



# Reconfiguração Dinâmica de Escalonadores com Restrição de Energia em Tempo-Real

Ríad Mattos Nassiffe  
riad@das.ufsc.br

Programa de Pós-Graduação em Engenharia de Automação e Sistemas  
Universidade Federal de Santa Catarina  
Florianópolis, SC, Brasil

4 de maio de 2011

# Sumário

- 1 Introdução
  - Sistema de Tempo Real
  - Consumo de Energia
  - Alta Variabilidade de Carga
- 2 Modelo Discreto
- 3 Algoritmo
  - Algoritmo de Programação Dinâmica
  - Relaxações
  - Heurística Gulosa Baseada em Densidade
- 4 Conclusão

# Sumário

## 1 Introdução

- Sistema de Tempo Real
- Consumo de Energia
- Alta Variabilidade de Carga

## 2 Modelo Discreto

## 3 Algoritmo

- Algoritmo de Programação Dinâmica
- Relaxações
- Heurística Gulosa Baseada em Densidade

## 4 Conclusão

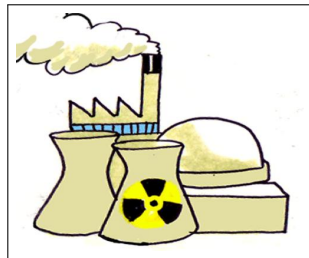
# Sistema de Tempo Real

Uma aplicação de tempo real é aquela que tem uma restrição de tempo (*deadline*) para responder a um estímulo. Aplicações de tempo real podem ser divididas em dois grupos:

- ▶ *Hard Real Time*: não permitem perda de *deadline*.
- ▶ *Soft Real Time*: toleram perda de *deadline*.

# Exemplos de Aplicações

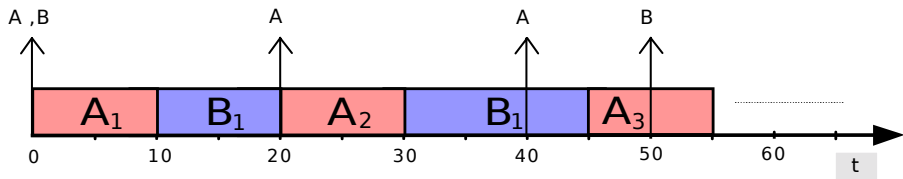
- ▶ Vídeo conferência
- ▶ Jogos *on-line*
- ▶ Programa de troca de mensagem
- ▶ Bolsa de valores
- ▶ Fábricas
- ▶ Aviões
- ▶ Sistemas de segurança
- ▶ Usinas nucleares



# Definição de Sistema Tempo Real

- ▶ Tarefas  $\tau_i$  com instâncias periódicas indexadas por  $i = \{1, \dots, n\}$
- ▶ Tempos de execução  $C_i$
- ▶ Períodos  $T_i$
- ▶ *Deadlines*  $D_i$
- ▶ Utilização  $U_i = C_i / T_i$

# Estratégia de Escalonamento de Tempo Real: *Earliest Deadline First*



## ► *Earliest Deadline First (EDF)*

- Ótimo
- Prioridade Dinâmica
  - Gera interferência entre as tarefas
- Teste de escalonabilidade simples:
  - $\sum_{i=1}^n U_i \leq 1$

|   | $C_i$ | $T_i$ | $U_i$ |
|---|-------|-------|-------|
| A | 10    | 20    | 0,5   |
| B | 25    | 50    | 0,5   |

# Energia

Equipamentos que podem funcionar com bateria ou não, dependendo da disponibilidade de energia.

- ▶ Satélites
- ▶ Sensores
- ▶ Robôs
- ▶ Servidores
- ▶ *etc...*



# Estratégia de Economia de Energia DPM (Dynamic Power Management)

Realiza economia de energia através da regulação da frequência e voltagem do processador

- ▶ Um processador pode operar em um determinado conjunto de frequências:
  - $\mathcal{F} = \{f_1, \dots, f_m\}$  onde  $0 < f \leq 1$  para todo  $f \in \mathcal{F}$ . Para  $\mathcal{F} = \{632, 5Mhz, 1, 265Ghz, 2, 53Ghz\}$  o conjunto normalizado por  $\mathcal{F} = \{0, 25; 0, 5; 1\}$ .
- ▶ A voltagem do processador decresce de forma linear à frequência utilizada
  - $v = Vf_j$

# Estratégia de Economia de Energia DPM (Dynamic Power Management)

Considera outros dispositivos do sistema além da *CPU*.

- ▶ Garante economia energia no sistema
- ▶ Ficando o modelo energético do sistema caracterizado por:

$$P_{sis} = P^S + (P^I + P^D)h$$

onde:

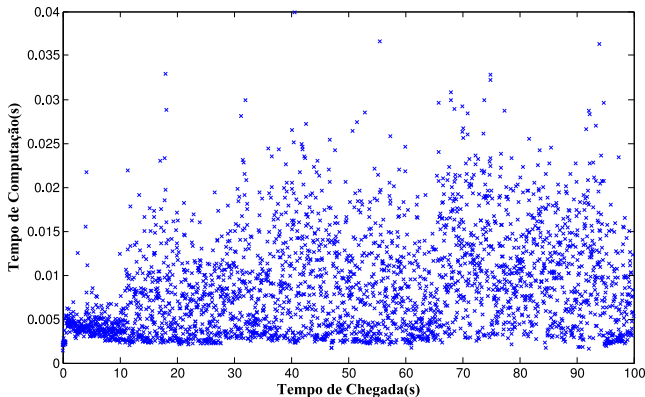
- $P^S$  é a potência mínima para manter os componentes do sistema ligados
- $P^I$  é o somatório das potências independentes da frequência da *CPU*
- $P^D$  é o somatório das potências dependentes da frequência da *CPU*
- $h$  é binário,  $h \in \{0, 1\}$

# Variabilidade de $C_i$ ou $T_i$

Aplicações com alta taxa de variação de carga computacional

- ▶ Aplicações de multimídia
- ▶ Robôs
- ▶ Sistemas de monitoramento
- ▶ *etc...*

# Problema de Alta Variabilidade de Carga



Sistemas com alta variabilidade de  $C_i$  ou  $T_i$ :

