Modelo de Tarefas Baseado em Instante Ideal

Fábio Rodrigues de la Rocha*, **Rômulo Silva de Oliveira**

LCMI – DAS – Universidade Federal de Santa Catarina
Caixa Postal 476, CEP 88040-900, Florianópolis – SC

frr@das.ufsc.br, romulo@das.ufsc.br

Resumo. *Em aplicações de automação, a aquisição de dados e o controle de processos são tarefas críticas em relação ao tempo, assumidas como periódicas. Aspectos relacionados com escalonadores de tempo real podem postergar a execução das tarefas, levando à redução de desempenho e instabilidade. Neste artigo é proposto um novo modelo de tarefas para lidar com a variabilidade na execução das tarefas. Em nosso modelo as tarefas possuem um instante ideal para aquisitar dados ou apresentar seus resultados. Estes são assumidos válidos se obtidos dentro de um entorno que circunda um instante ideal* d_i*, considerado o instante onde a contribuição para o sistema é máxima. Valores obtidos antes ou após este entorno não possuem utilidade.*

1. Introdução

Em muitas aplicações de controle digital, aquisição de dados e controle de processos são tarefas críticas em relação ao tempo, assumidas como instantâneas e executadas periodicamente numa taxa constante. Num sistema de controle digital o controlador é geralmente implementado em software utilizando a abstração de tarefa. O escalonador decide que tarefa deve executar segundo uma política. As tarefas concorrem pelo uso do processador e interagem com o ambiente aquisitando dados dos sensores, calculando e enviando sinais de controle. Nestes sistemas em geral são utilizadas tarefas periódicas executando sobre um escalonador preemptivo com prioridades.

Aspectos relacionados aos algoritmos de escalonamento de tempo real podem postergar a execução de uma tarefa e, conseqüentemente, atrasar a aquisição de dados e o controle do processo. Um destes aspectos é o *jitter* que existe nos escalonadores preemptivos com prioridade. Basicamente podemos ter três tipos de *jitter* (entrada, saída e liberação). O *jitter* de entrada é causado pela variação nos tempos de inicio entre sucessivas ativações de uma tarefa. Já o *jitter* de saída é causado pela variação do tempo de término entre sucessivas ativações de uma mesma tarefa. O *jitter* de liberação é causado pela variação entre o momento em que uma tarefa chega e o momento em que ela é liberada. Todos os tipos de *jitter* causam problemas nos sistemas de controle pois violam a premissa da teoria do controle que considera que as ações devem ser realizadas em tempos constantes. O *jitter* reduz o desempenho do sistema de controle e pode levar à instabilidade.

Em sistemas de controle considera-se que os dados obtidos dentro de um entorno que circunda o instante ótimo são válidos, porém de qualidade inferior comparando-se com os do instante ótimo. Este entorno representa um intervalo no qual os dados podem ser capturados e considerados válidos. Os dados capturados antes ou após este entorno não possuem valor pois estão muito defasados e são inúteis ao controlador. Para este exemplo, as funções utilidade baseadas em deadline empregadas na literatura de tempo

*Bolsista de doutorado do CNPq

real para representar o benefício obtido em relação ao momento em que a tarefa apresenta seus resultados não modela satisfatoriamente a realidade. Existem situações em que a função incorretamente forneceria um valor alto de contribuição para o sistema, quando na verdade os dados capturados neste momento seriam inúteis.

Neste trabalho são descritos modelos de funções utilidade que podem ser facilmente enquadradas em aplicações típicas de controle. Nestas, o termo deadline tal como um tempo máximo de término de uma tarefa desaparece. Em seu lugar é utilizado o instante ideal d_i e um entorno onde os valores contribuem para o sistema. O instante ideal não precisa necessariamente ser o tempo de término de uma tarefa, ele pode representar o momento onde a tarefa manifesta sua saída ou o momento onde a tarefa amostra informações de um sensor. A localização do ponto dentro da tarefa, que deve ser executado no instante ideal, é denominado ponto de ação g_i. O tempo em que o ponto de ação executa efetivamente é denominado tempo de ação A_i. Para estes sistemas, o modelo definido teria grande benefício melhorando a qualidade e desempenho. A pesquisa da literatura da área não revelou nenhuma definição como esta. Os trabalhos que mais se aproximam deste objetivo buscam reduzir o *jitter* utilizando diferentes técnicas e serão examinados na seção 3. Na seção 2 será examinado o modelo de tarefas proposto. Na seção 4 serão mostrados os resultados de simulações de abordagens que se destacam na literatura sob a ótica do problema do instante ideal. Na seção 5 são apresentadas as conclusões.

