Algoritmos Aproximativos

Consideraremos quatro problemas:
Escalonamento de tarefas,
Cobertura de vértices,
Caixeiro viajante,
Mochila.

São quatro Problemas de Otimização Combinatória, onde queremos encontrar um ótimo (máximo ou mínimo) de uma função definida sobre um domínio finito.

Problema TSP (G, p): Dados um grafo G e um custo positivo para cada aresta, determinar uma rota de peso mínimo que passe exatamente uma vez por cada vértice do grafo.

Um problema de otimização tem:
o conjunto de instâncias,
para cada instância I temos o conjunto $Sol(I)$ de soluções viáveis,
cada solução viável $S \in Sol(I)$ tem um custo $val(S)$, o valor de S.

Cada instância I possui um valor ótimo: o valor máximo para um problema de maximização, o valor mínimo para um problema de minimização, $opt(I)$.
Uma solução viável de valor ótimo é uma solução ótima S^*.
$opt(I) = val(S^*)$

Para o problema do caixeiro viajante:
- uma instância $I = (\text{Grafo completo}, p)$
- o conjunto de soluções viáveis de I
  $Sol(I) = \{ C : \text{circuito hamiltoniano} \}$
- Cada circuito $C \in Sol(I)$ tem um valor $val(C) = \sum_{e \in C} p(e)$
- buscamos $opt(I) = val(C^*)$, o valor do circuito ótimo C^*, de menor soma dos pesos.

Seja A um algoritmo que retorna uma solução viável $A(I)$ para o instância I.

A é α-aproximativo para um problema de minimização se

$$val(A(I)) \leq \alpha \, opt(I), \quad \alpha \geq 1$$

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

$$\frac{val(A(I))}{opt(I)} \leq \alpha.$$

Descreveremos uma 2-aproximação para o TSP.

Para comparar $val(A(I))$ com o desconhecido e difícil de calcular $opt(I)$ procuramos limites inferiores "bons".

Escalonamento de tarefas em máquinas idênticas

Multiprocessor scheduling problem GT[558]

Dadas m máquinas $M_1, \ldots, M_m$ e n tarefas com tempos $t_1, \ldots, t_n$, encontrar a atribuição que minimize o máximo tempo de operação de qualquer uma das máquinas.

tempo de operação: $t(M_j) = \sum_{i \in M_j} t_i$

makespan: $\max_j t(M_j)$

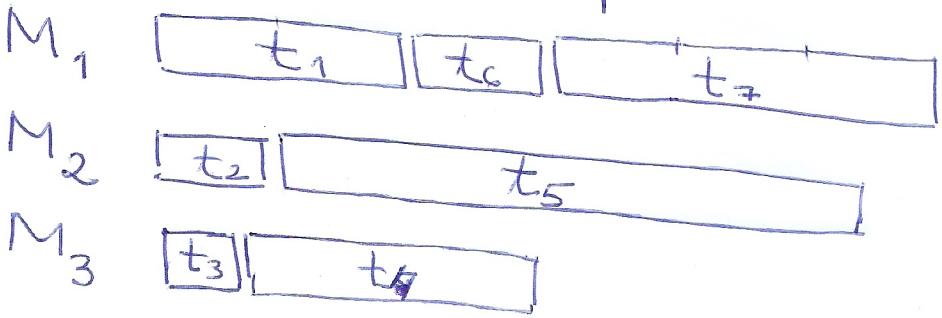
Escalonamento (m, n, t): Encontrar uma partição $\{M_1, \ldots, M_m\}$ de $\{1, \ldots, n\}$ que minimize $\max_j t(M_j)$.

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 com $t(M_k)$ mínimo
4. $M_k \leftarrow M_k \cup \{i\}$
5. retorne $\{M_1, \ldots, M_m\}$.

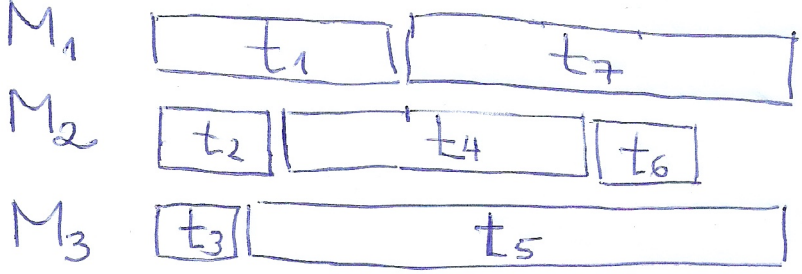
Exemplo com 3 máquinas e 7 tarefas, onde $t_1 = 4$, $t_2 = 2$, $t_3 = 1$, $t_4 = 5$, $t_5 = 9$, $t_6 = 2$ e $t_7 = 6$.

Escalonamento S_1



$t(M_1) = 12$
$= val(A(I))$

Escalonamento S_2



$t(M_1) = 10$
$= val(S_2)$
$= opt(I)$

Dois limites inferiores:

$$opt(m,n,t) \geq \max_i t_i$$

o tempo da tarefa mais longa

$$opt(m,n,t) \geq \frac{1}{m} \sum_{i=1}^{n} t_i$$

O mínimo é pelo menos o custo da média, o mínimo makespan ganha da distribuição por igual.

No exemplo, $\max_i t_i = 9$, e

$$\frac{1}{m} \sum_{i=1}^{n} t_i = \frac{29}{3}.$$

O custo da distribuição por igual já prova que $val(S_2) = 10 = opt(I)$.

Usaremos os limites inferiores para provar a razão de aproximação.

Teorema: O algoritmo Escalonamento é uma 2-aproximação polinomial para o problema Escalonamento(m,n,t).

prova: Seja x o valor de $t(M_k)$ imediatamente antes da linha 4.

Para toda máquina, $x \leq t(M_j) = \sum_{i \in M_j} t_i$

Portanto,

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

se x é menor que o tempo de execução de cada máquina, então x é menor que a média.

Logo, $x \leq \frac{1}{m} \sum_{i=1}^{n} t_i \leq opt(m,n,t)$

↑ limite da distribuição igual

Logo, imediatamente após executar 4. $M_k \leftarrow M_k \cup \{i\}$ temos

$t(M_k) = x + t_i \leq 2\, opt(m,n,t)$

↑ limite da tarefa longa

Impondo a barreira no fim da iteração para cada máquina, ao final

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

▨

Quando desenvolvemos um algoritmo aproximativo, ao estabelecer a razão de aproximação é natural perguntar:

- a nossa análise é justa? Podemos estabelecer uma razão menor? Ou existe uma instância I cujo valor retornado pelo algoritmo $val(A(I)) = 2\,opt(I)$?

Considere $n+1 = m^2$ tarefas cada uma com tempo unitário e 1 tarefa com tempo m. O escalonamento obtido pelo algoritmo é $2m = val(A(I))$, mas $opt(I) = m+1$.

- é passível estabelecer um algoritmo com razão menor?

Para o TSP, apresentaremos dois algoritmos aproximativos diferentes.