- ▶ é complicado determinar o pior caso
- ▶ desperdício de recurso ao garantir o pior caso

# Servidor de Tarefas Aperiódicas

- ▶ *Constant Bandwidth Server* (CBS)
  - Isolamento temporal
  - Fácil implementação
  - Utilizado nas simulações deste trabalho

# Constant Bandwidth Server

- ▶ Assume-se  $n$  servidores  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$
- ▶ Cada servidor  $S_i \in \mathcal{S}$  é definido pela tupla  $(Q_i, T_i)$ 
  - $Q_i$  representa o orçamento máximo
  - $T_i$  representa o período
- ▶ A utilização de cada servidor é dada por  $u_i = Q_i/T_i$
- ▶ Quando o orçamento termina:
  - Deadline do  $S_i$  é adiado por  $T_i$
  - O orçamento  $Q_i$  é recarregado

# Reconfiguração Dinâmica

- ▶ Servidores aperiódicos lidam bem com variabilidade, entretanto:
  - Alocação estática de reservas de banda pode não ser ideal
  - Pode ser necessário redistribuir tempo de processador dinamicamente
    - O que torna necessário a regulação de voltagem e frequência do processador, a cada nova configuração escolhida, para garantir a economia de energia
- ▶ Exemplos onde pode ser necessário reconfigurar o escalonador:
  - Em um sistema de segurança, ao detectar movimento em um ambiente
  - Em uma rede de sensores sem fio, ao remover um sensor
  - Quando um satélite pára de receber a luz do sol
- ▶ Suporte para reconfiguração é necessário em nível de escalonador

# Abordagem de Reconfiguração Dinâmica

- ▶ Reconfiguração baseada em otimização
  - Maximizar o benefício do sistema
  - Sujeito a restrição de escalonabilidade
  - Sujeito a restrição de energia

# Objetivos

- ▶ Realizar grandes ajustes em tempo de execução
  - Permitindo escalonamento dinâmico de tarefas
  - Maximizando o benefício total do sistema
  - Respeitando a restrição de escalonabilidade
  - Respeitando a restrição de consumo de energia

# Abordagem

- ▶ Modelar o sistema como um problema de otimização
  - Resolver o problema em tempo de execução ("rapidamente")
  - Aplicar os novos parâmetros no escalonador

# Notação

- ▶ Assume-se  $n$  servidores  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  do tipo CBS
  - O orçamento  $Q_i$  é expresso com base no tempo de uso da *CPU* e de outros recursos computacionais, que venham a ser utilizados.

$$Q_i = \frac{Q_i^D}{f_j} + Q_i^I$$

- Cada servidor possui uma função benefício  $A_i$ , proporcional à utilização  $u_i$  alocada para si e a frequência utilizada
- A utilização de cada servidor  $S_i$  é calculada por:

$$u_i = \frac{Q_i^I}{T_i} + \frac{Q_i^D}{T_i f_j}$$

# Notação

- ▶ Cada servidor  $S_i$  tem  $\kappa(i)$  modos de execução, que podem executar em  $m$  frequências diferentes
- ▶  $P^*$  é o consumo máximo de energia do sistema

# Notação

- ▶ O uso de *CPU* é limitada pela restrição de escalonabilidade *EDF*:

$$\sum_{S_i \in \mathcal{S}} u_i \leq 1 \quad (1)$$

- ▶ A restrição de energia é dada por:

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} P_i^{k,j} x_i^{k,j} \leq \beta P^* \quad (2)$$

# Sumário

## 1 Introdução

- Sistema de Tempo Real
- Consumo de Energia
- Alta Variabilidade de Carga

## 2 Modelo Discreto

## 3 Algoritmo

- Algoritmo de Programação Dinâmica
- Relaxações
- Heurística Gulosa Baseada em Densidade

## 4 Conclusão

# Descrição do Problema

Considerando um conjunto de tarefas multimodais, com restrições de escalonabilidade e energia, usando a estratégia *CBS* tem-se:

- ▶ Que cada configuração possível para um servidor  $S_i$  faz parte de um conjunto  $\Omega_i$ , sendo  $\Omega_i = \{(k, j) : k \in \kappa(i), j \in \mathcal{F}\}$
- ▶ Cada uma das configurações pertencentes a  $\Omega_i$  estarão ativadas se a variável binária  $x_i^{k,j}$  tiver valor 1
  - $i$  : Indica o servidor
  - $k$  : Modo de operação da tarefa
  - $j$  : Frequência de operação do processador

# Descrição do Problema

- ▶ O benefício é calculado por:

$$A(u_i^{k,j}, U_i, f_j) = \begin{cases} \frac{\min(u_i^{k,j}, U_i)}{U_i} + (2f_j) & \text{se } k = \kappa(i) \\ \frac{\min(u_i^{k,j}, U_i)}{U_i} + \left( \frac{\min(u_i^{k+1,j}, U_i)}{U_i} - \frac{\min(u_i^{k,j}, U_i)}{U_i} \right) (f_j) & \text{se } k \neq \kappa(i) \end{cases}$$

- ▶ A energia consumida por unidade de tempo pelo servidor  $S_i$  em um modo  $k$  e frequência  $f_j$  é definido por:

$$P_i^{k,j} = (P_i^{k,l} + C_{ef} V^2 f_j) u_i^{k,j}$$

# Modelo

$$P : \quad f = \max \sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} A_i^{k,j} x_i^{k,j} \quad (3a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_i} x_i^{k,j} = 1, \quad S_i \in \mathcal{S} \quad (3b)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} u_i^{k,j} x_i^{k,j} \leq 1 \quad (3c)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} P_i^{k,j} x_i^{k,j} \leq \beta P^* \quad (3d)$$

$$x_i^{k,j} \in \{0, 1\}, \quad S_i \in \mathcal{S}, \quad (k,j) \in \Omega_i \quad (3e)$$

# Sumário

## 1 Introdução

- Sistema de Tempo Real
- Consumo de Energia
- Alta Variabilidade de Carga

## 2 Modelo Discreto

## 3 Algoritmo

- Algoritmo de Programação Dinâmica
- Relaxações
- Heurística Gulosa Baseada em Densidade

