Os Desafios ao Desenvolvimento dos Algoritmos Quânticos

Por Marcos Eduardo Elias: doutor em matemática, pesquisador em Teoria da Computação, engenheiro e empresário em série.

I. Algumas Definições Preliminares

Um algoritmo roda em tempo polinomial (*Polynomial time*) se, dado uma entrada de tamanho n (número de bits ou elementos necessários para representar a entrada do problema), seu tempo de execução é limitado por uma função polinomial em n. Ou seja, se um algoritmo roda em tempo polinomial, significa que seu tempo de execução pode ser escrito como uma função polinomial de n, como $0(n^k)$ para algum número fixo k. O crescimento do tempo de execução não é exponencial, o que significa que ele continua viável mesmo para entradas moderadamente grandes. Um algoritmo roda em tempo polinomial determinístico (pertence, portanto, à classe P) em uma máquina determinística se ele sempre segue um único caminho de execução e resolve o problema em tempo polinomial. Novamente: Problemas em P são considerados eficientemente solúveis porque crescem em tempo polinomial. Isso significa que mesmo para entradas grandes, esses problemas podem ser resolvidos dentro de um tempo razoável.

A classe P contém todos os problemas de decisão que podem ser resolvidos em tempo polinomial por uma Máquina de Turing Determinística (DTM), sendo uma Máquina Determinística (DTM - Máquina de Turing Determinística) um modelo computacional que, para um determinado estado e entrada, sempre executa uma única transição bem definida. Ou seja, não há incerteza no que a máquina faz em cada passo.

A Máquina de Turing Não Determinística (NTM - Nondeterministic Turing Machine) é uma generalização da Máquina de Turing Determinística (DTM), mas com a diferença crucial de que, em cada passo, a máquina pode escolher entre múltiplas transições possíveis. Em um modelo teórico, ela pode explorar todos os caminhos simultaneamente, como se houvesse um número infinito de "universos paralelos".

O tempo polinomial não determinístico (classe NP) refere-se a problemas que podem ser verificados em tempo polinomial, mas não necessariamente resolvidos em tempo polinomial em uma máquina determinística. Em outras palavras, são problemas cujas soluções podem ser verificadas rapidamente (em tempo polinomial), mas que não podem não ser facilmente encontradas.

A classe NP-difícil (NP-Hard) contém problemas que são pelo menos tão difíceis quanto qualquer problema em NP, mas não necessariamente pertencem a NP. Um problema X é dito NP-difícil se qualquer problema em NP pode ser reduzido a X em tempo polinomial. Isso significa que, se pudéssemos resolver X rapidamente, também poderíamos resolver qualquer problema de NP rapidamente. Resolver um problema NP-difícil em tempo polinomial significaria resolver qualquer problema de NP também em tempo polinomial. No entanto, NP-difícil pode conter problemas que não estão em NP, ou seja, problemas para os quais a solução não pode ser verificada em tempo polinomial.

A classe NP-completo contém os problemas mais difíceis de NP. Esses problemas são simultaneamente membros de NP e NP-difíceis, ou seja, são problemas cuja solução pode ser verificada em tempo polinomial e que, se conseguirmos resolvê-los rapidamente, poderíamos resolver qualquer problema de NP rapidamente.

II. A Importância da Questão P = NP na Computação Teórica

A questão P = NP? é um dos problemas mais fundamentais e desafiadores da computação teórica. Ela questiona se todo problema cuja solução pode ser verificada em tempo polinomial também pode ser resolvido em tempo polinomial. Em outras palavras, será que existe um algoritmo eficiente (polinomial) para resolver qualquer problema cujas soluções podem ser verificadas rapidamente? Se P = NP, isso significaria que todos os problemas NP-completos, que são atualmente considerados intratáveis, poderiam ser resolvidos em tempo polinomial. Isso teria um impacto tremendo, pois abriria a possibilidade de resolver de maneira eficiente problemas muito complexos que hoje exigem tempo exponencial.

