Electronic Colloquium on Computational Complexity, Report No. 80 (2000)

Perfect Code is W[1]-complete

Marco Cesati

Department of Computer Science, Systems,
and Industrial Engineering
University of Rome "Tor Vergata"
via di Tor Vergata 110, I-00133 Rome, Italy
`cesati@uniroma2.it`

July 20, 2000

Abstract

We show that the parameterized problem PERFECT CODE belongs to W[1]. This result closes an old open question, because it was often conjectured that PERFECT CODE could be a natural problem having complexity degree intermediate between W[1] and W[2]. This result also shows W[1]-membership of the parameterized problem WEIGHTED EXACT CNF SATISFIABILITY.

1 Introduction

Parameterized Complexity [7] has been introduced by Downey and Fellows about ten years ago. It is a powerful framework with 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 *fixed parameter reduction*: a parameterized problem A reduces to a parameterized problem B if there is an FPT algorithm that transforms an instance (x, k) of A into an instance (x', k') of B, such that (1) k' depends only on k, and (2) $(x, k) \in A$ if and only if $(x', k') \in B$.

Furthermore, a hierarchy of classes $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \dots$ including likely fixed parameter intractable problems has been defined. Each W-class is the closure under fixed parameter

ISSN 1433-8092

reductions with respect to a kernel problem, which is usually formulated in terms of special mixed-type boolean circuits. For details, refer to [7].

Many 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 [4, 7]) 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 [3, 7]) do not.

The parameterized complexity of a problem complete for a W-class is precisely determined. However, many problems have been shown to be hard for some level and belonging to a higher level in the hierarchy. Although we can safely assume that these problems are not fixed parameter tractable, it seems that we still lack some information about their complexity. Directly citing from Downey and Fellows's book [7, page 487]:

> *[...] we find that there are many annoying gaps between hardness and membership results. For example, there is the* Perfect Code *problem that is hard for W[1] and a member of W[2]. It is very annoying not to know exactly where to put it. We conjecture that it either represents a natural degree intermediate between W[1] and W[2], or is complete for W[2]. In any case, it has resisted strenuous efforts to achieve a precise classification.*

This paper solves the question of the precise degree of parameterized complexity of the Perfect Code problem: we show that it belongs to W[1], and thus that it is W[1]-complete. As a corollary, the Weighted Exact CNF Satisfiability problem, which is equivalent to Perfect Code, also belongs to W[1].

2 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. Formally:

Perfect Code

Instance: A graph $G = (V, E)$.

Parameter: A positive integer 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 $\mathrm{N}[v] \cap V'$.

Downey and Fellows proved that the Perfect Code problem is W[1]-hard by means of a reduction from Independent Set [3]. Although Perfect Code may be easily placed in W[2] (see [3]), till now 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 ([7, pages 277 and 458]).

The following theorem represents the main result of this paper.

Theorem 1 *The* PERFECT CODE *problem belongs to W*[1].

Proof. We show a parameterized reduction from PERFECT CODE to the following problem:

> SHORT NONDETERMINISTIC TURING MACHINE COMPUTATION
>
> *Instance:* A singletape, singlehead nondeterministic Turing machine M; a word x on the input alphabet of M.
>
> *Parameter:* A positive integer k.
>
> *Question:* Is there a computation of M on input x that reaches a final accepting state in at most k steps?

Since SHORT NONDETERMINISTIC TURING MACHINE COMPUTATION is W[1]-complete [8, 1], the reduction proves that PERFECT CODE belongs to W[1].

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

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

and Q contains the internal states

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

(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 [2].)

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

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

$$\langle \Box, q_i, \sigma_v, q_{i+1}, +1 \rangle \in \Delta \quad (\forall\, i \in \{0, \ldots, k-2\}, \forall\, v \in V)$$

$$\langle \Box, q_{k-1}, \sigma_v, q_k, 0 \rangle \in \Delta \quad (\forall\, v \in V)$$

(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 $\mathrm{N}[v]$ of v. This is achieved by including the following instructions into the transition table Δ:

$$\left\langle \sigma_v, q_k, \sigma_v, q_v^l, -1 \right\rangle \in \Delta \quad (\forall\, v \in V)$$

$$\left\langle \sigma_w, q_v^l, \sigma_w, q', -1 \right\rangle \in \Delta \quad (\forall\, v, w \in V)$$

