WTR'2006 - 8o Workshop de Tempo Real
Curitiba - PR, 02 de junho de 2006

Determinação Dinâmica do Itinerário de Agentes Móveis Imprecisos com Deadline Firme

Luciana Rech, Carlos Montez, Rômulo de Oliveira

Departamento de Automação e Sistemas – Universidade Federal de Santa Catarina
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

{lrech, montez, romulo}@das.ufsc.br

Abstract. *In the mobile agent area, as part of an agent mission, it may exist the necessity of meeting a firm end-to-end deadline, although with some itinerary flexibility due to some optional resources. In this paper it is adopted an execution model where agents do not previously know their itineraries. The objective is to propose and evaluate heuristics for agent routing.*

Resumo. *Na área de agentes móveis, para um agente realizar sua missão pode haver a necessidade do cumprimento de um deadline fim a fim firme, contudo com flexibilidade no itinerário, decorrente da existência de recursos opcionais. No modelo de execução adotado neste artigo, considera-se que os agentes desconhecem antes da partida as possibilidades de seus itinerários. O objetivo é propor e avaliar heurísticas para roteamento de agentes móveis imprecisos com restrições temporais.*

1. Introdução

Um agente móvel é um elemento de software autocontido, responsável pela execução de uma tarefa, e que não está limitado ao sistema onde começa a sua execução, sendo capaz de migrar através de uma rede [Baek 2001]. Agentes móveis reduzem a carga na rede, executam assíncrona e autonomamente, e podem se adaptar dinamicamente, estabelecendo assim um paradigma para a programação em ambientes distribuídos.

Um cenário que ilustra o uso de agentes móveis com restrição temporal é um sistema de geração e distribuição de energia elétrica onde os dados encontram-se fisicamente espalhados. No contexto da indústria de energia elétrica, uma larga escala geográfica (podendo alcançar milhares de nodos [Tolbert 2001]) e respostas em tempo real são dois requisitos na implementação da supervisão do sistema. Considerando uma usina de geração de energia elétrica e as subestações que participam da transmissão, o agente móvel pode substituir o engenheiro com seu *laptop* visitando seções do sistema. Para alcançar o diagnóstico de uma falha, visando tomadas de decisão, o agente tem como função coletar dados, formar uma imagem do problema e decidir a próxima visita para coleta de dados.

Este artigo trata da utilização da mobilidade de código aliada ao cumprimento de requisitos temporais de tarefas em um sistema distribuído. É adotado um modelo de execução para aplicações baseadas em agentes móveis imprecisos com restrições de tempo real. O objetivo é desenvolver algoritmos para determinação dinâmica de seu itinerário. Considera-se que os agentes desconhecem antes da partida as possibilidades de seus itinerários. Um agente móvel impreciso é aquele capaz de reduzir a qualidade do resultado para conseguir atender o *deadline* da missão.

Este texto está dividido em 6 seções. Na seção 2 são apresentados alguns trabalhos encontrados na literatura relacionados ao tema. Na seção 3 é descrito o modelo de execução adotado. Nas seções 4 e 5 são introduzidas e avaliadas as heurísticas utilizadas para definição do itinerário. Na seção 6 são colocadas as conclusões.

2. Trabalhos relacionados

Em [Oida 2000] é descrito um sistema de roteamento baseado em agentes, que aloca recursos de rede para tarefas que solicitam comunicação ponto a ponto em tempo real. Em [Shin 2001], um protocolo de negociação baseado em agentes móveis, inteligentes e autônomos é aplicado em um sistema de escalonamento tempo real de uma linha de produção. Em [Ramamritham 2002] é proposto um sistema de gerenciamento de dados tempo real para tráfego de aplicações de navegação na rede. Em [Qu 2005] um algoritmo de roteamento baseado em agentes móveis é proposto. Em [Xu 2005] é descrito um método para tolerância a faltas utilizando agentes móveis com restrições temporais. Em [Baek 2001] é apresentado um método de planejamento de agentes móveis com restrições temporais visando minimizar o número de agentes e selecionar o melhor itinerário.

