Escalonando Tarefas Imprecisas com Restrição 0/1 e Dependências Intra-Tarefa / Inter-Tarefa

Rômulo Silva Oliveira

Instituto de Informática
Univ. Fed. do Rio Grande do Sul
Caixa Postal 15064
Porto Alegre-RS, 91501-970
romulo@inf.ufrgs.br

Joni Silva Fraga

Lab. de Controle e Microinformática
Univ. Fed. de Santa Catarina
Caixa Postal 476
Florianópolis-SC, CEP 88040-900
fraga@lcmi.ufsc.br

Resumo

Este trabalho apresenta um conjunto de heurísticas para serem usadas como política de admissão quando tarefas imprecisas são empregadas. É suposto que as tarefas imprecisas são programadas com a técnica de múltiplas versões e possuem dependências intra-tarefa e inter-tarefa. Estas heurísticas deverão ser usadas em conjunto com testes off-line de escalonabilidade e testes on-line de aceitação existentes. O teste off-line garante que a parte obrigatória de cada tarefa será sempre completada antes do seu deadline. O teste on-line de aceitação verifica se a execução de uma dada parte opcional poderia comprometer a execução das partes obrigatórias. O objetivo da política de admissão é maximizar a utilidade do sistema através da seleção de partes opcionais para execução. As heurísticas propostas são analisadas através de simulação.

Abstract

This work presents a set of heuristics to be used as admission policy when the system is made of imprecise tasks. We assume that imprecise tasks have intra-task and inter-task dependences and are programmed with the multiple versions technique. These heuristics are supposed to be used combined with off-line schedulability tests and on-line acceptance tests already described in the literature. The off-line test will guarantee that at least the mandatory part of each task will always be finished before its deadline. The on-line acceptance test checks if the execution of a specific optional part would jeopardize the execution of mandatory parts. The objective of the admission policy is to maximize the system utility through the selection of optional parts for execution. Simulation is used to analyse the proposed heuristics.

1. Introdução

Sistemas computacionais de tempo real (STR) são identificados como aqueles submetidos a requisitos de natureza temporal. Nestes sistemas, os resultados devem estar corretos não somente do ponto de vista lógico, mas também devem ser gerados no momento correto. As falhas de natureza temporal nestes sistemas são, em alguns casos, consideradas críticas no que diz respeito às suas consequências.

Um problema básico encontrado na construção de sistemas de tempo real é a alocação e o escalonamento das tarefas nos recursos computacionais disponíveis. Existe

uma dificuldade intrínsica em compatibilizar dois objetivos fundamentais ([BUR 91]): garantir que os resultados serão produzidos no momento desejado e dotar o sistema de flexibilidade para adaptar-se a um ambiente dinâmico e, assim, aumentar sua utilidade.

Tem sido proposto na literatura ([AUD 94a], [DAV 95], [TIA 95]) uma abordagem combinada integrando algoritmos off-line e on-line para alcançar simultaneamente garantia para as tarefas críticas e flexibilidade para melhorar a utilidade do sistema. Tipicamente, um teste de escalonabilidade é usado off-line para garantir que certas tarefas atenderão seus deadlines em um cenário de pior caso. Um teste de aceitação é então usado em tempo de execução para aceitar tarefas adicionais que irão aumentar a utilidade do sistema, sem sacrificar tarefas previamente garantidas.

É possível que, em determinados momentos, o conjunto de tarefas adicionais disponível para execução seja maior que aquilo que pode ser aceito. Então, uma política de admissão é usada para definir quais, entre as novas tarefas propostas, serão oferecidas ao teste de aceitação. As tarefas que passarem pela política de admissão e pelo teste de aceitação serão executadas. A política de admissão funciona como um filtro que, colocado antes do teste de aceitação, descarta as tarefas adicionais que acrescentam menos utilidade ao sistema.

A técnica de computação imprecisa ([LIU 94]) permite uma certa flexibilidade no escalonamento de sistemas de tempo real. Tarefas são compostas por uma parte obrigatória e uma parte opcional. Em caso de sobrecarga, partes opcionais poderão não ser executadas. Cada tarefa que é executada completamente adiciona algum valor ao sistema, isto é, aumenta sua utilidade.

Uma tarefa imprecisa pode ser implementada com a técnica de múltiplas versões. Em sua forma mais simples, cada tarefa imprecisa possui duas versões. A versão primária é capaz de gerar um resultado preciso, mas tem um maior tempo de execução no pior caso. A versão secundária é capaz de gerar um resultado aceitável (impreciso), com um menor tempo de execução no pior caso. A versão secundária corresponde a parte obrigatória da tarefa. A diferença entre as duas versões define a parte opcional. Quando múltiplas versões são usadas, é necessário decidir qual versão de uma tarefa executará antes da tarefa começar. Uma vez que a tarefa iniciou sua execução não é mais possível rever aquela decisão. Temos também que a parte opcional será completamente executada (quando a versão primária é escolhida) ou não será executada (quando a versão secundária é escolhida). Neste caso é dito que a tarefa possui uma restrição do tipo 0/1.

