UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL CHICAYBAN BASTOS

UMA GENERALIZAÇÃO DO ALGORITMO DE SHOR

NITERÓI

2023

UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL CHICAYBAN BASTOS

UMA GENERALIZAÇÃO DO ALGORITMO DE SHOR

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

Orientador: LUIS ANTONIO BRASIL KOWADA

NITERÓI

2023

Ficha catalográfica automática - SDC/BEE Gerada com informações fornecidas pelo autor

C532g	Chicayban Bastos, Daniel Uma generalização do algoritmo de Shor / Daniel Chicayban Bastos 2023. 53 f.: il.
	Orientador: Luis Antonio Brasil Kowada Kowada. Tese (doutorado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2023.
	 Fatoração. 2. Computação quântica. 3. Algoritmo Rô de Pollard. 4. Algoritmo de Shor. 5. Produção intelectual. Kowada, Luis Antonio Brasil Kowada, orientador. II. Universidade Federal Fluminense. Instituto de Computação.III. Título.
	CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

DANIEL CHICAYBAN BASTOS

UMA GENERALIZAÇÃO DO ALGORITMO DE SHOR

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

Aprovada em 9 de janeiro de 2023.

BANCA EXAMINADORA

ande ~~4

Prof. Dr. LUIS ANTONIO BRASIL KOWADA - Orientador, UFF

Estrio Borges de Aliveira Prof. Dr. FÁBIO BORGES DE OLIVEIRA, LNCC in de dura Prof. Dr. FRANKLIN DE LIMA MARQUEZINO, UFRJ

Prof. Dr. LUIS FELAPE IGNÁCIO CUNHA, UFF

Verten des Emter Sunge

Prof. Dr. UÉVERTON DOS SANTOS SOUZA, UFF

NITERÓI

2023

A Maria Cristina Bessa Moreira.

Agradecimentos

Ao Prof. Dr. Kowada, por sua amizade, por compartilhar sua pesquisa, suas descobertas, sua experiência, atenção e delicadeza. Pelos anos de trabalho gentilmente compartilhados; pelos desafios que enfrentamos. Muito obrigado!

A minha querida família, pela alegria de ser filho e irmão.

A Deus, por tudo.

Resumo

O algoritmo Rô de Pollard é uma estratégia clássica para fatoração de inteiros que não sejam potências primas. Apresentamos uma nova forma de entender o algoritmo Rô de Pollard; propomos um teorema que caracteriza as soluções do Problema do algoritmo Rô de Pollard no universo de busca do método, levando-nos a uma versão quântica do Rô de Pollard bem como a percepção de que o algoritmo de Shor é um caso particular dele. Nas implementações da versão quântica do algoritmo Rô de Pollard que apresentamos, utilizamos uma fórmula fechada que requer a computação da ordem de um número módulo o composto que se deseja fatorar, o que implica uma dependência do algoritmo quântico de cálculo da ordem, apresentando-se como um obstáculo à noção de que o algoritmo de Shor seja um caso particular de uma versão quântica do Rô de Pollard. Entretanto, se for possível encontrar fórmulas fechadas para funções cíclicas que não dependam da ordem de um número, a dependência do algoritmo quântico de cálculo da ordem desaparece. Da caracterização das soluções do Problema do algoritmo Rô de Pollard, extraímos uma estratégia que nos permite estudar estatisticamente o algoritmo de Shor sem qualquer necessidade de um computador quântico. Uma aplicação é produzir evidência direta de quantas vezes o algoritmo de Shor precisaria invocar o algoritmo quântico de cálculo da ordem contra compostos maiores que os usados em criptografia hoje. A aplicação também nos permite comparar todas as estratégias hoje disponíveis na literatura para preparar a entrada do algoritmo de Shor. Além disso, apresentamos uma técnica para o algoritmo de Shor original aproveitar uma ordem ímpar computada pelo algoritmo quântico de cálculo da ordem de elementos do grupo multiplicativo \mathbb{Z}_{N}^{*} . Desconsiderando a taxa de fracasso do algoritmo quântico de cálculo da ordem, mostramos que semiprimos pq tal que $p = 2\ell + 1$ e q = 2m + 1, sendo ℓ, m inteiros ímpares, podem ser fatorados pelo algoritmo de Shor com probabilidade 1 de sucesso. Por fim, incluímos uma descrição de um circuito quântico para a versão quântica do algoritmo Rô de Pollard em termos de portas elementares, mostrando como o método pode ser implementado em um computador quântico.

Palavras-chave: algoritmo Rô de Pollard; algoritmo de Shor; fatoração; computação quântica; período de funções; ordem de elementos em grupos finitos.

Abstract

Pollard's Rho is a well-known classical strategy for factoring integers that are not prime powers. We present a new perspective over the Pollard's Rho algorithm; propose a new theorem that characterizes the solutions of the Pollard's Rho Problem in the universe of search of the method, leading us to a quantum version of Pollard's Rho and to the understanding that Shor's algorithm is a particular case of the quantum-version of Pollard's Rho presented. In our implementation of the quantum version of Pollard's Rho, we use a closed-form formula that requires computing the order of a number modulo the composite we wish to factor, which implies a dependence on the order-finding algorithm, challenging the idea that Shor's algorithm is a particular case of a quantum version of Pollard's Rho. However, if it is possible to find closed-form formulas for cyclic functions that do not depend on the order of a number, the dependency vanishes. From the characterization of the solutions to the Pollard's Rho Problem, we produced a strategy that lets us study Shor's algorithm statistically without any need for a quantum computer. One application is to produce statistical evidence of how many times Shor's algorithm would invoke the order-finding method against composites greater than the ones used in cryptography today. The strategy also let us compare all methods available today in the literature for preparing the input to Shor's algorithm. We also present a technique for the original Shor's algorithm to use an odd order computed by the order-finding method. Not considering the failure rate of the order-finding method, we show that semiprimes pq such that $p = 2\ell + 1$, q = 2m + 1, where ℓ, m are odd integers, can be factored by Shor's algorithm with success probability 1. Last, we include a description of a quantum circuit for the quantum version of Pollard's Rho in terms of elementary gates, showing how the method can be implemented in a quantum computer.

Keywords: Pollard's Rho; Shor's algorithm; factoring; quantum computing; period-finding; order-finding; order of elements in finite groups.

Sumário

1	Intr	odução	1
	1.1	Uma descrição superficial do algoritmo de Shor	2
	1.2	O símbolo de Legendre e o símbolo de Jacobi	3
	1.3	O estado da arte relativo ao algoritmo de Shor	4
2	O al	goritmo Rô de Pollard	8
	2.1	A estratégia de Pollard	8
	2.2	A importância de uma função cíclica	13
	2.3	A complexidade do algoritmo	14
	2.4	Uma nova perspectiva sobre o método	15
3	Uma	a versão quântica do Rô de Pollard	19
	3.1	Uma caracterização das colisões não-triviais	19
	3.2	O paralelismo quântico	22
	3.3	Uma expressão da estratégia num modelo quântico	24
	3.4	Outras famílias de funções cíclicas para a versão quântica	25
	3.5	Sobre a família de funções super-exponenciais	28
	3.6	Sobre a família de polinômios quadráticos	29
	3.7	A complexidade da versão quântica	30
4	Uma	a generalização do algoritmo de Shor	31
	4.1	A estratégia de Shor	31
	4.2	Uma extensão do algoritmo de Shor	32
	4.3	O algoritmo de Shor como um caso particular do Rô de Pollard	34
	4.4	A equivalência ao Teorema de Anna M. Johnston	35

Bi	bliog	rafia	47
6	Con	clusões	45
	5.3	Condições para o Shor obter sucesso com probabilidade 1	41
	5.2	O algoritmo de Shor (e suas variações) contra inteiros grandes	40
	5.1	O comportamento do algoritmo de Shor relativo a um $x \mod N$	38
5	Apli	cações decorrentes da generalização	38
	4.5	O cálculo da ordem no modelo quântico	36

Capítulo 1

Introdução

Distinguir números primos de compostos era um problema bem conhecido entre os geômetras da antiguidade e possivelmente até antes (Hendy 1975). Há quem afirme que a primeira formulação do teorema fundamental da aritmética¹ já se encontrava (Hendy 1975) nos *Elementos* de Euclides (Heath 1956), obra publicada por volta do ano 300 a.C., apesar de faltar clareza sobre *exatamente* o que se sabia naquela época. Por exemplo, Hardy e Wright afirmam que o teorema fundamental da aritmética não parece ter sido enunciado antes de Gauss (Hardy e Wright 1975, primeira nota ao fim do capítulo 1) em 1801 (Gauss 1986), mas também explicam que Euclides sabia muito bem que a teoria dos números surgia do seu algoritmo para calcular o maior divisor comum entre dois outros. A Euclides faltava linguagem para falar em multiplicação ou exponenciação. Teria sido difícil enunciar o próprio teorema (Hardy e Wright 1975, capítulo 12, seção 12.5, páginas 181–182), mas isso não significa que os gregos não tinham o entendimento.

O crivo de Eratóstenes (Knuth 1997, capítulo 4, seção 4.5.4, exercício 8, página 412), conhecido desde o século terceiro a.C., ilustra tanto a distinção entre primos e compostos como a fatoração porque obtém-se uma fatoração de cada composto retido pelo crivo. Por exemplo, se o número 3 faz o número 15 ser retido pelo crivo, sabemos que 3 é fator de 15 e obtemos o outro fator dividindo 15 por 3.

Definição (o problema da fatoração). Obter uma fatoração de um número natural M é obter naturais a, b, caso existam, tais que M = ab, sendo 1 < a, b < M. Diz-se que a e b são fatores não-triviais de M. Por exemplo, se $M = 2130046 = 2 \times 1031 \times 1033$, então uma fatoração de M é 2×1065023 . Outra fatoração de M é 2062×1033 e uma outra é 1031×2066 . Os naturais 1 e M são chamados de fatores triviais.

No século dezenove, Gauss nos deu uma distinção explícita entre primalidade e fatoração bem

¹O teorema fundamental da aritmética garante que todo número natural maior que 1 é primo ou pode ser decomposto unicamente em um produto de potências primas. Por exemplo, o número 3 é primo e o número 504 pode ser descomposto no produto de potências prima $2^3 \times 3^2 \times 7$. Por "unicamente" não se considera a ordem dos fatores, ou seja, a decomposição $7 \times 2^3 \times 3^2$ não é considerada diferente de $2^3 \times 3^2 \times 7$. O número 1 não é considerado primo exatamente para não violar o teorema fundamental. Fôsse primo o número 1, então $504 = 1^4 2^3 \times 3^2 \times 7$ seria uma decomposição distinta das que escrevemos anteriormente.

como a redação moderna do teorema fundamental da aritmética, conclamando a ciência para que um algoritmo eficiente² de fatoração fosse desenvolvido (Gauss 1986, seção VI, artigo 329, página 396), defendendo assim a importância do problema. Desde então, vários métodos de fatoração foram projetados³, mas nenhum eficiente apareceu. O interesse por fatoração de números grandes cresce tremendamente (Wagstaff 2013, capítulo 1, seção 1.1, página 4) durante a segunda metade da década de setenta em função da introdução da criptografia de chave pública baseada na dificuldade de se fatorar números grandes (Rabin 1979; Rivest, Shamir e Adleman 1978). Já na década de noventa, Peter W. Shor anuncia sua estratégia (Shor 1994; Shor 1997; Shor 1999) eficiente para fatoração de inteiros, mas com a interessante qualificação (Aaronson 2008, página 65) de que seria preciso um computador quântico para a eficiência da estratégia. O algoritmo de Shor tem sido considerado uma indicação de que computadores quânticos são mais rápidos que máquinas de Turing clássicas no sentido de que sejam capazes de resolver certos problemas executando menos passos que as máquinas clássicas, mesmo as máquinas de Turing probabilísticas (Nielsen e Chuang 2004, capítulo 1, seção 1.1.1, página 7) porque muitos acreditam que o problema da fatoração de inteiros tenha complexidade superpolinomial⁴ (Lenstra 1994, seção 1).

1.1 Uma descrição superficial do algoritmo de Shor

Uma descrição superficial do algoritmo de Shor pode ser dada sem referência à rotina quântica da qual o método depende. A ordem de um número $x \mod N$ é o menor inteiro positivo r tal que $x^r \equiv 1 \mod N$. Se desejamos fatorar um número natural composto N que não seja uma potência de primos, devemos escolher um número $x \mod N$, computar sua ordem r e, sendo r par e $x^{r/2} \not\equiv -1 \mod N$, então $\operatorname{mdc}(x^{r/2} - 1, N)$ revela um fator não-trivial de N porque $x^{r/2} - 1 \mod N$ compartilha algum fator comum com N. Exceto por calcular a ordem de um número $x \mod N$, todos os passos que descrevemos são calculados eficientemente na computação clássica. É o cálculo da ordem de x que depende de uma rotina quântica, o que descreveremos na Seção 4.5.

A conexão entre os fatores de N e a ordem de um número $x \mod N$ é devida a Gary Miller (Miller 1976). A contribuição de Shor foi encontrar uma rota rápida para obter a ordem. Parte das origens das ideias de Shor vem de Daniel R. Simon. Aplicando a Transformada de Fourier Quântica à ideia

²Um procedimento cuja entrada sejam os inteiros $a_1, a_2, ..., a_k$ é *eficiente* se termina em "tempo polinomial" relativo a lg $a_1, \lg a_2, ..., \lg a_k$, respectivamente, ou seja, relativo ao tamanho da entrada. O termo "tempo polinomial" significa que uma função que descreva um limite superior para a quantidade de passos é ela própria limitada superiormente por um polinômio.

³Para uma detalhada história sobre fatoração anterior a 1950, consulte "Factoring Integers Before Computers" (Williams e Shallit 1994). Para uma exposição sobre métodos de fatoração clássicos, consulte "Prime Numbers: A Computational Perspective" (Crandall e Pomerance 2005).

⁴Um algoritmo de complexidade superpolinomial é um método cuja quantidade de passos suficientes para seu término cresce mais rápido (à medida que o tamanho da entrada do algoritmo cresce) que qualquer polinômio. Em outras palavras, a quantidade de passos executados por um método superpolinomial não pode ser limitada superiormente por um polinômio. Um exemplo de algoritmo superpolinomial é a estratégia de testar todas as possibilidades no Problema do Caixeiro Viajante. Solucionar o problema significa encontrar uma rota de tamanho mínimo que passe uma única vez por todas as cidades que o caixeiro precisa passar e que retorne ao ponto de partida. Se trabalharmos com n cidades, teremos n! rotas a serem verificadas. Com 4, 5, 6, 7 cidades, temos 24, 120, 720, 5040 rotas a verificar, respectivamente. Com 8 cidades, a tarefa passa de 5040 rotas a serem verificadas para 40320 rotas a serem verificadas.

de que uma computação pode ser representada por uma árvore, Simon notou que — em suas próprias palavras — a influência mútua entre diferentes galhos de uma computação pode produzir um efeito de cancelamento de galhos alternativos da árvore, deixando apenas alguns poucos galhos da árvore como opção de resultado, sendo todos possivelmente idênticos (Simon 1994). As ideias de Simon "inspiraram" (Simon 1994) Shor a desenvolver seus métodos e a trazer a computação quântica para o centro das atenções.

O método de Shor é probabilístico; ele não atinge seu objetivo em todas as execuções. Não só porque as condições que especificamos anteriormente — a ordem de x ser par ou $x^{r/2} \not\equiv -1 \mod N$ — podem não ser satisfeitas pela escolha de x, mas também porque a rotina quântica para calcular a ordem de x não produz o valor correto todas as vezes. O que ocorre quando $x^{r/2} \equiv -1 \mod N$ é que a estratégia de Shor parece ter vindo (Shor 1994; Shor 1997; Shor 1999) da identidade

$$(x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 = 0 \mod N$$
(1.1)

e essa identidade é trivialmente satisfeita quando $x^{r/2} \equiv -1 \mod N$, o que nos leva a obter o próprio N como resposta do algoritmo e não um fator não-trivial de N. Se $\operatorname{ord}(x, N)$ é ímpar, pode-se tentar um novo x na esperança de que sua ordem seja par.

Sobre a complexidade do método de Shor, o algoritmo assintoticamente pertence a $O(\lg^3 N)$ e estimativas mais finas da complexidade foram dadas por Beckman, Chari, Devabhaktuni, Presskill (Beckman, Chari, Devabhaktuni e Preskill 1996) e também por Stephane Beauregard (Beauregard 2002) em 2002. Beckman *et al.* provam que um composto de K bits pode ser fatorado por uma máquina capaz de armazenar 5K + 1 q-bits e que o cálculo da exponenciação modular (que é o gargalo do algoritmo de Shor) pode ser realizada com aproximadamente $72K^3$ portas lógicas quânticas elementares. Beauregard apresenta um circuito usando 2n + 3 q-bits e $O(n^3 \lg n)$ portas lógicas quânticas elementares com uma profundidade⁵ de $O(n^3)$ tal que o circuito é completamente geral, ou seja, o circuito não depende de qualquer propriedade do composto a ser fatorado.

1.2 O símbolo de Legendre e o símbolo de Jacobi

O estado da arte em preparação da entrada para o algoritmo de Shor, o que descreveremos na seção a seguir, faz uso do símbolo de Jacobi, cuja definição depende do símbolo de Legendre. Definimos ambos agora.

Um resíduo quadrático $a \in Z$ é um inteiro que possui uma raiz quadrada modular, ou seja, existe um $x \in \mathbb{Z}_N^*$ tal que $a \equiv x^2 \mod N$. Similarmente, um não-resíduo quadrático é um inteiro que não possui raiz quadrada.

O símbolo de Legendre é a função L(a, p) que indica se a é resíduo quadrático módulo p, sendo p um número primo e a, um inteiro módulo p. Por definição, L(a, p) = 1 se a é resíduo quadrático

 $^{{}^{5}}$ A profundidade de um circuito é o caminho mais longo possível que a computação pode percorrer. O tamanho de um caminho é um número natural representando o número de portas que são executadas.

módulo $p \in a \neq 0 \mod p$. De outra forma, L(a, p) = -1 quando a é não-resíduo quadrático módulo p. Já se a for múltiplo de p, então L(a, p) = 0.

Uma generalização do símbolo de Legendre é o símbolo de Jacobi. Seja n um número natural ímpar e a, um inteiro módulo n. Definimos o símbolo de Jacobi pela função J(a, n) que representa o produto dos símbolos de Legendre correspondentes aos fatores primos do natural n. Por exemplo, seja $N = p_1^{e_1} \cdots p_k^{e_k}$. Então, por definição,

$$J(a,n) = L(a,p_1)^{e_1} \cdots L(a,p_k)^{e_k},$$

sendo L o símbolo de Legendre. Se n é primo, então o símbolo de Jacobi se reduz ao símbolo de Legendre.

Como no símbolo de Legendre, se J(a, n) = -1, então a é não-resíduo quadrático módulo n. Se a é resíduo quadrático módulo n, então J(a, n) = 1. Entretanto, se J(a, n) = 1, então a pode ou não ser resíduo quadrático módulo n: condição necessária e suficiente para que a seja resíduo quadrático módulo n é que a seja resíduo quadrático módulo todo fator primo de n. Assim, quando a é não-resíduo quadrático módulo um número par de fatores primos de n, então J(a, n) = 1 sem relevar se a é ou não resíduo-quadrático módulo n.

Há algoritmo eficiente na computação clássica para os símbolos de Jacobi e Legendre (Menezes, Oorschot e Vanstone 1996, capítulo 2, seção 2.4, subseção 2.4.5, páginas 72–74).

Enumeramos a seguir as contribuições e recomendações de pesquisadores que deram seguimento ao algoritmo de Shor desde sua publicação em 1994.

1.3 O estado da arte relativo ao algoritmo de Shor

Emmanuel Knill publicou em 1995 a observação de que combinando duas execuções fracassadas do algoritmo de Shor, a probabilidade de sucesso pode ser melhorada. Ele sugeriu que os fracassos da rotina quântica em calcular a ordem r de x fosse usada para estimar a ordem correta de x. A ideia é computar mmc (r_1, r_2) , sendo r_1, r_2 dois dos candidatos para a ordem de x produzidos pelo fracasso da rotina quântica usada pelo algoritmo de Shor, porque a rotina quântica às vezes produz um divisor da ordem de x e não a ordem em si. Sua segunda sugestão foi reduzir o tamanho do primeiro registrador quântica usado no algoritmo quântico de cálculo da ordem, uma sugestão que acelera a rotina quântica, mas termina por exigir uma busca mais extensiva do algoritmo de frações continuadas, que é a uma parte clássica do método de Shor para se encontrar a ordem de x, trocando assim computação quântica por computação clássica, o que é interessante, dado o maior custo de uma operação quântica comparada à clássica (Knill 1995).

