Relatório da Análise de Desempenho dos Métodos para Detecção de Escalonabilidade

Cláudio Vieira [1], Vítor A. Barbosa [1]

[1] Universidade Federal da Bahia (UFBA)
Salvador – BA – Brazil

{claudio.vieira,vitor.barbosa}@ufba.br

Abstract. *This meta-paper analyses the performance of the algorithms to detect the schedulability of a task set. The algorithms analysed were the method that uses the DBF, the restrictions that use the DBF* and the QPA. For this analysis, two tests were done: the difference between the QPA and the DBF; the frequency of sufficiency of the DBF*.*

Resumo. *Este meta-artigo analisa o desempenho dos algoritmos para detecção da escalonabilidade de um conjunto de tarefas. Os algoritmos analisados foram o método que utiliza o DBF, as restrições que utilizam o DBF* e o QPA. Para essa análise, dois testes foram realizados: a diferença de velocidade entre o QPA e o DBF; a frequência de suficiência do DBF*.*

1. Introdução

Sistemas de tempo real devem cumprir prazos. Para garantir que isso ocorra é necessária a análise de programação e escalonabilidade a fim de que as tarefas concluam sua execução antes de seus deadlines. Os testes de escalonabilidade podem ser suficientes ou exatos. O EDF (*Earliest Deadline First*) é um algoritmo de prioridade dinâmica no qual o job que chegou com o deadline mais antigo é executado primeiro e os testes de escalonabilidade exata baseados em EDF precisavam verificar todos os deadlines absolutos no intervalo de tempo. Esse algoritmo foi apresentado primeiramente por Liu e Layland (1973), denominado *Deadline Driven Scheduling Algorithm*.

Contudo, havendo a possibilidade de um número muito grande desses deadlines absolutos, testes como o método que utiliza o DBF (*Demand Bound Function*) não seriam muito viáveis. O QPA (Quick convergence Processor-demand Analysis) surge como uma solução para este problema, uma vez que trata-se de um algoritmo de análise de demanda de processador de convergência rápida. Além de fornecer testes de escalonabilidade mais rápidos e simples para EDF, também é necessário e suficiente, reduzindo o esforço de cálculo exponencial. Através dele não mostra-se necessário verificar todos os prazos e registrar todos os valores dos deadlines no intervalo, mesmo quando o conjunto de tarefas é escalonável. Além desses dois algoritmos, existe um método não suficiente que utiliza a função DBF* e que é bem rápido. Se as restrições para esse método forem validadas, conclui-se que o conjunto de tarefas é escalonável. Porém o teste não é conclusivo em caso de não validação.

Neste relatório serão apresentados resultados de testes comparativos entre o DBF, QPA e DBF*. Na seção 2, será apresentada a fundamentação teórica dos algoritmos abordados neste artigo. Na seção 3, será explicado o processo de geração do conjunto de tarefas. Na seção 4, estão apresentados os resultados. Na seção 5, a análise dos resultados obtidos com os testes. Na seção 6, conclusão e considerações finais.

2. Algoritmos

Nesta seção, serão descritos os algoritmos utilizados para os testes.

2.1. DBF

A sigla DBF significa *Demand Bound Function*, que na verdade é uma função que calcula o máximo tempo de execução requerido de todos os jobs das tarefas que possuem seus tempos de chegada e seus deadlines em um intervalo contíguo de comprimento *t*. No artigo de Zhang e Burns (2009), é o cálculo de $h(t)$, denominado *Processor Demand Function* e descrito abaixo:

$$h(t) = \sum_{i=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} C_i.$$

Nesse relatório, escolhemos atribuir esse nome para o método exato que verifica se o conjunto de tarefas é escalonável ou não. Esse método é exponencial, pois existe um número exponencial de pontos a serem analisados. O pseudo-código para esse método está logo abaixo:

```
1   for i in taskset do:
2       k = 0
3       d_i = k × T_i + D_i
4       h_t = h(d_i)
5       while d_i < L && h_t ≤ d_i do:
6           k++
7           d_i = k × T_i + D_i
8           h_t = h(d_i)
9       if d_i < L && h_t > d_i then:
10          return Conjunto de tarefas não escalonável
11  return Conjunto de tarefas escalonável
```