## 4 Conclusão

# Programação Dinâmica

- ▶ Consiste em quebrar o problema em instâncias menores, resolvendo recursivamente e armazenando as soluções
- ▶ As menores instâncias devem ser mais fáceis de serem resolvidas
- ▶ Leva em conta todas as possível decisões das instâncias anteriores ao resolver a atual, sempre escolhendo a solução ótima
- ▶ Obtém um ótimo global

# Considerações

- ▶ Existe uma constante  $\Lambda$  que ao multiplicar a utilização produz um valor inteiro
- ▶ Existe uma constante  $\Gamma$  que ao multiplicar a energia disponível produz um valor inteiro

# Programação Dinâmica

$$P : \quad f = \max \sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} A_i^{k,j} x_i^{k,j} \quad (4a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_i} x_i^{k,j} = 1, \quad S_i \in \mathcal{S} \quad (4b)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} u_i^{k,j} x_i^{k,j} \Lambda \leq \Lambda \quad (4c)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} P_i^{k,j} x_i^{k,j} \Gamma \leq \Gamma \beta P^* \quad (4d)$$

$$x_i^{k,j} \in \{0, 1\}, \quad S_i \in \mathcal{S}, (k,j) \in \Omega_i \quad (4e)$$

# Considerações

- ▶ Para criar a instância menor do problema será necessário:
  - Restringir o problema aos  $I$  primeiros servidores
  - Cada conjunto de  $I$  servidores vai restringir a utilização a um valor  $\delta$  e a energia disponível a um valor  $\gamma$
- ▶ Algumas condições iniciais e de contorno devem ser atendidas:
  - $f_0(\delta, \gamma) = 0, \forall \delta \text{ e } \gamma$
  - $f_I(\delta, \gamma) = -\infty$  se  $\Lambda u_I^{k,j} > \delta$  ou  $\Gamma P_I^{k,j} > \gamma$

# Programação Dinâmica

$$P_l(\delta, \gamma) : \quad f_l(\delta, \gamma) = \max \sum_{(k,j) \in \Omega_l} A_l^{k,j} x_l^{k,j} + f_{l-1}(\delta - \delta_l, \gamma - \gamma_l) \quad (5a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_l} x_l^{k,j} = 1 \quad (5b)$$

$$\delta_l = \sum_{(k,j) \in \Omega_l} \Lambda u_l^{k,j} x_l^{k,j} \leq \delta \quad (5c)$$

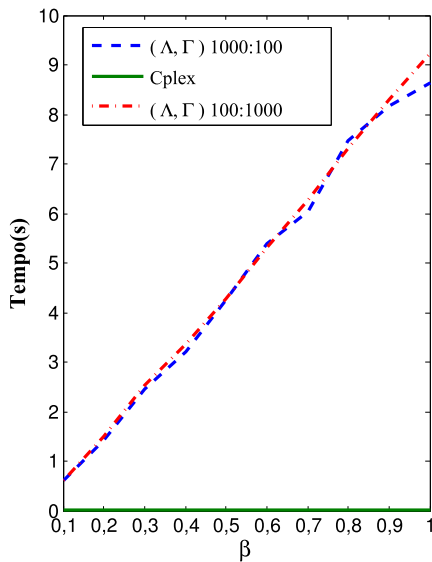
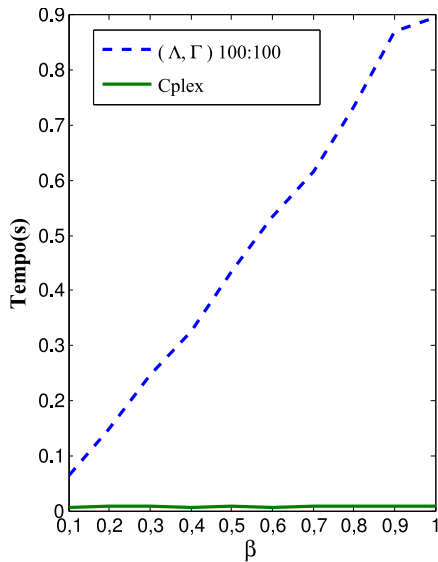
$$\gamma_l = \sum_{(k,j) \in \Omega_l} \Gamma P_l^{k,j} x_l^{k,j} \leq \gamma \quad (5d)$$

$$x_l^{k,j} \in \{0, 1\}, (k, j) \in \Omega_l \quad (5e)$$

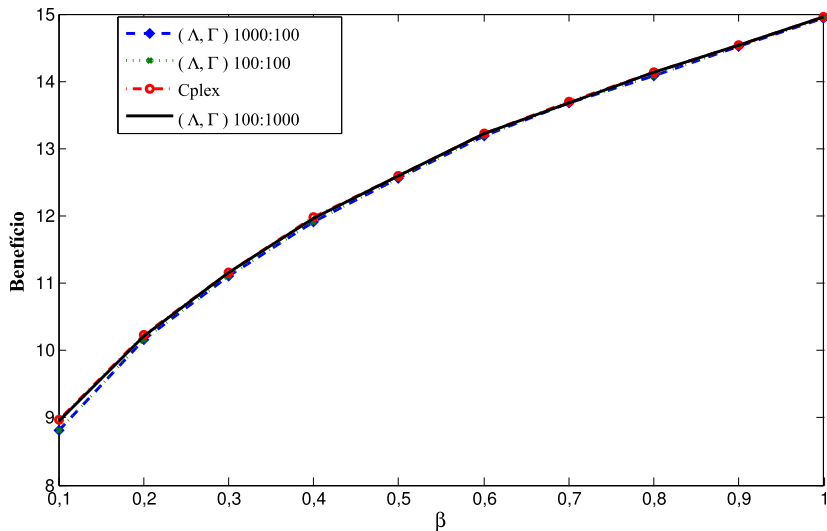
# Cenário para Avaliação Numérica

- ▶ Número de tarefas: 3
- ▶ Número de frequências: 4
- ▶ Número de modos: 3
- ▶ Com valores de  $(\Lambda, \Gamma)$ :  $(100, 100)$ ,  $(1000, 100)$  e  $(100, 1000)$

