ALGORITMOS APROXIMATIVOS: UMA ALTERNATIVA PARA PROBLEMAS NP - COMPLETOS

Laira Vieira Toscani - Jayme Luiz Szwarcfiter
Universidade Federal do Rio Grande do Sul
Universidade Federal do Rio de Janeiro
Brasil

1. INTRODUÇÃO

A análise da complexidade de um algoritmo é segundo Aho e Ullman /AHO 74/ o coração da Ciência da Computação.

Várias medidas de complexidade de um algoritmo surgiram historicamente, mas sem dúvida uma das mais importantes é a medida de tempo /COO 83/. Esse trabalho tratará da complexidade de tempo de algoritmo, usando o termo complexidade, simplesmente.

A complexidade de um algoritmo em muitos casos não depende só do tamanho da entrada, mas de certas propriedades da entrada, como por exemplo no problema de ordenar uma lista de números. Ordenar uma lista em que quase todos seus elementos estão ordenados, não requer o mesmo esforço necessário para ordenar uma lista de mesmo tamanho, mas com os elementos em grande desordem. Neste caso convém estudar a complexidade do caso médio, que é a média ponderada da complexidade de cada entrada possível e a probabilidade de sua ocorrência. A complexidade do caso médio pode ser mais significativa que a complexidade no pior caso, como no estudo de algoritmos de procura heurística, em Inteligência Artificial. Para esses algoritmos o pior caso tende a ser pessimista, todo algoritmo tem um caso em que funciona de maneira muito enificiente. Além disso é difícil definir limites precisos do pior caso, sendo uma análise probabilística mais natural /HUY 80/. Seguidamente, entretanto, encontrar a distribuição probabilística, para as instâncias, que melhor se adapta ao caso, não é fácil, pois a distribuição pode mudar de maneira imprevisível com o tempo, dificultando a análise da complexidade média. Além disso essa análise não nos diz coisa alguma sobre o comportamento de um algoritmo para uma instância particular. Neste trabalho será analisado somente a complexidade no pior caso, que por simplicidade será chamada só complexidade.

A identificação de P(conjunto dos problemas resolvíveis deterministicamente por algoritmos de complexidade polinomial) como a classe dos problemas tratáveis tem sido, em geral aceita /COO 83/, apesar de um algoritmo $O(n^{1000})$ ser um algoritmo bem ruim, um problema que não possui algoritmo polinomial certamente é intratável.

Outras classes importantes de problema são: NP, conjunto dos problemas resolvíveis não deterministicamente por algoritmo de complexidade polinomial e NP-completo, conjunto de problemas NP com a propriedade adicional de que a existência de algoritmo polinomial e determinístico, que o resolva, implica em P = NP ($P \subseteq NP$ trivialmente).

O fato de existirem centenas de problemas na classe NP-completa fortifica a conjectura de que P ≠ NP.

A pertinência de um problema à classe NP - completa é segundo Cook /COO 83/, para efeito prático o limite inferior de complexidade do problema, o qual pode ser interpretado como intratável.

Existem muitos problemas NP - completos na área de Otimização, Teoria dos Grafos e de Pesquisa Operacional. Os algoritmos conhecidos que os resolvem são portanto de complexidade não polinomial, o que torna-os impraticáveis para instâncias grandes. Felizmente muitas aplicações que requerem soluções para esses problemas não exigem uma solução exata. Baseados nesse fato osprojetistas de algoritmos desenvolveram novos métodos de solução de problemas, que encontram as soluções aproximadas, esses métodos são chamados algoritmos aproximativos ou heurísticos.

As duas principais classes de algoritmos aproximativos são: uma que garante sempre uma solução próxima da solução procurada e outra que produz uma solução ótima ou quase ótima, quase sempre /WEI 77/. Essa segunda é a classe dos algoritmos probabilísticos de grande utilidade em Criptoanálise. Este trabalho tratará somente da primeira classe de algoritmos, que serão chamados pelo nome genérico de algoritmos aproximativos.

