Algoritmos de aproximação

Marina Andretta

ICMC-USP

11 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

Problemas de otimização têm o objetivo de encontrar um ponto ótimo (mínimo ou máximo) de uma função definida sobre um certo domínio.

Os problemas de otimização combinatória são aqueles que têm domínio finito.

Apesar de os elementos do domínio geralmente poderem ser facilmente enumerados, testar todos os elementos na busca pelo melhor mostra-se inviável na prática, pois o domínio é tipicamente muito grande.

Introdução

O desenvolvimento de algoritmos de aproximação surgiu em resposta à dificuldade computacional de muitos dos problemas de otimização combinatória, que são *NP*-difíceis.

Nestes casos, pode valer a pena trocar a otimalidade por uma aproximação de boa qualidade que possa ser eficientemente calculada.

Esse compromisso entre perda de otimalidade e ganho em eficiência é o paradigma dos algoritmos de aproximação.

Introdução

É importante notar que um algoritmo de aproximação não é simplesmente uma heurística: ele garante encontrar, eficientemente, um elemento do domínio cujo valor tem uma relação pré-estabelecida com o valor ótimo.

É importante mencionar também o aparecimento de certos resultados negativos de aproximabilidade: para alguns problemas, aproximar é tão difícil quanto resolver.

Em termos mais técnicos, alguns problemas não admitem algoritmos de aproximação com razão melhor que um certo limiar, a menos que $P = NP$.

Problemas de otimização

Um problema de otimização tem três ingredientes principais:

- um conjunto de instâncias;
- um conjunto $Sol(I)$ de soluções viáveis (ou factíveis) para cada instância I;
- uma função que atribui um número $val(S)$ (chamado valor de S) a cada solução viável S.

Problemas de otimização

Quando o conjunto $Sol(I)$ das soluções viáveis associado a uma instância I é vazio, dizemos que a instância é inviável (ou infactível).

Um problema de minimização está interessado nas soluções viáveis de valor mínimo, enquanto um problema de maximização está interessado nas soluções viáveis de valor máximo.

Quando uma dessas alternativas (mínimo ou máximo) está subentendida, dizemos simplesmente valor ótimo e problema de otimização.

Exemplo

Problema: encontrar um circuito hamiltoniano de custo mínimo em um grafo com custos nas arestas.

Uma instância deste problema consiste em um grafo G e uma função c que associa um número não-negativo a cada aresta de G.

O conjunto das soluções viáveis de uma instância (G, c) é o conjunto de todos os circuitos hamiltonianos de G.

O valor de um circuito hamiltoniano C é $val(C) := \sum_{e \in C} c_e$.

Problemas de otimização

Uma solução viável cujo valor é ótimo é chamada solução ótima.

O valor de qualquer das soluções ótimas de uma instância I será denotado por $opt(I)$.

Portanto,

$$opt(I) := val(S^*),$$

onde S^* é uma solução ótima de I.

Algoritmos de aproximação

Considere um problema de otimização em que $val(S) \geq 0$ para toda solução viável S de qualquer instância do problema.

Seja A um algoritmo que, para toda instância viável I do problema, devolve uma solução viável $A(I)$ de I.

Se o problema é de minimização e

$$val(A(I)) \leq \alpha opt(I)$$

para toda instância I, dizemos que A é uma α-aproximação para o problema.

Algoritmos de aproximação

Dizemos que α é uma razão de aproximação do algoritmo.

É claro que $\alpha \geq 1$, uma vez que o problema é de minimização.

No caso de problema de maximização, basta refazer a definição com

$$val(A(I)) \geq \alpha opt(I).$$

É claro que, neste caso, $0 < \alpha < 1$.

Algoritmos de aproximação

Uma 1-aproximação para um problema de otimização é um algoritmo exato para o problema.

Observe que um algoritmo A é uma α-aproximação para um problema de minimização (maximização) se α é uma delimitação superior (inferior) para a razão entre $val(A(I))$ e $opt(I)$ para uma instância arbitrária I do problema.

Como o valor de $opt(I)$ é, em geral, tão difícil de calcular quanto uma solução ótima do problema, para demonstrar que um algoritmo é uma α-aproximação é essencial ter boas delimitações para o valor de $opt(I)$.

