Parameterized Parallel Complexity

Marco Cesati[1] and Miriam Di Ianni[2]

[1] Dip. di Informatica, Sistemi e Produzione, Univ. "Tor Vergata", via di Tor Vergata, I-00133 Roma, Italy.
cesati@uniroma2.it

[2] Istituto di Elettronica, Università di Perugia, via G. Duranti 1/A, I-06123 Perugia, Italy.
diianni@istel.ing.unipg.it

Abstract. We introduce a framework to study the parallel complexity of parameterized problems, and we propose some analogs of NC.

1 Introduction

The theory of NP-completeness [8] is a theoretical framework to explain the apparent asymptotical intractability of many problems. Yet, while many natural problems are intractable in the limit, the way by which they arrive at the intractable behaviour can vary considerably. For instance, deciding if the nodes of a graph can be properly colored by k colors is NP-complete even for a constant $k \geq 3$ [8], while there exists an algorithm deciding if the edges of a n nodes graph can be covered by k nodes in time $\mathcal{O}(n)$ for any fixed value of k. The parameterized complexity setting [6, 7] has been introduced in order to overcome the intrinsic inability of the standard NP-completeness model to give insight into this variety of behaviours.

In this paper we investigate the issue of which problems do admit efficient fixed parameter parallel algorithms. A first attempt to formalize the concept of efficiently fixed parameter parallelizable problems has been pursued by Bodlaender, Downey and Fellows: in a one-page abstract [3] they suggested the introduction of the class PNC as the parameterized analogue of NC. However, neither theoretical results nor applications to concrete natural problems were presented. We now want to give a deeper insight to such concept. According to the degree of efficiency we are interested in, several kinds of efficient parallelization for parameterized problems can be considered. In section 2, after reviewing some basic concepts of the parameterized complexity theory, we define the two classes of efficiently parallelizable parameterized problems, PNC and FPP, and we study their relationship with the class of sequentially tractable parameterized problems (FPT). We also present a non trivial tool for proving FPP-membership of parameterized graph problems based on the concept of treewidth and on the results in [1, 4, 2]. In section 3 we study the relationship between NC, PNC, and FPP. In section 4 we give two alternative characterizations of both FPP and

PNC, and we use them to prove the PNC-completeness of two parameterized structural problems.

2 The Classes PNC and FPP

Let Σ be a finite alphabet. A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$. Tipically, the second component represents a parameter $k \in \mathbb{N}$. The kth slice of the problem is defined as $L_k = \{ x \in \Sigma^* : \langle x, k \rangle \in L \}$. The class FPT of *fixed parameter tractable problems* contains all parameterized problems that have a solving algorithm with running time bounded by $f(k) |x|^\alpha$, where $\langle x, k \rangle$ is the instance of the problem, k is the parameter, f is an arbitrary function and α is a constant independent of x and k.

From now on, with the term "parallel algorithm" we always refer to a PRAM algorithm. The first class of efficiently parallelizable parameterized problems, PNC, has been defined in [3]. PNC (*parameterized analog of* NC) contains all parameterized problems which have a parallel solving algorithm with at most $g(k) |x|^\beta$ processors and running time bounded by $f(k)(\log |x|)^{h(k)}$, where $\langle x, k \rangle$ is the instance of the problem, k is the parameter, f, g and h are arbitrary functions, and β is a constant independent of x and k. The definition of PNC is coherent with the definition of FPT; indeed we can prove that PNC is a subset of FPT.

A drawback with the definition of PNC is the exponent in the logarithm which bounds the running time. We thus introduce the class of *fixed-parameter parallelizable problems* FPP that contains all parameterized problems having a parallel solving algorithm with at most $g(k) |x|^\beta$ processors and running time bounded by $f(k)(\log |x|)^\alpha$, where $\langle x, k \rangle$ is the instance of the problem, k is the parameter, f and g are arbitrary functions, and α and β are constants independent of x and k. Observe that, by definition, FPP $\subseteq$ PNC.