A maioria dos estudos de computação imprecisa consideram que o valor de uma tarefa é definido no momento que a tarefa chega. É suposto que ela não é afetada pelo que acontece a outras tarefas ou a liberações prévias da mesma tarefa. Os trabalhos [FEN 94] e [CHU 90] são uma exceção. Em [FEN 94] o modelo de tarefas inclui dependências inter-tarefa. Erros nos dados de entrada de uma tarefa podem estender o tempo máximo de execução de suas próprias partes obrigatórias e/ou opcionais. Exemplos de tarefas que apresentam dependências inter-tarefa aparecem em aplicações tais como reconhecimento de voz, sistemas de radar e processamento de imagem para navegação autônoma ([FEN 94], [KRO 94]). Em [CHU 90] o modelo de tarefas inclui

dependências intra-tarefa. É suposto que todas as tarefas devem ser executadas precisamente pelo menos uma vez a cada **Q** liberações. Exemplos de tarefas com dependências intra-tarefa aparecem em aplicações tais como sistemas de radar e sistemas de controle ([CHU 90]). Nos dois trabalhos citados o modelo de erro é tal que uma parte opcional poderia eventualmente tornar-se obrigatória ou aumentar o tempo global de execução obrigatória do sistema, perdendo assim seu carater "opcional".

Este trabalho apresenta um conjunto de heurísticas para serem usadas como política de admissão quando tarefas imprecisas são empregadas. Estas heurísticas deverão ser usadas em conjunto com testes off-line de escalonabilidade e testes on-line de aceitação existentes. O teste off-line garante que ao menos a parte obrigatória de cada tarefa será sempre completada antes do seu deadline. O teste on-line de aceitação verifica se a execução de uma dada parte opcional vai ou não comprometer a execução das partes obrigatórias. O objetivo da política de admissão é maximizar a utilidade do sistema através da seleção de partes opcionais para a execução. Com respeito a este objetivo, as heurísticas propostas são analisadas através de simulação.

Neste artigo é ainda suposto que as tarefas imprecisas possuem dependências intra-tarefa e inter-tarefa. Este estudo se limita a técnica de múltiplas versões onde duas versões são usadas na programação de tarefas imprecisas. Diferentemente do trabalho em [FEN 94] e [CHU 90], as partes opcionais no modelo usado permanecem sempre opcionais. Mesmo se nunca executadas, elas não aumentam o tempo de execução obrigatória de qualquer tarefa do sistema nem tornam-se, em qualquer sentido, obrigatórias. Uma versão simplificada deste artigo, descrevendo resultados preliminares, foi apresentada em [OLI 96]. Não é do conhecimento dos autores outros trabalhos onde tarefas imprecisas, com o tipo de dependência apresentado aqui, tenham sido estudadas.

O restante do artigo está organizado da seguinte forma: a seção 2 descreve o modelo de tarefas e o problema de escalonamento a ser resolvido. A seção 3 descreve as heurísticas propostas neste trabalho como políticas de admissão. Descreve também duas heurísticas conhecidas que serão usadas para fins de comparação. A seção 4 descreve como foram feitas as simulações que avaliam o desempenho das diferentes políticas de admissão. A seção 5 mostra os resultados das simulações. Finalmente, na seção 6 são feitos os comentários finais sobre o trabalho.

2. Formulação do Problema

É suposto um sistema com N tarefas periódicas ou esporádicas que executam em um único processador. Cada tarefa T_i é a fonte de um número infinito de liberações. Cada liberação de T_i é composta por uma parte obrigatória e uma parte opcional. O tempo de execução no pior caso de ambas as partes é conhecido off-line e denotado por M_i e O_i, respectivamente. É também suposto que existe uma restrição 0/1 para as partes opcionais. Isto é, cada parte opcional deve ser executada completamente ou não iniciada. Esta decisão deve ser tomada antes de iniciar a execução da tarefa.

Toda tarefa T_i possui um período (tarefa periódica) ou um intervalo mínimo entre liberações (tarefa esporádica) denotado por P_i. Também possui um deadline D_i,

relativo ao instante da liberação. É suposto que $\forall T_i$, $D_i \leq P_i$. É necessário garantir off-line que a parte obrigatória de cada tarefa é sempre concluída antes do deadline. Uma parte opcional somente adiciona valor ao sistema caso sua execução seja concluída antes do respectivo deadline.

As tarefas do sistema possuem uma prioridade fixa única. Sem perda de generalidade, vamos supor que as tarefas foram nomeadas de tal forma que T_i possui prioridade superior a T_j sempre que $i<j$. Vamos usar $hp(T_i)$ para denotar o conjunto de tarefas com prioridade superior a T_i. Vamos também usar $lp(T_i)$ para denotar o conjunto de tarefas com prioridade igual ou inferior a T_i.