Notação básica

Dada uma função c que associa um número a cada elemento e de um conjunto finito E, denotamos por $c(F)$ a soma dos valores de c nos elementos de $F \subseteq E$:

$$c(F) := \sum_{f \in F} c_f .$$

Problema de Escalonamento

Um problema bastante conhecido nos contextos de produção industrial e de sistemas operacionais é o de escalonamento (*scheduling*) de tarefas em máquinas.

Estamos interessados em uma das versões mais simples do problema: dadas m máquinas idênticas e n tarefas com tempos de execução pré-determinados, encontrar uma atribuição das tarefas às máquinas que minimize o tempo máximo de operação de qualquer uma das máquinas (*makespan*).

Problema de Escalonamento

Formalmente, o problema do escalonamento em máquinas idênticas (*multiprocessor scheduling problem*) consiste no seguinte:

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

Problema de Escalonamento

De acordo com nossa notação, $t(M_j) := \sum_{i \in M_j} t_i$.

Dizemos que uma partição de $\{1, ..., n\}$ em m blocos é um escalonamento e que o número $\max_j t(M_j)$ é o custo do escalonamento.

Com essa terminologia, o problema pode ser formulado como: dados m, n e t, encontrar um escalonamento de custo mínimo.

Problema de Escalonamento

Este problema é *NP*-difícil mesmo para duas máquinas, ou seja, quando $m = 2$.

O primeiro algoritmo de aproximação para o problema foi descrito e analisado por Graham e usa um critério muito simples: alocar as tarefas uma a uma, destinando cada tarefa à máquina menos ocupada.

Por esse critério, a escolha da máquina que vai receber determinada tarefa não depende dos tempos das tarefas que ainda não foram atribuídas a nenhuma máquina.

Algoritmo de aproximação para Escalonamento

Algoritmo Escalonamento-Graham(m, n, t):

1 para j de 1 a m, faça $M_j \leftarrow \emptyset$;

2 para i de 1 a n, faça

3 seja k uma máquina tal que $t(M_k)$ é mínimo;

4 faça $M_k \leftarrow M_k \cup \{i\}$;

5 devolva $\{M_1, ..., M_m\}$.

Algoritmo de aprox. para ESCALONAMENTO - exemplo

Considere uma instância do problema ESCALONAMENTO com

- 3 máquinas ($m = 3$),
- 7 tarefas ($n = 7$) e
- t_i, $i = 1, ..., 7$, dados por

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6

Algoritmo de aprox. para ESCALONAMENTO - exemplo

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6

M_1

M_2

M_3

Algoritmo de aprox. para ESCALONAMENTO - exemplo

$\vee$

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6

M_1 t_1

M_2

M_3

Algoritmo de aprox. para ESCALONAMENTO - exemplo

$\vee$

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6

M_1 t_1

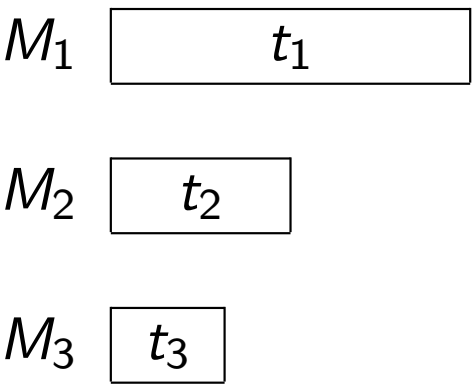
M_2 t_2

M_3

Algoritmo de aprox. para ESCALONAMENTO - exemplo

$\vee$

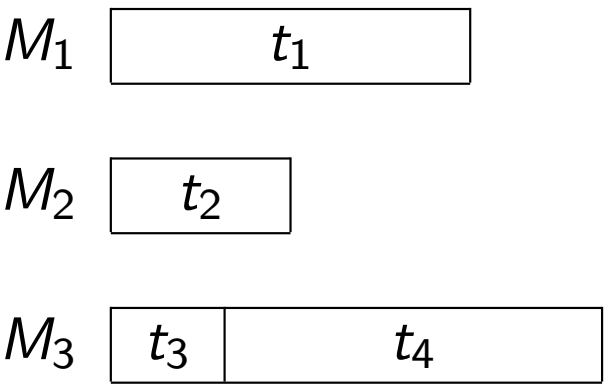
t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