2. Modelo proposto

Neste trabalho é suposto que uma aplicação é formada por um conjunto de *n* tarefas executando em um único processador. Tarefas são executadas segundo uma política de prioridades. Qualquer tarefa pode ser preemptada, ou seja, ter sua execução suspensa e retomada mais tarde. As tarefas podem experimentar situações de bloqueios devido a relações de exclusão mútua entre elas. Cada tarefa T_i, onde $1 \leq i \leq n$, é caracterizada por um máximo jitter de liberação J_i; tempo de execução no pior caso WCET (*Worst-Case Execution Time*) c_i e período (tarefa periódica) ou intervalo mínimo entre liberações (tarefa esporádica) P_i. A cada tarefa T_i está associado um instante ideal d_i, relativo a sua chegada, no qual a tarefa gera uma contribuição máxima para o sistema. Uma tarefa é dita viável se o seu tempo de ação A_i estiver dentro de um entorno $[s_i, l_i]$. Uma aplicação é viável quando todas as suas tarefas forem viáveis. O entorno $[s_i, l_i]$ representa o intervalo de interesse x_i tal que $s_i = d_i - x_i$ e $l_i = d_i + x_i$. A semântica da aplicação é considerada na definição de d_i e x_i. Por exemplo, x_i pode ser um intervalo de 10% em relação a d_i, sendo assim $x_i = (0.1)d_i$.

2.1. Tempo real *soft*

O modelo da função utilidade para tratar tarefas com requisitos de tempo real *soft* está ilustrado na figura 1. Supondo que o tempo de ação da k-ésima ativação da tarefa T_i é A_i^k, a contribuição desta ativação para o sistema é denotada V_i^k e dada por: $V_i^k = e^{-\left|A_i^k - d_i\right|}$. Observa-se que esta função tem seu máximo no instante ideal (representando a máxima contribuição para o sistema). A função assintoticamente aproxima-se do eixo *x*, visto que num sistema com requisitos *soft* tarefas que perdem seu deadline ainda possuem utilidade para o sistema.

2.2. Tempo real *firm*

O modelo da função utilidade para tratar tarefas com requisitos de tempo real *firm* está ilustrado na figura 2. Supondo que o tempo de ação da k-ésima ativação da tarefa T_i é A_i^k, a contribuição desta ativação para o sistema é denotada V_i^k e dada por: $V_i^k =$

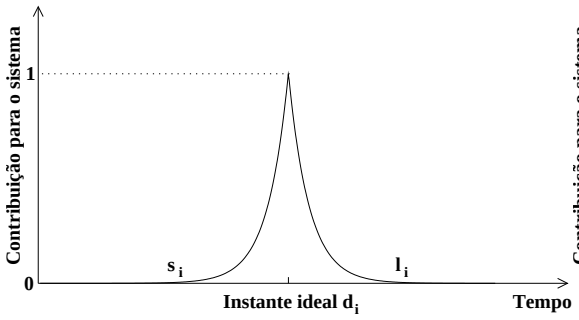

Figura 1: Função utilidade para requisitos temporais *soft*.

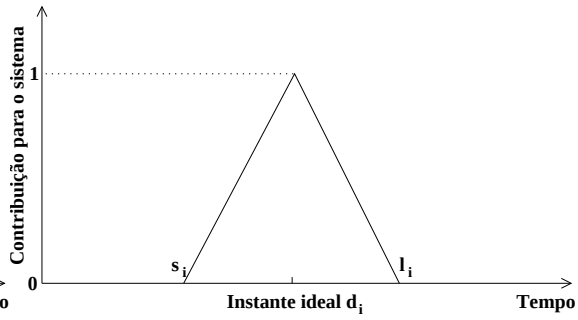

Figura 2: Função utilidade para requisitos temporais *firm*.