É suposto que cada tarefa T_i está associada com um valor nominal V_i. O valor nominal V_i representa a utilidade da tarefa T_i no sistema. O valor nominal não considera as dependências entre tarefas. Em tempo de execução, a cada liberação $T_{i,k}$ da tarefa T_i é associado um valor efetivo $V_{i,k}$. O valor efetivo é calculado a partir do valor nominal e considera as dependências entre tarefas. O valor adicionado por $T_{i,k}$ ao sistema será zero no caso de uma execução imprecisa ou $V_{i,k}$ quando a parte opcional for executada. O objetivo geral da solução de escalonamento é maximizar o valor total do sistema, dado pelo somatório dos valores adicionados por todas as liberações de todas as tarefas.

Ao longo deste trabalho vamos empregar $V(t)$ para denotar o valor total do sistema até o instante t. O valor $\Phi(t)$ representa o tempo total de processador, dentro do intervalo [0,t), que não foi usado para executar partes obrigatórias. A densidade média de valor $\Lambda(t)$, obtida pelo sistema até o instante t, será calculada por: $\Lambda(t) = V(t) / \Phi(t)$.

A dependência intra-tarefa entre as liberações $T_{i,k}$ e $T_{i,k+1}$, da tarefa T_i, é modelada assumindo-se que uma execução imprecisa de $T_{i,k}$ vai aumentar o valor efetivo de uma execução precisa de $T_{i,k+1}$. O valor efetivo da liberação $T_{i,k+1}$, denotado por $V_{i,k+1}$, será:

V_i	quando a execução de $T_{i,k}$ é precisa;
$V_i + (\alpha_i . V_{i,k})$	quando a execução de $T_{i,k}$ é imprecisa;

onde α_i , $0 \leq \alpha_i \leq 1$ é a taxa de recuperação da tarefa T_i. Ela descreve a parcela de valor que pode ser recuperada após uma dada liberação não ter sido executada precisamente.

Com respeito as dependências inter-tarefa, vamos supor que estas surgem em subconjuntos de tarefas que são liberadas sempre ao mesmo tempo. As diferentes prioridades estabelecem uma ordem de execução ou relação de precedência dentro do subconjunto. Desta forma, uma tarefa de prioridade superior é executada antes e poderá interferir na execução das tarefas do subconjunto que possuem prioridades inferiores.

A dependência inter-tarefa entre as liberações $T_{j,k}$ e $T_{i,k}$ é modelada assumindo-se que uma execução precisa de $T_{j,k}$ reduzirá o tempo de execução no pior caso das partes obrigatória e opcional de $T_{i,k}$. Em outras palavras, o tempo de execução de $T_{i,k}$ no pior caso será:

$M_i + O_i$	quando a execução de $T_{j,k}$ é imprecisa;
$\beta_{j,i} . M_i + \gamma_{j,i} . O_i$	quando a execução de $T_{j,k}$ é precisa;

onde $\beta_{j,i}$ e $\gamma_{j,i}$, $0<\beta_{j,i}\leq 1$, $0<\gamma_{j,i}\leq 1$, são os fatores de redução ligando T_j a T_i. Caso o tempo de execução de T_i dependa de mais de uma tarefa, é necessário aplicar os fatores de redução de todas as tarefas que interferem em T_i e executaram precisamente.

Considerando a liberação $T_{i,k}$ da tarefa T_i, a densidade de valor $\lambda_{i,k}$ de sua parte opcional é definida como o seu valor efetivo dividido pelo seu tempo de execução no pior caso, após as correções em função das dependências inter-tarefa. Temos então:

$$\lambda_{i,k} = \frac{V_{i,k}}{O_i \times \prod_{\forall j | T_j \in Pred(T_{i,k})} \gamma_{j,i}},$$

onde $Pred(T_{i,k})$ representa o conjunto de tarefas que interferem sobre o tempo máximo de execução de T_i e cuja k-ésima liberação foi executada completamente. O produtório no denominador reflete o fato de que várias tarefas podem preceder T_i e afetar seu comportamento.

É importante observar que nenhum dos dois tipos de dependência definidos é capaz de aumentar a carga obrigatória do sistema. A dependência intra-tarefa resulta em aumento no valor efetivo de algumas partes opcionais. A dependência inter-tarefa resulta na redução do tempo máximo de execução de algumas tarefas. De qualquer modo, a carga obrigatória do sistema não aumenta.

3. Descrição das Heurísticas

Nesta seção serão descritas diversas heurísticas que podem ser empregadas como política de admissão no modelo de tarefas considerado. As duas primeiras políticas (FCFS e AVDT) já foram descritas em [DAV 95] e serão simuladas para fins de comparação. As outras duas políticas são originais e seu desempenho será comparado com as duas primeiras em diferentes tipos de carga.