2. ALGUMAS DEFINIÇÕES

Todas definições apresentados nessa secção são baseados nos trabalhos de /GAR 79/ e /HOR 78/.

Um problema de otimização combinatorial pode ser um problema de minimização ou maximização e é caracterizado por uma terna constituída por um conjunto de instâncias, uma função candidata e uma função valor de solução.

Assim, se π é um problema de otimização, então $\pi = (D_\pi, S_\pi, m_\pi)$, onde D_π é o conjunto de instâncias de π; S_π é a função que associa a cada instância $I \in D_\pi$ um conjunto finito $S_\pi(I)$ de candidatos a solução para I; e m_π função que atribui a cada instância $I \in D_\pi$ e cada candidata a solução $\sigma \in S_\pi(I)$ um número racional positivo $m_\pi(I,\sigma)$, chamado valor solução para σ.

Seja π um problema de otimização (minimização/maximização), uma solução ótima para uma instância $I \in D_\pi$ é uma candidata a solução $\sigma^* \in S_\pi(I)$ de melhor valor, i.é tal que para todo $\sigma \in S_\pi(I)$, $m_\pi(I,\sigma^*) \leq m_\pi(I,\sigma)$ $(m_\pi(I,\sigma^*) \geq m_\pi(I,\sigma))$. E chama-se $OPT(I)$ a $m_\pi(I,\sigma^*)$.

Um algoritmo A é dito um algoritmo aproximativo para um problema $\pi = (D_\pi, S_\pi, m_\pi)$ sss para qualquer ins-

tância $I \in D_\pi$, A encontra $\sigma \in S_\pi(I)$ uma candidata a solução. E se diz que $A(I) = m_\pi(I,\sigma)$.

Um algoritmo A tal que para toda instância $I \in D_\pi$, $A(I) = OPT(I)$ é dito algoritmo de otimização para π.

3. MEDIDAS DE QUALIDADE DE ALGORITMOS APROXIMATIVOS

Os algoritmos aproximativos produzem uma aproximação da solução exata, mas então é preciso quantificar esta proximidade, para medir a qualidade do algoritmo.

Sejam π um problema de minimização (ou maximização), I uma instância de $\pi(I \in D_\pi)$ e A um algoritmo aproximativo para π, a razão $R_A(I)$, qualidade de A para I, é definida por

$$R_A(I) = \frac{A(I)}{OPT(I)} \quad (\text{ou } R_A(I) = \frac{OPT(I)}{A(I)}),$$

a qualidade absoluta de A, R_A é dada por:

$$R_A = \inf\{r \geq 1 \ / \ R_A(I) \leq r \text{ para toda instância } I \in D_\pi\}$$

e de qualidade assintótica R_A^∞ é dada por

$$R_A^\infty = \inf\{r \geq 1 \ / \text{ para algum } N \in Z^+, R_A(I) \leq r \text{ para todo } I \in D_\pi \text{ satisfazendo } OPT(I) \geq N\}$$

Muitas aplicações exigem uma qualidade mínima para aceitação de um algoritmo. Esta exigência é posta como requerimento de exatidão $\varepsilon > 0$, apartir dos quais são definidos esquemas aproximativos. É usado o termo esquema, porque para cada ε é definido um algoritmo A_ε.

Dado um problema π, um esquema aproximativo é um algoritmo A tal que dado um requerimento de exatidão ε, deriva um algoritmo A_ε tal que para uma instância $I \in D_\pi$, $R_{A_\varepsilon}(I) \leq 1 + \varepsilon$.

4. EXEMPLO

Para ilustrar é apresentado aqui um algoritmo aproximativo para o Problema do Mínimo Equivalente em Grafos (MEQ), esse algoritmo é devido a J. L. Szwarcfiter /SZW 85/.

Problema MEQ: "dado um digrafo D, encontrar o menor subgrafo gerador de D que preserve sua alcançabilidade".

Este problema é NP-difícil. O algoritmo aproximativo de J. L. Szwarcfiter tem $R_A \leq 2$.

4.1 - Algoritmo

O algoritmo foi desenvolvido para dígrafos fortemente conexos, mas uma modificação é sugerida para adequá-lo a dígrafos não fortemente conexos.