Entre os trabalhos citados, apenas em [Baek 2001] existe a preocupação de escolher um itinerário em que restrições temporais são respeitadas. Entretanto, no modelo de execução proposto neste artigo existe uma preocupação na escolha do melhor itinerário considerando o desempenho da execução da tarefa e a dificuldade em se cumprir o *deadline*. Neste contexto, a definição de "melhor" itinerário pode variar, conforme a métrica utilizada para avaliar itinerários. Técnicas de escalonamento adaptativo serão utilizadas com o objetivo de estabelecer um compromisso entre as visitas aos nodos e as restrições temporais da missão.

O problema considerado neste trabalho também é semelhante ao apresentado em [Balas 1989]. No entanto, algoritmos naquele trabalho são computacionalmente pesados e voltados para determinação estática do itinerário, quando não existem requisitos de tempo real envolvidos.

3. Modelo de execução

O modelo de execução proposto descreve o comportamento de uma aplicação baseada em agentes móveis, com restrições temporais, em um sistema distribuído formado por um conjunto de nodos conectados através de um serviço subjacente de comunicação. Cada agente móvel possui uma missão, associada com a visita a um determinado conjunto de nodos do sistema, os quais hospedam os recursos necessários para o agente. É suposto que este agente não se comunica com outros agentes. A missão do agente é formada por um conjunto de tarefas a serem executadas respeitando um *deadline*. Uma tarefa representa o uso de um recurso por um agente em um dado nodo. Um recurso é uma abstração e pode corresponder a um dispositivo, arquivo ou estrutura de dados.

O itinerário é o caminho definido por uma seqüência ordenada de nodos, e cada agente deste modelo possui flexibilidade na definição de seu itinerário. Esta flexibilidade está relacionada a alguns recursos que podem: i) ser utilizados em qualquer ordem, ii) aparecer replicados, iii) ser descartados ou iv) ser parcialmente utilizados (com respeito ao seu tempo de utilização).

3.1. Formalização do problema

Um sistema computacional é caracterizado por um conjunto ***N*** de nodos e um conjunto ***R*** de tipos de recursos. O conjunto ***N*** de nodos do sistema possui cardinalidade ***m***, ou seja, é composto por ***m*** nodos N_i , $N_i \in N$, onde $1 \leq i \leq m$. Em cada nodo N_i do sistema existe um conjunto R_i de um ou mais tipos de recursos pertencentes ao conjunto ***R***. Cada tipo de recurso pode ser encontrado em um ou mais nodos. Um itinerário ***I*** é um caminho descrito como uma seqüência ordenada de nodos N_i pertencentes ao conjunto ***N***. Um itinerário não precisa incluir todos os nodos do sistema, e pode incluir na seqüência que o define várias vezes o mesmo nodo.

Cada agente móvel ***Z*** possui uma missão ***M*** que define uma função benefício ***B*** para cada tipo de recurso. Esta função determina o quanto de benefício B_i cada tipo de recurso r_i contribui para a missão individual do agente.

O agente tem como objetivo agregar benefícios através do uso de vários recursos espalhados por vários nodos interconectados por uma rede de comunicação. A função objetivo do agente é maximizar o benefício total coletado ao longo de seu itinerário no sistema. O agente parte de um nodo N_0 externo ao sistema (nodo origem) e deve voltar a ele no final da missão. O nodo N_0 não possui instância dos recursos que formam o conjunto ***R***. Toda missão ***M*** possui um *deadline firme* ***D*** associado.

O diagrama de recursos de uma missão ***M*** (ex: Figura 1a) descreve as relações de precedência existentes para a utilização de recursos. Qualquer recurso utilizado em desacordo com este diagrama traz benefício nulo para a missão. O diagrama de recursos da missão descreve a flexibilidade e opções do agente ao realizar a missão. O esteriótipo <<*variable*>> indica que o tempo de uso do recurso ***r1*** é variável, podendo este ser liberado antes de seu benefício máximo.

O diagrama de nodos de uma missão ***M*** corresponde a um diagrama de atividades UML construído a partir do diagrama de recursos da missão ***M***, onde cada recurso é substituído pelo nodo ou nodos onde o mesmo aparece no sistema, apresentando assim o conjunto de itinerários possíveis para o agente. Por exemplo, a Figura 1b pode representar a possibilidade de itinerários diferentes para o agente cumprir sua missão, cujo diagrama de recursos está apresentado na Figura 1a.

