Problemas NP-Completos

Vinicius Fernandes Cabral[1]

[1]Instituto Federal Goiano - Campus - Rio Verde
(IF Goiano)

vinicius.fernandesrv@gmail.com

1. A classe P

Na teoria da complexidade computacional, P é o acrônimo em inglês para Tempo polinomial determinístico (Deterministic Polynomial time) que denota o conjunto de problemas que podem ser resolvidos em tempo polinomial por uma máquina de Turing determinística. Qualquer problema deste conjunto pode ser resolvido por um algoritmo com tempo de execução O(n sobre k), (com k constante).

Podemos ter a classe P como a classe de problemas que são solúveis para um computador real. Embora esta definição não seja exata. P é conhecido por conter vários problemas naturais, incluindo as versões de decisão de Programação linear, o cálculo do Máximo divisor comum e encontrar um emparelhamento máximo. Em 2002, foi mostrado que o problema em se determinar se um número é primo está em P.

Algoritmos em tempo polinomial são fechados sob composição. Intuitivamente, se alguém escreve uma função de tempo polinomial assumindo que as chamadas de função são em tempo constante e se as funções chamadas requerem tempo polinomial, então o algoritmo inteiro leva tempo polinomial. Isto é uma das razões principais de P ser considerada uma classe independente de máquina; qualquer recurso da máquina, como Acesso aleatório, que pode ser simulada em tempo polinomial, pode simplesmente ser composto com o algoritmo de tempo polinomial principal para reduzí-la em um algoritmo de tempo polinomial que pode ser rodada em uma máquina mais básica.

Toda a discussão acima tem dito que P significa ”fácil” e ”não em P” significa ”difícil”, uma hipótese conhecida como tese de Cobham. É uma hipótese comum e razoavelmente exata em teoria da complexidade, porém tem algumas ressalvas.

Primeiro, nem sempre é verdade na prática. Um algoritmo polinomial teórico pode ter muitos fatores constantes ou expoentes, tornando-o, assim, impraticável. Por outro lado, até mesmo se um problema se mostrar NP-completo, e mesmo se P diferente de NP, ainda pode haver abordagens eficazes para combater o problema na prática. Existem algoritmos para muitos problemas NP-completos, como o Problema da Mochila, o Problema do Caixeiro Viajante e o Problema de Satisfatibilidade Booleana, que podem resolver com otimização muitas instâncias do mundo real, em tempo razoável. A complexidade empírica da média de casos (tempo versus tamanho do problema) de tais algoritmos pode ser surpreendentemente baixa. Um exemplo famoso é o algoritmo simplex na programação linear, que funciona surpreendentemente bem na prática, apesar de ter pior caso de complexidade de tempo exponencial que anda junto com os mais conhecidos algoritmos de tempo polinomial.

Segundo, existem tipos de cálculos que não são compatíveis ao modelo de

máquina de Turing em que P e NP são definidos, assim como computação quântica e algoritmos randomizados.

2. A classe NP

NP é o acrônimo em inglês para Tempo polinomial não determinístico (Non-Deterministic Polynomial time) que denota o conjunto de problemas que são decidíveis em tempo polinomial por uma máquina de Turing não-determinística. Uma definição equivalente é o conjunto de problemas que podem ser verificados em tempo polinomial por uma máquina de Turing determinística.

Devido ao fato de existirem muitos problemas importantes nesta classe, existe um esforço intensivo para encontrar algoritmos para resolver problema NP em um tempo que seja polinomial em relação ao tamanho da entrada. Contudo, existe um grande número de problemas NP que resiste a tais tentativas, parecendo requerer um tempo super-polinomial. Se estes problemas realmente não podem ser resolvidos em tempo polinomial é um das grandes questões abertas na ciência da computação

Uma importante noção neste contexto é o conjunto de problemas de decisão NP-Completo, o qual é um subconjunto de NP e pode ser informalmente descrito como os mais difíceis problemas em NP. Se existe um algoritmo em tempo polinomial para um deles, então haverá um algoritmo em tempo polinomial para todos os problemas em NP. Devido a isto, e a falha das pesquisas em encontrar um algoritmo polinomial para qualquer problema NP-completo, uma vez que foi provado que um problema pode ser caracterizado como NP-completo este é um sinal largamente aceito que um algoritmo polinomial para este problema não deve existir.

