Fixed-parameter parallelizability

Jeffrey Finkelstein
Computer Science Department, Boston University
March 28, 2016

1 Introduction

TODO Change FPP to FPNC everywhere.

2 Definitions

Definition 2.1. A Boolean circuit, or simply a circuit, $C$, is an directed acyclic graph. The size of a circuit, denoted $\text{size}(C)$, is the number of vertices in the underlying graph. The depth of a circuit, denoted $\text{depth}(C)$, is the length of a longest path from the root to a sink.

Definition 2.2. A function $f$ is circuit-computable if there is a nonuniform family of Boolean circuits $\{C_n\}_{n \in \mathbb{N}}$ such that for each $x$ we have $f(x) = C_n(x)$, where $n = |x|$.

A language is circuit-decidable if it has a circuit-computable characteristic function. We may also require that the size and depth of each circuit $C_n$ in the family be circuit-computable from just $n$, the length of the input. In this case, we say the language is circuit-decidable with uniform size and depth.

Nonunifority is required in Theorem 4.3 and Theorem 4.4, for example, in which the size of the input relative to the size of the parameter for an instance of the parameterized problem selects which of two circuits to use. If the circuits were uniform, we would need to include both alternatives, thereby making the circuits too deep to meet the required circuit depth bounds.

Definition 2.3. A language is a set of binary strings. A parameterization is a computable function $\kappa$ from binary strings to natural numbers. A parameterized problem is a pair $(Q, \kappa)$, where $Q$ is a language and $\kappa$ is a parameterization.
Definition 2.4 (NNC\(_d^d\) and NC\(_d^d\)). Let \(d\) be a natural number. A language \(Q\) is in the class NNC\(_d^d[b(n)]\) if there is a nondeterministic circuit family \(\{C_n\}\) such that for each string \(x\) of length \(n\),

- \(x \in Q\) if and only if \(C_n(x) = 1\),
- \(\text{size}(C_n) \leq n^{O(1)}\),
- \(\text{depth}(C_n) \leq O(\log^d n)\),
- \(\text{nondet}(C_n) \leq b(n)\).

If \(b\) is the zero function, then the language is in the class NC\(_d^d\).

Definition 2.5 (WP\(_d^d[P]\) and FPP\(_d^d\)). Let \(d\) be a natural number. A parameterized problem \((Q, \kappa)\) is in the class WP\(_d^d[P]\) if there are circuit-computable functions \(f, g, h\), and a nondeterministic circuit family \(\{C_{n,k}\}\) such that for each string \(x\),

- \(x \in Q\) if and only if \(C_{n,k}(x) = 1\), where \(n = |x|\) and \(k = \kappa(x)\),
- \(\text{size}(C_{n,k}) \leq f(k)n^{O(1)}\),
- \(\text{depth}(C_{n,k}) \leq g(k)\log^d n\),
- \(\text{nondet}(C_{n,k}) \leq h(k)\log n\).

If \(h\) is the zero function, then the parameterized problem is in the class FPP\(_d^d\).

Definition 2.6. Suppose \(d\) is a natural number and \((Q, \kappa)\) is a parameterized problem. We say the problem \((Q, \kappa)\) is eventually in NC\(_d^d\) if there is a circuit-computable function \(h\) and a NC\(_d^d\) family of circuits \(\{C_n\}_{n \in \mathbb{N}}\) such that for each \(x\) of length \(n\), if \(n \geq h(\kappa(x))\), then \(x \in Q\) if and only if \(C_n(x) = 1\).

Throughout we will often assume without loss of generality that functions like circuit size and depth bounds, nondeterministism bounds, and polynomials, are increasing.

3 Fixed-parameter parallelizability

The parameterized vertex cover problem is in FPP\(^1\), but the underlying decision problem is NP-complete. We are interested in finding a problem in FPP whose underlying decision problem is P-complete.

3.1 Circuit evaluation problems

One way to do this is to choose \(Q\) to be a P-complete problem and \(\kappa\) to be the “degenerate” parameterization function \(\kappa(x) = |x|\).
Definition 3.1 (p-s-Circuit Evaluation).

Instance: Boolean circuit $C$ on $n$ inputs, with size $m$ and depth $d$.

Parameter: $m$.

Question: Does $C(x) = 1$?

Theorem 3.2. p-s-Circuit Evaluation is in FPP and the underlying decision problem is P-complete.

Proof. The circuit evaluation problem is P-complete by [12]. Since the parameterization is monotonically increasing with the size of the input, the problem is in FPP by the technique of [10, Proposition 1.7]. To find a non-degenerate example, we can parameterize the circuit evaluation problem by depth instead of size.

Definition 3.3 (p-d-Circuit Evaluation).

Instance: Boolean circuit $C$ on $n$ inputs, with size $m$ and depth $d$.

Parameter: $d$.

Question: Does $C(x) = 1$?

Theorem 3.4. p-d-Circuit Evaluation is in FPP and the underlying decision problem is P-complete.

Proof. As stated in the proof of the previous theorem, the circuit evaluation problem is P-complete. Evaluating the circuit $C$ of size $m$ and depth $d$ on inputs $x$ can be performed by the depth-universal circuit $U$ of [6]. The size of $U$ is $O(m)$, which is of the form $f(d)m^O(1)$ as required. The depth of $U$ is $O(d)$, which is of the form $g(d)\log^{O(1)} m$. Therefore the circuit evaluation problem parameterized by circuit depth is in FPP (and specifically, in the class FPAC$^0$ as defined by [1], therein denoted paraAC$^0$).

