Algoritmos aleatorizados com oráculo para MCSP: aplicações para o problema do resíduo quadrático e do logaritmo discreto

Nicollas M. Sdroievski **Murilo V. G. da Silva**

Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR) – Curitiba
Av. Sete de Setembro, 3165 – 80230-901 – Curitiba-PR, Brasil

Abstract. *In this paper we study the relation between three problems that are believed to be* $\mathbf{NP}$*-intermediate. More precisely, we explicitly present algorithms with access to an oracle for the minimum circuit size problem, which run in randomized polynomial time. The first algorithm solves the quadratic residue problem and the second algorith solves the discrete logarithm problem.*

Resumo. *Neste artigo estudamos a relação entre três problemas que são candidatos a serem* $\mathbf{NP}$*-intermediários. Mais precisamente, apresentamos explicitamente dois algoritmos polinomiais aleatorizados que fazem acesso à um oráculo para o problema de minimização de circuitos. O primeiro algoritmo resolve o problema do resíduo quadrático e o segundo o problema do logaritmo discreto.*

1. Introdução

Richard Ladner mostrou em 1975 que se $\mathbf{P} \neq \mathbf{NP}$, então existem problemas, conhecidos como problemas $\mathbf{NP}$-intermediários, que estão em $\mathbf{NP} \setminus \mathbf{P}$ e que não são $\mathbf{NP}$-completos [Ladner 1975]. Embora existam alguns candidatos a pertencerem a tal classe, mostrar diretamente que um problema é NP-intermediário é extremamente difícil, pois tal resultado implicaria que $\mathbf{P} \neq \mathbf{NP}$. Por conta disso, maior parte da pesquisa nesta área gira em torno criação de conexões entre problemas candidatos a serem $\mathbf{NP}$-intermediários e em esclarecer como estes problemas se relacionam com outras classes de complexidade [Arora and Barak 2009]. Em dois trabalhos nesta linha [Allender et al. 2006, Allender and Das 2014] foi mostrado que um oráculo para o problema de minimização de circuitos (MCSP) tem o poder de resolver em tempo polinomial aleatorizado problemas como fatoração e isomorfismo de grafos, e até mesmo qualquer problema da classe $\mathbf{SZK}$, que conjectura-se estar estritamente contida em $\mathbf{NP} \setminus \mathbf{P}$.

Neste artigo mostramos explicitamente dois algoritmos polinomiais aleatorizados com oráculo para MCSP, um para o problema do resíduo quadrático (RQUAD) e outro para o problema do logaritmo discreto (LOGD). Os três problemas (MCSP, RQUAD e LOGD) são candidatos a serem $\mathbf{NP}$-intermediários. O fato de que os algoritmos apresentados neste artigo devam existir já era conhecido, mas de maneira indireta, sendo consequência de reduções. Ambos os problemas estão em $\mathbf{RP}^{\text{MCSP}}$ pois possuem provas de conhecimento zero do tipo "v-bit" [Allender and Das 2014], e no caso específico de RQUAD sabemos também que RQUAD $\in \mathbf{ZPP}^{\text{MCSP}}$, pois este é redutível ao problema da fatoração, que por sua vez está em $\mathbf{ZPP}^{\text{MCSP}}$ [Allender et al. 2006]. Os dois algoritmos que apresentamos aqui são bastante simples e diretos.

Na seção 2 definimos os problemas MCSP, RQUAD e LOGD e na seção 3 descrevemos brevemente por que um oráculo para o problema MCSP é bastante poderoso. Na seção

4 apresentamos os dois algoritmos, que são a contribuição deste trabalho. Por questões de espaço, vamos omitir definições usuais de classes de complexidade, oráculos, provas iterativas, provas de conhecimento zero e teoria dos grupos. As definições e notações usadas aqui são as mesmas usadas em [Arora and Barak 2009].

2. Problemas candidatos à classe NP-intermediária

Pouco se sabe sobre a complexidade de MCSP[1], e embora acredite-se que o problema não esteja em $\mathbf{P}$, uma prova de $\mathbf{NP}$-completude utilizando reduções independentes à oráculos (como é o caso de todas as reduções atuais para MCSP) causaria um colapso ao segundo nível na hierarquia polinomial [Hirahara and Watanabe 2015], algo que conjectura-se improvável. Ambos RQUAD e LOGD também são considerados intratáveis, porém possuem algoritmos sub-exponenciais, o que refutaria e hipótese de tempo exponencial no caso de serem provados $\mathbf{NP}$-completos. Definimos agora os três problemas:

MCSP: dados (x, k), onde x é uma string que representa a tabela verdade de uma função booleana f_x e $k \in \mathbb{N}^*$, responder se f_x possui circuito de tamanho menor ou igual a k.

