Inaproximabilidade

Marina Andretta

ICMC-USP

26 de novembro de 2015

Baseado no livro Uma introdução sucinta a Algoritmos de Aproximação, de M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. Piña Jr., J. A. R. Soares e Y. Wakabayashi.

Introdução

Por que alguns problemas de otimização parecem ser mais difíceis de aproximar do que outros?

Como podemos medir ou comprovar estes diferentes graus de dificuldade?

Veremos agora como classificar problemas de otimização.

Introdução

Cook mostrou que a classe NP contém problemas completos, os chamados problemas NP-completos. Estes problemas são pelo menos tão difíceis quanto qualquer outro problema na classe.

Logo depois, Karp apresentou uma grande coleção de problemas NP-completos, que incluía vários problemas de decisão correspondentes a problemas de otimização combinatória bem conhecidos.

É crença geral que não existem algoritmos polinomiais para resolver problemas NP-completos, ou seja, acredita-se que $P \neq NP$.

Introdução

Assim como problemas de decisão são classificados conforme a sua dificuldade computacional, problemas de otimização são classificados conforme o seu grau de aproximabilidade por algoritmos polinomiais.

A seguir, serão definidas as classes *PO*, *FPTAS*, *PTAS*, *APX* e *NPO* de problemas de otimização.

Introdução

Estudamos durante o curso diversos algoritmos de aproximação para vários problemas *NP*-difíceis.

Para nenhum dos problemas estudados foi apresentada evidência de que não existem algoritmos com melhor razão de aproximação.

Mostraremos que, frequentemente, a dificuldade em se obter uma melhor razão de aproximação está intrinsecamente ligada à questão "$P \neq NP$?".

Introdução

Apresentamos resultados de inaproximabilidade que têm tipicamente a seguinte forma:

se existe uma α-aproximação polinomial para Π, então $P = NP$,

onde Π é um problema de otimização.

Este resultado indica que a existência de uma α-aproximação polinomial para Π é improvável.

Por outro lado, apesar de não se acreditar que este seja o caso, se $P = NP$ então, em particular, existe um algoritmo exato polinomial para cada problema tratado durante o curso e toda a discussão sobre algoritmos de aproximação perde parte de seu sentido.

Classes de problemas de otimização

Como já vimos, um problema de otimização tem três ingredientes:

- um conjunto de instâncias;
- um conjunto $Sol(I)$ de soluções viáveis de cada instância I; e
- uma função val que associa a cada instância I e solução S em $Sol(I)$ um valor racional não-negativo $val(I, S)$.

Se o problema é de minimização o objetivo é encontrar, para qualquer instância dada, uma solução viável de valor mínimo.

Se o problema é de maximização o objetivo é encontrar uma solução viável de valor máximo.

Classes de problemas de otimização

Em suma, um problema de otimização Π consiste no seguinte:

Problema $\Pi(I)$: Dada uma instância I, encontrar S em $Sol(I)$ que minimize (maximize) $val(I, S)$.

Uma solução ótima para um problema de otimização é uma solução viável que minimize (maximize) o valor da função val.

Denotamos por $opt(I)$ o valor de uma solução ótima da instância I.

Classes de problemas de otimização

A classe de problemas *NPO*, que é a extensão de *NP* a problemas de otimização, é formada pelos problemas de otimização para os quais:

- existe uma função polinomial p tal que $\langle S \rangle \leq p(\langle I \rangle)$ para toda instância I do problema e toda solução viável S de I;
- existe um algoritmo polinomial que decide se uma dada palavra é uma representação válida de uma instância do problema;
- existe um algoritmo polinomial que decide se um dado objeto é solução viável de uma dada instância do problema;
- existe um algoritmo polinomial que calcula $val(I, S)$, dados I e S.

Classes de problemas de otimização

Denotamos pela sigla *PO* o conjunto dos problemas em *NPO* para os quais existe um algoritmo exato polinomial.

Ou seja, esta é uma extensão da classe P a problemas de otimização.

A classe *APX* é formada pelos problemas em *NPO* para os quais existe uma α-aproximação polinomial para alguma constante α.

Classes de problemas de otimização

Um esquema de aproximação (*approximation scheme*) para um problema de otimização é um algoritmo A que

- recebe um número racional positivo ϵ e uma instância I e
- devolve uma solução viável $A(\epsilon, I)$ com um erro relativo de no máximo ϵ, ou seja,

$$(1 - \epsilon)opt(I) \leq val(I, A(\epsilon, I)) \leq (1 + \epsilon)opt(I).$$

No caso de problema de maximização, somente a desigualdade esquerda é relevante e ela só faz sentido para ϵ no intervalo $(0, 1)$.

