Highly parallel approximations for inherently sequential problems

Jeffrey Finkelstein
Computer Science Department, Boston University

May 5, 2016

1 Introduction

In this work we study classes of optimization problems that require inherently sequential algorithms to solve exactly but permit highly parallel algorithms for approximation solutions. \( \text{NC} \) is the class of computational problems decidable by a logarithmic space uniform family of Boolean circuits of bounded fan-in, polynomial size, and polylogarithmic depth. Such problems are considered both “efficient” (since \( \text{NC} \subseteq \text{P} \)) and “highly parallel” (since we might consider each gate in the circuit to be a processor working in parallel and the small depth of the circuit a small number of steps). By contrast, problems which are \( \text{P} \)-complete (under logarithmic space or even \( \text{NC} \) many-one reductions) are considered “inherently sequential”. Furthermore, all \( \text{NP} \)-hard and \( \text{PSPACE} \)-hard problems are also inherently sequential, since \( \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \). Just as we hope to find efficient approximation algorithms for optimization problems for which it is intractable to compute an exact solution, so too do we hope to find efficient and highly parallel approximation algorithms for optimization problems for which computing the exact solution is inherently sequential. (However, just like hardness of approximation for \( \text{NP} \)-hard problems, in some cases even approximating a solution is inherently sequential!)

2 Definitions

Throughout this work, \( \Sigma = \{0, 1\} \) and inputs and outputs are encoded in binary. The set of all finite strings is denoted \( \Sigma^* \), and for each \( x \in \Sigma^* \), we denote the
length of $x$ by $|x|$. We denote the set of all polynomials by $\text{poly}$ and the set of all polylogarithmic functions by $\text{polylog}$. The set of integers is denoted $\mathbb{Z}$, the set of rationals $\mathbb{Q}$, and their positive subsets $\mathbb{Z}^+$ and $\mathbb{Q}^+$. The natural numbers, defined as $\mathbb{Z}^+ \cup \{0\}$, is denoted $\mathbb{N}$. Vectors are formatted in bold face, like $x$.

The all-ones vector is denoted $\mathbf{1}$.

2.1 Optimization problems and approximation algorithms

We adapt the definitions of [26] from logarithmic space approximability to $\text{NC}$ approximability.

**Definition 2.1** ([2]). An optimization problem is a four-tuple, $(I,S,m,t)$, where the set $I \subseteq \Sigma^*$ is called the instance set, the set $S \subseteq I \times \Sigma^*$ is called the solution relation, the function $m: S \rightarrow \mathbb{Z}^+$ is called the measure function, and $t \in \{\text{min}, \text{max}\}$ is called the type of the optimization.

An optimization problem in which the measure function has rational values can be transformed into one in which the measure function has integer values [2, Page 23] (for example, by expressing each measure as the integer numerator of the equivalent rational number whose denominator is the greatest common divisor of all the values $m(x,y)$ for each fixed $x$).

**Definition 2.2** ([26]). Let $P$ be an optimization problem, so $P = (I,S,m,t)$, and let $x \in I$.

1. Let $S(x) = \{y \in \Sigma^* | (x,y) \in S\}$; we call this the solutions for $x$.

2. Define $m^*(x)$ by

$$m^*(x) = \begin{cases} \min \{m(x,y) | y \in S(x)\} & \text{if } t = \text{min} \\ \max \{m(x,y) | y \in S(x)\} & \text{if } t = \text{max} \end{cases}$$

for all $x \in \Sigma^*$; we call this the optimal measure for $x$. Let $m^*(x)$ be undefined if $S(x) = \emptyset$.

3. Let $S^*(x) = \{y \in \Sigma^* | m(x,y) = m^*(x)\}$; we call this the set of optimal solutions for $x$.

4. Let $R(x,y) = \max \left( \frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)} \right)$; we call this the performance ratio of the solution $y$.

5. Let $P_3 = \{x \in \Sigma^* | S(x) \neq \emptyset\}$; we call this the existence problem.

6. Let $P_{\text{opt}<} = \{(x,z) \in P_3 \times \mathbb{N} | \exists y \in \Sigma^* : m(x,y) < z\}$ and $P_{\text{opt}>} = \{(x,z) \in P_3 \times \mathbb{N} | \exists y \in \Sigma^* : m(x,y) > z\}$; we call these the budget problems.
7. Let \( f : \Sigma^* \rightarrow \Sigma^* \). We say \( f \) produces solutions for \( P \) if for all \( x \in P_3 \) we have \( f(x) \in S(x) \). We say \( f \) produces optimal solutions for \( P \) if for all \( x \in P_3 \) we have \( f(x) \in S^*(x) \).

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 2.3.** Let \( P \) be an optimization problem, let \( r : \mathbb{N} \rightarrow \mathbb{Q}^+ \), and let \( f : I \rightarrow \Sigma^* \). We say \( f \) is an \( r \)-approximator for \( P \) if it produces solutions for \( P \) and \( R(x, f(x)) \leq r(|x|) \) for all \( x \in P_3 \).

If \( r \) is the constant function with value \( \delta \), we simply say \( f \) is a \( \delta \)-approximator for \( P \).

**Definition 2.4.** Let \( P \) be an optimization problem and let \( f : I \times \mathbb{N} \rightarrow \Sigma^* \). We say \( f \) is an approximation scheme for \( P \) if for all \( x \in P_3 \) and all positive integers \( k \) we have \( f(x, k) \in S(x) \) and \( R(x, f(x, k)) \leq 1 + \frac{1}{k} \).

### 2.2 Classes of optimization problems

The study of efficient approximations for intractable problems begins with the following definition of \( \text{NP} \) optimization problems. We will adapt this definition to explore efficient and highly parallel approximations for inherently sequential problems.

**Definition 2.5.** The complexity class \( \text{NPO} \) is the class of all optimization problems \((I, S, m, t)\) such that the following conditions hold.