0 quando $A_i^k < s_i$ ou $A_i^k > l_i$, $V_i^k = 1 - \left|\frac{A_i^k - d_i}{d_i - s_i}\right|$ quando $s_i \leq A_i^k \leq l_i$ Observa-se que esta função tem seu máximo no instante ideal (representando a máxima contribuição para o sistema) e decresce linearmente dentro do entorno até atingir zero. Antes e após o entorno de interesse, a função tem valor nulo, caracterizando a situação em que uma tarefa perdeu o seu deadline, caso no qual não existe benefício no término da tarefa.

2.3. Tempo real *hard*

O modelo da função utilidade para tratar tarefas com requisitos de tempo real *hard* é uma variação da função utilidade para tarefas com requisitos *firm*. Supondo que o tempo de ação da k-ésima ativação da tarefa T_i é A_i^k, a contribuição desta ativação para o sistema é denotada V_i^k e dada por: $V_i^k = -\infty$ quando $A_i^k < s_i$ ou $A_i^k > l_i$, $V_i^k = 1 - \left|\frac{A_i^k - d_i}{d_i - s_i}\right|$ quando $s_i \leq A_i^k \leq l_i$ Esta função tem seu máximo no instante ideal (representando a máxima contribuição para o sistema) e decresce linearmente dentro do entorno até atingir zero. Antes e após o entorno de interesse, a função tem valor $-\infty$, caracterizando a situação em que uma tarefa com requisitos *hard* perdeu o seu deadline.

3. Abordagens encontradas na literatura

Na busca por soluções disponíveis na literatura para lidar com o problema do instante ideal temos: 1) Uso de escalonadores do tipo executivo cíclico; 2) Alteração do algoritmo do controlador 3) Minimização do *jitter* de saída através de ajustes dos parâmetros das tarefas, tipicamente o deadline; 4) Uso de modelos de tarefas alternativos que contribuam para reduzir o *jitter* de saída.

A abordagem 1) soluciona de forma trivial o problema do instante ideal, mas incorre em diversos problemas práticos em virtude da inflexibilidade deste método (H. Tokuda, 1987) (Locke, 1992). A abordagem 2) considera o projeto de um sistema de controle para lidar com o *jitter* de saída das tarefas. Esta solução é muito mais no sentido de projeto do sistema de controle do que de sistemas de tempo real e desta forma não será considerada (Lincoln, 2002). Neste trabalho, o interesse maior é pelas duas últimas abordagens pois tratam de problemas do domínio de tempo real e consideram o uso de escalonadores com prioridades. A abordagem 3) utiliza métodos para minimizar o *jitter* de saída de um sistema de tarefas escalonável segundo algum algoritmo. Faz uso de ajustes nos parâmetros das tarefas, tal como deadline. Desta forma obtém-se um novo sistema de tarefas também escalonável mas que apresenta um menor *jitter* de saída. Esta classe de solução é representada por trabalhos como (k. Baruah et al., 1999) e (kim et al., 2000). Em virtude de sua importância para este estudo, o trabalho (k. Baruah et al., 1999) será analisado mais profundamente. A última abordagem (4) é representada pelos trabalhos (Tindell, 1992),(Han e Lin, 1992) e (Tindell, 1994), o qual será analisado aqui.

3.1. Solução via controle de *jitter* de saída

No artigo (k. Baruah et al., 1999) é apresentado um algoritmo polinomial para minimizar o *jitter* de saída de um conjunto de tarefas que toma como entrada um conjunto de tarefas previamente escalonado com o EDF e produz como saída um novo conjunto de tarefas também escalonável, mas com um menor *jitter* de saída. Define-se como o *jitter* de saída a variação entre o tempo de conclusão de sucessivas ativações de uma mesma tarefa. Uma tarefa possui um intervalo mínimo e máximo p_i^{min} e p_i^{max} entre sucessivos términos. Numa escala de execução livre de *jitter*, essa variação não existe $p_i^{min} = p_i^{max} = p_i$.