Algoritmo

entrada: D = (V,E)

1. $C \leftarrow \emptyset$; $i \leftarrow 1$;
2. Escolha arbitrariamente um ciclo C_1 em D e faça
3. $D_1 \leftarrow C_1$; $C \leftarrow C \cup \{\text{nodos de } C_1\}$;
4. Enquanto $C \neq V$ faça
5. Apartir de v encontre w em D e P(v,w) t.q. v, w $\in$ C e
6. exceto pelos extremos v e w, P(v,w) não contém
7. nodos de C;
8. $C_{i+1} \leftarrow$ P(v,w) + um caminho de w a v de D_i.
9. $D_{i+1} \leftarrow D_i + C_{i+1}$;
10. $C \leftarrow C \cup \{\text{nodos de } C_{i+1}\}$;
11. $i \leftarrow i + 1$;
12. fim-enquanto
13. fim-com-saída: D_i

Se o digrafo não é fortemente conexo, aplique o algoritmo para cada componente fortemente conexo e depois inclua as arestas (s_i, s_j) de D, sendo s_i e s_j nodos de componentes fortemente conexos diferentes.

4.2 Complexidade do algoritmo

A linha 2 tem complexidade $O(\max\{|V|,|E|\})$. Na linha 5, apartir de v é construido um caminho P(v,w), até atingir um nodo w e D. Neste caminho cada aresta de D é considerada no máximo uma vez. A complexidade total do algoritmo é então $O(\max\{|V|,|E|\})$.

4.3 Qualidade do algoritmo

Quanto a qualidade do algoritmo se pode considerar dois casos:

Caso 1: D fortemente conexo

A saída do algoritmo, D_i, é o digrafo constituído dos ciclos $C_1, C_2, \ldots, C_i$. O primeiro ciclo C_1 cobre

$n_1 > 1$ nodos e tem n_1 arestas. Cada ciclo subsequente C_j $1 < j \leq i$, cobre $n_j > 1$ novos nodos (não cobertos por C_1, C_2, ..., C_{i-j}) e tem precisamente $n_j + 1$ arestas não pertencentes a ciclo algum entre C_1, C_2, ..., C_{j-1}.

$$\sum_{j=1}^{i} n_j = |V|$$

E o número de arestas de D_i é $(\sum_{j=1}^{i} n_j+1)-1 =$

$$= i+\sum_{j=1}^{i} n_j-1$$

$$= i + |V| - 1$$

$$\leq 2\,|V| - 1$$

Então a solução encontrada pelo algoritmo aproximativo é $S \leq 2\,V - 1$, enquanto a solução ótima é $S^* \geq |V|$

Logo $R_A = \frac{S}{S^*} \leq \frac{2|V|-1}{|V|} < 2$

Caso 2: D não é fortemente conexo

Se o digrafo não é fortemente conexo as arestas que conectam os diferentes componentes ocorrem na solução aproximada, na mesma freqüência que aparecem na solução exata, portanto a relação $R_A < 2$ se mantêm.

5. RESULTADOS CORRELATOS

Algoritmos aproximativos resolveram satisfatoriamente muitos problemas, mas para outros problemas não foi possível obter-se um bom algoritmo aproximativo, outros ainda possuem algoritmos pseudopolinomiais. Neste secção estas questões são consideradas.

5.1 Problemas que não admitem algoritmo aproximativo polinomial

O problema do Caixeiro Viajante é um dos mais conhecidos problemas NP-completos e pode ser definido assim:

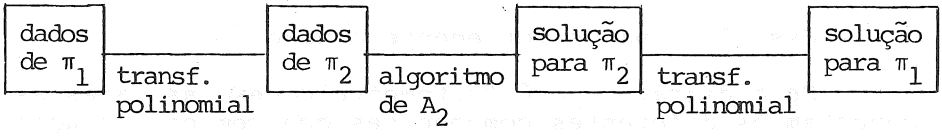
Problema C V: Seja um conjunto finito $C = \{c_1, c_2, \ldots c_n\}$ de cidades e $d(c_i, c_j) \in Z^+$ a distância entre as cidades c_i e $c_j \in C$. A questão é achar o menor percurso para o Caixeiro Viajante, que saindo de uma das cidades visite todas as outras cidades exatamente uma vez e retorne a cidade de origem.

