\})$$

go to the leftmost vertex before searching for vertices reachable from v

$\langle \square, q^c_v, \square, q^d_v, +1 \rangle \in \Delta \quad (\forall v \in V' - S')$	head at left end, go to the right and mark the vertices reachable from v
$\langle w, q^d_v, w', q^d_v, +1 \rangle \in \Delta \quad (\forall v \in V' - S', \forall w \in N[v])$	mark w as reached if it is adjacent to v
$\langle w, q^d_v, w', q^d_v, +1 \rangle \in \Delta$ $(\forall v, w \in V' - S', w \notin N[v], N[v] \cap N[w] \cap S' \neq \emptyset)$	mark w as reached if there exists a vertex in S' that is adjacent to both w and v
$\langle w, q^d_v, w, q^d_v, +1 \rangle \in \Delta$ $(\forall v, w \in V' - S', w \notin N[v], N[v] \cap N[w] \cap S' = \emptyset)$	if w is neither adjacent to v nor adjacent to a vertex in S' that is adjacent to v, leave it unreached
$\langle \sigma, q^d_v, \sigma, q^d_v, +1, \rangle \in \Delta$ $(\forall v \in V' - S', \forall \sigma \in \{w', w'' : w \in V' - S'\})$	skip over already reached vertices
$\langle \square, q^d_v, \square, q^e, -1 \rangle \in \Delta \quad (\forall v \in V' - S')$	head at right end, go back to left end
$\langle \sigma, q^e, \sigma, q^e, -1, \rangle \in \Delta$ $(\forall \sigma \in \{v', v'' : v \in V' - S'\})$	keep moving to the left end (the internal state indicates that till now no unreached vertex has been seen)
$\langle v, q^e, v, q^f, -1 \rangle \in \Delta \quad (\forall v \in V' - S')$	record that, while going back to left end, a unreached vertex has been seen
$\langle \sigma, q^f, \sigma, q^f, -1 \rangle \in \Delta$ $(\forall \sigma \in \backslash\{\mathrm{v,v',v''} : \mathrm{v} \in \mathrm{V'} - \mathrm{S'}\backslash\})$	keep moving to the left end (the internal state indicates that at least one unreached vertex still exists)
$\langle \square, q^e, \square, q_A, 0 \rangle \in \Delta$	head at left end, all vertices have been marked as reached, hence accept
$\langle \square, q^f, \square, q^g, +1 \rangle \in \Delta$	head at left end, at least one unreached vertex exists: start searching for a reached vertex that has not yet been analyzed
$\langle \sigma, q^g, \sigma, q^g, +1 \rangle \in \Delta$ $(\forall \sigma \in \{v, v'' : v \in V' - S'\})$	skip over unreached vertices and reached vertices that have already been analyzed
$\langle v', q^g, v'', q^c_v, -1 \rangle \in \Delta \quad (\forall v \in V' - S')$	v is a reached vertex still to be analyzed; mark v as analyzed, and go to the leftmost vertex
$\langle \square, q^g, \square, q_R, 0 \rangle \in \Delta$	no reached vertex still to be analyzed has been found: reject, because a unreached vertex exists

It is straightforward to verify that both phases 1 and 2 end after exactly k steps. Moreover, it easy to see that phase 3 ends after at most $\mathcal{O}(k^2)$ steps. Thus, $[M, k']$, $k' = \mathcal{O}(k^2)$, is a yes-instance of Short Multi-tape Nondeterministic Turing Machine Computation if and only if $[G' = (V', E'), S', k]$ is a yes-instance of Steiner Tree. $\square$

5. Membership in W[P]

Finally, Turing machines can be useful in establishing membership results for the class W[P]. Again, we need to identify a model of Turing machine whose parameterized computation problem is W[P]-complete.

The class W[P] is quite large; it includes all parameterized problems that reduce to L_F, where F is the family of decision circuits with any depth and weft. The depth of the decision circuit roughly corresponds to the number of steps of a Turing machine, while the Hamming weight of the input corresponds to the number of nondeterministic steps of the Turing machine. Thus, a natural version of the Turing machine computation problem is the following:

Bounded Nondeterminism Turing Machine Computation
Instance: A nondeterministic Turing machine T; an input word w; an integer n (encoded in unary); a positive k.
Parameter: k.
Question: Does $T(w)$ nondeterministically accept in at most n steps and using at most k nondeterministic steps?

We now prove that the Bounded Nondeterminism Turing Machine Computation problem is W[P]-complete.

Theorem 5. Bounded Nondeterminism Turing Machine Computation *belongs to* W[P].

Proof. Let $T = (\Sigma, Q, \Delta)$ be a nondeterministic Turing machine, let $w \in \Sigma^*$ and let n, k be integers. We shall show how to construct a circuit C with $n|\Delta| + 1$ input lines such that there exists a weight k input vector x such that $C(x) = 1$ if and only if there exists an accepting computation path of $T(w)$ having at most n steps and at most k nondeterministic steps.

Let us denote the input lines as $x[-1, -1]$ and $x[i, j]$ $(0 \leqslant i < n,\ 0 \leqslant j < |\Delta|)$. On any accepted input vector x, the first input line $x[-1, -1]$ is always 0. From now on, let us define $\underline{0}$ as $x[-1, -1]$, and $\underline{1}$ as $\neg x[-1, -1]$. The other input lines are partitioned in n blocks $x[i, \cdot]$; any block has $|\Delta|$ input lines and encodes a *nondeterministic* transition applied in the ith step of the computation path. If the ith step of the computation path is deterministic, the input lines $x[i, \cdot]$ are all 0's.

We shall consider several gates encoding various information about the computation path corresponding to x:

- $\gamma(i)$ $(0 \leqslant i < n)$ indicates whether the ith step is nondeterministic

$$\gamma(i) = \sum_{j=0}^{|\Delta|-1} x[i, j].$$

- $\tau(i,l)$ $(0 \leqslant i < n,\ 0 \leqslant l < |Q|)$ encodes the internal state at step i:

  $$\tau(0,l) = \begin{cases} \underline{0} & \text{if } l \neq 0, \\ \underline{1} & \text{if } l = 0 \end{cases}$$

  (where the index of the initial state is 0)

  $$\tau(i,l) = \sum_{j \in J} (x[i-1,j] + (\neg\gamma(i-1) \cdot \alpha(i-1,j)))$$