3.1 Ordem de Chegada (FCFS)

Nesta política todas as partes opcionais são sempre consideradas para execução. Quando uma tarefa vai iniciar sua execução, o teste de aceitação é sempre aplicado sobre sua parte opcional. Caso ela seja aceita, então a versão precisa da tarefa é executada. Note que esta política de admissão não considera o fato das tarefas possuirem diferentes valores, refletindo diferentes graus de utilidade para o sistema.

3.2 Adaptive Value Density Threshold (AVDT)

Nesta política o suporte de execução mantém atualizado o valor $\Lambda(t)$, ou seja, a densidade média de valor obtida pelo sistema até o momento. O valor $\Lambda(t)$ é usado como um limite mínimo para a densidade de valor das partes opcionais. Somente partes opcionais que possuem uma densidade de valor $\lambda_{i,k}$ maior ou igual a $\Lambda(t)$ serão consideradas. Entre elas, serão executadas aquelas que passarem no teste de aceitação.

3.3 Compensated Value Density Threshold (CVDT)

Esta política de admissão, introduzida no presente texto, emprega o mesmo princípio do AVDT. Como antes, somente partes opcionais que possuem densidade de valor $\lambda_{i,k}$ igual ou superior a um determinado limite mínimo serão consideradas para execução. Deste conjunto, aquelas que passarem no teste de aceitação serão executadas. Entretanto, o limite mínimo utilizado é diferente daquele empregado no AVDT.

O limite mínimo utilizado no CVDT, denotado por ζ, é dado por:

$$\zeta = \Lambda(t) \times \mathrm{Min}(5 \times \pi(t), 1.1), \qquad \text{[Eq. 1]}$$

onde $\pi(t)$ representa a taxa de rejeição do teste de aceitação, ou seja, o número de partes opcionais rejeitadas dividido pelo número de partes opcionais submetidas ao teste.

Como antes, a densidade média de valor obtida pelo sistema $\Lambda(t)$ é usada. Mas agora ela é multiplicada por um fator de correção que leva em consideração a taxa de rejeição do teste de aceitação. Quando a taxa de rejeição for 20%, o próprio $\Lambda(t)$ é usado. Quando a taxa de rejeição for maior que 20%, indicando que a política de admissão está deixando passar muitos candidatos, o valor limite aumenta. Quando a taxa de rejeição é menor que 20%, indicando portanto que o teste de aceitação está recebendo poucos candidatos, o valor limite diminui. O valor 1.1 aparece na equação [Eq.1] apenas para limitar o efeito do multiplicador. Desta forma, o fator de correção poderá variar entre zero (quando $\pi(t)=0$) e 1.1 (quando $5 \times \pi(t) \geq 1.1$).

Os valores 1.1 como limitante e 20% como taxa de rejeição ideal foram escolhidos com base em diversas experiências. É possível que, para diferentes tipos de aplicações, um ajuste fino destes valores resultasse em melhor desempenho do sistema.

3.4 Compensated Value Density Threshold for Inter-Task Dependences (INTER)

Esta política é semelhante a anterior no sentido que compara uma densidade de valor com um limite mínimo. Como antes, somente partes opcionais que possuem uma densidade igual ou superior ao limite mínimo serão consideradas para execução. A política INTER emprega o mesmo limite mínimo ζ, dado pela equação [Eq.1].

No momento que a liberação $T_{i,k}$ vai iniciar sua execução, o limite mínimo ζ é comparado com a densidade de valor $\varepsilon_{i,k}$, dada por:

$$\varepsilon_{i,k} = \frac{V_{i,k} + 0.5 \times \zeta \times \sum_{\forall j | T_j \in Succ(T_i)} [(1 - \beta_{i,j}) \times M_j \times \prod_{\forall h | T_h \in Pred(T_{j,k})} \beta_{h,j}]}{O_i \times \prod_{\forall h | T_h \in Pred(T_{i,k})} \gamma_{h,i}},$$

ou ainda, dada pela equação [Eq. 2] abaixo:

$$\varepsilon_{i,k} = \lambda_{i,k} + \frac{0.5 \times \zeta \times \sum_{\forall j | T_j \in Succ(T_i)} [(1 - \beta_{i,j}) \times M_j \times \prod_{\forall h | T_h \in Pred(T_{j,k})} \beta_{h,j}]}{O_i \times \prod_{\forall h | T_h \in Pred(T_{i,k})} \gamma_{h,i}},$$

onde $V_{i,k}$ representa o valor efetivo associado com $T_{i,k}$, ζ é o limite mínimo dado pela equação [Eq.1], $Succ(T_i)$ representa o conjunto de tarefas cujo tempo máximo de execução pode ser afetado por T_i, $Pred(T_{i,k})$ e $Pred(T_{j,k})$ representam os conjuntos de tarefas que interferem sobre o tempo máximo de execução de, respectivamente, T_i e T_j, e cuja k-ésima liberação foi executada completamente.