Several parameterized parallel problems can be proved to be included in FPP by directly showing a PRAM algorithm for them. Among them we recall parameterized VERTEX COVER and MAX LEAF SPANNING TREE. However, we can now present a different way for proving FPP-membership, based on the concept of treewidth. *Treewidth* was introduced by Robertson and Seymour [9], and it has proved to be a useful tool in the design of graph algorithms. Bodlaender and Hagerup [4] showed that there exists an optimal parallel algorithm on a EREW PRAM using $\mathcal{O}((\log n)^2)$ time and $\mathcal{O}(n)$ space which is able to construct a minimum-width tree decomposition of G or correctly decide that $tw(G) > k$. Therefore parameterized TREEWIDTH belongs to FPP.

The concept of *treewidth* turns out to be an useful tool for proving FPP-membership. In [1] it is shown that many problems that are (likely) intractable for general graphs are in NC when restricted to graphs of bounded treewidth. In particular, the authors prove that some graph properties (*MS* and *EMS properties*) are verifiable in NC for graphs of bounded treewidth. A careful study of the proofs in [1] reveals that such properties are also verifiable in FPP under the same hypothesis, since any optimal tree decomposition of width k can be transformed in $\mathcal{O}(\log n)$ time and $\mathcal{O}(n)$ operations on a EREW PRAM into a binary tree decomposition of depth $\mathcal{O}(\log n)$ and width at most $3k+2$ [4]. Therefore: *all problems involving MS or EMS properties are in* FPP *when restricted to graphs of parameterized treewidth.* Thus, for any (E)MS property P, the following Bounded Treewidth Graphs Verifying P problem belongs to FPP: given a graph G, is G satisfying P and such that $tw(G) \leq k$ (k being the parameter)? Lists of graph problems defined over (E)MS properties can be found in [2].

In a few cases it is possible to prove FPP-membership even without restrictions on the treewidth since some properties directly imply a bound on the treewidth. Therefore several natural problems (like Feedback Vertex Set) are in FPP, because (i) yes-instances have bounded treewidth, and (ii) the corresponding property is (E)MS.

3 Relationship Between FPP, PNC, and NC

Trivially, the slices of every parameterized problem belonging to FPP or PNC are included in NC. It is now worthwhile to verify whether the converse is true, that is, whether the parameterized versions of all problems whose slices belong to NC are included in FPP or PNC. Consider the Clique problem. Each slice Clique$_k$ is trivially in NC. Indeed, Clique$_k$ can be easily decided by a parallel algorithm that uses $\mathcal{O}(n^k)$ processors and requires constant time. Since Clique is likely not in FPT [7], it is also likely not in PNC. Thus, slice-membership to NC is not a sufficient condition for membership to PNC. Therefore, the classes FPP and PNC are somewhat "orthogonal" to NC, as much as the definition of FPT is orthogonal to the one of P. Nevertheless, there is a strong relation between P-completeness and FPT-completeness, where the completeness for FPT is defined with respect to parameterized reductions preserving membership to FPP or PNC. Indeed, consider the Weighted Circuit Value problem WCV: its instance consists of the description of a logical decision circuit C and of an input word w; the parameter k is the Hamming weight of w, and the question is whether C accepts w. We have proved the following.

Lemma 1. *WCV is in* FPT*, while each slice* WCV$_k$ *is* P*-complete.*

Previous lemma has an important consequence. Let L, L' be parameterized problems. We say that L FPP-reduces (PNC-reduces) to L' if there are an FPP-algorithm (PNC-algorithm) Φ and a function f such that $\Phi(x,k)$ computes $\langle x', k' = f(k)\rangle$ and $\langle x,k\rangle \in L$ if and only if $\langle x',k'\rangle \in L'$. It is not hard to prove the following.

Corollary 1. *Let $\leq_\Phi$ be either the* FPP*-reduction or the* PNC*-reduction. If there exists an* FPT*-hard problem L with respect to $\leq_\Phi$ such that all its slices L_k are included in* NC*, then* P=NC.