  (where $i > 0$ and J is the subset of $\{0, \dots |\Delta| - 1\}$ corresponding to the indices of the transitions whose new state has index l).
- $\alpha(i,l)$ $(0 \leqslant i < n,\ 0 \leqslant l < |\Delta|)$ indicates whether the lth transition is applicable at the ith step:

  $$\alpha(i,l) = \tau(i,\bar{q}) \cdot \sum_{j=0}^{n-1} (\beta(i,j) \cdot \sigma(i,j,\bar{s}))$$

  (where $\bar{q}$ is the index of the old state of the lth transition, and $\bar{s}$ is the index of the old symbol of the lth transition).
- $\beta(i,l)$ $(0 \leqslant i < n,\ -n < l < n)$ encodes the head position on the tape

  $$\beta(0,l) = \begin{cases} \underline{0} & \text{if } l \neq 0, \\ \underline{1} & \text{if } l = 0, \end{cases}$$

  $$\begin{aligned} \beta(i,l) = & \beta(i-1,l) \cdot \sum_{j \in J^0} (x[i-1,j] + (\neg\gamma(i-1) \cdot \alpha(i-1,j))) \\ & + \beta(i-1,l-1) \cdot \sum_{j \in J^+} (x[i-1,j] + (\neg\gamma(i-1) \cdot \alpha(i-1,j))) \\ & + \beta(i-1,l+1) \cdot \sum_{j \in J^-} (x[i-1,j] + (\neg\gamma(i-1) \cdot \alpha(i-1,j))) \end{aligned}$$

  (where $i > 0$ and J^0, J^+, and J^- are the subsets of $\{0, \dots, |\Delta| - 1\}$ corresponding to the indices of the transitions whose head moves are, respectively, 0, +1, and −1).
- $\sigma(i,l,s)$ $(0 \leqslant i < n,\ -n < l < n,\ 0 \leqslant s < |\Sigma|)$ encodes the content of the tape cells at step i:

  $$\sigma(0,l,s) = \begin{cases} \underline{1} & \text{if } (0 \leqslant l < |w|) \wedge (s \text{ is the index of } w[l]), \\ \underline{0} & \text{if } (0 \leqslant l < |w|) \wedge (s \text{ is not the index of } w[l]), \\ \underline{1} & \text{if } (l < 0 \vee l \geqslant |w|) \wedge (s = 0), \\ \underline{0} & \text{if } (l < 0 \vee l \geqslant |w|) \wedge (s \neq 0) \end{cases}$$

  (where the index of the blank symbol is 0, and $w[l]$ represents the lth character of the input word $w \in \Sigma^*$).

  $$\begin{aligned} \sigma(i,l,s) = & \neg\beta(i-1,l) \cdot \sigma(i-1,l,s) \\ & + \beta(i-1,l) \cdot \sum_{j \in J} (x[i-1,j] + (\neg\gamma(i-1) \cdot \alpha(i-1,j))) \end{aligned}$$

(where J is the subset of $\{0, \ldots, |\Delta| - 1\}$ corresponding to the indices of the transitions whose new symbol has index s).

The circuit C has a final *And* gate, corresponding to the following expression:

$$E = E_0 \cdot E_1 \cdot E_2 \cdot E_2 \cdot E_4.$$

The first term enforces the first input line to be always set to 0:

$$E_0 = \neg x[-1, -1].$$

The second term ensures that at most one input line in each block $x[i, \cdot]$ is set to 1:

$$E_1 = \prod_{i=0}^{n-1} \prod_{\substack{j,j'=0 \\ j \neq j'}}^{|\Delta|-1} (\neg x[i,j] + \neg x[i,j']).$$

The third term checks that any instruction specified by the input lines can be legally applied:

$$E_2 = \prod_{i=0}^{n-1} \prod_{j=0}^{|\Delta|-1} (\neg x[i,j] + \alpha(i,j)).$$

The fourth term forces a value 1 in any input line block corresponding to a nondeterministic step:

$$E_2 = \prod_{i=0}^{n-1} \prod_{\substack{j,j'=0 \\ j \neq j'}}^{|\Delta|-1} (\neg\gamma(i) + \neg\alpha(i,j) + \neg\alpha(i,j')).$$

The fifth term ensures that the computation path is an accepting one:

$$E_4 = \tau(n, \bar{q}),$$

where $\bar{q}$ is the index corresponding to the accepting state q_A.

It is easy to verify that the circuit C accepts a weight k input word x if and only if x encodes the k nondeterministic steps of a nondeterministic accepting computation path of $T(w)$. In particular, the term E_2 ensures that every block of input lines $x[i, \cdot]$ having no line set to 1 corresponds to a deterministic step; the circuit can thus derive the transition to be applied from the internal state and the symbol under the head of the Turing machine.

We can build the circuit C from an instance of bounded nondeterminism Turing machine computation with a parameterized reduction; notice, however, that we can do this only because the length of the computation path in the problem is encoded in unary. □

Theorem 6. Bounded Nondeterminism Turing Machine Computation *is* W[P]-*hard.*

Proof. We show a parameterized reduction from the Chain Reaction Closure problem [1] to Bounded Nondeterminism Turing Machine Computation. Chain Reaction Closure is W[P]-complete [1], and it is defined as follows:

Chain Reaction Closure

Instance: A directed graph $D = (V, A)$; a positive integer k.

Parameter: k.

Question: Does there exist a set V' of k vertices of D whose chain reaction closure is V? (A chain reaction closure of V' is the smallest superset S of V' such that if $u, u' \in S$ and arcs ux, ux' are in A then $x \in S$.)

Given a directed graph $D = (V, A)$ with n vertices and a positive integer k, let us construct a nondeterministic Turing machine $T = (\Sigma, Q, \Delta)$, where Σ includes the alphabet symbols

$$\Sigma = \{\Box\} \cup \{v : v \in V\}$$

and Q contains the internal states

$$Q = \{q_A, q_R, q^a, q^h\} \cup \{q_i : 0 \leqslant i \leqslant k-1\}$$
$$\cup \{q_v^b, q_v^c, q_v^d, q_v^e, q_v^f, q_v^g : v \in V\} \cup \{q_i^l : 0 \leqslant i \leqslant n-1\}.$$

When the Turing machine starts, the internal state is q_0 and all tape cells contain the blank symbol ($\Box$). The machine guesses k different vertices that represent V'. It then generates the chain reaction closure of V', and computes its size.