# Análise do Tempo Computacional

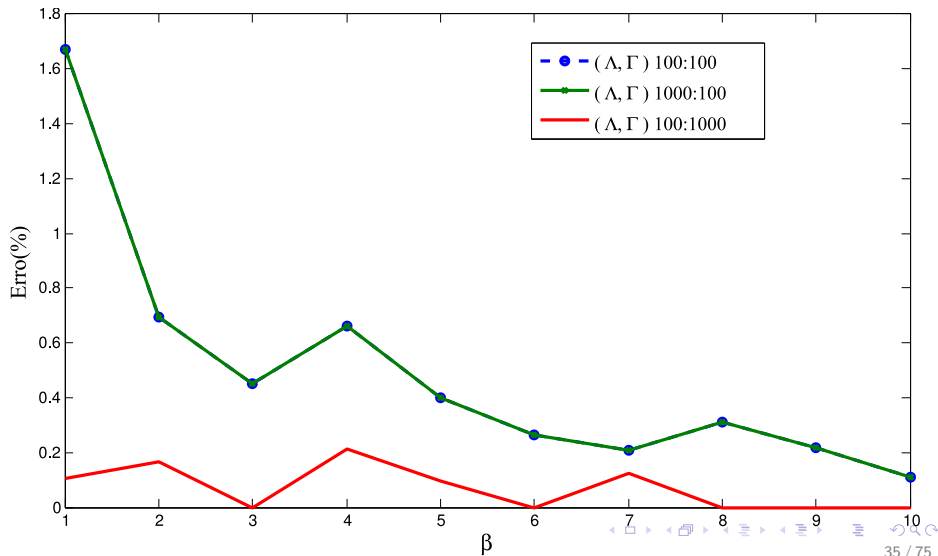


# Análise do Benefício



# Análise do Erro

$$Erro = \frac{(\text{Benefício Ótimo} - \text{Benefício Heurística})}{\text{Benefício Ótimo}}$$



# Observações Sobre Programação Dinâmica

- ▶ A programação dinâmica é viável apenas em sistemas pequenos ou que não precisam de resposta em um curto espaço de tempo
- ▶ Uma vez computada a solução ela pode ser armazenada e reutilizada depois
- ▶ Pode encontrar a solução ótima ou quase ótima
- ▶ Utilização de um *solver* pode ser mais eficiente

# Relaxações

Um problema de otimização, denotado por  $R$ , definido por  $r^* = \max\{r(\mathbf{x}) : \mathbf{x} \in \mathcal{R}\}$ , é uma relaxação do problema  $P$ , representado por  $f^* = \max\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{P}\}$ , se:

- i)  $\mathcal{P} \subseteq \mathcal{R}$
  - ii)  $r(\mathbf{x}) \geq f(\mathbf{x})$  para todo  $\mathbf{x} \in \mathcal{P}$ , sendo  $\mathbf{x}$  factível
- ▶ A relaxação se resolvida até sua otimalidade, obtém  $r^*$  induzindo um limite superior do valor ótimo do problema  $P$
  - ▶ Relaxações estabelecem certificados de qualidade para uma solução factível e são importantes no projeto de algoritmos

# Relaxação *Surrogate* Dual

$$P_s : \quad f_s(\tau) = \max \sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} A_i^{k,j} x_i^{k,j} \quad (6a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_i} x_i^{k,j} = 1, \quad S_i \in \mathcal{S} \quad (6b)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} (\tau_u u_i^{k,j} + \tau_P P_i^{k,j}) x_i^{k,j} \leq \tau_u + \tau_P \beta P^* \quad (6c)$$

$$x_i^{k,j} \in \{0, 1\}, S_i \in \mathcal{S}, (k, j) \in \Omega_i \quad (6d)$$

- Induz um limite superior para  $P$ , ou seja,  $f_s(\tau) \geq f$ , para qualquer  $\tau \geq 0$

# Relaxação *Surrogate* Dual

- ▶ Resolver o problema surrogate acima pode ser difícil, pois é equivalente ao problema da mochila
- ▶ Para o algoritmo proposto é necessário determinar o parâmetro  $\tau$

# Relaxação Lagrangeana Dual

$$LP(\lambda) : \quad f_L(\lambda) = \max \sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} (A_i^{k,j} - \lambda_P P_i^{k,j} - \lambda_u u_i^{k,j}) x_i^{k,j} + \lambda_P \beta P^* + \lambda_u \quad (7a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_i} x_i^{k,j} = 1, \quad S_i \in \mathcal{S} \quad (7b)$$

$$x_i^{k,j} \in \{0, 1\}, S_i \in \mathcal{S}, (k, j) \in \Omega_i \quad (7c)$$

- Induz um limite superior para  $P$  ou seja,  $f_L(\lambda) \geq f$ , para qualquer  $\lambda \geq 0$ .

$$LD : f_L(\lambda^*) = \min_{\lambda \geq 0} f_L(\lambda) \quad (8)$$

- ▶ Computar o problema de relaxação Lagrangeana acima é simples
- ▶ Pode-se usar o algoritmo do subgradiente para resolver de forma aproximada
  - É um problema convexo não diferenciável

# Heurística Gulosa Baseada em Densidade

- ▶ A ideia da heurística é atribuir a cada configuração uma densidade
- ▶ A densidade deve ser atribuída de acordo com benefício que a configuração traz ao sistema, relativo ao consumo de recursos:

$$\frac{A_i^{k,j}}{(\tau_u^* u_i^{k,j} + \tau_P^* P_i^{k,j})}$$

# Heurística Gulosa Baseada em Densidade

## Definição

$X \subseteq \Omega$  é um conjunto de servidores factível se:

- 1 para cada  $S_i \in \mathcal{S}$  existe exactamente uma configuração  $(i, k, j) \in X$
- 2  $\sum_{(i,k,j) \in X} u_i^{k,j} \leq 1$  e  $\sum_{(i,k,j) \in X} P_i^{k,j} \leq \beta P^*$ .

## Hipótese

Existe um conjunto factível  $X(\beta)$  das configurações dos servidores para o limite de consumo de energia estabelecido por  $\beta$

# Heurística Gulosa Baseada em Densidade

O problema da heurística é definido com base em uma solução dada  $X$ , que deve ser melhorada.

