On the **TFNP** complexity of factoring

Joshua Buresh-Oppenheim
Simon Fraser University
jburesho@cs.sfu.ca

November 21, 2006

In the following note, we place certain versions of factoring in the context of combinatorial subclasses of **TFNP** defined by Papadimitriou [Pap94]. We observe that the power of the complexity class **PPA** is sufficient to factor a large, natural class of numbers that include Blum integers. In addition, with some access to randomness, a more general class of numbers can be factored using **PPP**. Both of these observations are based on well-known techniques from algorithmic number theory (see, for example, [BS96]), but nevertheless begin to answer an open question of [Pap94].

1 Preliminaries

Each problem in **FNP** is defined by a polytime relation $R \subset \Sigma^* \times \Sigma^*$ and an integer k such that $(x,y) \in R \Rightarrow |y| \leq |x|^k$. Specifically, the search problem associated with R is, given x, to find y such that $(x,y) \in R$ or output "no" if there is no such y. The class **TFNP** is the set of all **FNP** search problems such that, for all x, there is a y such that $(x,y) \in R$.

We now describe two very natural problems in **TFNP** that give rise to entire subclasses via reduction.

Definition 1. *Let* **PIGEON** *be the following total* **NP** *search problem: given input* $(1^n, C)$*, where* C *is a boolean circuit computing a function from* $\{0,1\}^n$ *to* $\{0,1\}^n$*, return either (i)* $x \in \{0,1\}^n$ *such that* $C(x) = \overline{0}$*; or (ii)* $x,y \in \{0,1\}^n$ *such that* $C(x) = C(y)$*.*

We define **PPP** (Polynomial Pigeonhole Principle) as the class of all total **NP** search problems that are poly-time reducible to **PIGEON**.

Definition 2. *Let* **LONELY** *be the following total* **NP** *search problem: given input* $(1^n, C)$*, where* C *is a boolean circuit computing a function from* $\{0,1\}^n$ *to* $\{0,1\}^n$*, return either (i)* $x \in \{0,1\}^n \setminus \{\overline{0}\}$ *such that* $C(\overline{0}) = x$*; (ii)* $x,y \in \{0,1\}^n$ *such that* $C(x) = y$*, but* $C(y) \neq x$*; or (iii)* $x \in \{0,1\}^n \setminus \{\overline{0}\}$ *such that* $C(x) = x$*.*

We define **PPA** (Polynomial Parity Argument) as the class of all total **NP** search problems that are poly-time reducible to **LONELY**.

Note that **LONELY**, which embodies the combinatorial principle that there is no perfect matching on an odd number of nodes, is not the original defining problem for **PPA**. The original problem is based on the principle that in any undirected graph of degree at most 2, there is an even number of leaves. To see that the two are equivalent, see [BCE+98].

Finally, we define **FZPP** to be the class of problems in **FNP** solvable by a randomized Turing Machine in expected polynomial time. Similarly, **FP** is the class of problems in **FNP** solvable by a deterministic Turing

Machine in polynomial time. We define total versions of these two classes as $\mathbf{TFZPP} = \mathbf{FZPP} \cap \mathbf{TFNP}$ and $\mathbf{TFP} = \mathbf{FP} \cap \mathbf{TFNP}$.

Definition 3. *Call N a* good integer *if it is an odd, positive integer such that -1 is not a quadratic residue modulo N. Call N* 4good *if it is congruent to 1 modulo 4 in addition to being good.*

Note that, in particular, Blum integers are 4good integers.

We consider the following search problem (and its restriction). Notice that both are total **NP** search problems:

Good-Integer-Factoring (GIF): Given a positive integer N, return
(i) N, if N is prime or if $N = 1$;
(ii) a non-trivial factor of N or a square root of -1 modulo N, otherwise.

4Good-Integer-Factoring (4GIF): Given a positive integer N, return
(i) N, if $N \neq 1 \bmod 4$ or if N is prime or if $N = 1$;
(ii) a non-trivial factor of N or a square root of -1 modulo N, otherwise.

We are now ready to give the main results. We will show that **4GIF** $\in$ **PPA** and **GIF** $\in \mathbf{TFZPP}^{\mathbf{PPP}}$. In what follows, all computations are done modulo N unless otherwise stated.

Definition 4. *Given $a \in \mathbb{Z}/N$, N odd, call a* positive *if $a \in \{1, ..., \frac{N-1}{2}\}$ and* negative *if $a \in \{\frac{N+1}{2}, ..., N-1\}$. Let $|a|$ denote a if a is positive, and $N-a$ if a is negative.*

Theorem 5. **4GIF** $\in$ **PPA**.