Quando $\operatorname{ord}(x, N)$ é ímpar, o algoritmo de Shor desiste, podendo ser reiniciado com outro x na esperança de que a ordem da nova escolha seja par. Se N = pq, sendo p, q primos, Gregor Leander obteve condições em podemos nos certificar de que uma escolha x tenha ordem par com alta probabilidade (Leander 2002). O conselho é escolher x tal que J(x, N) = -1, sendo J(x, N) o símbolo de Jacobi de x sobre N, o que pode ser calculado em tempo polinomial num modelo clássico de computação. A sugestão levanta o limite inferior da probabilidade de sucesso de 1/2 para 3/4.

Leander provou que se p é um primo ímpar e a é um não-resíduo quadrático em Z_p^* , então a ordem de a é par. Eis por quê. Seja g um gerador de Z_p^* de forma que $a = g^s$ para algum s. Como a é não-resíduo quadrático, então s é ímpar. Seja $r = \operatorname{ord}(a, p)$. Então $a^r \equiv (g^s)^r \equiv g^{sr} \equiv 1 \mod p$. Como $\operatorname{ord}(g, p) = p - 1$, então sr = (p - 1)c para algum inteiro c. Assim, se p - 1 é par e s é ímpar, então r tem que ser par. Daí se deduz que, para todo $y \in \mathbb{Z}_N^*$, se J(y, p) = -1 ou J(y, q) = -1, então $\operatorname{ord}(y, N)$ é par. Além disso, Leander provou que a probabilidade de um número aleatório $y \in \mathbb{Z}_N^*$ satisfazendo J(y,N) = -1 ter ordem par e ainda satisfazer $y^{r/2} \not\equiv 1 \mod N$ é pelo menos 3/4, sendo $r = \operatorname{ord}(y,N)$.

Em 2013, Martín-López, Laing, Lawson, Alvarez, Zhou e O'Brien notaram que se $r = \operatorname{ord}(x, N)$ for ímpar, mas x for resíduo quadrático, então encontrar $y \in Z$ tal que $y = x^2 \mod N$ pode nos permitir extrair um fator não-trivial de N por computar $mdc(y^r - 1, N) \in mdc(y^r + 1, N)$ se N for da forma $N = p_1 p_2$, sendo $p_1, p_2 \equiv 3 \mod 4$. Martín-López *et al.* também comentam que, quando apresentamos o algoritmo de Shor na prática, a apresentação é mais convincente escolhendo-se um coprimo x cuja ordem não é uma potência de dois: tais coprimos são sensíveis a imperfeições no circuito (Martín-López et al. 2013; Smolin, G. Smith e Vargo 2013). Lawson em 2015 observou que essa estratégia de obter um fator não-trivial a partir de uma ordem ímpar não é sempre possível. Por exemplo, se tentamos fatorar N = 21 usando x = 16, que possui ord(x, 21) = 3, obtemos o fator trivial $1 = \text{mdc}(16^{3/2} + 1, 21) = \text{mdc}(16^{3/2} - 1, 21)$ porque 21 divide $16^{3/2} - 1 = 63$. Para a estratégia funcionar, N não pode dividir $x^{r/2} - 1$ nem $x^{r/2} + 1$. A publicação de Lawson de 2015 responde qual ganho é obtido na taxa de sucesso do algoritmo de Shor em decorrência dessa ideia, avisando que o ganho é pequeno. Calculando a probabilidade, Lawson assume compostos da forma $N = p_1 p_2$ com a propriedade de que $p_i - 1 = 2q_i$ sendo que q_i é ímpar. Como $2q_i$ não é múltiplo de 4, então $p_i - 1 \not\equiv 0 \mod 4$, implicando $p_i \not\equiv 1 \mod 4$. Portanto, $p_i \equiv 3 \mod 4$. Comparando seu trabalho com as contribuições de Leander, Lawson reporta que suas propostas são melhores que as de Leander em apenas um fator de $1-1/(4\sqrt{N})$, o que é minúsculo — menos de 1% quando N tem sete bits ou mais. Todavia, Thomas Lawson nos deu a indicação de que ordens ímpares poderiam ser usadas.

Anna M. Johnston notou que o algoritmo de Shor pode ser estendido para ordens ímpares e compreendeu quais critérios levam o método estendido a fatorar N com sucesso (Johnston 2017, seção 4, páginas 3–4). Seja N = AB, sendo A, B fatores não-triviais de N, primos entre si. Seja $r = \operatorname{ord}(x, N)$ e d um divisor de r. Suponha que $x^{r/d} \equiv 1 \mod A$ e $x^{r/d} \not\equiv 1 \mod B$. Então $1 < \operatorname{mdc}(x^{r/d} - 1, N) < N$. Esse resultado mostra que pode-se fatorar N encontrando um divisor primo d da ordem r desde que $x^{r/d} \equiv 1 \mod A$ e $x^{r/d} \not\equiv 1 \mod B$ ou vice-versa. O Teorema de Anna Johnston é equivalente ao teorema que apresentamos na Seção 3.1, um resultado que encontramos através da tradução do algoritmo Rô de Pollard clássico para um modelo quântico (Chicayban Bastos e Kowada 2022). Johnston não fez qualquer conexão com o algoritmo Rô de Pollard. Provavelmente em

virtude da diferente rota que nos levou ao resultado, obtivemos uma perspectiva diferente, chegando às aplicações apresentadas pelo Capítulo 5 bem como à extensão do algoritmo de Shor a ordens ímpares e a sugestão de que o algoritmo de Shor seja um caso particular da versão quântica do algoritmo Rô de Pollard. A equivalência entre os resultados é provada na Seção 4.4.

Guoliang Xu, Daowen Qiu, Xiangfu Zou, e Jozef Gruska publicaram em 2018 a observação de que se o algoritmo quântico de cálculo da ordem fracassar em produzir a ordem de x, ainda assim devemos tentar a fatoração de N computando $mdc(x^c - 1, N)$ sendo c um candidato para a ordem de x produzido pelo fracasso do algoritmo quântico de cálculo da ordem. A observação de Xu, Qiu, Zou pode ser explicada pelo teorema de Johnston: uma vez que o algoritmo quântico de cálculo da ordem às vezes produz um divisor c da ordem r de x, então pode ser o caso em que c = r/d sendo d um divisor de r e $x^c \equiv 1 \mod A$ enquanto $x^c \not\equiv 1 \mod B$ ou vice-versa.

Em 2021, Martin Ekerå mostrou que calcular a ordem de um único elemento $x \in \mathbb{Z}_N^*$ é informação suficiente para fatorar N completamente com alta probabilidade. Il
ustramos o método através de um exemplo. Suponha que queiramos fatorar $N = 2701125 = 3^2 \times 5^3 \times 7^4$. Escolha aleatoriamente um elemento de \mathbb{Z}_N^* — escolhemos 2 — e compute sua ordem pelo algoritmo quântico de cálculo da ordem, obtendo $r = \operatorname{ord}(2, N) = 102900$. Faremos agora uma busca clássica e precisamos coletar uma série de primos até um certo limite, que pode ser a quantidade de bits do composto — B = 22 para nosso exemplo. Além desses primos, precisamos também de um múltiplo r' da ordem r, que é composto pelas potências primas formadas por cada primo da série coletada, sendo que os expoentes dessas potências primas são escolhidos em função de um múltiplo m da quantidade de bits de N — escolhemos m = 22. Se q é um primo da lista, então a potência é $q^{\eta(q,m)}$, sendo $\eta(q,m)$ o maior expoente de q não maior que m. Então $r' = r \times 2^4 \times 3^2 \times 5 \times 7 \times 11 \times 13 \times 17 \times 19$. Escreva $r' = 2^t o$, sendo o impar. Agora sorteie y mod N e compute a sequência $\operatorname{mdc}(y^{o}-1, N), \operatorname{mdc}(y^{2o}-1, N), \operatorname{mdc}(y^{2^{2o}}-1, N), \operatorname{mdc}(y^{2^{3o}}-1, N), \operatorname{mdc}(y^{2^{3o$ $(1, N), \cdots, \mathrm{mdc}(y^{2^{t_o}} - 1, N)$. Computando essa sequência relativa a nosso exemplo com a primeira escolha aleatória de y = 3, obtemos mdc $(3^{o}-1, N) = 1$, mdc $(3^{2o}-1, N) = 7^{4}$, mdc $(3^{2^{2o}}-1, N) = 5^{3} \times 7^{4}$ e a partir daí não obtemos novos fatores com o y = 3. Fazemos então nova escolha aleatória — como y = 4 — e computamos mdc $(4^{o} - 1, N) = 3^{2} \times 7^{4}$, o que completa a fatoração prima de N em nosso exemplo. Em outras palavras, à medida que os fatores não-triviais aparecem, colete-os em um conjunto, verificando quando todas as potências primas de N foram encontradas. Quando uma escolha de y não for suficiente para revelar toda a fatoração, escolha outro e continue até um número limite de escolhas. Métodos clássicos são empregados para reduzir potências primas aos primos com o objetivo de obter a fatoração prima completa (Ekerå 2021).

Como descrição do estado da arte, o algoritmo de Shor não é a história toda. Resultados interessantes já foram publicados que não são baseados no algoritmo de Shor: GEECM, uma versão quântica do método de curvas elípticas usando uma curva Edward é frequentemente mais rápido que algoritmo de Shor e todos os algoritmos de fatoração pré-quânticos quando nos restringimos a inteiros de tamanho limitado. O método de Shor não é competitivo contra outros métodos que se desempenham bem em encontrar pequenos primos (Bernstein, Heninger, Lou e Valenta 2017). O estado da arte em fatoração não está apenas interessado em grandes inteiros, embora grandes inteiros sejam obviamente de grande importância dada sua utilidade em criptografia.

Capítulo 2

O algoritmo Rô de Pollard

"Elegância" é palavra que sugere graciosidade, beleza; na ciência, sugere habilidade, genialidade e particularmente simplicidade. Eric Bach bem descreveu em 1991:

In 1975, J. M. Pollard published his famous "rho" method for integer factorization. The algorithm is simple, elegant, and often used in practice when a brute-force search for divisors fails.

O algoritmo é admirável por sua simplicidade e versatilidade: o método é curto, fácil de implementar, consome uma quantidade desprezível de memória e é essencialmente uma habilidosa aplicação do algoritmo de Euclides que computa o maior divisor comum entre dois inteiros. É também extensível, sendo, por exemplo, aplicável ao problema do logaritmo discreto (Stinson e Paterson 2018, capítulo 7, seção 7.2.2, páginas 260–262). Também é misterioso: faz uso de funções cíclicas das quais sabemos tão pouco (Gathen e Gerhard 2013, capítulo 19, seção 19.4, página 548). O argumento disponível para estimar sua complexidade decorre de uma heurística em que se supõe que as funções cíclicas usadas pelo método produzem sequências aleatórias de números, sendo certo que não são aleatórias. Se substituíssemos a pseudoaleatoriedade das funções cíclicas usadas pelo algoritmo Rô de Pollard por verdadeira aleatoriedade, o algoritmo perderia sua vantagem em classe de complexidade quando comparado à divisão tentativa no mesmo modelo computacional.

2.1 A estratégia de Pollard

O algoritmo Rô (Pollard 1975) pode ser visto como uma busca por um par de números naturais que revelem um fator não-trivial do composto que se deseja fatorar, sendo que o composto não deve ser uma potência prima (Chicayban Bastos 2019, capítulo 2, seção 2.2). O universo de busca é uma sequência de números naturais módulo N de onde o algoritmo seleciona pares de números a cada iteração, sendo N o natural que se deseja fatorar. A sequência é tipicamente gerada pela iteração de um polinômio de coeficientes inteiros módulo N. Por exemplo, se desejamos fatorar $N = 3127 = 53 \times 59$, poderíamos usar o polinômio iterado $\mathbb{N} \to \mathbb{N} : f(x) = x^2 + 8 \mod 3127$ ou outros.

Produzir uma sequência por iterar um polinômio f significa escolher arbitrariamente um elemento

inicial x_0 no domínio de f e então enumerar a sequência $x_0, f(x_0), f(f(x_0)), f(f(f(x_0))), \dots$ Por exemplo, usando o polinômio $f(x) = x^2 + 1 \mod 35$ e valor inicial $x_0 = 0$, obtemos a sequência 0, 1, 2, 5, 26, 12, 5, \dots

Notação. Quando for preciso, em vez de escrevermos $f(f(f(x_0)))$, escreveremos $f^3(x_0)$, ou seja, $f^m(x_0)$ representa *m* aplicações recursivas de *f* a x_0 .



Figura 2.1: As sequências gerada por $x^2 + 8 \mod 3127$ e $x^2 + 8 \mod 53$ com $n_0 = p_0 = 2$ e uma colisão mod 53 em (p_8, p_{12}) relacionada com a colisão mod N em (n_8, n_{12}) .

Podemos representar essas sequências (produzidas pela iteração de um polinômio dado um valor inicial) por um grafo direcionado em que cada vértice x_i é um elemento da sequência e que as únicas arestas do grafo são (x_i, x_{i+1}) , conectando os vértices adjacentes x_i e x_{i+1} , sendo i = 0, 1, 2, ... Todo polinômio de coeficientes inteiros iterado $\mathbb{N} \to \mathbb{N}$: $f(x) \mod N$ produz um grafo que contém um ciclo porque a imagem de um elemento x_i pertence ao conjunto $\{0, 1, 2, ..., N - 1\}$, que é finito, o que necessariamente implica a existência de índices j > i tal que $x_j = x_i$. O fato do polinômio ser de coeficientes inteiros faz (Chicayban Bastos 2019, capítulo 2, seção 3, teorema 2.3.1, página 28) com que $x_i = x_j \mod N$ implique $x_{i+\delta} = x_{j+\delta} \mod N$ para todos os naturais $\delta \ge 0$, indicando que há uma quantidade relevante de colisões no ciclo, o que estudamos mais a fundo na Seção 2.2. Uma vez que o fechamento do ciclo não ocorre necessariamente no vértice x_0 , mas em algum vértice x_{μ} , uma representação visual do grafo nos remete à letra grega Rô, explicando o nome do algoritmo, o que ilustramos na Figura 2.1.

A Figura 2.1 exibe duas sequências rô, uma relativa ao polinômio $x^2 + 8 \mod 3127$ e outra relativa ao polinômio $x^2 + 8 \mod 53$, sendo $3127 = 53 \times 59$. Essas duas sequências possuem uma relação número-teórica que é a essência da estratégia de Pollard. Observe na figura que os números $p_8 e p_{12}$ (da sequência módulo 53) são iguais em decorrência do ciclo de tamanho 4. De posse dos índices 8 e 12, podemos considerar o par de números $\{n_8, n_{12}\} = \{615, 456\}$ na sequência módulo 3127 e computar mdc(615 - 456, 3127) = 53, obtendo um fator não-trivial de 3127. Em outras palavras, essas duas sequências possuem uma relação de onde se pode extrair informação sobre os fatores de 3127, o que estabelecemos no Teorema 2.1.1.

Teorema 2.1.1. Seja N = pq, sendo p,q primos. Seja (n_k) uma sequência de números naturais reduzidos módulo N. Seja (p_k) a sequência produzida por reduzir módulo p cada $n_i \in (n_k)$. Se $p_i = p_j \in (p_k)$, sendo $i \neq j$, então $1 < \text{mdc}(n_i - n_j, N) \leq N$, sendo $n_i, n_j \in (n_k)$.

Prova. Divida $n_i \in n_j$ por p. Pelo Algoritmo da Divisão (McCoy e Janusz 1992, capítulo 4, seção 4.4, página 98) e pela definição da sequência (p_k) , temos

$$n_i = q_1 p + p_i$$
$$n_j = q_2 p + p_j.$$

Como $p_i = p_j$, reescrevamos essas equações como

$$n_i = q_1 p + p_i$$
$$n_i = q_2 p + p_i.$$

Subtraindo essas equações, obtemos

$$n_i - n_j = (q_1 p + p_i) - (q_2 p + p_i)$$

= $q_1 p - q_2 p$
= $(q_1 - q_2)p$,

implicando que p é um fator de $n_i - n_j$. Como p é primo, então 1 < p e, logo, $1 < \text{mdc}(n_i - n_j, N) \le N$, como desejado.

No Teorema 2.1.1, os pares $\{p_i, p_j\}$ e $\{n_i, n_j\}$ são chamados de colisões. Essa terminologia vem do estudo de funções *hash* (Stinson e Paterson 2018, capítulo 5, seção 5.2, problema 5.3, página 140).

Quando h é uma função hash e x, y são elementos distintos do domínio satisfazendo h(x) = h(y), dizemos que $\{x, y\}$ é uma colisão (Stinson e Paterson 2018, capítulo 6, seção 6.6.2, páginas 213–216). Seguindo os passos de Stinson e Paterson, estendemos a terminologia por adicionar os qualificadores "trivial" e "não-trivial" às colisões relativas aos pares selecionados pelo algoritmo Rô de Pollard, o que formalizamos na Definição 2.1.2.

Definição 2.1.2 (colisão). Seja (n_k) uma sequência de números naturais reduzidos módulo N. Dado um par $\{n_i, n_j\}$ de elementos de (n_k) , se $1 < mdc(n_i - n_j, N)$, dizemos que $\{n_i, n_j\}$ é uma colisão relativa a (n_k) . Se $mdc(n_i - n_j, N) = N$, dizemos que $\{n_i, n_j\}$ é uma colisão trivial relativa a (n_k) . Se $1 < mdc(n_i - n_j, N) < N$, dizemos que $\{n_i, n_j\}$ é uma colisão não-trivial relativa a (n_k) . Quando o contexto esclarece, evitamos dizer a que sequência a colisão é relativa.

De posse desses novos termos, podemos definir o problema do algoritmo Rô de Pollard como a tarefa de encontrar uma colisão não-trivial, se uma existir, numa sequência infinita $n_0, n_1, n_2, ...$ de números naturais reduzidos módulo N produzida por uma função cíclica, como as sequências geradas pelos polinômios de coeficientes inteiros que usamos como ilustração até o momento. Esclarecemos o significado de "função cíclica" na Definição 2.1.3.

Definição 2.1.3 (função cíclica). Dizemos que uma função $\mathbb{N} \to \mathbb{N} : g(i)$ é cíclica se existir inteiro λ tal que $g(i) = g(i + \lambda)$ implique $g(i + k) = g(i + k + \lambda)$ para todo $k \ge 0$.

A Figura 2.1 sugere que uma colisão módulo 53 nos revela uma colisão módulo 3127, o que é garantido pelo Teorema 2.1.1. Entretanto, quando buscamos pelos fatores não-triviais de 3127, não temos os números 53 e 59 disponíveis, logo não temos as sequências módulo 53, 59 disponíveis onde realizar uma busca por colisões. Sem alternativas disponíveis, o algoritmo Rô de Pollard recorre a escolher pares de números da sequência módulo N que possam eventualmente produzir uma colisão não-trivial. Mas o método não pode buscar por uma colisão não-trivial sem um limite superior de iterações porque nem sempre uma colisão não-trivial existe no universo de busca¹. Já uma colisão trivial sempre existe; por exemplo, se λ é o tamanho do ciclo e μ o tamanho da cauda, então $\{n_i, n_{i+\lambda}\}$ é sempre uma colisão trivial porque $n_i = n_{i+\lambda}$ implica mdc $(n_i - n_{i+\lambda}, N) = mdc(0, N) = N$. Podemos então definir que a estratégia desista de encontrar um fator não-trivial assim que ela encontrar uma colisão trivial, ou seja, o método termina fracassado, podendo ser reiniciado com uma nova sequência, seja por via de um novo polinômio, uma outra função cíclica ou um novo valor inicial, fazendo com que a estratégia seja probabilística, embora sua probabilidade de fracasso seja desprezível (Menezes, Oorschot e Vanstone 1996, capítulo 3, seção 3.2.2, página 91).

Comentário. A letra grega μ é tipicamente usada para descrever o índice da sequência onde o ciclo começa, e a letra λ é similarmente usada para representar o tamanho do ciclo, mas há quem inverta os papéis de μ e λ , acidentalmente que seja (Menezes, Oorschot e Vanstone 1996, capítulo 3, nota 3.8, página 91).

¹Um exemplo em que todas as colisões são triviais é N = 3551 e $f(x) = x^2 + 8 \mod 3551$ com valor inicial $x_0 = 38$.