A questão P = NP também é crucial porque mapeia os limites do que pode ser computado eficientemente. Resolver ou provar essa questão ajudaria a entender melhor os limites da computação clássica e, por consequência, da computação prática. Ela está diretamente relacionada com complexidade algorítmica, reduções polinomiais e a hierarquia de complexidade, que são os pilares de toda a computação teórica moderna.

III. Os Impactos da Computação Quântica sobre a Questão P = NP

Na edição de 15 de fevereiro de 2007, a revista The Economist afirmou que, em princípio, computadores quânticos poderiam resolver rapidamente um conjunto particularmente difícil de desafios matemáticos chamados problemas NP-completos, que nem os mais avançados supercomputadores tradicionais conseguiriam resolver rapidamente, por mais que evoluíssem.

Se realmente conseguíssemos construir um computador mágico capaz de resolver um problema NP-completo instantaneamente, o mundo seria um lugar muito diferente: poderíamos pedir ao nosso computador mágico para procurar quaisquer padrões que pudessem existir em dados do mercado de ações, em registros climáticos ou na atividade cerebral. Ao contrário dos computadores atuais, encontrar esses padrões seria algo completamente rotineiro e não exigiria nenhum entendimento detalhado do assunto do problema.

III.1 Armazenagem e Manipulação Quântica

A mecânica quântica possibilita armazenar e manipular uma quantidade enorme de informações nos estados de um número relativamente pequeno de partículas. Para entender como isso ocorre, imagine que temos 1.000 partículas e que cada partícula, ao ser medida, pode ser encontrada girando para cima ou para baixo. Para os nossos propósitos, o que significa uma partícula girar para cima ou para baixo é irrelevante; o que importa é que há uma propriedade da partícula que assume um de dois valores quando medida.

Para descrever o estado quântico dessa coleção de partículas, é necessário especificar um número para cada possível resultado da medição das partículas. Esses números são chamados de amplitudes dos resultados possíveis e estão relacionados à probabilidade de cada resultado, mas, diferentemente das probabilidades, as amplitudes quânticas podem ser positivas ou

negativas (na verdade, são números complexos). Por exemplo, é necessária uma amplitude para a possibilidade de todas as 1.000 partículas serem encontradas girando para cima, outra amplitude para a possibilidade de as primeiras 500 partículas estarem girando para cima enquanto as 500 restantes giram para baixo, e assim por diante. Há 2^{1000} possíveis resultados, ou aproximadamente 10^{300}, o que significa que esse é o número de amplitudes necessárias—mais do que o número de átomos no universo visível! O termo técnico para essa situação é que as 1.000 partículas estão em uma superposição desses 10^{300} estados.

Em outras palavras, podemos armazenar 10^{300} números simultaneamente em nossas 1.000 partículas. Então, ao realizar diversas operações nas partículas e em algumas auxiliares—talvez atingindo-as com uma sequência de pulsos de laser ou ondas de rádio—é possível executar um algoritmo que transforma todos os 10^{300} números (cada um uma solução potencial) ao mesmo tempo. Se, ao final, pudéssemos ler com precisão o estado quântico final das partículas, teríamos, de fato, um computador mágico: ele seria capaz de verificar 10^{300} soluções possíveis para um problema, e no final poderíamos rapidamente identificar a correta.

III.2 Colapso da Função de Onda

Infelizmente, há um problema...

Quando as partículas são medidas (o que é necessário para ler o estado final delas), as regras da mecânica quântica determinam que a medição selecionará apenas uma das 10^{300} possibilidades de forma aleatória, e todas as outras desaparecerão.

Pareceria que não estamos em melhor situação do que se usássemos um computador clássico e tentássemos uma solução aleatória—em ambos os casos, acabaríamos conhecendo apenas uma solução possível.

Explicação:

Superposição: Antes da medição, a partícula pode existir simultaneamente em várias posições, velocidades ou outros estados, dependendo das variáveis em questão.