3.2 Approximable optimization problems

Yet another way to do this is to find an optimization problem whose budget problem is P-complete while admitting a highly parallel approximation scheme via a “fixed-parameter parallelizable” algorithm.

Definition 3.5. An optimization problem $O$ is a four-tuple $(I, S, m, t)$, where $I$ is the set of instances, $S$ is the set of pairs $(x, w)$ where $w$ is a solution for $x$, the function $m$ computes the measure (or objective value) for such a pair, and $t$ is either min or max.

Definition 3.6. The standard parameterization of a minimization problem $O$, denoted $p-O$, is $(Q, \kappa)$, where $Q = \{(x, k) \mid m^*(x) \leq k\}$ and $\kappa(x, k) = k$. The inequality is reversed for a maximization problem.

Definition 3.7. Suppose $(I, S, m, t)$ is an optimization problem and $(x, y) \in S$. The performance ratio of the solution $y$ (with respect to $x$), denoted $R(x, y)$, is defined by

$$R(x, y) = \max \left( \frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)} \right)$$
The performance ratio $R(x,y)$ is a number in the interval $[1, \infty)$. The closer $R(x,y)$ is to 1, the better the solution $y$ is for $x$, and the closer $R(x,y)$ to $\infty$, the worse the solution.

**Definition 3.8.** An approximation scheme for an optimization problem is a function $A$ such that for all $x$ and all positive integers $k$ we have $(x, A(x,k)) \in S$ and $R(x, A(x,k)) \leq 1 + \frac{1}{k}$.

An approximation scheme induces a family of functions, $\{A_k\}_{k \in \mathbb{N}}$, that form progressively better approximations for the optimization problem.

**Definition 3.9.** Suppose $O$ is an optimization problem with $O = (I, S, m, t)$ with $I$ and $S$ in NC and $m$ in FNC. An optimization problem $O$ is in NCAS if there is an approximation scheme $A$ for $O$ such that for each $k$, we have $A_k \in \text{FNC}$, where $A_k(x) = A(x,k)$ for each $x$. The problem is in FNCAS if there is an approximation scheme $A$ for $O$ such that $A \in \text{FNC}$ (i.e. on both inputs).

This definition is adapted from [10, Definition 1.31]

**Definition 3.10.** An optimization problem $O$ is in ENCAS if there is a circuit family $\{A_{n,k}\}$ and circuit-computable functions $f$ and $g$ such that

- $\{A_{n,k}\}$ is an approximation scheme for $O$,
- $\text{size}(A_{n,k}) \leq f(k)n^{O(1)}$,
- $\text{depth}(A_{n,k}) \leq g(k)\log^{O(1)}n$.

**Proposition 3.11.** $\text{FNCAS} \subseteq \text{ENCAS} \subseteq \text{NCAS}$.

This theorem is an adaptation of [10, Theorem 1.32].

**Theorem 3.12.** Let $O$ be an optimization problem. If $O$ is in ENCAS, then $p-O$ is in FPP.

**Proof.** Assume without loss of generality that $O$ is a minimization problem; the proof is similar if it is a maximization problem. Let $\{m_n\}$ be the NC circuit family that computes the measure function. Let $\{A_{n,k}\}$ be the circuit family such that

- $R(x, A_{n,k}(x,k)) \leq 1 + \frac{1}{k}$ for each $x$ and $k$,
- $\text{size}(A_{n,k}) \leq f(k)n^{O(1)}$,
- $\text{depth}(A_{n,k}) \leq g(k)\log^{O(1)}n$,

for some circuit-computable functions $f$ and $g$. Define the circuit family $\{C_{n,k}\}$ as

$$C_{n,k}(x,k) = 1 \iff m(x, A_{n,k+1}(x,k+1)) \leq k,$$

so $C_{n,k}$ outputs 1 if and only if the approximate solution corresponding to parameter $k + 1$ measures less than $k + 1$. (The function $m$ is really a circuit
as well, chosen from a family of circuits depending on the number of bits in its inputs.)

The size of $C_{n,k}$ is $O(\text{size}(m) + \text{size}(A_{n,k+1}))$, which is $f'(k)n^{O(1)}$ for some circuit-computable function $f'$. The depth is $O(\text{depth}(m) + \text{depth}(A_{n,k+1}))$, which is $g'(k + 1)\log^{O(1)} n$ for some circuit-computable function $g'$. It remains to show correctness of $C_{n,k}$.

Let $x$ be a string, let $k$ be a natural number, and let $y = A_{n,k+1}(x, k + 1)$. If $C_{n,k} = 1$, then $m(x, y) \leq k$, so $m^*(k) \leq k$ and therefore $(x, k) \in p-O$. For the converse, if $C_{n,k} = 0$, then $m(x, y) \geq k + 1$, so

$$m^*(x) \geq \frac{m(x, y)}{1 + \frac{1}{k+1}} \geq \frac{k + 1}{1 + \frac{1}{k+1}} = \frac{(k+1)^2}{k + 2} > k.$$  

Thus $(x, k) \notin p-O$. Therefore, we conclude that $p-O$ is in FPP.

The converse does not hold: the minimum vertex cover problem is a counterexample because it is in FPP \[1, \text{Theorem 4.5}\]. TODO Show an example of an optimization problem whose budget problem is P-complete and whose standard parameterization is in FPP but for which no ENCAS exists.

Our goal now reduces to finding an optimization problem in ENCAS whose budget problem is P-complete.