Há também uma caracterização lógica mais simples do problema NP; ela está expressa precisamente em uma linguagem baseada em lógica de segunda-ordem restrita pela exclusão da qualificação universal alem de relações, funções, e subconjuntos.

NP podem ser vistos com um tipo muito simples de sistema de prova interativa, onde a prova vem junto com o certificado de prova e a verificação é uma maquina determinística em tempo polinomial que a checa. Ela é completa porque a correta expressão de prova será obtida para ela considerando que ela exista, e isto é legitimo porque a certificação não pode ser aceita se não existir uma aceitável expressão de prova.

Um grande resultado da teoria da complexidade é que problemas NP podem ser caracterizados como os problemas solucionáveis por provas de checagem probabilística onde os verificadores usam O(logn) bit's aleatórios e examinar somente um número constantes de bits da sentença de prova (a classe PCP(log n, 1)). Mais informalmente, isto significa que a verificação NP descrita acima pode ser substituída por uma checagem de poucos lugares na sentença de prova, e usando um número limitado de lançamento de moedas podemos determinar a resposta correta com grande probabilidade. Isso permite vários resultados a cerca da dificuldade de algoritmos aproximados serem provados.

3. A classe NP-difícil

NP-difícil (ou NP-hard, ou NP-complexo) na teoria da complexidade computacional, é uma classe de problemas que são, informalmente, "Pelo menos tão difíceis quanto os problemas mais difíceis em NP". Um problema H é NP-difícil se e somente se existe um problema NP-completo L que é Turing-redutível em tempo polinomial.

Em outras palavras, L pode ser resolvido em tempo polinomial por uma Máquina de Turing não determinística com um oráculo para H. Informalmente, podemos pensar em um algoritmo que pode chamar tal Máquina de Turing Não-Determinística como uma sub-rotina para resolver H, e resolver L em tempo polinomial, se a chamada da sub-rotina leva apenas um passo para computar. Problemas NP-difíceis podem ser de qualquer tipo: problemas de decisão, problemas de pesquisa ou problemas de otimização.

Um erro comum é pensar que NP em NP-difícil representa não-polinomial. Embora seja amplamente suspeito de que não existem algoritmos de tempo polinomial para problemas NP- difíceis, isto nunca foi provado. Além disso, a classe NP contém também todos os problemas que podem ser resolvidos em tempo polinomial.

4. A questão P = NP?

É fácil entender que a classe NP inclui a classe P, ou seja, que todo problema polinomial é polinomialmente verificável. O bom senso sugere que P é apenas uma pequena parte de NP. Surpreendentemente, ninguém encontrou ainda um problema de NP que não esteja em P, isto é, um problema polinomialmente verificável para o qual não existe algoritmo polinomial.

Esta situação abre caminho para a suspeita de que talvez P seja igual a NP . A maioria dos especialistas não acredita nessa possibilidade.

A relação entre as classes de complexidade P e NP é estudada na teoria da complexidade computacional, parte da teoria da computação que lida com os recursos necessários computacionais para se resolver um determinado problema. Os recursos mais comuns são o tempo (quantos passos é preciso para se resolver um problema) e espaço (a quantidade de memória necessária para se resolver um problema). Em tal análise, um modelo de computador cujo tempo deve ser analisado é necessário. Geralmente, esses modelos assumem que o computador é determinística (dado o estado atual do computador e todas as entradas, há apenas uma ação possível que o computador pode realizar) e sequencial (executa ações uma após a outra).

Nesta teoria, a classe P consiste em todos os problemas de decisão (definido abaixo) que podem ser resolvidos por uma máquina determinística sequencial em uma quantidade de tempo que é polinomial para o tamanho da entrada; a classe NP consiste em todos os problemas de decisão cujas soluções positivas podem ser verificadas em tempo polinomial dada a informação correta, ou de forma análoga, cujas soluções podem ser encontradas em tempo polinomial em uma máquina não determinística. Claramente, P nao esta contido em NP. Sem dúvida a maior questão sem respostas na ciência da computação teórica se diz respeito à relação entre essas duas classes:

P é igual a NP?