No modelo de tarefas são contempladas tarefas periódicas, dadas por T_i e caracterizadas por um tempo de computação e_i um período p_i e um parâmetro ϕ que apresenta um elemento denominado tolerância de uma tarefa i ao *jitter* de saída. Esse último parâmetro possui a seguinte semântica: tarefas com um ϕ grande possuem grande tolerância ao *jitter*. Considera-se inicialmente que o deadline é igual ao período. Define-se o *jitter* absoluto de uma tarefa como sendo $AbsJitter(T_i) \stackrel{\text{def}}{=} max(p_i^{max} - p_i, p_i - p_i^{min})$ e o *jitter* absoluto de uma escala $AbsJitter(T_i) \stackrel{\text{def}}{=} \underset{T_i \in \Gamma}{max}\{AbsJitter(T_i)\}$. Além desses, defini-se como o *jitter* ponderado de uma tarefa e de uma escala de execução como sendo o *jitter* de uma tarefa divido pela tolerância ao *jitter* ϕ. Dessa forma temos: $WtdJitter(T_i) \stackrel{\text{def}}{=} \frac{AbsJitter(T_i)}{\phi}$ e $WtdJitter(\Gamma) \stackrel{\text{def}}{=} \underset{T_i \in \Gamma}{max} \frac{AbsJitter(T_i)}{\phi}$. Após este equacionamento o objetivo é minimizar a equação $WtdJitter(\Gamma)$. Antes, são calculados o *jitter* que uma tarefa T_i possui, sendo $Jitter(T_i) \leq (p_i - e_i)$. O que resulta em $WdtJitter(\Gamma) \leq \underset{T_i \in \Gamma}{max}\{\frac{p_i - e_i}{\phi_i}\} = \underset{T_i \in \Gamma}{max}\{\frac{e_i}{\phi_i}(\frac{1}{\rho_i} - 1)\}$. Um limite mais apertado pode ser obtido nos casos em que a utilização ρ das tarefas do sistema é menor do que 1, resultando em: $WdtJitter(\Gamma) \leq \underset{T_i \in \Gamma}{max}\{\frac{e_i}{\phi_i}(\frac{\rho}{\rho_i} - 1)\}$. Com o objetivo de minimizar a função de *jitter* absoluto de uma escala de execução $AbsJitter(\Gamma)$, substitui-se a parcela de processador utilizado por uma tarefa ρ_i por um θ_i, sendo $\theta_i \geq \rho_i$ para todas as tarefas. O objetivo é aumentar a utilização do sistema de tarefas, sendo $\sum_{i=1}^{n} \theta_i \leq 1$. O algoritmo de minimização escolhe novos valores para as parcelas de uso do processador para cada tarefa, sendo que esse novo valor deve ser maior ou igual ao ρ original. Utiliza-se uma pesquisa binária para encontrar um novo valor para o *jitter* mais apertado para o conjunto de tarefas. No final, obtém-se novos valores para os deadlines.

3.2. Solução via adição de *offsets*

O artigo (Tindell, 1994), descreve tarefas que fazem uso de *offsets*, um intervalo fixo entre chegadas de conjuntos de tarefas. Tarefas com *offsets* são utilizadas principalmente por três razões. Primeiramente para modelar aplicações nas quais o tempo de chegada de uma tarefa deve ser um valor fixo em relação ao inicio da tarefa anterior. Em segundo lugar para melhorar a escalonabilidade do sistema. Em geral os testes de escalonabilidade sempre levam em consideração a pior situação de escalonamento. Este instante crítico ocorre quando todas as tarefas são liberadas simultaneamente, resultando na maior carga no sistema. Se o sistema de tarefas é escalonável nesta situação limite, ele será escalonável sempre. Esta forma de testar o sistema de tarefas é pessimista, pois em situações onde tarefas apresentam *offsets* esse instante crítico é menor e seria possível escalonar um conjunto de tarefas que anteriormente seria recusado. Para isto, necessita-se de um teste que suporte o modelo de tarefas com *offsets*.