Se o problema é de minimização, somente a desigualdade direita interessa.

Classes de problemas de otimização

Dizemos que A é um esquema de aproximação polinomial (*polynomial-time approximation scheme*) se o algoritmo $A(\epsilon, \cdot)$ é polinomial para todo ϵ fixo.

Além disso, quando a quantidade de tempo consumida pelo algoritmo A é limitada por $p(\langle I \rangle, 1/\epsilon)$ para alguma função polinomial p, então A é esquema de aproximação plenamente polinomial (*fully polynomial-time approximation scheme*).

Por exemplo, para o problema da mochila (MOCHILA), o algoritmo MOCHILA-IK$_\epsilon$ é um esquema de aproximação plenamente polinomial.

Classes de problemas de otimização

A classe *PTAS* é composta pelos problemas em *NPO* que possuem um esquema de aproximação polinomial.

Já classe *FPTAS* é composta pelos problemas em *NPO* que admitem um esquema de aproximação plenamente polinomial.

Das definições, segue imediatamente que

$$PO \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq NPO.$$

Como será visto mais adiante, se alguma das inclusões acima não for estrita, então $P = NP$. Por outro lado, se $P = NP$, então $PO = NPO$.

Classes de problemas de otimização

Como pode ser verificado, todos os problemas de otimização estudados ao longo do curso estão em *NPO*.

Assim, os algoritmos de aproximação vistos distribuem esses problemas nas classes definidas acima.

Classes de problemas de otimização

Teorema 1: O problema Mochila está em *FPTAS*.

Teorema 2 (de Hochbaum e Shmoys): O problema Escalonamento está em *PTAS*.

Teorema 3: Os problemas Empacotamento, MaxSat, MinCV, MinFS e TSPM estão em *APX*.

Teorema 4: Os problemas MinCC, MinMCut, MinTC e TSP estão em *NPO*.

NP-completude e inaproximabilidade

A partir de agora vamos mostrar que a dificuldade em obter-se uma melhor razão de aproximação para alguns problemas tratados ao longo do curso é herdada da teoria de *NP*-completude e da clássica questão "$P \neq NP$?".

Teorema 5: Se $APX = NPO$, então $P = NP$.

Demonstração: Pelo teorema 4, o problema TSP está em NPO.

Logo, é suficiente mostrar o seguinte teorema, de Sahni e Gonzalez: se o problema TSP está em APX, então $P = NP$.

Seja A uma α-aproximação polinomial para o TSP, onde α é uma constante.

Utilizando A, pode-se criar um algoritmo polinomial que resolve o problema do circuito hamiltoniano, denotado por PCH.

O algoritmo é o seguinte:

1. dado um grafo G, instância arbitrária do PCH, construa a instância (K, c) do TSP, onde K é o grafo completo com conjunto V_G de vértices e custo $c_e = 1$ se a aresta e está em G e $c_e = \alpha|V_G|$ se e não está em G.
2. Aplique o algoritmo A à instância (K, c).

Note que uma solução viável da instância (K, c) do TSP é um circuito hamiltoniano em K e o valor da solução viável é o custo do circuito.

Se G é hamiltoniano então $opt(K, c) = |V_G|$ e, se G não é hamiltoniano, então $opt(K, c) > \alpha|V_G|$.

Ou seja, $val((K, c), A(K, c)) \leq \alpha|V_G|$ se e somente se G é hamiltoniano.

Note que a execução do algoritmo consome uma quantidade de tempo limitada por uma função polinomial de $\langle G \rangle$.

Assim, a existência de um A mostra que PCH está em P.

Como PCH é *NP*-completo, temos que $P = NP$.

Teorema 6: Se $PTAS = APX$, então $P = NP$.

Demonstração: Pelo teorema 3, o problema EMPACOTAMENTO está em APX.

Portanto, basta provar que se o problema EMPACOTAMENTO está em $PTAS$, então $P = NP$.

NP-completude e inaproximabilidade

Seja A um esquema de aproximação polinomial para o problema Empacotamento.

Utilizando o algoritmo $A(1/3, \cdot)$, pode-se projetar um algoritmo polinomial o problema Partição (que é *NP*-completo).

Problema Partição(n, v): Dados um inteiro positivo n e, para cada i em $\{1, ..., n\}$, um número v_i em $\mathbb{Q}_{\geq}$, decidir se existe uma partição $\{B_1, B_2\}$ de $\{1, ..., n\}$ tal que $v(B_1) = v(B_2)$.