Algoritmo de aprox. para ESCALONAMENTO - exemplo

$\vee$

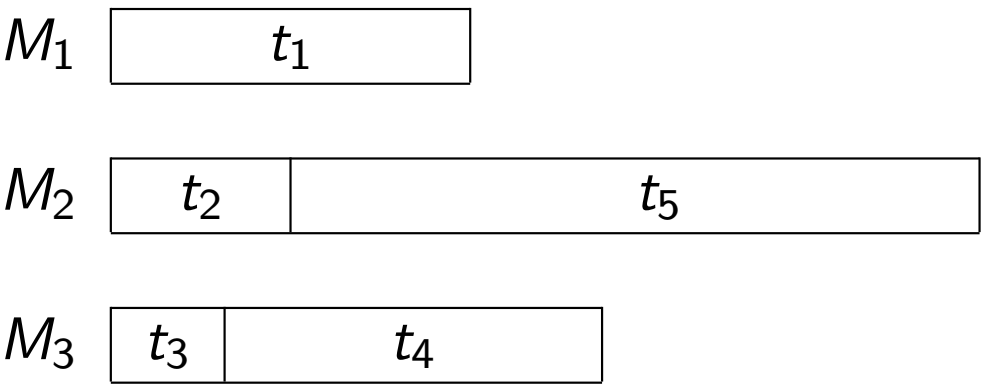
t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



Algoritmo de aprox. para ESCALONAMENTO - exemplo

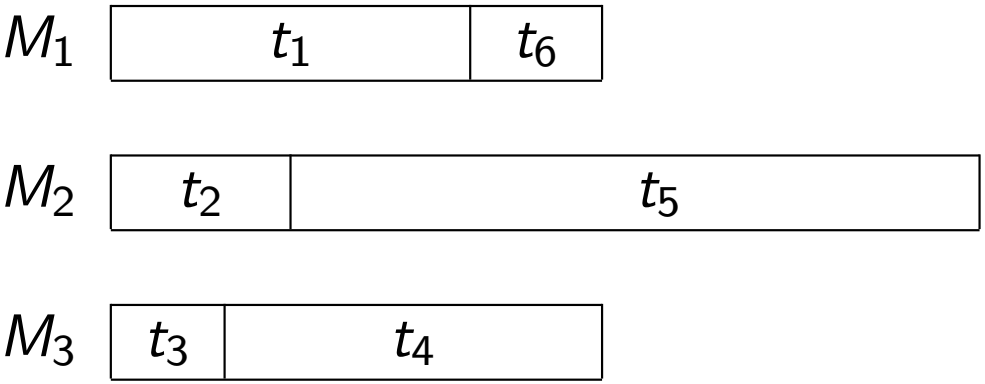
$\vee$

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



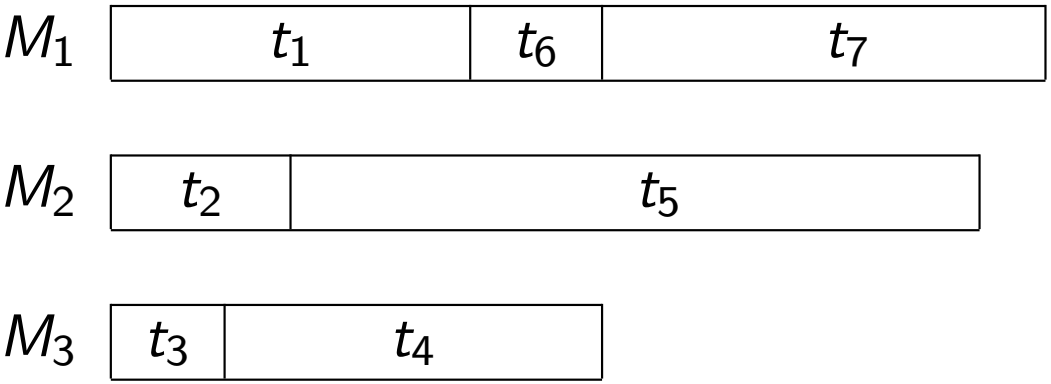
Algoritmo de aprox. para ESCALONAMENTO - exemplo

					$\vee$	
t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



Algoritmo de aprox. para ESCALONAMENTO - exemplo

						$\vee$
t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



Algoritmo de aproximação para Escalonamento

Claramente, ao final do algoritmo Escalonamento-Graham, $\{M_1, ..., M_m\}$ é uma partição de $\{1, ..., n\}$, ou seja, um escalonamento.

Há duas delimitações simples para o custo $opt(m, n, t)$ de um escalonamento ótimo: o tempo da tarefa mais longa e o custo que obteríamos se pudéssemos distribuir as tarefas por igual entre as m máquinas.