No cálculo da densidade $\varepsilon_{i,k}$ é considerado o fato de uma execução precisa de $T_{i,k}$ reduzir o tempo de execução obrigatória das suas tarefas sucessoras. O valor que uma execução precisa de $T_{i,k}$ contribui para o sistema é ampliado pela provável execução futura de outras partes opcionais. Estas futuras execuções aproveitariam a redução no tempo máximo de execução obrigatória das tarefas sucessoras de $T_{i,k}$. A equação [Eq.2] supõe uma densidade de valor $0.5\times\zeta$ para tais execuções futuras.

A equação [Eq.2] leva também em consideração o fato de uma tarefa poder possuir mais de um predecessor. Assim, a expressão

$$M_j \times \prod_{\forall h | T_h \in Pred(T_{j,k})} \beta_{h,j}$$

representa o tempo máximo de execução obrigatória da liberação $T_{j,k}$, após as correções em função dos predecessores de T_j que executaram completamente.

4. Simulação

As heurísticas descritas na seção anterior são avaliadas através de simulação. Desta forma, o desempenho das heurísticas propostas neste artigo será comparado com o desempenho de duas políticas existentes, isto é, FCFS e AVDT. Para efeitos da simulação é necessário definir qual será o teste off-line de escalonabilidade usado para garantir que as partes obrigatórias são escalonáveis. Também é necessário definir qual será o teste de aceitação on-line a ser empregado. Cabe ressaltar que as heurísticas apresentadas na seção anterior não estão atreladas especificamente aos testes usados na simulação. Outros testes de escalonabilidade e de aceitação poderiam ser usados.

4.1 Teste Off-Line de Escalonabilidade

Neste trabalho é assumido que as tarefas recebem uma prioridade fixa segundo a política do deadline monotônico [LEU 82]. É usado também o teste de escalonabilidade exato proposto em [AUD 93] para calcular o tempo máximo de resposta R_i de cada tarefa T_i. O valor R_i é calculado iterativamente, a partir de um valor inicial $R_i^0=C_i$, pela equação:

$$R_i^{k+1} = C_i + \sum_{\forall T_j \in hp(T_i)} \left\lceil R_i^k \div P_j \right\rceil \times C_j \ , \qquad \text{[Eq. 3]}$$

onde $hp(T_i)$ é o conjunto de tarefas com prioridades maiores que T_i. A iteração termina quando $R_i^{k+1} = R_i^k$ ou então quando $R_i^k > D_i$. A tarefa T_i será escalonável se $R_i \le D_i$. É importante observar que apenas as partes obrigatórias recebem uma garantia off-line. Logo, a equação [Eq.3] utiliza $C_i = M_i$.

4.2 Teste On-Line de Aceitação

No modelo de tarefas adotado, sempre que uma tarefa vai iniciar sua execução ocorre uma solicitação de garantia para a sua parte opcional. Esta solicitação possui tempo de execução O_i e deadline igual ao deadline da parte obrigatória. Caso a parte opcional passe pela política de admissão, ela vai para o teste de aceitação.

Como em [DAV 95] vamos usar o teste de aceitação baseado no "Dynamic Approximate Slack Stealing" (DASS), descrito em [AUD 94b]. Este algorítmo mantém contadores de folga $S_i(t)$ para cada nível de prioridade. No momento de aceitar ou não uma parte opcional, estes contadores são consultados. A parte opcional de uma tarefa T_i poderá ser aceita caso $\forall j \in lp(i)$, $O_i \leq S_j(t)$. Caso a parte opcional O_i seja aceita, é necessário corrigir a tabela de folgas fazendo: $\forall j \in lp(i)$, $S_j(t) \leftarrow S_j(t) - O_i$.

Observe que a garantia para a parte opcional de uma liberação $T_{i,k}$ é dada no início da execução da tarefa e não no instante da liberação. Isto é feito para aproveitar, no teste de aceitação, as folgas ("slack") geradas pelas tarefas de maior prioridade que executam e são concluidas exatamente entre a liberação e o início de $T_{i,k}$. Esta garantia dada no início da execução da tarefa não pode ser revogada, dentro do modelo das múltiplas versões.

4.3 Carga de Tarefas

A carga de tarefas foi definida de maneira semelhante a outros trabalhos presentes na literatura ([AUD 94b], [DAV 95]). Para efeito de simulação, foi desprezado o custo ("overhead") dos testes de aceitação e da política de admissão. É importante notar que todas as políticas de admissão simuladas empregam o mesmo teste de aceitação. Logo, este custo em particular é o mesmo para todas.