Em uma pesquisa realizada em 2002 por 100 pesquisadores, 61 acreditavam que a resposta fosse não, 9 acreditavam que a resposta era sim e 22 não tinham certeza; 8 acreditavam que a questão pode ser independente dos axiomas aceitos atualmente e, portando impossível de se provar ou refutar.

De acordo com uma pesquisa qualquer, muitos cientistas da computação acreditam que P diferente NP. A principal razão para essa crença é que, após décadas estudando

estes problemas, ninguém foi capaz de encontrar um algoritmo de tempo polinomial para qualquer um dos mais de 3000 problemas NP-completos conhecidos. Esses algoritmos foram procurados muito antes do conceito de NP-completude ter sido definido (21 problemas NP-completos de Karp, entre os primeiros encontrados, eram todos problemas bem conhecidos existentes no momento em que foram mostrados para ser NP-completo). Além disso, o resultado P = NP implicaria em muitos outros resultados surpreendentes que estão acredita-se serem falsos atualmente, como NP = co-NP e P = PH.

Também é intuitivamente argumentado que a existência de problemas que são difíceis de resolver, mas para os quais as soluções são fáceis de verificar, combinam com a experiência do mundo real.

Se P = NP, então o mundo seria um lugar profundamente diferente do que supomos que ele seja. Não haveria nenhum valor especial em ”saltos criativos,” nenhuma lacuna fundamental entre resolver um problema e reconhecer a solução, uma vez que ele é encontrado. Todos que puderam apreciar uma sinfonia seria Mozart; todos que poderia seguir um argumento passo a passo seria Gauss ... - Scott Aaronson, MIT

Por outro lado, alguns pesquisadores acreditam que há excesso de confiança em acreditar que P diferente NP e que os investigadores devem explorar provas de que P = NP também. Por exemplo, essas declarações foram feitas em 2002:

O principal argumento em favor da P diferente NP é a total carência de progresso fundamental na área de busca exaustiva. Isto é, na minha opinião, um argumento muito fraco. O espaço de algoritmos é muito grande e nós estamos apenas no início de sua exploração. A resolução do Último Teorema de Fermat mostra também que as questões muito simples podem ser resolvidos apenas por teorias muito profundas. -Moshe Y. Vardi, Rice University

Estar ligado a uma especulação não é um bom guia para o planejamento de pesquisa. Um deve sempre tentar ambas as direções de todos os problemas. O preconceito tem levado matemáticos famosos a falhar ao resolver problemas conhecidos cuja solução era oposta às suas expectativas, apesar de terem sido desenvolvidos todos os métodos necessários. -Anil Nerode, Cornell University

5. Problemas NP-completos

Na teoria da complexidade computacional, a classe de complexidade é o subconjunto dos problemas NP de tal modo que todo problema em NP se pode reduzir, com uma redução de tempo polinomial, a um dos problemas NP-completo. Pode-se dizer que os problemas de NP-completo são os problemas mais difíceis de NP e muito provavelmente não formem parte da classe de complexidade P. A razão é que se conseguisse encontrar uma maneira de resolver qualquer problema NP-completo rapidamente (em tempo polinomial), então poderiam ser utilizados algoritmos para resolver todos problemas NP rapidamente.

NP-completo é um subconjunto de NP, o conjunto de todos os problemas de decisão os quais suas soluções podem ser verificadas em tempo polinomial; NP pode ser equivalentemente definida como o conjunto de problemas de decisão que podem ser solucionados em tempo polinomial em uma Máquina de Turing não determinística. Um problema p em NP também está em NPC Se e somente se todos os outros problemas em NP podem ser transformados em p em tempo polinomial.