Colapso: Quando a medição é feita, a superposição desaparece, e a partícula "escolhe" um dos estados possíveis, colapsando a função de onda para um único valor. Isso é o que chamamos de colapso da função de onda.

Um exemplo famoso é o experimento da dupla fenda, onde partículas (como elétrons) passam por duas fendas simultaneamente, criando um padrão de interferência no detector. No entanto, se medirmos pela qual fenda a partícula passou, o padrão de interferência desaparece, e a partícula se comporta como se tivesse passado por uma única fenda.

III.3 Os Truques de Peter Shor para Lidar com o Colapso da Função de Onda

Felizmente, ainda temos truques que podemos usar para extrair alguma vantagem das partículas quânticas. As amplitudes podem se cancelar quando amplitudes positivas se combinam com negativas, um fenômeno chamado de interferência destrutiva. Assim, um bom algoritmo de computador quântico garantiria que os caminhos computacionais que levam a uma resposta errada fossem cancelados dessa forma. Também garantiria que os caminhos que levam a uma resposta correta tivessem todas as amplitudes com o mesmo sinal—resultando

em interferência construtiva e, assim, aumentando a probabilidade de encontrá-los quando as partículas forem medidas no final.

Para quais problemas computacionais podemos coreografar esse tipo de interferência, usando menos etapas do que seriam necessárias para resolver o problema de forma clássica?

Em 1994, Peter Shor, no Instituto de Tecnologia de Massachusetts (MIT), encontrou o primeiro exemplo de um algoritmo quântico que poderia acelerar dramaticamente a solução de um problema prático. Em particular, Shor mostrou como um computador quântico poderia fatorar um número com **n** dígitos usando um número de etapas que aumenta apenas como cerca de **n^2**—ou seja, em tempo polinomial. Como mencionado anteriormente, o melhor algoritmo conhecido para computadores clássicos usa um número de etapas que aumenta exponencialmente.

Assim, pelo menos para a fatoração, é realmente possível obter uma aceleração exponencial em relação aos algoritmos clássicos conhecidos ao usar métodos quânticos. **Para criar seu algoritmo, Shor explorou certas propriedades matemáticas de números compostos e seus fatores que são particularmente adequadas para produzir o tipo de interferência construtiva e destrutiva que um computador quântico pode aproveitar ao máximo.**

Para entender o algoritmo de Shor, precisamos considerar algumas propriedades matemáticas e numéricas essenciais:

Ciclicidade e Período:

Uma das chaves do algoritmo de Shor é encontrar o período de uma função gerada por potências de números. Ou seja, dado um número $\boldsymbol{a}$ e um número composto $\boldsymbol{N}$, a ideia é descobrir o menor número $\boldsymbol{r}$ (chamado de "período") tal que:

$$a^r \, mod \, N = 1$$

Este processo é crucial porque o período $\boldsymbol{r}$ pode ser usado para obter os fatores de $\boldsymbol{N}$.

Fatoração via Período:

Uma vez que o período $\boldsymbol{r}$ é encontrado, é possível usar a propriedade de que $\boldsymbol{r}$ ajuda a revelar divisores de $\boldsymbol{N}$. Se o período $\boldsymbol{r}$ for ímpar ou não fornecer um fator direto, o algoritmo tenta valores ajustados até encontrar um fator não trivial de $\boldsymbol{N}$.

Interferência Quântica (Construtiva e Destrutiva)

Agora, o ponto crucial do algoritmo de Shor é como ele usa a interferência quântica para encontrar o período de maneira eficiente. A interferência quântica é uma propriedade única dos sistemas quânticos, onde as amplitudes das probabilidades podem se somar (interferência construtiva) ou se cancelar (interferência destrutiva) dependendo da fase das ondas quânticas envolvidas.

