Universidade Federal de Uberlândia - UFU
Programa de Pós Graduação em Ciência da Computação
Inteligência Artificial

Balanceamento de Carga

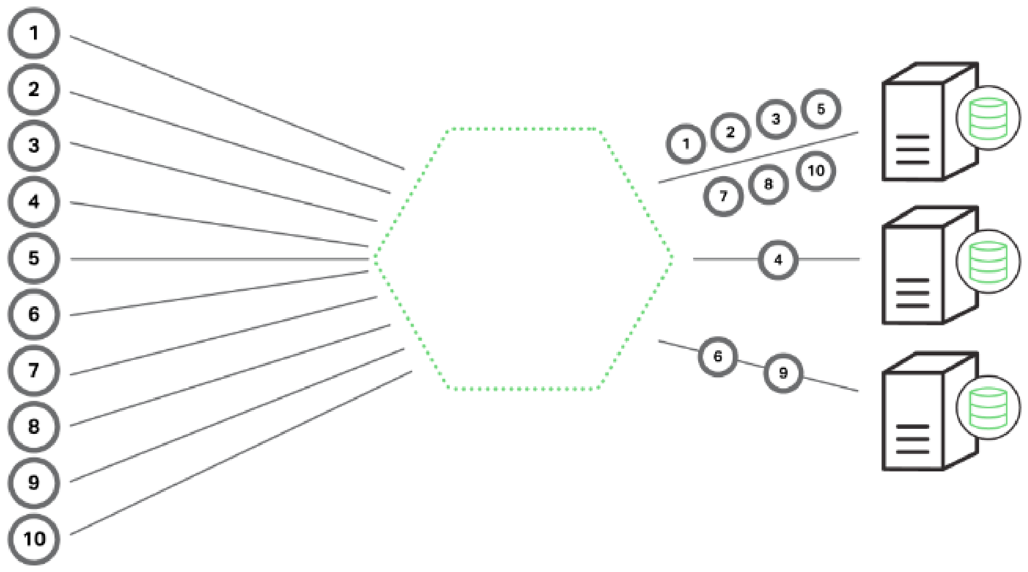
CIBELE MARA FONSECA
MALENA ALVES RUFINO

DISCIPLINA: ANÁLISE DE ALGORITMOS

PROFESSORA DRA.: MÁRCIA APARECIDA FERNANDES

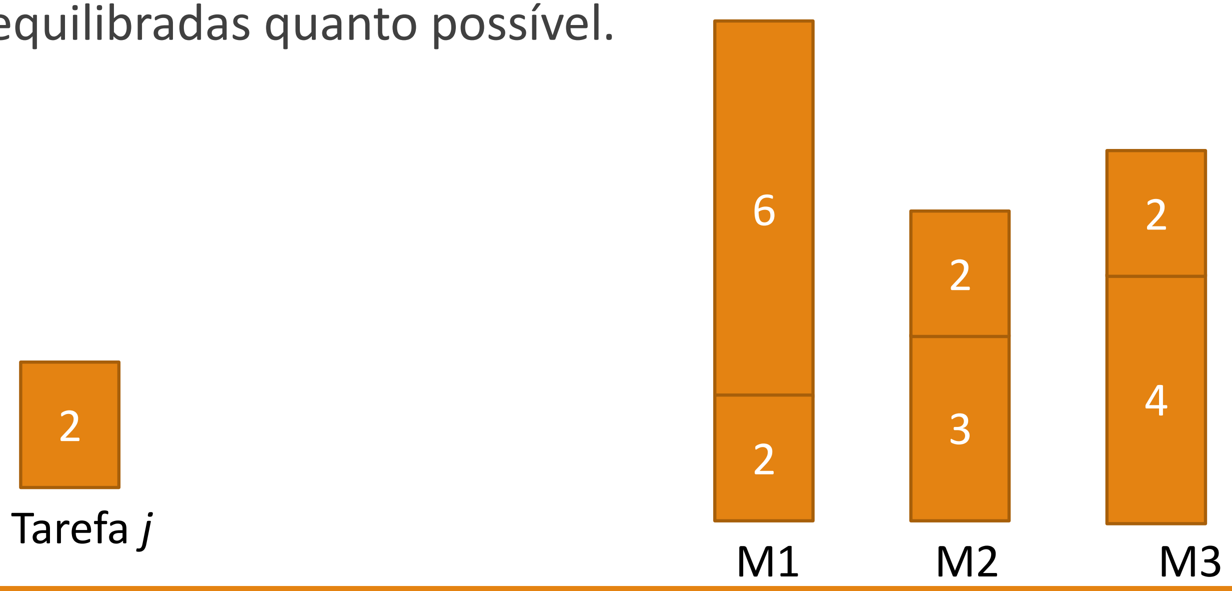
Balanceamento de Carga

Surge quando vários servidores precisam processar um conjunto de tarefas. A versão básica consiste em servidores idênticos e cada um pode ser usado para atender a qualquer uma das tarefas.