Este problema, interpretado na Teoria dos Grafos consiste em achar um circuito hamiltoniano de custo mínimo, para o grafo G = (V,E), onde V=C conjunto de cidades e E, o conjunto de arestas, é definido como o conjunto de todos pares (c_i, c_j) tal que $d(c_i, c_j) < \infty$, d dá o custo de cada aresta e o custo de um caminho em G é a soma dos custos de cada aresta do caminho.

Através da Programação Dinâmica é possível projetar um algoritmo para o problema CV com complexidade $0(n^2\ 2^n)$ /TOS 86/ que é bem melhor que o algoritmo trivial que o enumera os n! permutações dos n nodos do grafo. Mas seria interessante se obter um algoritmo aproximativo de complexidade polinomial. Infelizmente a existência de tal algoritmo implica em P=NP /GAR 79/.

5.2 Transformação Polinomial

Dados dois problemas NP-completos π_1 e π_2, π_1 pode ser transformado polinomialmente em π_2, segundo o diagrama abaixo e o algoritmo A_2 que resolve π_2 pode ser usado para resolver π_1.



Assim, se π_2 tem algoritmo polinomial que o resolva, π_1 também tem.

Logo, é de se esperar que se π_2 tem um algoritmo aproximativo A_2' de tempo polinomial, o mesmo método gere um algoritmo aproximativo polinomial, A_1', para π_1. Infelizmente isto não acontece.

Sejam os dois seguintes problemas de Teoria dos Grafos:

Problema do Máximo Conjunto Independente (MCI): Dado um grafo G = (V,E), $V' \leq V$ é um conjunto independente se $\forall u, v \in V$ $(u,v) \notin E$. Um conjunto máximo independente é um conjunto independente V_{max} tal que para todo conjunto independente V', $V' \not\subseteq V_{max}$.

Problema da Cobertura de Vértices Mínima (CVM): Dado um grafo G = (V,E), $V' \subseteq V$ é uma cobertura de vértices de G se $\forall$ $(u,v) \in E$, $u \in V'$ ou $v \in V'$. Cobertura de vértices mínima é uma cobertura de vértices V_{min} tal que para toda cobertura de vértice V', $V' \not\subseteq V_{min}$.

Estes dois problemas: MCI e CVM são NP-completos e estão intimamente ligados da seguinte forma: se G=(V,E) é um grafo e $V' \subseteq V$ é máximo conjunto independente de G, V - V' é uma cobertura de vértices mínima para G. Entretanto o problema MCI não possui algoritmo aproximativo com $R_A < \infty$ enquanto que para o problema CVM tem algoritmo aproximativo com $R_A \leq 2$ /GAR 79/.

Para exemplificar como a transformação que preserva a otimalidade de uma solução não necessariamente preserva a qualidade de uma solução aproximada /GAR 79/ apresenta o seguinte caso: suponha que se tem um grafo G=(V,E), com 1000 vértices ($|V|$ =1000), a cobertura de vértices mínima para G tem 490 vértices e o algoritmo aproximativo A, para o problema tem $R_A \leq 2$ portanto a solução encontrada pelo algoritmo, V' é tal que $|V'| \leq 980$. Usando a transformação entre os problemas apresentada antes (máximo conjunto independente = V - V'), chega-se a seguinte razão:

$$R_A(I) = \frac{1000-490}{1000-980} = \frac{510}{20} = 25,5$$

5.3 Algoritmos pseudopolinomiais

A complexidade de um algoritmo é calculada em função do tamanho da entrada, mas então depende da codificação da entrada. Um número k codificado em binário é representado por $\lfloor \log_2 k \rfloor + 1$ dígitos. Esta codificação é tradicionalmente aceita e uma codificação em uma base maior que dois é também aceita porque não altera a grandeza da complexidade do algoritmo, i.é se no primeiro caso, um algoritmo é de complexidade polinomial ou exponencial, continua sendo polinomial ou exponencial se a base de codificação for alterada para uma base maior. Mas se for utilizada a representação unária. O resultado não é o mesmo. Este enfoque propicia a definição de outros problemas.