Especificada uma condição de término, precisamos também de um método de seleção de pares. Um sorteio aleatório por pares de elementos $\{x_i, x_j\}$ da sequência módulo N não seria interessante porque a sequência módulo N é gerada pela iteração de uma função cíclica como $x^2 + c \mod N$ cuja fórmula fechada não se encontra facilmente. Sem uma fórmula fechada, computar o *i*-ésimo elemento da sequência requer a computação de todos os elementos anteriores $x_{i-1}, x_{i-2}, ..., x_0$. Em outras palavras, não é barato computar um elemento arbitrário da sequência. Uma estratégia trivial de seleção de

não é barato computar um elemento arbitrário da sequência. Uma estratégia trivial de seleção de pares seria iterar a função cíclica e verificar se cada n_i , sendo i = 0, 1, 2, ..., produzido pela função iterada pertence a uma lista L inicialmente vazia. Se a verificação for negativa, armazenamos n_i em L e repetimos o procedimento. Se a verificação for positiva, encontramos $n_i = n_j$ sendo $i \neq j$, ou seja, encontramos a colisão $\{n_i, n_j\}$. Se a colisão for trivial, encerramos com fracasso; se for não-trivial, terminamos com sucesso. Buscar a colisão $\{n_i, n_j\}$ em L pode ser feito de forma eficiente ordenando-se os valores n_i em uma lista K e usando-se a busca binária para detectar uma colisão, garantindo-nos não mais que $\Theta(\lg |K|)$ comparações (Stinson e Paterson 2018, capítulo 5, seção 5.2.2, algoritmo 5.3, página 144). Entretanto, o espaço necessário para a lista K cresce linearmente com N e é possível evitar todo esse consumo de espaço usando um resultado devido a Robert W. Floyd (Knuth 1997, capítulo 3, exercício 6b, página 7), permitindo-nos essencialmente manter apenas dois elementos da sequência em memória em cada iteração.

Usando a estratégia de Floyd, Pollard em 1975 efetivamente nos instruiu a manter dois índices t = i, c = 2i em memória, sendo i = 1, 2, 3, ... Usando a letra t como o propósito de sugerir o trajeto de uma tartaruga ao longo da sequência módulo N e, similarmente, usando c para sugerir o trajeto de um coelho, possuindo o dobro da velocidade da tartaruga, é um fato (Chicayban Bastos 2019, capítulo 2, seção 2.1, teorema 2.1.3) que o coelho colide com a tartaruga num índice que é múltiplo do tamanho do ciclo, o que nos permite obter uma condição de parada.

Analisemos o método de Floyd em mais detalhes. Suponha que λ seja o tamanho do ciclo e μ , o tamanho da cauda da sequência. Até o início do ciclo, o coelho e a tartaruga ultrapassam μ elementos da sequência. É certo que eles se encontram dentro do ciclo porque fora do ciclo o coelho está sempre à frente da tartaruga, exceto pela posição inicial de largada. Seja k a distância entre o início do ciclo e o elemento da sequência onde o coelho e a tartaruga se encontram. Seja V_c o número natural que descreve a quantidade de voltas dada no ciclo pelo coelho e, similarmente, V_t para a tartaruga. Podemos dizer que o coelho percorreu $\mu + \lambda V_c + k$ elementos da sequência e que a tartaruga percorreu $\mu + \lambda V_t + k$. Como ambos encontram-se no mesmo elemento e a velocidade do coelho é duas vezes a velocidade da tartaruga, deduzimos $\mu + \lambda V_c + k = 2(\mu + \lambda V_t + k)$, implicando

$$\mu + k = \lambda (V_c - 2V_t) = \lambda M, \tag{2.1}$$

sendo $M = V_c - 2V_t$. Como assumimos que $\mu + k$ é o índice do elemento da sequência onde o coelho e tartaruga se encontram, então a Equação 2.1 nos informa que eles se encontram em um índice da sequência que é múltiplo de λ , já que M é um número natural. Assim, Floyd nos garante que se percorrermos a sequência observando-se apenas os índices i e 2i para i = 1, 2, 3, ..., eventualmente esses índices descreverão uma colisão. Com essa garantia em mãos, esboçamos uma expressão da estratégia de Pollard no Algoritmo 1.

Algoritmo 1. O algoritmo Rô de Pollard com polinômio f(x) e valor inicial x_0 . O procedimento assume que N não é potência prima.

```
algoritmo POLLARD(N, f, x_0):

t \leftarrow c \leftarrow x_0

repita

t \leftarrow f(t)

c \leftarrow f(f(c))

d \leftarrow mdc(t - c, N)

se d = N então

retorne "sem resposta"

fim se

se 1 < d \in d < N então

retorne d

fim se

fim repita

fim algoritmo
```

Comentário. A estratégia de Floyd não parece ser ótima. Algoritmos provavelmente mais rápidos que o de Floyd foram investigados, tendo sido encontrado algoritmos de detecção de ciclos aproximadamente 36% mais rápidos que o de Floyd, provocando um ganho de performance de 24% na fatoração quando usado com o algoritmo Rô de Pollard, sendo ambos percentuais relativos ao caso médio (Brent 1980).

2.2 A importância de uma função cíclica

É importante para o algoritmo Rô de Pollard que a função iterada usada seja cíclica (Definição 2.1.3) porque, se $x_i = x_j \mod N$, sendo $x_i \in x_j$ elementos da sequência gerada pela função cíclica iterada, então $x_{i+\delta} = x_{j+\delta} \mod N$ para todos os naturais $\delta \ge 0$. Usemos a Figura 2.1 para ilustrar a importância da função cíclica para o algoritmo Rô de Pollard. Por exemplo, o par $\{n_8 = 615, n_{12} = 456\}$ é uma colisão não-trivial e, como $n_8 \in n_{12}$ estão ambos dentro do ciclo, então também são colisões nãotriviais os pares $\{n_9, n_{13}\}$, $\{n_{10}, n_{14}\}$, $\{n_{11}, n_{15}\}$, $\{n_{12}, n_{16}\}$, $\{n_{13}, n_{17}\}$, $\{n_{14}, n_{18}\}$. Como o método deseja encontrar qualquer colisão não-trivial, o fato de sempre haver muitas quando existe uma reduz, em média, a quantidade de tentativas necessárias para encontrá-la.

As colisões não-triviais que enumeramos acima — $\{n_8, n_{12}\}, \{n_9, n_{13}\}, \{n_9, ...\}$ — são formadas por elementos da sequência que estão a quatro passos de distância, ou seja, há quatro arestas entre n_8 e n_{12} . Podemos também encontrar colisões não-triviais na Figura 2.1 a cada três passos; por exemplo, $\{n_8, n_{11}\}$ é uma colisão não-trivial, implicando que também o são os pares $\{n_9, n_{12}\}, \{n_{10}, n_{13}\}, ...$ As Quatro e três são os tamanhos dos ciclos gerados pela função cíclica reduzida módulo 53 e módulo 59, respectivamente, indicando que quanto menor for um dos ciclos gerados pela função cíclica reduzida módulo p, sendo p um divisor primo de N, mais colisões não-triviais haverá. Por isso o algoritmo Rô de Pollard é parte do estado-da-arte em fatoração clássica: quando N não é tão pequeno a ser fatorado pela divisão tentativa, mas possui um primo p relativamente pequeno, estão presentes no universo de busca do algoritmo Rô de Pollard colisões não-triviais a cada λ_p passos do ciclo, sendo λ_p o tamanho do ciclo módulo p, que tende a ser pequeno já que p é pequeno. Estudamos essa relação entre os tamanhos dos ciclos gerado pela função iterada na Seção 2.4 e mais a fundo no Capítulo 3.

Polinômios de coeficientes inteiros são funções cíclicas, sendo alguns mais baratos de se computar que outros. Em implementações, é comum a escolha do polinômio ter a forma a forma $x^2 + c$, sendo c uma constante, excluindo-se c = 0 e c = -2 (Cormen, Leiserson, Rivest e Stein 2009, capítulo 31, seção 31.9, página 980) (Koblitz 1994, capítulo V, seção 2, exercício 7, página 143) (Menezes, Oorschot e Vanstone 1996, capítulo 3, seção 3.2.2, nota 3.12, página 92) (Pollard 1975, página 333). Polinômios de grau maior que 2 seriam mais caros de se computar por envolverem exponenciações. Também não se sabe mais sobre outros polinômios do que, por exemplo, x^2+1 (Gathen e Gerhard 2013, capítulo 19, seção 19.4, página 548). O caso c = 0 não deve ser usado porque se $f(x_i) = 0$, então a sequência a partir daí produz 0 a cada índice, fechando um ciclo de tamanho 1 (Knuth 1997, capítulo 3, seção 3.2.1.2, página 19). O caso c = -2 sofre a mesma dificuldade: se $f(x_i) = 2$, então $x_{i+1} = x_i^2 - 2 = 2$, fechando um ciclo de tamanho 1.

2.3 A complexidade do algoritmo

O algoritmo termina quando uma colisão qualquer é encontrada envolvendo a tartaruga e o coelho. Enquanto a tartaruga não entra no ciclo, o coelho está à frente dela, sem chance de alcançá-la. Logo, o coelho não pegará a tartaruga em menos de μ iterações. Depois que a tartaruga entra no ciclo, o coelho deixa de estar à frente dela e passa então a persegui-la, reduzindo-se a distância entre eles em uma unidade a cada iteração. Percorre-se o ciclo completamente em λ iterações, logo, eles se encontram em $\mu + \lambda$ iterações no pior caso (Chicayban Bastos 2019, capítulo 2, seção 2.1, páginas 21–22), gerando um custo de $\mathcal{O}(\mu + \lambda)$ operações no pior caso.

Entretanto, uma colisão não-trivial pode ocorrer antes do coelho encontrar a tartaruga. Em média, quantas iterações ocorrem até a obtenção de uma colisão não-trivial, assumindo que uma exista? É com isso que nos preocupamos agora.

Enquanto o coelho e a tartaruga percorrem a sequência módulo N, podemos imaginar que ambos também percorrem as sequências módulo $p \in q$, assumindo-se que N = pq, sendo p, q ambos primos. Assim que o coelho encontra a tartaruga na sequência módulo p ou módulo q, uma colisão ocorre na sequência módulo N, o que é garantido pelo Teorema 2.1.1. Em média, quanto menor é o ciclo módulo p ou módulo q, menor é a quantidade de iterações necessárias para uma colisão: quanto menor é o ciclo módulo p, mais rapidamente o coelho tende a pegar a tartaruga no ciclo módulo p. Assim, assumindo que p seja o menor fator primo de N, então o número médio de iterações até a ocorrência de uma colisão é função de p, explicando por que a estratégia se desempenha bem quando o composto possui pequenos fatores primos. Uma estimativa da dificuldade de se encontrar uma colisão para uma função hash arbitrária é provida pelo problema do aniversário (Stinson e Paterson 2018, capítulo 5, seção 5.2.2, páginas 143–145), o que é aplicável ao caso médio do algoritmo Rô de Pollard, assumindo-se que uma função hash ideal seja um gerador de números aleatórios. Na estimativa do caso médio do algoritmo Rô de Pollard, a função iterada toma o lugar da função hash. A cada iteração, um par de números da sequência módulo N é "sorteado" e verificamos por uma colisão. É certo que os polinômios usados no algoritmo Rô de Pollard não produzem uma sequência aleatória de números e, por isso, a estimativa que se consegue fazer hoje é um argumento heurístico, não uma prova matemática. "Very little is known in a rigorous sense about why [Pollard's Rho] works" (Bach 1991, section 1, página 139).

Extraindo-se Q números naturais com reposição a partir do conjunto $\{1, 2, 3, ..., M\}$, a probabilidade aproximada de se encontrar uma colisão entre esses números pode ser expressa por (Chicayban Bastos 2019, capítulo 2, seção 2.4, teorema 2.4.2)

$$P(Q) \approx 1 - \exp\left(-\frac{Q^2 - Q}{2M}\right)$$

Se se deseja uma probabilidade x de se encontrar uma colisão, então a quantidade média Q de extrações necessárias para obter uma colisão é aproximadamente (Chicayban Bastos 2019, capítulo 2, seção 2.4, teorema 2.4.3)

$$Q \approx \sqrt{2M \ln \frac{1}{1-x}}.$$

Pedindo-se por uma probabilidade x = 1/2, obtemos $Q \approx \sqrt{2(\ln 2)M} \in \mathcal{O}(\sqrt{M})$, mas M é a quantidade de números possíveis no ciclo módulo p, que é p - 1, assumindo que o composto a ser fatorado seja da forma N = pq, sendo p, q primos. Logo, após a extração de $\mathcal{O}(\sqrt{p-1}) = \mathcal{O}(p^{1/2})$ números, há uma probabilidade 1/2 de se encontrar uma colisão. Os compostos usados na criptografia RSA hoje tendem a ser da forma N = pq, sendo p, q de tamanhos da ordem da raiz quadrada de N, logo a complexidade de caso médio do algoritmo Rô de Pollard pertence a $\mathcal{O}(N^{1/4})$.

Comentário. A estimativa $\mathcal{O}(N^{1/4})$ se refere à quantidade de extrações de números oriundos da função f iterada pelo algoritmo. Verificamos por uma colisão a cada par de números obtidos, o que custa três invocações de f. Em outras palavras, enquanto $\mathcal{O}(N^{1/4})$ é uma expressão da quantidade média de números efetivamente extraídos, $(1/2)\mathcal{O}(N^{1/4})$ é a quantidade média de colisões verificadas e $(3/2)\mathcal{O}(N^{1/4})$ é o custo médio incorrido por f.

2.4 Uma nova perspectiva sobre o método

Se estudarmos o tamanho do ciclo da sequência módulo N, obteremos uma nova perspectiva sobre a estratégia de Pollard. Já sabemos por que há informação sobre os fatores de N na sequência percorrida pelas iterações do Algoritmo 1, mas não temos ainda uma caracterização dos índices da sequência módulo N que produzem pares que relevam um fator não-trivial de N. Em outras palavras, quais são os pares de elementos da sequência módulo N que são os desejáveis pelo algoritmo Rô de Pollard? Uma caracterização desses pares é obtida pelo Teorema 3.1.5 e a investigação que faremos agora nos serve como uma introdução ao resultado.

A Figura 2.1 exibe um ciclo de tamanho $\lambda = 12$ e um ciclo de tamanho $\lambda_{53} = 4$. Se desenhássemos o grafo da sequência módulo 59, encontraríamos um ciclo de tamanho $\lambda_{59} = 3$. Um resultado bem conhecido (provado no Lema 3.1.4) é que o tamanho do ciclo módulo N é o menor múltiplo comum entre os ciclos λ_{53} e λ_{59} , ou seja, $12 = \lambda = \text{mmc}(\lambda_{53}, \lambda_{59}) = \text{mmc}(4, 3)$. A Figura 2.1 apresenta sequências que possuem uma cauda; a investigação que desejamos fazer neste momento será mais facilmente observada se trabalharmos com sequências sem cauda, então usaremos um novo exemplo.

Considere o polinômio $x^2 + 3 \mod 143 = 11 \times 13$. Se iterarmos esse polinômio usando o valor inicial 4, obteremos sequências módulo 143, 11, 13 sem caudas, o que nos permite expô-las na tabela abaixo de forma que a primeira coluna de cada uma é o início do ciclo de cada uma, o que nos facilita observar as colunas em que se localiza o último elemento de cada ciclo, representado na tabela pelo sublinhamento.



Figura 2.2: O polinômio $x^2 + 3 \mod 143 = 11 \times 13$ iterado em sequências módulo 143, 11 e 13, respectivamente da esquerda para a direita. A escolha do valor inicial $x_0 = 4$ produz grafos sem caudas.

Na tabela abaixo, a coluna 1 representa o primeiro elemento do ciclo de cada sequência. A razão por que exibimos quinze colunas é que o ciclo da sequência módulo 143 tem tamanho 15. Em cada linha, sublinhamos o último número do ciclo de cada sequência. Por terem tamanhos diferentes, os ciclos módulo 11 e 13 se fecham em colunas diferentes, mas eventualmente se fecham juntos quando chegamos a uma coluna de número igual ao menor múltiplo comum entre os tamanhos dos ciclos módulo 11 e 13.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
4	19	78	81	129	56	136	52	133	103	30	45	26	107	<u>12</u>	 mod	143
4	8	<u>1</u>	4	8	<u>1</u>	4	8	<u>1</u>	4	8	<u>1</u>	4	8	<u>1</u>	 mod	11
4	6	0	3	<u>12</u>	4	6	0	3	<u>12</u>	4	6	0	3	<u>12</u>	 mod	13

O Teorema 2.1.1 informa que uma colisão módulo 143 é encontrada quando temos, por exemplo, uma colisão módulo 11. Onde estão as colisões módulo 11 na tabela? Elas estão em, por exemplo, no par de colunas (2, 5) porque o número 8 aparece em ambas colunas; outra é obtida no par de colunas (3, 6) porque o número 1 aparece nas duas; outra é obtida em (7, 13) porque o número 4 aparece em ambas. Quais são os números da sequência módulo 143 dessas colunas-colisão módulo 11? Relativo ao par (2, 5), obtemos os números 19 e 129, o que nos revela o fator não-trivial 11 = mdc(19 - 129, 143).

Eis uma interpretação gráfica dessa tabela. Cada barra abaixo representa uma linha na tabela. O ciclo módulo 143 tem tamanho quinze e é representado pela primeira barra λ . As outras barras representam os ciclos módulo 11 e módulo 13, respectivamente, sendo que justapomos as barras menores de forma que obtenhamos um tamanho quinze para cada uma delas — uma forma geométrica de computar o menor múltiplo comum. O ciclo módulo 11 tem tamanho três e é representado pela barra λ_{11} . O ciclo módulo 13 tem tamanho cinco e é representado pela barra λ_{13} .

$\lambda = 15$								
$\lambda_{11} = 3$								
$\lambda_{13} = 5$								

Figura 2.3: Uma representação gráfica dos ciclos módulo 11 e módulo 13. Justapomos os ciclos menores até cada um deles alcançar o tamanho $\lambda = 15$, que é o tamanho do ciclo da sequência na qual o algoritmo Rô de Pollard faz sua busca por uma colisão não-trivial.

Na página 16, perguntamos quais eram os pares de elementos da sequência módulo N que são os desejáveis pelo algoritmo Rô de Pollard? Usando a representação gráfica da Figura 2.3, podemos responder que os pares desejáveis são aqueles demarcados pelas extremidades das barras $\lambda_{11} e \lambda_{13}$ que não estão emparelhados. O emparelhamento ocorre a cada múltiplo de 15, quando então obteríamos uma colisão trivial.

Não é coincidência que o fator não-trivial 11 seja revelado pelo algoritmo de Euclides em vez do fator não-trivial 13. Se desejarmos pelo fator não-trivial 13, precisaremos de uma colisão módulo 13, o que encontramos, por exemplo, em (1, 6), (2, 7), (2, 12), (1, 11), (3, 8) et cetera. Tomando o par (3, 8) como exemplo, obtemos os números 78 e 52 da sequência módulo 143, o que nos revela o fator não-trivial 13 = mdc(78 - 52, 143).

Já quando tomamos as colunas (1, 16), por exemplo, devemos observar que são colunas distantes uma da outra em 15 unidades, sendo 15 um múltiplo comum entre os tamanhos dos ciclos módulo 11 e 13. Ao localizarmos os números da sequência módulo 143 relativos às colunas 1 e 16, observamos que eles são o número 4, ou seja, são iguais, o que implica em obtermos o fator trivial 143 = mdc(4-4, 143). Em outras palavras, os pares em que estamos em busca são os pares formados por colunas afastadas uma da outra por um múltiplo do tamanho do ciclo módulo 11 que não seja múltiplo do tamanho do ciclo módulo 13. Alternativamente, serve a nós também colunas afastadas uma da outra por um múltiplo do tamanho do ciclo módulo 13 que não seja múltiplo do tamanho do ciclo módulo 11. O converso dessa relação não produz uma colisão não-trivial, o que nos dá uma caracterização da posição dos índices em que encontramos as colisões não-triviais.

Na página 11, definimos o problema do algoritmo Rô de Pollardcomo a tarefa de encontrar uma colisão não-trivial, se uma existir. Com a nova perspectiva dada pelo Teorema 3.1.5, é interessante redefinirmos o problema como a tarefa de encontrar um divisor m do tamanho do ciclo módulo N = pq que seja múltiplo do ciclo módulo p e que não seja múltiplo do ciclo módulo q (ou vice-versa), sendo ambos p, q primos. De posse de m, obtemos um fator não-trivial computando mdc $(x - f^m(x), N)$, sendo x um elemento do ciclo da sequência módulo N. A computação quântica nos oferece em tempo polinomial o período de funções periódicas. A partir do período λ , podemos procurar por um divisor primo d de λ e experimentar mdc $(x - f^{\lambda/d}, N)$. Se λ/d for múltiplo do tamanho do ciclo módulo p e não for múltiplo do tamanho do ciclo módulo q, encontraremos um fator não-trivial de N. A seguir, estabelecemos o resultado e exibimos uma expressão de uma versão quântica da estratégia.