Phase 1: *guess k different vertices.* Assume a fixed ordering $v_0, v_1, \ldots, v_{n-1}$ of the vertices of D. The Turing machine nondeterministically chooses k different vertices of D writing the corresponding symbols onto the tape. This is achieved by including the following instructions in the transition table Δ:

$\langle \Box, q_i, v, q_{i+1}, +1 \rangle \in \Delta$ $(\forall v \in V, \forall i \in \{0, \ldots, k-2\})$	guess k vertices, leaving the head on the rightmost one
$\langle \Box, q_{k-1}, v, q^a, 0 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle v, q^a, v, q_v^b, -1 \rangle \in \Delta \quad (\forall v \in V)$	go to the left end checking whether v has been guessed several times
$\langle w, q_v^b, w, q_v^b, -1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$	skip over vertices different than v
$\langle v, q_v^b, v, q_R, 0 \rangle \in \Delta \quad (\forall v \in V)$	v has been guessed at least twice, hence reject
$\langle \Box, q_v^b, \Box, q_v^c, +1 \rangle \in \Delta \quad (\forall v \in V)$	head at left end, go back to the vertex v
$\langle w, q_v^c, w, q_v^c, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$	skip over vertices different than v
$\langle v, q_v^c, v, q^a, -1 \rangle \in \Delta \quad (\forall v \in V)$	head on vertex v, restart checking the vertex at the left of v
$\langle \Box, q^a, \Box, q_{v_0}^d, +1 \rangle \in \Delta$	no more vertices to check, continue with the next phase

Phase 2: *compute the chain reaction closure.* The Turing machine considers in turn each vertex $x \in V$ and scans the symbols on the tape. If it finds two vertices u and u' on tape such that $ux \in A$ and $u'x \in A$, then it writes x on tape (if x is not already there) and restarts considering the first vertex v_0 of D. The phase ends when the Turing machine considers the last vertex v_{n-1} of D and detects that it cannot be written on the tape. Internal states q_v^d, q_v^e, and q_v^g are used to check whether v can be written on the tape; they indicate respectively that no witness, one witness u, and two witnesses u and u' have been found. Internal state q_v^f indicates that v cannot be written on the tape and that the vertex following v in the fixed ordering must be considered. Internal state q^h is used to restart, from the beginning, the loop over the vertices of D after a symbol has been written on tape.

$\langle w, q_v^d, w, q_v^e, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w, wv \in A)$	w is the first witness for v, move to the right to search for the second witness
$\langle w, q_v^e, w, q_v^g, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w, wv \in A)$	w is the second witness for v, move to the right to verify that v is not already on tape
$\langle w, q_v^d, w, q_v^d, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w, wv \notin A)$	w is not a witness for v, move to the right to search for a witness
$\langle w, q_v^e, w, q_v^e, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w, wv \notin A)$	
$\langle v, q_v^d, v, q_v^f, -1 \rangle \in \Delta \quad (\forall v \in V)$	v is already on the tape, consider the next vertex in V
$\langle v, q_v^e, v, q_v^f, -1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle v, q_v^g, v, q_v^f, -1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle w, q_v^g, w, q_v^g, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$	$w \neq v$, hence move to the right (v already has two witnesses, thus there is no need to check whether $wv \in A$)
$\langle \square, q_v^d, \square, q_v^f, -1 \rangle \in \Delta \quad (\forall v \in V)$	head at right end: since either no witness or just one witness for v has been found, consider the next vertex in D
$\langle \square, q_v^e, \square, q_v^f, -1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle \square, q_v^g, v, q^h, -1 \rangle \in \Delta \quad (\forall v \in V)$	head at right end: v has two witnesses, hence write v onto the tape, and go to the left end to restart from the beginning
$\langle v, q^h, v, q^h, -1 \rangle \in \Delta \quad (\forall v \in V)$	keep moving to the left until the left end is reached

$\langle \square, q^h, \square, q^d_{v_0}, +1 \rangle \in \Delta$	head at left end: restart from the beginning (search for witnesses relative to the first vertex v_0)
$\langle w, q^f_v, w, q^f_v, -1 \rangle \in \Delta \quad (\forall v, w \in V)$	keep moving to the left until the left end is reached (the internal state records the vertex v just analyzed)
$\langle \square, q^f_{v_i}, \square, q^d_{v_{i+1}}, +1 \rangle \in \Delta \quad (\forall i \in \{0, \dots, n-2\})$	head at left end, and vertex v_i has just been analyzed: move to the right and start considering vertex v_{i+1}
$\langle \square, q^f_{v_{n-1}}, \square, q^l_0, +1 \rangle \in \Delta$	head at left end, and the last vertex v_{n-1} has just been analyzed: continue with the next phase

Phase 3: *compute the size of the chain reaction closure.* The Turing machine simply counts the number of symbols written on tape, and accepts if the total is n; the internal state acts as a counter.

$\langle v, q^l_i, v, q^l_{i+1}, +1 \rangle \in \Delta$ $(\forall v \in V,\ \forall i \in \{0, \dots, n-2\})$	keep moving to the right counting the number of vertices written on the tape
$\langle v, q^l_{n-1}, v, q_A, +1 \rangle \in \Delta \quad (\forall v \in V)$	the counter is equal to $n-1$, and another vertex has been found: accept
$\langle \square, q^l_i, \square, q_R, 0 \rangle \in \Delta \quad (\forall i \in \{0, \dots, n-1\})$	head at right end, and the counter is smaller than n: reject

It is straightforward to verify that the transition table Δ includes $\mathcal{O}(n^2)$ instructions and that it can be derived by simply looking at the directed graph D. The Turing machine accepts if and only if it can nondeterministically guess a subset of k different vertices whose chain reaction closure is as large as D. Moreover, it either accepts or rejects in at most $k + 2kn + 2(k+1)n + \cdots + 2n^2 = \mathcal{O}(n^3)$ steps. $\square$

The previous theorems enable us to formulate the following suggestion:

Turing way to W[P]-membership In order to show that a parameterized problem L belongs to W[P], devise a parameterized reduction from L to the Bounded Nondeterminism Turing Machine Computation problem.

5.1. The α-Balanced Separator problem

As a first example of W[P]-membership result by means of a reduction to a Turing machine computation problem, consider the following problem:

α-Balanced Separator
Instance: A graph $G = (V, E)$; a positive integer k.
Parameter: k.
Question: Does there exist a subset S of at most k vertices such that each connected component of $G[V - S]$ has at most $\alpha|V|$ vertices?

This parameterized problem is W[1]-hard (reduction from [3]); as far as we know, no membership result was previously known.

Theorem 7. *The* α-Balanced Separator *problem belongs to* W[P].

Proof. We show a parameterized reduction from α-Balanced Separator to the Bounded Nondeterminism Turing Machine Computation problem. Since the latter problem is W[P]-complete, the reduction proves that α-Balanced Separator belongs to W[P].