Offsets também podem limitar o *jitter* de saída, pois tarefas podem terminar antes de seu tempo limite para execução e a interferência sobre as demais tarefas pode ser menor que o tempo de execução de pior caso. Desta forma, uma tarefa pode apresentar grandes variações quanto ao momento de término. Uma solução para reduzir o *jitter* de

saída de uma tarefa A seria acrescentar uma tarefa B de prioridade mais alta. A tarefa A é responsável pela parte principal de computação e quando termina escreve seus resultados num *buffer* compartilhado com B. A tarefa B chega um tempo *offset* depois. A tarefa B tem prioridade mais alta e o tempo de resposta no pior caso da tarefa B será pequeno. Desta forma consegue-se uma variação menor no momento em que B termina e assim do par de tarefas como um todo. Para isso, deve-se respeitar algumas condições: A tarefa B deve iniciar depois que a tarefa A terminou $O_B > r_A$. O deadline do sistema como um todo deve ser respeitado $O_B + r_B \leq D_A$ sendo D_A o deadline original da tarefa A.

4. Simulações

Nesta seção são apresentados estudos empíricos para avaliar a adequação das abordagens da seção 3 sob a ótica do problema ideal.

4.1. Simulação da solução via controle de *jitter* de saída

Na seção 2 foram criadas métricas para expressar a contribuição de uma ativação de tarefa para o sistema em relação ao instante em que esta tarefa apresenta seus resultados ou amostra dados de sensores, segundo o modelo do instante ideal. Na simulação, estas métricas são coletadas para cada tarefa simulada e os resultados são apresentados na tabela 1. Nestas simulações, o ponto de ação foi posicionado a 50% do tempo de computação efetivo das tarefas e o instante ideal posicionado na metade do tempo de computação máximo da tarefa. Ativações em que o tempo de computação é menor que o WCET resultam num tempo de ação A_i anterior ao instante ideal. Atrasos na execução das tarefas em decorrência das relações de prioridade e de preempções resultam num deslocamento do tempo de ação para além do instante ideal. As métricas para tarefas *soft*, *firm* e *hard* levam em consideração essa diferença em relação ao instante ideal para determinar o valor de contribuição em função do tempo.

Nas tabelas 1 e 2 estão relacionados os resultados de uma simulação para tarefas geradas aleatoriamente com tempos de computação inteiros entre 1 e 10. Os períodos foram gerados aleatoriamente (números inteiros) para resultar numa carga (utilização) aproximada de cada tarefa entre 0.2 e 0.9. Para cada carga foram gerados e simulados 300 conjuntos formados por 5 tarefas por 50000 unidades de tempo. Para cada ativação de tarefa foram obtidos valores na forma de contribuição para o sistema considerando $x_i = (0.1)d_i$ (3 conjuntos de valores: *soft*, *firm* e *hard*). No final de cada tarefa simulada foi calculada a média aritmética das ativações executadas para se obter o valor de contribuição *soft*, *firm* e *hard* de cada tarefa. Para uma simulação com 5 tarefas foi calculada uma média resultando na contribuição para o sistema. Os resultados das tabelas são valores médios obtidos da forma descrita acima que informam a contribuição para o sistema em relação ao máximo possível (valores entre 0 e 1).

Embora o método de minimização possa reduzir o *jitter* de saída de um sistema de tarefas, isso não garante bons resultados para o instante ideal. As simulações realizadas não mostram nenhuma melhoria significativa na perspectiva do instante ideal utilizando a técnica de minimização quando o instante ideal não corresponde ao final da tarefa. Numa situação extrema em que o *jitter* de saída de uma tarefa seja nulo, temos que não existe variação entre seus tempos de término. Ainda assim, o instante de ação pode ocorrer num tempo variável e diferente do instante ideal. Esta situação pode ser representada como na figura 3 (tempo real *firm*). Desta forma, o uso de minimização não soluciona definitivamente o problema e a contribuição para o sistema não será máxima.