Proof. Given N, test if it is prime or 1 or if it is not congruent to 1 modulo 4. If so, return N. Otherwise, we create an instance of **LONELY** on strings of length $length(N)$. We want the number 1 to correspond to the $\overline{0}$ element of **LONELY**, so in general let the number x be coded by the binary string of length $length(N)$ whose value is $x-1$. We now describe the matching in terms of numbers (which can be easily translated into strings).

The number 1, of course, will be matched with itself. Any odd number x greater than $(N-1)/2$ and less than $2^{length(N)}$ will be matched with $x+1$, and any even number greater than $(N-1)/2$ will be matched with $x-1$ (here $x+1$ and $x-1$ are not taken modulo N). Note that $(N-1)/2$ is even, so there are an even number of numbers x such that $(N-1)/2 < x \leq 2^{length(N)}$. Any number $1 \leq x \leq (N-1)/2$ that is not a unit in $\mathbb{Z}/N$ will be matched with itself. Finally, any number $1 \leq x \leq (N-1)/2$ that is a unit will be matched with $|x^{-1}|$.

If x is positive, then $y = |x^{-1}| \Rightarrow x = |y^{-1}|$, so there will be no solutions to **LONELY** of type (ii). Hence we consider solutions of type (iii): any number other than 1 that is matched with itself is not a unit, is a square root of 1 or is a square root of -1. Hence, given a solution x to **LONELY**, we do three computations:
(1) if $x^2 = -1$, then return x;
(2) otherwise, if $gcd(x,N) \neq 1$, then return $gcd(x,N)$ as a nontrivial factor of N;
(3) otherwise, if $x^2 = 1$, then $(x+1)(x-1) = 0$ and we know $|x| \neq 1$, so $gcd(x+1,N)$ is a nontrivial factor of N. □

Theorem 6. **GIF** $\in \mathbf{TFZPP}^{\mathbf{PPP}}$.

Proof. Given N, test if N is prime or 1; if so, output N. If N is even, output 2.

Otherwise, we construct an instance of **PIGEON** on strings of length $length(N)$ as follows: begin by choosing a random positive number a (modulo N). The $\overline{0}$ string will stand for the number -1; otherwise,

each binary string will stand for its value as a number. Map -1 to a; if $x > (N-1)/2$, map x to itself; otherwise, map x to $|x^2|$.

We now analyze the possible solutions the **PPP** oracle could return. Since -1 is negative, nothing will ever map to -1 (even if N is not good), so there will be no solutions to **PIGEON** of type (i). If the oracle returns -1 and y that both map to a, then clearly either a or $-a$ is a quadratic residue. If this is the case, we repeat the whole process with a new random a. Since N is the product of at least two odd primes, neither a nor $-a$ will be a quadratic residue with probability at least one half. Otherwise, assume the oracle returns x and y that both map to z. If $x^2 = -y^2$, then N is not good and we return xy^{-1}, which is a square root of -1 (if y is not a unit—i.e. if $gcd(y,N) \neq 1$—then return $gcd(y,N)$). Finally, if $x^2 = y^2$, then $x^2 - y^2 = 0$ and $(x+y)(x-y) = 0$. Since x and y are positive and $x \neq y$, it must be the case that $gcd(x+y,N) \neq 1$, so we return this. □

2 Open Questions

It would be nice to derandomize the **PPP** result. Derandomization follows from the Extended Riemann Hypothesis. That is, the ERH implies that **GIF** $\in$ **PPP**. This is because the ERH guarantees that there is an appropriate element a in the range $1...O(\log^2 N)$. Without such an assumption, it is unknown how to find an appropriate a in deterministic polynomial time. In fact, it also unknown how to find a quadratic non-residue in $\mathbb{Z}/p$ for prime p. Of course, derandomizing our reduction does not require solving either of these long standing open problems since we have a **PPP** oracle at our disposal. Hence, a nice intermediate open question might be to show that finding a quadratic non-residue modulo a prime p is in **PPP**. This begins to address another open problem of [Pap94]: is there a relationship between **TFZPP** and **PPP** or **PPA**?

Of course, the biggest and most interesting open problem is to show that factoring every number N, not just good integers, is in one or both of these classes.

References

[BCE+98] Paul W. Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of *NP* search problems. *Journal of Computer and System Sciences*, 57:3–19, 1998.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms*. MIT Press, 1996.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.