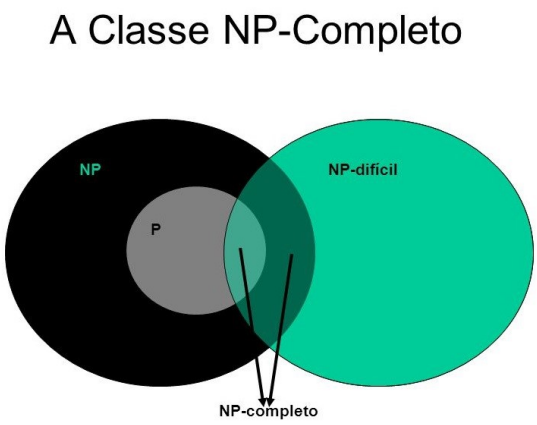

Figure 1. Diagrama para o conjunto de problemas

Problemas NP-completo são estudados porque a habilidade de rapidamente verificar soluções para um problema (NP) parece correlacionar-se com a capacidade de resolver rapidamente esse problema (P). Não é sabido se todos os problemas em NP podem ser rapidamente resolvidos - isso é chamado de problema P versus NP. Mas se qualquer problema em NP- completo pode ser resolvido rapidamente, então todo problema em NP também pode ser, por causa da definição de NP-completo afirma que todo problema em NP deve ser rapidamente redutível para todo problema em NP-completo (ou seja, pode ser reduzido em tempo polinomial). Por causa disso, é geralmente falado que os problemas NP-completo são mais difíceis que os problemas NP em geral.

5.1. Problema do Clique

O problema do clique refere-se a qualquer problema que possui como objetivo encontrar subgrafos completos (”cliques”) em um grafo. Como exemplo, o problema de encontrar conjuntos de nós em que todos os elementos estão conectados entre si.

Um clique em um grafo não-orientado é um subconjunto de seus vértices tais que cada dois vértices do subconjunto são conectados por uma aresta. Clique é um dos conceitos mais básicos na teoria dos grafos e são utilizados em vários problemas matemáticos e construções em grafos. O Clique vem sendo estudado na ciência da computação: a tarefa de achar se existe um clique de um dado tamanho em um grafo (o problema do clique) é NP- completo, mas apesar de sua dificuldade, vários algoritmos para encontrar clique foram estudados.

Em um grafo não direcionado G = (V, E) é um subconjunto de vértices C não esta contido em V, tal que para cada dois vértices em C, existe uma aresta os conectando. Isso se equivale a dizer que um subgrafo induzido de C é completo (em alguns casos, o termo clique também pode ser referência ao subgrafo).

Por exemplo, o problema clique surge no cenário seguinte. Considere uma rede social, onde os vértices do grafo representam pessoas, e as arestas representam o conhecimento mútuo. Para encontrar um maior subconjunto de pessoas, em que todas conhecem umas as outras, pode-se sistematicamente inspecionar todos os subconjuntos, um processo que é muito demorado para ser prático para as redes sociais, mesmo que pequenas.

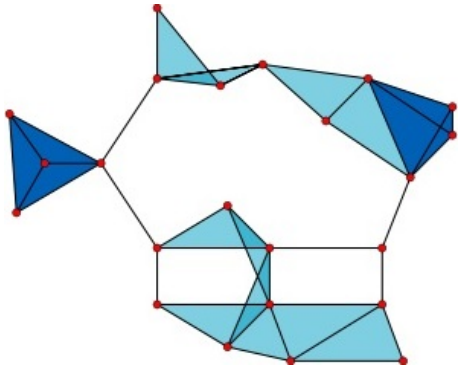

Figure 2. Grafo com diferentes cliques

Embora a pesquisa por força bruta possa ser melhorada através de algoritmos mais eficientes, todos estes algoritmos levam tempo exponencial para resolver o problema. Portanto, grande parte da teoria sobre o problema do clique é dedicado à identificação de tipos especiais de gráfo que admitem algoritmos mais eficientes, ou a definição da dificuldade computacional do problema geral em vários modelos de computação. Junto com seus aplicativos em redes sociais, o clique também tem muitas aplicações em bioinformática e química computacional.

Problemas que envolvem o clique:

Encontrar o clique máximo (um clique com o maior número de vértices); encontrar o clique com maior valor em um grafo valorado; listar todos os cliques máximos (cliques que não podem ser ampliados); resolver o problema de decisão de testar se um grafo contém um clique maior que um tamanho determinado.