1. The instance set \( I \) is decidable by a deterministic polynomial time Turing machine.
2. The solution relation \( S \) is decidable by a deterministic polynomial time Turing machine and is polynomially bounded (that is, the length of \( y \) is bounded by a polynomial in the length of \( x \) for all \((x, y) \in S\)).
3. The measure function \( m \) is computable by a deterministic polynomial time Turing machine.

The second condition is the most important in this definition; it is the analog of polynomial time verifiability in \( \text{NP} \).

**Definition 2.6.** The complexity class \( \text{PO} \) is the subclass of \( \text{NPO} \) in which for each optimization problem \( P \) there exists a function \( f \) in \( \text{FP} \) that produces optimal solutions for \( P \).

We now wish to translate these definitions to the setting of efficient and highly parallel verifiability. In order to take advantage of results and techniques from the study of \( \text{NPO} \) and \( \text{PO} \), we will start by considering a model of computation in which we allow highly parallel computation access to a polynomial amount of nondeterminism. First we define the necessary circuit classes, then we define the corresponding classes of optimization problems.
Definition 2.7.

1. NC is the class of decision problems decidable by a logarithmic space uniform family of Boolean circuits with polynomial size, polylogarithmic depth, and fan-in two.

2. FNC is the class of functions \( f \) computable by an NC circuit in which the output of the circuit is (the binary encoding of) \( f(x) \).

3. \( \text{NNC}(f(n)) \) is the class of languages computable by a logarithmic space uniform NC circuit family augmented with \( O(f(n)) \) nondeterministic gates for each input length \( n \) \(^{29}\). A nondeterministic gate takes no inputs and yields a single (nondeterministic) output bit.

If \( F \) is a class of functions, then \( \text{NNC}(F) = \bigcup_{f \in F} \text{NNC}(f(n)) \).

\( \text{NNC}(\text{poly}) \), also known as \( \text{GC}(\text{poly}, \text{NC}) \) \(^4\) and \( \beta P \) \(^{19} \), is an unusual class which may warrant some further explanation. NC has the same relationship to \( \text{NNC}(\text{poly}) \) as P does to NP (thus an equivalent definition of \( \text{NNC}(\text{poly}) \) is one in which each language has an efficient and highly parallel verification procedure; as in the definition of \( \text{NPO} \) in Definition 2.5, it is this formulation which we use when defining \( \text{NNCO} \) in Definition 2.8). Wolf \(^{29} \) notes that \( \text{NNC} (\log n) = \text{NC} \) and \( \text{NNC} (\text{poly}) = \text{NP} \), and suggests that \( \text{NNC} (\text{polylog}) \) may be an interesting intermediary class, possibly incomparable with P. Cai and Chen \(^4\) prove that for each natural number \( k \) and \( i \), there is a complete problem for \( \text{NNC}^k (\log^i n) \) under logarithmic space many-one reductions.

Definition 2.8. The complexity class \( \text{NNCO}(\text{poly}) \) is the class of all optimization problems \( (I, S, m, t) \) such that the following conditions hold.

1. The instance set \( I \) is decidable by an NC circuit family.

2. The solution relation \( S \) is decidable by an NC circuit family and is polynomially bounded (that is, the length of \( y \) is bounded by a polynomial in the length of \( x \) for all \((x, y) \in S\)).

3. The measure function \( m \) is computable by an FNC circuit family.

For the sake of brevity, we write \( \text{NNCO} \) instead of \( \text{NNCO}(\text{poly}) \).

We can now proceed to define classes of approximable optimization problems contained in \( \text{NNCO} \).

Definition 2.9. Suppose \( P \) is an optimization problem in \( \text{NNCO} \).

1. \( P \in \text{ApxNCO} \) if there is an \( r \)-approximator in FNC for \( P \), where \( r(n) \in O(1) \) for all \( n \in \mathbb{N} \).

2. \( P \in \text{NCAS} \) if there is an approximation scheme \( f \) for \( P \) such that \( f_k \in \text{FNC} \) for each \( k \in \mathbb{N} \), where \( f_k(x) = f(x,k) \) for all \( x \in \Sigma^* \).
3. $P \in \text{FNCAS}$ if there is an approximation scheme $f$ for $P$ such that $f \in \text{FNC}$ in the sense that the size of the circuit is polynomial in both $|x|$ and $k$ and the depth of the circuit is polylogarithmic in both $|x|$ and $k$.

4. $P \in \text{NCO}$ if there is a function $f$ in $\text{FNC}$ that produces optimal solutions for $P$.

For the NC approximation classes defined above, it is crucial that the solution relation is verifiable in NC. In all previous works (for example, [10, 24]), the implicit definition of, say, $\text{NCX}$, which corresponds to our class $\text{ApxNCO}$, requires only that the solution relation is verifiable in polynomial time. We believe we are the first to make this important distinction; some solution relations are harder to verify than others.

Each of the classes in [Definition 2.9] includes the one defined below it. This chain of inclusions provides a hierarchy that classifies approximability of problems in $\text{NNCO}$, and hence in $\text{NPO}$. However, our intention is to determine the approximability of optimization problems corresponding to P-complete decision problems, not those corresponding to NP-complete decision problems. Therefore we consider the classes $\text{PO} \cap \text{NNCO}$, $\text{PO} \cap \text{ApxNCO}$, etc. in order to more accurately capture the notion of highly parallel approximability of inherently sequential problems. The instance set, solution relation, and measure function of optimization problems in these classes are computable in NC, and furthermore, there is a polynomial time algorithm that produces optimal solutions.

Figure 1 shows the inclusions among some of the complexity classes defined in this section.

### 2.3 Reductions among approximation problems

There are many reductions for approximation problems; nine of them are defined in a survey paper by Crescenzi [5], and there are more defined elsewhere. We will use a logarithmic space-bounded version of the “AP reduction”, considered by approximation experts to be a reasonable reduction to use when constructing complete problems [5, Section 2] [2, Section 8.6]. Although the original definition is from [9, Definition 9] (an preliminary version of [8, Definition 2.5]), the definition here is from [2, Definition 8.3].

