Unambiguous parallel computation with short witnesses

Jeffrey Finkelstein
Computer Science Department, Boston University

October 15, 2014

This work describes the relationship between problems that have many short witnesses and those that have one short witness, for problems that are verifiable in parallel. These proofs follow not the original proof of the Valiant–Vazirani Theorem, but [4] and [1].

Definition 0.1. An $NC^d$ circuit is a Boolean circuit with fan-in two, polynomial size, and $O(\log^d n)$ depth.

Definition 0.2. $NNC^d(\log^k n)$ is the class of languages decided by $L$-uniform $NC^d$ circuits augmented with $O(\log^k n)$ nondeterministic gates. $UNC^d(\log^k n)$ is the subset of $NNC^d(\log^k n)$ in which strings in the language have exactly one witness and strings not in the language have no witnesses. $RNC^d(\log^k n)$ is the class of languages decided by $L$-uniform $NC^d$ circuits augmented with $O(\log^k n)$ bits of randomness that have soundness $1$ and completeness $\frac{1}{2}$.

Definition 0.3. If $C$ is a Boolean circuit with inputs $x_1, \ldots, x_n$, a partial assignment to its inputs is a function $\alpha: S \rightarrow \{0, 1\}$, where $S$ is some subset of $\{x_1, \ldots, x_n\}$. The partial assignment is called a total assignment, or simply an assignment, if $|S| = n$. For any fixed partial assignment $\alpha$ with domain $S$, an extension assignment, or simply an extension, is a partial assignment with domain $(\{x_1, \ldots, x_n\} \setminus S)$. (Thus, if $\beta$ is an extension of $\alpha$, then $\alpha \cup \beta$ is a total assignment.)

We will examine the following two restrictions of the Boolean Formula Satisfiability problem.

Definition 0.4. $\log^k n$-Partial $NC^d$ Satisfiability is the language of all pairs $(C, \alpha)$, where $C$ is an $NC^d$ circuit with $n$ inputs and $\alpha$ is a partial assignment to all but the first $\log^k n$ inputs of $C$, such that there is an extension of $\alpha$
that satisfies $C$. Unambiguous $\log^k n$-Partial NC$^d$ Satisfiability is the
restriction of the above language to only those pairs for which there is at
most one satisfying assignment.

$\log^k n$-Partial NC$^d$ Satisfiability is complete for NNC$^d(\log^k n)$; its
unambiguous restriction is in UNCC$^d(\log^k n)$. (When $k = 1$ and $d = 1$, the former
problem is equivalent to Boolean Formula Evaluation.)

**Definition 0.5.** A collection of functions $H_{n,m}$ from \{0, 1\} to \{0, 1\}$^m$ is a
pairwise independent hash family if for all distinct $x_1$ and $x_2$ in \{0, 1\}$^n$ and all
$y_1$ and $y_2$ in \{0, 1\}$^m$,

$$\Pr_{h \leftarrow H_{n,m}}[h(x_1) = y_1 \land h(x_2) = y_2] = \frac{1}{2^m}.$$  

A pairwise independent hash family is called NC$^d$-explicit if the function
$(h, x) \mapsto h(x)$ is computable in NC$^d$.

**Lemma 0.6.** There is an NC$^d$-explicit pairwise independent hash family for all $d > 2$. 

**Proof.** There is a simple construction of an NC$^2$ pairwise independent hash
family. The collection of functions defined by $H_{n,m} = \{h_{A,b} \mid h_{A,b}(x) = Ax + b\}$,
where $A$ is an $m \times n$ Boolean matrix and $b$ is an $m$-dimensional bit vector, is a
well-known pairwise independent hash family. Boolean matrix multiplication is
computable in NC$^2$, so this family is NC$^2$-explicit.

I believe we can also construct NC$^0$ pairwise independent hash families, using

**Lemma 0.7.** Suppose $n$ and $m$ are positive integers, $H_{n,m}$ is a pairwise in-
dependent hash family, and $S$ is a subset of \{0, 1\}$^n$. If $2^{m-2} \leq |S| \leq 2^{m-1}$
then

$$\Pr_{h \leftarrow H_{n,m}}[\text{there is a unique } x \in S \text{ such that } h(x) = 0^m] \geq \frac{1}{8}.$$  

The specific string $0^m$ in the previous lemma is irrelevant; any fixed binary
string would work just as well.

**Theorem 0.8** (Valiant–Vazirani). There is a probabilistic logarithmic space
computable function $f$ such that for all NC$^d$ circuits $C$ on $n$ inputs, and all
partial assignments $\alpha$ to all but the first $\log^k n$ of the inputs, the following
conditions are satisfied.

- $f(C, \alpha)$ outputs a circuit $C'$ and a partial assignment to its inputs $\alpha'$.
- If there is an extension of $\alpha$ that satisfies $C$ then with probability at least
  $\frac{1}{8 \log n}$ there is a unique extension of $\alpha'$ that satisfies $C'$.
- If there is no extension of $\alpha$ that satisfies $C$ then with probability 1 there
  is no extension of $\alpha'$ that satisfies $C'$. 

2
The probabilities are over the uniform random bits used by $f$.

Proof. Let $H_{\log^k n,m}$ be an $NC^d$-explicit pairwise independent hash family as guaranteed by Lemma 0.6. Define the function $f$ as follows on input circuit $C$ and partial assignment given as bits $x_{\log^k n+1}, x_n$.