Como a interferência entra no algoritmo de Shor:

Superposição: O algoritmo começa criando uma superposição quântica de vários estados possíveis. Isso é feito usando portas quânticas que manipulam Qubits (a unidade de informação quântica). Cada Qubits pode estar em uma superposição de estados |0⟩ e |1⟩.

Transformações Quânticas: O algoritmo utiliza uma transformação quântica (conhecida como Transformada de Fourier Quântica) para transformar a superposição de estados em um formato que revela o período $\boldsymbol{r}$. A Transformada de Fourier é uma técnica matemática clássica que lida com a decomposição de funções em suas frequências componentes. No contexto quântico, essa técnica permite que o algoritmo extraia o período da função:

$$a^r \, mod \, N = 1,$$ de forma eficiente.

A **Transformada de Fourier Quântica (QFT)** é uma versão quântica da Transformada de Fourier clássica, um algoritmo fundamental em muitas áreas da matemática e da engenharia, especialmente para decompor sinais em suas frequências componentes. A QFT é um dos componentes-chave de muitos algoritmos quânticos, como o algoritmo de Shor, que é usado para fatoração de números inteiros.

Transformada de Fourier Clássica vs. Quântica

Transformada de Fourier Clássica (DFT - Discrete Fourier Transform):

A DFT transforma uma sequência de valores discretos $x_0, x_{1,\ldots}, x_{N-1}$ em uma sequência de coeficientes complexos $X_0, X_{1,\ldots}, X_{N-1}$ que representam as frequências componentes do sinal. A fórmula clássica é dada por:

$$X_k = \sum_{n=0}^{N-1} x_n e \frac{-2\pi i}{N} kn$$

A DFT tem complexidade $\boldsymbol{O(N^2)}$ para ser calculada de forma direta.

Transformada de Fourier Quântica (QFT):

A QFT é uma operação quântica que transforma uma superposição de estados em uma nova superposição, onde as amplitudes dos estados estão relacionadas às frequências de Fourier do sistema. A diferença crucial é que a QFT pode ser realizada de forma eficiente usando um número exponencialmente menor de operações do que a DFT clássica. O tempo de execução da QFT é $\boldsymbol{0((log\, N)^2)}$, o que é muito mais rápido do que o tempo clássico $\boldsymbol{O(N^2)}$.

Interferência:

Quando a superposição de estados é projetada corretamente e passa pela transformação de Fourier, ocorre um fenômeno de interferência:

Interferência construtiva ocorre quando as probabilidades das diferentes soluções se somam, amplificando a solução correta (o valor do período $\boldsymbol{r}$).

Interferência destrutiva ocorre quando as probabilidades das soluções erradas se cancelam, reduzindo a chance de obter resultados incorretos.

Extração do Período:

A interferência quântica permite que o algoritmo encontre o período r de maneira muito mais eficiente do que qualquer algoritmo clássico. O processo de medição da função quântica resultante colapsa a superposição em um valor único, que pode ser usado para derivar divisores não triviais de N, com grande probabilidade de sucesso.

Retomando...

Assim, para fatoração, é realmente possível obter uma aceleração exponencial em relação aos algoritmos clássicos conhecidos ao usar métodos quânticos. Mas, apesar de uma ideia equivocada amplamente difundida, **o problema de fatoração não é conhecido e nem considerado como NP-completo**.

Para criar seu algoritmo, Shor explorou certas propriedades matemáticas de números compostos e seus fatores que são particularmente adequadas para produzir o tipo de interferência construtiva e destrutiva que um computador quântico pode aproveitar ao máximo. Não se sabe se os problemas NP-completos compartilham dessas propriedades especiais.

Até hoje, nós, pesquisadores, encontramos apenas alguns algoritmos quânticos que parecem oferecer uma aceleração de tempo exponencial para polinomial em um problema.

A questão permanece sem resposta:

Existe um algoritmo quântico eficiente para resolver problemas NP-completos?