Problema

- Um conjunto de máquinas $M_1, \ldots, M_m$;
- Um conjunto de n tarefas;
- Cada tarefa j tem um tempo de processamento t_j;
- Procura-se atribuir cada tarefa a uma das máquinas para que as cargas colocadas em todas as máquinas sejam tão equilibradas quanto possível.



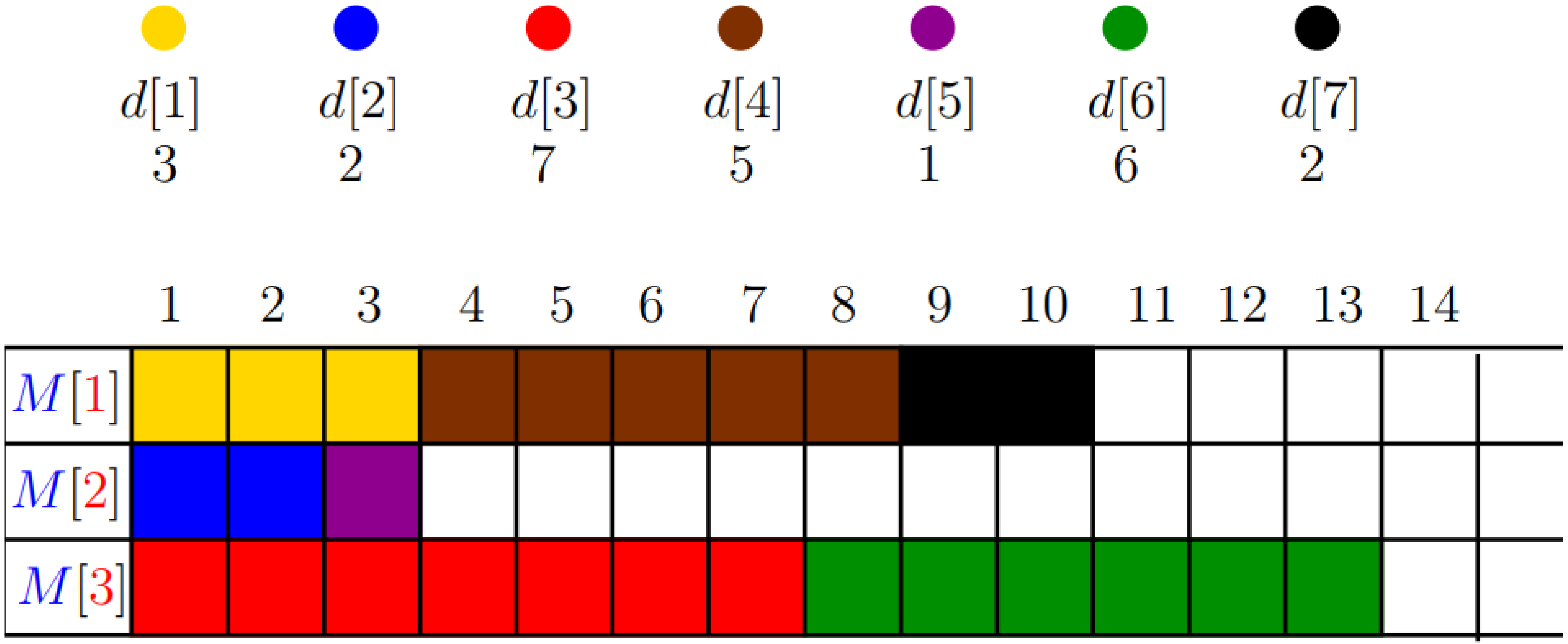
Problema

- *A(i)* é o conjunto de tarefas atribuídas à máquina *Mi*
- A máquina *Mi* trabalha por um tempo total de