Capítulo 3

Uma versão quântica do Rô de Pollard

Em nossa dissertação de mestrado (Chicayban Bastos 2019), produzimos uma primeira tradução do algoritmo Rô de Pollard para um modelo quântico de computação. Lá, tivemos a esperança de aplicar a Transformada de Fourier Quântica (Nielsen e Chuang 2004, capítulo 5) ao universo de busca do problema e obter um ganho exponencial, mas encontramos dificuldades. Recorremos então ao plano alternativo de aplicar a busca quântica (Grover 1996), melhorando o método apenas pelo fator quadrático já esperado. Naquele mesmo ano, Bernstein, Heninger, Lou and Valenta traduziram o método de fatoração clássico de curvas elípticas por aplicar o algoritmo de Grover e os resultados foram encorajadores (Bernstein, Heninger, Lou e Valenta 2017). Outras dificuldades técnicas na implementação nos provocaram um atraso de um fator quadrático também, anulando o ganho obtido com o método de Grover. Escrevemos na dissertação que nosso trabalho poderia resultar numa "redescoberta do algoritmo de Shor" (Chicayban Bastos 2019, capítulo 5, página 84), o que se mostrou verdadeiro. O Teorema 3.1.5 caracteriza as soluções do Problema do algoritmo Rô de Pollard em seu universo de busca, levando-nos a uma versão quântica do método bem como a percepção de que o algoritmo de Shor é um caso particular dele, o que será apresentado na Seção 4.3.

3.1 Uma caracterização das colisões não-triviais

A asserção de unicidade dada pelo teorema fundamental da aritmética (Gauss 1986, seção II, artigo 16, página 6) garante que se dois números naturais têm fatoração prima idêntica, então eles são o mesmo número. Se não são os mesmos números, tem que haver pelo menos uma potência prima que apareça na fatoração prima de um deles e não na do outro. Em outras palavras, dois números diferentes podem ser distinguidos por uma potência prima. As Definições 3.1.1–3.1.2 e o Exemplo 3.1.3 formalizam e ilustram essa noção.

Definição 3.1.1. Seja e(p,N) o maior expoente do número primo p que divide N, ou seja, enquanto $p^{e(p,N)}$ divide N, a potência $p^{e(p,N)+1}$ não divide N. Se p não aparece na fatoração prima de N, definimos que e(p,N) = 0.

Definição 3.1.2 (primo distintivo). Sejam M, N dois números naturais tal que $M \neq N$. Chamamos

de potência prima distintiva relativa a M qualquer fator $p^{e(p,M)}$ de M tal que $e(p,M) \neq e(p,N)$. Similarmente, dizemos que p é um primo distintivo relativo a M, N.

Exemplo 3.1.3. Seja $N = 2 \cdot 3^2 \cdot 5^2$, $M = 2 \cdot 3^0 \cdot 7$. Então $p^{e(p,N)} = 3^2$ é fator de N, sendo 2 o maior expoente de 3 tal que 3^2 divide N e e(3,M) = 0 porque 0 é o maior expoente de 3 tal que 3^0 divide M. Podemos distinguir N de M pelo fato de que 3^2 aparece na fatoração prima de N mas não na fatoração prima de M.

Para obtermos o Teorema 3.1.5, precisaremos do Lema 3.1.4, que é o resultado que estabelece o tamanho do ciclo módulo N como o menor múltiplo comum entre os tamanhos dos ciclos módulo p, q. Provamos uma generalização do resultado para naturais A, B primos entre si cujo produto seja N.

Lema 3.1.4. Seja N = AB, sendo A, B fatores não-triviais de N e primos entre si. Dada uma função $f: S \to S$, sendo S um subconjunto finito não-vazio dos números naturais, seja λ o tamanho do ciclo da sequência (n_k) gerada pela regra $n_{i+1} \equiv f(n_i) \mod N$ para todo $i \ge 1$, sendo arbitrado como valor inicial um elemento em S. Então $\lambda = \operatorname{mmc}(\lambda_A, \lambda_B)$, sendo $\lambda_A \ e \ \lambda_B$ os tamanhos dos ciclos de suas sequências correspondentes obtidas por reduzir cada elemento $n_k \in (n_k)$ módulo A and B, respectivamente.

Prova. Fixe um elemento x no ciclo de (n_k) . Por definição, λ é o menor inteiro positivo tal que $f^{\lambda}(x) \equiv x \mod N$, implicando $f^{\lambda}(x) - x \equiv 0 \mod N$. Em outras palavras, $f^{\lambda}(x) - x$ é múltiplo de N. Uma vez que N = AB, qualquer múltiplo de N é múltiplo de ambos A e B, ou seja, $f^{\lambda}(x) - x \equiv 0 \mod A$, B, implicando $f^{\lambda}(x) \equiv x \mod A$, B, o que deixa claro que λ é múltiplo comum de ambos λ_A e λ_B . Falta mostrar que λ é o menor múltiplo.

Por contradição, suponha que λ não seja o menor múltiplo comum entre λ_A e λ_B . Então existe um inteiro positivo $\mu < \lambda$ tal que μ é múltiplo de ambos λ_A e λ_B . Isso significa que $f^{\mu}(x) \equiv x \mod A, B$, o que implica $f^{\mu}(x) - x \equiv 0 \mod A, B$. Mas essa última congruência implica que $f^{\mu}(x) - x$ é múltiplo de N também, então $f^{\mu}(x) - x \equiv 0 \mod N$ implicando $f^{\mu}(x) \equiv x \mod N$. Em outras palavras, existe um inteiro positivo $\mu < \lambda$ tal que $f^{\mu}(x) \equiv x \mod N$, violando a hipótese de que λ é o menor inteiro positivo com essa propriedade. Logo, μ não existe e $\lambda = \text{mmc}(\lambda_A, \lambda_B)$, como desejado.

Agora estamos equipados para o Teorema 3.1.5.

Teorema 3.1.5 (uma caracterização de colisões não-triviais). Seja N = AB, sendo A, B fatores não-triviais de N e primos entre si. Seja (n_k) a sequência infinita gerada pela função cíclica g(k)que mapeia os números naturais 0, 1, 2, ... aos elementos $n_k \in \mathbb{Z}_N^*$. Seja (a_k) e (b_k) as sequências infinitas geradas por reduzir-se módulo A, B cada elemento $n_k \in (n_k)$. Então $\lambda_A \neq \lambda_B$ se e somente se existir um número natural $m < \lambda$ tal que m seja múltiplo de λ_A e m não seja múltiplo de λ_B , ou vice-versa, sendo λ o tamanho do ciclo da sequência (n_k) e λ_A, λ_B , o tamanho de suas correspondentes sequências. Além disso,

$$1 < \mathrm{mdc}(g(o+m) - g(o), N) < N$$

para qualquer elemento g(o) no ciclo de (n_k) .

Prova. A implicação conversa é facilmente provada observando que se houver número natural $m < \lambda$ tal que m seja múltiplo de λ_A e m não seja múltiplo λ_B , então $\lambda_A \neq \lambda_B$. (Se m é múltiplo de λ_B e não múltiplo de λ_A , então novamente $\lambda_A \neq \lambda_B$.)

Provamos agora a implicação direta. Precisamos mostrar que (1) um certo número natural mexiste se $\lambda_A \neq \lambda_B$ e também (2) que m satisfaz 1 < mdc(g(o + m) - g(o), N) < N para qualquer elemento g(o) fixo do ciclo de (n_k) gerado por g. Para provar a existência de m, podemos tomar o caminho em que (a) mostramos que existe um natural $m < \lambda$ tal que m é múltiplo de λ_A e m não é múltiplo de λ_B , ou podemos tomar o caminho em que (b) mostramos que m não é múltiplo de λ_A e m

Seja $\lambda_A \neq \lambda_B$. Pelo Lema 3.1.4, $\lambda = \text{mmc}(\lambda_A, \lambda_B)$. Escreva

$$\lambda_A = \prod_p p^{e(p,\lambda_A)}$$
 e $\lambda_B = \prod_p p^{e(p,\lambda_B)}$

Seja q um primo distintivo relativo a λ_A , λ_B no sentido da Definição 3.1.2. Sem perda de generalidade, assuma $e(q, \lambda_A) < e(q, \lambda_B)$. Seja $m = \lambda/q$. Por construção, $m < \lambda$. Dado que q é um primo distintivo relativo a λ_A , λ_B e que $e(q, \lambda_A) < e(q, \lambda_B)$, então $e(q, m) = e(q, \lambda_B) - 1$ porque $m = \lambda/q$ e $\lambda = \text{mmc}(\lambda_A, \lambda_B)$, implicando que m não é múltiplo de λ_B . Na fatoração prima de λ_A , entretanto, o maior valor possível para $e(q, \lambda_A)$ é $e(q, \lambda_B) - 1$, então m é múltiplo de λ_A , como desejado. Falta mostrar que 1 < mdc(g(o + m) - g(o), N) < N.

Fixe um elemento g(o) no ciclo de (n_k) gerado por g. Por definição, λ é o menor inteiro positivo tal que $g(o+\lambda) \equiv g(o) \mod N$. Uma vez que $m < \lambda$, então $g(o+m) \not\equiv g(o) \mod N$: de outra forma mseria o tamanho do ciclo de (n_k) . Então, $g(o+m) - g(o) \not\equiv 0 \mod N$, implicando que g(o+m) - g(o)não é múltiplo de N. Já estabelecemos que m é múltiplo de λ_A , logo $g(o+m) \equiv g(o) \mod A$, o que implica que g(o+m) - g(o) é múltiplo de A. Assim, g(o+m) - g(o) é múltiplo de A mas não é múltiplo de N, de onde obtemos

$$1 < \mathrm{mdc}(g(o+m) - g(o), N) < N,$$

como desejado.

A prova do Teorema 3.1.5 sugere que podemos fatorar N desde que possamos encontrar o número m quando ele existir. A versão clássica do algoritmo Rô de Pollard procura por uma colisão não-trivial através da escolha de pares de números que estejam no ciclo da sequência gerada por um polinômio de coeficientes inteiros. Podemos substituir a busca executada pela versão clássica por uma busca clássica pelo número m antecedida pelo cálculo do tamanho do ciclo da sequência (n_k) , o que pode ser feito em tempo polinomial por um computador quântico. Em outras palavras, o Teorema 3.1.5 sugere uma versão quântica do algoritmo Rô de Pollard.

3.2 O paralelismo quântico

Se traduzíssemos para o modelo quântico a versão clássica do algoritmo Rô de Pollard de uma forma direta, não seria óbvio como tirar vantagem do paralelismo quântico porque, por exemplo, o polinômio $x^2 + 1 \mod N$, tipicamente usado na prática, não nos oferece uma fórmula fechada¹ fácil de encontrar. O objetivo desta seção é mostrar por que o paralelismo quântico exige uma fórmula fechada, mas antes precisamos introduzir um mínimo dos elementos básicos usados na computação quântica.

Um bit quântico — também escrito "q-bit" — é representado matematicamente como um vetor. A notação comumente usada pela mecânica quântica é a notação de Paul Dirac (Nielsen e Chuang 2004, capítulo 1, seção 1.2, página 13). O objetivo de Dirac é que o símbolo $|\cdot\rangle$ sirva como uma função, cujo argumento seja a descrição de um vetor. "If, for example, we were talking about displacement vectors in ordinary three-dimensional space, we could have a vector |5 horizontal centimeters northeast \rangle " (Mermin 2007, capítulo 1, seção 1.2, página 5). Dessa forma, $|\psi\rangle$ descreve um vetor expresso por seus componentes numa certa base como um vetor-coluna, ou seja,

$$|\psi\rangle = \begin{bmatrix} a_1\\a_2\\\vdots\\a_n \end{bmatrix}.$$

Assim, a expressão $U|\psi\rangle$ representa o produto da matrix U pelo vetor $|\psi\rangle$. Por ser um vetor, um q-bit pode ser escrito como uma combinação linear dos vetores da base do espaço vetorial, o que permite ao q-bit a possibilidade de estar em uma "superposição" — como assim é tipicamente chamado — dos vetores da base, o que não é possível no modelo clássico já que o bit clássico é o número zero ou o número um.

Operadores quânticos muito usados na computação quântica recebem seus próprios nomes. Por exemplo, o operador Hadamard — em homenagem a Jacques S. Hadamard — é comumente representado por H. Aplicando H ao q-bit $|0\rangle$, obtemos o estado $1/\sqrt{2}(|0\rangle + |1\rangle)$, que é também chamado de estado de Bell — em homenagem a John S. Bell — e é tipicamente descrito por $|+\rangle$. Em símbolos, $|+\rangle := H|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$.

Quando fazemos a leitura de um q-bit no estado $|+\rangle$, obtemos o bit clássico 0 em aproximadamente 50% das vezes. A probabilidade de se obter o bit clássico zero é dada pelo quadrado do coeficiente² associado ao vetor $|0\rangle$ na combinação linear. Em outras palavras, obtemos o bit zero 50% das vezes porque o quadrado de $1/\sqrt{2}$ é 1/2.

O paralelismo quântico é radicalmente diferente do paralelismo clássico. No clássico, duas opera-

¹Informalmente, diz-se que uma expressão aritmética f(n) está em sua fórmula fechada (Graham, Knuth e Patashnik 1989, capítulo 1, seção 1.2, página 7) se a expressamos com no máximo um número fixo de operações convencionadas e independentes de n. Em particular, reticências não são permitidas se elas expressam um número variável de operações na expressão de f(n).

²Os coeficientes da combinação linear representando um estado quântico são chamados de "amplitudes."

ções são feitas ao mesmo tempo por dois circuitos diferentes. No quântico, "duas operações" diferentes são feitas ao mesmo tempo pelo mesmo circuito. Na verdade é uma operação só, mas temos a sensação de que mais de uma operação ocorre. Vejamos um exemplo. No computador quântico é possível implementar a função $\{0,1\} \rightarrow \{0,1\}: f(x)$, ou seja, uma função de um q-bit para um q-bit, usando dois q-bits $|x\rangle|y\rangle$. Usando U_f , é possível transformar esse estado em $|x\rangle|f(x) \oplus y\rangle$, sendo \oplus adição módulo 2. Colocando $|x\rangle = |+\rangle \in |y\rangle = |0\rangle$, obtemos

$$U_{f}|+\rangle|0\rangle = U_{f}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$$
$$= \frac{1}{\sqrt{2}}(U_{f}|0\rangle|0\rangle + U_{f}|1\rangle|0\rangle)$$
$$= \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle)$$

evidenciando duas computações de f em apenas uma operação quântica³. Entretanto, assim que fizermos a leitura do segundo q-bit de saída de U_f , a superposição colapsa a um dos vetores da base do espaço vetorial. Nesse exemplo, a superposição colapsa para $|0\rangle$ em 50% dos casos e colapsa para $|1\rangle$ nos outros 50% dos casos.

Tirar proveito desse paralelismo requer criatividade. O objetivo de um algoritmo quântico típico é manipular os coeficientes da combinação linear representando o estado quântico que codifica o problema a ser resolvido de forma que, ao fim da execução do algoritmo, a resposta desejada possua razoável probabilidade de ser obtida quando a leitura do estado quântico for feita. Basta que a probabilidade seja razoável: mesmo que o algoritmo fracasse algumas vezes, a probabilidade de sucesso cresce exponencialmente à medida que tentamos novas execuções. Por exemplo, com 5% de probabilidade de sucesso, esperamos obter uma resposta desejada a cada vinte execuções em média, o que não é um alto preço a ser pago quando a estratégia é várias ordens de grandeza mais eficiente que a alternativa.

Na versão quântica do algoritmo Rô de Pollard, desejamos calcular o tamanho do ciclo de uma função cíclica. O algoritmo quântico de cálculo do período prepara um estado quântico representado por uma combinação linear de todos os números que fazem parte do ciclo da função. Essa preparação ocorre em uma única operação se e somente se pudermos representar um elemento do ciclo através de uma fórmula fechada. Por exemplo, se desejarmos o décimo elemento da sequência gerada pelo polinômio iterado $p(x) = x^2 + 1 \mod N$ a partir de $x_0 = 2$, mesmo um computador quântico será obrigado a calcular o nono, oitavo, sétimo, ... Até o momento não se conhece uma fórmula que possa calcular o décimo elemento de p(x) sem calcular o nono. Em contrapartida, o polinômio iterado $s(x) = 2x \mod N$ com valor inicial $x_0 = 1$ tem fórmula fechada $2^i \mod N$. Assim, se desejamos um estado quântico com todos os números do ciclo de s(x), preparamos uma combinação linear com todos os números naturais do domínio da função e aplicamos um operador U_s que represente a função s(x), provendo-nos com um estado quântico que possui informação sobre todo o ciclo da função s(x).

³Observe que as equações aqui representam um único passo de um computador quântico. Reescrever uma equação significa tomar uma segunda perspectiva sobre um mesmo fato. A computação de U_f é uma única operação para um computador quântico, mesmo que a expliquemos em múltiplos passos matemáticos.

3.3 Uma expressão da estratégia num modelo quântico

Para uma versão quântica, precisaremos de uma função cíclica para a qual tenhamos uma fórmula fechada. É fácil encontrar uma fórmula fechada para a família de polinômios lineares

$$f(x) = ax \bmod N,\tag{3.1}$$

sendo $1 < a \mod N$ um número natural fixo coprimo com N, provendo-nos um caminho a seguir. Veremos na Seção 4.3 que são exatamente esses polinômios lineares que reduzem a versão quântica do algoritmo Rô de Pollard ao algoritmo de Shor.

Algoritmo 2. Uma versão quântica do algoritmo Rô de Pollaro	d usando fórmula fechada g .
${f algoritmo}$ QUÂNTICO-R $\hat{ m O}(g,N)$	
$r_g \leftarrow \text{QUÂNTICO-CALCULA-PERÍODO}(g)$	
repita para cada d em divisores-primos-pequenos (r_g)	\triangleright Um procedimento eficiente.
$p \leftarrow \mathrm{mdc}(g(N + r_g/d) - g(N), N)$	
${f se} \ 1 ão$	
retorne p	⊳ Colisão não-trivial encontrada.
fim se	
fim repita	
retorne nenhum	
fim algoritmo	

Seja $x_0 = 1 = g(0)$ o primeiro elemento da sequência gerada por qualquer membro da família definida pela Equação 3.1. Se escolhermos a = 3, obtemos a sequência 1, 3, 9, 27, ... mod N. Esse polinômio iterado tem fórmula fechada $g(i) \equiv a^i \mod N$. Como a gera um subgrupo cíclico de \mathbb{Z}_N^* , então sabemos que o tamanho do ciclo de g é a ordem de a no grupo \mathbb{Z}_N^* , fazendo com que o ciclo de g tenha um grafo circular, já que escolhemos $x_0 = 1 = g(0)$ como o primeiro elemento da sequência.

Comentário. O Teorema 3.1.5 nos exige encontrar um elemento dentro do ciclo, o que o Algoritmo 2 chama de g(N), mas a família de polinômios lineares definida pela Equação 3.1 nos permite tomar g(0) como um elemento do ciclo já que qualquer g(i) é elemento do ciclo.

O objetivo da versão quântica do algoritmo Rô de Pollard é procurar por uma colisão não-trivial nesse ciclo. Como seu primeiro passo, o Algoritmo 2 computa r_g , que é o período do ciclo de g. Em seguida, ele procura por um pequeno divisor primo d de r_g tal que d possa ser encontrado em até $\mathcal{O}(\lg^c N)$ operações para algum inteiro c. Se r_g tiver um divisor d pequeno o suficiente, então o Algoritmo 2 computa

$$\operatorname{mdc}(g(0+r_g/d)-g(0),N) = \operatorname{mdc}(3^{r_g/d}-1,N).$$

Ilustramos o método usando a família definida pela Equação 3.1 no Exemplo 3.3.1.