carga	soft	firm	hard
0.2	0.6290	0.3573	0.3573
0.3	0.5807	0.3098	0.3098
0.4	0.5343	0.2831	0.2831
0.5	0.4875	0.2368	0.2368
0.6	0.4344	0.2057	0.2057
0.7	0.3930	0.1791	0.1791
0.8	0.3556	0.1590	0.1590
0.9	0.3156	0.1278	0.1278

Tabela 1: Antes da otimização.

carga	soft	firm	hard
0.2	0.6423	0.3583	0.3583
0.3	0.5950	0.3164	0.3164
0.4	0.5387	0.2751	0.2751
0.5	0.4845	0.2297	0.2297
0.6	0.4377	0.1966	0.1966
0.7	0.3876	0.1791	0.1791
0.8	0.3414	0.1392	0.1392
0.9	0.3065	0.1236	0.1236

Tabela 2: Após a otimização.

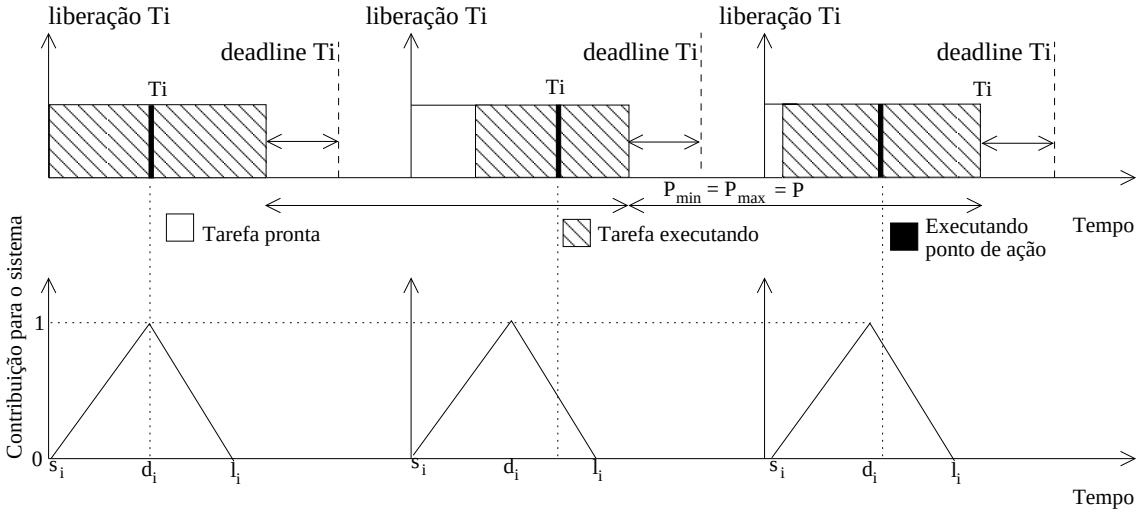

Figura 3: *Jitter* de saída nulo não soluciona o problema.

4.2. Simulação da solução via adição de *offsets*

O foco principal desta solução é reduzir o *jitter* de saída de uma tarefa fazendo uso da divisão em subtarefas e da utilização de *offsets*. Podemos modelar o problema do instante ideal considerando que a parte da tarefa em que está o ponto de ação deve ser tratada como uma subtarefa de alta prioridade. Desta forma, é possível reduzir a variação do tempo de ação. Como o ponto de ação pode estar localizado em qualquer posição da tarefa e não necessariamente no final, seria mais interessante que as transações fossem compostas por 3 subtarefas. Se fossem utilizadas apenas duas subtarefas, a segunda metade (do ponto de ação até o final) teria uma prioridade mais alta e poderia causar interferência nas demais subtarefas quando somente o ponto de ação precisa ocorrer no tempo correto. Em relação a subtarefa responsável pelo instante ideal, observamos que ela possui poucas instruções a executar e o fluxo de execução se mantém o mesmo, independentemente do estado da aplicação, então é possível considerar que esta tarefa possui um tempo de computação constante C_B.