$$T_i = \sum_{j \in A(i)} t_j,$$

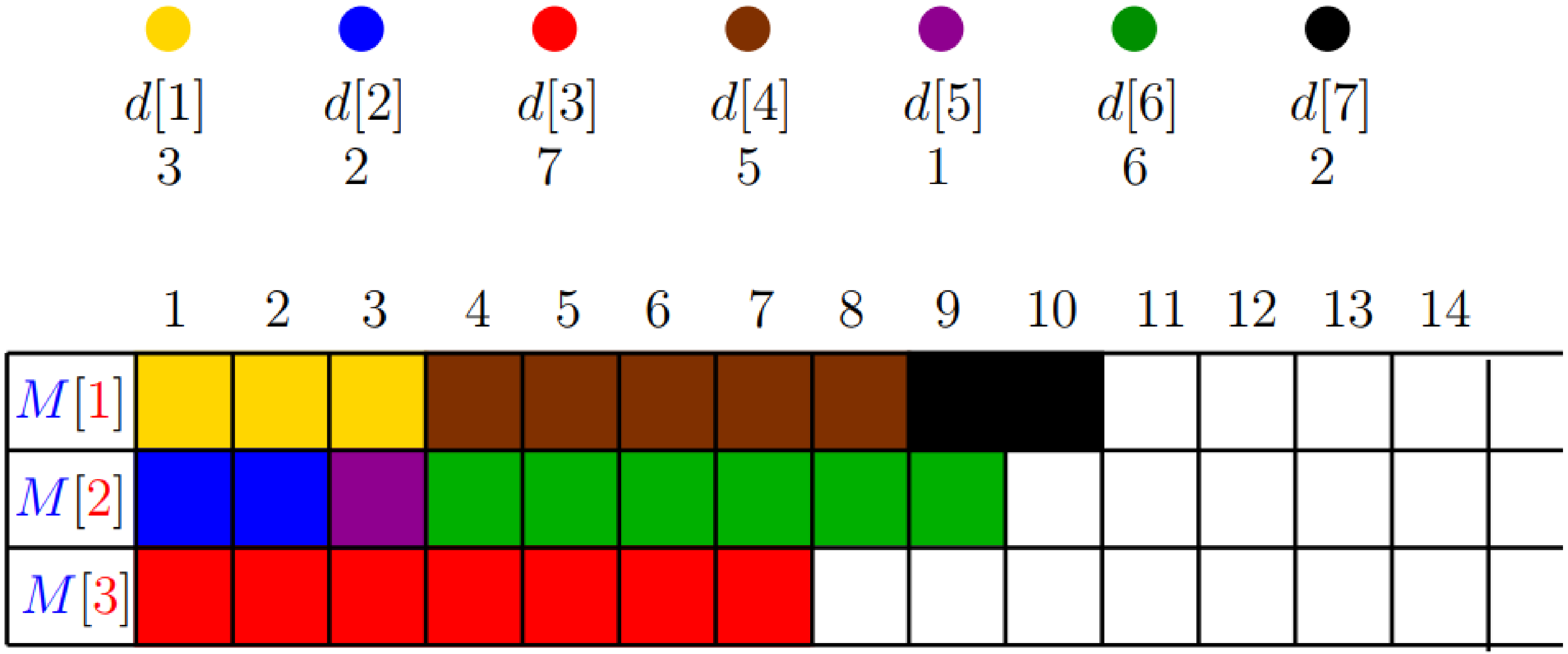
- Esta é a carga na máquina *Mi*
- Procuramos minimizar o *makespan,* que é a carga máxima em qualquer máquina, $T = max_i T_i$
- O problema de escalonamento consiste em encontrar uma atribuição de *makespan* mínimo.

