Parallelism with limited nondeterminism
Dissertation prospectus

Jeffrey Finkelstein
Computer Science Department, Boston University
February 10, 2016

1 Introduction

Since parallel computing is again becoming a topic of interest in computer science, it is important to revisit the theoretical foundations of highly parallel computing. “Inherently sequential” computational problems see no significant speedup when run on highly parallel computers. Just as there are efficient approximations for intractable optimization problems, so too are there efficient and highly parallel approximations for optimization problems that are tractable but inherently sequential. (From here forward, we write “parallel” instead of “efficient and highly parallel” and “sequential” instead of “inherently sequential”.) For example, the problem of computing the optimal vector in a positive linear program, a problem relevant to distributed flow control within a network of routers, is sequential, but a vector very close to the optimal one can be computed quickly in parallel. We intend to develop the theory of structural complexity for parallel approximations for tractable sequential problems. This area has not been well-studied, and when it has been studied, the results focus mostly on parallel approximations for intractable optimization problems (that is, NC approximations for NP-complete problems), not parallel approximations for tractable sequential problems (that is, NC approximations for P-complete problems). This prospectus describes work we have already completed and work that remains in developing this theory.

The two sections below discuss two main approaches to proving the limitations of parallel approximations for sequential problems. The first section discusses the complexity classes associated with parallel approximation algorithms, the
second discusses augmenting a parallel algorithm with a small amount of non-
determinism as a technique to prove approximability or inapproximability of
sequential problems. Both concern inapproximability by NC algorithms, but the
former considers P optimization problems while the latter considers NNC(polylog)
problems. Together, these form the basis for proving approximability and in-
approximability for tractable sequential problems by parallel algorithms. This
line of research, along with the fact that the complexity of solving a computa-
tional problem exactly seems to have little to with the approximability of
that problem, emphasizes the need to further determine the relative difficulty of
computational problems with respect to the complexity of decision, verification,
and approximation.

2 Parallelism

The NP-complete problems are intractable, whereas problems in P are tractable.
Similarly, the P-complete problems are tractable but sequential, whereas prob-
lems in NC are both tractable and parallelizable. The theory of approximability
of intractable problems does not provide any guarantees on parallelizability,
and ignores the approximability of tractable sequential problems by parallel
algorithms. As parallel and distributed systems become more widespread, par-
allelizability of programs becomes more valuable. An important and widely
used tractable optimization problem is the linear programming problem, which
is solvable in polynomial time, but has no NC approximation algorithm unless
NC = P. We propose translating the theory of polynomial-time approximations
for NP-complete optimization problems to the setting of parallel approxima-
tions for P-complete optimization problems in order to classify and explain
(in-)approximability by parallel algorithms for important problems like linear
programming. This section defines new complexity classes for optimization
problems that are solvable exactly in polynomial time and approximable to
various degrees by parallel algorithms, then proposes using these classifications
to provide provable guarantees on (in-)approximability.

First, we show that for optimization problems, the complexity of the verifi-
cation procedure matters. Specifically, for some problems, verifying a solution
is a sequential problem, whereas for others, verification can be done in parallel.
This allows us to define a hierarchy of complexity classes between NCO and
NPO consisting of optimization problems that can be approximated in parallel.
We show that this hierarchy (as well as the one resulting from its intersection
with PO) is likely strict. Second, we prove that the linear programming problem
is complete for the class of maximization problems that can be solved exactly
in polynomial time. This new classification for the problem explains why any
parallel approximation for linear programming implies NC = P, a fact known
since 1991. In contrast, not all tractable sequential problems are inapproximable:
there are sequential problems that admit parallel constant-factor approxima-
tions up to a certain fixed threshold, as well as those that admit “fully parallel”
approximation schemes.
These findings demonstrate a rich hierarchy of optimization problems that are all efficiently solvable, but have various different levels of approximability by parallel algorithms. Thus, for some problems, when an exact solution is not required, distributed and parallel systems may be utilized to compute an approximate solution, with various approximation ratios. Future work includes developing further strategies for proving approximability and inapproximability using familiar tools from computational complexity theory.