Em muitos aspectos, a forma de modelar o problema do instante inicial é similar aos trabalhos (Crespo et al., 1999) e (Balbastre et al., 2000) principalmente por que neste problema também convêm dividir as tarefas em 3 subtarefas considerando aplicações de controle. No entanto, uma diferença está no objetivo final no qual devemos minimizar a variação de uma parcela da tarefa original (correspondente tempo de ação) sem a necessidade de cuidados em relação ao *jitter* de saída da tarefa.

A divisão em subtarefas apresenta alguns problemas pois a parte da tarefa que corresponde ao ponto de ação deve possuir prioridade maior e deve ser liberada após

passado um *offset* da chegada da transação. Esse *offset* deve ser calculado de forma que a parte anterior da tarefa já tenha terminado $O_B > r_A$. Desta forma, torna-se necessário calcular o tempo de resposta da parte inicial da tarefa $r_A = C_A + \sum_{i \epsilon hp(T_A)} C_i \lceil \frac{r_A}{P_i} \rceil$. O valor do *offset* calculado representa um tempo mínimo para o inicio da subtarefa B e no caso em que o *offset* (tempo de resposta de A) seja menor que o instante ideal, podemos considerar o instante ideal como novo *offset* $O_B = max(r_A, \text{instante ideal})$.

Infelizmente, no caso em que o tempo de resposta da subtarefa A seja maior que o instante ideal, não é possível atingir o objetivo desejado. Além disso, considerando-se que o mesmo conjunto de tarefas da primeira simulação foi utilizado (com o EDF como escalonador), espera-se que algumas desses conjuntos sejam não escalonáveis nesta nova simulação. Como o objetivo é avaliar o uso de *offsets* para o problema do instante ideal, a violação de deadlines torna-se irrelevante.

Para associar as prioridades às subtarefas, escolhemos agrupa-las segundo grupos de prioridades. As subtarefas que representam a parte B pertencem ao grupo de prioridade mais alto. A parte A pertence ao grupo de prioridade médio e a prioridade mais baixa é do grupo no qual está a parte C. Dentro de cada grupo é utilizada a política *Deadline Monotonic* para a atribuição de prioridades. Para evitar que a parte B inicie antes da parte A existe o *offset* que garante que a tarefa A já tenha terminado. Para evitar que a parte C inicie antes da parte B escolheu-se utilizar o mesmo *offset* de B. Como B possui prioridade sempre mais alta que a tarefa C, a subtarefa B ganha o processador antes de C.

Numa primeira simulação (ilustrada na tabela 3), o ponto de ação foi posicionado a 50% do tempo de computação efetivo das tarefas e o instante ideal posicionado na metade do tempo de computação máximo da tarefa, todos os demais parâmetros estão consistentes com os da simulação anterior. Intuitivamente podemos imaginar que a técnica de subdivisão de tarefas e uso do *offset* resultaria em valores de contribuição para o sistema elevados. Infelizmente, a tarefa no qual o ponto de ação está localizado somente é liberada depois de um tempo calculado como o máximo entre o valor do instante ideal e o tempo de resposta da subtarefa anterior. O cenário mais favorável é quando o instante ideal não é menor que o tempo de resposta da subtarefa A. Além deste, um caminho que resultaria em benefícios para o sistema seria utilizar métodos para cálculo do tempo de resposta de pior caso que fossem menos pessimistas. Na literatura de tempo real existem trabalhos para análise do tempo de resposta em sistemas de tarefas que fazem uso de *offsets*. Em sistemas com *offsets* o instante crítico é relaxado, pois somente algumas tarefas estarão liberadas para rodar, resultando numa interferência menor nas tarefas e num tempo de resposta menos pessimista (Tindell, 1992), (Gutierrez e Harbour, 1998) e (Mäki-Turja e Nolin, 2004).