- ▶  $\Omega(X) = \Omega - X$
- ▶  $\Delta P_i^{k,j} x_i^{k,j} = P_i^{k,j} - P_i^{(k,j)X,i}$
- ▶  $\Delta u_i^{k,j} x_i^{k,j} = u_i^{k,j} - u_i^{(k,j)X,i}$
- ▶  $\Delta A_i^{k,j} x_i^{k,j} = A_i^{k,j} - A_i^{(k,j)X,i}$

# Problema Inicial

$$P : \quad f = \max \sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} A_i^{k,j} x_i^{k,j} \quad (9a)$$

$$\text{s.a : } \sum_{(k,j) \in \Omega_i} x_i^{k,j} = 1, \quad S_i \in \mathcal{S} \quad (9b)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} u_i^{k,j} x_i^{k,j} \leq 1 \quad (9c)$$

$$\sum_{S_i \in \mathcal{S}} \sum_{(k,j) \in \Omega_i} P_i^{k,j} x_i^{k,j} \leq \beta P^* \quad (9d)$$

$$x_i^{k,j} \in \{0, 1\}, \quad S_i \in \mathcal{S}, \quad (k,j) \in \Omega_i \quad (9e)$$

# Heurística Gulosa Baseada em Densidade - Problema Equivalente

$$P(X) : \quad f = \max \quad \sum_{(i,k,j) \in \Omega(X)} \Delta A_i^{k,j} x_i^{k,j} + \sum_{S_i \in \mathcal{S}} A_i^{(k,j)_{X,i}} \quad (10a)$$

$$\text{s.a :} \quad \sum_{(k,j) \in \Omega_i(X)} x_i^{k,j} \leq 1, \quad S_i \in \mathcal{S} \quad (10b)$$

$$\sum_{(i,k,j) \in \Omega(X)} \Delta u_i^{k,j} x_i^{k,j} \leq 1 - \sum_{S_i \in \mathcal{S}} u_i^{(k,j)_{X,i}} \quad (10c)$$

$$\sum_{(i,k,j) \in \Omega(X)} \Delta P_i^{k,j} x_i^{k,j} \leq \beta P^* - \sum_{S_i \in \mathcal{S}} P_i^{(k,j)_{X,i}} \quad (10d)$$

$$x_i^{k,j} \in \{0, 1\}, (i, k, j) \in \Omega(X) \quad (10e)$$

# Heurística Gulosa Baseada em Densidade

- 1 Resolve o Lagrangeano dual  $LD$  e encontra valores aproximados para os multiplicadores Lagrangeanos e uma solução factível  $\mathbf{x}^{\text{best}}$
- 2 Definir uma configuração inicial,  $X$ , viável usando  $\mathbf{x}^{\text{best}}$  se a solução candidata for factível, caso contrario  $X = \{(i, 1, 1) : S_i \in \mathcal{S}\}$
- 3 Obter o problema  $P(X)$  usando  $X$
- 4 Ordenar as configuração de forma não crescente pela densidade
- 5 Analisar as configurações segundo a ordem, mudando as configurações escolhidas se uma outra configuração melhorar o benefício global e continuar obedecendo as restrições.

# Heurística Gulosa Baseada em Densidade

$$\tilde{A} = 5$$

$$\tilde{A} = 4$$

$$\tilde{A} = 6$$

$$\tilde{A} = 1$$

$$\tilde{A} = 3$$

$$\tilde{A} = 5$$

$$\tilde{A} = 2$$

$$\tilde{A} = 2$$

# Heurística Gulosa Baseada em Densidade

$$\tilde{A} = 6$$

$$\tilde{A} = 5$$

$$\tilde{A} = 5$$

$$\tilde{A} = 4$$

$$\tilde{A} = 3$$

$$\tilde{A} = 2$$

$$\tilde{A} = 2$$

$$\tilde{A} = 1$$

# Heurística Gulosa Baseada em Densidade

$U=0,46; P=30; A=1,8$



$\tilde{A} = 3$

$\tilde{A} = 2$



$U=0,36; P=20; A=1,75$

Benefício: 3,55

Utilização: 82%

Energia: 50(Máximo 65)