Para todo $r_i \in R$, o benefício $B_{i,t}$ obtido por utilizar o recurso r_i durante o intervalo de tempo ***t*** é dado por:

1) r_i do tipo *variable* $B_{i,t} = B_i \; x \; min(1, t/Ci)$

onde B_i é o benefício máximo obtido de r_i e C_i é o tempo máximo de computação associado com r_i (C_i é o tempo que o agente deve ficar com o recurso para ganhar B_i e ***t*** é o tempo que ele realmente ficou).

2) r_i do tipo normal $B_{i,t} = B_i$ **se** $t \geq C_i$

$B_{i,t} = 0$ **se** $t < C_i$

Nas equações acima são consideradas relações de precedência e de opcionalidade:

- se ***ri*** precede ***rj***, e ***ri*** não foi utilizado, então $B_j = 0$;
- se ***ri'*** e ***ri"*** são réplicas de ***ri***, e ***ri'*** já foi usado, então benefício máximo de ***ri"***=0.

O benefício efetivo da missão é obtido através do somatório dos benefícios realizados a partir de cada recurso no itinerário: $B = \sum B_i(y)$ para todo $ri \in R$

sendo $B_i(y)$ o benefício realizado pela utilização do recurso ***ri*** que o agente visitou para o cumprimento de sua missão ao seguir o itinerário ***y***, ***B*** é o benefício gerado pela utilização de todos os recursos da missão.

O diagrama de recursos vai sendo conhecido pelo agente móvel ao longo do caminho, em função dos dados que serão obtidos nos nodos. Dessa forma, o agente móvel é definido como míope, ou seja, a cada momento ele conhece apenas os possíveis próximos nodos. Por conseguinte, o problema descrito neste artigo não pode ser resolvido utilizando otimização, pois na origem o diagrama de recursos não é conhecido; e durante o percurso os nodos visitados possuem capacidade de processamento limitada e podem não comportar os cálculos de otimização necessários. Exemplos desse tipo de ambientes podem ser encontrados em aplicações de supervisão de fábricas (agentes móveis na linha de produção [Shin e Jung 2001]).

Os agentes móveis carregam os resultados obtidos nos nodos visitados, mas com relação ao seu tamanho é assumido que o crescimento é pequeno, por este motivo o tamanho do agente é desconhecido.