O valor de *L* é o mínimo entre L_a e L_b, que são limites superiores do intervalo no qual o algoritmo deve testar. Eles são definidos da seguinte forma:

$$L_a = \max\left\{D_1, \ldots, D_n, \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}\right\}.$$

$$w^0 = \sum_{i=1}^{n} C_i,$$

$$w^{m+1} = \sum_{i=1}^{n} \left\lceil \frac{w^m}{T_i} \right\rceil C_i,$$

onde a recorrência para quando $w^{m+1} = w^m$, e então $L_b = w^{m+1}$.

2.2. DBF*

O DBF* na verdade é uma função utilizada nas restrições que verificam se uma tarefa τ_i pode ser atribuída com sucesso a um determinado processador. Porém, essas restrições não são capazes de informar se o conjunto de tarefas não é escalonável em caso de falha. Portanto, ele é apenas suficiente. Sua complexidade polinomial, $O(n^2)$, torna-se relevante na classificação da escalonabilidade de um conjunto de tarefas, pois os outros dois algoritmos exatos (DBF e QPA) são exponenciais. Dessa forma, em certos conjuntos de tarefas, pode-se ganhar tempo com o DBF*. A função DBF* é descrita abaixo:

$$\mathrm{DBF}^*(\tau_j, t) \stackrel{\text{def}}{=} \begin{cases} C_j + (t - D_j)u_j, & \text{if } t \geq D_j \\ 0, & \text{otherwise} \end{cases}$$

O pseudo-código para o algoritmo DBF* (restrições de verificação) encontra-se abaixo:

1 **for** i **in** taskset **do:**

2 **if** $D_i - \sum_{j \in taskset} DBF^*(\tau_j, D_i) < C_i$ **then:**

3 **return** Teste não conclusivo

4 **if** $1 - (U - U_i) < U_i$ **then:**

5 **return** Teste não conclusivo

6 **return** Conjunto de tarefas escalonável

2.3. QPA

O algoritmo QPA é suficiente e necessário, porém não demanda muito tempo como o DBF, pois ignora muitos prazos desnecessários. Embora pule muitos pontos, ele ainda pode ter que calcular muitos pontos. O pseudo-código para o QPA encontra-se abaixo.

1 $t = max\{d_i | d_i < L\}$

2 **while** $h(t) \leq t$ && $h(t) > d_{min}$ **do:**

3 **if** $(h(t) < t)$**then:** $t = h(t)$

4 **else** $t = max\{d_i | d_i < t\}$

5 **if** $h(t) \leq d_{min}$ **then**

6 **return** Conjunto de tarefas escalonável

7 **else**

8 **return** Conjunto de tarefas não escalonável

L é o mínimo entre L_a e L_b, descritos anteriormente, $h(t)$ é a mesma função utilizada no DBF e o d_{min} é o menor deadline do conjunto de tarefas. $max\{d_i | d_i < t\}$ pode ser calculado de forma linear no número de tarefas.

3. Geração do Conjunto de Tarefas

A geração de um conjunto de tarefas foi realizada de acordo com a política de geração do Zhang e Burns (2009), dividida em 3 partes: utilizações, períodos e deadlines relativos. A política de geração de utilizações é baseada no UUnifast algorithm, que retorna uma sequência de números aleatórios entre [0,1] com soma igual a utilização desejada para o conjunto de tarefas. Na geração de períodos, o autor atribui valores aleatórios de acordo com uma distribuição exponencial limitada por Tmax/Tmin, o alcance desejado para o conjunto de tarefas. Por fim, na geração de deadlines, escolhe-se valores aleatórios entre $[a, b]$, sendo a dependente do tempo de execução da tarefa $C_i = T_i \times U_i$, enquanto $b = 1.2 \times T_i$.