Esses problemas são todos difíceis: o problema de decisão clique é NP-completo (um dos 21 problemas NP-Completo de Karp), e listar todos os cliques máximos pode exigir tempo exponencial. No entanto, existem algoritmos para esses problemas que são executados em tempo exponencial ou que lidam com grafos de entrada mais especializados em tempo polinomial.

5.2. Problema do caixeiro-viajante

O Problema do Caixeiro Viajante (PCV) é um problema que tenta determinar a menor rota para percorrer uma série de cidades (visitando uma única vez cada uma delas), retornando à cidade de origem. Ele é um problema de otimização NP-difícil inspirado na necessidade dos vendedores em realizar entregas em diversos locais (as cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.

Nos anos de 1800, problemas relacionados com o PCV começaram a ser desenvolvidos por dois matemáticos: o escocês William Rowan Hamilton e o britânico Thomas Penyngton Kerkman. A forma geral do PCV parece ter sido, pela primeira vez, estudada por matemáticos nos anos de 1930 em Harvard e Viena. O problema foi posteriormente estudado por Hassler Whitney e Merril Flood em Princeton. Exceptuando pequenas variações ortográficas, como traveling vs travelling ou salesman vs salesman's, o nome do problema ficou globalmente conhecido por volta do ano 1950.

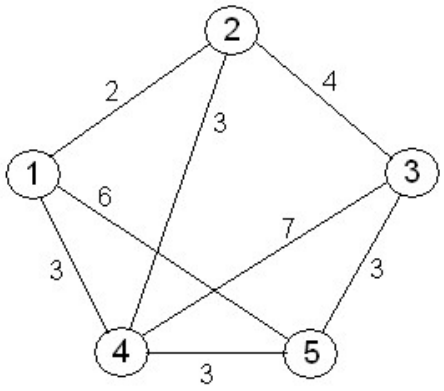

Figure 3. Exemplo de rotas

O problema do caixeiro-viajante representado na figura consiste na procura de um circuito que possua a menor distância, começando numa cidade qualquer, entre várias, visitando cada cidade precisamente uma vez e regressando à cidade inicial.

O tamanho do espaço de procura aumenta exponencialmente dependendo de n, o número de cidades, uma vez que existem. Embora tenham sido desenvolvidos bons algoritmos de aproximação para o PCV, o problema continua a oferecer uma grande atracção para a aplicação de novos algoritmos, tais como os evolucionários. Isto deve-se, essencialmente, às seguintes razões:

A problemática do PCV pode ser entendida facilmente, uma vez que se aproxima dos problemas populares do mundo real; O PCV demonstra o caso mais simples dos problemas de requisição que são de enorme relevância para a programação de processos industriais;

Existem vários conjuntos de dados sobre o PCV que estão disponíveis em literatura, de tal forma que os resultados são comparáveis mesmo que o óptimo global não seja ainda definitivamente conhecido; Relativamente à complexidade computacional, o PCV, como um problema NP-completo, é conhecido por representar uma larga classe de problemas para os quais não existem algoritmos polinomiais em séries temporais determinísticos.

A solução do PCV pode ser determinada por diferentes métodos. Estes, podem ser agrupados em métodos exactos e heurísticos. Os primeiros têm por base procedimentos branch-and-bound (em inglês) , isto é, de enumeração implícita em árvore onde é necessário inserir um limite inferior, no leque de soluções do problema. Existem limites inferiores triviais, como por exemplo, o elemento mínimo das soluções encontradas. Contudo, estes tipos de métodos demonstram muita dificuldade quando aplicados a problemas muito complexos, isto é, um PCV com muitas cidades, uma vez que a árvore de enumeração é muito extensa.

Métodos de melhoria de circuitos procuram melhorar o circuito obtido através de algum outro método. Para tal, o método mais utilizado, elaborado por Lin e Kernighan em 1973, denomina-se por k-opt. Nesta proposta, k arcos são substituídos, no circuito, por outros k arcos com o objectivo de diminuir a distância total percorrida. Quanto maior for o valor de k, melhor é a precisão do método, mas maior é o esforço computacional.

