Algoritmos de Aproximação

Segundo Semestre de 2012

Introdução

WS: The Design of Approximation Algorithms
David Williamson & David Shmoys, 2011

WS cap 1

(um pouquinho do cap 1 do livro do colóquio)

Problemas de Otimização Combinatória

conjunto de instâncias

para cada instância I,
um conjunto $\mathrm{Sol}(I)$ de soluções viáveis e
uma função $\mathrm{val}(S)$ para cada solução S em $\mathrm{Sol}(I)$

Problemas de Otimização Combinatória

conjunto de instâncias

para cada instância I,
um conjunto $\mathrm{Sol}(I)$ de soluções viáveis e
uma função $\mathrm{val}(S)$ para cada solução S em $\mathrm{Sol}(I)$

Se $\mathrm{Sol}(I)$ é vazio, I é inviável, caso contrário, I é viável.

Problemas de Otimização Combinatória

conjunto de instâncias

para cada instância I,
um conjunto $\mathrm{Sol}(I)$ de soluções viáveis e
uma função $\mathrm{val}(S)$ para cada solução S em $\mathrm{Sol}(I)$

Se $\mathrm{Sol}(I)$ é vazio, I é inviável, caso contrário, I é viável.

Problema de minimização: Dada uma instância I,
encontrar uma solução S em $\mathrm{Sol}(I)$ tal que $\mathrm{val}(S)$ é mínimo.

Problema de maximização: Dada uma instância I,
encontrar uma solução S em $\mathrm{Sol}(I)$ tal que $\mathrm{val}(S)$ é máximo.

Problemas de Otimização Combinatória

conjunto de instâncias

para cada instância I,
um conjunto $\mathrm{Sol}(I)$ de soluções viáveis e
uma função $\mathrm{val}(S)$ para cada solução S em $\mathrm{Sol}(I)$

Se $\mathrm{Sol}(I)$ é vazio, I é inviável, caso contrário, I é viável.

Problema de minimização: Dada uma instância I,
encontrar uma solução S em $\mathrm{Sol}(I)$ tal que $\mathrm{val}(S)$ é mínimo.

Problema de maximização: Dada uma instância I,
encontrar uma solução S em $\mathrm{Sol}(I)$ tal que $\mathrm{val}(S)$ é máximo.

$\mathrm{opt}(I)$: valor ótimo (valor de uma solução ótima)

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Podemos ter algoritmos que (1) encontram soluções ótimas (2) em tempo polinomial (3) para qualquer instância.

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Podemos ter algoritmos que (1) encontram soluções ótimas (2) em tempo polinomial (3) para qualquer instância.

Abrimos mão de (3) quando procuramos casos particulares do problema que conseguimos resolver eficientemente.

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Podemos ter algoritmos que (1) encontram soluções ótimas (2) em tempo polinomial (3) para qualquer instância.

Abrimos mão de (3) quando procuramos casos particulares do problema que conseguimos resolver eficientemente.

Em programação inteira por exemplo abre-se mão de (2).

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Podemos ter algoritmos que (1) encontram soluções ótimas (2) em tempo polinomial (3) para qualquer instância.

Abrimos mão de (3) quando procuramos casos particulares do problema que conseguimos resolver eficientemente.

Em programação inteira por exemplo abre-se mão de (2).

Em algoritmos de aproximação, abrimos mão de (1):

Como lidar com problemas NP-difíceis?

Um dito em engenharia:

"Rápido. Barato. Confiável. Escolha dois."

Podemos ter algoritmos que (1) encontram soluções ótimas (2) em tempo polinomial (3) para qualquer instância.

Abrimos mão de (3) quando procuramos casos particulares do problema que conseguimos resolver eficientemente.

Em programação inteira por exemplo abre-se mão de (2).

Em algoritmos de aproximação, abrimos mão de (1):

buscamos *boas* soluções que
possam ser obtidas eficientemente.

Algoritmos de aproximação

Algoritmo A é de aproximação se existe $\alpha > 0$ tal que

$$\mathrm{val}(A(I)) \leq \alpha \cdot \mathrm{opt}\,(I)$$

para toda instância I do problema (de minimização).

Algoritmos de aproximação

Algoritmo A é de aproximação se existe $\alpha > 0$ tal que

$$\mathrm{val}(A(I)) \leq \alpha \cdot \mathrm{opt}\,(I)$$

para toda instância I do problema (de minimização).

A é uma α-aproximação e

α é a razão de aproximação (ou garantia de performance).

Algoritmos de aproximação

Algoritmo A é de aproximação se existe $\alpha > 0$ tal que

$$\mathrm{val}(A(I)) \leq \alpha \cdot \mathrm{opt}\,(I)$$

para toda instância I do problema (de minimização).

A é uma α-aproximação e

α é a razão de aproximação (ou garantia de performance).

$\alpha > 1$ para problemas de minimização

Algoritmos de aproximação

Algoritmo A é de aproximação se existe $\alpha > 0$ tal que

$$\mathrm{val}(A(I)) \leq \alpha \cdot \mathrm{opt}\,(I)$$

para toda instância I do problema (de minimização).

A é uma α-aproximação e

α é a razão de aproximação (ou garantia de performance).

$\alpha > 1$ para problemas de minimização

Para problemas de maximização,
a desigualdade é invertida e $\alpha < 1$.