$\tilde{A} = 6$

$\tilde{A} = 5$

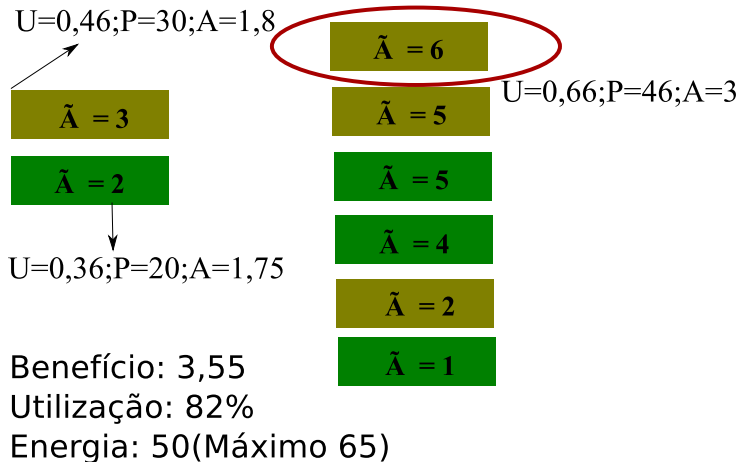
$\tilde{A} = 5$

$\tilde{A} = 4$

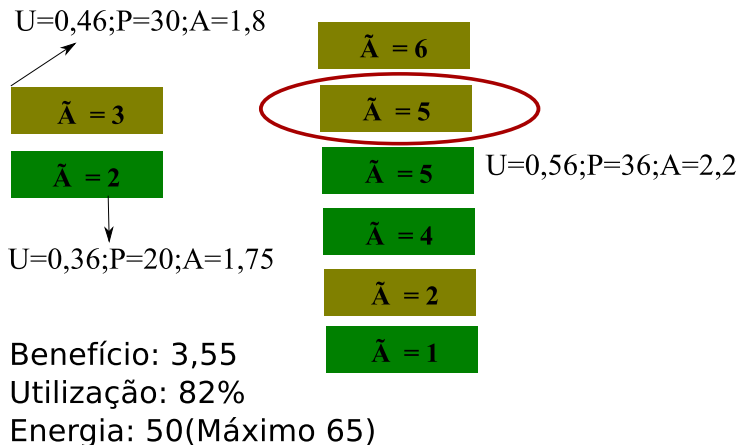
$\tilde{A} = 2$

$\tilde{A} = 1$

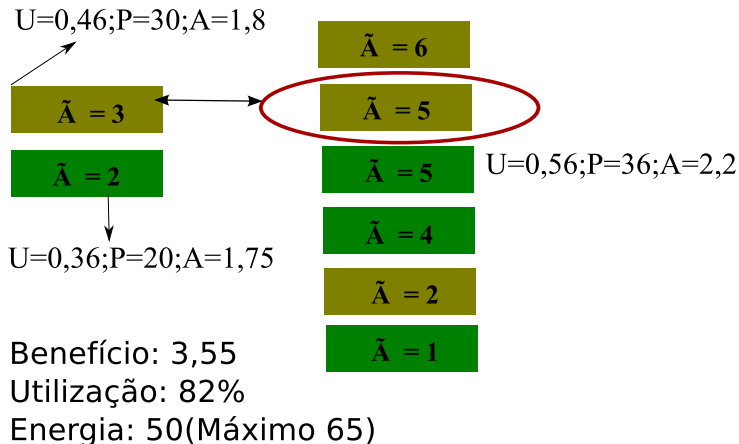
# Heurística Gulosa Baseada em Densidade



# Heurística Gulosa Baseada em Densidade



# Heurística Gulosa Baseada em Densidade



# Heurística Gulosa Baseada em Densidade

$U=0,56; P=36; A=2,2$

$\tilde{A} = 5$

$\tilde{A} = 2$

$U=0,36; P=20; A=1,75$

Benefício: 3,95

Utilização: 92%

Energia: 56(Máximo 65)

$\tilde{A} = 6$

$\tilde{A} = 3$

$\tilde{A} = 5$

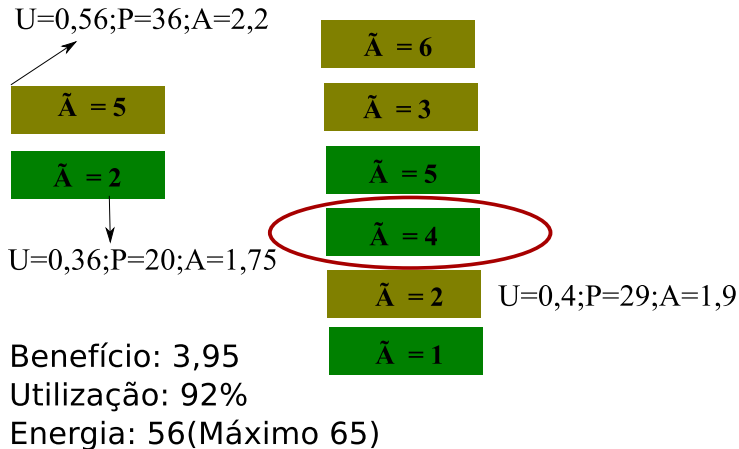
$\tilde{A} = 4$

$\tilde{A} = 2$

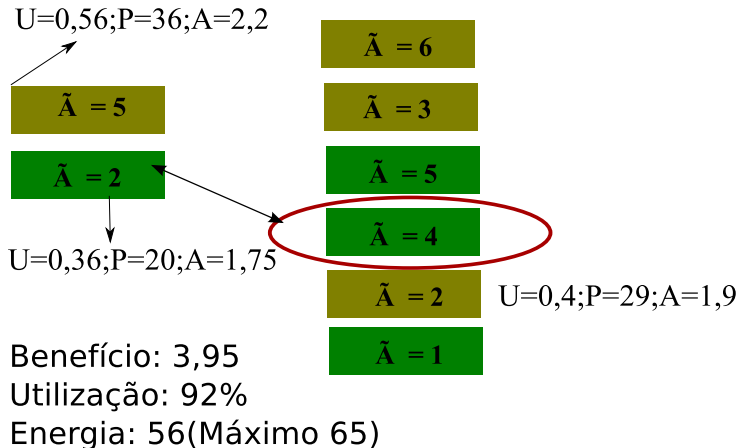
$\tilde{A} = 1$

$U=0,6; P=19; A=2,2$

# Heurística Gulosa Baseada em Densidade



# Heurística Gulosa Baseada em Densidade



# Heurística Gulosa Baseada em Densidade

$U=0,56; P=36; A=2,2$

$\tilde{A} = 5$

$\tilde{A} = 4$

$U=0,4; P=29; A=1,9$

Benefício: 4,1

Utilização: 96%

Energia: 65(Máximo 65)

$\tilde{A} = 6$

$\tilde{A} = 3$

$\tilde{A} = 5$

$\tilde{A} = 2$

$\tilde{A} = 2$

$\tilde{A} = 1$

$U=0,6; P=6; A=0,75$

# Heurística Gulosa Baseada em Densidade

$U=0,56; P=36; A=2,2$

$\tilde{A} = 5$

$\tilde{A} = 4$

$U=0,4; P=29; A=1,9$

Benefício: 4,1

Utilização: 96%

Energia: 65(Máximo 65)