Exemplo



Makespan = 13

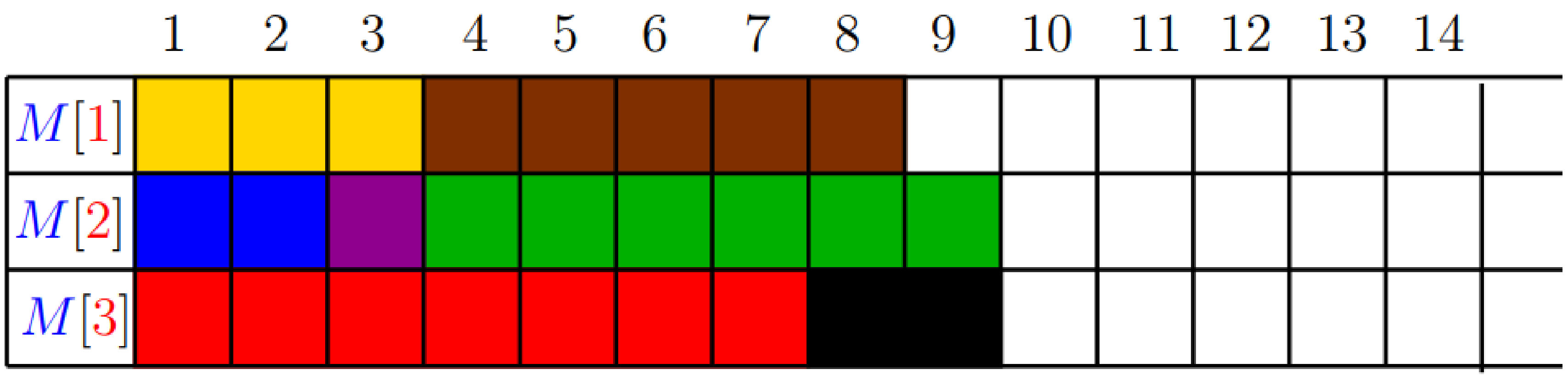
Exemplo



Makespan = 10

Exemplo

$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2



Makespan = 9

Ideia do Algoritmo de Força Bruta

- Testar todas as combinações possíveis;
- Permutação do conjunto de tarefas;
- Distribuição de todas as combinações pelas máquinas.

Algoritmo de Força Bruta

Balanceamento_Forca_Bruta

Comece sem nenhuma tarefa atribuída

Conjunto Ti = 0 e A(i)=Ø para todas as máquinas Mi

Para todas as combinações obtidas com as n tarefas

For j = 1, . . . , n

Mi é alterada a medida que uma tarefa j é colocada na máquina Mi

Atribuir tarefa j para máquina Mi

Conjunto A(i)←A(i) ∪ {j}

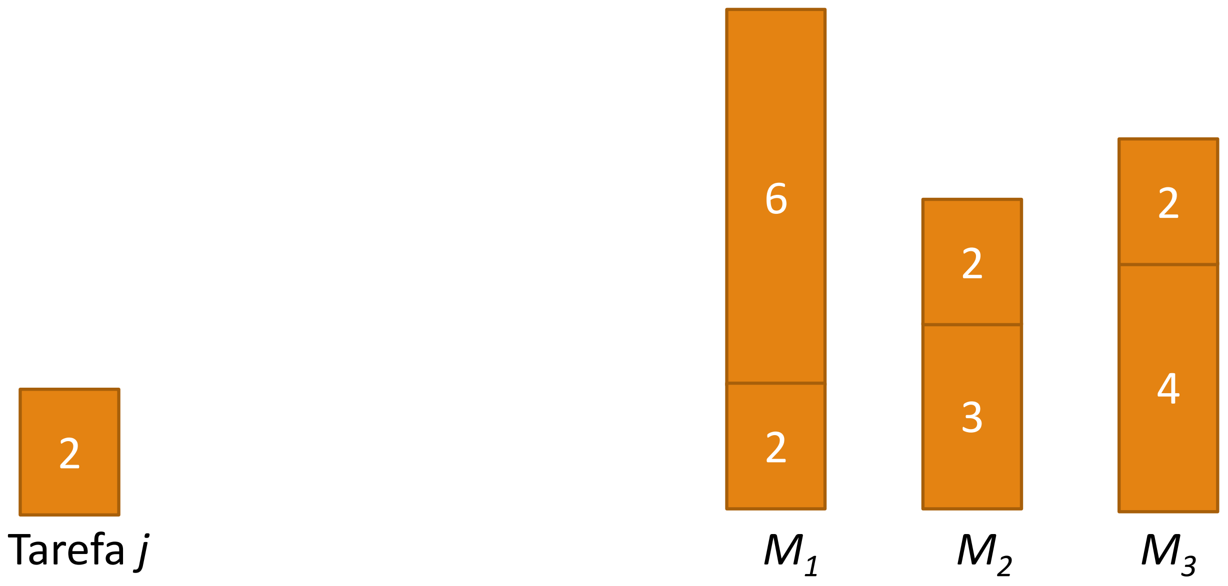
Conjunto Ti←Ti + tj

EndFor

$\boxed{O(n!)}$

Ideia do Algoritmo Aproximado

- Guloso;
- Recebe as tarefas em qualquer ordem;
- Cada tarefa é atribuída à máquina cuja carga é a menor até o momento.