Foram empregados conjuntos de 18 tarefas periódicas em todos os experimentos. Nestes conjuntos, 6 tarefas possuem período entre 20 e 200 ticks, outras 6 possuem período entre 200 e 2000 e as 6 tarefas restantes possuem período entre 2000 e 20000. Os deadlines das tarefas foram gerados de forma aleatória, a partir de uma distribuição uniforme entre 20 e o período da tarefa em questão.

Os tempos máximos de computação obrigatória foram gerados também aleatoriamente, mas de maneira que a carga obrigatória total fosse de 30%, 60% ou 90%. Os tempos máximos de computação opcional igualmente foram gerados de forma aleatória, mas de tal sorte que a carga total no processador variasse de 60% a 300%. Não existe uma relação previamente definida entre o tempo máximo de execução da parte opcional O_i da tarefa T_i e o seu tempo máximo de execução obrigatória M_i.

Os valores V_i das partes opcionais foram estabelecidos também aleatoriamente seguindo uma distribuição uniforme entre 1 e 10. Tanto as taxas de recuperação α_i quanto os fatores de redução $\beta_{i,j}$ e $\gamma_{i,j}$ das tarefas foram escolhidos segundo uma distribuição uniforme entre 0 e 1. Para avaliar o impacto da dependência inter-tarefa, foi suposto que as 18 tarefas do conjunto estão divididas em 6 triplas. Tarefas pertencentes a uma mesma tripla possuem período igual e dependência inter-tarefa.

Somente foram considerados conjuntos de tarefas cujas partes obrigatórias podiam ser garantidas off-line. Para cada combinação de carga obrigatória e opcional foram simulados 20 conjuntos de tarefas diferentes. Os resultados apresentados correspondem a média destas 20 execuções.

5. Resultados da Simulação

Nesta seção serão apresentados os resultados das simulações realizadas. As tabelas de 1 a 9 resumem estes resultados. A figura de mérito observada é o valor total obtido pela heurística em questão em relação ao valor total obtido por FCFS, para a mesma carga de tarefas. Por exemplo, um valor 1.5 significa que a heurística em questão obteve 50% a mais de valor do que quando FCFS for usado.

O custo dos algoritmos não foi incluido nas simulações realizadas. Como dito antes, todos utilizam o mesmo teste de aceitação. Com respeito ao custo ("overhead") das políticas de admissão, é importante notar que os valores efetivos e os tempos máximos de execução podem ser mantidos pelo suporte de execução. Eles seriam atualizados sempre que uma tarefa é executada precisamente. Desta forma, todos os produtórios que aparecem nas equações já estariam calculados no momento de executar a política de admissão. Neste caso, FCFS, AVDT e CVDT teriam uma complexidade O(1). Já INTER possuiria uma complexidade O(N) em função do somatório que aparece no numerador. No caso extremo, todas as tarefas são sucessoras de T_i e podem ser afetadas por sua execução precisa.

As tabelas 1, 2 e 3 descrevem os resultados obtidos quando existe apenas dependência intra-tarefa. Nesta situação, CVDT e INTER apresentam o mesmo comportamento. Isto é esperado pois na falta de dependência inter-tarefa temos sempre que $\varepsilon_{i,k}=\lambda_{i,k}$ e as duas heurísticas ficam iguais. Em todas as combinações de carga obrigatória e opcional estudadas, CVDT e INTER foram superiores a FCFS e AVDT. O melhor desempenho de CVDT e INTER com relação a AVDT está na capacidade daqueles em detectar quando o processador está subutilizado e então baixar o limite mínimo de forma adequada.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%	270%
AVDT	0.961	0.946	1.052	1.170	1.281	1.396	1.460	1.567	1.583
CVDT	1.000	1.008	1.139	1.291	1.389	1.543	1.621	1.766	1.731
INTER	1.000	1.008	1.139	1.291	1.389	1.543	1.621	1.766	1.731

Tabela 1 - Carga obrigatória de 30% com dependência intra-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%
AVDT	0.964	1.167	1.370	1.504	1.589	1.763	1.648	1.732
CVDT	1.018	1.268	1.509	1.638	1.773	1.894	1.779	1.831
INTER	1.018	1.268	1.509	1.638	1.773	1.894	1.779	1.831

Tabela 2 - Carga obrigatória de 60% com dependência intra-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%
AVDT	1.431	1.566	1.466	1.569	1.440	1.404	1.428
CVDT	1.533	1.644	1.590	1.734	1.544	1.529	1.541
INTER	1.533	1.644	1.590	1.734	1.544	1.529	1.541

Tabela 3 - Carga obrigatória de 90% com dependência intra-tarefa.

As tabelas 4, 5 e 6 mostram os resultados da simulação quando apenas dependência inter-tarefa está presente. Novamente CVDT e INTER apresentam um desempenho superior ao AVDT. Embora as equações associadas com INTER

considerem explicitamente a dependência inter-tarefa, esta heurística resulta em valores similares ao CVDT para todos os cenários de carga obrigatória estudados.