**Definition 2.10.** [2, Definition 8.3] Let $P$ and $Q$ be optimization problems in $\text{NNCO}$, with $P = (I_P, S_P, m_P, t_P)$ and $Q = (I_Q, S_Q, m_Q, t_Q)$. We say $P$ AP reduces to $Q$ and write $P \leq_{AP} Q$ if there are functions $f$ and $g$ and a constant $\alpha \in \mathbb{R} \cap [1, \infty)$ such that

1. for all $x \in I_P$ and all $r \in Q \cap (1, \infty)$, we have $f(x, r) \in I_Q$,
2. for all $x \in I_P$ and all $r \in Q \cap (1, \infty)$, if $S_P(x) \neq \emptyset$ then $S_Q(f(x, r)) \neq \emptyset$,
3. for all $x \in I_P$, all $r \in Q \cap (1, \infty)$, and all $y \in S_Q(f(x, r))$, we have $g(x, y, r) \in S_P(x)$,
4. $f$ and $g$ are computable in logarithmic space for any fixed $r$, and
Figure 1: The structure of classes of optimization problems that have both efficiently computable exact solutions and highly parallel approximate solutions, the subclasses of $\text{PO} \cap \text{NNCO}$. Many of these inclusions are strict under some widely-held complexity-theoretic assumptions; see subsection 3.4.

\[
\begin{align*}
\text{NPO} & \quad \uparrow \\
\text{ApxPO} & \quad \uparrow \\
\text{PTAS} & \quad \downarrow \quad \text{NNCO} \\
\text{PO} & \quad \downarrow \quad \text{ApxNCO} \\
\text{PO} \cap \text{NNCO} & \quad \downarrow \\
\text{PO} \cap \text{ApxNCO} & \\
\text{PO} \cap \text{NCAS} & \\
\text{NCO} & 
\end{align*}
\]
5. for all \( x \in I_P \), all \( r > 1 \), and all \( y \in S_Q(f(x,r)) \),

\[
R_Q(f(x,r),y) \leq r \implies R_P(x,g(x,y,r)) \leq 1 + \alpha(r-1).
\]

For a class \( \mathcal{C} \) of optimization problems, we say a problem \( Q \) is hard for \( \mathcal{C} \) if for all problems \( P \) in \( \mathcal{C} \) there is a logarithmic space AP reduction from \( P \) to \( Q \). If furthermore \( Q \) is in \( \mathcal{C} \) we say \( Q \) is complete for \( \mathcal{C} \).

3 Completeness and collapses in classes of approximable optimization problems

3.1 Completeness in classes of inapproximable problems

This section shows that Maximum Variable-Weighted Satisfiability is complete for NNCO and Maximum Weighted Circuit Satisfiability is complete for NPO. Furthermore, the latter problem is not in NNCO unless \( NC = P \). Thus there are optimization problems whose corresponding budget problems are of equal computational complexity—they are both \( NP \)-complete—but whose solution relations are of different computational complexity, under reasonable complexity theoretic assumptions.

**Definition 3.1** (Maximum Variable-Weighted Satisfiability).

**Instance:** Boolean formula \( \phi \) on variables \( x_1, \ldots, x_n \), weights in \( Q^+ \) for each variable \( w_1, \ldots, w_n \).

**Solution:** assignment \( \alpha \) to the variables that satisfies \( \phi \).

**Measure:** \( \max(1, \Sigma_{i=1}^n \alpha(x_i)w_i) \).

**Type:** maximization.

**Definition 3.2** (Maximum Weighted Circuit Satisfiability).

**Instance:** Boolean circuit \( C \) with inputs \( x_1, \ldots, x_n \), weights in \( Q^+ \) for each input \( w_1, \ldots, w_n \).

**Solution:** assignment \( \alpha \) to the inputs such that \( C(\alpha(x_1), \ldots, \alpha(x_n)) \) outputs 1.

**Measure:** \( \max(1, \Sigma_{i=1}^n \alpha(x_i)w_i) \).

**Type:** maximization.

**Theorem 3.3.** Maximum Variable-Weighted Satisfiability is complete for NNCO under logarithmic space AP reductions.

**Proof.** This problem is complete for the class of maximization problems in NPO under polynomial time AP reductions [22, Theorem 3.1]. A close inspection reveals that the functions of the reduction can be computed in logarithmic space. There is furthermore a polynomial time AP reduction from the Minimum Variable-Weighted Satisfiability problem, which is complete for the class of all minimization problems in NPO, to Maximum Variable-Weighted Satisfiability [2, Theorem 8.4], and a close inspection of the reduction reveals
that it can also be implemented in logarithmic space. Thus this problem is complete for \textbf{NPO} under logarithmic space AP reductions.

Next, we show that \textsc{Maximum Variable-Weighted Satisfiability} is in \textbf{NNCO}. The measure function is computable in \textbf{FNC} because the basic arithmetic operations and summation are both computable in \textbf{FNC}. The solution set is decidable in \textbf{NC} because Boolean formula evaluation is computable in \textbf{NC} \cite{3}. Since \textbf{NNCO} $\subseteq$ \textbf{NPO} we conclude that the problem is complete for \textbf{NNCO}.

By converting a Boolean formula into its equivalent Boolean circuit, we get the following corollary.

\textbf{Corollary 3.4.} \textsc{Maximum Weighted Circuit Satisfiability} is complete for \textbf{NPO} under logarithmic space AP reductions.

Sam H. suggested an initial version of the following theorem.

\textbf{Theorem 3.5.} \textbf{NNCO} = \textbf{NPO} if and only if \textbf{NC} = \textbf{P}.