4. Resultados

Dois testes foram realizados para análise:

1. A diferença entre o número de cálculos do $h(t)$ no DBF e no QPA;
2. A frequência de suficiência do algoritmo DBF*.

4.1. Número de Cálculos do h(t) no DBF e no QPA

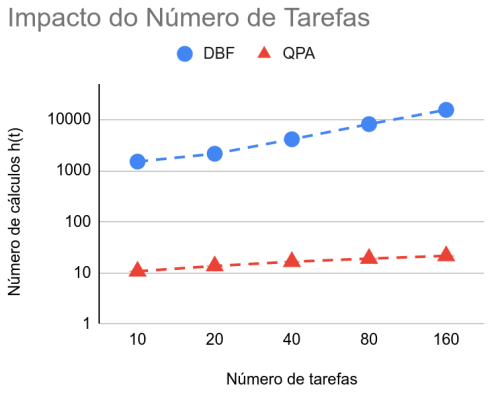
No teste 1, existem dois tipos de análise, cada um subdividido em 3 etapas, baseado nos experimentos de Zhang e Burns (2009). O primeiro tipo analisa conjuntos de tarefas escalonáveis, enquanto o segundo analisa conjuntos de tarefas não escalonáveis. A razão para separação dos testes está relacionada ao número de cálculos do h(t). Nos conjuntos de tarefas escalonáveis, o DBF checa todos os deadlines. Já nos conjuntos de tarefas não escalonáveis, o algoritmo pode parar antes caso haja uma perda de deadline.

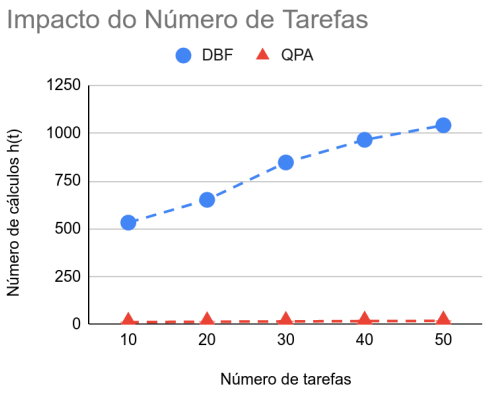
As 3 etapas são iguais para cada tipo. Trata-se da variação de: número de tarefas; alcance dos períodos; e utilização. Para isso, gerou-se um gráfico para cada etapa. Cada ponto é a média dos 6000 conjuntos de tarefas escalonáveis/não escalonáveis gerados de acordo com a política descrita na seção 3.

4.1.1. Impacto no Número de Tarefas

Na variação do número de tarefas, fixa-se a utilização em 0.9 e o alcance (Tmax/Tmin) em 1000. O Gráfico 1 retrata os resultados obtidos para os conjuntos escalonáveis em

escala logarítmica (ambos os eixos). O Gráfico 2 retrata os resultados obtidos para os conjuntos não escalonáveis sem escala logarítmica.

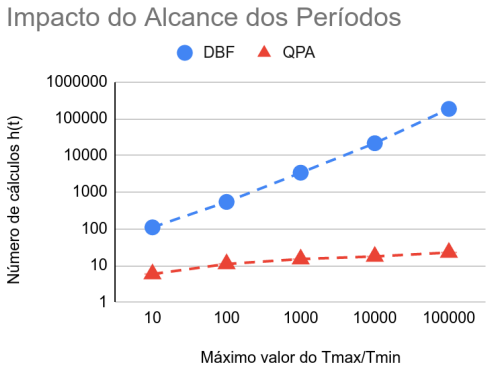


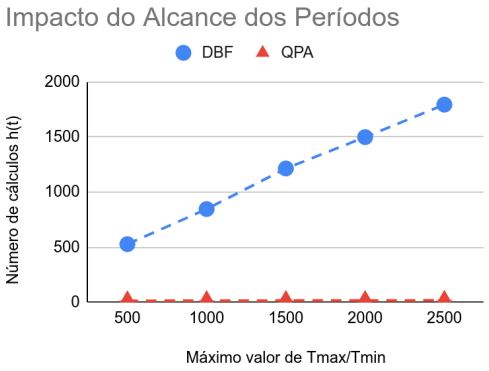


Gráficos 1 e 2 - Impacto do Número de Tarefas nos escalonáveis e nos não escalonáveis respectivamente.