**Definition 3.13 (Maximum Flow).**

- **Instance:** directed graph $G$, a natural number capacity $c_e$ for each edge $e$, source node $s$, and target node $t$.
- **Solution:** flow $F$, defined as a real number $F_e$ for each edge $e$ such that $F_e \leq c_e$ and at each vertex the total in-flow is at least the total out-flow.
- **Measure:** total in-flow at $t$.
- **Type:** maximization.

**Theorem 3.14.** If $\text{NC} = \text{RNC}$, then the budget problem for Maximum Flow is P-complete and the standard parameterization is in FPP.

**Proof.** The budget problem for Maximum Flow is P-complete \[11, \text{Problem A.4.4}\]. The Maximum Flow problem is in randomized FNCAS \[7, \text{Theorem 4.5.2}\]. If $\text{NC} = \text{RNC}$, then randomized FNCAS equals deterministic FNCAS. Thus, the problem is in ENCAS, by Proposition 3.12. Finally, the standard parameterization is in FPP by Theorem 3.12. □

TODO Can the randomization part of the RNC algorithm for MaxFlow be absorbed in the fixed-parameter part of the algorithm?

### 4 FPP is to NC as WP is to NNC

#### 4.1 Example problem in WP

What kind of problems are in the class $\text{WP}[P]$?
Definition 4.1 (Group Rank).

Instance: finite group $G$ given as a product table, positive integer $k$.

Question: Does $G$ have a generating set of cardinality $k$?

Let $p$-Group Rank denote the standard parameterization of Group Rank.

Theorem 4.2. $p$-Group Rank is in WP$^1[P]$.

Proof. The (unparameterized) language Group Rank is in $\beta_2\text{FOLL}$ [8, Theorem 4.2]. The algorithm nondeterministically chooses a set of $k$ group elements, each requiring $O(\log n)$ bits to represent, then (deterministically) verifies that the set generates the entire group. Thus there is a nondeterministic circuit family that decides Group Rank with (parameter-less) size and depth bounds of $n^{O(1)}$ and $\log \log n$, respectively, and using $O(k) \log n$ nondeterministic bits. Therefore the parameterized problem is in WP$^1[P]$. \qed

4.2 Does FPP = WP[P]?

This theorem is a characterization of parameterized problems in the class FPP$^d$ as problems for which the parameter suggests an input size threshold, below which it suffices to perform an exhaustive search to solve the problem and after which we require an efficient algorithm. It is the adaptation of [10, Theorem 1.37].

Theorem 4.3. Suppose $d$ is a natural number and $(Q, \kappa)$ is a parameterized problem. If $Q$ is circuit-decidable with uniform size and depth and $(Q, \kappa)$ is eventually in NC$^d$, then $(Q, \kappa)$ is in FPP$^d$.

Proof. Let $h$ be the function that defines the lower bound on the length after which $(Q, \kappa)$ is in NC$^d$. The nonuniform family of circuits $\{A_{n,k}\}$ that decides $(Q, \kappa)$ is defined by

$$A_{n,k} = \begin{cases} C^1_n & \text{if } n < h(k) \\ C^2_n & \text{otherwise,} \end{cases}$$

where $\{C^1_n\}$ is the family of circuits that decides $Q$ and $\{C^2_n\}$ is the family of NC$^d$ circuits that eventually decides $(Q, \kappa)$. The correctness of $A_{n,k}$ follows from the correctness of $C^1_n$ and $C^2_n$.

Let $S$ and $D$ be the circuit-computable function that give the size and depth, respectively, of $C^1_n$ from $n$. If $n < h(k)$, then

$$\text{size}(A_{n,k}) = \text{size}(C^1_n) = S(n) \leq S(h(k)), \quad \text{depth}(A_{n,k}) = \text{depth}(C^1_n) = D(n) \leq D(h(k)).$$

If $n \geq h(k)$, then

$$\text{size}(A_{n,k}) = \text{size}(C^2_n) = n^{O(1)}, \quad \text{depth}(A_{n,k}) = \text{depth}(C^2_n) = O(\log^d n).$$
An upper bound for the size of $A_{n,k}$ in either case is $S(h(k))n^{O(1)}$ and for the depth $O(D(h(k)) \log^d n)$. Since $S$ and $D$ are computable, composing each with $h$ yields another computable function, so these size and depth bounds meet the requirements of the definition of $FPP^d$.

The next theorem is similar in that it is a characterization of $WP^d[P]$, a nondeterministic extension of $FPP^d$. It is the adaptation of [10, Proposition 3.7]. The parameterized problems in $WP^d[P]$ of most interest to us are those for which there is a nondeterministic $NC^d$ algorithm with nondeterminism limited by a function of the parameter.

TODO There’s some wiggle room with this theorem: if we change the definition of $WP^d[P]$ to have depth $g(k) + \log^d n$, then we can get $NNC^d$ instead of $NNC^{d+\epsilon}$ in the conclusion. This sort of depth bound appears in [1].

**Theorem 4.4.** Suppose $d$ is a natural number, $e$ is a positive integer, and $(Q, \kappa)$ is a parameterized problem. If $(Q, \kappa) \in WP^d[P]$ and $Q \in NNC^d[\text{poly}]$, then there is a nondeterministic $NC^{d+\epsilon}$ circuit family $\{C_{n,k}\}$ and a circuit-computable function $h$ such that $\{C_{n,k}\}$ decides $Q$ and nondet($C_{n,k}$) $\leq h(k) \log n$.