Exemplo 3.3.1. Seja p = 19 e q = 11 de forma que N = pq = 209. Escolhemos $x_0 = 1$ e a = 3 para determinarmos a função g da família definida pela Equação 3.1. O primeiro passo do Algoritmo 2 é computar o período de g, que é $r_g = 90$. O próximo passo é procurar por um divisor de r_g , cujo

primeiro é 2. Tendo um divisor do período de g, o algoritmo computa

$$mdc(g(0+90/2) - g(0), N) = mdc(3^{90/2} - 1, 209),$$

produzindo o desejado fator 11. A razão por que atingimos o objetivo é que $\operatorname{ord}(3,11) = 5 \neq 18 = \operatorname{ord}(3,19)$ e $r_g/2 = 45$ é múltiplo de $\operatorname{ord}(3,11)$ mas não de $\operatorname{ord}(3,19)$, então o Teorema 3.1.5 garante que $\{g(45), g(0)\} = \{56, 1\}$ é uma colisão não-trivial.

Comentário. O procedimento "divisores-primos-pequenos" em uso na versão quântica do algoritmo Rô de Pollard (Algoritmo 2) retorna imediatamente após o primeiro divisor primo de r_g ser encontrado. Em invocações subsequentes, o procedimento continua de onde parou, provendo ao procedimento QUÂNTICO-RÔ o próximo divisor primo de r_g e assim sucessivamente até exaurir as possibilidades do procedimento. Linguagens de programação chamam esse tipo de procedimento de "gerador". Suporte a geradores não é meio de abstração necessário para se implementar essa versão quântica do algoritmo Rô de Pollard (Algoritmo 2). Usamos os geradores com o objetivo de uma exposição mais clara do procedimento. Geradores foram concebidos por Barbara Liskov, Alan Snyder, Russell Atkinson and Craig Schaffert e implementados em CLU (Liskov, Snyder, Atkinson e Schaffert 1977).

3.4 Outras famílias de funções cíclicas para a versão quântica

A Equação 3.1 não é a única família que pode ser usada com o Algoritmo 2. As funções da família

$$g(i) \equiv a^{b^i \mod m} \mod N \tag{3.2}$$

contêm um ciclo a partir de um número natural não-negativo *i* sempre que *m* é múltiplo de *r*, assumindo-se a, b, N naturais fixos e $r = \operatorname{ord}(a, N)$, o que provamos na Seção 3.5. Sendo cíclicas, são úteis à versão quântica do algoritmo Rô de Pollard.

Se desejamos por um algoritmo polinomial, precisamos encontrar um meio de manter o expoente b^i pequeno, o que é possível se o reduzirmos módulo m. Entretanto, como m é múltiplo de ord(a, N), teremos uma dependência do algoritmo quântico de cálculo da ordem. Se conseguirmos encontrar fórmulas fechadas para funções cíclicas que não dependam da ordem de um número, a dependência desaparece. A sugestão de que o algoritmo Rô de Pollard generaliza o algoritmo de Shor é sustentada pelo Teorema 3.1.5, que não depende de qualquer fórmula fechada.

Diferente da família de funções definidas pela Equação 3.1, que gera um subgrupo cíclico de \mathbb{Z}_N^* , a família de funções definida pela Equação 3.2 gera sequências que fecham um ciclo em si mesmas mas não necessariamente no primeiro elemento. Para aplicarmos o Teorema 3.1.5, precisamos encontrar um elemento que pertença ao ciclo. Podemos tomar g(N) como valor inicial: sabemos que g(N) tem que pertencer ao ciclo porque a sequência não pode ter mais que N elementos distintos.

Descrevemos os passos do método novamente usando agora a Equação 3.2. O procedimento QUÂNTICO-RÔ no Algoritmo 2 consome N, o composto que se deseja fatorar. Em seguida, computa o

período r_g de g com valor inicial g(N), o que torna g uma função periódica. O método então procura por um divisor primo d de r_g experimentando os menores primos até o n-ésimo menor primo, sendo n a quantidade de bits em N, o que nos certifica de que o procedimento "divisores-primos-pequenos" termine em tempo polinomial e, consequentemente, o QUÂNTICO-RÔ seja também limitado acima por um polinômio. Se o procedimento encontrar d que satisfaça o Teorema 3.1.5, então

$$1 < \mathrm{mdc}(g(N + r_q/d) - g(N), N) < N.$$

Ilustramos os passos do uso da nova família de funções no Exemplo 3.4.1.

Exemplo 3.4.1. Seja p = 31, q = 43 de forma N = pq = 1333. Seja a = 3, b = 2 de forma que g seja a função $g(i) \equiv 3^{\gamma} \mod 1333$, sendo $\gamma = 2^{i \mod 210}$, e escolha g(1333) = 9 como valor inicial. O período de g é $r_g = 12$. Sendo 2 um divisor do período de g, computamos

$$mdc(g(N + r_q/2) - g(N), N) = mdc(g(1333 + 6) - g(1333), 1333) = 43,$$

como desejado. A razão por que atingimos o objetivo neste exemplo é que $\lambda_p = 4 \neq 6 = \lambda_q e r_g/2 = 6$ é um múltiplo de λ_q mas não de λ_p , então o Teorema 3.1.5 garante que $\{g(1333+6), g(1333)\} = \{826, 9\}$ é uma colisão não-trivial.

A versão clássica do método de Pollard tipicamente usa um polinômio quadrático de coeficientes inteiros como $x^2 + 1 \mod N$. Usando um polinômio quadrático nessa versão quântica do método nos exige encontrar uma fórmula fechada. A família de funções iteradas

$$f(x) = ax^2 + bx + \frac{b^2 - 2b}{4a}$$
(3.3)

tem fórmula fechada

$$f^{n}(x) = (2\beta^{2^{n}} - b)/(2a)$$

sendo $\beta = (2ax + b)/2$, o que provamos na Seção 3.6. Assim como no caso da família definida pela Equação 3.2, precisamos evitar que o expoente β^{2^n} cresça, o que é possível se reduzirmos 2^n módulo ord (β, N) ou módulo $\varphi(N)$, sendo φ a função φ de Euler, que denota o número de inteiros no intervalo [1, N] que são coprimos com N. Obtemos

$$f^{n}(x) = (2\beta^{\gamma} - b)(2a)^{-1} \mod N,$$

sendo $\gamma = 2^n \mod r \in r = \operatorname{ord}(\beta, N)$. Um último requerimento é escolher $a \in b$ tais que $f^n(x)$ represente um polinômio de coeficientes inteiros. Se escolhermos a = 1, b = 2 e um valor inicial x_0 para a sequência, obtemos o polinômio iterado

$$n_{i+1} \equiv f(x_i) \equiv x_i^2 + 2x_i \mod N$$

cuja fórmula fechada é

$$n_{i+1} \equiv g(i) \equiv ((x_0+1)^{\gamma} - 1) \mod N,$$

sendo $\gamma = 2^i \mod r \in r = \operatorname{ord}(x_0 + 1, N)$. Novamente escolhemos g(N) como valor inicial porque sabemos que g(N) é um elemento no ciclo da sequência gerada pelo polinômio f(x), o que nos dá uma função periódica. O Exemplo 3.4.2 ilustra o método com a família de polinômios quadráticos definidos pela Equação 3.3.

Exemplo 3.4.2. Seja p = 7907, q = 7919 de forma que N = pq = 62615533. Seja a = 1, b = 2 de forma que f seja o polinômio de coeficientes inteiros

$$x_{i+1} \equiv f(x_i) \equiv x_i^2 + 2x_i \mod N$$

e escolha $x_0 = 3$ como valor inicial. Uma fórmula fechada para f é

$$g(i) \equiv ((x_0+1)^{\gamma} - 1) \mod N$$

sendo $\gamma = 2^i \mod r \in r = \operatorname{ord}(x_0 + 1, N)$. O Algoritmo 2 precisa do número $r = \operatorname{ord}(4, N) = 15649927$, um inteiro ímpar, e então computa $x_N = g(N) = 10689696$, que é o *n*-ésimo elemento da sequência gerada por g e é um elemento no ciclo de g. Restringindo g por trocar seu valor inicial para x_N , obtemos uma função periódica cujo período $r_g = 608652$ é calculado pelo algoritmo quântico de cálculo do período. O Algoritmo 2 então procura por uma colisão não-trivial por tentar pares $\{g(N+r_g/d), g(N)\}$ assumindo que ele consiga encontrar um divisor primo d de r_g . Como 608652 é par, d = 2 é o primeiro divisor primo de r_g que encontramos. O algoritmo encontra uma colisão não-trivial usando d = 2porque a fatoração prima de

$$r_g/2 = 2 \times 3^2 \times 11 \times 29 \times 53,$$
$$\lambda_p = 2 \times 3 \times 11 \times 29,$$
$$\lambda_q = 2^2 \times 3^2 \times 53$$

mostra que $r_g/2$ é múltiplo de λ_p mas não de λ_q , implicando que $\{g(r_g/2), g(N)\}$ é uma colisão nãotrivial, o que confirmamos por computar

$$mdc(16896691 - 10689696, 62615533) = 7907,$$

como desejado.

Descrevemos no Apêndice A a implementação de um circuito quântico em termos de portas quânticas elementares em que fatoramos o número $143 = 11 \times 13$ usando a fórmula fechada da família definida pela Equação 3.3 usando valor inicial $x_0 = 2$.

Comentário. Uma outra família de funções que poderíamos usar é

$$g(i) \equiv (2\beta^{2^{i}} + 2\beta^{-2^{i}} - b)(2a)^{-1} \mod N,$$

sendo $\beta = (2ax_0 + b) \pm ((2ax_0 + b)^2 - 16)^{1/2} 4^{-1}$. Essa família corresponde à família de funções

$$f(x) = ax^{2} + bx + (b^{2} - 2b - 8)/(4a)$$

O procedimento QUÂNTICO-RÔ do Algoritmo 2 usa apenas g e não f. A exigência importante é que g corresponda a uma função cíclica porque assim temos certeza que classes de pares representando colisões estarão igualmente espaçados no ciclo. Não é qualquer fórmula fechada que nos atende.

3.5 Sobre a família de funções super-exponenciais

Apresentamos na Seção 3.3 que a família de funções definida pela Equação 3.2 pode ser usada com a versão quântica do algoritmo Rô de Pollard. Preocupamo-nos agora em mostrar que qualquer membro da família possui um ciclo.

Teorema 3.5.1. Seja a, b, N números naturais fixos e r = ord(a, N). Se $m \ \acute{e} \ um \ m\'ultiplo \ de \ r, \ então$ as funções da família

$$g(i) \equiv a^{b^i \mod m} \mod N$$

têm um ciclo começando a partir de algum natural i.

Prova. Aplicando indução em i, basta mostrar que

$$g(i+k) \equiv g(i)$$
 implica $g(i+k+1) \equiv g(i+1)$

para algum inteiro positivo k, para todo $i \ge 0$.

Dado que temos um módulo no expoente, podemos escrever g como

$$g(i) \equiv a^{b^i + qm} \equiv a^{b^i} a^{qm} \mod N_i$$

Como m é múltiplo de r, então $a^{qm} \equiv 1 \equiv (a^m)^q$ e, logo, $g(i) \equiv a^{b^i} \mod N$, o que nos permite escrever

$$g(i+1) \equiv a^{b^{i+1}} \mod N \equiv a^{b^i b} \mod N \equiv \left(a^{b^i}\right)^b \mod N \equiv g(i)^b.$$

Similarmente, podemos escrever $g(i + k + 1) \equiv g(i + k)^b$ para algum inteiro positivo k. Agora, uma vez que $g(i) \equiv g(i + k)$ por hipótese, deduzimos

$$g(i+k+1) \equiv g(i)^b \equiv g(i+1),$$

para todo $i \ge 0$, como desejado.

Enfatizamos a imposição de que *m* seja múltiplo de $r = \operatorname{ord}(a, N)$ porque, de outra forma, não teríamos um ciclo. Considere o exemplo em que $N = 541 \times 547$, a = 3, b = 2 e m = 3571. Por ser primo, *m* não é múltiplo de $r = \operatorname{ord}(a, N) = 1890$. Ocorre que $g(i) \equiv 3^{2^i \mod 3571} \mod N \in g(0) \equiv 3 \equiv g(183)$ mas $g(1) \equiv 9 \not\equiv 233025 \equiv g(184)$. Logo, *g* não é uma função cíclica.

3.6 Sobre a família de polinômios quadráticos

Concentremo-nos agora na equivalência entre a família de polinômios quadráticos iterados definida pela Equação 3.3 na Seção 3.3 e sua a fórmula fechada usada pelo Algoritmo 2.

Teorema 3.6.1. A fórmula fechada para a família

$$f(x) = ax^2 + bx + \frac{b^2 - 2b}{4a}$$

de polinômios quadráticos iterados é

$$f^{i}(x_{0}) = \frac{2\beta^{2^{i}} - b}{2a},$$

sendo $\beta = (2ax_0 + b)/2$ e f^i representando a *i*-ésima iteração de f tendo $x = x_0$ como valor inicial.

Prova. Aplicamos indução em i. O primeiro elemento da sequência gerada por $f \in x_0$ por definição, o que verificamos por computar

$$f^{0}(x_{0}) = x_{0} = \frac{2ax_{0} + b - b}{2a} = 2\left(\frac{[2ax_{0} + b]/2}{2a}\right) - \frac{b}{2a} = \frac{2\beta - b}{2a} = \frac{2\beta^{2^{0}} - b}{2a}$$

Agora, suponha

$$f^k(x_0) = \frac{2\beta^{2^k} - b}{2a},$$

para algum $k \ge 0$ sendo $\beta = (2ax_0 + b)/2$. Precisamos mostrar que

$$f^{k+1}(x_0) = \frac{2\beta^{2^{k+1}} - b}{2a}.$$

 Como

$$f^{k+1}(x_0) = f(f^k(x_0)),$$

podemos deduzir

$$\begin{split} f^{k+1}(x_0) &= f\left(\frac{2\beta^{2^k} - b}{2a}\right) \\ &= a\left(\frac{2\beta^{2^k} - b}{2a}\right)^2 + b\left(\frac{2\beta^{2^k} - b}{2a}\right) + \frac{b^2 - 2b}{4a} \\ &= \frac{4\beta^{2^{k+1}} - 4\beta^{2^k}b + b^2}{4a} + \frac{2\beta^{2^k}b - b^2}{2a} + \frac{b^2/2 - b}{2a} \\ &= \frac{2\beta^{2^{k+1}} - 2\beta^{2^k}b + b^2/2 + 2\beta^{2^k}b - b^2 + b^2/2 - b}{2a} \\ &= \frac{2\beta^{2^{k+1}} - b}{2a}, \end{split}$$

como desejado.

г		т
L		
L		
L		
•		

3.7 A complexidade da versão quântica

Assintoticamente, a complexidade da versão quântica do algoritmo Rô de Pollard (Algoritmo 2) é a mesma que o do algoritmo de Shor (Algoritmo 3), que é $\mathcal{O}(\lg^3 N)$. Usando uma amostra de todos os pares de primos extraídos do conjunto dos mil menores primos, tentamos fatorar cada semiprimo usando três algoritmos: o algoritmo de Shor original de 1994, o algoritmo de Shor estendido para ordens ímpares e a versão quântica de algoritmo Rô de Pollard usando polinômios quadráticos escolhidos aleatoriamente. Computamos a taxa de sucesso de cada um deles, assumindo que ordens de elementos de \mathbb{Z}_N^* fossem sempre corretamente obtidas, o que não é a verdadeira taxa de sucesso em decorrência da probabilidade não-nula de fracasso do algoritmo quântico de cálculo da ordem. A taxa de sucesso relativa ao algoritmo de Shor, considerando nossas assunções, não parece ser maior que 75%, e ambas a versão estendida do algoritmo de Shor e a versão quântica do algoritmo Rô de Pollard com polinômios escolhidos aleatoriamente parecem empatar em aproximadamente 98%. Entretanto, parece haver uma pequena vantagem para a versão quântica do algoritmo Rô de Pollard. Assumindo que o algoritmo de Shor compute a exponenciação modular usando o algoritmo de Schönhage–Strassen (Schönhage e Strassen 1971), sua complexidade pertence a $\mathcal{O}(n^2 \lg n \lg \lg n)$, sendo n a quantidade de bits do composto que se deseja fatorar. A versão quântica do algoritmo Rô de Pollard usando a família definida pela Equação 3.3 também tem que computar exponenciações modulares, mas contra o módulo $r = \operatorname{ord}(\beta, N)$, que tende a ser menor que N. Observando que a quantidade de bits em r é floor($\lg r$) + $1 \approx (1/4) \lg N$ se assumirmos que o tamanho de r tende a ser da ordem de $N^{1/4}$, então essa versão quântica do algoritmo Rô de Pollard termina em até $\mathcal{O}((1/4)n^2 \lg n \lg \lg n)$ operações, o que é um fator constante menor que a complexidade do algoritmo de Shor, que termina em até $\mathcal{O}(n^2 \lg n \lg \lg n)$ operações. Parece a nós, portanto, que se o algoritmo de Shor fracassar em fatorar o composto desejado e conseguir corretamente calcular $\operatorname{ord}(\beta, N)$, pode-se vislumbrar uma pequena vantagem em empregar uma versão quântica do Rô de Pollard em vez de repetir o método de Shor uma segunda vez.

Capítulo 4

Uma generalização do algoritmo de Shor

Peter W. Shor trouxe a computação quântica para o centro das atenções em 1994 com o ganho exponencial em sua estratégia (Shor 1994; Shor 1997; Shor 1999) de encontrar a ordem de um elemento $x \mod N \in \mathbb{Z}_N^*$.

Existe uma redução via uma transformação em tempo polinomial do problema da fatoração ao problema de se encontrar a ordem de um elemento (Miller 1976). Podendo-se computar $\operatorname{ord}(x, N)$ em tempo polinomial num modelo quântico de computação, o resto do trabalho de fatorar um composto N pode ser feito eficientemente num modelo clássico de computação, provendo-nos uma solução eficiente para o problema da fatoração.

4.1 A estratégia de Shor

A estratégia do algoritmo de Shor parece vir (Shor 1994; Shor 1997; Shor 1999) da identidade

$$(x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 = 0 \mod N,$$
(4.1)

sendo r a ordem de $x \mod N$. Quando r é par, r/2 é um número inteiro, implicando que a identidade expressa pela Equação 4.1 nos mostra como encontrar os fatores de N, a não ser quando $x^{r/2} \equiv$ $-1 \mod N$. Quando $x^{r/2} \equiv -1 \mod N$, o fator $x^{r/2} + 1$ é zero módulo N, o que satisfaz a Equação 4.1, mas não revela um fator não-trivial. Então o algoritmo de Shor enfrenta esse obstáculo de natureza número-teórica, tornando-o uma estratégia probabilística por essa razão. Uma outra razão é o cálculo da ordem no modelo quântico: a estratégia de Shor nem sempre obtém a ordem de x, adicionando uma segunda possibilidade de fracasso. O Algoritmo 3 expressa a estratégia e o Exemplo 4.1.1 ilustra o método.

Exemplo 4.1.1. Seja p = 19 e q = 11 de forma que N = pq = 209. Escolha x = 3, que é coprimo com N. O primeiro passo do Algoritmo 3 é computar $r = \operatorname{ord}(x, 209) = 90$. Sendo r par, o procedimento experimenta e obtém o fator $11 = \operatorname{mdc}(3^{90/2} - 1, 209)$. Não teríamos obtido um fator não-trivial se tivéssemos escolhido x = 2 porque $\operatorname{ord}(2, N) = 90$ também, mas $2^{90/2} \equiv 208 \equiv -1 \mod 209$ fazendo com que $x^{r/2} + 1 \equiv 0 \mod 209$. Já com x = 3, essa condição não ocorre: $3^{90/2} - 1 \equiv 56 \not\equiv -1 \mod 209$.

prime, e « coprime com r.t.	
algoritmo $SHOR(x, N)$	
$r \leftarrow \text{QUÂNTICO-ORDEM}(x, N)$	
se r é par então	
$p \leftarrow \mathrm{mdc}(x^{r/2} - 1, N)$	
se $1 então$	
retorne p	
fim se	
$\mathbf{fim} \ \mathbf{se}$	
retorne nenhum	
fim algoritmo	

Algoritmo 3. O algoritmo de Shor usando $x \mod N$, sendo N um composto que não é uma potência prima, e x coprimo com N.

Se tivéssemos escolhido x = 4, encontraríamos $r = \operatorname{ord}(4, 209) = 45$, que é ímpar e, logo, r/2 não é inteiro, fracassando também a execução do Algoritmo 3.

No Algoritmo 3, assumimos que N não é potência prima (Shor 1994, seção 6, página 130). Verificar que um número não é uma potência prima pode ser feito em tempo polinomial (Menezes, Oorschot e Vanstone 1996, capítulo 3, nota 3.6, página 89) no modelo clássico de computação. Sempre que o algoritmo de Shor é mencionado, assumimos que essa verificação é aplicada ao composto que se deseja fatorar.