RQUAD: dados (z, n), onde $z \in \mathbb{Z}_n^*$, responder se existe $x \in \mathbb{Z}_n^*$ tal que $z = x^2 (\text{mod } n)$. Note que o conjunto dos resíduos quadráticos módulo um inteiro n, chamado de RQ_n é um grupo com multiplicação. Dessa forma, se $x, y \in RQ_n$, então $xy \in RQ_n$. Além disso, se $y \in RQ_n$ e r é um elemento aleatório de RQ_n, então yr também é um elemento aleatório de RQ_n. Por último, se $y \in RQ_n$ e $x \notin RQ_n$, então $xy \notin RQ_n$.

LOGD: dados (z, b, n) onde $z, b \in \mathbb{Z}_n$ e n é primo, encontrar um inteiro $x \in \mathbb{Z}_n$ tal que $z = b^x (\text{mod } n)$. Note que x pode não existir caso b não seja um gerador de $\mathbb{Z}_n$. Denotando por $H \subseteq \mathbb{Z}_n$ o subgrupo gerado por b, observe que se $x, y \in H$, então $xy \in H$. Além disso, se $y \in H$ e r é um elemento aleatório de H, então yr também é um elemento aleatório de H. Por último, se $y \in H$ e $x \notin H$, então $xy \notin H$.

3. O poder do problema de Minimização de Circuitos

Começamos esta seção com a definição de complexidade de Kolmogorov limitada em tempo, um conceito que tem conexões interessantes com MCSP. Seja U uma Máquina de Turing e x uma string, definimos:

$$\text{KT}_U(x) = \min\{|d|+t; \forall b \in \{0, 1, *\} \forall i \leq |x|+1 : U^d(i, b) \text{ aceita em } t \text{ passos sse } x_i = b\}$$

Ou seja, a máquina U, com acesso a descrição d, deve reconhecer corretamente os símbolos de x, dado seu índice. Como esse mecanismo não determina o tamanho de x, assumimos que para a posição $i = |x| + 1$ o símbolo correto é $*$. Para simplificar, podemos fixar uma Máquina de Turing Universal U qualquer e definir $\text{KT}(x) = \text{KT}_U(x)$, pois a medida KT pode sofrer um *overhead* no máximo logarítmico caso a máquina U seja substituída por alguma outra [Allender et al. 2006].

Considerando que a string x representa a tabela verdade de uma função booleana f_x, temos que o tamanho do menor circuito que computa f_x é aproximadamente o valor de $\text{KT}(x)$. Sendo s o tamanho desse circuito e $|x| = m$, temos [Allender and Das 2014]:

$$\left(\frac{s}{\log m}\right)^{\frac{1}{4}} \leq \text{KT}(x) \leq \mathcal{O}(s^2(\log s + \log\log m))$$

[1] A sigla MCSP vem do inglês *minimum circuit size problem.*

Levando em consideração que strings x geradas por geradores pseudoaleatórios possuem $\mathrm{KT}(x)$ pequeno, uma máquina com oráculo para MCSP pode tomar como entrada a string x e aceitar se e somente se x possui circuitos maiores que, por exemplo, $\sqrt{|x|}$. Isso é interessante pois garantimos que essa máquina aceita a maior parte das strings (pois quase todas as funções booleanas precisam de circuitos grandes para serem computadas [Arora and Barak 2009]), porém não aceita nenhuma string x com $\mathrm{KT}(x)$ pequeno, o que torna essa máquina um ótimo teste para distinguir dentre a distribuição uniforme e uma distribuição gerada por um gerador pseudoaleatório [Allender and Das 2014].

Devido à forte e clássica conexão entre geradores pseudoaleatórios e funções irreversíveis [Hästad et al. 1999], temos então o seguinte teorema:

Teorema 3.1. *[Allender et al. 2006] Seja L uma linguagem de densidade polinomial [2] tal que para algum $\epsilon > 0$ e para todo $x \in L$, $KT(x) \geq |x|^{\epsilon}$. Seja $f(y, x)$ uma função computável em tempo polinomial no tamanho de x. Então existe um algoritmo aleatorizado polinomial N^L e um polinômio q tal que:*

$$Pr_{x\in\{0,1\}^n,s}[f(y, N^L(y, f(y, x), s)) = f(y, x)] \geq \frac{1}{q(n)}$$

A string s acima representa as escolhas aleatórias de N^L. Pelas características já apontadas de MCSP, podemos utilizar um oráculo para esse problema no lugar da linguagem L do Teorema 3.1. Esse poder de inverter funções polinomiais em uma fração polinomial das entradas permite à uma máquina com oráculo para MCSP ser utilizada para obter algoritmos aleatorizados para vários problemas candidatos à $\mathbf{NP}$-intermediários, como fatoração [Allender et al. 2006], isomorfismo de grafos e distância estatística, um problema que é $\mathbf{SZK}$-completo [Allender and Das 2014].

O seguinte teorema é de especial interesse, pois conecta provas de conhecimento zero perfeitas do tipo "v-bit" (ambos RQUAD e LOGD admitem tais provas [Goldwasser et al. 1989, Malka 2008]) com o problema MCSP.