**Proof.** Assume $(Q, \kappa) \in WP^d[P]$ and $Q \in NNC^d[\text{poly}]$. Define the circuit family $\{A_{n,k}\}$ by

$$A_{n,k} = \begin{cases} C^1_{n,k} & \text{if } f_1(k) \leq n \text{ and } g_1(k) \leq \log^e n \\ C^2_n & \text{otherwise,} \end{cases}$$

where

- $\{C^1_{n,k}\}$ is the $WP^d[P]$ circuit family,
- $f_1$ and $g_1$ are the circuit-computable functions on the parameter that appear in the size and depth bounds, respectively, for $C^1_{n,k}$,
- $\{C^2_n\}$ is the $NNC^d[\text{poly}]$ circuit family.

The fact that $A_{n,k}$ correctly decides $Q$ follows from the correctness of the circuits $C^1_{n,k}$ and $C^2_n$. It remains to show that $A_{n,k}$ has the required size, depth, and nondeterminism bounds.

If $f_1(k) \leq n$ and $g_1(k) \leq \log^e n$, then

$$\begin{align*}
\text{size}(A_{n,k}) &\leq \text{size}(C^1_{n,k}) \leq f_1(k)n^{O(1)} \leq n^{O(1)} \\
\text{depth}(A_{n,k}) &\leq \text{depth}(C^1_{n,k}) \leq g_1(k) \log^d n \leq \log^{d+\epsilon} n,
\end{align*}$$

and otherwise

$$\begin{align*}
\text{size}(A_{n,k}) &\leq \text{size}(C^2_n) \leq n^{O(1)} \\
\text{depth}(A_{n,k}) &\leq \text{depth}(C^2_n) \leq \log^d n.
\end{align*}$$

The overall size and depth upper bounds for $A_{n,k}$ are thus $n^{O(1)}$ and $O(\log^{d+\epsilon} n)$, respectively.
For the nondeterminism, there are three cases. First, if \( f_1(k) \leq n \) and \( g_1(k) \leq \log^e n \), then
\[
\text{nondet}(A_{n,k}) \leq \text{nondet}(C_{n,k}^1) \leq h_1(k) \log n,
\]
where \( h_1 \) is the circuit-computable of the parameter that appears in the nondeterminism bound for \( C_{n,k}^1 \). Second, if \( f_1(k) > n \), then
\[
\text{nondet}(A_{n,k}) \leq \text{nondet}(C_{n,k}^2) \leq n^{O(1)} \leq f_1(k)^{O(1)}.
\]
Finally, if \( g_1(k) > \log^e n \), then \( 2^{(g_1(k))^{1/e}} > n \), which implies
\[
\text{nondet}(A_{n,k}) \leq \text{nondet}(C_{n,k}^0) \leq n^{O(1)} \leq 2^{O((g_1(k))^{1/e})}.
\]
If we choose \( h(k) = \max\{h_1(k), f_1(k)^{O(1)}, 2^{O((g_1(k))^{1/e})}\} \), then we can conclude \( \text{nondet}(A_{n,k}) \leq h(k) \log n \).

\[\text{Assumption 4.5. There is a parameterized problem } (Q, \kappa) \text{ complete for WP}^d[P] \text{ under FPP}^d \text{ many-one reductions such that } Q \in \text{NNC}^d[\text{poly}].\]

This theorem is an adaptation of one direction of [10, Theorem 3.29].

\[\text{Theorem 4.6 (Theorem 3.29, part i). Assume Assumption 4.5. Suppose } d \text{ is a natural number and } e \text{ is a positive integer. If there is a circuit-computable, nondecreasing, unbounded function } i \text{ such that } \text{NC}^d = \text{NNC}^{d+e}[i(n) \log n], \text{then } \text{FPP}^d = \text{WP}^d[P]. \text{ TODO Does } i \text{ need to be computable here?}\]

\[\text{Proof. Assume } \text{NC}^d = \text{NNC}^{d+e}[i(n) \log n]. \text{ By Assumption 4.5, let } (Q, \kappa) \text{ be a parameterized problem complete for WP}^d[P] \text{ with } Q \in \text{NNC}^d[\text{poly}]. \text{ By Theorem 4.4, there is a circuit-computable function } h \text{ and an NNC}^{d+e} \text{ circuit family } \{C_{n,k}\} \text{ such that } \text{nondet}(C_{n,k}) \leq h(k) \log n.\]

Let \( f \) be a circuit-computable function such that \( n \geq f(k) \) implies \( h(k) \leq i(n) \).

\[\text{TODO Explain why such an } f \text{ must exist.} \text{ Now consider the set } Q^+, \text{ defined by }
Q^+ = Q \cap \{x \mid |x| \geq f(\kappa(x))\}.
\]

Since \( |x| \geq f(\kappa(x)) \) for each \( x \in Q^+ \), we have \( h(\kappa(x)) \leq i(|x|) \). Thus \( Q^+ \) is in \( \text{NNC}^{d+e}[i(n) \log n] \). By assumption, \( Q^+ \) is therefore also in \( \text{NC}^d \). This means \( Q \) is eventually in \( \text{NC}^d \), so we have \( Q \in \text{FPP}^d \) by Theorem 4.3.