Given a graph $G = (V, E)$ with n vertices and a positive integer k, let us construct a nondeterministic Turing machine $T = (\Sigma, Q, \Delta)$, where Σ includes the alphabet symbols

$$\Sigma = \{\square, \$, \times\} \cup \{v, v' : v \in V\}$$

and Q contains the internal states

$$\begin{aligned} Q = &\{q_A, q_R, q^a, q^e, q^g, q^h, q^m\} \cup \{q_i : 0 \leqslant i \leqslant k\} \\ &\cup \{q_v^b, q_v^c, q_v^d, q_v^f, q_v^h, q_v^l, q_v^s : v \in V\} \\ &\cup \{q_i^{e,v} : v \in V, 0 \leqslant i \leqslant \lfloor \alpha n \rfloor\} \cup \{q_v^{f,i} : v \in V, 0 \leqslant i \leqslant n\}. \end{aligned}$$

When the Turing machine starts, the internal state is q_0 and all tape cells contain the blank symbol ($\square$). The machine guesses at most k vertices of G that represents the subset S; next, it computes, for each vertex $v \in V$, the connected component of $G[V - S]$ including v and checks that its size doesn't exceed $\alpha|V|$.

Phase 1: *guess k vertices.* The Turing machine nondeterministically chooses k vertices of G (the subset S) writing the corresponding symbols onto the tape (since there could be some duplicated vertices, S may have less than k vertices). This is achieved by including the following instructions in the transition table Δ:

$\langle \square, q_i, v, q_{i+1}, +1 \rangle \in \Delta$ $(\forall i \in \{0, \ldots, k-1\}, \forall v \in V)$	guess k vertices and leave the head at the right end
$\langle \square, q_k, \$, q^a, +1 \rangle \in \Delta$	mark the right end of the guessed vertices with a \$ symbol, then move to the right and continue with the next phase

Phase 2: *generate the connected components.* For each vertex $v \in V - S$, the Turing machine computes the connected component of $G[V - S]$ that includes v. In the following, assume a fixed ordering $v_0, v_1, \ldots, v_{n-1}$ of the vertices of G. Internal state q^a controls the loop over all vertices of V, from v_0 to v_{n-1}.

$\langle \Box, q^a, v_0, q^b_{v_0}, -1 \rangle \in \Delta$	start the loop with the first vertex v_0
$\langle v_i, q^a, v_{i+1}, q^b_{v_{i+1}}, -1 \rangle \in \Delta \quad (\forall i \in \{0, \ldots, n-2\})$	work on v_i is finished, hence start the next iteration of the loop on v_{i+1}
$\langle v_{n-1}, q^a, v_{n-1}, q_A, 0 \rangle \in \Delta$	work on the last vertex v_{n-1} is finished, hence accept

Phase 2*a*: *check that* $v \notin S$. Internal states q^b_v scan the guessed vertices at the left of the \$ symbol; internal states q^c_v and q^d_v denote, respectively, that $v \in S$ and $v \notin S$.

$\langle \$, q^b_v, \$, q^b_v, -1 \rangle \in \Delta \quad (\forall v \in V)$	move to the left and skip the \$ symbol
$\langle \times, q^b_v, \times, q^b_v, -1 \rangle \in \Delta \quad (\forall v \in V)$	move to the left and skip the $\times$ symbol
$\langle u, q^b_v, u, q^b_v, -1 \rangle \in \Delta \quad (\forall u, v \in V, u \neq v)$	guessed vertex u is not equal to v, all right
$\langle v, q^b_v, v, q^c_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	found v at the left of the \$ symbol: record that $v \in S$ and go back to the \$ symbol
$\langle \Box, q^b_v, \Box, q^d_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	head at left end: record that $v \notin S$ and go back to the \$ symbol
$\langle u, q^c_v, u, q^c_v, +1 \rangle \in \Delta \quad (\forall u, v \in V, u \neq v)$	
$\langle u, q^d_v, u, q^d_v, +1 \rangle \in \Delta \quad (\forall u, v \in V, u \neq v)$	keep moving to the right until the \$ symbol is found
$\langle \times, q^c_v, \times, q^c_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle \times, q^d_v, \times, q^d_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	while moving to the right, skip the $\times$ symbols
$\langle \$, q^c_v, \$, q^a, +1 \rangle \in \Delta \quad (\forall v \in V)$	move to the right and start a new iteration of the external loop, because $v \in S$
$\langle \$, q^d_v, \$, q^e, +1 \rangle \in \Delta \quad (\forall v \in V)$	move to the right and continue with Phase 2b, because $v \notin S$

Phase 2*b*: *generate the connected component including* v. For each vertex w at the right of the \$ symbol, write onto the tape the vertices in $N[w] - S$. We can further distinguish three sub-phases:

Phase $2b(i)$: *generate the neighborhood of* w.

$\langle w, q^e, w', q^f_w, +1 \rangle \in \Delta \quad (\forall w \in V)$	mark w as being expanded, and go to the right end
$\langle u, q^f_w, u, q^f_w, +1 \rangle \in \Delta \quad (\forall u, w \in V)$	keep moving to the right until the right end is reached
$\langle \times, q^f_w, \times, q^f_w, +1 \rangle \in \Delta \quad (\forall w \in V)$	move to the right and skip the $\times$ symbol
$\langle \square, q^f_w, u_0, q^{f,1}_w, +1 \rangle \in \Delta \quad (\forall w \in V)$	write onto the tape the vertices adjacent to w; assume that they are $\{u_0, u_1, \ldots, u_{m_w}\}$; then, continue with Phase 2b(ii)
$\langle \square, q^{f,i}_w, u_i, q^{f,i+1}_w, +1 \rangle \in \Delta$ $(\forall w \in V, \forall i \in \{1, \ldots, m_w - 1\})$	
$\langle \square, q^{f,m_w}_w, u_{m_w}, q^g, -1 \rangle \in \Delta \quad (\forall w \in V)$	