Apesar de muitos esforços, esse tipo de algoritmo ainda foi encontrado—embora, como era de se esperar, os cientistas da computação não possam provar que ele não existe. Afinal, nem podemos provar que não existe um algoritmo clássico de tempo polinomial para resolver problemas NP-completos.

O que podemos dizer é que um algoritmo quântico capaz de resolver problemas NP-completos de forma eficiente teria que explorar a estrutura dos problemas, assim como o algoritmo de Shor, mas de uma maneira que ainda está além das técnicas atuais.

III.4 Algoritmos Quânticos e as Caixas Pretas

Não é possível obter uma aceleração exponencial tratando os problemas como "caixas pretas" sem estrutura, consistindo em um número exponencial de soluções a serem testadas em paralelo. Ainda assim, algum ganho pode ser extraído dessa abordagem de "caixa preta", e os cientistas da computação determinaram quão eficaz—e quão limitada—essa aceleração pode ser. O algoritmo que produz essa aceleração é o segundo principal algoritmo quântico.

A abordagem de "caixa preta" pode ser ilustrada ao imaginar que você está procurando a solução para um problema difícil e que a única operação que sabe realizar é adivinhar uma solução e verificar se ela funciona. Suponha que haja S soluções possíveis, onde S cresce exponencialmente à medida que o tamanho do problema n aumenta. Você pode ter sorte e

adivinhar a solução na primeira tentativa, mas, no pior caso, você precisará de S tentativas, e, em média, precisará de S/2.

Agora, suponha que você possa perguntar sobre todas as soluções possíveis em superposição quântica...

Em 1996, **Lov Grover, dos Bell Laboratories**, desenvolveu um algoritmo para encontrar a solução correta em tal cenário usando apenas cerca de $\sqrt{S}$ etapas. **Um ganho de desempenho de S/2 para $\sqrt{S}$ é um avanço útil para alguns problemas**—se houver um milhão de soluções possíveis, você precisará de cerca de 1.000 etapas em vez de 500.000.

Mas a raiz quadrada não transforma um tempo exponencial em tempo polinomial; ela apenas reduz a magnitude do exponencial. E o algoritmo de Grover é o melhor que se pode obter para esse tipo de busca em "caixa preta": pesquisadores demonstraram que qualquer algoritmo quântico de "caixa preta" precisa de pelo menos $\sqrt{S}$ etapas.

Nas últimas décadas pesquisadores demonstraram ganhos semelhantes (significativos, porém limitados) são a característica para muitos outros problemas além de buscar em uma lista, como contar votos em uma eleição, encontrar a rota mais curta em um mapa e jogar jogos de estratégia como xadrez ou Go.

Um problema que apresentou dificuldade particular foi o chamado problema de colisão, que consiste em encontrar dois itens idênticos, ou que "colidem", em uma longa lista. Se existisse um algoritmo quântico rápido para resolver esse problema, muitos dos blocos básicos da segurança no comércio eletrônico seriam inúteis em um mundo com computadores quânticos.

Buscar um item em uma lista é como procurar uma agulha em um palheiro, enquanto procurar uma colisão é como buscar duas peças de feno idênticas, o que dá ao problema um tipo de estrutura que um computador quântico poderia potencialmente explorar. No entanto, em 2002 foi demonstrado que, dentro do modelo de "caixa preta", qualquer algoritmo quântico precisa de tempo exponencial para resolver o problema de colisão.

Admitidamente, essas limitações de "caixa preta" não descartam a possibilidade de que algoritmos quânticos eficientes para problemas NP-completos ou até mesmo mais difíceis estejam esperando para serem descobertos. Se tais algoritmos existissem, no entanto, eles teriam que explorar a estrutura dos problemas de maneiras que são completamente diferentes de tudo o que já vimos, da mesma forma que algoritmos clássicos eficientes para os mesmos problemas teriam que fazer.

A "mágica quântica" por si só não fará o trabalho.