$\tilde{A} = 6$

$\tilde{A} = 3$

$\tilde{A} = 5$

$\tilde{A} = 2$

$\tilde{A} = 2$

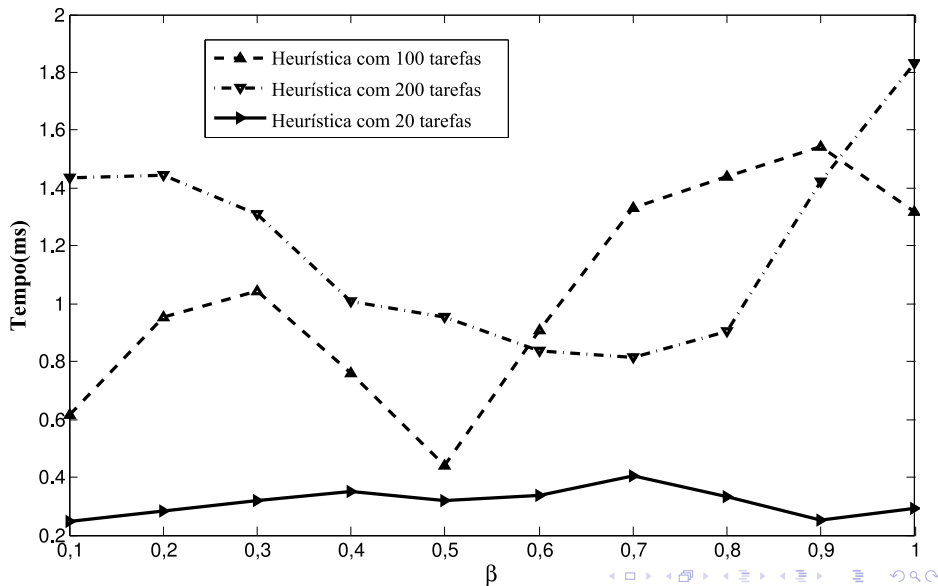
$\tilde{A} = 1$

$U=0,8; P=12; A=0,45$

# Cenários para Avaliação Numérica

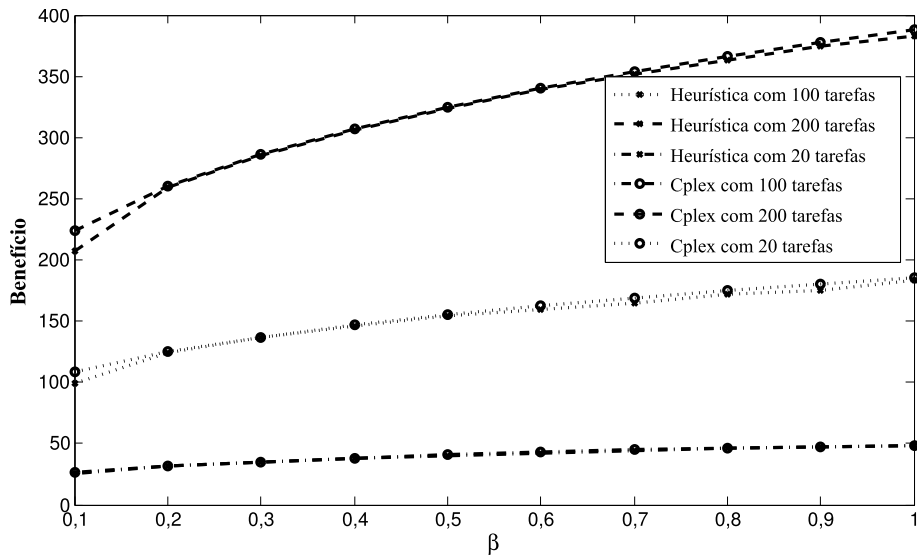
- ▶ Número de tarefas: 20, 100 e 200
- ▶ Número de frequências: 4
- ▶ Número de modos: 3

# Análise do Tempo Computacional



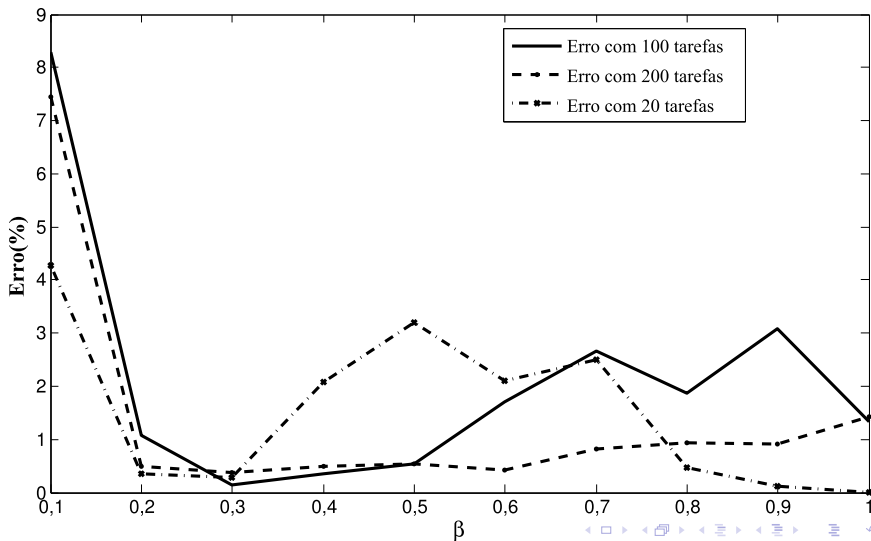
# Análise do Tempo Computacional

# Análise de Benefício



# Análise do Erro

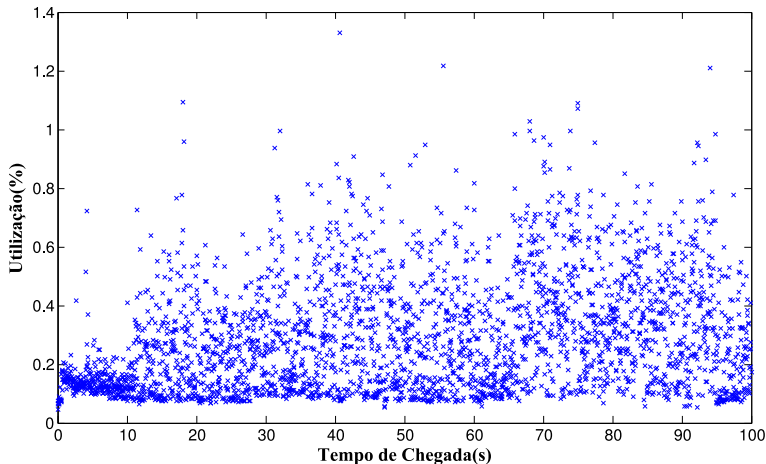
$$Erro = \frac{(\text{Benefício Otimizado} - \text{Benefício Heurística})}{\text{Benefício Otimizado}}$$



# Observações da Heurística Gulosa Baseada em Densidade

- ▶ Relaxação Lagrangeana gera limites razoáveis
- ▶ O desempenho do algoritmo do subgradiente depende de um ajuste de parâmetros
- ▶ A heurística é rápida
- ▶ O resultado é satisfatório, ficando com erro entre 0% e 8.4%