Phase $2b(ii)$: *cancel the duplicated vertices.*

$\langle w, q^g, w, q^g, -1 \rangle \in \Delta \quad (\forall w \in V)$	keep moving to the left until the left end is reached
$\langle w', q^g, w', q^g, -1 \rangle \in \Delta \quad (\forall w \in V)$	
$\langle \times, q^g, \times, q^g, -1 \rangle \in \Delta$	while moving to the left, skip over any \$ and $\times$ symbol
$\langle \$, q^g, \$, q^g, -1 \rangle \in \Delta$	
$\langle \square, q^g, \square, q^h, +1 \rangle \in \Delta$	head at left end: start deleting the duplicated vertices
$\langle v, q^h, v, q^h_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	start scanning the vertices at the right of v to find duplicates of v
$\langle v', q^h, v', q^h_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle u, q^h_v, u, q^h_v, +1 \rangle \in \Delta \quad (\forall u, v \in V, u \neq v)$	u is different than v, continue
$\langle u', q^h_v, u', q^h_v, +1 \rangle \in \Delta \quad (\forall u, v \in V,\ u \neq v)$	
$\langle v, q^h_v, \times, q^h_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	found a duplicate of v: delete it by writing a $\times$ symbol
$\langle \$, q^h_v, \$, q^h_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	move to the right and skip \$ and $\times$ symbols
$\langle \times, q^h_v, \times, q^h_v, +1 \rangle \in \Delta \quad (\forall v \in V)$	

$\langle \square, q_v^h, \square, q_v^l, -1 \rangle \in \Delta \quad (\forall v \in V)$	head at right end, go left back to the vertex v
$\langle u, q_v^l, u, q_v^l, -1 \rangle \in \Delta \quad (\forall u, v \in V,\ u \neq v)$	keep moving to the left until v is reached, skip over \$ and $\times$ symbols
$\langle u', q_v^l, u', q_v^l, -1 \rangle \in \Delta \quad (\forall u, v \in V, u \neq v)$	
$\langle \$, q_v^l, \$, q_v^l, -1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle \times, q_v^l, \times, q_v^l, -1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle v, q_v^l, v, q^h, +1 \rangle \in \Delta \quad (\forall v \in V)$	head on vertex v, move to the right and start removing duplicates of the next vertex
$\langle v', q_v^l, v', q^h, +1 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle \$, q^h, \$, q^h, +1 \rangle \in \Delta$	searching for another vertex to check for duplicates, skip \$ and $\times$ symbols
$\langle \times, q^h, \times, q^h, +1 \rangle \in \Delta$	
$\langle \square, q^h, \square, q^m, -1 \rangle \in \Delta$	head at right end, no more vertices to check for duplicates: continue with Phase 2b(iii)

Phase 2b(*iii*): *locate the next vertex to expand.*

$\langle w, q^m, w, q^m, -1 \rangle \in \Delta \quad (\forall w \in V)$	keep moving to the left until the marked vertex is found
$\langle \times, q^m, \times, q^m, -1 \rangle \in \Delta$	
$\langle v', q^m, v, q^e, +1 \rangle \in \Delta \quad (\forall v \in V)$	unmark the vertex, and start expanding the next vertex on the right (Phase 2b(i) restarts)
$\langle \times, q^e, \times, q^e, +1 \rangle \in \Delta$	skip deleted vertices ($\times$ symbols)
$\langle \square, q^e, \square, q_0^{e,v_0}, -1 \rangle \in \Delta$	head at right end: all vertices at the right of the \$ symbol have been expanded, continue with Phase 2c

Phase 2c: *compute the size of the connected component.* The Turing machine hangs if there are more than $\lfloor \alpha n \rfloor$ vertices on the right of the \$ symbol. While counting the vertices, the Turing machine erases the tape portion at the right of the \$ symbol. The internal state $q_i^{e,v}$ records both the counter value i and the last vertex v that has been counted; the internal state q_v^s writes back onto the tape the leftmost vertex at the right of the \$ symbol before generating the next connected component.

$\langle v, q_i^{e,u}, \square, q_{i+1}^{e,v}, -1 \rangle \in \Delta$	keep moving to the left until the \$ symbol is found; record the current counter value and the last vertex seen; skip deleted vertices ($\times$ symbols)

$(\forall u, v \in V, \forall i \in \{0, \ldots, \lfloor \alpha n \rfloor - 1\})$

$\langle \times, q_i^{e,u}, \Box, q_i^{e,u}, -1 \rangle \in \Delta$

$(\forall u \in V, \forall i \in \{0, \ldots, \lfloor \alpha n \rfloor\})$

$\langle \$, q_i^{e,u}, \$, q_u^s, +1 \rangle \in \Delta$ head on the \$ symbol: write back the last wiped-out vertex on the right of \$, and continue with the next iteration of the main loop of Phase 2

$(\forall u \in V, \forall i \in \{0, \ldots, \lfloor \alpha n \rfloor\})$

$\langle \Box, q_v^s, v, q^a, 0 \rangle \in \Delta \quad (\forall v \in V)$

It is straightforward to verify that the Turing machine has $\mathcal{O}(n)$ symbols, $\mathcal{O}(n^2)$ internal states, and $\mathcal{O}(n^3)$ transitions.

Phase 1 ends after exactly k steps. Phase 2 consists of several nested loops. We iterate over the n vertices of G. For each of these we check whether it was previously guessed (Phase 2a, $\mathcal{O}(k)$ steps), and then we write onto the tape the vertices in its neighborhood (Phase 2b(i), $\mathcal{O}(n)$ steps). Next, we remove the vertices belonging to S and the duplicated vertices (Phase 2b(ii), $\mathcal{O}((m+k)^2)$ steps, where m is the number of symbols at the left of the \$ symbol). We select another vertex among those at the left of \$ (Phase 2b(iii), $\mathcal{O}(m)$ steps), and we continue with Phase 2b(i). Eventually, the number of symbols written at the right of the \$ symbol (including the $\times$ symbols) is $m = \mathcal{O}(n^2)$, because we systematically cancel the duplicated vertices. Thus, the whole Phase 2b requires $\mathcal{O}(n^4)$ steps. Phase 2c ends in $\mathcal{O}(n^2)$ steps. Since Phases 2a–2c are executed once for every vertex of G, the Turing machine either hangs up or accepts in $\mathcal{O}(n^5)$ steps.

It is easy to verify that the Turing machine accepts if and only if the guessed symbols at the left of the \$ symbol represent a subset S that is an effective solution of the α-Balanced Separator problem. $\Box$

5.2. The Maximal Irredundant Set problem

Let us show a second example of W[P]-membership result by means of a reduction to a Turing machine computation problem. We want to place in the W-hierarchy the following parameterized problem:

Maximal Irredundant Set

Instance: A graph $G = (V, E)$; a positive integer k.

Parameter: k.

Question: Is there a set $V' \subseteq V$ of cardinality k such that (1) each vertex $u \in V'$ has a private neighbor and (2) V' is not a proper subset of any $V'' \subseteq V$ which also has this property? (A *private neighbor* of a vertex $u \in V'$ is a vertex $u' \in N[u]$ (possibly $u' = u$) with the property that for every vertex $v \in V' - \{u\}$, $u' \notin N[v]$.)