Algoritmo Aproximado (Guloso)

Balanceamento_Guloso:

Comece sem nenhuma tarefa atribuída

Conjunto Ti = 0 e A(i)=Ø para todas as máquinas Mi

For j = 1, . . . , n

    Mi é a máquina com menor carga

    Atribui tarefa j para máquina Mi

    Conjunto A(i)← A(i) ∪ {j}

    Conjunto Ti ←Ti + tj

EndFor

$\boxed{O(nm)}$

Razão de Aproximação

- T é a solução obtida pelo algoritmo aproximado
- T* é a solução ótima
- Precisamos comparar a nossa solução com o valor ideal T*

(embora não saibamos qual é esse valor e não temos esperança de computá-lo)

- Precisaremos de um limite inferior ótimo.

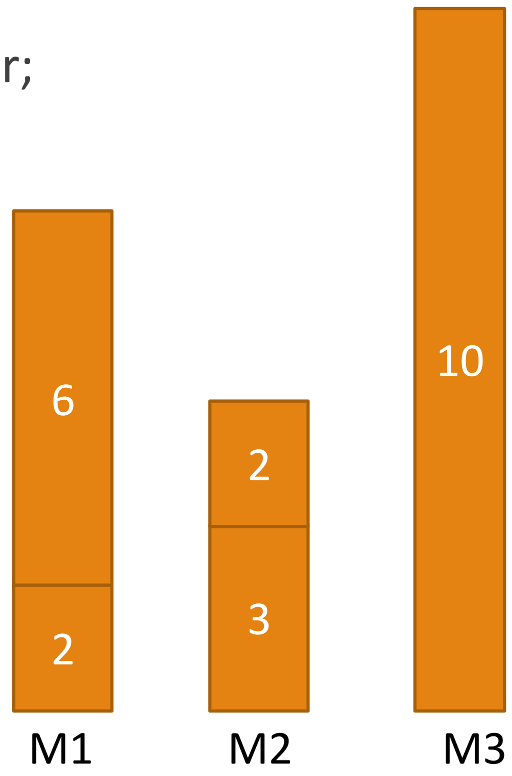
Razão de Aproximação

- Existem muitos limites inferiores possíveis no ótimo;
- Uma ideia para um limite inferior é baseada em considerar o tempo total de processamento $\Sigma_j\ t_j$;
- Uma das máquinas m deve fazer pelo menos uma fração de 1/m;
- O *makespan* será pelo menos

$$T^* \geq \frac{1}{m} \sum_j t_j$$

Razão de Aproximação

- Suponha que tenhamos uma tarefa extremamente longa em relação à soma de todos os tempos de processamento;
- A solução ideal colocaria essa tarefa em uma máquina e será a última a terminar;
- Nesse caso, o algoritmo guloso realmente produziria a solução ótima;
- Porém o limite inferior anterior não é forte o suficiente para estabelecer isso.



Razão de Aproximação

- Esta situação sugere outro limite inferior adicional em T*;
- O makespan ótimo será pelo menos o tempo da maior tarefa, ou seja:

$$T^* \geq \max_j t_j$$

Razão de Aproximação

- Assim, a solução gulosa aproximada produz um makespan

$$T \leq 2T^*$$

Prova

- Sabemos que o limite inferior de T* é

$$T^* \geq \frac{1}{m} \sum_j t_j \qquad \text{ou} \qquad T^* \geq \max_j t_j$$

- Consideramos que uma máquina *Mi* atinja a maior carga
- Qual foi a última tarefa *j* colocada em *Mi*?
- Se *tj* não for muito maior em relação às outras tarefas, trata-se do primeiro limite inferior
- Se *tj* for muito grande, trata-se do segundo limite inferior