As tabelas 7, 8 e 9 mostram os valores obtidos nas simulações quando existe simultaneamente dependência intra-tarefa e inter-tarefa. Os resultados são semelhantes aos obtidos com dependência inter-tarefa. Temos CVDT e INTER sempre melhores que AVDT. Temos também INTER com comportamento similar ao CVDT.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%	270%
AVDT	0.953	0.917	0.909	0.931	0.968	1.088	1.199	1.281	1.403
CVDT	1.001	1.000	1.013	1.066	1.136	1.293	1.457	1.617	1.691
INTER	1.000	1.000	1.012	1.065	1.135	1.293	1.457	1.615	1.689

Tabela 4 - Carga obrigatória de 30% com dependência inter-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%
AVDT	0.934	0.884	0.905	1.048	1.120	1.331	1.272	1.532
CVDT	0.999	1.001	1.051	1.235	1.438	1.696	1.590	1.818
INTER	1.000	1.002	1.056	1.236	1.447	1.699	1.605	1.832

Tabela 5 - Carga obrigatória de 60% com dependência inter-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%
AVDT	1.142	1.250	1.121	1.120	1.124	1.054	1.112
CVDT	1.332	1.467	1.355	1.379	1.345	1.308	1.316
INTER	1.333	1.457	1.364	1.384	1.343	1.314	1.319

Tabela 6 - Carga obrigatória de 90% com dependência inter-tarefa.

Também foram realizadas experiências contemplando sistemas onde a utilidade está concentrada em algumas poucas tarefas. As condições destas simulações foram as mesmas descritas na seção 4.3, exceto que agora 3 tarefas possuem valor nominal 20, 3 tarefas possuem valor nominal 10 e todas as demais possuem valor nominal 1. Os valores obtidos foram semelhantes aos apresentados nas tabelas de 1 a 9.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%	270%
AVDT	0.972	0.949	0.938	0.957	0.995	1.081	1.155	1.225	1.271
CVDT	1.000	1.000	1.007	1.050	1.099	1.210	1.309	1.425	1.460
INTER	1.000	1.000	1.007	1.050	1.098	1.211	1.308	1.426	1.460

Tabela 7 - Carga obrigatória de 30% com dependência intra-tarefa e inter-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%	240%
AVDT	0.956	0.929	0.942	1.043	1.129	1.256	1.227	1.391
CVDT	1.000	1.001	1.039	1.169	1.324	1.469	1.427	1.545
INTER	1.000	1.001	1.042	1.173	1.329	1.471	1.435	1.550

Tabela 8 - Carga obrigatória de 60% com dependência intra-tarefa e inter-tarefa.

Opcional:	30%	60%	90%	120%	150%	180%	210%
AVDT	1.090	1.184	1.104	1.128	1.117	1.071	1.083
CVDT	1.209	1.308	1.236	1.265	1.240	1.213	1.227
INTER	1.211	1.300	1.247	1.265	1.241	1.215	1.230

Tabela 9 - Carga obrigatória de 90% com dependência intra-tarefa e inter-tarefa.

Não é possível uma comparação direta dos resultados apresentados acima com os resultados apresentados em [CHU 90] e [FEN 94]. Em [CHU 90] existia a necessidade de garantir que as partes opcionais de todas as tarefas seriam eventualmente executadas. Em [FEN 94], embora uma parte opcional pudesse não ser jamais executada, haveria então um aumento no tempo de execução de partes obrigatórias, as quais então teriam que ser necessariamente executadas. Neste trabalho, uma parte opcional com valor nominal baixo provavelmente jamais será executada em um sistema com sobrecarga. Este comportamento é coerente com o objetivo de reservar as sobras de processador para as partes opcionais que mais utilidade adicionem ao sistema.

É possível fazer uma comparação direta entre as políticas propostas neste artigo e o AVDT. Ambas as políticas CVDT e INTER apresentam resultados em geral melhores que AVDT. Os melhores resultados do CVDT e do INTER em relação ao AVDT são devidos ao novo limite usado. A política AVDT emprega $\Lambda(t)$ como limite mínimo. Em cargas baixas, o AVDT rejeita partes opcionais com baixa densidade de valor mesmo que o processador esteja livre. O limite mínimo ζ, usado por CVDT e por INTER, considera o grau de ocupação do processador.