Maximal Irredundant Set is W[2]-hard (reduction from Dominating Set [2]); no membership result was previously known.

Theorem 8. *The* Maximal Irredundant Set *problem belongs to* W[P].

Proof. We show a parameterized reduction from Maximal Irredundant Set to the Bounded Nondeterminism Turing Machine Computation problem. Since the latter problem is W[P]-complete, the reduction proves that Maximal Irredundant Set belongs to W[P].

Given a graph $G = (V, E)$ with n vertices and a positive integer k, let us construct a nondeterministic Turing machine $T = (\Sigma, Q, \Delta)$, where Σ includes the alphabet symbols

$$\Sigma = \{\Box, \$\} \cup \{v : v \in V\}$$

and Q contains the internal states

$$Q = \{q_A, q_R, q^{\mathrm{IS}}, q^a, q^f, q^g, q^N, q_0^N, q_1^N, q^Y, q_0^Y, q_1^Y, q_2^Y, q_2^Y\} \\ \cup \{q_i : 0 \leqslant i \leqslant k-1\} \cup \{q_v^e, q_v^g, q_v^h : v \in V\} \cup \{q_{u,z}^b, q_{u,z}^c, q_{u,z}^d : u, z \in V\}.$$

Our Turing machine operates according to the following algorithm: it guesses k distinct vertices of G (that is, the subset V'), and then it verifies that V' is an irredundant set. Later, it considers each vertex $v \in V - V'$ and checks whether $v \cup V'$ is still an irredundant set. If so, V' cannot be a maximal irredundant set. On the other hand, if $v \cup V'$ is not an irredundant set for each vertex $v \in V - V'$, then V' is a maximal irredundant set, since if W is an irredundant set then $W - \{x\}$ remains an irredundant set.

For our convenience, we shall define a sub-routine that checks whether a given subset of vertices is an irredundant set.

Sub-routine "check irredundant set". This procedure checks whether a subset of vertices given in input is an irredundant set for the graph G. It is invoked by entering in the internal state q^{IS} while the head is reading a blank symbol. The subset of vertices to be checked is on the cells at the right of the head, and it is terminated by a blank symbol. The procedure terminates leaving the tape unchanged, and entering either the internal state q^Y (if the subset is an irredundant set) or the internal state q^N (if the subset is not an irredundant set).

Internal state q^a checks whether the symbol under the head has a private neighbor. Internal states $q_{u,z}^b$ and $q_{u,z}^c$ take care of checking whether $z \in N[u]$ is a private neighbor of u. Internal state $q_{u,z}^d$ indicates that $z \in N[u]$ is not a private neighbor, hence the Turing machine should consider the next element in $N[u]$, or enter q^N if z was the last element. Internal state q_u^e indicates that a private neighbor of u was found, and therefore that the Turing machine should check the next vertex in the subset, or enter q^Y if u was the last element in the subset. Internal state q^f is used to move the head to the front of the subset of vertices before entering the internal state q^Y.

$\langle \Box, q^{\mathrm{IS}}, \Box, q^a, +1 \rangle \in \Delta$	move to the right and start checking the first vertex
$\langle u, q^a, u, q_{u,z_0}^b, -1 \rangle \in \Delta \quad (\forall u \in V)$	search for a private neighbor for u, starting with the first vertex z_0 in $N[u]$ (assume a fixed ordering of $N[u]$: $z_0, z_1, \ldots, z_l$)
$\langle v, q_{u,z}^b, v, q_{u,z}^b, -1 \rangle \in \Delta \quad (\forall u, v \in V,\ \forall z \in N[u])$	go to the left end

$\langle \square, q^b_{u,z}, \square, q^c_{u,z}, +1 \rangle \in \Delta \quad (\forall u \in V,\ \forall z \in N[u])$	head on the left end, start checking whether z is a private neighbor for u
$\langle v, q^c_{u,z}, v, q^c_{u,z}, +1 \rangle \in \Delta$ $(\forall u, v \in V,\ v \neq u,\ \forall z \notin N[v])$ $\langle u, q^c_{u,z}, u, q^c_{u,z}, +1 \rangle \in \Delta \quad (\forall u \in V,\ \forall z \in N[u])$	z is fine till now, keep checking
$\langle v, q^c_{u,z}, v, q^d_{u,z}, -1 \rangle \in \Delta$ $(\forall u, v \in V,\ v \neq u,\ \forall z \in N[v])$	z is not a private neighbor, go back to the left end to check the next vertex in $N[u]$
$\langle v, q^d_{u,z}, v, q^d_{u,z}, -1 \rangle \in \Delta \quad (\forall u, v \in V,\ \forall z \in N[u])$	keep moving to the left until a blank is found
$\langle \square, q^d_{u,z_i}, \square, q^c_{u,z_{i+1}}, +1 \rangle \in \Delta$ $(\forall u \in V, \forall z_i \in N[u], i \neq l)$ $\langle \square, q^d_{u,z_l}, \square, q^N, 0 \rangle \in \Delta \quad (\forall u \in V,\ z_l \in N[u])$	head at left end: if there is a unchecked vertex in $N[u]$, check it; otherwise, terminate the sub-routine by answering "No"
$\langle \square, q^c_{u,z}, \square, q^e_u, -1 \rangle \in \Delta \quad (\forall u \in V,\ \forall z \in N[u])$	head at right end: z is a private neighbor, hence guessed vertex u is fine; go back to the vertex u before checking the next guessed vertex
$\langle v, q^e_u, v, q^e_u, -1 \rangle \in \Delta \quad (\forall u, v \in V,\ v \neq u)$ $\langle u, q^e_u, u, q^a, +1 \rangle \in \Delta \quad (\forall u \in V)$	keep moving to the left until vertex u is reached; then, check the guessed vertex at the right of v
$\langle \square, q^a, \square, q^f, -1 \rangle \in \Delta$	head at right end, no more guessed vertices to check; go back to the left end
$\langle u, q^f, u, q^f, -1 \rangle \in \Delta \quad (\forall u \in V)$ $\langle \square, q^f, \square, q^Y, 0 \rangle \in \Delta$	keep moving to the left until a blank is found, then terminate the sub-routine by answering "Yes"

It is easy to verify that, when invoked on a subset of m vertices, the sub-routine "returns" after $\mathcal{O}(nm^2)$ steps.

Using the "check irredundant set" sub-routine, we can easily write the main program of our Turing machine. It works in three phases: in the first phase it guesses the k-element subset of vertices, in the second phase it checks that the guessed subset is an irredundant set, and in the third phase it tries to augment the irredundant set by adding a vertex of G.