Prova

- Quando atribuímos a tarefa j a máquina Mi, a carga das máquinas era pelo menos Ti-tj
- Para encontrar a carga mínima de todas as máquinas nesta situação, temos que

$$\Sigma_k T_k \geq m(T_i - t_j)$$

- O que é equivalente a

$$T_i - t_j \leq \frac{1}{m} \sum_k T_k$$

- O lado direito da desigualdade é igual ao valor do primeiro limite inferior ótimo, logo

$$Ti\text{-}tj \leq T^*$$

Prova

- Contabilizamos a parte final de Mi, que é a tarefa restante j
- Usamos o segundo limite inferior que diz que $Tj \leq T^*$
- Somando

$$Ti\text{-}tj \leq T^* + \ Tj \leq T^*$$

- Temos

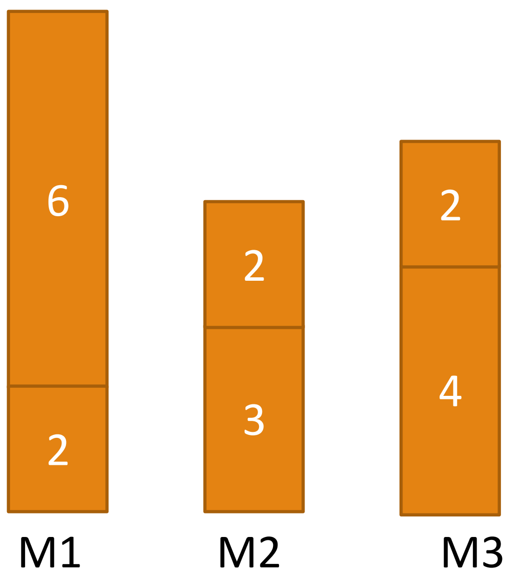
$$Ti \leq 2T^*$$

- Como T=Ti, encontramos a razão 2

$$T \leq 2T^*$$

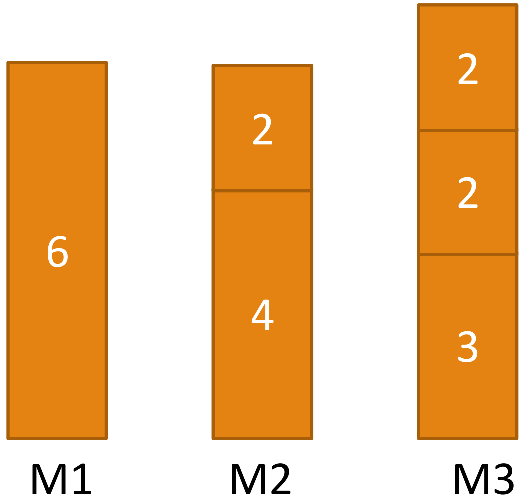
Algoritmo Aproximado (Guloso)

- Não é a solução ideal
- Observando o exemplo onde temos 3 máquinas e uma sequência de 6 tarefas com tempos 2, 3, 4, 6, 2, 2;
- Obtemos makespan 8
- Se a sequência de tempos fosse 6, 4, 3, 2, 2, 2 o makespan seria 7.
- Então aprimoramos o algoritmo colocando as maiores tarefas primeiro.



Ideia do Algoritmo Aproximado Aprimorado

- Cada tarefa entra na máquina com menor carga, assim como o anterior.
- Porém as tarefas são colocadas em ordem decrescente antes de serem distribuídas pelas máquinas.
- Exemplo
  - Ordem de chegada: 6, 4, 3, 2, 2, 2;
  - Distribuição das máquinas.



Algoritmo Aproximado Aprimorado

Balanceamento_Ordenado

Comece sem nenhuma tarefa atribuída

Conjunto Ti = 0 e A(i)=Ø para todas as máquinas Mi

Ordenar tarefas em ordem decrescente de tempos de processamento tj

Assuma t1 ≥ t2 ≥ . . . ≥ tn

For j = 1, . . . , n

    Deixe Mi ser a máquina com menor carga

    Atribuir tarefa j para máquina Mi

    Conjunto A(i)←A(i) ∪ {j}

    Conjunto Ti←Ti + tj

EndFor

$\boxed{O(nm)}$

Razão de Aproximação

- Com esse algoritmo obtemos uma razão de aproximação de 1,5, ou seja

$$T \leq 3/2T^*.$$

Prova

- Se tivermos até m tarefas a solução gulosa é ótima
- Se tivermos mais de m tarefas, temos o seguinte limite inferior ótimo:

$$T^* \geq 2t_{m+1}$$

- Considere os m+1 primeiras tarefas ordenadas
- Tem-se m máquinas e m+1 tarefas, então pelo menos uma máquina terá duas tarefas
- Essa máquina terá pelo menos $2t_{m+1}$