A política INTER considera explicitamente as dependências inter-tarefa. Entretanto, ela não apresentou resultados melhores que CVDT nas combinações de carga estudadas neste trabalho. Em [OLI 96] foram feitas simulações semelhantes às apresentadas aqui, mas com $\gamma_{i,j}=1$ para todas as tarefas. Aquele trabalho encontrou um desempenho melhor de INTER com relação a CVDT para cargas obrigatórias altas. Como a política INTER considera os valores de $\beta_{i,j}$ mas ignora os valores $\gamma_{i,j}$, seu comportamento é melhor quando todas as tarefas apresentam o mesmo valor de $\gamma_{i,j}$ e ele deixa de ser significativo. Como descrito na seção 4.3, os valores apresentados nas tabelas 1 a 9 consideraram $\gamma_{i,j}$ um valor aleatório entre 0 e 1.

6. Conclusões

Neste trabalho foram apresentadas duas novas heurísticas, próprias para serem empregadas como política de admissão em sistemas de tarefas imprecisas. Seu comportamento foi analisado através de simulação e comparado com outros dois algoritmos presentes na literatura. Foram considerados casos onde existem relações de dependência entre diferentes liberações de uma mesma tarefa e/ou dependência entre tarefas diferentes. As heurísticas propostas apresentaram um desempenho sempre melhor que outras existentes na literatura.

É importante notar a natureza dinâmica de diversos valores considerados pelas heurísticas, tais como, $\Lambda(t)$, $\pi(t)$, etc. Estes valores devem ser considerados apenas em um intervalo de tempo recente e não pela vida total do sistema. Desta forma seria

melhor tratada uma situação de mudança no modo de operação da aplicação ("mode change"), em termos dos valores de suas tarefas. Uma questão em aberto é determinar o intervalo de tempo adequado para a determinação de tais valores.

A abordagem empregada neste trabalho considera que o comportamento opcional do sistema não é conhecido de forma determinista mas sim probabilista. Neste sentido, ela é similar aos algoritmos de substituição de página empregados na implementação de memória virtual. As decisões são tomadas segundo a suposição de que o futuro próximo será semelhante ao passado recente. A mesma suposição é adotada aqui com respeito aos valores que descrevem a história da aplicação.

Várias questões permanecem ainda em aberto na literatura. Entre elas, como incorporar o conceito de computação imprecisa e de valores nominais de tarefas às técnicas de projeto de software. Para que as soluções de escalonamento discutidas neste artigo venham a ser usadas em aplicações reais é essencial sua integração com metodologias de desenvolvimento de software.

7. Referências

[AUD 93] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, A. J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. Software Engineering Journal, Vol. 8, No. 5, pp. 284-292, 1993.

[AUD 94a] N. C. Audsley, A. Burns, R. I. Davis, A. J. Wellings. Integrating Best Effort and Fixed Priority Scheduling. Proceedings of the IEEE Real-Time Systems Symposium, pp. 12-21, San Juan, Puerto Rico, december 1994.

[AUD 94b] N. C. Audsley, R. I. Davis, A. Burns. Mechanisms for Enhancing the Flexibility and Utility of Hard Real-Time Systems. Proceedings of the IEEE Real-Time Systems Symposium, pp. 12-21, San Juan, Puerto Rico, december 1994.

[BUR 91] A. Burns, A. J. Wellings. Criticality and Utility in the Next Generation. The Journal of Real-Time Systems, Vol. 3, correspondence, pp. 351-354, 1991.

[CHU 90] J.-Y. Chung, J. W.-S. Liu, K.-J. Lin. Scheduling Periodic Jobs That Allow Inprecise Results. IEEE Trans. on Comp., Vol.39, No.9, pp.1156-1174, sep 1990.

[DAV 95] R. Davis, S. Punnekkat, N. Audsley, A. Burns. Flexible Scheduling for Adaptable Real-Time Systems. Proc. IEEE Real-Time Technology and Applications Symposium, pp. 230-239, may 1995.

[FEN 94] W. Feng, J. W.-S. Liu. Algorithms for Scheduling Tasks with Input Error and End-to-End Deadlines. DCS Tech. Report #1888, Univ. of Illinois at U-C, 1994.

[KRO 94] E. Krotkov, R. Hoffman. Terrain Mapping for a Walking Planetary Rover. IEEE Trans. on Robotics and Automation, Vol.10, No.6, pp.728-739, dec 1994.

[LEU 82] J. Y. T. Leung, J. Whitehead. On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks. Performance Evaluation, 2(4), pp.237-250, december 1982.

[LIU 94] J. W.-S. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, J.-Y. Chung. Imprecise Computations. Proceedings of the IEEE, Vol. 82, No. 1, pp. 68-82, january 1994.

[OLI 96] R. S. Oliveira, J. S. Fraga. Scheduling Imprecise Computation Tasks with Intra-Task / Inter-Task Dependence. Proceedings of the 21st IFAC/IFIP Workshop on Real Time Programming, Gramado-RS-Brasil, pp. 51-56, november 1996.

[TIA 95] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun. L.-C. Wu, J.W.-S. Liu, Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times. Proceedings of the IEEE Real-Time Technology and Applications Symposium, pp. 164-173, may 1995.