Phase 1: *guess k different vertices of G.*

$\langle \Box, q_i, v, q_{i+1}, +1 \rangle \in \Delta$ $(\forall v \in V, \forall i \in \{0, \ldots, k-2\})$	guess k vertices, leaving the head on the right-most one
$\langle \Box, q_{k-1}, v, q^g, 0 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle v, q^g, v, q_v^g, -1 \rangle \in \Delta \quad (\forall v \in V)$	start checking whether the vertex v under the head is duplicated (looking only at the vertices at the left of v)
$\langle w, q_v^g, w, q_v^g, -1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$	keep moving to the left, rejecting if the vertex v is found
$\langle v, q_v^g, v, q_R, 0 \rangle \in \Delta \quad (\forall v \in V)$	
$\langle \Box, q_v^g, \Box, q_v^h, +1 \rangle \in \Delta \quad (\forall v \in V)$	head at left end: no duplicates of v have been found, go back on v
$\langle w, q_v^h, w, q_v^h, +1 \rangle \in \Delta \quad (\forall v, w \in V, v \neq w)$	keep moving to the right until the vertex v is found; then start checking for duplicates of the vertex at the left of v
$\langle v, q_v^h, v, q^g, -1 \rangle \in \Delta \quad (\forall v \in V)$	

Phase 2: *check whether the guessed vertices are an irredundant set.* If the answer is negative (q^N), the Turing machine rejects (remember that all tape cells except those containing the guessed subset are blank). Otherwise, if the answer is positive (q^Y), the Turing machine continues with the third phase.

$\langle \Box, q^g, \Box, q^{\mathrm{IS}}, 0 \rangle \in \Delta$	Phase 1 ends with the head at the left of all guessed vertices; invoke the sub-routine to check whether they are an irredundant set
$\langle \Box, q^N, \Box, q_0^N, -1 \rangle \in \Delta$	the sub-routine answered "No"; move to the left to check whether it was applied to the original subset of guessed vertices; in the affirmative case, reject
$\langle \Box, q_0^N, \Box, q_R, 0 \rangle \in \Delta$	
$\langle \Box, q^Y, \Box, q_0^Y, -1 \rangle \in \Delta$	the sub-routine answered "Yes"; move to the left to check whether it was applied to the original subset of guessed vertices; in the affirmative case, move to the left to leave room for another vertex, and write a \$ symbol in the

	tape; then go back to the blank cell at the left of the guessed vertices, and continue with Phase 3
$\langle \square, q_0^Y, \square, q_1^Y, -1 \rangle \in \Delta$	
$\langle \square, q_1^Y, \$, q_2^Y, +1 \rangle \in \Delta$	
$\langle \square, q_2^Y, \square, q_2^Y, +1 \rangle \in \Delta$	

Phase 3: *for each vertex* $v \in V$, *add* v *to the subset and check whether it is an irredundant set.* The Turing machine rejects whenever it finds an augmented subset that is an irredundant set; if no augmented subsets are irredundant, the Turing machine accepts. In Phase 2 the Turing machine has written the \$ symbol at the left of the blank symbol that delimits the augmented subsets, so that it can distinguish an augmented subset from the original subset of guessed vertices. Assume a fixed ordering $v_0, v_1, \ldots, v_{n-1}$ of the vertices of G. Notice that this phase uses some instructions introduced in Phase 2.

$\langle \square, q_2^Y, v_0, q^{\mathrm{IS}}, -1 \rangle \in \Delta$	add the first vertex v_0 of G to the subset of guessed vertices, and ask the sub-routine whether the resulting subset is irredundant
$\langle \$, q_0^N, \$, q_1^N, +1 \rangle \in \Delta$	the sub-routine answered “No” and it was applied to an augmented subset: go back to the leftmost vertex of the subset
$\langle \square, q_1^N, \square, q_1^N, +1 \rangle \in \Delta$	
$\langle v_i, q_1^N, v_{i+1}, q^{\mathrm{IS}}, -1 \rangle \in \Delta \quad (\forall i \in \{0, \ldots, n-2\})$	replace the leftmost vertex v_i of the subset with v_{i+1}, then check the new subset
$\langle v_{n-1}, q_1^N, \square, q_A, 0 \rangle \in \Delta$	the leftmost vertex of the subset is v_{n-1}: accept, because no superset of the guessed vertices is irredundant
$\langle \$, q_0^Y, \$, q_R, 0 \rangle \in \Delta$	the sub-routine answered “Yes”, and it was applied on an augmented subset: reject

It is straightforward to verify that the Turing machine has $\mathcal{O}(n)$ symbols, $\mathcal{O}(n^2)$ states, and $\mathcal{O}(n^3)$ transitions; it is also easy to see that the Turing machine halts in $\mathcal{O}(k^2 n^2)$ steps, and that it accepts if and only if it guesses a maximal irredundant set. □

5.3. The bounded DFA Intersection problem

As a third example of W[P]-membership result by means of a reduction to a Turing machine computation problem, consider the following problem:

Bounded DFA Intersection

Instance: A set of k deterministic finite state automata $A_1, \ldots, A_k$ having the same input alphabet Σ; a positive integer m.

Parameter: either m, or (k, m).

Question: Is there a string $X \in \Sigma^m$ that is accepted by each A_i, $1 \leqslant i \leqslant k$?

We consider two different parameterized versions of the same problem; in the first version (Bounded DFA Intersection (m)) the parameter represents the length m of the string X, while in the second version (Bounded DFA Intersection (k, m)) the parameters are both the string length m and the number k of automata. The first version is W[2]-hard [13], while the second version is W[1]-hard [13]. No previous membership results are known.

Theorem 9. *The* Bounded DFA Intersection (m) *problem belongs to* W[P].

Proof. We show a parameterized reduction from Bounded DFA Intersection (m) to the Bounded Nondeterminism Turing Machine Computation problem. Since the latter problem is W[P]-complete, the reduction proves that Bounded DFA Intersection (m) belongs to W[P].

Given k deterministic finite state automata $A_1, \ldots, A_k$ with input alphabet Σ and a positive integer m, let us construct a nondeterministic Turing machine $T = (\Sigma', Q, \Delta)$, where Σ includes the alphabet symbols

$$\Sigma' = \{\square\} \cup \Sigma$$

and Q contains the internal states

$$\begin{aligned} Q = &\{q_A, q_R\} \cup \{q_i : 0 \leqslant i \leqslant m-1\} \\ &\cup \{q^{(i)} : 1 \leqslant i \leqslant k\} \cup \{q_j^{(i)} : 1 \leqslant i \leqslant k,\ q_j \in Q_{A_i}\}, \end{aligned}$$

where Q_{A_i} represents the set of internal states of A_i.