1. Choose $m$ uniformly at random from $\{2, \ldots, \log^k n + 1\}$.
2. Choose $h$ uniformly at random from $H_{\log^k n,m}$.
3. Let $C'(x_1, \ldots, x_n)$ be the circuit computing $(C(x_1, \ldots, x_n) = 1) \land (h(x_1 \circ \cdots \circ x_{\log^k n}) = 0^m)$.

(We only use the the first $\log^k n$ bits of the input in the hash function because we expect a partial assignment to fix the value of the remaining inputs.)

4. Output $C'$ and the same partial assignment $\alpha$.

The probability that $m$ is chosen such that the number of satisfying assignments to the original circuit $C$ is between $2^{m-2}$ and $2^{m-1} \leq \frac{1}{\log n}$. Conditioned on that event, the probability that $h$ is chosen as in Lemma 0.7 is at least $\frac{1}{8}$. Thus if there is an extension of $\alpha$ satisfying $C$ then the probability that there is a unique extension satisfying $C'$ is at least $\frac{1}{8 \log^k n}$. If the original circuit has no satisfying extension, then neither does $C'$, so the probability that $C'$ has a satisfying extension is 0.

The efficiency of $f$ is merely the efficiency of constructing the relatively simple circuit $C'$, and so can be computed in logarithmic space. Finally, $f$ uses $\log^k n$ random bits, or $k \log \log n$ bits, to choose $m$ (in binary) and uses $O(\log^k n)$ random bits to choose $h$ (TODO explain why: see Pseudorandomness). Thus the total number of random bits used is $O(\log^k n + \log \log n)$.

Corollary 0.9. Let $d$ be a positive integer (greater than ...) and $k$ be any non-negative integer. If $NC^d = UNC^d(\log^k n)$ then $NNC^d(\log^k n) \subseteq RNC^d(\log^{2k} n)$, and furthermore $NNC^d(\text{polylog}) = RNC^d(\text{polylog})$.

Proof. The “furthermore” part of the conclusion follows from the syntactic inclusion $RNC^d(\log^k n) \subseteq NNC^d(\log^k n)$, so we will prove that the hypothesis implies $NNC^d(\log^k n) \subseteq RNC^d(\log^{2k} n)$.

This is a corollary of the proof of the previous theorem. It suffices to show an $RNC^d$ algorithm for $\log^k n$-PARTIAL $NC^d$ SATISFIABILITY, which is a complete problem for $NNC^d(\log^k n)$ under logarithmic space many-one reductions. By hypothesis, there is an $NC^d$ algorithm, call it $U$, that decides UNAMBIGUOUS $\log^k n$-PARTIAL $NC^d$ SATISFIABILITY, which is in $UNC^d(\log^k n)$.

Consider the function $f$ from the proof of the previous theorem. Instead of choosing $m$ at random in the first step of the function $f$, we will try all possible values of $m$ in parallel in order to find the correct number of satisfying assignments. Consider an algorithm, call it $A$, that proceeds as follows on inputs $C$ and $\alpha$. 

3
1. For each $m \in \{2, \ldots, \log k n + 1\}$ in parallel:
   (a) Choose $h$ uniformly at random from $H_{\log k n, m}$.
   (b) Let $C'_m$ be the circuit whose construction was described in the previous
       theorem (that is, step 3 of the definition of $f$).

2. Accept if and only if
   \[
   \bigvee_{m=2}^{\log k n+1} U(C'_m, \alpha) = 1.
   \]

If there is at least one satisfying extension for $C$, then exactly one of the $m$
will correspond to the correct number of extensions (specifically, there will be
exactly one $m$ such that the number of extensions is between $2^{m-2}$ and $2^{m-1}$).
The probability that $U$ will accept that $C'_m$ is at least $1/8$, by Lemma 0.7, so $A$
will accept with at least that probability (since it accepts if any of the $U(C'_m, \alpha)$
output 1). As before, if $C$ has no satisfying extensions then neither does any of
the $C'$, so $A$ rejects with probability 1.

In order to get the completeness probability to be greater than $1/2$, we amplify
the probability of success of $A$ by running some number of independent instances
of $A$ in parallel and accepting if and only if at least one of the instances accepts. If
the probability that each instance accepts is at least $1/8$, then the probability that
each instance rejects is less than $7/8$. If $t$ instances of $A$ are run independently in
parallel, the probability that they all reject is less than $(7/8)^t$. Thus the probability
that at least one accepts is at least $1 - (7/8)^t$. This is greater than $1/2$ if $t$ is at least
6. Therefore the overall algorithm, call it $A'$, consists of 6 parallel instances of
$A$ with a disjunction at the final layer.

Now we determine the efficiency of the algorithm to ensure that it is in
$\text{RNC}_d(\log k n)$. Each instance of $A$ consists of a disjunction of $O(\log k n)$ instances
of the $\text{NC}_d^c$ circuit $U$. The inputs to each of those instances of $U$ are the outputs
from $C'_m$, itself a conjunction of two $\text{NC}_d^c$ circuits (namely, the circuits for $C$ and
$h$). Hence $A$ is an $\text{NC}_d^c$ circuit, and the disjunction of six instances of $A$ is also
an $\text{NC}_d^c$ circuit. Furthermore, the number of random bits used in $A$ equals the
total number of bits used to choose each of the hash functions $h$. Each hash
function requires $O(\log k n)$ bits (TODO this is a known fact, state it), and
there are $O(\log k n)$ of them. Therefore the total number of random bits required
is $O(\log^{2k} n)$. \hfill \Box

TODO can we use randomness-efficient error reduction to reduce the amount
of randomness required down to $O(\log k n)$, thereby allowing the conclusion
$\text{NNC}_d^c(\log k n) = \text{RNC}_d^c(\log k n)$ in the above corollary?

References