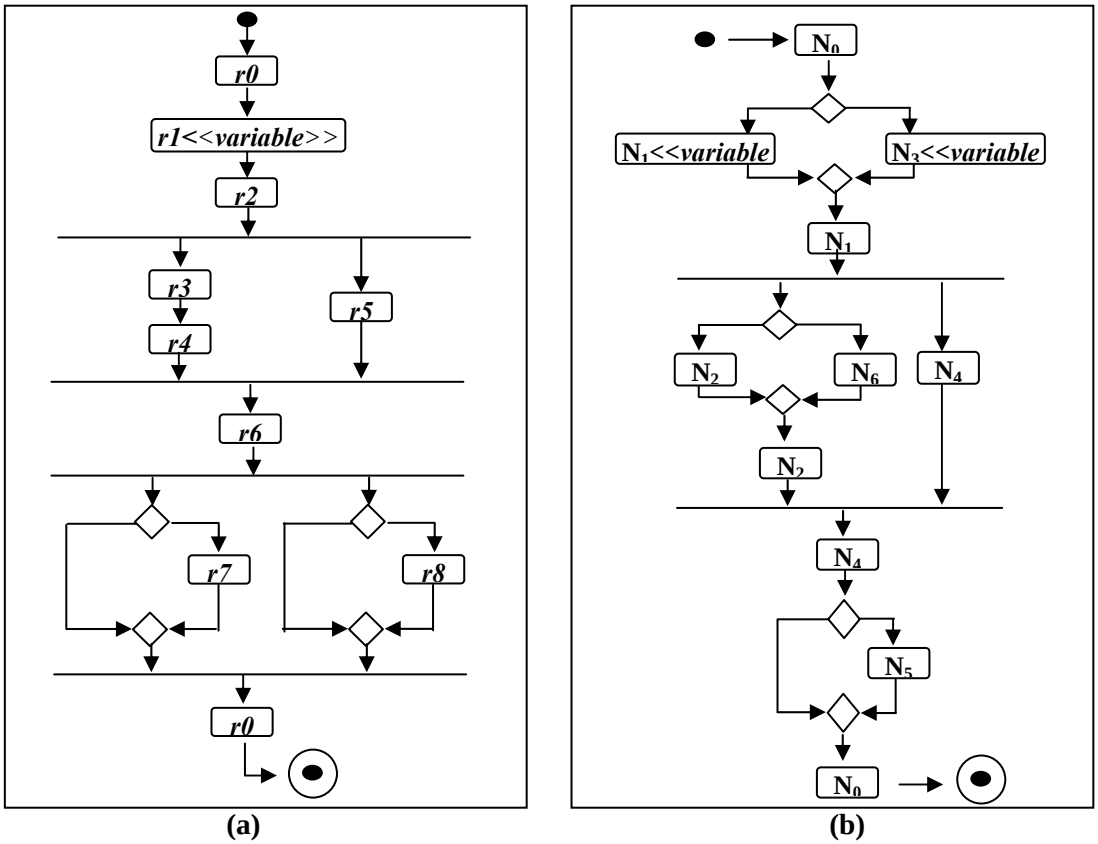

Figura 1. (a) Diagrama de recursos da missão *M*, (b) Diagrama de nodos da missão *M*.

4. Algoritmos para definição do itinerário

Em [Rech 2005] foi definido um modelo de agente móvel impreciso de tempo real e sugeridas algumas heurísticas simples para guiar os agentes móveis na definição de seu itinerário. Neste artigo, diferentemente de [Rech 2005], vamos considerar que as possibilidades de itinerário são desconhecidas antes da partida do agente móvel, e compararemos o desempenho das heurísticas segundo diferentes métricas de qualidade.

Estas heurísticas tomam decisões baseadas apenas na observação dos possíveis próximos nodos a visitar, desconsiderando desdobramentos futuros das decisões de curto prazo. Sempre que o agente precisar decidir entre dois caminhos equivalentes, segundo a heurística usada, é tomada uma decisão aleatória. As heurísticas empregadas são simples (com mínimo esforço computacional) e míopes (trabalham sobre diagrama de recursos parcialmente conhecido).

Algoritmo Preguiçoso (*Lazy Algorithm*) – Procura executar os recursos o mais rápido possível, ignorando seus benefícios. Recursos opcionais não são executados, e aqueles com estereótipo <<*variable*>> são executados com o menor tempo possível. Sempre que houver uma alternativa (quando não há relação de precedência entre dois recursos), será escolhido o recurso que apresentar menor tempo de computação. No caso de existência de réplicas de um recurso, é verificado se existe uma instância deste recurso no nodo onde se encontra o agente; caso não haja, será escolhido o nodo mais próximo.

Algoritmo Guloso (*Greedy Algorithm*) – Sempre que existir uma alternativa será escolhido o recurso que apresentar maior benefício para a missão. Este algoritmo sempre executa nodos opcionais. Em casos onde o recurso seja do tipo <<*variable*>>, será usado todo o tempo solicitado para obter o benefício máximo. Da mesma forma que no algoritmo anterior, no caso da existência de réplica de um recurso, será dada preferência ao recurso situado no nodo mais próximo.

Algoritmo Aleatório (*Random Algorithm*) – Este algoritmo escolhe aleatoriamente o próximo nodo a ser visitado pelo agente. Este comportamento só é possível se existirem réplicas de um determinado recurso ou não existir relação de precedência entre o próximo recurso e algum dos demais recursos a serem executados. Em casos onde o recurso seja do tipo <<*variable*>>, será escolhido de forma aleatória um valor entre zero e o tempo de computação necessário para alcançar o benefício máximo para este recurso.

Algoritmo Ponderado (*Higher Density Algorithm*) – Este algoritmo – baseado na política AVDT (*Average Density Threshold*) [DAVIS 1995] – elege como próximo recurso a ser executado aquele que apresentar a melhor relação benefício/tempo. Esta característica é possível apenas quando não existir relação de precedência entre dois recursos. A densidade do benefício do recurso é obtida por: $db_i = B_i / (C_i + L_{j,i} + Q_i)$ onde: db_i = densidade do benefício do recurso i, B_i = benefício obtido pela execução do recurso i, C_i tempo de computação utilizado pelo recurso i, $L_{j,i}$ é o tempo gasto com o deslocamento até o nodo Ni, e Q_i é o tempo na fila do processador local esperando para executar. Valores estimados podem ser usados quando os exatos não forem conhecidos.