Algoritmos de aproximação

Algoritmo A é de aproximação se existe $\alpha > 0$ tal que

$$\mathrm{val}(A(I)) \leq \alpha \cdot \mathrm{opt}\,(I)$$

para toda instância I do problema (de minimização).

A é uma α-aproximação e

α é a razão de aproximação (ou garantia de performance).

$\alpha > 1$ para problemas de minimização

Para problemas de maximização,
a desigualdade é invertida e $\alpha < 1$.

Objetivo: α tão perto de 1 quanto possível

Esquema de aproximação polinomial

PTAS: Família de algoritmos $\{A_\epsilon\}$ para todo $\epsilon > 0$
onde A_ϵ é uma $(1 \pm \epsilon)$-aproximação para o problema.

Esquema de aproximação polinomial

PTAS: Família de algoritmos $\{A_\epsilon\}$ para todo $\epsilon > 0$
onde A_ϵ é uma $(1 \pm \epsilon)$-aproximação para o problema.

Existe PTAS para o Problema da Mochila e para o
TSP Euclideano (pontos no plano e distância euclideana).

Esquema de aproximação polinomial

PTAS: Família de algoritmos $\{A_\epsilon\}$ para todo $\epsilon > 0$
onde A_ϵ é uma $(1 \pm \epsilon)$-aproximação para o problema.

Existe PTAS para o Problema da Mochila e para o
TSP Euclideano (pontos no plano e distância euclideana).

MAXSNP: classe de problemas, contendo vários
problemas importantes, para a qual vale o seguinte:

Teorema: Para todo problema em MAXSNP,
não existe PTAS a menos que P $=$ NP.

Esquema de aproximação polinomial

PTAS: Família de algoritmos $\{A_\epsilon\}$ para todo $\epsilon > 0$
onde A_ϵ é uma $(1 \pm \epsilon)$-aproximação para o problema.

Existe PTAS para o Problema da Mochila e para o
TSP Euclideano (pontos no plano e distância euclideana).

MAXSNP: classe de problemas, contendo vários
problemas importantes, para a qual vale o seguinte:

Teorema: Para todo problema em MAXSNP,
não existe PTAS a menos que P $=$ NP.

Existem problemas ainda mais difíceis...

Problema do clique máximo

G: um grafo

S: um conjunto de vértices de G

S é um clique se os vértices de S são 2-a-2 adjacentes.

Problema do clique máximo

G: um grafo

S: um conjunto de vértices de G

S é um clique se os vértices de S são 2-a-2 adjacentes.

Problema do clique máximo: Dado um grafo G,
encontrar um clique de tamanho máximo em G.

Problema do clique máximo

G: um grafo

S: um conjunto de vértices de G

S é um clique se os vértices de S são 2-a-2 adjacentes.

Problema do clique máximo: Dado um grafo G,
encontrar um clique de tamanho máximo em G.

Teorema: A menos que P $=$ NP, não existe
$\mathrm{O}(n^{\epsilon-1})$-aproximação para o problema do clique máximo,
onde n é o número de vértices do grafo
e $\epsilon > 0$ é uma constante qualquer.

Problema do clique máximo

G: um grafo

S: um conjunto de vértices de G

S é um clique se os vértices de S são 2-a-2 adjacentes.

Problema do clique máximo: Dado um grafo G,
encontrar um clique de tamanho máximo em G.

Teorema: A menos que P $=$ NP, não existe
$\mathrm{O}(n^{\epsilon-1})$-aproximação para o problema do clique máximo,
onde n é o número de vértices do grafo
e $\epsilon > 0$ é uma constante qualquer.

Note que há uma $\frac{1}{n}$-aproximação trivial para o problema.

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Como (P) pode ser resolvido em tempo polinomial,
o consumo de tempo do algoritmo é polinomial.

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Como (P) pode ser resolvido em tempo polinomial, o consumo de tempo do algoritmo é polinomial.

Mas I é mesmo uma cobertura?

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Como (P) pode ser resolvido em tempo polinomial, o consumo de tempo do algoritmo é polinomial.

Mas I é mesmo uma cobertura?

Como $\sum_{j:e\in S_j} x^*_j \geq 1$ tem exatamente $f_e \leq f$ termos, um deles tem que valer pelo menos $1/f$.

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Mas I é mesmo uma cobertura?

Como $\sum_{j:e\in S_j} x^*_j \geq 1$ tem exatamente $f_e \leq f$ termos, um deles tem que valer pelo menos $1/f$.

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Mas I é mesmo uma cobertura?

Como $\sum_{j:e\in S_j} x^*_j \geq 1$ tem exatamente $f_e \leq f$ termos, um deles tem que valer pelo menos $1/f$.

Então I é uma cobertura.

Arredondamento determinístico

ARREDDET $(E, \mathcal{S}, c)$ $\quad \triangleright\ \mathcal{S} = \{S_1, \ldots, S_m\}$

1 $x^* \leftarrow$ solução da relaxação linear (P)
2 **para** cada e em E **faça**
3 $\quad f_e \leftarrow |\{j : e \in S_j\}|$
4 $f \leftarrow \max\{f_e : e \in E\}$
5 $I \leftarrow \{j : x^*_j \geq 1/f\}$
6 **devolva** I

Mas I é mesmo uma cobertura?

Como $\sum_{j:e\in S_j} x^*_j \geq 1$ tem exatamente $f_e \leq f$ termos, um deles tem que valer pelo menos $1/f$.

Então I é uma cobertura.

ARREDDET é uma f-aproximação. Prova feita na aula.