Com a variação do valor k, no método de melhoria k-opt, a heurística de Lin e Kernighan aumentaria a sua eficiência quando comparada com o mesmo método utilizando um valor de k fixo. Desta forma, o método não fica mais simples, contudo passa a ser possível a aplicação do método a problemas mais abrangentes com resultados favoráveis.

Para além deste método, existem outros métodos de melhorias baseados em metaheurísticas do tipo simulated annealing e busca tabu. Estes, para além de se basearem no desenvolvimento k-opt, procuram uma solução que não a dada pelos métodos anteriores. No simulated annealing é utilizado um controlo de possibilidades de solução melhores partindo de piores, no início. Na busca tabu os movimentos considerados tabu, isto é, que não se podem efectuar, mesmo que melhorem a solução são temporariamente interditos com o objectivo de se alcançar soluções piores no início que no final poderão ser consideradas melhores (Laporte, 2002) e (Heslgaun, 2000), citados por (Cunha, 2002).

5.3. Problema da soma dos subconjuntos

Em ciência da computação, o problema da soma de subconjuntos é um importante problema da teoria da complexidade computacional e criptografia. O problema é este: dado um conjunto de inteiros, existe um subconjunto não-vazio cuja soma é 0? Por exemplo, dado o conjunto (–7, –3, –2, 5, 8), a resposta é sim porque o subconjunto (–3, –2, 5) resulta numa soma que dá zero. O problema é NP-completo.

Um problema equivalente é este: dado um conjunto de inteiros e um inteiro s, existe algum subconjunto que some s? Problema da soma dos subconjuntos também pode ser pensado como um caso especial do problema da mochila. Um caso interessante de soma subconjunto é o problema de partição, em que s é a metade da soma de todos os elementos do conjunto.

A complexidade (dificuldade de resolver) da soma de subconjuntos poderá ser vista de dois parâmetros, N, o número de variáveis de decisão, e P, a precisão do problema (indicado como o número de bits necessários para indicar/resolver o problema).

A complexidade do melhor algoritmo conhecido é exponencial no menor dos dois parâmetros N e P. Assim, o problema é mais difícil se N e P são da mesma ordem. E torna-se fácil se nem N ou P são pequenas.

Se N (número de variáveis) é pequeno, então a busca exaustiva pela solução é possível. Se P é um número pequeno fixo, então existem algoritmos de programação dinâmica que podem resolver.

O que acaba acontecendo é que o problema se torna aparentemente não exponencial quando se é prático para considerar o espaço da solução. Existem duas maneiras de medir o espaço de solução do Problema da soma dos subconjuntos. Uma delas é contar o número de maneiras que as variáveis podem ser combinadas. Existem 2N possíveis maneiras de combinar as variáveis. Contudo, com N = 10, existem somente 1024 possíveis combinações para checar. Estes podem ser contadas facilmente com uma branching search. A outra maneira é considerar a combinação de todos os valores numéricos possíveis. Estes poderão ser contados facilmente com um algoritmo de programação dinâmica. Quando N = P e ambos são grandes, então não há nenhum aspecto do espaço de soluções que podem ser contados facilmente.

6. Consequências da resolução do problema

Uma das razões de o problema atrair tanta atenção são as conseqüências da resposta. Qualquer direção de resolução iria avançar a teoria enormemente, e talvez tenha enormes conseqüências práticas também.

P = NP

A prova de que P = NP poderia ter conseqüências práticas magníficas, se a prova levasse a métodos eficientes para resolver alguns dos importantes problemas na NP. Também é possível que uma prova não levaria diretamente para métodos eficientes, talvez se a prova não for construtiva, ou se o tamanho do polinômio delimitador for grande demais para ser eficiente na prática. As conseqüências, tanto positivas como negativas, originam-se desde que vários problemas NP-completos são fundamentais em muitos campos. Criptografia, por exemplo, depende de certos problemas sendo difíceis. Uma solução construtiva e eficiente para um problema NP-completo, tais como 3-SAT, iria quebrar a maioria dos sistemas de criptação existentes, incluindo criptografia de chave pública, uma base para muitas aplicações de segurança modernos, como transações econômicas seguras através da Internet, e codificações simétricas, como AES ou 3DES, usadas para a criptografia de dados de comunicações. Estes teriam de ser modificados ou substituídos por soluções de informação teoricamente seguras.