O agente mantém o valor ***db*** médio até o momento. Quando um recurso é opcional, ele somente será executado se sua densidade for superior a densidade média da missão até aquele momento.

Versões com Relógio – Algoritmos que utilizam relógio neste artigo são probabilistas. No início de sua execução seguem seu comportamento original; à medida que o deadline da missão se aproxima, estes algoritmos ganham características semelhantes as do Algoritmo Preguiçoso, ou seja, buscam executar o mais rápido possível desconsiderando os benefícios com a execução de cada recurso. A cada momento do percurso a probabilidade do algoritmo comportar-se como o Algoritmo Preguiçoso é diretamente proporcional ao percentual de quanto do deadline já foi consumido. Recursos opcionais somente serão executados se o deadline permitir. Para estas versões

das heurísticas é necessária a sincronização relógio.

Os algoritmos que apresentam esta variação são: Algoritmo Guloso com Relógio e Algoritmo Ponderado com Relógio.

5. Comparativo entre as heurísticas

Para definição dos valores empregados, foi utilizado o modelo de execução descrito e o diagrama de nodos (Figura 1b). Baseado nestes dados foi criado um grafo composto por 51 vértices e 60 arestas, onde cada vértice representa um estado no comportamento do agente (por exemplo, chegada no nodo, execução do recurso) e cada aresta representa o benefício e o tempo consumido para a execução de uma determinada ação.

Para o cálculo do tempo de computação total da missão foram considerados: (i) os tempos de computação de cada recurso (t_1=10, t_2= 2, t_3= 7, t_4=2, t_5=8, t_6=5, t_7=7, t_8=3); (ii) o tempo na fila do processador, distribuição uniforme entre 1 e 3 (carga leve), ou entre 1 e 10 (carga pesada); e (iii) o tempo de deslocamento ***L*** entre os nodos (exponencial com média 2). A qualidade do algoritmo (benefício global do algoritmo ***G***) é calculada seguindo a seguinte métrica: $G=\Sigma B\_DA/NT$; onde ***B_DA*** é o benefício obtido pela execução dos recursos com *deadline* atendidos (B_1=10, B_2=3, B_3=3, B_4=7, B_5=5, B_6=3, B_7=6, B_8=8); e ***NT*** é o número de tentativas.