\textit{Proof.} \textbf{NNCO} $\subseteq$ \textbf{NPO} by definition. If \textbf{NC} = \textbf{P}, then \textbf{NNCO} = \textbf{NPO} by definition. If \textbf{NNCO} = \textbf{NPO}, then \textsc{Maximum Weighted Circuit Satisfiability} is in \textbf{NNCO}, thus there is an \textbf{NC} algorithm that decides its solution relation. Its solution relation is precisely the \textsc{Circuit Value} problem, which is \textbf{P}-complete \cite{15, Problem A.1.1}. An \textbf{NC} algorithm for a \textbf{P}-complete decision problem implies \textbf{NC} = \textbf{P}.

Contrast this with the fact that \textbf{NNC}(\text{poly}) = \textbf{NP} \cite{29, Theorem 2.2} (and in fact, \textbf{NNC}$_1$(\text{poly}) = \textbf{NP}). So the classes of decision problems are equal whereas the classes of corresponding optimization problems are not, unless \textbf{NC} = \textbf{P}.

\subsection{Completeness in classes of polynomial time solvable problems}

This section shows results nearly analogous to those in the previous section, but in the intersection of both \textbf{NPO} and \textbf{NNCO} with \textbf{PO}. The results here show completeness with respect to maximization problems only; we conjecture that both of the problems defined below are also complete with respect to minimization problems.

\textbf{Definition 3.6 (Maximum Double Circuit Value).}

\begin{itemize}
  \item \textbf{Instance:} two Boolean circuits $C_1$ and $C_2$, binary string $x$.
  \item \textbf{Solution:} binary string $y$ such that $C_1(x) = y$ and $|x| + |y|$ equals the number of inputs to $C_2$.
  \item \textbf{Measure:} $\max(1, C_2(x, y))$.
  \item \textbf{Type:} maximization.
\end{itemize}

In this problem, the Boolean circuits may output binary strings of polynomial length interpreted as non-negative integers. This problem is constructed so that the circuit $C_1$ can simulate an algorithm that produces an optimal solution for an optimization problem and the circuit $C_2$ can simulate an algorithm that outputs
the measure of a solution for that problem. Also, each input has exactly one solution, so this problem is quite artificial.

**Definition 3.7 (Linear Programming).**

**Instance:** $m \times n$ integer matrix $A$, integer vector $b$ of length $m$, integer vector $c$ of length $n$.

**Solution:** non-negative rational vector $x$ of length $n$ such that $Ax \leq b$.

**Measure:** $\max(1, c^T x)$.

**Type:** maximization.

**Theorem 3.8.** Maximum Double Circuit Value is complete for the class of maximization problems in PO under logarithmic space AP reductions.

**Proof.** Since Circuit Value is in P, both the solution and the measure function are computable in polynomial time. Therefore Maximum Double Circuit Value is in PO. Our goal is now to exhibit an AP reduction from any language in PO to Maximum Double Circuit Value. For the sake of brevity, suppose Maximum Double Circuit Value is defined by $(I_C, S_C, m_C, \max)$.

Let $P$ be a maximization problem in PO, where $P = (I_P, S_P, m_P, \max)$. Let $x$ be an element of $I_P$. Suppose $E$ is the deterministic polynomial time Turing machine that produces optimal solutions for $P$. Define $f$ by $f(x) = (C_E, C_m, x)$ for all $x \in I_P$, where $C_E$ is the Boolean circuit of polynomial size that simulates the action of $E$ on input $x$ and $C_m$ is the circuit that simulates $m_P$ on inputs $x$ and $E(x)$. These circuits exist and are computable from $x$ in logarithmic space [20]. Define $g$ by $g(x, y) = y$ for all strings $x$ and all $y$ in $S_C(f(x))$. Let $\alpha = 1$.

Now, for any $x \in I_P$ and any $y \in S_C(f(x))$, we have

$$m_P(x, g(x, y)) = m_P(x, y) = C_m(x, y) = m_C((C_E, C_m, x), y) = m_C(f(x), y).$$

Since these measures are equal for all instances $x$ and solutions $y$, we have shown that $(f, g, \alpha)$ is a logarithmic space AP reduction from $P$ to Maximum Double Circuit Value. \qed

**Theorem 3.9.** Linear Programming is complete for PO under logarithmic space AP reductions.

**Proof.** Linear Programming is in PO by the ellipsoid algorithm [17]. We reduce Maximum Double Circuit Value to Linear Programming. The reduction is essentially the same as the reduction from Circuit Value to Linear Programming given (implicitly) in the hint beneath [15, Problem A.4.1]. We repeat it here for the sake of completeness.

Define the instance transducer $f$ as follows. Suppose $(C_1, C_2, x)$ is an instance of Maximum Double Circuit Value, and let $x = x_1 \cdots x_n$. For each of the circuits $C_1$ and $C_2$, the transducer $f$ adds the following inequalities to the linear program.

1. For each bit of $x$, represent a 1 bit at index $i$ by $x_i = 1$ and a 0 bit by $x_i = 0$. 

9
2. Represent a NOT gate, \( g = \neg h \), by the equation \( g = 1 - h \) and the inequality \( 0 \leq g \leq 1 \).

3. Represent an AND gate, \( g = h_1 \land h_2 \), by the inequalities \( g \leq h_1 \), \( g \leq h_2 \), \( h_1 + h_2 - 1 \leq g \), and \( 0 \leq g \leq 1 \).

4. Represent an OR gate, \( g = h_1 \lor h_2 \), by the inequalities \( h_1 \leq g \), \( h_2 \leq g \), \( g \leq h_1 + h_2 \), and \( 0 \leq g \leq 1 \).