Quando r é impar, uma extensão (Johnston 2017, seções 2–3, páginas 2–3) do algoritmo de Shor existe em que mais tentativas de fatoração são feitas contra N. Investigamos essa extensão agora na Seção 4.2.

4.2 Uma extensão do algoritmo de Shor

Estratégias particulares para tirar vantagem do uso de ordens ímpares no algoritmo de Shor foram exploradas (Cao 2005; Lawson 2015; Xu, Qiu, Zou e Gruska 2018) e uma extensão geral do método para ordens ímpares foi eventualmente publicada (Johnston 2017). Mais recentemente, apresentamos uma perspectiva diferente do mesmo resultado (Chicayban Bastos e Kowada 2021). Mostraremos agora como a extensão do algoritmo de Shor decorre naturalmente de duas abordagens diferentes, uma delas por via do Teorema 3.1.5.

A razão por que o algoritmo de Shor precisa que a ordem de x seja par é devido à Equação 4.1. Entretanto, podemos generalizar a Equação 4.1, o que nos leva imediatamente a uma extensão de ordens ímpares como a expressa pelo Algoritmo 4. Por exemplo, se 3 divide r, então $(x^{r/3} - 1)(1 + x^{r/3} + x^{2r/3}) = x^r - 1 \equiv 0 \mod N$. De uma forma geral,

$$(x^{r/d} - 1)\left(\sum_{i=0}^{d-1} x^{ir/d}\right) = x^r - 1 \equiv 0 \mod N,$$
(4.2)

sempre que d divide r. O Exemplo 4.2.1 ilustra a vantagem de se usar a versão estendida expressa pelo Algoritmo 4.

algoritmo SHOR-ESTENDIDO (x, N)
$r \leftarrow \operatorname{Qu}\hat{\operatorname{A}}\operatorname{NTICO-ORDEM}(x,N)$
repita para cada d em divisores-primos-pequenos (r)
$p \leftarrow \mathrm{mdc}(x^{r/d} - 1, N)$
$\mathbf{se} \ 1$
retorne p
fim se
fim repita
retorne nenhum
fim algoritmo

Algoritmo 4. Uma extensão do algoritmo de Shor em que um número fixo de pequenos divisores primos da ordem de $x \in \mathbb{Z}_N^*$ é considerado. O procedimento assume que N não é potência prima.

Exemplo 4.2.1. Seja p = 7907, q = 7919 de forma que N = pq = 62615533. Se escolhermos x = 3, obteremos $r_g = \operatorname{ord}(3, N) = 15649927$. Como N tem 26 bits, nosso procedimento verifica se algum dos 26 menores primos divide r_g . Os onze menores primos não dividem r_g , mas o décimo-segundo primo é 37, que divide r_g . Então o Algoritmo 4 encontra um fator não-trivial de N computando

$$mdc(x^{r_g/37} - 1, N) = mdc(48604330 - 1, 62615533) = 7907,$$

como desejado.

Similar à estratégia de Shor de 1994, a fórmula da Equação 4.2 também produz um procedimento que precisa impor que $x^{r/d} \not\equiv -1 \mod N$ porque, de outra forma, a Equação 4.2 também é trivialmente satisfeita sem produzir um fator não-trivial. A Equação 4.2 provê uma simples generalização do algoritmo de Shor, expressa pelo Algoritmo 4, mas não parece prover um entendimento satisfatório sobre a imposição $x^{r/d} \not\equiv -1 \mod N$ quando comparada ao entendimento que obtemos a partir do Teorema 3.1.5, o que ilustramos no Exemplo 4.2.2.

Exemplo 4.2.2. O Exemplo 4.2.1 fatora N = 62615533 usando o Algoritmo 4 a partir da escolha x = 3, obtendo $r_g = \operatorname{ord}(3,N) = 15649927$, um inteiro ímpar. O Algoritmo 4 verifica se os onze menores primos dividem r_g sem sucesso, mas encontra o décimo-segundo primo, 37, como divisor de r_g e assim obtém o fator não-trivial 7907. O objetivo é atingido, mas a fórmula da Equação 4.2 parece menos satisfatória quando a comparamos com a perspectiva provida pelo Teorema 3.1.5, que nos oferece uma visão intuitiva do porquê o divisor 37 produz a fatoração. Observando que as fatorações primas de

$$r_g/37 = 59 \times 67 \times 107,$$

 $r_p = 59 \times 67,$
 $r_a = 37 \times 107.$

sendo $r_p = \operatorname{ord}(3,7907)$ e $r_q = \operatorname{ord}(3,7919)$, compreendemos que $r_g/37$ é múltiplo de r_p mas não é múltiplo de r_q , garantindo que $1 < \operatorname{mdc}(x^{r/37} - 1, N) < N$. A intuição provida pela Seção 2.4 nos dá a imagem de que os elementos $x^{r/37}$ e 1 estão posicionados no ciclo (do grafo que representa o subgrupo de \mathbb{Z}_N^* gerado por 3) tal que a distância entre eles é expressa por um múltiplo de r_p que não é múltiplo

de r_q . Por "distância", queremos dizer o menor inteiro k tal que $3^{r/37}3^k = 3^{(r/37)+k} = 1$, ou seja, por "distância" queremos dizer quantas arestas do grafo precisamos percorrer para partir de $3^{r/37}$ e chegar a 1. (O número 1 está sempre presente no ciclo já que o grafo em questão é uma representação do subgrupo cíclico gerado por $3 \in \mathbb{Z}_N^*$.) Sabemos que essa distância é um número múltiplo de r_p e não de r_q porque o fator não-trivial obtido foi 7907; fôsse 7919, diríamos o converso.

Quando estudamos a família de polinômios lineares definida pela Equação 3.1 na Seção 3.3, anunciamos que eram exatamente esse polinômios que reduziam a versão quântica do algoritmo Rô de Pollard ao algoritmo de Shor. A seção a seguir procura emparelhar os passos de ambos métodos com o objetivo de esclarecer a proximidade entre o algoritmo de Shor e a versão quântica do algoritmo Rô de Pollard quando usa um polinômio da família $ax \mod N$ e valor inicial $x_0 = 1$.

4.3 O algoritmo de Shor como um caso particular do Rô de Pollard

A versão quântica do algoritmo Rô de Pollard que apresentamos na Seção 3.3 requer uma função cíclica que possua uma fórmula fechada. A primeira alternativa que apresentamos foi os polinômios lineares da família $ax \mod N$ definida pela Equação 3.1. O uso dessa família faz com que a versão quântica do algoritmo Rô de Pollard (Algoritmo 2) execute essencialmente os mesmos passos do algoritmo de Shor (Algoritmo 3).

Revisitemos o Exemplo 3.3.1, que serviu para ilustrar a versão quântica do algoritmo Rô de Pollard, mas agora com a intenção de explicitar que efetivamente computamos os passos do algoritmo de Shor. O exemplo fatora $N = 209 = 19 \times 11$ por escolher $x_0 = 1$ e a = 3 para determinar $g(i) = 3^i$, fórmula fechada do polinômio linear $3x \mod N$ definido pela Equação 3.1. Calcular o período de g(i) é o mesmo que calcular ord(3, N). Enquanto a versão quântica do algoritmo Rô de Pollard (Algoritmo 2) computa mdc(g(0+90/2) - g(0), 209), o algoritmo de Shor (Algoritmo 3) computa mdc $(3^{90/2} - 1, 209)$, que são computações idênticas já que $g(i) = 3^i$ e $x_0 = 1$ para qualquer g.

A subtração $3^{90/2}-1$ feita pelo algoritmo de Shor (Algoritmo 3) representa a colisão

$$\{g(45),g(0)\} = \{g(90/2),g(0)\}.$$

O algoritmo de Shor procura por colisões não-triviais em que um dos componentes do par é sempre o número 1; a versão quântica do algoritmo Rô de Pollard tem mais liberdade. Não parece absurdo dizer que a estratégia de Pollard relativa à família de polinômios lineares $ax \mod N$ definida pela Equação 3.1 é essencialmente o algoritmo de Shor quando traduzida para um modelo quântico de computação.

4.4 A equivalência ao Teorema de Anna M. Johnston

Em 2017, Anna Johnston publicou a observação de que uma extensão do algoritmo de Shor para ordens ímpares pode ser feita (Johnston 2017). Se d é divisor primo de r, então é possível encontrar um elemento $x^{r/d}$ que possui ordem d (Johnston 2017, seção 5, páginas 4–6), o que implica $x^{r/d} \equiv 1 \mod A$ e $x^{r/d} \not\equiv 1 \mod B$, sendo N = AB o composto que se deseja fatorar, assumindo-se A, B primos entre si. Daí $x^{r/d} - 1 \mod N$ é isomórfico a (0 mod $A, h - 1 \mod B$), sendo $h \not\equiv 1 \mod B$, implicando $\operatorname{mdc}(x^{r/d} - 1, N) = A$. Organizamos esse resultado através do enunciado do Teorema 4.4.1.

Teorema 4.4.1 (Anna M. Johnston). Seja N = AB, sendo A, B fatores não-triviais de N e primos entre si. Seja $r = \operatorname{ord}(x, N)$ para algum $x \mod N$. Assuma $x^{r/d} \equiv 1 \mod A$ e $x^{r/d} \not\equiv 1 \mod B$ para algum divisor primo d de r. Então $1 < \operatorname{mdc}(x^{r/d} - 1, N) < N$.

Teorema. O Teorema 3.1.5 implica o Teorema 4.4.1.

Prova. Seja N = AB, sendo A, B primos entre si. Seja 1 < x tal que mdc(x, N) = 1. Seja $r_A = ord(x, A)$ e $r_B = ord(x, B)$. Assuma $r_A \neq r_B$. Pelo Teorema 3.1.5, existe m < r tal que m é múltiplo de r_A e não é múltiplo de r_B . Além disso, $1 < mdc(x^m - 1, N) < N$. Precisamos mostrar que $x^m \equiv 1 \mod A$ e $x^m \not\equiv 1 \mod A$. Já temos o fato de que $x^m - 1 \equiv 0 \mod N$. Como m é múltiplo de r_A , então $A = mdc(x^m - 1, N)$, o que implica $x^m - 1 \equiv 0 \mod A$ e, logo, $x^m \equiv 1 \mod A$. Mas, se $A = mdc(x^m - 1, N)$, então necessariamente $x^m - 1 \not\equiv 0 \mod B$ porque, de outra forma, $x^m - 1$ seria múltiplo de ambos A e B, violando a hipótese de que $mdc(x^m - 1, N) < N$. □

Teorema. O Teorema 4.4.1 implica o Teorema 3.1.5.

Prova. Dadas as premissas do Teorema 4.4.1, precisamos mostrar que existe número m < r tal que m é múltiplo de r_A e m não é múltiplo de r_B , sendo $r = \operatorname{ord}(x, N)$ para algum $x \in \mathbb{Z}_N^*$, $N = AB \operatorname{com} A, B$ primos entre si e $r_A = \operatorname{ord}(x, A), r_B = \operatorname{ord}(x, B)$. Mostraremos que r/d satisfaz o Teorema 3.1.5, sendo d um divisor primo de r, deduzindo que m = r/d é o número que procuramos. Observe que r/d < r já que a primalidade de d significa 1 < d. Por hipótese, $x^{r/d} \equiv 1 \mod A$ implica que r/d é múltiplo de $\operatorname{ord}(x, A) = r_A$. Similarmente, $x^{r/d} \not\equiv 1 \mod B$ implica que r/d não é múltiplo de $\operatorname{ord}(x, B) = r_B$. Seja m = r/d, satisfazendo o Teorema 3.1.5, como desejado.

Comentário. O artigo da Anna Johnston usa a letra r para representar o divisor primo da ordem e a letra s para representar a ordem de $x \in Z$. Grande parte da literatura usa r como a ordem de um elemento $x \in \mathbb{Z}_N^*$, então apresentamos as observações de Johnston com essas adaptações. O número $x^{r/d}$ é frequentemente escrito como b_d por Johnston, sendo que no artigo veríamos " b_r ", já que ela usa r como o divisor primo da ordem de x.

Comentário. Johnston comenta que quando a ordem é ímpar, embora um fator não-trivial possa ser $mdc(x^{r/d}-1, N)$, sendo d um divisor primo da ordem, o outro fator não-trivial não é revelado (Johnston 2017, seção 3, página 3) por $mdc(x^{r/d}+1, N)$. O artigo não oferece uma expressão que revele o outro fator não-trivial, mas a identidade expressa pela Equação 4.2 responde. Por exemplo, se d =

3, então $x^r - 1 = (x^{r/3} - 1)(1 + x^{r/3} + x^{2r/3})$. Logo, o outro fator não-trivial seria revelado por $\operatorname{mdc}(x^{r/3} + x^{2r/3} + 1, N)$. É certo que pode-se chegar a ele mais simplesmente por $N/\operatorname{mdc}(x^{r/d} - 1, N)$.

Comentário. Johnston também afirma (Johnston 2017, seção 3, página 2) que $\operatorname{ord}(x^{r/d}, N) = d$, sendo d um divisor primo de $\operatorname{ord}(x, N)$. Eis uma prova do fato. Seja $1 < x \mod N$, sendo x coprimo com N. Seja $r = \operatorname{ord}(x, N)$. Então $x^r \equiv 1 \mod N$. Se um primo d divide r, então $x^{r/d} \equiv y \mod N \not\equiv$ $1 \mod N$ para algum $y \mod N$. Precisamos mostrar que $\operatorname{ord}(y, N) = d$. Por definição, $\operatorname{ord}(y, N)$ é o menor natural s tal que $y^s \equiv (x^{r/d})^s \equiv x^{rs/d} \equiv 1 \mod N$. Como $\operatorname{ord}(x, N) = r$, então rs/d é múltiplo de r, ou seja, $r \leq rs/d$, implicando que s = d é o menor s que satisfaz a desigualdade $r \leq rs/d$. Logo, d é a ordem de $x^{r/d}$, como desejado.

4.5 O cálculo da ordem no modelo quântico

Eis como calcular a ordem de $x \in \mathbb{Z}_N^*$ usando o procedimento quântico concebido por Peter Shor em 1994. Seja $N = pq \mod p, q$ primos. Escolha $x \mod N$ aleatoriamente. Verifique que x é coprimo com N: se não for, a ordem de x não existe e um fator não-trivial de N seria revelado computando-se $\operatorname{mdc}(x, N)$. Estime uma quantidade suficiente q de q-bits. Peter Shor sugeriu a quantidade q tal que $N^2 \leq q < 2N^2$ sendo q uma potência de 2. Construa o estado

$$\frac{1}{\sqrt{q}}\sum_{u=0}^{q-1}|u\rangle|0\rangle$$

usando dois registradores quânticos, o primeiro de tamanho q q-bits e o segundo de tamanho lg N q-bits. No segundo registrador, armazenaremos a saída da função $x^u \mod N$; o primeiro registrador armazena a superposição da entrada de $x^u \mod N$. Aplicando o operador que implementa a função $x^u \mod N$, obtemos o estado

$$\frac{1}{\sqrt{q}}\sum_{u=0}^{q-1}|u\rangle|x^u \bmod N\rangle,$$

o que é feito em tempo $\mathcal{O}(\lg^3 N)$, a etapa mais lenta do algoritmo de Shor. Aplique a Transformada de Fourier Quântica no primeiro registrador, obtendo

$$\frac{1}{q} \sum_{u}^{q-1} \sum_{v}^{q-1} \exp(2\pi i u v/q) |v\rangle |x^u \bmod N\rangle,$$

o que é feito em tempo $\mathcal{O}(\lg^2 N)$. Leia o primeiro registrador, obtendo a informação clássica v, o que encerra a parte quântica do procedimento de Shor. Agora, com razoável probabilidade, é possível extrair r através do número v/q. Usando o algoritmo das frações continuadas, compute cada possível fração que se aproxime de v/q experimentando se o denominador de cada uma delas é o número r desejado, o que é feito computando $x^{r'} \mod N$, sendo r' o denominador da aproximação. O Exemplo 4.5.1 ilustra como fazer.

Exemplo 4.5.1. Seja N = 35 e x = 2. A ordem de x é r = 12. Aplicando as instruções de Shor, usamos q = 64. Leitura do último estado quântico produz os números v = 5 ou v = 11 com razoável

probabilidade. Tomemos primeiro v = 11, o que produz o número 11/64 = 0.171875. A fração continuada de 0,171875 é [0; 5, 1, 4, 2], ou seja,

$$\frac{1}{5 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}}},$$

o que gera os racionais candidatos 1/5, 1/6, 5/29 e 11/64. Nenhum desses denominadores é a ordem de x = 2, o que seria descoberto computando-se $2^5 \mod 35, 2^6 \mod 35, 2^{29} \mod 35, 2^{64} \equiv 2^{29} \mod 35$. Passemos então ao candidato v = 5, o que produz o número racional 5/64 = 0.078125. A fração continuada de 0,078125 é [0; 12, 1, 4], ou seja,

$$\frac{1}{12 + \frac{1}{1 + \frac{1}{4}}}$$

mostrando que a primeira aproximação 0 + 1/12 = 1/12 revela a ordem r = 12.

Em palavras, o método de Shor é colocar todos os argumentos possíveis de $x^u \mod N$ em superposição uniforme e aplicar a Transformada de Fourier Quântica, o que modifica as amplitudes do estado de forma que as amplitudes produzindo probabilidades razoáveis estejam espaçadas por aproximadamente um múltiplo do inverso multiplicativo da ordem de x. Essa aproximação é obtida pelo método das frações continuadas e, com alguma sorte, a aproximação traz o denominador desejado, que é a ordem de x. Admirável estratégia.

Capítulo 5

Aplicações decorrentes da generalização

O Teorema 3.1.5 sugere como detectar o destino da escolha de x no algoritmo de Shor, restando a nós apenas detectar se as ordens de x módulo p, q podem ser distinguidas (no sentido da Definição 3.1.2) por um divisor primo de ordem de x módulo N, o que apresentamos na Seção 5.1. A mesma estratégia nos permite comparar as variações do algoritmo de Shor propostas na literatura, o que ilustramos na Seção 5.2. Por último, na Seção 5.3, apresentamos um novo teorema que dá condições suficientes para o algoritmo de Shor obter probabilidade 1 quanto à escolha de x, assumindo que a ordem do número $x \mod N$ foi computada corretamente pelo algoritmo quântico de cálculo da ordem. O Teorema 5.3.2 se aplica ao conjunto de compostos N = pq sendo p, q primos tais que $p = 2\ell + 1, q = 2m + 1 \in \ell, m$ são inteiros ímpares com $\ell \neq m$.

5.1 O comportamento do algoritmo de Shor relativo a um $x \mod N$

Responderemos "sim" à seguinte pergunta. Se nos for dada a fatoração prima de um grande composto N e um coprimo $x \mod N$, é possível dizer se o algoritmo de Shor conseguiria fatorá-lo sem executá-lo num computador quântico? Certamente não podemos hoje executar o algoritmo e olhar a resposta: uma vez que o composto é grande, precisaríamos de um computador quântico com grande capacidade, o que não temos. Mesmo tendo a fatoração prima de N, não é óbvio dizer se o algoritmo de Shor fatoraria N dado um x fixo. Até o momento, a única forma de saber o destino de uma execução do algoritmo de Shor seria calcular a ordem de $x \in \mathbb{Z}_N^*$, mas computar a ordem de um x arbitrário é impensável sem um computador quântico de grande capacidade, visto que não sabemos eficientemente calcular ordens de elementos em grupos num modelo clássico de computação. O Teorema 3.1.5 oferece uma estratégia.

Suponha que nos sejam dados dois primos grandes p, q de tamanhos similares de forma que N = pqseja um composto grande difícil de fatorar. Tendo p, q de antemão, sabemos que $\varphi(p) = p - 1$ and $\varphi(q) = q - 1$, sendo φ a função φ de Euler. O Teorema de Lagrange¹ nos garante que $\varphi(p)$ é múltiplo

¹O Teorema de Lagrange (J. E. Maxfield e M. W. Maxfield 1992, capítulo 3, teorema 3-8, página 65) nos diz que se existir um subgrupo H de um grupo finito G, seu tamanho é de um divisor do tamanho de G.

da ordem de qualquer $x \in Z_p^*$, ou seja, $x^{p-1} \equiv 1 \mod p$. O Teorema 3.1.5 nos garante que se pudermos encontrar um primo t distintivo relativo a $\operatorname{ord}(x,p)$, $\operatorname{ord}(x,q)$, então $c = \operatorname{ord}(x,N)/t$ é um expoente tal que $1 < \operatorname{mdc}(x^c - 1, N) < N$. O algoritmo de Shor espera que t = 2 seja tal primo distintivo, então, para saber se o algoritmo de Shor obtém sucesso ou não, basta descobrir se 2 é ou não distintivo relativo a $\operatorname{ord}(x,p)$, $\operatorname{ord}(x,q)$ no sentido da Definição 3.1.2. A detecção da distintividade de 2 pode ser feita por descobrir a maior potência de 2 que seja um fator de $\operatorname{ord}(x,p)$ e a maior potência de 2 que seja um fator de $\operatorname{ord}(x,q)$. Se elas forem diferentes, o algoritmo de Shor obtém sucesso; se não, ele fracassa. Vejamos primeiro como usar essa estratégia num exemplo concreto.