4.1.2. Impacto do Alcance dos Períodos

Na variação do alcance dos períodos, fixa-se o número de tarefas em 30 e a utilização em 0.9. O Gráfico 3 retrata os resultados obtidos para os conjuntos escalonáveis em escala logarítmica (ambos os eixos). O Gráfico 4 retrata os resultados obtidos para os conjuntos não escalonáveis sem escala logarítmica.

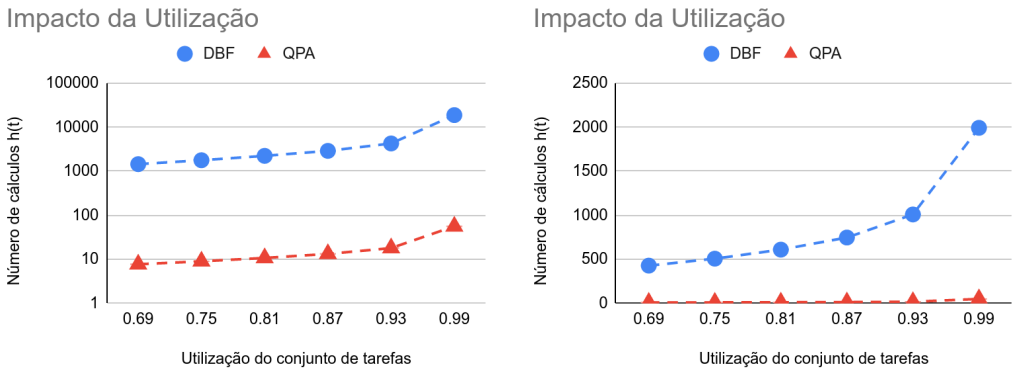




Gráficos 3 e 4 - Impacto do Alcance dos Períodos nos escalonáveis e nos não escalonáveis respectivamente.

4.1.3. Impacto da Utilização

Na variação da utilização, fixa-se o número de tarefas em 30 e o alcance (Tmax/Tmin) em 1000. O Gráfico 5 retrata os resultados obtidos para conjuntos escalonáveis em escala logarítmica (apenas no eixo y). O Gráfico 6 retrata os resultados obtidos para conjuntos não escalonáveis sem escala logarítmica.



Gráficos 5 e 6 - Impacto da Utilização nos escalonáveis e nos não escalonáveis respectivamente.

4.2. Frequência de Suficiência do DBF*

No teste 2, analisa-se a suficiência do DBF*. Como o algoritmo DBF* é mais rápido que um método exato, vale a pena utilizá-lo para verificar a escalonabilidade do conjunto, pois em caso do conjunto ser escalonável, não seria necessário executar um método exato para concluir o teste.

Dessa maneira, executou-se o pseudo-código abaixo para 6000 conjuntos de tarefas, variando da mesma maneira que na seção 4.1. o número de tarefas (Gráfico 7), o alcance dos períodos (Gráfico 8) e a utilização (Gráfico 9).

```
1   if DBF * validated then
2       return Conjunto de tarefas escalonável
3   else if QPA validated then
4       return Conjunto de tarefas escalonável
5   else
6       return Conjunto de tarefas não escalonável
```