Suppose \( y_1, \ldots, y_s \) are the variables corresponding to the output gates of \( C_1 \), and suppose \( \mu_t, \ldots, \mu_1 \) are the variables corresponding to the output gates of \( C_2 \), numbered from least significant bit to most significant bit (that is, right-to-left). The components of the object function \( c \) are assigned to be \( 2^i \) where the component corresponds to the variable \( \mu_i \) and 0 everywhere else. The function \( f \) is computable in logarithmic space because the transformation can proceed gatewise, requiring only a logarithmic number of bits to record the index of the current gate. Suppose \( x \) is a solution to \( f((C_1, C_2, x)) \), that is, an assignment to the variables described above that satisfies all the inequalities. Define the solution transducer \( g \) by \( g((C_1, C_2, x), x) = y \), where \( y = y_1 \cdots y_s \). This is also computable in logarithmic space by finding the index, in binary, of the necessary gates \( y_1, \ldots, y_s \). Let \( \alpha = 1 \).

By structural induction on the gates of the circuits we see that a gate has value 1 on input \( x \) if and only if the solution vector \( x \) has a value 1 in the corresponding component, and \( x \) must be a vector over \( \{0, 1\} \). Since the linear program correctly simulates the circuits, we see that

\[
m_A((C_1, C_2, x), g((C_1, C_2, x), x)) = m_A((C_1, C_2, x), y) = C_2(x, y) = \mu_t \cdots \mu_1 = \sum_{i=1}^{t} 2^i \mu_i = m_B(f((C_1, C_2, x), x)),
\]

where \( m_A \) is the measure function for \textsc{Maximum Double Circuit Value} and \( m_B \) is the measure function for \textsc{Linear Programming}. Since these measures are equal, we have shown that \( (f, g, \alpha) \) is a logarithmic space AP reduction from \textsc{Maximum Double Circuit Value} to \textsc{Linear Programming}. Since the former is complete for the class of maximization problems in \( \text{PO} \), so is \textsc{Linear Programming}.

The reduction in the proof of \textbf{Theorem 3.9} is more evidence that approximability is not closely related to the complexity of verification. Although \textsc{Maximum Double Circuit Value} is not in \( \text{PO} \cap \text{NNCO} \) unless \( \text{NC} = \text{P} \) (because its solution relation is \( \text{P}\)-complete), \textsc{Linear Programming} is not only in \( \text{PO} \) but also in \( \text{NNCO} \), since matrix multiplication is in \( \text{NC} \). This yields the following corollaries.

\textbf{Corollary 3.10.} \( \text{PO} \cap \text{NNCO} \) is not closed under logarithmic space AP reductions unless \( \text{NC} = \text{P} \).
**Corollary 3.11.** **Linear Programming** is complete for the class of maximization problems in \( PO \cap NNCO \) under logarithmic space \( AP \) reductions.

An equivalence analogous to that of Theorem 3.5 also holds in the intersection with \( PO \).

**Theorem 3.12.** \( PO \cap NNCO = PO \) if and only if \( NC = P \).

**Proof.** \( PO \cap NNCO \subseteq PO \) by definition. If \( NC = P \), then \( PO \cap NNCO = PO \) by definition. If \( PO \cap NNCO = PO \), then **Maximum Double Circuit Value** is in \( PO \cap NNCO \), thus there is an \( NC \) algorithm that decides its solution relation. Its solution relation is a generalization of the **Circuit Value** problem, which is \( P \)-complete (as long as the length of the output \( y \) remains polynomial in the length of the input, this generalization remains \( P \)-complete). An \( NC \) algorithm for a \( P \)-complete decision problem implies \( NC = P \).

### 3.3 Completeness in classes of approximable problems

In order to construct an optimization problem complete for, say, \( PO \cap ApxNCO \), we need to use either

1. an analog of the PCP theorem with \( NC \) verifiers for polynomial time decision problems, or
2. a canonical, “universal” complete problem for \( PO \cap ApxNCO \).

These are the only two known ways for showing completeness in constant-factor approximation classes. The first approach is difficult to apply because it is not obvious how to construct a PCP for a deterministic time complexity class \( (PO) \). See [13] for more information on that approach. The second approach is difficult to apply because although this technique has worked in the past for constructing a complete problem for \( ApxPO \) [6, Lemma 2], it is not clear how to guarantee a polynomial time computable function that produces optimal solutions for such a problem.

However, we know what a problem complete for \( PO \cap ApxNCO \) should look like. It should be exactly solvable in polynomial time and admit an \( NC \) approximation algorithm. It should also have threshold behavior in the following sense. If the problem were approximable for all \( r > 1 \), then it would be in \( NCAS \). If the problem were not approximable for any \( r \geq 1 \), then it would not even be in \( ApxNCO \). Therefore there should be some constant \( r_0 \) such that the problem is approximable for all \( r \in (r_0, \infty) \) and not approximable for all \( r \in (1, r_0) \).

#### 3.3.1 High weight subgraph problems

There is in fact a family of maximization problems that has these properties: **Induced Subgraph of High Weight for Linear Extremal Properties** [10, Chapter 3]. A concrete example of a maximization problem in this family is **Maximum High Degree Subgraph**. In the definition below, the degree of a graph \( G \), denoted \( \deg(G) \), is defined by \( \deg(G) = \min_{v \in V(G)} \deg(v) \).
**Definition 3.13** (Maximum High Degree Subgraph).

**Instance:** undirected graph $G$.

**Solution:** vertex-induced subgraph $H$.

**Measure:** $\deg(H)$.

**Type:** maximization.

This problem is exactly solvable in polynomial time, has an $r$-approximator in $\text{FNC}$ for all $r \in (2, \infty)$ and has no $r$-approximator in $\text{FNC}$ for all $r \in (1, 2)$ unless $\text{NC} = \text{P}$ [1]. (The existence or non-existence of a 2-approximator seems to remain unknown.) We suspect this family of problems is complete for $\text{PO} \cap \text{ApxNCO}$ under logarithmic space AP reductions.

**Conjecture 3.14.** Induced Subgraph of High Weight for Linear Extremal Properties is complete for the class of maximization problems in $\text{PO} \cap \text{ApxNCO}$ under logarithmic space AP reductions.