2.1 Motivating example: computing free group rank

Let’s first consider a particular example of a sequential problem. The free group rank problem is defined as follows: given a finite alphabet $S$ and a finite set $U$ of freely reduced words over $S$, determine the rank of the free subgroup generated by $U$. This problem can be solved in polynomial time via an algorithm known as the Nielsen reduction. However, a more efficient solution for this particular problem is to reduce it to a graph problem in which the graph can be interpreted as a generator for all freely reduced words in the subgroup. This new problem, which seems to have not been studied before, is then to determine a minimum equivalent graph that generates the same subgroup. Though it remains P-complete, the problem can be solved with a simpler and more efficient—but still polynomial-time—algorithm known as Stallings’ folding process (which is in turn a sequential algorithm).

Can this problem of finding a minimum equivalent generating graph be approximated by a parallel algorithm? To what degree can it be approximated? How does it relate to the parallel approximability of other “minimum equivalent program” problems? We develop the theory necessary to answer these types of questions.

2.2 Approximability

Knowing that an optimization problem has no parallelizable exact solution is not wholly satisfying without knowing about its approximability. For comparison with the more familiar NP setting, we know that some intractable problems have polynomial-time approximation schemes. We therefore propose a new theoretical framework for studying parallel approximations for tractable sequential optimization problems.

We first consider the optimization problems in NPO whose solution relations are decidable by an NC circuit family. We propose three new complexity classes, NNCO, ApxNCO, and NCAS. The first is the subclass of NPO in which the instance set, solution relation, and measure function of the optimization problem are all decidable (or computable, in the last case) by an NC circuit family. The second is the subset of NNCO in which the optimization problem admits an NC-computable constant-factor approximation algorithm. The third is the subset of ApxNCO in which the optimization problem admits an NC-computable approximation scheme. These are analogous to the standard complexity classes NPO, APX, and PTAS, and as such, these classes are distinct, unless NC = NP.
Theorem 2.1. If $NC \neq P$, then $NCO \subseteq NCAS \subseteq ApxNCO \subseteq NNCO \subseteq NPO$.

Our real interest is in those optimization problems that have exact polynomial time solutions, so we propose three new complexity classes, $NNCO^*$, $ApxNCO^*$, and $NCAS^*$, each of which is the intersection of the appropriate complexity class with $PO$. In other words, $ApxNCO^*$, for example, is the class of optimization problems for which there is a polynomial-time algorithm that outputs optimal solutions and for which there is a $NC$-computable constant-factor approximation algorithm. Again, these classes are distinct under a common complexity theoretic assumption.

Theorem 2.2. If $NC \neq P$, then $NCO \subseteq NCAS^* \subseteq ApxNCO^* \subseteq NNCO^* \subseteq PO$.

Now what kind of optimization problems are in these classes? First, consider the maximum double circuit value problem: given two Boolean circuits $E$ and $M$ and an input $x$, find a string $y$ such that $E(x) = y$ and $M(x, y)$ is maximized. This problem is in $PO$, since it involves evaluating a circuit, both to verify a solution and to evaluate the objective function, and in fact complete under logarithmic-space approximation-preserving reductions. Next, consider the linear programming problem: given an integer matrix $A$, and integer vectors $b$ and $c$, find a rational vector $x$ such that $c^T x$ is maximized. The linear programming problem is in $NNCO^*$, since optimal solutions are computable in polynomial time, and the solution relation and measure function are computable by reduction to matrix multiplication, which is computable in $NC$.

Now we are faced with a dilemma: there is a logarithmic-space approximation-preserving reduction from the maximum double circuit value problem to the linear programming, proving that the linear programming problem is also complete for $PO$. However, this tells us that $NNCO^*$ is not closed under approximation-preserving reductions—evidence that the complexity of verification and the complexity of approximation are not closely related. In other words, there is more work to be done in establishing the complexity classes and reductions that take into account complexity of decision, verification, and approximation.

2.3 Future work

We would like to provide a complete problem for $ApxNCO^*$, thereby providing an initial problem that has an approximation ratio lower bound in this class. In order to do this, we need either (1) an analog of the PCP theorem with $NC$ verifiers for polynomial time decision problems, or (2) a canonical, “universal” complete problem for $ApxNCO^*$. These seem to be 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 probabilistically checkable proof for a deterministic time complexity class like $PO$; see subsection 3.2 below 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 $APX$ [2, Lemma 2], it is not clear
how to guarantee a polynomial time computable function that produces optimal solutions for such a problem.

