# O problema do escalonamento do Funcionário Preguiçoso (LBSP)

Edith Z. Sonco Mamani

# Introdução

- É introduzido uma nova classe de problemas de escalonamento de tarefas, Arkin [1] , neste caso vamos olhar o problema a partir de ponto de vista dos trabalhadores que executam as tarefas.
- Alguns trabalhadores podem não ter a motivação para executar seu trabalho em seus níveis maximos de eficiência.
- Eles não têm participação nos lucros da empresa, ou porque simplesmente são preguicosos.

- Empregador:  Mais trabalho! Pagar menos!

- Empregador:  Mais trabalho! Pagar menos!
- Comprador: Melhor qualidade! Mais barato!

- Empregador:  Mais trabalho! Pagar menos!
- Comprador: Melhor qualidade! Mais barato!
- Empregado: Mais dinheiro! Menos trabalho!

Ling GAI et al. 2007

# Exemplo

Arkin et al. 2002

- São as 3:00 p.m.
- Dilbert vai para casa às 5:00 p.m.
- Dilbert tem duas tarefas que foram dadas a ele: um requer 10 min, o outro requer uma hora.
- Se existe uma tarefa em seu "in-box", Dilbert deve trabalhar sobre ela.
- Ele também sabe que às 3:15, outra tarefa vai aparecer.

# Exemplo

- Solução 1: Se Dilbert começa primeiro a tarefa de 10 minutos, ele estará livre para participar da reunião de pessoal às 3:15 e depois pode trabalhar sobre a tarefa de uma hora de 4:00 até 5:00.

# Exemplo

- Solução 1: Se Dilbert começa primeiro a tarefa de 10 minutos, ele estará livre para participar da reunião de pessoal às 3:15 e depois pode trabalhar sobre a tarefa de uma hora de 4:00 até 5:00.
- Solução 2: Se Dilbert forma parte da tarefa de uma hora às 3:15, ele pode ser dispensado da reunião. Depois de terminar o trabalho de 10-minutos as 4:10, ele terá 50 min livres.

# Exemplo

- Solução 1: Se Dilbert começa primeiro a tarefa de 10 minutos, ele estará livre para participar da reunião de pessoal às 3:15 e depois pode trabalhar sobre a tarefa de uma hora de 4:00 até 5:00.
- Solução 2: Se Dilbert forma parte da tarefa de uma hora às 3:15, ele pode ser dispensado da reunião. Depois de terminar o trabalho de 10-minutos as 4:10, ele terá 50 min livres.
- **Solução escolhida: Dilbert prefere a segunda opção.**

# O modelo

- Conjunto de tarefas: $J=\{ J_1, J_2, ..., J_n \}$
- Tempo de processamento: $t_1,...,t_n$
- Tempo de chegada de tarefa $i$ : $a_i$
- Deadline de tarefa $i$ : $d_i$
- Tempo de liberação da tarefa $i$ : $r_i$
- Prazos rígidos: $I_i=[a_i,d_i]$
- Tempo crítico: $c_i=d_i-t_i$
- Processadores : único processador.

# Requirimento Guloso

- O funcionário escolhe um subconjunto de tarefas a executar.
- Seu objetivo é minimizar o seu esforço, ele prefere permanecer ocioso o tempo todo e deixar todos as tarefas não executadas.
- Este cenário é proibido por: o requerimento guloso.
- Uma tarefa é executável se ela chegou, o prazo não passou, e ela não está totalmente processada.

# Funções Objetivo

- [min-time-spent]
- [min-weighted-sum-of-completed-jobs]
- [min-makespan]
- [min-number-of-job-completed]

# Parâmetros adicionais

- Deve-se explicitamente permitir ou proibir preempção nas tarefas. Se uma tarefa é preemptiva, é interrompido e pode ser retomado mais tarde. Se preempção é proibida, então uma vez que uma tarefa é iniciado, ele deve ser completada sem interrupções.
- Também é preciso especificar se o planejamento ocorre off-line ou on-line.

# Resultados da literatura

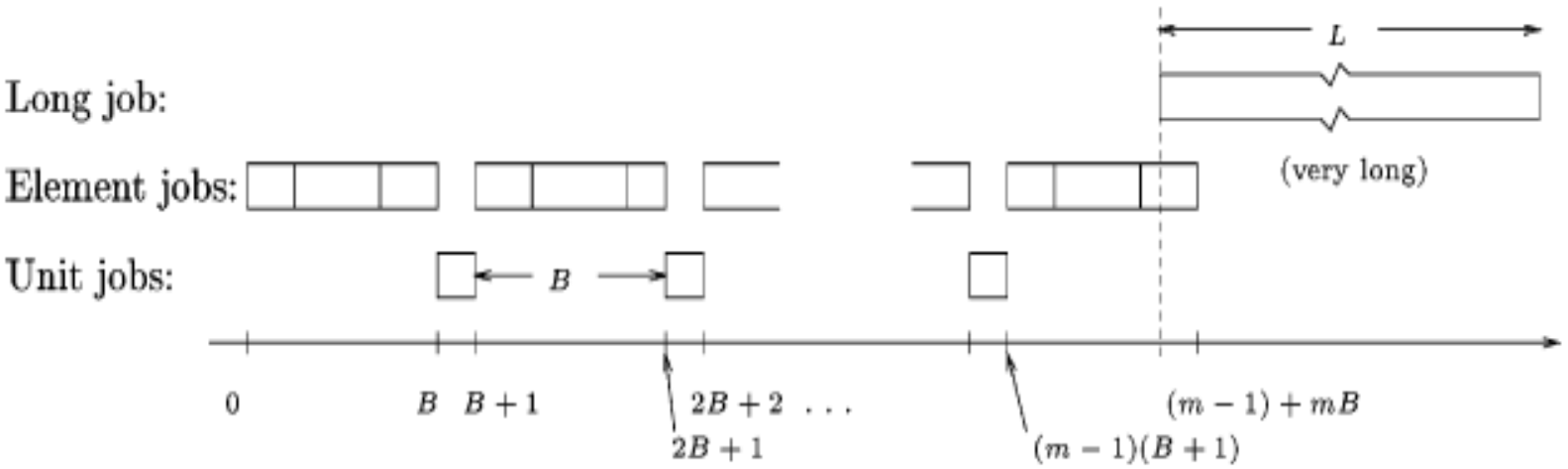
- São desenvolvidos algoritmos e resultados sólidos para várias versões do LBP.
- Podemos tirar algumas características gerais dessa nova classe de problemas de escalonamento e descrever:
  - (1) situações em que os algoritmos de escalonamento tradicionais estendem ao LBP e
  - (2) situações em que estes algoritmos não se aplicam mais.