### 3.3.2 Restrictions of linear programming

Although **Linear Programming** is $\text{P}$-complete and admits no $\text{NC}$ approximation algorithm, **Positive Linear Programming**, the restriction of **Linear Programming** to inputs in which all entries of $A$, $b$, and $c$ are non-negative, admits a $\text{NC}$ approximation scheme [21], even though the corresponding budget problem remains $\text{P}$-complete [28, Theorem 4]. These results beg the question “is there some restriction of **Linear Programming** less strict than **Positive Linear Programming** that exhibits the properties of a complete problem for $\text{PO} \cap \text{ApxNCO}$ as defined above?” If we relax the non-negativity requirement and allow a small number of equality constraints that can be violated by a small amount, then there is still an $\text{NC}$ approximation scheme [28, Theorem 5.2]. On the other hand, if we have even just one equality constraint and don’t allow the equality violations, the problem becomes hard to approximate again (to within a constant factor) [12, Theorem 3.1] [28, Remark 2]. Similarly, if we allow $A$ to have negative entries, the problem is hard to approximate [11, Corollary 2]. The $(\gamma, \kappa)$ form of **Linear Programming** is hard to approximate [11, Proposition 1]; the $k$-normal form reduces to **Positive Linear Programming** and so has an approximation scheme [27, Theorem 2].

There remains one candidate restriction that may have the properties we seek.

**Definition 3.15** (Linear Programming with Triplets).
Instance: $m \times n$ Boolean matrices $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$, non-negative rational vectors $b^{(1)}$ and $b^{(2)}$ of length $m$, non-negative rational vector $c$ of length $n$, and set of triples of indices $T \subseteq \{1, \ldots, n\}^3$ satisfying

1. there is at least one non-zero entry in each row of $A^{(1)}$ and each row of $A^{(2)}$, and

2. there is a constant $\gamma \in (1, \infty)$ such that $M^* \leq \gamma M$ for all measures $M$ of this instance, where $M^*$ is the optimal measure of the instance.

Solution: non-negative rational vectors $x$ and $f$ of length $n$ satisfying the inequalities

$$
\begin{align*}
  x_k + (1 - f_i) &\leq 1 \\
  x_k + (1 - f_j) &\leq 1 \\
  f_k + f_i + f_j &\leq 2
\end{align*}
$$

for all $(i, j, k) \in T$

$$
\begin{align*}
  A^{(1)}x &= b^{(1)} \\
  A^{(2)}x + A^{(3)}f &= b^{(2)}
\end{align*}

\begin{align*}
  x &\leq 1 \\
  f &\leq 1
\end{align*}

Measure: $\max(1, c^T f)$.

Type: maximization.

Although this optimization problem is not approximable within $\frac{2n^\epsilon}{N^\epsilon+1}$ for any positive $\epsilon$ unless $\text{NC} = \text{P}$ [25, Corollary 1], we know the value of the optimal measure of an instance to within a multiplicative factor of $\gamma$.

Conjecture 3.16. Linear Programming with Triplets is complete for the class of maximization problems in $\text{PO} \cap \text{ApxNCO}$ under logarithmic space $\text{AP}$ reductions.

3.3.3 Linear program for high degree subgraph

Perhaps we can consider a linear programming relaxation of Maximum High Degree Subgraph and show that such a restriction is more approximable than Linear Programming but less approximable than Positive Linear Programming.

Suppose the vertices of a graph $G$ are identified with the integers $\{1, \ldots, n\}$. We can represent a subgraph $H$ of a graph as a subset of the vertices, and if the graph has $n$ vertices, this can be an indicator vector $x$ of length $n$. Since we also want to maximize the minimum degree of the chosen subgraph, we introduce a new variable $d$ that has a value between 0 and $n$ that will be bounded above by
the degree of each vertex. We want the constraints to reflect that if a vertex is in the subgraph, then the degree of that vertex (with respect to the subgraph) is at least \(d\), and if a vertex is not in the subgraph then we don’t care about its degree. In other words, we want “if \(x_i = 1\) then \(\text{deg}(i) \geq d\),” where \(\text{deg}(i) = \sum_{j=1}^{n} a_{ij} x_j\) and \(a_{ij}\) is the entry at row \(i\), column \(j\) in the adjacency matrix of the graph. Equivalently, we want “\(x_i \neq 1\) or \(\text{deg}(i) \geq d\)”, or more specifically, “\(x_i < 1\) or \(\text{deg}(i) \geq d\)”. We can combine the two inequalities to get a single constraint “\(x_i + d \leq 1 + \text{deg}(x_i)\)”. However, as stated above we want this constraint to be always satisfied if \(x_i = 0\) (that is, when the vertex \(i\) is not in the subgraph \(H\)); in this form, that is not always true. We can assume without loss of generality that \(d\), the minimum degree of \(H\), will always be less than or equal to \(n - 1\) (since a graph without self-loops cannot have a vertex of degree \(n\) anyway). Thus we can ensure the constraint is always satisfied if we modify it so that \(x_i = 0\) implies \(d\) is less than the right side, “\(nx_i + d \leq n + \text{deg}(x_i)\)”. Now if \(x_i = 1\) then \(d \leq \text{deg}(x_i)\) as required, and if \(x_i = 0\) then \(d \leq n - 1 \leq n \leq n + \text{deg}(x_i)\) is always satisfied.

There is one final restriction: we require a non-empty subgraph, so we want at least one of the entries of \(x\) to be 1. Therefore, the proposed linear program is

\[
\begin{align*}
\text{maximize} & \quad d \\
\text{subject to} & \quad nx + d1 \leq n1 + Ax \\
 & \quad 1^T x > 0 \\
 & \quad 0 \leq x \leq 1 \\
 & \quad 0 \leq d \leq n - 1,
\end{align*}
\]

where \(A\) is the adjacency matrix of the graph \(G\), \(n\) is the number of vertices in \(G\), \(0\) is the all zeros vector, \(1\) is the all ones vector, and \(\leq\) for vectors denotes component-wise inequality. (If we restrict \(x\) and \(d\) to be integer-valued, then this is exactly the same problem.) Now let us define this in the language of an optimization problem.