We propose three candidate complete problems for ApxNCO*. Since we know the linear programming problem is not approximable by a parallel algorithm but positive linear programming is in NCAS* [4], two of these candidates are compromises between the unrestricted form of linear programming and the highly restricted positive linear programming.

- The maximum high degree subgraph problem. Given an undirected graph, find a vertex-induced subgraph $H$ that maximizes $\text{deg}(H)$, the minimum degree of any vertex in $H$. This problem has threshold behavior for its parallel approximability [1], as we expect from a complete problem.

- The linear programming for high degree subgraph problem. This is the linear programming relaxation of the above combinatorial problem.

- The linear programming with triplets problem. Given a linear program with a certain specific form and an approximation guarantee on the objective function values, find a solution vector that maximizes the objective.

Future techniques for refining our classification of optimization problems based on their parallelizability include defining fixed-parameter parallelizable problems, defining average-case parallelizable problems, and using a descriptive complexity definition for NC to construct a syntactic definition of ApxNCO.

## 3 Limited nondeterminism

As stated in the previous section, we would like to show that some problems are hard to approximate on parallel machines, and one way of doing that involves using an analog of the PCP Theorem, adapted for parallel verifiers. Unfortunately, the techniques used to prove the original PCP Theorem rely on the fact that NP can be interpreted as the class of languages for which there is an efficient verification procedure given a brief witness to language membership; no such obvious interpretation of P exists. Indeed, this question was already on the minds of researchers soon after the original proof of the PCP Theorem.

An intriguing question is whether the known non-approximability results for sequential algorithms can be improved when we restrict to NC algorithms (under the assumption that $P \neq NC$). A possible way may be to devise probabilistic proof systems for P more efficient that the currently known proof systems for NP. Such a result would have a great independent interest. However, it is not clear why proofs for P should be easier to check than proofs for NP (they only appear to be easier to generate).

Perhaps P has proof systems which are easy to check in NC, but this remains unclear. Instead, we consider proof systems for the class NNC(polylog), the class
of languages decidable by NC circuit families augmented with a polylogarithmic number of nondeterministic gates. We hope that this will be a valuable first step toward understanding proof systems for $P$. We consider $\text{NNC}(\text{polylog})$ for two reasons. First, it is defined in such a way that it explicitly has short proof systems which are easy to verify in parallel, just as $\text{NP}$ is defined in such a way that it explicitly has short proof systems which are easy to verify efficiently. Second, it, like $P$, lies between NC and NP.

Our study of $\text{NNC}(\text{polylog})$ demonstrates that it is interesting in its own right. Many properties familiar from the study of nondeterministic polynomial time computation translate directly to $\text{NNC}(\text{polylog})$, for example, the Valiant–Vazirani theorem. It also contains several very natural problems, including the problem of computing the rank of a finite group given as its Cayley table; other algebraic problems can be solved with limited nondeterminism, given such an explicit representation of the algebraic structure. Of particular interest are the algebraic problems common in cryptographic primitives: the discrete logarithm problem, the RSA problem, modular exponentiation, etc., though we have yet to determine whether these can be computed with limited nondeterminism.

Although our original intention was to show a result like $\text{NNC}(\text{polylog}) = \text{PCP}[O(\log \log n), O(1)]$, our research reveals that proving such an equality is equivalent to proving $\text{NNC}(\text{polylog}) = \text{NC}$, or in other words, that a polylogarithmic amount of nondeterminism can be simulated deterministically by an NC circuit family. This should be seen as evidence that such a result is unlikely; in fact, we show that such a simulation implies a deterministic subexponential time algorithm for SATISFIABILITY! We are still, however, able to show that certain PCP classes are contained in $\text{NNC}(\text{polylog})$.

### 3.1 Motivating example: computing Cayley group rank

Let’s first consider a particular example of a problem that can be decided with limited nondeterminism, or equivalently, a problem for which any solution is short and verification is parallelizable. The Cayley group rank problem is defined as follows: given the Cayley table of a group $G$ of order $n$, compute the rank of the group. Since finite groups have generating sets of cardinality $\log n$, this problem can be solved by the following simple algorithm: nondeterministically choose a set of at most $\log n$ elements of $G$, check if that set generates the group, and if so, output the cardinality of the set. Since each element of a set of cardinality $n$ can be represented with $O(\log n)$ bits, the total number of nondeterministically chosen bits is $O(\log^2 n)$. Verifying that a set generates a group can be done with a small circuit, so this problem is in $\text{NNC}(\log^2 n)$. In fact, we provide a new and more efficient verification procedure for this particular problem, still using only $O(\log^2 n)$ bits of nondeterminism.