O algoritmo polinomial para PARTIÇÃO é o seguinte:

1. dada uma instância arbitrária (n, v) do problema, calcule $\sigma := \left(\sum_i v_i\right)/2$.
2. Podemos supor que $\sigma \neq 0$ e que $v_i \leq \sigma$ para todo i.

   Seja $c_i := v_i/\sigma$ para cada i.
3. Utilize o algoritmo $A(1/3, \cdot)$ para resolver EMPACOTAMENTO(n, c).
4. A resposta do problema PARTIÇÃO(n, v) é SIM se e somente se $val((n, c), A(1/3, (n, c))) = 2$.

Se a resposta a PARTIÇÃO(n, v) é SIM então o valor ótimo $opt(n, c)$ de EMPACOTAMENTO(n, c) é 2.

Caso contrário, é pelo menos 3.

Como $val((n, c), A(1/3, (n, c)))$ é um número inteiro e

$$val((n, c), A(1/3, (n, c))) \leq (1 + 1/3)opt(n, c) = (4/3)opt(n, c),$$

a resposta do problema PARTIÇÃO(n, v) é SIM se e somente se $val((n, c), A(1/3, (n, c))) = 2$.

Note que o algoritmo proposto consome uma quantidade de tempo limitada por uma função polinomial de $\langle n \rangle + \langle v \rangle$.

Logo, ele resolve Partição em tempo polinomial.

Como Partição é *NP*-completo, temos que $P = NP$.

Se I é uma instância de um problema, então $Max(I)$ é definido como o maior número inteiro em valor absoluto que ocorre em I.

Defina $Max(I) := 0$ se nenhum número inteiro ocorre em I (MaxSat é um exemplo em que isso ocorre).

Estamos supondo que os números racionais são representados como quociente entre dois números inteiros.

Assim, por exemplo, se o número $3/17$ ocorre em I temos que $Max(I) \geq 17$.

Teorema 7 (Garey e Johnson): Seja Π um problema de otimização *NP*-difícil no sentido forte tal que $val(I, S)$ é um número inteiro não-negativo para toda instância I e todo S em $Sol(I)$.

Seja ainda $p(n, m)$ uma função polinomial tal que

$$opt(I) \leq p(\langle I \rangle, Max(I))$$

para toda instância I de Π.

Se Π está em *FPTAS*, então $P = NP$.

Demonstração: Suponha que Π é um problema de minimização (argumentos simétricos se aplicam a problemas de maximização) e seja A um esquema de aproximação plenamente polinomial para Π.

Utilizando o esquema A, pode-se projetar um algoritmo pseudopolinomial para resolver o problema Π, provando assim que $P = NP$.

Dada uma instância I, calcule $\epsilon := 1/(p(\langle I \rangle, Max(I)) + 1)$ e aplique o algoritmo A aos argumentos ϵ e I.

Em uma quantidade de tempo limitada por uma função polinomial em $\langle I \rangle$ e $1/\epsilon$ (portanto, em tempo pseudopolinomial) o algoritmo A devolve um elemento $A(\epsilon, I)$ em $Sol(I)$ que satisfaz $val(I, A(\epsilon, I)) \leq (1+\epsilon)opt(I)$.

Ou seja,

$$val(I, A(\epsilon, I)) - opt(I) \leq \epsilon opt(I) = \frac{opt(I)}{p(\langle I \rangle, Max(I)) + 1} < 1,$$

para toda instância I.

NP-completude e inaproximabilidade

Como $val(I, A(\epsilon, I))$ e $opt(I)$ são inteiros, $val(I, A(\epsilon, I)) = opt(I)$.

Ou seja, o algoritmo pseudopolinomial projetado resolve Π exatamente.

Como Π é *NP*-difícil no sentido forte, temos que $P = NP$.

NP-completude e inaproximabilidade

No teorema 7, a hipótese dos valores das soluções viáveis serem números inteiros é apenas aparentemente restritiva.

Todos os problemas formulados ao longo do curso envolvem números racionais. Entretanto, cada um desses problemas pode ser reformulado como um problema em que os valores das soluções viáveis são números inteiros e que é polinomialmente equivalente ao problema original.

Mais ainda, um algoritmo de aproximação para um dos problemas pode ser transformado em um algoritmo de mesma razão de aproximação para o outro.

NP-completude e inaproximabilidade

Considere, por exemplo, a reformulação abaixo do problema Escalonamento.