Prova

- A prova é semelhante a do algoritmo anterior
- Sendo Mi a máquina com maior carga
- Se Mi tiver uma única tarefa, então temos o makespan ideal
- Se Mi tem pelos menos duas tarefas e tj é o tempo da última tarefa atribuída à máquina
- j ≥ m+1, já que o algoritmo atribui as primeiras m tarefas a máquinas distintas
- Então

$$t_j \le t_{m+1} \le 1/2T^*$$

- Temos que

$$tj \le 1/2T^*$$

Prova

- Ao somar as duas desigualdades

$$Ti-tj \leq T^* + tj \leq \ 1/2T^*$$

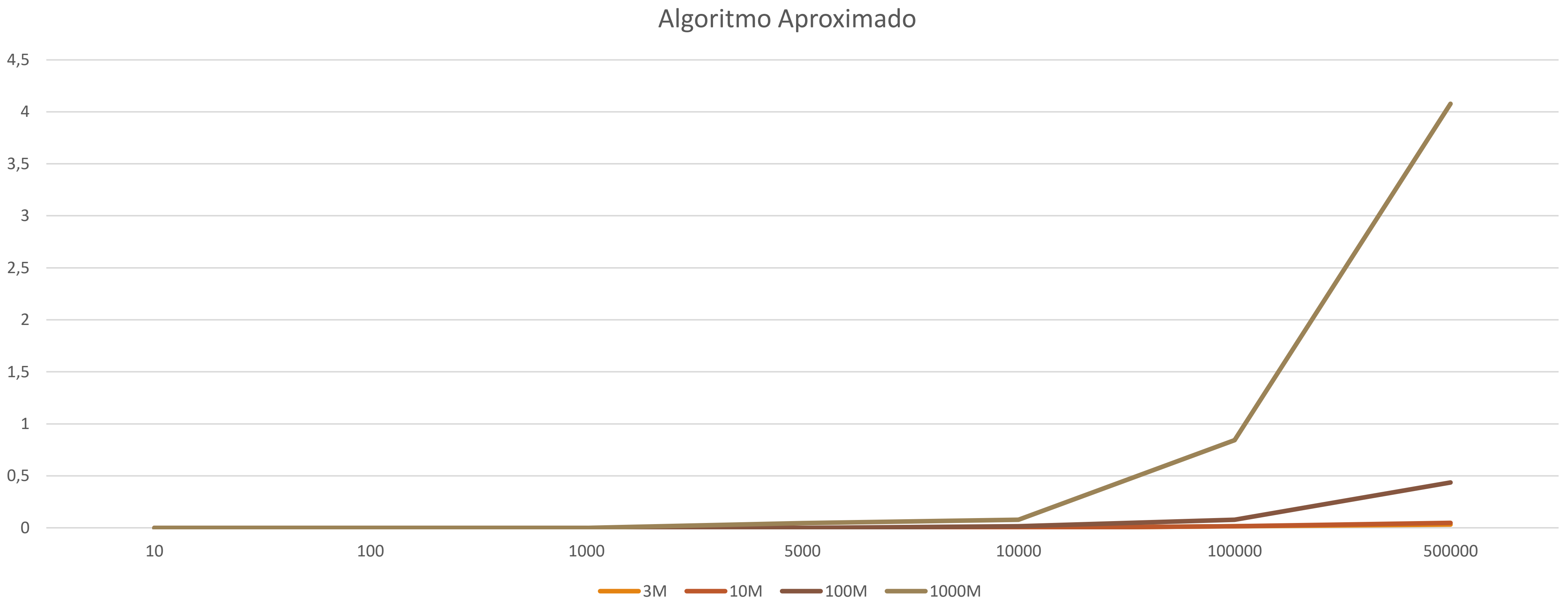
- Temos que:

$$Ti \leq 3/2T^*$$

- Que é igual a:

$$T \leq 3/2T^*$$

Algoritmo Aproximado



Algoritmo Exponencial – Força Bruta

Referências

- KLEINBERG ,Jon and TARDOS , Éva . Algorithm Design. ADDISON-WESLEY,2005
- Aulas de PGC101 - Análise de Algoritmos
- https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pearson/11ApproximationAlgorithms-2x2.pdf

Dúvidas?