Does the succinctness of the certificate in this problem tell us something about the approximability of this problem? Can a probabilistically checkable proof system with a parallel verifier solve this problem with a very small amount of randomness and few queries to the proof? We develop the theory necessary to answer these types of questions.
3.2 Probabilistically-checkable proofs

We prove that the following inclusion chains are equivalent in the sense that the corresponding complexity classes in the two chains are equal,

\[ \text{NC} \subseteq \text{NNC}(\text{polylog}) \subseteq \text{NP} \]

and

\[ \text{PCP}[O(\log \log n), O(1)] \subseteq \text{PCP}[O(\log \log n), \text{polylog}] \subseteq \text{PCP}[O(\log n), O(1)], \]

where the PCP verifier is an \text{NC} machine. This equivalence allows us to show that certain tradeoffs in the two PCP parameters are unlikely.

**Theorem 3.1.**

1. If \( \text{PCP}^{\text{NC}}[O(\log \log n), O(1)] = \text{PCP}^{\text{NC}}[O(\log \log n), \text{polylog}] \), then \( \text{NP} \subseteq \text{SUBEXP} \).

2. If \( \text{PCP}^{\text{NC}}[O(\log \log n), \text{polylog}] = \text{PCP}^{\text{NC}}[O(\log n), O(1)] \), then \( \text{NP} \subseteq \text{QP} \).

The first part of this corollary provides evidence that for certain classes of computational problems, an \text{NC} PCP verifier cannot reduce the number of necessary queries. The second part provides evidence that for certain classes of computational problems, a verifier cannot reduce randomness in exchange for an increase in the number of necessary queries.

But our real goal is to determine a PCP characterization of \( \text{P} \). The classes \( \text{P} \) and \( \text{NNC}(\text{polylog}) \) are conjectured incomparable \[6\]. Using the results above, this conjecture implies that \( \text{P} \) and \( \text{PCP}^{\text{NC}}[O(\log \log n), \text{polylog}] \) are incomparable. **Theorem 3.2** shows consequences of a PCP characterization for \( \text{P} \).

**Theorem 3.2.**

1. If \( \text{P} \subseteq \text{PCP}^{\text{NC}}[O(\log \log n), \text{polylog}] \) then \( \text{P} \subsetneq \text{polyL} \).

2. If \( \text{polyL} \subsetneq \text{P} \) then \( \text{PCP}^{\text{NC}}[O(\log \log n), \text{polylog}] \subsetneq \text{P} \).

Although \( \text{P} \neq \text{polyL} \) (since the former has a complete problem under logarithmic space many-one reductions while the latter does not), whether one is a strict subset of the other remains unknown. The two are conjectured to be incomparable \[3\] Section 2.5.1], and this makes sense. If \( \text{P} \subsetneq \text{polyL} \), then \( \text{P} \subsetneq \text{QP} \) (by exhaustive search over the quasipolynomial number of configurations of the \text{polyL} machine) and \( \text{P} \subsetneq \text{PSPACE} \) by the space hierarchy theorem, which would resolve a major open problem in computational complexity theory. If \( \text{polyL} \subsetneq \text{P} \), then \( \text{L} \subsetneq \text{L}^2 \subsetneq \cdots \subsetneq \text{polyL} \subsetneq \text{P} \), another longstanding open problem.

3.3 Future work

As we stated in subsection 2.3, we would like to provide a complete problem for \( \text{ApxNCO}^* \) by using a PCP characterization of \( \text{P} \). We have only provided a
PCP characterization of $\text{NNC}(\text{polylog})$, but we may still be able to use that to construct a problem complete for the class of optimization problems with optimal solutions computable in $\text{NNC}(\text{polylog})$, but with $\text{NC}$-computable constant-factor approximations. As we have shown natural problems in $\text{NNC}(\text{polylog})$, this would still be of interest, and a start toward showing the same for $\text{ApxNCO}^*$.

References