Um algoritmo A é dito de complexidade pseudopolinomial, quando sua complexidade é polinomial para o tamanho da entrada, quando a entrada é codificada em unário.

Um problema NP-completo que admite algoritmo pseudopolinomial é dito "NP-completo fraco". Existem também problemas cuja possível existência de algoritmo pseudopolinomial implica na igualdade P=NP. Esses problemas são chamados "NP-completos forte". O problema do Caixeiro Viajante é NP-completo forte /GAR 79/.

Um exemplo de problema pseudopolinomial é o problema da partição /GAR 79/.

Existem problemas cujos valores dos dados de entrada crescem polinomialmente com a quantidade de dados, nesses casos um algoritmo pseudopolinomail é na verdade um

algoritmo polinomial e se o problema é NP-completo é NP--completo forte. Um exemplo, é o problema do circuito Hamiltoniano: a entrada é um grafo G(V,E), o maior valor de um dado, então, não necessita exceder de n = |V| ou m = |E| e a quantidade de dados é max {m,n}, mesmo se codificada em unário /SZW 84/.

Um resultado interessante foi obtido por Ibarra e Kim, que converteram um algoritmo pseudopolinomial para o problema da Mochila num algoritmo aproximativo com uma perda limitada de exatidão /GAR 79/.

A idéia é utilizar um método de desenvolvimento de algoritmos aproximativos que Horowitz em /HOR 78/ chama de "rouding", que consiste em dada uma instância de um problema de otimização encontrar outra instância tal que a solução ótima do segundo interpretado na instância original é uma boa aproximação da solução da instância original. E a solução ótima da instância modificada é obtida em tempo polinomial. Esse método pode ser aplicado a qualquer problema de otimização do tipo: Calcular

$$\max_{x} \sum_{i=1}^{n} p_i x_i \text{ restrito à } \sum_{i=1}^{n} a_{ij} x_j \leq t \quad 1 \leq j \leq n, x_i = 0$$

ou 1, $1 \leq i \leq n$, p_i, $a_{ij} \geq 0$.

6. CONCLUSÕES

Muitos problemas importantes, de grande aplicabilidade são NP-completos, portanto os algoritmos conhecidos que os resolvem são ineficientes, assim duas alternativas se apresentam:

- Evitar a procura exaustiva, fazendo uma escolha inteligente, evitando soluções parciais que certamente não levam a uma solução total. Nesse sentido é muito útil a técnica de desenvolvimento de algoritmo Programação Dinâmica /TOS 86/ e /HOR 78/.

- Para problemas de otimização, ao invés de procurar a solução ótima, procurar uma boa solução em um tempo razoável, i. é procurar um algoritmo aproximativo. Duas técnicas são usadas para projeto de algoritmos aproximativos. Uma consiste em construir iterativamente conjuntos de soluções parciais, dividir o domínio das soluções parciais em intervalos, escolher uma solução em cada intervalo, até alcançar uma solução total na vizinhança (tão próxima quanto se queira) da solução ótima. Outra é modificar a instância original de maneira a, em tempo polinomial encontrar a solução ótima da instância modificada, que deve ser uma boa aproximação (tão boa quanto se queira) da solução ótima da instância original /HOR 78/.

Através de algoritmos aproximativos muitos problemas, como o exemplo da secção 5, estão satisfatoriamente resolvidos.

Infelizmente existem também os problemas que não admitem bons algoritmos aproximativos em tempo polinomial, a não ser se P = NP, como o problema do Caixeiro Viajante apresentado na secção 5.1. Assim o problema de achar um algoritmo aproximativo polinomial, de qualidade assintótica limitada para o CV é então também um problema intratável. Para o problema de coloração de grafos não se conhece algoritmo aproximativo com qualidade absoluta limitada ($R_A < \infty$).