# Simulação - Codificação de Vídeo

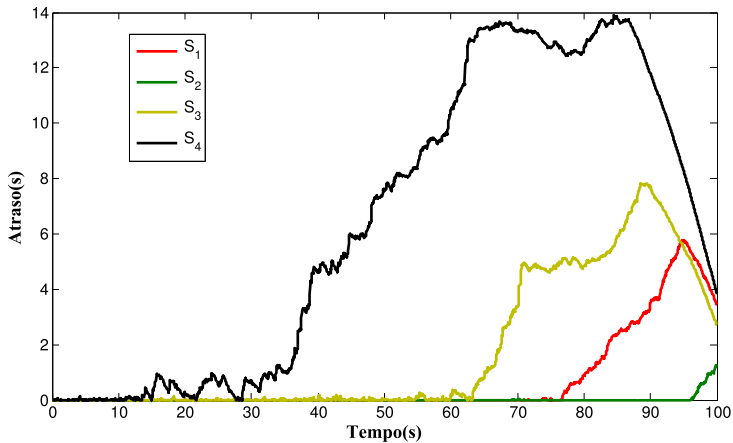


- ▶ Histórico de codificação de um vídeo de 30fps
- ▶ Utilização média de 21,62%, máxima de 126,51% e mínima 4,02%

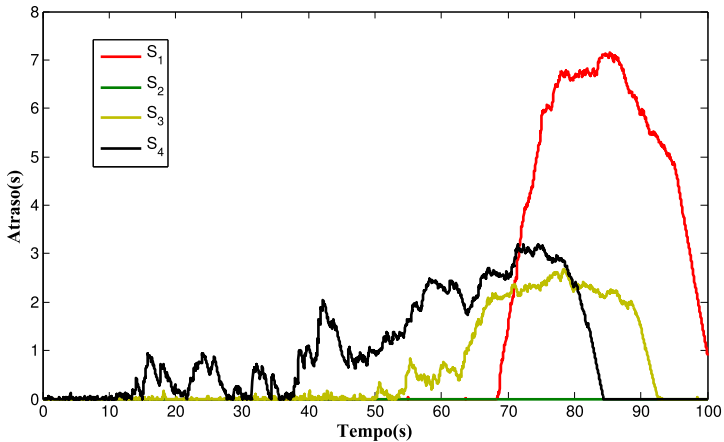
# Cenário para Avaliação Numérica

- ▶ Número de tarefas: 4
- ▶ Número de frequências: 6
- ▶ Número de modos: 3

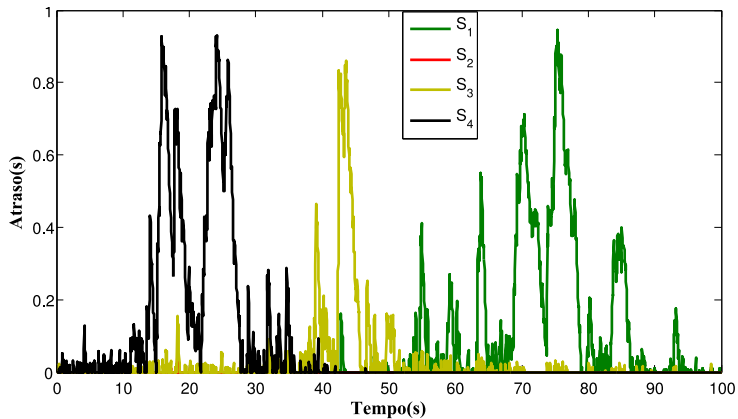
# Simulação - Codificação de Vídeo



# Simulação - Codificação de Vídeo



# Simulação - Codificação de Vídeo



# Simulação - Codificação de Vídeo

**Tabela:** Resultados da Simulação sem Reconfiguração.

|                      | $S_1$    | $S_2$    | $S_3$    | $S_4$    | Total  |
|----------------------|----------|----------|----------|----------|--------|
| Tempo(s)             | 18,9522  | 18,3117  | 18,7313  | 19,2576  |        |
| Energia(J)           | 482,1061 | 465,8130 | 476,4868 | 489,8748 | 1914,3 |
| Benefício do Sistema |          |          | 11,96    |          |        |

**Tabela:** Resultados da Simulação com  $\beta = 1$ .

|                      | $S_1$    | $S_2$    | $S_3$    | $S_4$   | Total  |
|----------------------|----------|----------|----------|---------|--------|
| Tempo(s)             | 20,9685  | 19,2458  | 18,3131  | 15,055  |        |
| Energia(J)           | 533,3967 | 489,5747 | 189,9408 | 382,969 | 1595,9 |
| Benefício do Sistema |          |          | 9,96     |         |        |

**Tabela:** Resultados da Simulação com  $\beta = 0,7$ .

|                      | $S_1$    | $S_2$    | $S_3$    | $S_4$   | Total  |
|----------------------|----------|----------|----------|---------|--------|
| Tempo(s)             | 20,9685  | 19,2458  | 18,3132  | 7,7122  |        |
| Energia(J)           | 533,3967 | 489,5747 | 189,9408 | 48,7658 | 1265,1 |
| Benefício do Sistema |          |          | 9,17     |         |        |

Caso o sistema conseguisse executar utilizando 100% da *CPU* teria consumido 1875J, o que implica que:

- ▶ na simulação sem reconfiguração foi gasto 2,08% da energia disponível no pior caso
- ▶ na simulação com reconfiguração e  $\beta = 1$  foi gasto 85,11% da energia disponível
- ▶ na simulação com reconfiguração e  $\beta = 0,7$  foi gasto 67,72% da energia disponível

# Sumário

- 1 Introdução
  - Sistema de Tempo Real
  - Consumo de Energia
  - Alta Variabilidade de Carga
- 2 Modelo Discreto
- 3 Algoritmo
  - Algoritmo de Programação Dinâmica
  - Relaxações
  - Heurística Gulosa Baseada em Densidade
- 4 Conclusão

# Contribuição

- ▶ Infraestrutura para reconfiguração dinâmica de escalonadores em tempo real
  - Modelo de otimização discreto
  - Escalonamento respeitando restrição de energia
  - Algoritmo rápido, com solução próxima da ótima
- ▶ Avaliação
  - Análise numérica para conjuntos pequenos e grandes de tarefas
  - Simulação de um sistema de codificação de vídeo, com dados reais do *mencoder*

# Submissão

R. M. Nassiffe, E. Camponogara e G. Lima, "Optimizing quality of service in real-time systems under energy constraints," 23th IEEE Euromicro Conf. on Real-Time Systems, 2011.

Obrigado!