Seja p = 7907, q = 7919 de forma que N = 62615533. Como o algoritmo de Shor se comportaria se escolhêssemos x = 5? Aplicando o Teorema 3.1.5, basta detectarmos se 2 é um primo distintivo relativo a ord(5,7907) e ord(5,7919). Sabemos imediatamente que $\varphi(p) = 7908$ e que $\varphi(q) = 7918$. Calculamos agora a maior potência de 2 que divida ord(5, p) através de uma sequência de exponenciações modulares. Certamente $5^{\varphi(p)} = 1 \mod p$, então podemos remover um fator de 2 do expoente $\varphi(p)$ e computar nova exponenciação modular com o expo
ente $\varphi(p)/2$, obtendo $5^{\varphi(p)/2} = 5^{3953} \equiv 7906 \mod p$. Se $5^{\varphi(p)/2} \neq 1$, então deduzimos que $\varphi(p)/2$ não é múltiplo de ord(5, p), implicando que a maior potência de 2 que divide ord(5, p) é uma unidade a mais que a potência de 2 que divide 3953. Como 3953 é ímpar, deduzimos finalmente que a maior potência de 2 que divide ord(5, p) é 2¹. Computando similarmente para q, deduzimos que $\operatorname{ord}(5,q)$ é ímpar, logo 2⁰ é a maior potência de 2 que divide $\operatorname{ord}(5,q)$. Como essas potências de 2 são diferentes, 2 é um primo distintivo relativo a ord(5, p), ord(5, q) no sentido da Definição 3.1.2. Logo, o Teorema 3.1.5 garante que o algoritmo de Shor fatora N usando x = 5, assumindo que o algoritmo quântico de cálculo da ordem corretamente calcula ord(5, N). Por outro lado, se tivéssemos feito a escolha de x = 3, descobriríamos que 2 não é um primo distintivo relativo a relativo a ord(3, p) e ord(3, q), deduzindo, pelo Teorema 3.1.5, que o algoritmo de Shor fracassaria.

O algoritmo. Compute $x^{\gamma(i)} \mod p$ sendo $\gamma(i) = (p-1)/2^i$ para i = 1, 2, 3, ... até que $\gamma(i)$ não seja um número inteiro ou até que $1 < x^{\gamma(i)} \mod p$. Assim, sabemos que e(2, p-1) - i + 1 é o expoente de 2 na fatoração prima de $\operatorname{ord}(x, p)$, sendo e(2, p-1) o maior expoente de 2 que divide p-1, assim como especificado pela Definição 3.1.1. Faça similarmente para $\operatorname{ord}(x, q)$. Se a potência de 2 relativa a $\operatorname{ord}(x, p)$ for diferente da potência de 2 relativa a $\operatorname{ord}(x, q)$, então o algoritmo de Shor obteria sucesso desde que $\operatorname{ord}(x, N)$ fosse corretamente calculada pela rotina quântica; se as potências de 2 forem as mesmas, então $\operatorname{ord}(x, N)/2$ é tanto múltiplo de $\operatorname{ord}(x, p)$ quanto de $\operatorname{ord}(x, q)$ e, assim, o algoritmo de Shor fracassa, não importando o algoritmo quântico de cálculo da ordem.

O Algoritmo 5 expressa a estratégia em maior precisão. Com ele, podemos produzir uma estatística relativa a quantas vezes o algoritmo de Shor seria obrigado a invocar o algoritmo quântico de cálculo da ordem contra uma amostra aleatória de inteiros de tamanhos dos que se usa hoje em criptografia, o que nos permite comparar as estratégias para a escolha de $x \mod N$ disponíveis na literatura.

Algoritmo 5. Um predicado que afirma se ur	n primo t é distintivo relativo a $\operatorname{ord}(x, p), \operatorname{ord}(x, q).$
função distintivo? (t, x, p, q) retorne expoente-na-ordem $(t, x, p) \neq$	\triangleright É verdadeiro que t distingue $\operatorname{ord}(x,p)$ de $\operatorname{ord}(x,q)$? EXPOENTE-NA-ORDEM (t,x,q)
fim função	
função EXPOENTE-NA-ORDEM $(t, x, p, i = 0)$ se inteiro? $((p-1)/t^i)$ e $1 \equiv x^{(p-1)/t^i}$ m retorne EXPOENTE-NA-ORDEM (t, x, p)	⊳ Inteiros t, p são primos. od p então p, i + 1)
fim se	
retorne potências-de- $2((p-1)/t^{i-1})$ -	-i + 1
fim função	

5.2 O algoritmo de Shor (e suas variações) contra inteiros grandes

Quantas vezes, em média, o algoritmo de Shor teria que invocar o algoritmo quântico de cálculo da ordem, assumindo que a rotina quântica sempre calculasse a ordem de x corretamente? Evidência estatística exibida pela Tabela 5.1 sugere que seria aproximadamente 1.5 vezes em média. A Tabela 5.1, que consolida estatísticas produzidas pelo Algoritmo 5, sugere também que escolher x tal que J(x, N) = -1 torna o algoritmo de Shor aproximadamente 20% mais econômico² que o algoritmo de Shor de 1994 quanto a solicitar novas execuções do algoritmo quântico de cálculo da ordem e torna a versão estendida (Algoritmo 4) aproximadamente 30% mais econômica³ que o algoritmo de Shor de 1994 quanto a solicitar novas execuções do algoritmo quântico de cálculo da ordem.

tamanho da amostra	bits em N	Shor	J(x,N) = -1	estendido
10.000	256	1,427	1,144	1,001
500	512	1,432	1,141	1,000
500	1024	1,513	$1,\!127$	1,000
500	2048	1,378	1,110	1,002
500	4096	1,462	1,111	1,001

Tabela 5.1: Quantidade média de vezes que o algoritmo de Shor invocaria o algoritmo quântico de cálculo da ordem, assumindo que cada invocação produz corretamente a ordem $x \in \mathbb{Z}_N^*$. A média é descrita pela coluna "Shor". Os compostos N = pq foram construídos por multiplicar dois primos p, q escolhidos ao acaso. Similarmente, a coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um não-resíduo quadrático x é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor com x satisfazendo J(x, N) = -1. A coluna "tamanho" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para tamanhos acima de 256 bits, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

Na Tabela 5.1, apresentamos os resultados de computar médias do número de vezes que o algoritmo de Shor invocaria o algoritmo quântico de cálculo da ordem para calcular a ordem de x, assumindo que a rotina quântica sempre produzisse a resposta correta. Os compostos foram construídos pelo produto de dois primos distintos (de tamanhos similares) escolhidos ao acaso. A coluna "Shor" mostra

²Por exemplo, na amostra de 2048 bits, obtemos $1 - 1.111/1.378 \approx 19\%$.

³Por exemplo, na amostra de 2048 bits, obtemos $1 - 1.001/1.462 \approx 31\%$.

tamanho da amostra	bits em N	Shor	J(x,N) = -1	estendido
10.000	256	11	9	2
500	512	7	6	1
500	1024	9	5	1
500	2048	5	3	2
500	4096	10	5	2

Tabela 5.2: Número máximo de vezes que o algoritmo de Shor invocaria o algoritmo quântico de cálculo da ordem por execução. O número máximo é descrito pela coluna "Shor". Os compostos N = pq foram construídos por multiplicar dois primos p, q escolhidos ao acaso. Similarmente, a coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um não-resíduo quadrático x é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor com x satisfazendo J(x, N) = -1. A coluna "tamanho da amostra" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para tamanhos acima de 256 bits, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

as médias. A coluna "tamanho da amostra" descreve o número de compostos em cada amostra. Para compostos de tamanho 256 bits, a amostra continha dez mil compostos. Para tamanhos acima de 256 bits, reduzimos as amostras para quinhentos compostos.

No pior caso, quantas vezes o algoritmo de Shor teria que insistir em escolher diversos números x até fatorar um certo composto do tamanho que se usa em criptografia hoje? A Tabela 5.2, que consolida estatísticas produzidas pelo Algoritmo 5, sugere que seria próximo de 10 vezes, assumindo que a rotina quântica sempre produzisse a resposta correta. Similar à Tabela 5.1, os compostos foram construídos pelo produto de dois primos distintos (de tamanhos similares) escolhidos ao acaso⁴.

5.3 Condições para o Shor obter sucesso com probabilidade 1

Assumindo que o algoritmo quântico de cálculo da ordem calcule corretamente a ordem de coprimos $x \mod N$, o algoritmo de Shor é capaz de fatorar um produto de primos distintos da forma 2c+1 com probabilidade 1 desde que x > 1 seja selecionado satisfazendo J(x, N) = -1, mostrando assim que tais compostos são os casos mais fáceis para o algoritmo.

Um produto de safe primes pode ser fatorado pelo algoritmo de Shor com probabilidade 1 - 4/Nusando uma única execução bem-sucedida do algoritmo quântico de cálculo da ordem (Grosshans, Lawson, Morain e B. Smith 2015). O Teorema 5.3.2, entretanto, não se restringe a safe primes e aumenta a probabilidade para 1, assumindo que o algoritmo quântico de cálculo da ordem produza a ordem de x sem fracasso. Durante a prova do Teorema 5.3.2, precisamos do seguinte lema.

⁴A biblioteca OpenSSL gera compostos RSA usando o método incremental (Menezes, Oorschot e Vanstone 1996, capítulo 4, seção 4.4, subseção 4.4.1, nota 4.51), o que pode ser visto no código-fonte original nos procedimentos RSA_generate_key_ex, BN_generate_prime_ex, probable_prime, bn_is_prime_int no *commit* 0dca5ede0d[...] de 6 de março de 2021. Sabe-se que o método incremental introduz um inócuo viés de não-uniformidade (Brandt e Damgård 2003, seção 2). Nossas amostras geram um novo candidato primo aleatório toda vez que o candidato anterior não passa no teste Miller-Rabin. Assim geramos primos uniformemente. Uma vez que *strong* ou *safe primes* não são necessários (Rivest e Silverman 1999) para o criptossistema RSA, nosso método de seleção satisfaz a demanda atual para compostos RSA.

Lema 5.3.1. Seja p = 2c + 1, sendo c um inteiro ímpar. Seja x > 1. Se J(x, p) = -1, sendo J(x, p)o símbolo de Jacobi de x sobre p, então ord(x, p) é par. Por outro lado, se J(x, p) = 1, então ord(x, p)é ímpar.

Prova. Seja J(x, p) = -1 e $r = \operatorname{ord}(x, p)$. O Teorema de Lagrange nos garante que $\varphi(p) = p - 1 = 2c$ é múltiplo de r. Uma propriedade do símbolo de Jacobi é que $J(x, p) \equiv x^{(p-1)/2} \mod p$, logo podemos deduzir que $x^c \equiv -1 \mod p$. Como $r = \operatorname{ord}(x, p)$, então c não é múltiplo de r. Portanto, $1 < \operatorname{mdc}(r, 2)$ e, assim, r é par, como desejado.

Agora assuma que J(x,p) = 1. Usando novamente uma das propriedades do símbolo de Jacobi, podemos afirmar que $x^c \equiv 1 \mod p$. Como $r = \operatorname{ord}(x,p)$, então c é múltiplo de r. Por hipótese, c é ímpar, então r é ímpar, como desejado.

Teorema 5.3.2. Seja N = pq sendo p, q primos tais que $p = 2\ell + 1$, q = 2m + 1 e ℓ, m são inteiros ímpares com $\ell \neq m$. Para cada $x \in \mathbb{Z}_N^*$ tal que mdc(x, N) = 1 com x > 1, temos

J(x, N) = -1 se e somente se $1 < mdc(x^{r/2} - 1, N) < N$,

sendo J(x, N) o símbolo de Jacobi de x sobre N.

Prova. Provamos a implicação direta primeiro. Sem perda de generalidade, assuma J(x,p) = -1 e J(x,q) = 1. Seja $r_p = \operatorname{ord}(x,p)$ e $r_q = \operatorname{ord}(x,q)$. Pelo Lema 5.3.1, sabemos que r_p é par e r_q é ímpar. Pelo Lema 3.1.4, $r = \operatorname{mmc}(r_p, r_q)$ e, assim, r é par.

O Teorema de Lagrange garante que 2ℓ é múltiplo de r_p , e ℓ é ímpar por hipótese. Junto com o fato de que r_q é ímpar, concluímos que r pode ter no máximo um fator 2. Então, r/2 é ímpar. Como r/2 é ímpar, r/2 não pode ser múltiplo de r_p porque r_p é par. Logo, $x^{r/2} \neq 1 \mod p$, implicando $x^{r/2} - 1 \neq 0 \mod p$. Em outras palavras, $x^{r/2} - 1$ não é múltiplo de p. Similarmente, se r_q é ímpar, então r/2 tem que ser múltiplo de r_q . Assim, $x^{r/2} \equiv 1 \mod q$, implicando $x^{r/2} - 1 \equiv 0 \mod q$. Em outras palavras, $x^{r/2} = 1 \mod q$, implicando $x^{r/2} - 1 \equiv 0 \mod q$. Em outras palavras, $x^{r/2} = 1 \mod q$, implicando $x^{r/2} - 1 \equiv 0 \mod q$. Em outras palavras, $x^{r/2} - 1$ é múltiplo de q. Juntando os resultados, obtemos

$$1 < mdc(x^{r/2} - 1, N) < N.$$

Agora provamos a implicação conversa por considerar sua implicação reversa, ou seja, assumimos que J(x, N) = 1 e mostramos que $1 < \text{mdc}(x^{r/2}-1, N) < N$ não é satisfeito. Se J(x, N) = 1 com J(x, p) = -1 = J(x, q), então, pelo Lema 5.3.1, ambos r_p e r_q são pares. Estabelecemos anteriormente que r/2 é ímpar. Logo, $x^{r/2} \neq 1 \mod p, q$, implicando $x^{r/2} - 1 \neq 0 \mod p, q$. Portanto, $\text{mdc}(x^{r/2}-1, N) = 1$. Por outro lado, se $J(x, N) = 1 \mod J(x, p) = 1 = J(x, q)$, então ambos r_p , r_q são ímpares, implicando que $r = \text{mmc}(r_p, r_q)$ é ímpar também. Então r/2 não é um inteiro, como desejado.

Aplicando ambos Teoremas 5.3.2 e 3.1.5, computamos as estatísticas das Tabelas 5.3 e 5.4.

Na Tabela 5.3, apresentamos os resultados de se computar médias do número de invocações do algoritmo quântico de cálculo da ordem. Os compostos foram construídos pelo produto de dois primos p, q de tamanhos similares tal que ambos p-1, q-1 não são divisíveis por 4. A coluna "Shor" exibe o

tamanho da amostra	bits em N	Shor	J(x,N) = -1	estendido
10.000	256	2,002	1	1
500	512	1,984	1	1
500	1024	1,946	1	1
500	2048	1,972	1	1
500	4096	2,086	1	1

Tabela 5.3: Número médio de invocações do algoritmo quântico de cálculo da ordem. Os compostos foram construídos pelo produto de dois primos p, q escolhidos ao acaso tal que ambos p-1 e q-1 não são divisíveis por 4, ou seja, compostos que satisfaçam o Teorema 5.3.2. A coluna "Shor" exibe o número médio de execuções do algoritmo quântico de cálculo da ordem no algoritmo de Shor de 1994 sem qualquer refinamento. Similarmente, a coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um número x não-resíduo quadrático é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor (Algoritmo 4) escolhendo-se x tal que J(x, N) = -1. A coluna "tamanho" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para outros tamanhos, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

tamanho da amostra	bits em N	Shor	J(x,N) = -1	estendido
10.000	256	16	1	1
500	512	9	1	1
500	1024	8	1	1
500	2048	12	1	1
500	4096	8	1	1

Tabela 5.4: Número máximo de invocações do algoritmo quântico de cálculo da ordem por execução do algoritmo de Shor. Os compostos foram construídos pelo produto de dois primos p, q escolhidos ao acaso tal que ambos p-1 e q-1 não são divisíveis por 4, ou seja, compostos que satisfaçam o Teorema 5.3.2. A coluna "Shor" exibe o número máximo de execuções do algoritmo quântico de cálculo da ordem no algoritmo de Shor de 1994 sem qualquer refinamento. Similarmente, a coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um número x não-resíduo quadrático é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor (Algoritmo 4) escolhendo-se x tal que J(x, N) = -1. A coluna "tamanho" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para outros tamanhos, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

número médio de execuções do algoritmo quântico de cálculo da ordem pelo algoritmo de Shor de 1994 sem qualquer refinamento. Similarmente, a coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um número x não-resíduo quadrático é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor (Algoritmo 4) escolhendo-se x tal que J(x, N) = -1. A coluna "tamanho" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para outros tamanhos, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

Na Tabela 5.4, apresentamos o maior número de invocações do algoritmo quântico de cálculo da ordem dentre todas as execuções em cada amostra. Similar à Tabela 5.3, os compostos foram construídos pelo produto de dois primos p, q de tamanhos similares tal que ambos p-1, q-1 não são divisíveis por 4. A coluna "Shor" exibe o maior número de execuções do algoritmo quântico de cálculo

da ordem pelo algoritmo de Shor de 1994 sem qualquer refinamento. A coluna J(x, N) = -1 é relativa ao refinamento do algoritmo de Shor em que um número x não-resíduo quadrático é sempre selecionado. A coluna "estendido" é a versão estendida do algoritmo de Shor (Algoritmo 4) escolhendo-se x tal que J(x, N) = -1. A coluna "tamanho" descreve o tamanho da amostra. (Para N com tamanho de 256 bits, a amostra continha dez mil compostos. Para tamanhos acima de 256 bits, reduzimos as amostras para quinhentos.) A coluna "bits em N" descreve a quantidade de bits dos compostos em cada amostra.

Capítulo 6

Conclusões

Em nossa dissertação de mestrado (Chicayban Bastos 2019), apresentamos uma primeira tradução do algoritmo Rô de Pollard para um modelo quântico de computação. Tivemos a esperança de aplicar a Transformada de Fourier Quântica (Nielsen e Chuang 2004, capítulo 5) ao universo de busca do problema e obter um ganho exponencial. Sem sucesso na obtenção no resultado, recorremos a um plano alternativo de aplicar a busca quântica (Grover 1996), melhorando o método preliminarmente pelo fator quadrático esperado, mas dificuldades técnicas na implementação nos provocaram um atraso de um fator quadrático também, anulando o ganho obtido com o método de Grover. Todavia, escrevemos na dissertação que nosso trabalho poderia resultar numa "redescoberta do algoritmo de Shor" (Chicayban Bastos 2019, capítulo 5, página 84), o que se mostrou verdadeiro.

O Teorema 3.1.5 caracteriza os pares de números que o algoritmo Rô de Pollard deseja encontrar. A mesma caracterização se aplica ao algoritmo de Shor. Por isso o consideramos como um caso particular do Rô de Pollard. Apesar da caracterização, fatorar um composto por via da versão quântica do algoritmo Rô de Pollard (Algoritmo 2, Capítulo 3) exige a computação do período de uma função cíclica, além de precisarmos de um divisor desse período também.

Nas implementações propostas do Algoritmo 2, precisamos de fórmulas fechadas para as funções cíclicas usadas. As fórmulas fechadas que encontramos (Capítulo 3, Seções 3.5–3.6) têm uma dependência do algoritmo quântico de cálculo da ordem, o que pode ser visto como um obstáculo ao entendimento de que o algoritmo de Shor seja um caso particular de uma versão quântica do algoritmo Rô de Pollard.