Problema EscalonamentoInt(m, n, t): Dados inteiros positivos m e n e um tempo t_i em $\mathbb{Z}_{\geq}$ para cada i em $\{1, ..., n\}$, encontrar uma partição $\{M_1, ..., M_m\}$ de $\{1, ..., n\}$ que minimize $\max_j t(M_j)$.

Os problemas Escalonamento e EscalonamentoInt são polinomialmente equivalentes.

Além disso, uma α-aproximação polinomial para um dos problemas pode ser transformada em uma α-aproximação polinomial para o outro.

Logo, pelo teorema 2, EscalonamentoInt também está em *PTAS*.

NP-completude e inaproximabilidade

As hipóteses principais do teorema 7 são que as soluções viáveis de Π têm valor inteiro "não muito grande" e que Π é *NP*-difícil no sentido forte.

Estas hipóteses se aplicam a vários problemas, como MaxSat, MinCV e o EscalonamentoInt, que estão em *APX*.

Desta forma, se algum desses problemas admitir um esquema de aproximação plenamente polinomial, então $P = NP$.

NP-completude e inaproximabilidade

Como, em particular, EscalonamentoInt está em *PTAS*, então é improvável que existam esquemas de aproximação plenamente polinomiais para todos os problemas em *PTAS*.

Teorema 8: Se $FPTAS = PTAS$, então $P = NP$.

Como Mochila é um problema *NP*-difícil que está em *FPTAS*, então é improvável que existam algoritmos polinomiais para todos os problemas em *FPTAS*.

Teorema 9: Se $PO = FPTAS$, então $P = NP$.

Finalmente, o teorema a seguir mostra que, com exceção do teorema 7, vale a recíproca de cada teorema visto nesta aula.

Teorema 10: Se $P = NP$, então $PO = NPO$.

Completude para problemas de otimização

Informalmente, problemas completos de uma certa classe são aqueles pelo menos tão difíceis quanto qualquer outro da classe.

Para a classe NP, isto significa que um algoritmo polinomial para um problema NP-completo pode ser transformado em um algoritmo polinomial para resolver qualquer outro problema em NP.

Portanto, se algum problema NP-completo está em P, então $P = NP$.

Completude para problemas de otimização

O conceito de completude pode ser estendido para problemas de otimização.

Da mesma maneira que foi conveniente para a classe *NP* utilizar uma redução que preserva polinomialidade, para problemas de otimização é necessário um tipo de redução que, além da polinomialidade, preserve também a razão de aproximação.

Concretamente, usaremos uma redução que preserva a existência de um esquema de aproximação polinomial.

Completude para problemas de otimização

Uma *AP*-redução de um problema de otimização Π a um problema de otimização Π' é um terno (f, g, β) em que f e g são algoritmos e β é um número racional positivo tais que:

(AP1) f recebe um número racional positivo δ e uma instância I de Π, e devolve uma instância $f(\delta, I)$ de Π';

(AP2) g recebe um número racional positivo δ, uma instância I de Π e um elemento S' em $Sol(f(\delta, I))$, e devolve uma instância $g(\delta, I, S')$ em $Sol(I)$;

(AP3) para todo número racional positivo δ, os algoritmos $f(\delta, \cdot)$ e $g(\delta, \cdot, \cdot)$ são polinomiais; e

Completude para problemas de otimização

(AP4) para toda instância I de Π, todo número racional positivo δ, e todo S' em $Sol(f(\delta, I))$, vale que, se

$$(1-\delta)opt(f(\delta, I)) \leq val(f(\delta, I), S') \leq (1+\delta)opt(f(\delta, I)),$$

então

$$(1-\beta\delta)opt(I) \leq val(I, g(\delta, I, S')) \leq (1+\beta\delta)opt(I).$$

Completude para problemas de otimização

Usaremos a notação $\Pi \leq_{AP} \Pi'$ para denotar a existência de uma AP-redução de Π a Π'.

Teorema 11: Se $\Pi_1 \leq_{AP} \Pi_2$ e $\Pi_2 \leq_{AP} \Pi_3$, então $\Pi_1 \leq_{AP} \Pi_3$.

Teorema 12: Se Π está em NPO, Π' está em APX e $\Pi \leq_{AP} \Pi'$, então Π está em APX.

Completude para problemas de otimização

O próximo teorema mostra que AP-redução preserva a existência de um esquema de aproximação polinomial.

Teorema 13: Se Π está em NPO, Π' está em $PTAS$ e $\Pi \leq_{AP} \Pi'$, então Π está em $PTAS$.