**Definition 3.17 (Special Linear Programming).**

**Instance:** \(m \times n\) Boolean matrix \(A\), non-negative rational vector \(c\) of length \(n\), non-negative rational \(c_0\).

**Solution:** non-negative rational vector \(x\) of length \(n\) and rational \(d\) satisfying the inequalities

\[
\begin{align*}
\text{maximize} & \quad 1, c^T x + c_0 d \\
\text{subject to} & \quad nx + d1 \leq n1 + Ax \\
 & \quad 1^T x > 0 \\
 & \quad 0 \leq x \leq 1 \\
 & \quad 0 \leq d \leq n - 1,
\end{align*}
\]

**Measure:** \(\max(1, c^T x + c_0 d)\).

**Type:** maximization.

**Conjecture 3.18.** Special Linear Programming is complete for \(\text{PO} \cap \text{ApxNCO}\) under logarithmic space \(\text{AP}\) reductions.
3.3.4 Approximable problems in NNC(polylog)

In the previous sections we sought a problem complete for \( \text{PO} \cap \text{ApxNCO} \). Perhaps we can find a problem complete for \( \text{NNCO} \cap \text{ApxNCO} \).

**Definition 3.19 (Partial Satisfiability).**

Instance: Boolean formula \( \phi \) on variables \( x_1, \ldots, x_n \), non-negative rational weights \( w_1, \ldots, w_n \) such that \( W \leq \sum_{i=1}^{n} w_i \leq 2W \) for some positive rational \( W \), partial truth assignment \( \alpha \) to the first \( n - \log k n \) of the variables.

Solution: partial truth assignment \( \beta \) to the remaining \( \log k n \) variables.

Measure: \( \max(1, \sum_{i=1}^{n} (\alpha \cup \beta)(x_i)w_i) \) if \( \alpha \cup \beta \) satisfies \( \phi \), otherwise \( W \).

Type: maximization.

For each fixed positive integer \( k \), the budget problem for Partial Satisfiability is complete for \( \text{NNC}^1(\log^k n) \).

**Theorem 3.20.** For every positive integer \( k \), Partial Satisfiability \( \text{opt} > \) is complete for \( \text{NNC}^1(\log^k n) \) under logarithmic space many-one reductions.

**Proof.**

**Todo 3.21.** Fill me in.

The optimization problem is trivially 2-approximable (each assignment has measure at least \( W \), and the optimal assignment has measure at most \( 2W \)) so the problem is in \( \text{ApxNCO} \).

**Conjecture 3.22.** Partial Satisfiability is complete for \( \text{ApxNCO} \) under logarithmic space \( \text{AP} \) reductions.

This would be surprising if true because the optimization problem ... is also complete for \( \text{ApxNCO} \), and therefore any optimization problem that permits a highly parallel approximation but requires a large solution can be transformed into a problem that requires a much smaller solution.

3.4 Hierarchies

The classes of approximable optimization problems are also likely distinct; this is well-known for polynomial time approximability.

**Theorem 3.23 ([2] Exercise 8.1]).** If \( P \neq \text{NP} \) then

\[
\text{PO} \subsetneq \text{PTAS} \subsetneq \text{ApxPO} \subsetneq \text{NPO}.
\]

A natural analog holds for NC approximation classes.

**Theorem 3.24.** If \( NC \neq \text{NP} \) then \( \text{NCO} \subsetneq \text{NCAS} \subsetneq \text{ApxNCO} \subsetneq \text{NNCO} \).
Proof. We begin by showing that ApxNCO = NNCO implies NC = NNC(poly), and hence NC = NP. Let L be a decision problem complete for NNC(poly) under logarithmic space many-one reductions (for example, Satisfiability). Suppose $S_L$ is the relation decidable in NC and $p$ is the polynomial such that $x \in L$ if and only if there is a string $y$ of length $p(|x|)$ such that $(x, y) \in S_L$ for all strings $x$. Define the optimization problem $P$ by $P = (I, S, m, t)$, where

$I = \Sigma^*$,

$S = \{(x, y) \mid |y| \leq p(|x|)\}$,

$m(x, y) = \begin{cases} 1 & \text{if } (x, y) \in S_L, \\ 0 & \text{otherwise, and} \end{cases}$

$t = \max .$

(Technically, the measure function must be positive; we can overcome this by translating the measure function up by some positive value.) Since $S_L$ is in NC, the measure function $m$ is in FNC. The sets $I$ and $S$ are trivially in NC, and $S$ is polynomially bounded, so $P$ is in NNC(poly). By hypothesis $P$ is also in ApxNCO, so there is an NC computable function $A$ that is an $r$-approximator for $P$, for some constant $r \geq 1$. Assume without loss of generality that $A$ enters a special state, call it $\bot$, if $x$ has no solution in $S_L$.

Suppose $x$ is a string that has a solution in $S_L$. Then $m^*(x) = 1$ and thus $m(x, A(x)) \geq \frac{1}{r} > 0$. Define a new algorithm $D$ by “on input $x$, accept if and only if $m(x, A(x)) > 0$”. If $x$ has a solution, then $m(x, A(x)) > 0$, otherwise $A$ will output $\bot$ and $D$ will reject. Furthermore, $D$ is computable by an NC circuit because both $A$ and $m$ are. Therefore $D$ is an NC circuit that decides $L$, so NC = NNC(poly).