This lemma is an adaptation of [10, Lemma 3.24].

\[\text{Lemma 4.7. Suppose } e \text{ is a positive integer and } f \text{ and } g \text{ are increasing, circuit-computable functions. There are functions } i_f \text{ and } i_{g,e} \text{ such that }
\]
\[
\begin{itemize}
  \item \( f(i_f(n)) \leq n \) for each \( n \geq f(1) \),
  \item \( g(i_{g,e}(n)) \leq \log^e n \) for each \( n \geq g(1) \).
\end{itemize}
\]
Furthermore, these functions are circuit-computable, nondecreasing, unbounded.
Proof. Define $i_f$ by

$$i_f(n) = \begin{cases} 
\max\{j \in \mathbb{N} \mid f(j) \leq n\} & \text{if } n \geq f(1) \\
1 & \text{otherwise},
\end{cases}$$

and $i_{g,e}$ by

$$i_{g,e}(n) = \begin{cases} 
\max\{j \in \mathbb{N} \mid g(j) \leq \log^e n\} & \text{if } n \geq g(1) \\
1 & \text{otherwise}.
\end{cases}$$

It is straightforward to prove that these functions are nondecreasing and unbounded. To compute $i_f$ (computing $i_{g,e}$ is similar), we compute $f(1), \ldots, f(n)$ in parallel, filter by only those values that are at most $n$, and choosing the index of the rightmost value that passes the filter.

This is the $\text{NC}^d$ bounded version of the parameterized problem from [10, Lemma 3.26].

**Definition 4.8** ($p$-$\log$-$\text{NC}^d$-$\text{Circuit Sat}$).

- **Instance:** Boolean circuit $C$ on $n$ inputs, with size $m$ and depth $O(\log^d n)$.
- **Parameter:** $n/\log m$.
- **Question:** Is $C$ satisfiable?

**Lemma 4.9.** $p$-$\log$-$\text{NC}^d$-$\text{Circuit Sat}$ is in $\text{WP}^d[P]$.

**Proof.** The depth-universal circuit $U$ of [6] proves membership of this problem in $\text{WP}^d[P]$. The number of nondeterministic bits required by the universal circuit is simply $n$, the size of the input to the circuit $C$, which is of the form $h(k) \log m$ if we choose $h(k) = k = n/\log m$ where $k = n(x)$.

**TODO** Ideally, it should be complete for it too (this would prove the assumption above), but the problem comes in the reduction from an arbitrary parameterized problem to this one: the circuit depth needs to be polylogarithmic, but it may be larger. Perhaps we can use the other complete problems from [3] here?

This is an adaptation of the other direction of [10, Theorem 3.29].

**Theorem 4.10.** Suppose $e$ is a positive integer. If $\text{FPP}^d = \text{WP}^d[P]$, then there is a circuit-computable, nondecreasing, unbounded function $i$ such that $\text{NC}^{d+e} = \text{NNC}^d[i(n) \log n]$.

**Proof.** Assume $\text{FPP}^d = \text{WP}^d[P]$. Since $p$-$\log$-$\text{NC}^d$-$\text{Circuit Sat}$ is in $\text{WP}^d[P]$, it is now in $\text{FPP}^d$ as well. Thus there is a deterministic circuit family $\{C_{m,k}\}$ and circuit-computable functions $f$ and $g$ such that

- for each instance $D$, we have $D \in \text{NC}^d$-$\text{Circuit Sat}$ if and only if $C_{m,k}(D) = 1$, 

9
• $\text{size}(C_{m,k'}) \leq f(k')m^{O(1)}$,

• $\text{depth}(C_{m,k'}) \leq g(k') \log^d m$.

Assume without loss of generality that $f$ and $g$ are increasing. Let $i_f$ and $i_{g,e}$ be the functions corresponding to $f$ and $g$, respectively, guaranteed by Lemma 4.7. Let $i(n) = \min(i_f(n), i_{g,e}(n))$, for each natural number $n$.

Let $Q \in \text{NCC}^d[i(n) \log n]$ Suppose $\{D_n\}$ is the family of $\text{NCC}^d[i(n) \log n]$ circuits that decides $Q$. For each input $x$, let $D_x$ denote $D_n$ with $x$ hardcoded.

Let $R_n$ be the circuit-computable function $x \mapsto D_x$, where $n$ denotes the length of $x$. Then

• for each $x$, we have $x \in Q$ if and only if $D_x$ is satisfiable,

• $\text{size}(R_n) \leq O(\text{size}(D_x)) = O(\text{size}(D_n)) = n^{O(1)}$,

• $\text{depth}(R_n) \leq O(1)$.

If $\kappa$ denotes the parameter function for $p$-$\log$-$\text{NC}^d$-Circuit SAT, we can now define the family $\{A_n\}$ as $A_n = C_{m,k'} \circ R_n$, where

$$m = |R_n(x)| = O(\text{size}(R_n)) = n^{O(1)},$$

and

$$k' = \kappa(R_n(x)).$$

We claim that $\{A_n\}$ is the $\text{NC}^{d+e}$ circuit family that decides $Q$.