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

$$\left\langle \Box, q_v^l, \Box, q_v^r, +1 \right\rangle \in \Delta \quad (\forall\, v \in V)$$

$$\langle \sigma_w, q_v^r, \sigma_w, q_w^r, +1 \rangle \in \Delta \quad (\forall\, v, w \in V, v \neq w)$$

$$\langle \sigma_v, q_v^r, s_m, q_k, -1 \rangle \in \Delta \quad (\forall\, v \in V),\ \text{where } m = |\mathrm{N}[v]|$$

$$\langle \Box, q_k, \Box, q_k, +1 \rangle \in \Delta$$

Phase 3: taking the sum. The tape now contains exactly k symbols s_i; 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 non-overlapping 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:

$$\langle s_i, q_k, s_i, q_i^s, +1 \rangle \in \Delta \quad (\forall\, i \in \{1, \ldots, n\})$$

$$\langle s_j, q_i^s, s_j, q_t^s, +1 \rangle \in \Delta \quad (\forall\, i, j \in \{1, \ldots, n\}, i + j \leq n),\ \text{where } t = i + j$$

$$\langle \Box, q_n^s, \Box, q_A, 0 \rangle \in \Delta$$

$$\left\langle \Box, q_j^s, \Box, q_R, 0 \right\rangle \in \Delta \quad (\forall j \in \{1, \ldots, n-1\})$$

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 $k^2 + 4k + 3$ steps if and only if there exists a perfect code of size k in G. *Q.E.D.*

2.1 Weighted Exact CNF Satisfiability

The following problem is equivalent to PERFECT CODE:

WEIGHTED EXACT CNF SATISFIABILITY

Instance: A boolean expression E in conjunctive normal form.

Parameter: A positive integer k.

Question: Is there a truth assignment of weight k to the variables of E that makes exactly one literal in each clause of E true?

Downey and Fellows showed that PERFECT CODE reduces to WEIGHTED EXACT CNF SATISFIABILITY, and vice versa [3, 7]. Therefore, an immediate corollary of Theorem 1 is that WEIGHTED EXACT CNF SATISFIABILITY is W[1]-complete. Notice, however, that WEIGHTED CNF SATISFIABILITY (the same as above, but without the restriction that exactly one literal in each clause is true) is W[2]-complete [3].

3 Conclusions

W[1]-membership of PERFECT CODE easily derives from a singletape, singlehead Turing machine that guesses and verifies a candidate perfect code with a "short" computation. In general, it seems fruitful to think in terms of Turing machine computations when trying to prove membership in W[1]. This technique could also be applied to show membership in W[2], because the multihead, multitape Turing machine computation problem is W[2]-complete [2, 6].

References

[1] L. Cai, J. Chen, R. G. Downey, M. R. Fellows. The parameterized complexity of short computation and factorization. In *Proceedings of the Sacks Conference 1993, Archive for Math Logic*, 1997.

[2] M. Cesati, M. Di Ianni. *Computation Models for Parameterized Complexity.* Math. Log. Quart. 43 (1997) 179–202.

[3] R. G. Downey, M. R. Fellows. *Fixed-parameter tractability and completeness I: Basic results*, SIAM J. Comput. 24 (1995) 873–921.

[4] R. G. Downey, M. R. Fellows. *Fixed-parameter tractability and completeness II: On completeness for W[1]*, Theoret. Computer Sci. 141 (1995) 109–131.

[5] R. G. Downey, M. R. Fellows. Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy. In *Complexity Theory: Current Research*, S. H. K. Ambos-Spies and U. Schöning, Eds., Cambridge University Press, 191–226, 1993.

[6] R. G. Downey, M. R. Fellows. *Threshold dominating sets and an improved characterization of W[2].* Theoret. Computer Sci. 209 (1998) 123–140.

[7] R. G. Downey, M. R. Fellows. Parameterized Complexity. Springer-Verlag New York, Inc., 1999.

[8] 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 2nd Workshop on Structural Complexity and Recursion-theoretic Methods in Logic Programming, Lecture Notes in Computer Science 813*, 89–101, 1994.