Por outro lado, há enormes conseqüências positivas que poderiam acompanhar a representação tratável muitos problemas atualmente matematicamente intratáveis. Por exemplo, muitos problemas em pesquisas de operações são NP-completo, tais como alguns tipos de programação inteira, é o problema do CAIXEIRO VIAJANTE, para citar dois dos exemplos mais famosos. Soluções eficazes para esses problemas teriam enormes implicações para logísticas. Muitos outros problemas importantes, como alguns problemas na previsão da estrutura de proteínas, também são NP-completos; se estes problemas fossem eficiente resolvíveis, poderiam estimular avanços consideráveis na biologia.

Mas tais mudanças podem enfraquecer significativamente se comparadas à revolução de um método eficiente para a resolução de problemas NP-completos poderia causar na própria matemática. De acordo com Stephen Cook, ... seria transformar a matemática, deixando que um computador encontre uma prova formal de qualquer teorema que tem uma prova de um tamanho razoável, uma vez que as provas formais podem ser facilmente reconhecidas em tempo polinomial. Problemas-exemplo podem muito bem incluir todos os problemas-prêmio CMI.

Pesquisadores matemáticos gastam suas carreiras tentando provar teoremas, e algumas provas têm levado décadas ou mesmo séculos para serem decifradas - por exemplo, o Último Teorema de Fermat levou mais de três séculos para ser provado. Um método que fosse garantido para encontrar provas de teoremas deveria existir de um tamanho ”razoável”. Iria essencialmente acabar com essa luta.

P diferemte de NP

A prova que mostrou que P diferente NP faltaria os benefícios práticos computacionais de um prova de que P = NP, mas que, no entanto, representam um avanço muito significativo na teoria da complexidade computacional e fornecer orientações para pesquisas futuras. Permitiria mostrar de uma maneira formal de que muitos proble-

mas comuns não podem ser resolvidos de forma eficiente, de modo que a atenção de pesquisadores pode ser focado em soluções parciais ou soluções para outros problemas. Devido à crença generalizada em P diferente NP, grande parte desse foco de pesquisa já foi realizada.

P diferente NP também ainda deixa aberta a complexidade média-caso de problemas difíceis em NP. Por exemplo, é possível que SAT requeira tempo exponencial no pior caso, mas que quase todas as instâncias selecionadas aleatoriamente de que são resolução eficiente. Russell Impagliazzo descreveu cinco hipotético "mundos" que poderia resultar das diversas soluções possíveis para a questão caso a complexidade da média. Estes vão desde "Algorithmica", onde P = NP e problemas como SAT pode ser resolvido de forma eficiente em todas as instâncias, para "Cryptomania", onde P diferente NP e gerando casos de problemas de difícil fora P é fácil, com três possibilidades intermediárias refletindo diferentes distribuições possíveis de dificuldade mais casos de problemas NP difíceis.

7. Referências

CUNHA, Claudio Barbieri da; BONASSER, Ulisses de Oliveira; ABRAHÃO, Fernando Teixeira Mendes – Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante [Em Linha]. São Paulo: ANPET, 2002. [Consult. 13 Abr. 2009]. Disponível em WWW: URL:http://www.ptr.poli.usp.br/ptr/docentes/cbcunha/files/2 opt_ TSP_ Anpet_ 2002_ CBC.pdf

Abello, J.; Pardalos, P. M.; Resende, M. G. C. (1999), On maximum clique problems in very large graphs, in: Abello, J.; Vitter, J., External Memory Algorithms, ISBN 0821811843, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 50, American Mathematical Society, pp. 119–130.

Goldreich, Oded (2010). P, Np, and Np-Completeness. Cambridge: Cambridge University Press. ISBN 9780521122542.

Bomze, I. M.; Budinich, M.; Pardalos, P. M.; Pelillo, M. (1999), The maximum clique problem, Handbook of Combinatorial Optimization, 4, Kluwer Academic Publishers, pp. 1–74.