The correctness of $A_n$ follows from the correctness of both $R_n$ and $C_{m,k'}$. The size of $R_n$ is $n^{O(1)}$ and the size of $C_{m,k'}$ is $f(k')m^{O(1)}$. Assuming $n \leq m \leq n^{O(1)}$,

\[
\text{size}(A_n) = n^{O(1)} + f(k')m^{O(1)} \\
\leq f(k')n^{O(1)} \\
= f\left(\frac{i(n) \log n}{\log m}\right)n^{O(1)} \\
\leq f\left(\frac{i_f(n) \log n}{\log n}\right)n^{O(1)} \\
\leq f(i_f(n))n^{O(1)} \\
\leq n^{O(1)}.
\]
The depth of $R_n$ is $O(1)$ and the depth of $C_{m,k'}$ is $g(k') \log^d m$. Thus,

$$\text{depth}(A_n) = O(1) + g(k') \log^d m$$
$$\leq O(g(k') \log^d n)$$
$$= O(g\left(\frac{i(n) \log n}{\log m}\right) \log^d n)$$
$$\leq O(g\left(\frac{i_{g,e}(n) \log n}{\log n}\right) \log^d n)$$
$$\leq O(g(i_{g,e}(n)) \log^d n)$$
$$\leq O((\log^e n)(\log^d n))$$
$$\leq O(\log^{d+e} n).$$

We have shown that $\{A_n\}$ is a $\text{NC}^{d+e}$ circuit family that decides $Q$, thus $Q \in \text{NC}^{d+e}$. Since $Q$ was an arbitrary element of $\text{NNC}^{i(n) \log n}$, we conclude that $\text{NC}^{d+e} = \text{NNC}^{i(n) \log n}$.

### 4.3 Does $\text{WP}[P] = \text{W}[P]$?

**Theorem 4.11.** $\text{WP}[P] \subseteq \text{W}[P]$.

**Proof.** This follows from the usual simulation of a circuit by a Turing machine, which can be done in linear time with respect to the size of the circuit. The number of nondeterministic bits required by the Turing machine is exactly the same as the number required by the circuit.

Whether the converse inclusion holds is not so clear.

**Conjecture 4.12.** $\text{WP}[P] \subset \text{W}[P]$ unless $\text{NC} = \text{P}$.

**Justification.** If we use our intuition that $\text{WP}[P]$ is like $\text{NNC}[\text{poly}]$ and $\text{W}[P]$ is like $\text{NP}$, then we might expect $\text{WP}[P] = \text{W}[P]$, since $\text{NNC}[\text{poly}] = \text{NP}$ [13, Theorem 2.2]. However, the technique used to show $\text{NP} \subseteq \text{NNC}[\text{poly}]$ requires the use of $t(n)^2$ nondeterministic bits, where $t(n)$ is the running time of the nondeterministic Turing machine (which is $f(k)n^{O(1)}$ in this case). Other approaches I attempted face a similar problem. I suspect instead that $\text{WP}[P] = \text{W}[P]$ if and only if $\text{NC} = \text{P}$, similar to the way $\text{NNCO} = \text{NPO}$ [9] if and only if $\text{NC} = \text{P}$.

### 5 The $\text{WP}$ hierarchy

For the necessary background in logic, see [10, Chapter 4].
5.1 Definition of finite levels of the hierarchy

Let $\Phi$ be a class of formulas and let $\phi$ be an element of $\Phi$ with one free
relation variable $X$ of arity $s$. Define the weighted definability problem and the
corresponding parameterized problem as follows.

**Definition 5.1** ($WD_{\phi}$ [10, Section 4.3]).

*Instance:* structure $A$, natural number $k$.

*Question:* Is there an $S \subseteq A^s$ such that $|S| = k$ and $A \models \phi(S)$?

**Definition 5.2** ($p$-WD$_{\phi}$ [10, Section 5.1]).

*Instance:* structure $A$, natural number $k$.

*Parameter:* $k$

*Question:* Is there an $S \subseteq A^s$ such that $|S| = k$ and $A \models \phi(S)$?

This definition is adapted from [10] Definition 5.1.

**Definition 5.3.** For each natural number $d$ and positive integer $t$, let $WP^d[t] = [\ldots]$

5.2 Example problem in finite levels of the hierarchy

TODO Show a natural problem that is actually in a finite level of the
hierarchy here.

In [subsection 4.1] we proved that $p$-GROUP RANK is in $WP^1[P]$. Is this
problem in a fixed finite level of the WP hierarchy? Let us try to express this
problem in first-order logic.

**Definition 5.4** (Axiomatization of groups). The signature for groups has a
constant symbol for the identity element denoted $e$, a unary function for the
inverse denoted $x^{-1}$, and a binary function for the group operation. Let the
group axioms be defined as first-order formulae as follows.

\[
\begin{align*}
\text{HASIDENTITY} &= \forall x \left( ex = x \land xe = x \right) \\
\text{HASINVERSES} &= \forall x \left( x^{-1}x = xx^{-1} = e \right) \\
\text{ISASSOCIATIVE} &= \forall x\forall y\forall z \left( (xy)z = x(yz) \right)
\end{align*}
\]

Let $\text{IsGROUP}$ be the conjunction of these three formulae. Now for any finite
structure $A$, we have $A \models \text{IsGROUP}$ if and only if $A$ is a group. Let $\text{GROUP}$
denote the class of all finite structures that are valid groups, that is, the class
of all structures that model $\text{IsGROUP}$. (Since the universal quantifiers can be
placed in the beginning of the formula, $\text{IsGROUP}$ is in $\Pi_1$.)

Let $\text{IsMEMBER}(g, X)$ denote the formula for which $G \models \text{IsMEMBER}(g, X)$
effectively when $g \in \langle S \rangle_\varphi$. In [2, Subsection 3.3] the authors fail to show that this
formula is in $\text{FO}[\log \log n]$, but we do know that the problem of deciding whether
$G \models \text{IsMEMBER}(g, X)$, given a group $G$, a subset $S$ of $G$, and a group element
g, is in $L$ [8, Lemma 3.5]. Since $L = \text{FO}(\text{DTC})$, our best upper bound for the
definability of $\text{IsMEMBER}$ is $\text{FO}(\text{DTC})$. 