Demonstração: Seja A' um esquema de aproximação polinomial para Π' e seja (f, g, β) uma AP-redução de Π a Π'.

Utilizando o esquema A' e os algoritmos f e g criaremos um esquema de aproximação polinomial A para Π.

O algoritmo A abaixo recebe um número racional positivo ϵ e uma instância I de Π e devolve S em $Sol(I)$ com erro relativo de no máximo ϵ.

Completude para problemas de otimização

Algoritmo $A(\epsilon, I)$:

1. faça $\delta \leftarrow \epsilon/\beta$;
2. faça $I' \leftarrow f(\delta, I)$;
3. faça $S' \leftarrow A'(\delta, I')$;
4. faça $S \leftarrow g(\delta, I, S')$;
5. devolva S.

Completude para problemas de otimização

De (AP3) e do fato de A' ser um esquema de aproximação polinomial, segue que, para todo ϵ, o algoritmo $A(\epsilon, \cdot)$ é polinomial.

As propriedades (AP1) e (AP2) e a definição de esquema de aproximação garantem que

1. I' é uma instância de Π';
2. S' está em $Sol(I') = Sol(f(\delta, I))$;
3. S está em $Sol(I)$; e
4. $(1-\delta)opt(I') \leq val(I', S') = val(f(\delta, I), S') \leq (1+\delta)opt(f(\delta, I))$.

Completude para problemas de otimização

Logo, de (AP4), tem-se que

$$(1-\epsilon)opt(I) \leq val(I,S) \leq (1+\epsilon)opt(I),$$

já que $\beta\delta = \epsilon$.

Em suma, $A(\epsilon, \cdot)$ é um algoritmo polinomial que recebe uma instância I de Π e devolve uma solução viável $A(\epsilon, \cdot)$ com erro relativo de no máximo ϵ, como desejado.

Como Π está em *NPO*, tem-se que Π está em *PTAS*.

Completude para problemas de otimização

Um problema Π em *APX* é *APX*-completo se cada problema em *APX* pode ser *AP*-reduzido a Π.

O primeiro problema que se demonstrou ser *APX*-completo foi MaxSat.

Teorema 14 (Papadimitriou e Yannakakis; Khanna, Motwani, Sudan e Vazirani): O problema MaxSat é *APX*-completo.

Completude para problemas de otimização

Um problema Π, não necessariamente em APX, é APX-difícil se a existência de um esquema de aproximação polinomial para Π implica em $P = NP$.

Assim, por exemplo, MaxSat é APX-difícil, já que, pelos teoremas 6 e 13, a existência de um esquema de aproximação polinomial para qualquer problema APX-completo implica em $P = NP$.

Note que, se $PTAS \neq APX$, então todo problema APX-completo está em $APX \setminus PTAS$.

Teorema 15: Os problemas Empacotamento, MaxSat, MinCC, MinCV, MinFS, MinMCut, MinTC, TSPM e TSP são APX-difíceis.

Completude para problemas de otimização

Um problema Π em *NPO* é *NPO*-completo se cada problema *NPO* pode ser *AP*-reduzido a Π.

Observe que se $APX \neq NPO$, então todo problema *NPO*-completo está em $NPO \setminus APX$.

Teorema 16 (Orponen e Mannila): O problema TSP é *NPO*-completo.

Limiares de aproximação

O limiar de aproximação (*approximation threshold*) de um problema de minimização é o maior limitante inferior de todos os α para os quais existe uma α-aproximação polinomial para o problema.

No caso de problema de maximização troca-se, na definição acima, maior limitante inferior por menor limitante superior.

É claro que, se $P = NP$, então o limiar de aproximação de todo problema em NPO é 1.

Na tabela a seguir são apresentadas, sob a hipótese de $P \neq NP$, delimitações para os limiares de aproximação de problemas abordados no curso.

Limiares de aproximação

problema	limiar de aproximação
MOCHILA	$= 1$ (o problema está em *FPTAS*)
ESCALONAMENTO	$= 1$ (o problema está em *PTAS*)
EMPACOTAMENTO	$\geq 3/2$
MAXSAT	$\leq 7/8$
MINCV	$\geq 7/6$
TSPM	$\geq 131/130$
MINCC$(E, \mathcal{S}, c)$	$> \epsilon \log(\|E\|)$, para alguma constante $\epsilon > 0$
MINTC$(E, \mathcal{S}, c)$	$> \epsilon \log(\|E\|)$, para alguma constante $\epsilon > 0$
TSP(G, c)	$> f(\langle G, c \rangle)$, para toda função f computável em tempo polinomial