Entretanto, a perspectiva que propomos é dada pelo Teorema 3.1.5 e não por qualquer implementação de uma versão quântica do algoritmo Rô de Pollard. A prova do Teorema 3.1.5 sugere um procedimento que possa ser implementado para obter a fatoração. É durante a implementação desse procedimento que enfrentamos a dependência do algoritmo quântico de cálculo da ordem. As famílias de funções cíclicas que possuam fórmulas fechadas conhecidas não são numerosas. Publicações que abordam o tema (Friedlander, Pomerance e Shparlinski 2001; Martin e Pomerance 2004; Kurlberg e Pomerance 2004) estudam estratégias que não são facilmente extensíveis às funções cíclicas que precisamos para a versão quântica do algoritmo Rô de Pollard. Mais importante talvez seja o fato de que o Teorema 3.1.5 nos deu um novo entendimento para o algoritmo Rô de Pollard e esse novo entendimento nos permitiu obter informações que até então pareciam inacessíveis como as estatísticas obtidas e apresentadas no Capítulo 5. Por exemplo, com que taxa de sucesso o algoritmo de Shor invoca o algoritmo quântico de cálculo da ordem? As Tabelas 5.1– 5.2 oferecem dados por via do Teorema 3.1.5 e do Algoritmo 5. Tivéssemos computadores quânticos

grandes em funcionamento hoje, saberíamos bem o que esperar como comportamento deles, assumindo que as máquinas se comportassem com uma estabilidade comparável aos computadores clássicos que temos hoje.

Oportunidades futuras são, por exemplo, explorar a rotina quântica do algoritmo de Shor na esperança de obter teoremas a respeito da Transformada de Fourier Quântica. As estatísticas apresentadas nas Seções 5.1–5.2 assumem que a ordem de $x \mod N$ é sempre obtida. Estratégias triviais de reproduzir na computação clássica os resultados da Transformada de Fourier Quântica exigem possuem crescimento exponencial. Na literatura hoje disponível encontra-se simulações da Transformada de Fourier Quântica para números pequenos (Dang, Hill e Hollenberg 2019; Jaques e Häner 2022). Shor observou que a Transformada de Fourier Quântica produz amplitudes de magnitude razoável aproximadamente a cada q/r unidades da origem (Shor 1999, seção 5, página 320), sendo r a ordem de $x \mod N$, q a potência de 2 satisfazendo $n^2 \leq q < 2n^2$ e n a quantidade de bits em N. Podemos usar esse fato e computar apenas esses grupos de pontos do gráfico da Transformada de Fourier Quântica? Pode ser uma estratégia que nos permita avançar o estado da arte em simulações clássicas do algoritmo de Shor e permitir obter estatísticas realistas sobre o método.

Bibliografia

- Aaronson, Scott (2008). "The limits of quantum". Em: Scientific American 298.3, pp. 62-69.
- Bach, Eric (1991). "Toward a Theory of Pollard's Rho Method". Em: Information and Computation 90, 139-155.
- Beauregard, Stephane (2002). Circuit for Shor's Algorithm using 2n + 3 qubits.
- Beckman, David, Amalavoyal N. Chari, Srikrishna Devabhaktuni e John Preskill (1996). "Efficient networks for quantum factoring". Em: *Physical Review A* 54.2, p. 1034.
- Bernstein, Daniel J., Nadia Heninger, Paul Lou e Luke Valenta (2017). "Post-quantum RSA". Em: 8th International Workshop, PQCrypto2017. Springer, pp. 311–329. ISBN: 978-3-319-59879-6.
- Brandt, Jørgen e Ivan Damgård (2003). "On Generation of Probable Primes by Incremental Search".
 Em: Advances in Cryptology—CRYPTO'92: 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16–20, 1992. Proceedings. Vol. 740. 1. Springer, p. 358.
- Brent, Richard P. (1980). "An Improved Monte Carlo Factorization Algorithm". Em: *BIT 20: 176–184, doi:10.1007/BF01933190*.
- Cao, Zhengjun (2005). A Note on Shor's Quantum Algorithm for Prime Factorization. Cryptology ePrint Archive, Report 2005/051.
- Chicayban Bastos, Daniel (2019). "Uma versão quântica do algoritmo Rô de Pollard". Diss. de mestr. Universidade Federal Fluminense.
- Chicayban Bastos, Daniel e Luis Antonio Kowada (2021). "How to detect whether Shor's Algorithm succeeds against large integers without a quantum computer". Em: XI Latin and American Algorithms, Graphs and Optimization Symposium, pp. 132–138.
- (2022). "A Quantum Version of Pollard's Rho of Which Shor's Algorithm is a Particular Case".
 Em: Computing and Combinatorics. Ed. por Yong Zhang, Dongjing Miao e Rolf Möhring. Cham: Springer International Publishing, pp. 212–219. ISBN: 978-3-031-22105-7.
- Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest e Clifford Stein (2009). Introduction to algorithms. 3^a ed. The MIT Press. ISBN: 978-0-262-03384-8.
- Crandall, Richard e Carl Pomerance (2005). Prime numbers. Springer. ISBN: 978-0387-25282-7.
- Dang, Aidan, Charles D. Hill e Lloyd C.L. Hollenberg (2019). "Optimising matrix product state simulations of Shor's algorithm". Em: *Quantum* 3, p. 116.
- Ekerå, Martin (2021). "On completely factoring any integer efficiently in a single run of an order-finding algorithm". Em: *Quantum Information Processing* 20.6, pp. 1–14.
- Friedlander, John, Carl Pomerance e Igor Shparlinski (2001). "Period of the power generator and small values of Carmichael's function". Em: *Mathematics of computation* 70.236, pp. 1591–1605.

- Gathen, Joachim von zur e Jürgen Gerhard (2013). *Modern Computer Algebra*. 3^a ed. Cambridge University Press. ISBN: 978-1-107-03903-2.
- Gauss, Carl F. (1986). *Disquisitiones Arithmeticae*. English Edition. Springer-Verlag. ISBN: 0-387-96254-9.
- Graham, Ronald L., Donald E. Knuth e Oren Patashnik (1989). *Concrete mathematics: a foundation for computer science*. 2nd. Addison-Wesley Publishing Company, Inc. ISBN: 0-201-55802-5.
- Grosshans, Frédéric, Thomas Lawson, François Morain e Benjamin Smith (2015). Factoring Safe Semiprimes with a Single Quantum Query. arXiv: 1511.04385 [quant-ph].
- Grover, Lov K. (1996). "A fast quantum mechanical algorithm for database search". Em: *Proceedings* of the twenty-eighth annual ACM symposium on Theory of computing. ACM, pp. 212–219.
- Hardy, Godfrey Harold e Edward Maitland Wright (1975). An introduction to the theory of numbers. Oxford University Press. ISBN: 0-19-853310-7.
- Heath, Thomas Little (1956). The thirteen books of Euclid's Elements. Courier Corporation.
- Hendy, M.D. (1975). "Euclid and the fundamental theorem of arithmetic". Em: Historia Mathematica 2.2, pp. 189–191.
- Jaques, Samuel e Thomas Häner (2022). "Leveraging state sparsity for more efficient quantum simulations". Em: ACM Transactions on Quantum Computing 3.3, pp. 1–17.
- Johnston, Anna M. (2017). Shor's Algorithm and Factoring: Don't Throw Away the Odd Orders. Cryptology ePrint Archive, Report 2017/083.
- Knill, Emmanuel (1995). On Shor's quantum factor finding algorithm: increasing the probability of success and tradeoffs involving the Fourier transform modulus.
- Knuth, Donald E. (1997). The Art of Computer Programming, volume 2, seminumerical algorithms. 3^a ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 978-0-201-89684-8.
- Koblitz, Neal (1994). A course in number theory and cryptography. 2^a ed. Graduate texts in mathematics. Springer. ISBN: 0-387-94293-9.
- Kurlberg, Pär e Carl Pomerance (2004). "On the period of the linear congruential and power generators". Em: *arXiv preprint math/0405120*.
- Lawson, Thomas (2015). "Odd orders in Shor's factoring algorithm". Em: Quantum Information Processing 14.3, pp. 831–838.
- Leander, Gregor (2002). "Improving the Success Probability for Shor's Factoring Algorithm". Em: arXiv preprint quant-ph/0208183.
- Lenstra, Arjen K (1994). "Factoring". Em: International Workshop on Distributed Algorithms. Springer, pp. 28–38.
- Liskov, Barbara, Alan Snyder, Russell Atkinson e Craig Schaffert (1977). "Abstraction mechanisms in CLU". Em: Communications of the ACM 20.8, pp. 564–576.
- Martin, Greg e Carl Pomerance (2004). "The iterated Carmichael λ -function and the number of cycles of the power generator". Em: arXiv preprint math/0406335.

- Martín-López, E. et al. (2013). "Experimental Realisation of Shor's Quantum Factoring Algorithm using Qubit Recycling". Em: 2013 Conference on Lasers and Electro-Optics - International Quantum Electronics Conference. Optica Publishing Group, IB_6_3.
- Maxfield, John E. e Margaret W. Maxfield (1992). *Abstract algebra and solution by radicals*. Dover Publications, Inc. ISBN: 0-486-67121-6.
- McCoy, Neal H. e Gerald J. Janusz (1992). Introduction to Modern Algebra. 5^a ed. Wm. C. Brown Publishers. ISBN: 0-697-08570-8.
- Menezes, A. J., P. C. Oorschot e S. A. Vanstone (1996). *Handbook of Applied Cryptography*. 3^a ed. Discrete Mathematics and Its Applications. CRC Press. ISBN: 0-8493-8523-7.
- Mermin, N. David (2007). *Quantum Computer Science: An Introduction*. 1^a ed. Cambridge University Press. ISBN: 978-0-521-87658-2.
- Miller, Gary L. (1976). "Riemann's Hypothesis and Tests for Primality". Em: Journal of computer and system sciences 13.3, pp. 300–317.
- Nielsen, Michael A. e Isaac L. Chuang (2004). Quantum computation and quantum information. 10th Anniversary Edition. Cambridge Series on Information and the Natural Sciences. Cambridge University Press. ISBN: 0-521-63503-9.
- Pollard, John M. (1975). "A Monte Carlo method for factorization". Em: *BIT Numerical Mathematics* 15 (3): 331–334.
- Rabin, Michael O. (jan. de 1979). Digitalized signatures and public-key functions as intractable as factorization. Rel. técn. MIT/LCS/TR-212. Cambridge, Massachusetts: Massachusetts Institute of Technology.
- Rivest, Ronald L., Adi Shamir e Leonard Adleman (fev. de 1978). "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". Em: Commun. ACM 21.2, pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342.
- Rivest, Ronald L. e Robert D. Silverman (1999). "Are Strong Primes Needed for RSA?" Em: *The 1997* RSA Laboratories Seminar Series, Seminar Proceedings.
- Schönhage, Arnold e Volker Strassen (1971). "Schnelle Multiplikation grosser Zahlen". Em: Computing 7.3, pp. 281–292.
- Shor, Peter W. (1994). "Algorithms for Quantum Computation: Discrete Logarithms and Factoring".
 Em: Proceedings of the 35th Annual Symposium on Foundations of Computer Science. SFCS '94.
 Washington, DC, USA: IEEE Computer Society, pp. 124–134. ISBN: 0-8186-6580-7. DOI: 10.1109/ SFCS.1994.365700.
- (1997). "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". Em: SIAM Journal on Computing 26.5, pp. 1484–1509. DOI: 10.1137/s0097539795293172.
- (1999). "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". Em: SIAM review 41.2, pp. 303–332.
- Simon, Daniel R. (1994). "On the Power of Quantum Computation". Em: Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, pp. 116–123. ISBN: 0-8186-6580-7.

- Smolin, John A, Graeme Smith e Alexander Vargo (2013). "Oversimplifying quantum factoring". Em: *Nature* 499.7457, pp. 163–165.
- Stinson, Douglas Robert e Maura B. Paterson (2018). Cryptography. Theory and Practice. 4^a ed. CRC Press. ISBN: 978-1-1381-9701-5.
- van Meter, Rodney e Kohei M. Itoh (2005). "Fast quantum modular exponentiation". Em: *Physical Review A* 71.5, pp. 052320-1–052320-12. DOI: 10.1103/PhysRevA.71.052320.
- Vedral, Vlatko, Adriano Barenco e Artur Ekert (1996). "Quantum networks for elementary arithmetic operations". Em: *Physical Review A* 54.1, pp. 147–153. DOI: 10.1103/PhysRevA.54.147.
- Wagstaff, Samuel S. (2013). *The Joy of Factoring*. Vol. 68. American Mathematical Society. ISBN: 978-1-4704-1048-3.
- Williams, Hugh C e Jeffrey Outlaw Shallit (1994). "Factoring integers before computers". Em: *Mathematics of computation* 48.
- Xu, Guoliang, Daowen Qiu, Xiangfu Zou e Jozef Gruska (2018). "Improving the Success Probability for Shor's Factorization Algorithm". Em: *Reversibility and Universality*. Springer, pp. 447–462.

APÊNDICE A – Uma descrição de um circuito quântico para o algoritmo Rô de Pollard

Descrevemos agora um circuito quântico usando portas lógicas quânticas elementares que implementam uma versão quântica do algoritmo Rô de Pollard. Por clareza, implementamos o circuito para a função $f(x) = x^2 + 2x \mod N$. Em outras palavras, escolhemos a = 1 e b = 2 na Equação 3.3 de forma que $\alpha \equiv 2 \mod N$ e

$$g(i) \equiv (x_0 + 1)^{2^i \mod r} - 1 \mod N,$$

sendo $r = \operatorname{ord}(x_0+1,N) e x_0$ é um primeiro elemento da sequência escolhido adequadamente. Qualquer outra função da família definida pela Equação 3.3 poderia ser implementada por passos similares.

$$\begin{aligned} |x\rangle - & \underset{\text{point}}{\overset{\text{Z}}{\xrightarrow{}}} & - |x\rangle \\ |y\rangle - & \underset{\text{vo}}{\overset{\text{W}}{\xrightarrow{}}} & - |y \oplus c^{x} \mod N \end{aligned}$$

Figura A.1: Um operador para a exponenciação modular

Tomemos $N = 11 \times 13$ usando $x_0 = 2$ como exemplo. Primeiro precisamos invocar o algoritmo quântico de cálculo da ordem para computar

$$r = \operatorname{ord}(x_0 + 1, 143) = \operatorname{ord}(3, 143) = 15,$$

usando o operador ilustrado pela Figura A.1.



Figura A.2: O circuito para operador U usando $x_0 = 2$.

Definimos o operador U para calcular g(i) como $U|i\rangle|y\rangle \rightarrow |i\rangle|y \oplus f^i(x_0)\rangle$. No exemplo,

$$U|i\rangle|y\rangle \rightarrow |i\rangle|y \oplus (3^{2^i \mod 15} - 1 \mod 143)\rangle.$$

O circuito que implementa U é illustrado pela Figura A.2.

Os operadores usados em U são o operador da exponenciação modular descrito pela Figura A.1 e o operador modular SUB, ambos operadores bem conhecidos (van Meter e M. Itoh 2005; Vedral, Barenco e Ekert 1996).

Agora descrevemos os passos do algoritmo quântico de cálculo do período ilustrado pela Figura A.3. O estado inicial do sistema é

$$|\Psi_0\rangle = |N\rangle_n |0\rangle_\ell |0\rangle_n,$$

sendo $n = \text{floor}(\lg N) + 1$ a quantidade de bits necessária para representar $N \in 2^{\ell}$, o número de elementos calculados pelo operador QFT. Em geral, o tamanho ℓ do segundo registrador satisfaz (Shor 1999) $N^2 \leq 2^{\ell} < N^2 + 1$. Os bits auxiliares não são exibidos pela Figura A.3.



Figura A.3: O algoritmo quântico de cálculo do período usando U da Figura A.2 para fatorar N.

Depois que as portas Hadamard são aplicadas, obtemos

$$|\Psi_1\rangle = |143\rangle_n H^{\otimes \ell} |0\rangle_\ell |0\rangle_n = \frac{1}{2^{\ell/2}} \sum_{i=0}^{2^{\ell-1}} |143\rangle_n |i\rangle_\ell |0\rangle_n$$

Nossa estratégia para restringir g ao ciclo, antes de U ser aplicado, é dar a g um valor inicial que é um elemento no ciclo, então usamos o operador ADDER para deslocar o registrador N bits à frente, produzindo o estado

$$|\Psi_2\rangle = \frac{1}{2^{\ell/2}} \sum_{i=0}^{2^{\ell-1}} |143\rangle_n |143+i\rangle_\ell |0\rangle_n.$$

O operador ADDER é definido como $ADD: |A\rangle|B\rangle \rightarrow |A\rangle|A+B\rangle$. O tamanho do segundo registrador precisa ser maior ou igual ao primeiro registrador neste operador. Veja Vedral, Barenco and Eckert para mais informações sobre a implementação do operador ADDER (Vedral, Barenco e Ekert 1996). Um q-bit $|0\rangle$ extra é necessário para evitar um possível *overflow*. Em nosso caso, esse q-bit extra é parte do segundo registrador do ADDER, mas a porta Hadamard não é aplicada a esse q-bit. Por simplicidade, esse q-bit é omitido na Figura A.3 e na descrição dos estados do sistema.

O próximo passo é a aplicação de U, levando o estado do sistema a

$$|\Psi_3\rangle = U|\Psi_2\rangle = \frac{1}{2^{\ell/2}}|143\rangle_n \sum_{i=0}^{2^{\ell}-1} |143+i\rangle_\ell |3^{2^{143+i} \bmod 15} - 1 \bmod 143\rangle_n$$

Se N = 143, então $2^{143} \equiv 2^3 \mod 15$ e n = 8. Tomando-se $\ell = \text{floor}(\lg N^2) + 1 = 15$, obtemos

$$|\Psi_3\rangle = \frac{1}{\sqrt{32768}} |143\rangle \sum_{i=0}^{32767} |143+i\rangle |3^{2^{3+i} \mod 15} - 1 \mod 143\rangle.$$

Podemos reescrever $|\Psi_3\rangle$ como

$$\begin{split} |\Psi_{3}\rangle &= \frac{1}{\sqrt{32768}} |143\rangle \left(\begin{array}{c} (|143\rangle + |147\rangle + |151\rangle + |155\rangle + \dots + |32907\rangle \right) & |125\rangle + \\ & (|144\rangle + |148\rangle + |152\rangle + |156\rangle + \dots + |32908\rangle) & |2\rangle & + \\ & (|145\rangle + |149\rangle + |153\rangle + |157\rangle + \dots + |32909\rangle) & |8\rangle & + \\ \end{split}$$

 $(|146\rangle + |150\rangle + |154\rangle + |158\rangle + \dots + |32910\rangle) |80\rangle$).

O próximo passo é a aplicação do operador reverso de ADDER, produzindo o estado

$$\begin{split} |\Psi_{4}\rangle &= \frac{1}{\sqrt{32768}} |143\rangle ((|0\rangle + |4\rangle + |8\rangle + |12\rangle + \dots + |32764\rangle) & |125\rangle + \\ & (|1\rangle + |5\rangle + |9\rangle + |13\rangle + \dots + |32765\rangle) & |2\rangle + \\ & (|2\rangle + |6\rangle + |10\rangle + |14\rangle + \dots + |32766\rangle) & |8\rangle + \\ & (|3\rangle + |7\rangle + |11\rangle + |15\rangle + \dots + |32767\rangle) & |80\rangle). \end{split}$$

O estado final antes de uma medição é $|\Psi_5\rangle = QFT^{\dagger}|\Psi_4\rangle$, ou seja,

$$|\Psi_5\rangle = \frac{1}{32768} |143\rangle \sum_{i=0}^{32767} \sum_{k=0}^{32767} e^{2\pi \iota i k/32768} |i\rangle |3^{2^{3+i} \bmod 15} - 1 \bmod 143\rangle,$$

sendo $\iota = \sqrt{-1}$.

Usamos agora o princípio da medição implícita (Nielsen e Chuang 2004, capítulo 4, seção 4.4, páginas 185–187) para assumir que o segundo registrador foi medido, produzindo um resultado aleatório de {2, 8, 80, 125}. Então, neste exemplo, há quatro possíveis resultados da medição do primeiro registrador do estado $|\Psi_5\rangle$, sendo que todos têm a mesma probabilidade de serem medidos. Podemos obter 0, 8192, 16384 ou 24576, ou seja, $0/2^{\ell}$, $r_g/2^{\ell}$, $2r_g/2^{\ell}$ ou $3r_g/2^{\ell}$, sendo $r_g = 4$ o período do ciclo produzido pela função g. Para mais informações sobre extrair r_g a partir da medição de $|\Psi_5\rangle$, consulte (Shor 1999) e (Nielsen e Chuang 2004, capítulo 5, seção 5.3.1, página 226).

Por fim, computamos

$$m = \mathrm{mdc}(g(N + r_q/2) - g(N), N) = \mathrm{mdc}(8 - 125, 143) = 13,$$

como desejado.