12
In this setting, Group Rank is $\text{WD}_\phi$, where

$$\phi(X) = \text{IsGroup} \land \forall g (\text{IsMember}(g, X)).$$

In the parameterized setting, $p$-Group Rank is $p$-$\text{WD}_\phi$ and is reducible to $p$-$\text{MC}(\text{FO(DTC)})$ via the reductions given above. This upper bound does not allow us to say for sure whether $p$-Group Rank is in one of the finite levels of the WP hierarchy.

5.3 Does WP[$t$] = W[$t$]?

TODO I think no, unless P = NC; prove it.

5.4 WP[1] probably does not equal FPP

TODO fill me in

6 Parameterized complexity within P

What we really care about is the parameterized parallel complexity of problems in FPP, and specifically those with underlying decision problems that are P-complete. The WP hierarchy does not capture this. This is similar to the way NNC[log] allows us to consider efficient highly parallel verification but does not capture polynomial-time decision.

6.1 Completeness in FPT

We define P-completeness so that problems that are P-complete are unlikely to see a significant decrease in time complexity when parallelism is allowed, under the assumption that NC $\neq$ P. Let us define FPT-completeness similarly, so that FPT-complete problems are unlikely to see a significant decrease in “parameterized” time complexity when “parameterized” parallelism is allowed, under the assumption that FPP $\neq$ FPT. This subsection proves the existence of parameterized problems that are not in FPP (under the assumption FPP $\neq$ FPT) and whose underlying decision problems are P-complete, complementing ???, which proves the existence of parameterized problems in FPP whose underlying decision problems are P-complete. **TODO Describe why the assumption FPP $\neq$ FPT is reasonable.**

**Definition 6.1** (FPT-completeness). A parameterized problem $(Q, \kappa)$ is FPT-hard if for each parameterized problem $(R, \lambda)$, there is an FPP many-one reduction from $(R, \lambda)$ to $(Q, \kappa)$. If furthermore $(Q, \kappa)$ is in FPT, then it is FPT-complete.

**Theorem 6.2.** If an FPT-complete problem is in FPP, then FPP = FPT.

**Proof.** Follows from the downward closure of FPP under FPP many-one reductions. ☐
**TODO** Show negative consequences of $FPP = FPT$; \[5\] already basically did this, but not clearly.

The following problem is adapted from *Short Deterministic Turing Machine Computation* in \[4\].

**Definition 6.3** (*$p$-Bounded Halting Problem*).

- **Instance:** deterministic Turing machine $M$, binary string $x$ of length $n$, positive integer $t$ in unary, positive integer $c$.
- **Parameter:** $t/n^c$
- **Question:** Does $M$ accept $x$ within $t$ steps?

**Theorem 6.4.** *$p$-Bounded Halting Problem* is FPT-complete.

**Proof.** The underlying decision problem is in P (by a standard simulation on the deterministic universal Turing machine), so the parameterized problem is in FPT. To show FPT-hardness, we use a generic reduction. Let $(Q, \kappa)$ be an arbitrary parameterized problem in FPT and let $M$ be the deterministic Turing machine that decides $Q$ in time $f_M(k)n^c$ for some (circuit-)computable function $f_M$ and some positive integer $c$. The reduction is $x \mapsto (M, x, 1^{f_M(k)n^c}, c)$. This is computable by a (nonuniform) circuit family of constant depth and size $f(k)n^{O(1)}$, where $f$ is a circuit-computable function. The parameter of the reduced instance is $f_M(k)n^c/n^c$, or simply $f_M(k)$, which satisfies the parameter bound required by the definition of FPT reduction. Therefore we conclude that *$p$-Bounded Halting Problem* is FPT-complete. \qed

The following problem is a generalization of the problem BS-BD-CVP from \[5\].

**Definition 6.5** (*$p$-Small Circuit Evaluation*).

- **Instance:** single-output Boolean circuit $C$ on $n$ inputs, binary string $x$ of length $n$, positive integer $k$, positive integers $\alpha$ and $\beta$, multi-output Boolean circuits $f$ and $g$ on $\lceil \log k \rceil$ inputs with $\text{size}(C) \leq f(k)n^\alpha$ and $\text{depth}(C) \leq g(k)n^\beta$.
- **Parameter:** $k$
- **Question:** Does $C(x) = 1$?

This theorem is related to \[5\] Corollary 2, where the authors prove that the BS-BD-CVP problem is complete for the class PNC (a class that lives between FPP and FPT) under FPP many-one reductions. While their reduction is a generic reduction, ours is a reduction from the parameterized bounded halting problem.

**Theorem 6.6.** *$p$-Small Circuit Evaluation* is FPT-complete.

**Proof.** Membership in FPT is straightforward to prove: use the natural algorithm for evaluating a circuit which can be performed in linear time with respect to the size of the circuit. We must also compute $f(k)n^\alpha$ and $g(k)n^\beta$, and compare these with the size and depth of the circuit $C$. Both of these are polynomial-time
algorithms with respect to the size of the input, and hence the problem is in FPT.

Now we prove FPT-hardness. The reduction from p-BOUNDED HALTING PROBLEM is

\[(M, x, 1^t, c) \mapsto (C_M, x, \alpha, \beta, f, g, k),\]

where

- \(C_M\) is the standard circuit of size \(O(t^2)\) and depth \(O(t)\) simulating \(t\) steps of the action of \(M\) on inputs of length \(n\),
- \(\alpha = 2c\),
- \(\beta = c\),
- \(f\) is the function that squares the value of its input,
- \(g\) is the identity function,
- \(k = t/n^c\).