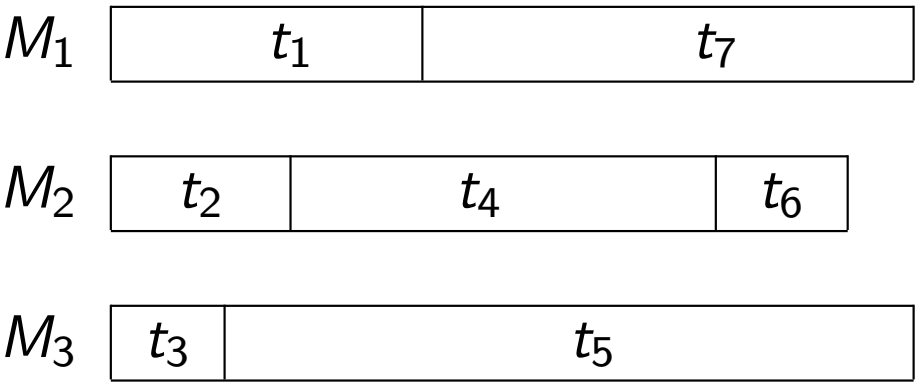
Assim, temos

$$opt(m, n, t) \geq \max_i t_i \quad \text{e} \quad opt(m, n, t) \geq \frac{1}{m} \sum_{i=1}^{n} t_i.$$

Outro escalonamento para o exemplo

Na instância do exemplo, um possível escalonamento é

t_1	t_2	t_3	t_4	t_5	t_6	t_7
4	2	1	5	9	2	6



Outro escalonamento para o exemplo

Como

$$\frac{1}{m}\sum_{i=1}^{n} t_i = \frac{1}{3}(4+2+1+5+9+2+6) = \frac{29}{3} = 9.666...,$$

e o escalonamento obtido tem valor 10, sabemos que ele é ótimo.

Algoritmo de aproximação para Escalonamento

Teorema: O algoritmo Escalonamento-Graham é uma 2-aproximação polinomial para o Escalonamento(m, n, t), quando $n \geq m$.

Algoritmo de aproximação para Escalonamento

Demonstração: claramente, o algoritmo consome tempo polinomial em n, já que $m \leq n$. Ou seja, ele é um algoritmo polinomial.

Vamos mostrar agora que ele é uma 2-aproximação polinomial para o Escalonamento(m, n, t).

Seja τ o valor de $t(M_k)$ imediatamente antes da execução da linha 4 do algoritmo em uma iteração qualquer.

É claro que $\tau \leq t(M_j)$ para todo j.

Algoritmo de aproximação para ESCALONAMENTO

Assim,

$$\tau \leq \frac{1}{m} \sum_{j=1}^{m} t(M_j) \leq \frac{1}{m} \sum_{i=1}^{n} t_i.$$

Como $\frac{1}{m} \sum_{i=1}^{n} t_i \leq opt(m, n, t)$, segue que

$$\tau \leq opt(m, n, t).$$

Algoritmo de aproximação para ESCALONAMENTO

Assim, imediatamente depois da execução da linha 4 do algoritmo, temos

$$t(M_k) = \tau + t_i.$$

Como $\max_i t_i \leq opt(m, n, t)$, temos

$$t(M_k) \leq 2opt(m, n, t).$$

Algoritmo de aproximação para Escalonamento

Esta delimitação vale no fim de cada iteração, ou seja, ela vale cada vez que uma tarefa i é atribuida a uma máquina k. Portanto, a delimitação se aplica a cada uma das máquinas.

Logo, no fim da última iteração,

$$\max_j t(M_j) \leq 2opt(m, n, t).$$

Portanto, o algoritmo Escalonamento-Graham é uma 2-aproximação polinomial para o Escalonamento(m, n, t), quando $n \geq m$.

Algoritmo de aproximação para Escalonamento

Como visto, o algoritmo processa os dados sem conhecimento prévio da sua totalidade.

Várias situações reais demandam esse tipo de abordagem, como o escalonamento de tarefas em processadores, que precisa de algoritmos rápidos e que forneça soluções viáveis cujo valor seja próximo do valor ótimo (como acontece com o algoritmo Escalonamento-Graham).

Esse foi o primeiro algoritmo de aproximação de que se tem notícia, o que o torna um marco nessa teoria.