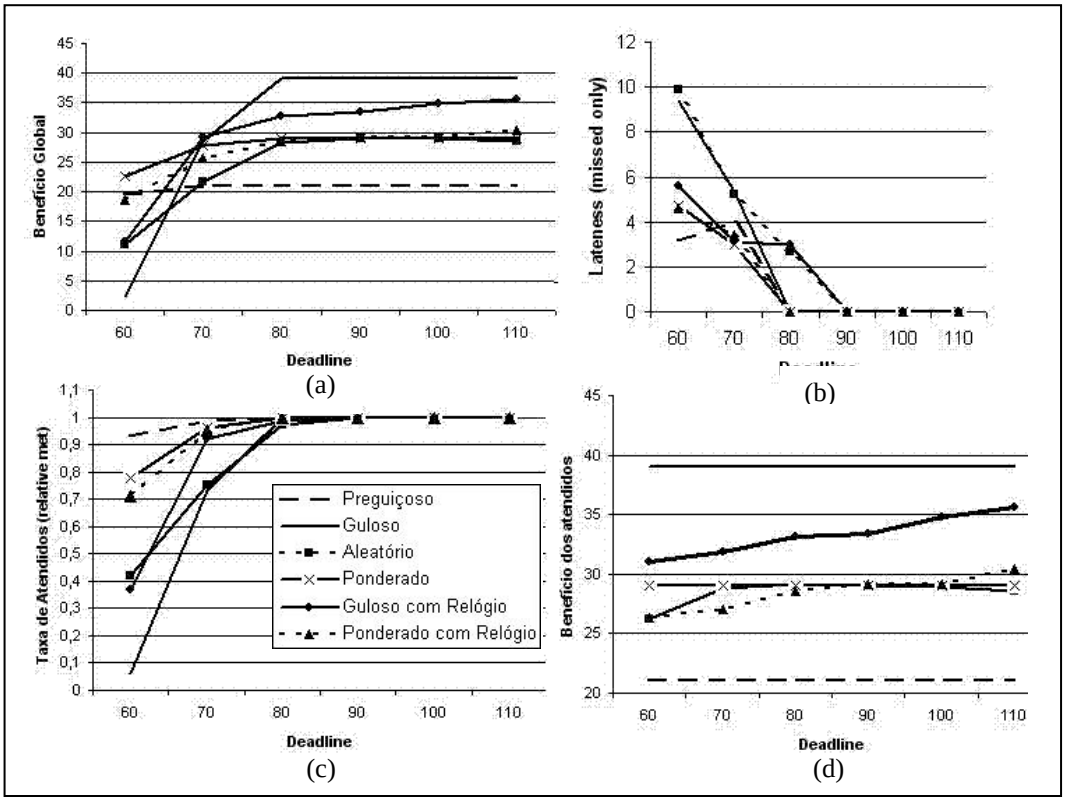

Figura 2. Comportamento das heurísticas utilizando uma carga leve no sistema.

5.1. Resultado da Simulação

As figuras 2 e 3 apresentam o comportamento dos algoritmos para situações em que a carga nos nodos encontra-se, respectivamente, leve ou pesada. Em (a) está sendo analisado o benefício global ***G*** obtido por cada algoritmo, em (b) o atraso médio ***L*** dos agentes perdidos, em (c) o percentual de agentes que foram atendidos respeitando o

deadline e em (d) o benefício médio ***B*** considerando apenas os deadlines atendidos.

Analisando o gráfico 2(a) percebe-se que apartir de um determinado momento (ponto em que todos os agentes da missão são atendidos) o benefício global obtido pelos algoritmos permanece estável, independente do tempo restante entre o deadline e o tempo gasto com a missão. Com deadlines apertados (por exemplo, 60) o Algoritmo Preguiçoso apresentou o melhor desempenho, com deadlines folgados (por exemplo, 100) o Algoritmo Guloso e Guloso com Relógio obtiveram os maiores índices de benefício.

No gráfico (b) a taxa de agentes perdidos tende a zero, o algoritmo Preguiçoso apresenta o menor índice de agentes perdidos com deadlines apertados.