This reduction is computable in the appropriate size and depth bounds, and its correctness follows from the correctness of the standard deterministic Turing machine-to-circuit reduction. To check that the reduced instance is well-formed, let us verify that the circuit \(C_M\) meets the size and depth requirements. The size of \(C_M\) is \(O(t^2)\), which is \(O((t/n^c)n^c)^2\), or simply \(f(k)n^\alpha\). Similarly the depth of \(C_M\) is \(O(t)\), which is \(O((t/n^c)n^c)\), or simply \(g(k)n^\beta\). (There are some constants in the size and depth bounds that we have ignored, but those can be incorporated into the circuit-computable functions \(f\) and \(g\).) Finally, the parameter in the original instance, \(t/n^c\), is exactly the parameter of the reduced instance, so this reduction meets the necessary parameter bound. Therefore we have shown a correct FPP many-one reduction from an FPT-complete problem.

As expected, if any of these FPT-complete problems are fixed-parameter parallelizable, then FPP = FPT.

It seems that most P-complete problems will end up being FPT-complete under this notion of completeness. This doesn’t really help us distinguish between different P-complete problems based on how fixed-parameter parallelizable they are. In the next subsection, we try a different approach.

### 6.2 Parameterized complexity of efficient verification

Consider the parameterized weighted circuit satisfiability and circuit evaluation problems.

**Definition 6.7 (p-Weighted Circuit Satisfiability).**

**Instance:** Boolean circuit \(C\), natural number \(k\).

**Parameter:** \(k\).

**Question:** Is there binary string \(x\) such that the Hamming weight of \(x\) is \(k\) and \(C(x) = 1\)?
The corresponding parameterized weighted circuit evaluation problem would then be as follows. Let \( \|x\|_1 \) denote the Hamming weight (that is, the number of ones) in \( x \).

**Definition 6.8 (\( p \)-Circuit \( k \)-Evaluation).**

- **Instance:** Boolean circuit \( C \), binary string \( x \), natural number \( k \).
- **Parameter:** \( k \).
- **Question:** Does \( \|x\|_1 = k \) and \( C(x) = 1 \)?

This is a little silly, since the Hamming weight of \( x \) can be computed trivially, but it is technically “verification” problem corresponding to the satisfiability problem above. Instead, we use a problem that is equivalent but a little less silly.

**Definition 6.9 (\( p \)-Weighted Circuit Evaluation).**

- **Instance:** Boolean circuit \( C \), binary string \( x \).
- **Parameter:** \( \|x\|_1 \).
- **Question:** Does \( C(x) = 1 \)?

These parameterized problems are equivalent under FPP many-one reductions because the parameterization is computable in NC. TODO Reference for circuit computing Hamming weight; see [http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4781532](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4781532)

**Proposition 6.10.** \( p \)-Circuit \( k \)-Evaluation \( \equiv_{\text{FPP}}^{\text{m}} \) \( p \)-Weighted Circuit Evaluation.

In the setting of decision problems, we know that \( \text{NP} \) can be characterized as the closure of the circuit satisfiability problem under polynomial-time many-one reductions,

\[
\text{NP} = [\text{Circuit Satisfiability}]^{\leq_{\text{m}}^p},
\]

and \( \text{P} \) as the closure of the circuit evaluation problem under \( \text{NC}^1 \) many-one reductions,

\[
\text{P} = [\text{Circuit Evaluation}]^{\leq_{\text{m}}^{\text{NC}^1}}.
\]

In the setting of parameterized problems, the class \( \text{W}[\text{P}] \) can be characterized as the closure of the parameterized weighted circuit satisfiability problem under fixed-parameter tractable many-one reductions,

\[
\text{W}[\text{P}] = [\text{p-Weighted Circuit Satisfiability}]^{\leq_{\text{m}}^{\text{FPT}}}.\]

Following the above pattern, we define a new class as the closure of the parameterized weighted circuit evaluation problem under fixed-parameter parallelizable many-one reductions,

\[
\text{E}[\text{P}] = [\text{p-Weighted Circuit Evaluation}]^{\leq_{\text{m}}^{\text{FPP}}}.\]

("E" for evaluation).

Since the underlying decision problem, the problem of evaluating a circuit on a given input, is in \( \text{P} \), the parameterized problem \( \text{p-Weighted Circuit Evaluation} \) is trivially in \( \text{FPT} \). Since \( \text{FPP} \) reductions compose, \( \text{FPP} \) is a subset of \( \text{FPT} \), and \( \text{FPT} \) is closed under \( \text{FPT} \) reductions, we conclude that \( \text{E}[\text{P}] \) is a subset of \( \text{FPT} \).
Theorem 6.11. $E[P] \subseteq \text{FPT}$.

Is $E[P] = \text{FPT}$? The standard simulation of a deterministic Turing machine by a circuit, as in [12], for example, fails to provide a FPP many-one reduction to the parameterized weighted circuit evaluation problem, since the natural reduction would be of the form $x \mapsto (C, x, \|x\|_1)$, but the parameter value $\|x\|_1$ is not necessarily bounded by a function of $\kappa(x)$. TODO Show something bad happens if $E[P] = \text{FPT}$.

TODO Show a problem that is known to be in FPT (for example one of the deterministic Turing machine computation problems maybe) is actually in $E[P]$.

The same issue prevents us from showing that $\text{FPP} \subseteq E[P]$.

References


17