# LBP: sem preempção

- Se uma tarefa é iniciada, então é realizada sem interrupção até sua conclusão.
- O problema do funcionario preguiçoso(LBP) sem interrupção é  fortemente NP-completo para as funções objetivo (1)-(3)
  - Subset Sum
  - Redução do problema 3-partição

# LBP: sem preempção(cont.)

- Tarefas elemento
- Tarefas de unidade
- Tarefas grandes



# LBP: sem preempção(cont.)

- Algoritmos para casos especiais.
  - Tarefas de comprimento de unidade
    - Todas as tarefas tem tempo de processamento de uma unidade
    - Política de escalonamento LDD

# LBP: sem preempção(cont.)

- Algoritmos para casos especiais.
  - Tarefas com tempo comum de liberação($r_i$)
    - Todas as tarefas tem tempo de liberação $a_i=0$
    - Solução em tempo pseudo-polinomial com programação dinâmica.
    - Existe um ótimo escalonamento que executa as tarefas EDD.
    - Caso especial de : [min-weighted-sum-of-completed-jobs]

# LBP: preempção

- Restrição I: Trabalhar na tarefa *i* no tempo $\tau$ , so e necessario que o tempo atual $\tau$ encontra-se dentro do intervalo da tarefa $I_i$: $a_i \leq \tau \leq d_i$.
- Restrição II: Trabalhar na tarefa i no tempo $\tau$ , é necessario que a tarefa tenha uma chance de ser concluda.
- Restrição III: Trabalhar na tarefa i, e necessario que $\tau \in I_i$.

# LBP: preempção

- Minimizando o tempo total de trabalho
  - O LBP com preempção , sob a restrição I tem solução polinomial.
    - Solução: procedimento a ultima data de vencimento primeiro (LDD).
  - P LBP com preempção, sob a restrição II tem solução NP-completo.
    - Solução: igual ao do makespan.

# LBP: preempção(cont.)

- Minimizando o a soma ponderada dos pesos de tarefas concluídas
  - O LBP com preempção , sob a restrição I tem solução polinomial.
    - Solução: procedimento a primeira data de vencimento primeiro (EDD).
  - P LBP com preempção, sob a restrição II tem solução NP-completo.
    - Solução: igual ao do makespan.

# LBP: preempção(cont.)

- Minimizando o makespan
  - O LBP com preempção , sob a restrição I tem solução polinomial.
    - Solução: procedimento a ultima data de vencimento primeiro (LDD).
  - P LBP com preempção, sob a restrição II tem solução NP-completo, se os tempos de chegada son os mesmos .
    - Solução: Subset Sum.

# LBP: preempção(cont.)

- Minimizando o makespan
  - O funcionário fica livre para ir a casa no tempo T:
    - O funcionario deve ter acabado de concluir uma tarefa no tempo T.
    - O funcionario deve car ocupado o tempo inteiro de 0 ate o tempo T.
    - Se o funcionario inicia uma tarefa, então ele deve termina-la

# Resumo

Tabela 1: Resumo de resultados sem interrupções; ver secção 3. Arkin [1]

| Instância | Objetivo | Complexidade |
|---|---|---|
| Tarefas de comprimento 1:$t_1 = t_2 = ... = t_n = 1$ | 1 | Tempo polinomial |
| Tarefas de intervalos curtos:$\forall i d_i - a_i < 2t_i$ | 1-3 | Tempo pseudo-polinomial. |
| Ratios $R = O(1) e \Delta = O(1)$ | 1-3 | Tempo pseudo-polinomial. |
| Mesmo tempo de chegada:$a_1 = a_2 = ... = a_n$ | 1-3 | Tempo pseudo-polinomial.<br>(Débilmente)NP-completos<br>Difícil para aproximar |
| Tarefas de tamanho de ratio | 1-2 | Fortemente NP-completos<br>Difícil para aproximar |

# Resumo(cont)

Tabela 2: Resumo de resultados com interrupções; ver secção 4. Arkin [1]

| Preempção | Objetivo | Complexidade |
|---|---|---|
| I | 1-3 | Tempo polinomial |
| II | 1-3 | (Débilmente)NP-completos, mesmo quando $a_1 = a_2 = ... = a_n$ . |
| III | 1-3 | (Débilmente)NP-completos, mesmo quando $a_1 = a_2 = ... = a_n$. Difícil para aproximar |

# Referências


- [1] J.S.B. Mitchell S.S. Skiena. E.M. Arkin, M.A. Bender. The lazy bureaucrat scheduling problem, 2002.
- [2] Shari A. Esfahbod B, Ghodsi M. Common-deadline lazy bureaucrat scheduling problems, 2003.
- [3] Ling Gai e Guochuan Zhang, On lazy bureaucrat scheduling with common deadlines

Obrigada!!