Teorema 3.2. *[Allender and Das 2014] Todas as linguagens pertencentes à classe $\mathbf{PZK}$ com protocolos do tipo "v-bit" também pertencem à $\mathbf{RP}^{\mathrm{MCSP}}$*

4. Algoritmos aleatorizados com oráculo para minimização de circuitos

Nessa seção se encontra a principal contribuição desse trabalho, algoritmos aleatorizados para RQUAD e LOGD, que se utilizam de um oráculo para MCSP. Pelo Teorema 3.2 já era conhecido que ambos pertencem a $\mathbf{RP}^{\mathrm{MCSP}}$, porém a prova do referido teorema não apresenta diretamente os algoritmos.

Para o problema RQUAD, definimos primeiramente a função polinomial $f : \mathbb{Z}_n^* \times \mathbb{Z}_n^* \to \mathbb{Z}_n^*$, que em conjunto com a máquina N^{MCSP} nos permitirá encontrar um certificado para a instância de RQUAD. Definimos $f(u, v) = f_u(v) = uv^2 (\mathrm{mod}\ n)$.

Pelo teorema 3.1, então sabemos que existe um algoritmo aleatorizado polinomial com oráculo N^{MCSP} e um polinômio q tal que:

$$\mathrm{Pr}_{v\in\mathbb{Z}_n^*,s}[f_u(N^{\mathrm{MCSP}}(u, f_u(v), s)) = f_u(v)] \geq \frac{1}{q(|v|)}$$

[2] L possui pelo menos $2^n/n^k$ strings para cada tamanho n e para algum k

Isso é, dado um resultado $f_u(v) = y$, o algoritmo N^{MCSP} consegue encontrar um outro número $t \in \mathbb{Z}_n^*$ tal que $f_u(v) = f_u(t)$ com probabilidade igual ou maior que $1/q(|v|)$.

Ideia principal do algoritmo: computar $y = f_z(r)$ (onde r é um elemento aleatório de $\mathbb{Z}_n^*$) porém executar o algoritmo N^{MCSP} com o argumento r^2 fixo para inverter o resultado, e não z. Esse passo é possível pois se $z \in RQ_N$, então o elemento y aparece na imagem de f_{r^2}, e o valor x tal que $f_{r^2}(x) = y$ é justamente o certificado para z.

Algoritmo para RQUAD: Dados como entrada (z, n) tal que $z \in \mathbb{Z}_n^*$, o algoritmo proposto executa independentemente os seguintes passos $2q(|n|)$ vezes:

(1) Sorteie aleatoriamente um número $r \in \mathbb{Z}_n^*$ e uma string aleatória s. (2) Compute $y = f_z(r) = zr^2 \pmod n$. (3) Execute $N^{\text{MCSP}}(r^2, y, s)$ e obtenha o número $x \in \mathbb{Z}_n^*$. (4) Reporte sucesso se $z = x^2 \pmod n$.

O algoritmo aceita se pelo menos umas das tentativas obtém sucesso. Observe que se $z \notin RQ_n$, então não haverá nenhum sucesso, pois y não aparece na imagem de f_{r^2} já que é o produto de um resíduo e um não resíduo. Por outro lado, se $z \in RQ_n$ temos probabilidade $1/q(|n|)$ de encontrar x tal que $z = x^2 (\text{mod } n)$, e executar os passos apresentados $2q(|n|)$ vezes garante que a probabilidade de que haja pelo menos um sucesso seja maior ou igual a $3/4$. Dessa forma, o algoritmo apresentado é da classe $\mathbf{RP}^{\text{MCSP}}$.

Podemos utilizar a mesma ideia para encontrar um algoritmo para o problema do LOGD, tendo como entrada (z, b, n), primeiro testamos se n é primo. Definimos a função $f : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ como $f(u, v) = f_u(v) = ub^v (\text{mod } n)$. Basta então computar $y = f_z(r)$ para r aleatório e utilizar a máquina N^{MCSP} com a entrada (b^r, y, s) para tentar inverter a função com o parâmetro fixo b^r. O algoritmo então é completamente análogo, também é executado $2q(|n|)$ vezes e apresenta x caso esse seja encontrado, caso contrário indica que z não está na órbita de b. Argumentos semelhantes aos utilizados na ideia do algoritmo para RQUAD podem ser apresentados para mostrar que esse algoritmo é da classe $\mathbf{RP}^{\text{MCSP}}$.

Referências

Allender, E., Buhrman, H., Koucký, M., van Melkebeek, D., and Ronneburger, D. (2006). Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493.

Allender, E. and Das, B. (2014). Zero knowledge and circuit minimization. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:68.

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition.

Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208.

Hirahara, S. and Watanabe, O. (2015). Limits of minimum circuit size problem as oracle. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:198.

Hästad, J., Impagliazzo, R., Levin, L. A., and Luby, M. (1999). A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396.

Ladner, R. E. (1975). On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171.

Malka, L. (2008). *A Study of Perfect Zero-knowledge Proofs*. PhD thesis, Victoria, B.C., Canada, Canada. AAINR47335.