Corollary 1 implies that every FPT-hard parameterized problem must have at least one slice which is not in NC unless P = NC. Thus, the notion of FPT-completeness does not add anything to the classical notion of P-completeness to characterize hardly parallelizable parameterized problems. In particular, to distinguish between parallel intractability in the two contexts, we must find an hardly parallelizable parameterized problem with slices in NC. This is the aim of next section. However, we can hope to characterize "FPP's intractability" by proving the PNC-completeness of some problem (in the hypothesis FPP $\neq$ PNC), but we have no way to characterize "PNC's intractability".

4 Alternative Characterizations of FPP and PNC

NC is primarly defined as the class of problems that can be decided by uniform families of polynomial size, polylogarithmic depth circuits. Successively, it has been proved that uniform families of circuits and PRAM algorithms are polynomially related. In this paper, we have taken the inverse approach, by initially defining the classes FPP and PNC in terms of PRAM algorithms. In this section, we extend the relation between PRAM algorithms and uniform families of circuits to the parameterized setting. In this case, *uniform* means that the circuit C_n^k, able to decide instances of size n when the value of the parameter is k, can be derived in space $f(k)(\log n)^\alpha$, for some function f and constant α. It is possible to prove the following theorem by essentially using the same proof technique in [10].

Theorem 1. *A uniform circuit family of size $s(n,k)$ and depth $d(n,k)$ can be simulated by a PRAM-PRIORITY algorithm using $O(s(n,k))$ active processors and running in time $O(d(n,k))$.*

Conversely, there are a constant c and a polynomial p such that, for any FPP *(*PNC*) algorithm Φ operating in time $T(n,k)$ with processor bound $P(n,k)$, there exist a constant d_Φ and, for any pair $\langle n,k\rangle$, a circuit C_n^k of size at most $d_\Phi\, p(T(n,k),P(n,k),n)$ and depth $c\,T(n,k)$ realizing the same input-output behaviour of Φ on inputs of size n. Furthermore, the family $\{C_n^k\}$ is uniform.*

The previous theorem allows us to show the first PNC-complete problem. It is denoted as Bounded Size-Bounded Depth-Circuit Value Problem (in short, BS-BD-CVP) and is defined as follows: given a constant α, three functions f, g and h, a boolean circuit C with n input lines, an input vector x and an integral parameter k, decide if the circuit C (having size $g(k)\,n^\alpha$ and depth $h(k)(\log n)^{f(k)}$) accepts x. We have proved the following.

Corollary 2. BS-BD-CVP *is* PNC*-complete with respect to* FPP*-reductions.*

A second characterization of the classes of parameterized parallel complexity is based on random access alternating Turing machines. There is a strong relation between the time required by a random access alternating Turing machine and the depth of a simulating circuit, and between the space required by a random access alternating Turing machine and the size of a simulating circuit [5]. Such relation also holds in the parameterized setting. For the sake of brevity, we do not state the corresponding theorem. We only want to remark that such a characterization allows us to define another PNC-complete problem, Random Access Alternating Turing Machine Computation (in short, RA-ATMC): given a random access alternating Turing machine AT, an input word x and an integral parameter k, does $AT(x)$ halts within $\mathcal{O}(f(k)(\log|x|)^k)$ steps with $\mathrm{SPACE}^*(n) \in \mathcal{O}(g(k)\log n)$?

Corollary 3. RA-ATMC *is* PNC-*complete with respect to* FPP-*reductions.*

References

1. Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12, 308–340, 1991.
2. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. Technical Report, Department of Computer Science, Utrecht University, 1995.
3. Hans L. Bodlaender, Rodney G. Downey, and Michael R. Fellows. Applications of parameterized complexity to problems of parallel and distributed computation, 1994. Unpublished extended abstract.
4. Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. Technical Report UU-CS-1995-25, Department of Computer Science, Utrecht University, 1995.
5. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. of the ACM*, 28, 114–133, 1981.
6. Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. *Feasible Mathematics II*, 219–244. Birkhäuser, Boston, 1994.
7. Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *TCS*, 141, 109–131, 1995.
8. Michael R. Garey and Davis S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., New York, 1979.
9. Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7, 309–322, 1986.
10. Larry Stockmeyer and Uzi Vishkin. Simulation of parallel random access machines by circuits. *SIAM J. Comput.*, 13(2), 409–422, 1984.