Na segunda simulação, o ponto de ação foi posicionada a 50% do tempo de computação efetivo e o instante ideal posicionado a $3WCET$ da tarefa. Com o instante ideal deslocado, o tempo de resposta da subtarefa A não é mais o fator dominante para calcular o *offset* da subtarefa B. Desta forma, o tempo de ação da subtarefa B aproxima-se do valor do instante ideal, resultando numa contribuição alta tal como mostrada na tabela 4. Interferências causadas por tarefas de maior prioridade reduzem a contribuição para o sistema impedindo que esta seja máxima.

5. Conclusão

Neste trabalho foi proposto um novo modelo de tarefas baseado no instante ideal. O modelo proposto seria benéfico tipicamente em aplicações de controle melhorando a qualidade e desempenho do controlador. Não é do conhecimento dos autores a definição de um modelo de tarefas que trate diretamente o problema do instante ideal como o exposto

carga	soft	firm	hard
0.20	0.0320	$-\infty$	$-\infty$
0.30	0.0335	$-\infty$	$-\infty$
0.40	0.0324	$-\infty$	$-\infty$
0.50	0.0321	$-\infty$	$-\infty$
0.60	0.0331	$-\infty$	$-\infty$
0.70	0.0329	$-\infty$	$-\infty$
0.80	0.0322	$-\infty$	$-\infty$
0.90	0.0324	$-\infty$	$-\infty$

Tabela 3: Primeira simulação.

carga	soft	firm	hard
0.2	0.8989	0.8766	0.8766
0.3	0.8971	0.8774	0.8774
0.4	0.8977	0.8761	0.8761
0.5	0.9113	0.8887	0.8887
0.6	0.9069	0.8852	0.8852
0.7	0.9120	0.8925	0.8925
0.8	0.9029	0.8811	0.8811
0.9	0.9003	0.8753	0.8753

Tabela 4: Segunda simulação.

no trabalho. Foram feitas simulações com abordagens propícias existentes na literatura considerando a ótica do problema do instante ideal. Os resultados das simulações mostram que as abordagens existentes na literatura não solucionam completamente o problema, criando a necessidade de desenvolvimento de novos algoritmos de escalonamento específicos para o problema apresentado.

Referências

Balbastre, P., Ripoll, I., e Crespo, A. (2000). Control tasks delay reduction under static and dynamic scheduling policies. In *Seventh International Conference on Real-Time Systems and Applications (RTCSA'00)*, páginas 522–526.

Crespo, A., Ripoll, I., e Albertos, P. (1999). Reducing delays in rt control: the control action interval. In *14th IFAC World Congress on Automatic Control*. Elsevier Science.

Gutierrez, J. P. e Harbour, M. G. (1998). Schedulability analysis for tasks with static and dynamic offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)* .

H. Tokuda, J.W. Wendorf, H. W. (1987). Implementation of a time-driven scheduler for real-time operating systems. In *Proc. of the IEEE Real-Time Systems Symposium*, páginas 271–280.

Han, C. e Lin, K. (1992). Scheduling distance-constrained real-time tasks. *Proceedings of the Real-Time Systems Symposium*, páginas 300–308.

k. Baruah, S., Buttazzo, G., Gorinsky, S., e Lipari, G. (1999). Scheduling periodic task systems to minimize output jitter. In *Sixth International Conference on Real-Time Computing Systems and Applications*, páginas 62–66.

kim, T., heonshik Shin, e Chang, N. (2000). Deadline assignment to reduce output jitter of real-time tasks. *Proc. of The 16th IFAC Workshop on Distributed Computer Control Systems*, páginas 67–72.

Lincoln, B. (2002). Jitter compensation in digital control systems. In *Proceedings of the 2002 American Control Conference*.

Locke, C. D. (1992). Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives. *Real-Time Systems*, 4(1):37–53.

Mäki-Turja, J. e Nolin, M. (2004). Tighter response-times for tasks with offsets. *Real-time and Embedded Computing Systems and Applications Conference (RTCSA)*.

Tindell, K. (1992). Using offset information to analyse static priority pre-emptively scheduled task sets. Technical report, University of York, YCS-92.

Tindell, K. (1994). Adding time-offsets to schedulability analysis. Technical report, University of York, Computer Science Dept, YCS-94-221.