The Turing machine operates in two phases: it guesses a string of length m, and then it simulates the executions of the automata $A_1, \ldots, A_k$ on the guessed string, in turn. The Turing machine accepts if and only if all deterministic automata accept the guessed string.

Phase 1: *guess a string of length m.*

$\langle \square, q_i, s, q_{i+1}, -1 \rangle \in \Delta$ $(\forall i \in \{0, \ldots, m-2\},\ \forall s \in \Sigma)$	guess a string of length m, leaving the head on the leftmost symbol, then enter the initial state $q_0^{(1)}$ of the first automaton
$\langle \square, q_{m-1}, s, q_0^{(1)}, 0 \rangle \in \Delta \quad (\forall s \in \Sigma)$	

Phase 2: *for each* $i \in \{1, \ldots, k\}$, *emulate* A_i *on the guessed string*. In the following, $(q_j \xrightarrow{s} q_h)$ represents the rule of automaton A_i that forces state q_h from state q_j when reading input symbol s.

$\langle s, q_j^{(i)}, s, q_h^{(i)}, +1 \rangle \in \Delta$ $(\forall i \in \{1, \ldots, k\}, \forall (q_j \xrightarrow{s} q_h) \in A_i)$	keep emulating the automaton A_i
$\langle \square, q_j^{(i)}, \square, q^{(i)}, -1 \rangle \in \Delta$	head at the right end, go back to the left end if the automaton A_i accepts the string

$(\forall i \in \{1, \ldots, k\}, \forall q_j \in Q_{A_i}$ s.t. q_j "accepts")
$\langle s, q^{(i)}, s, q^{(i)}, -1 \rangle \in \Delta \quad (\forall i \in \{1, \ldots, k\}, \forall s \in \Sigma)$

$\langle \square, q^{(i)}, \square, q_0^{(i+1)}, +1 \rangle \in \Delta \quad (\forall i \in \{1, \ldots, k-1\})$ head at the left end, start emulating the next automaton A_{i+1}

$\langle \square, q^{(k)}, \square, q_A, 0 \rangle \in \Delta$ no more automata to emulate, therefore accept

It is easy to verify that the Turing machine T accepts if it guesses a string of length m that is accepted by every automaton; otherwise, T hangs up while simulating the execution of a rejecting automaton. It is also straightforward to verify that T uses m nondeterministic steps and it accepts in $\mathcal{O}(km)$ steps. Finally, a parameterized reduction may derive a description of the Turing machine T from the description of the k deterministic finite state automata $A_1, \ldots, A_k$. $\square$

Corollary 10. *The* Bounded DFA Intersection (k, m) *problem belongs to* W[1].

Proof. Just consider the same Turing machine T described in the proof of Theorem 9. When k is a parameter, the Turing machine halts in a parameterized number of steps. Therefore, Bounded DFA Intersection (k, m) reduces to the W[1]-complete Short Nondeterministic Turing Machine Computation problem [4], and hence it belongs to W[1]. $\square$

6. Conclusions

We have proposed a general method to establish membership results in the parameterized classes W[1], W[2], and W[P]. To validate our proposal, we have established some membership results for widely known parameterized problems: Perfect Code (in W[1]), Steiner Tree (in W[2]), α-Balanced Separator, Maximal Irredundant Set, and Bounded DFA Intersection (in W[P]).

We have also proved that the Short Multi-tape Nondeterministic Turing Machine Computation problem belongs to W[2]. This is the first exact characterization of the class W[2] according to the Turing machine model: candidate solutions for problems in W[2] can be generated and verified with a parameterized number of steps by using a unbounded number of tapes. (Recall that problems in W[1] can be generated and verified with a parameterized number of steps by using a constant or parameterized number of tapes.)

Finally, we have proved that the Bounded Nondeterminism Turing Machine Computation problem is W[P]-complete. Again, this is the first exact characterization of the class W[P] according to the Turing machine model.

Acknowledgments

We thank the anonymous referees for their helpful comments and suggestions. We are also grateful to Stephane Demri for pointing out a mistake in a preliminary version of the paper.

References

[1] K.R. Abrahamson, R.G. Downey, M.R. Fellows, Fixed parameter tractability and completeness IV: On completeness for $W[P]$ and PSPACE analogues, Ann. Pure Appl. Logic 73 (1995) 235–276.
[2] H.L. Bodlaender, B. de Fluiter, Intervalizing k-colored graphs, in: Proceedings of the 22th International Colloquium on Automata, Languages, and Programming (ICALP'95), Lecture Notes in Computer Science, Vol. 944, Springer, Berlin, 1995, 87–98.
[3] H.L. Bodlaender, D. Kratsch, Private communication, 1994.
[4] L. Cai, J. Chen, R.G. Downey, M.R. Fellows, The parameterized complexity of short computation and factorization, Arch. Math. Logic 36 (4/5) (1997) 321–337.
[5] M. Cesati, Perfect Code is $W[1]$-complete, Inform. Process. Lett. 81 (3) (2002) 163–168.
[6] M. Cesati, M. Di Ianni, Computation models for parameterized complexity, MLQ Math. Log. Q 43 (1997) 179–202.
[7] R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness II: On completeness for $W[1]$, Theoret. Comput. Sci. 141 (1995) 109–131.
[8] R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness I: basic results, SIAM J. Comput. 24 (1995) 873–921.
[9] R.G. Downey, M.R. Fellows, Threshold dominating sets and an improved characterization of $W[2]$, Theoret. Comput. Sci. 209 (1998) 123–140.
[10] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, New York, 1999.
[11] R.G. Downey, M.R. Fellows, B. Kapron, M.T. Hallett, H.T. Wareham, Parameterized complexity of some problems in logic and linguistics (Extended Abstract), in: Proceedings of Second Workshop on Structural Complexity and Recursion-theoretic Methods in Logic Programming, Lecture Notes in Computer Science, Vol. 813, Springer, Berlin, 1994, pp. 89–101.
[12] S. Dreyfus, R. Wagner, The Steiner problem in graphs, Networks 1 (1971) 195–207.
[13] H.T. Wareham, The parameterized complexity of intersection and composition operations on sets of finite-state automata, in: Proceedings of the Fifth International Conference on Implementation and Application of Automata, Lecture Notes in Computer Science, Vol. 2088, Springer, Berlin, 2000, 302–310.