If NCAS = ApxNCO we can use a similar argument with $m(x, y) = \frac{1}{r}$ in the second case to produce NC = NP. This technique does not seem to work when attempting to prove NCO = NCAS implies NC = NP. Instead we consider Maximum Independent Set for Planar Graphs; this problem is in NCAS [10, Theorem 5.2.1], and its budget problem is NP-complete [14]. Therefore an exact NC algorithm for it implies NC = NP.

As a corollary to this theorem, since Maximum Variable-Weighted Satisfiability is complete for the class NNCO under logarithmic space AP reductions, it admits no NC approximation algorithm unless NC = NP.

Theorem 3.25. If NC $\neq$ P then

$\text{NCO} \subseteq \text{PO} \cap \text{NCAS} \subseteq \text{PO} \cap \text{ApxNCO} \subseteq \text{PO} \cap \text{NNCO} \subseteq \text{PO}.$

Proof. From Theorem 3.12 we know that PO $\cap$ NNCO = PO implies NC = P. If either PO $\cap$ NCAS = PO $\cap$ ApxNCO or PO $\cap$ ApxNCO = PO $\cap$ NNCO, we can use the same technique as in Theorem 3.24. Instead of a problem complete for NNC(poly), use a problem complete for P (which is a subset of NNC(poly) anyway). Then the optimization problem $P$ is in PO, but an NC approximation
algorithm for it implies an NC algorithm for the decision problem \( L \), and therefore \( NC = P \).

Suppose now that \( NCO = PO \cap NCAS \). Consider **Positive Linear Programming**, the restriction of **Linear Programming** to only non-negative inputs. This problem is in \( PO \) (because it is a restriction of **Linear Programming**) and in \( NCAS \). However, its budget problem remains \( P \)-complete. If \( NCO = PO \cap NCAS \), then there is an NC algorithm that solves this \( P \)-complete problem exactly, and hence \( NC = P \).

A similar proof shows that, for example, \( PO \cap ApxNCO = ApxNCO \) if and only if \( P = NP \). It can be extended to \( PO \cap NCAS \) as well. Compare this theorem with Theorem 3.12.

**Theorem 3.26.** \( PO \cap ApxNCO = ApxNCO \) if and only if \( P = NP \).

**Proof.** The equation \( PO \cap ApxNCO = ApxNCO \) is equivalent to the inclusion \( ApxNCO \subseteq PO \). For the reverse implication, \( P = NP \) implies \( PO = NPO \) by definition, and therefore \( ApxNCO \subseteq NNCO \subseteq NPO = PO \). For the forward implication, the problem \( \text{MAXIMUM } k \text{-CNF SATISFIABILITY} \) problem is in \( ApxNCO \) by [2, Theorem 8.6]. By hypothesis, it is now in \( PO \) as well. Since its budget problem is \( NP \)-complete, we conclude that \( P = NP \). □

Since **Linear Programming** is complete for the class of maximization problems in \( PO \cap NNCO \) under logarithmic space AP reductions, it admits no NC approximation algorithm unless \( NC = P \). This result suggests an explanation for the fact that \( r \)-approximating **Linear Programming** for any \( r \geq 1 \) is \( P \)-complete [10, Theorem 8.2.7], and further, the fact that any NC approximation algorithm for **Linear Programming** implies \( NC = P \) [10, Theorem 8.2.8]; [Corollary 3.11] [Theorem 3.25] and the fact that AP reductions compose imply that any highly parallel approximation for **Linear Programming** necessitates \( NC = P \).

The hierarchy theorem also provides a simple proof of a result of [10] (although they do not define \( ApxNCO \) in the same way).

**Corollary 3.27** ([10, Theorem 8.2.9]). \( ApxNCO = ApxPO \) if and only if \( NC = P \).

**Proof.** If \( NC = P \) then \( ApxNCO = ApxPO \) by definition. If \( ApxPO \subseteq ApxNCO \) then \( PO \subseteq NNCO \), and hence \( PO \cap NNCO = PO \). By [Theorem 3.25] we conclude that \( NC = P \). □

This result is true if we replace the first equality with \( NCAS = PTAS \), or, indeed, any equality that implies \( PO \subseteq NNCO \).

## 4 Syntactic characterization of ApxNCO

In this section, \( \text{cl}(\mathcal{C}, \leq) \) denotes the closure of the complexity class \( \mathcal{C} \) under \( \leq \) reductions.
In [23], the authors introduce a wealth of problems which are complete under “L reductions” for MaxSNP, the class of maximization problems in syntactic NP (SNP), which is a syntactic characterization of NP. Further work showed that cl([MaxSNP, ≤₈]) = ApxPOpₚb [18, Theorem 1], and since cl(ApxPOpₚb, ≤₈PTAS) = ApxPOpₚb [7], we conclude cl([MaxSNP, ≤₈PTAS]) = ApxPOpₚb [18]. We also have cl([MaxSNP, ≤₈]) = cl([MaxNP, ≤₈]) [18, Theorem 2] and Maximum Satisfiability is complete for MaxNP under ≤₈ reductions. Can these results translate to NC in any way?

According to [10, Theorem 9.1.3], MaxSNP ⊆ NCX. They also ask the provide the following open question: is there a (possibly weaker) reduction ≤₈ such that cl([MaxSNP, ≤₈]) = NCX? It may help to know that if P is complete for MaxSNP under ≤₈ reductions, then P exhibits threshold behavior for constant factor NC approximation algorithms [24, Theorem 9] (see also [10, Theorem 9.2.3]).

**Todo 4.1.** We know that FO[polylog] = NC [16, Theorem 5.2]; can we use this to construct a syntactic definition of ApxNCO? Can we more easily construct a complete problem using this characterization?

## 5 Further questions

**Todo 5.1.** We may propose a Frankenstein reduction that accounts for both the complexity of approximation and the complexity of verification.

**Todo 5.2** (Thanks to Rita for this one). Do there exist, for example, ApxPO-complete optimization problems which are based on both NP-complete problems and PSPACE-complete problems? Σ₂P-complete problems?

## References