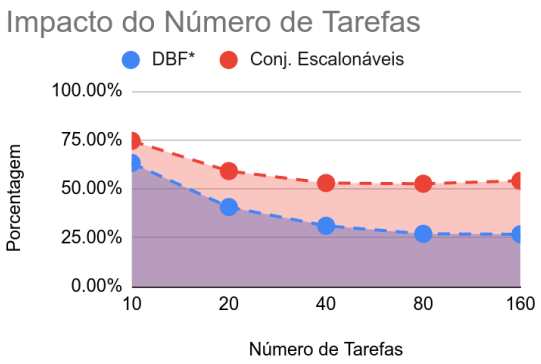


Gráfico 7. Impacto do Número de Tarefas

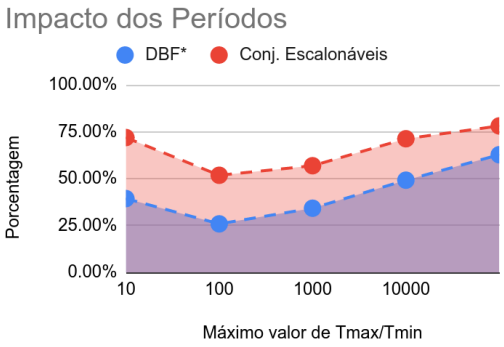


Gráfico 8. Impacto do Alcance dos Períodos

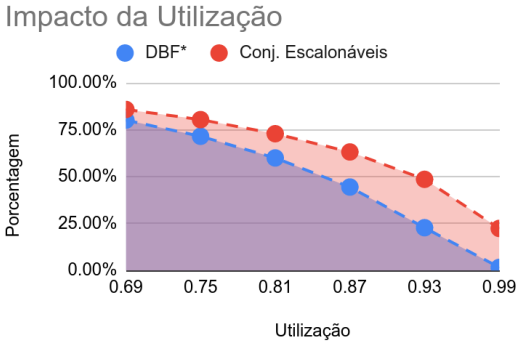


Gráfico 9. Impacto da Utilização

5. Análise dos Resultados

Nessa seção, discute-se os resultados descritos na seção 4.

5.1. Número de Cálculos do h(t) no DBF e no QPA

No primeiro teste, é possível notar uma diferença clara no número de cálculos do h(t) entre os dois algoritmos. Em todos os experimentos, independente da escalonabilidade do conjunto de tarefas e do tipo de variação, o QPA performou melhor que o DBF, com um crescimento menos acentuado.

5.2. Frequência de Suficiência do DBF*

No segundo teste, percebeu-se que o acréscimo do número de tarefas e da utilização tornou-se mais esporádica a frequência de suficiência do DBF*, sendo necessário executar mais vezes o QPA para concluir a análise. O único experimento que obteve um resultado diferente foi na variação do alcance dos períodos. Nesse caso, o aumento ocasionou uma maior frequência de suficiência do DBF*.

O ponto em comum encontrado nos três experimentos é o fato de que quando o DBF* não conseguiu ser suficiente, a maioria dos conjuntos de tarefas não era escalonável. Em outras palavras, o QPA mostrou que a maioria dos conjuntos que o DBF* não concluiu escalonabilidade eram não escalonáveis.

6. Conclusão

A partir dos resultados apresentados neste relatório, podemos concluir do primeiro teste que, comparado ao DBF, o QPA é um algoritmo que otimiza muito os testes de escalonabilidade das tarefas.

Ao comparar os resultados do segundo teste, há uma diferença na frequência de suficiência do DBF* quando variamos os parâmetros. A partir de um determinado ponto, o DBF* torna-se menos conclusivo, pois as restrições não são satisfeitas. Por isso, o DBF* passa a não ser suficiente para provar a escalonabilidade dos conjuntos de tarefas, sendo necessário executar o QPA para concluir a análise.

Conforme aumenta-se o índice de utilização do processador, mais difícil fica para o DBF* ser suficiente e o QPA assume essa tarefa de provar, aumentando seu número de testes. Os resultados dos experimentos indicam que quando o DBF* é ineficaz ao concluir a escalonabilidade, a maioria dos conjuntos de tarefas não são escalonáveis, fato posteriormente comprovado pela execução do QPA.

Referências

F. Zhang and A. Burns, "Schedulability Analysis for Real-Time Systems with EDF Scheduling," in IEEE Transactions on Computers, vol. 58, no. 9, pp. 1250-1258, Sept. 2009, doi: 10.1109/TC.2009.58.

Bini, E., Buttazzo, G.C. Measuring the Performance of Schedulability Tests. *Real-Time Syst* 30**,** 129–154 (2005). https://doi.org/10.1007/s11241-005-0507-9

C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM 20, 1 (Jan. 1973), 46–61. DOI:https://doi.org/10.1145/321738.321743