Com base nessa percepção, muitos cientistas da computação agora conjecturam não apenas que P ≠ NP, mas também que computadores quânticos não podem resolver problemas NP-completos em tempo polinomial.

Tudo o que sabemos é consistente com a possibilidade de que os computadores quânticos sejam o fim da linha—ou seja, que eles sejam o tipo mais geral de computador compatível com as leis da física. No entanto, os físicos ainda não têm uma teoria final da física, então não se

pode descartar a possibilidade de que, algum dia, uma teoria futura revele um meio físico para resolver problemas NP-completos de maneira eficiente.

Uma das características centrais da mecânica quântica é uma propriedade matemática chamada linearidade...

Em 1998, Daniel S. Abrams e Seth Lloyd, ambos na época no MIT, mostraram que, se um pequeno termo não linear fosse adicionado às equações da mecânica quântica, os computadores quânticos seriam capazes de resolver problemas NP-completos de maneira eficiente. Antes que fiquemos muito animados, devemos perceber que, se tal termo não linear existisse, também seria possível violar o princípio da incerteza de Heisenberg e enviar sinais mais rápido que a velocidade da luz. Como apontaram Abrams e Lloyd, talvez a melhor interpretação desses resultados seja que eles ajudam a explicar por que a mecânica quântica é linear.

Outro tipo especulativo de máquina alcançaria habilidades computacionais extravagantes ao comprimir um número infinito de passos em um tempo finito. Infelizmente, de acordo com o entendimento atual dos físicos, o tempo parece se degenerar em um mar de flutuações quânticas—algo como uma espuma em vez de uma linha suave e uniforme—na escala de 10^{-43} segundos (o tempo de Planck), o que parece tornar esse tipo de máquina impossível.

Se o tempo não pode ser dividido em finas fatias arbitrárias, então talvez outra maneira de resolver problemas NP-completos de forma eficiente seria explorar a viagem no tempo. Os físicos que estudam o assunto falam não sobre máquinas do tempo, mas sobre curvas temporais fechadas (CTCs, do inglês Closed Timelike Curves).

Essencialmente, uma CTC é uma rota pelo espaço e pelo tempo que matéria ou energia poderiam percorrer para encontrar-se consigo mesmas no passado, formando um ciclo fechado. A teoria física atual é inconclusiva sobre se CTCs podem existir, mas isso não nos impede de perguntar quais seriam as consequências para a ciência da computação se elas realmente existissem.

Parece óbvio como alguém poderia usar uma CTC para acelerar um cálculo: programar o computador para levar o tempo que for necessário para resolver o problema e, em seguida, enviar a resposta de volta no tempo para si mesmo, em um ponto antes de o computador começar. No entanto, essa ideia simples não funciona, porque ignora o famoso paradoxo do avô, em que você volta no tempo para matar seu próprio avô (mas então você nunca nasce, então nunca volta no tempo, e assim seu avô vive para ter filhos, e mais tarde você nasce, mas então...). Em nosso caso, o que aconteceria se você desligasse o computador depois de receber sua resposta do futuro?

Em 1991, o físico David Deutsch, da Universidade de Oxford, definiu um modelo de computação com CTCs que evita essa dificuldade. No modelo de Deutsch, a natureza garante que, à medida que os eventos se desenrolam ao longo da linha do tempo circular que compõe a CTC, nenhum paradoxo jamais surge. Esse fato pode ser explorado para programar um computador que opere em loop dentro da CTC para resolver problemas difíceis.

De fato, ao usar uma CTC (curva temporal fechada), poderíamos resolver de forma eficiente não apenas problemas NP, mas até mesmo problemas de uma classe aparentemente maior

chamada PSPACE. PSPACE é a classe de problemas que podem ser resolvidos em um computador convencional usando uma quantidade polinomial de memória, mas que possivelmente exigem um tempo exponencial. Na prática, uma CTC tornaria o tempo e o espaço intercambiáveis como recursos computacionais.