Outros problemas, ainda, estão numa classe meio nebulosa, como o de decidir se duas expressões regulares são equivalentes e o de decidir se uma dada palavra é gerada por uma certa gramática sensível do contexto. Esses problemas são NP-difíceis, i.é são reduzíveis polinomialmente a problemas NP-completos. Como problemas NP-difíceis tem a propriedade de não serem resolvidos em tempo polinomial a não ser se P=NP, mas não se sabe se são eles, problemas NP ou não /WEI 77/. Ou problemas como: decidir se dois grafos são isomorfos, ou dado um inteiro, saber se ele é primo. Esses são problemas NP, mas não foi possível reduzí-los polinomialmente a um problema NP-completo, nem se conhece algoritmos polinomiais que os resolva.

Esforços estão sendo dispendidos no sentido de estender os resultados positivos obtidos na área, como transformar um algoritmo pseudopolinomial em um algoritmo aproximativo polinomial, com perda limitada de qualidade; aplicar técnicas de otimização para melhorar algoritmos aproximativos e transformar algoritmos aproximativos, para aplicá-los a outros problemas. Nem sempre, entretanto, os resultados são os esperados. A existência ou não de algoritmos aproximativos de qualidade assintótica limitada parece não respeitar o fato de muitos problemas estarem intimamente relacionados por uma transformação polinomial, não sendo possível usar a mesma transformação para transformar um algoritmo aproximativo para um problema em um algoritmo aproximativo para o outro, sem perda considerável de qualidade, como no exemplo da secção 5.2.

No fracasso de uma tentativa de alcançar um resultado positivo (como encontrar um bom algoritmo aproximativo, ou transformar um algoritmo pseudopolinomial em um bom algoritmo aproximativo, ou ...) é tentado provar que este resultado só poderá ser atingido se P=NP. Alcançada essa prova, o caso é considerado fechado.

Na área de Escalonamento de Tarefas muitos são os problemas NP-completos /GAR 79/, /HOR 78/, /PIN 83/. Algoritmos aproximativos podem ser claramente a melhor solução, como por exemplo para o problema de encontrar o escalonamento ótimo de tarefas com penalidades, no caso em que

o custo em tempo de computador necessário para encontrar a solução ótima excede a maior penalidade.

Para outros problemas os algoritmos aproximativos não tem uma vantagem tão clara, mas para instâncias relativamente grandes são a única solução viável.

BIBLIOGRAFIA

/AHO 74/ AHO, A.V.; HOPCROFT, J.E. & ULLMAN, J.D. The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, 1974.

/COO 83/ COOK, S.A. An overview of Computational Complexity CACM 26,6, 1983. pp401-7.

/GAR 79/ GAREY, M.R.; JOHNSON, D.S. Computers and Intractability. A guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.

/HOR 78/ HOROWITZ, E., SAHNI, S. Fundamentals of Computer Algorithms, Computer Science, Press Rockville, 1978.

/HUY 80/ HUYN, N.; DECHTER, R. & PEARL, J. Probabilistic Analysis of the Complexity of A* Artificial Intelligence. 15(1980). pp241-54.

/PIN 83/ PINHO, A de A. Algoritmos Polinomiais Exatos ou Aproximativos para certos Problemas em Scheduling Tese de Doutorado. UFRJ, Rio de Janeiro, 1983.

/SZW 84/ SZWARCFITER, J.L. Grafos e Algoritmos Computacionais. Editora Campos, Rio de Janeiro, 1984.

/SZW 86/ SZWARCFITER, J.L. On Digraphs with a Rooted Tree Structure. Network vol. 15 (1985) 49-57.

/TOS 86/ TOSCANI, L.V. & VELOSO, Paulo, A.S. Especificação Formal e Análise da Complexidade da Programação Dinâmica. Porto Alegre, CPGCC/UFRGS, 1986.

/WEI 77/ WEIDE, B. A Survey of Analysis Techniques for Discrete Algorithms. Computing Surveys, 9,4. 1977. pp.291-313.