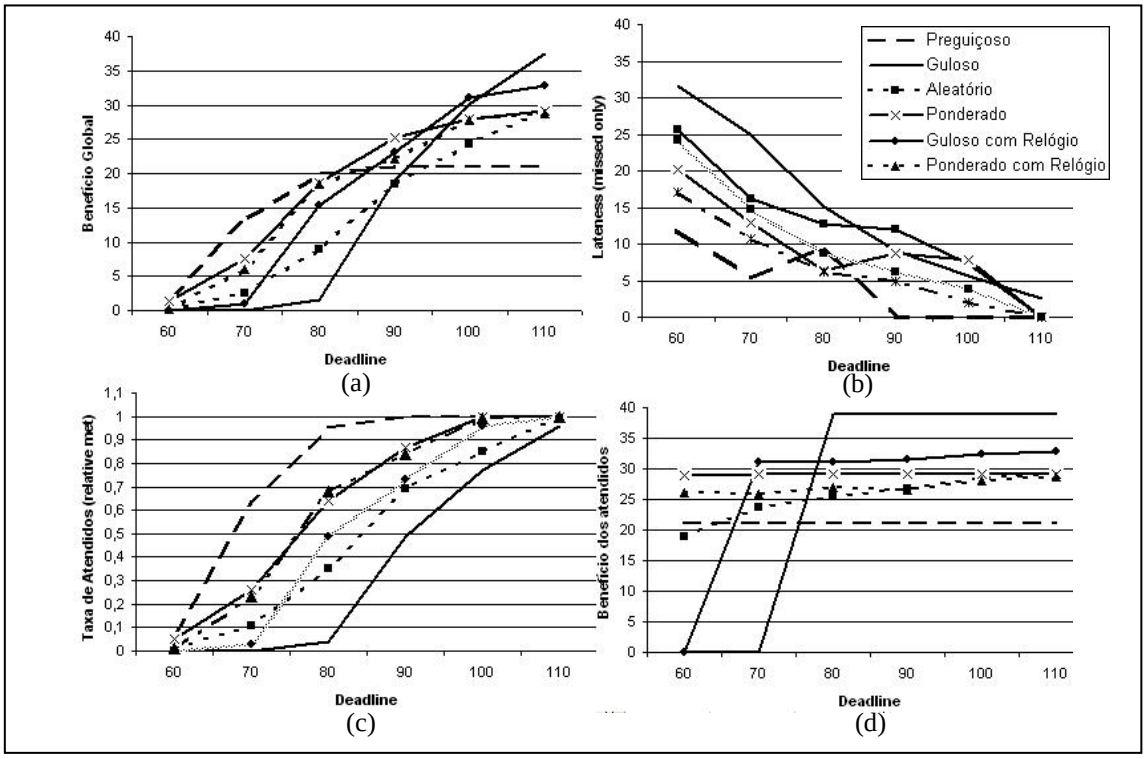

Figura 3. Comportamento das heurísticas utilizando uma carga pesada no sistema.

Observando gráfico (d) percebemos que o benefício individual dos deadlines atendidos é inalterado para algoritmos sem relógio (com exceção do algoritmo Aleatório). Algoritmos que utilizam relógio apresentaram um pequeno acréscimo no benefício obtido. O algoritmo Guloso apresentou o maior índice de benefício.

O algoritmo Preguiçoso obteve 100% de agentes atendidos com deadlines apertados (por exemplo, 80), enquanto que os algoritmos Guloso e Ponderado (versões com e sem relógio) necessitam de deadlines maiores para obter 100% de atendidos.

A figura 3 apresenta o comportamento dos algoritmos em situações onde a carga nos nodos é pesada. As características comportamentais e os resultados encontrados são semelhantes a situações onde a carga nos nodos é leve. É importante lembrar que, neste caso, o tempo total para se cumprir a missão é maior em função do tempo em que o agente espera na fila para executar os recursos nos nodos.

A definição de melhor algoritmo dependerá das condições de carga nos nodos e também das características e objetivos da missão. Os algoritmos Guloso e Preguiçoso

representam os dois extremos, sendo o Guloso melhor para deadlines folgados e o Preguiçoso melhor para deadlines apertados. Entretanto, no modelo de execução considerado o agente desconhece o diagrama de recursos além dos nodos imediatamente seguintes. Neste caso, o algoritmo Ponderado representa uma solução de compromisso. A adição de relógio ao Ponderado não resultou em melhoria significativa.

6. Conclusões

Este artigo apresentou um modelo de execução para agentes móveis imprecisos com restrições temporais, porém com flexibilidade na definição de seu itinerário. Também foi proposto e avaliado o comportamento de várias heurísticas que podem ser usadas pelo agente móvel na escolha do itinerário, as quais podem ser usadas como ponto de partida para algoritmos mais complexos. Este modelo de execução é inovador ao associar restrição de tempo e o conceito de recursos opcionais à tecnologia de agentes móveis. Uma comparação entre as várias heurísticas mostrou que existe a necessidade de estabelecer um compromisso entre valor da missão e tempo de execução.

Atualmente, estamos avaliando novas heurísticas, mais complexas, que combinam características de várias das heurísticas apresentadas. Além disso, estamos trabalhando com heurísticas adaptativas, que mudam seu comportamento considerando um histórico de benefícios conseguidos em execuções passadas do agente móvel.

7. Referências

Baek, J.W., et all. (2001) "*Timed Mobile Agent Planning for Distributed Information Retrieval*". Proc. of AGENTS´01. Montreal. Quebec. Canadá. Junho.

Balas, E. (1989) "*The Prize Collecting Traveling Salesman Problem*" John Wiley & Sons, Inc. Networks, Vol.19, p.621-636.

Davis, A. et all. (1995) "*Flexible Scheduling for Adaptable Real-Time Systems*", *Proc. IEEE Real-Time Tech. and App. Symp,* pp. 230-239. Maio.

Oida, K., e Sekido, M. (2000) *"ARS: An efficient agent-based routing system for QoS guarantees"* Elsevier Science, *Computer Communications* 23, p. 1437-1447.

Qu, W., Shen, H., and Jin, Y. (2005) "*Theoretical Analysis on a Traffic-Based Routing Algorithm of Mobile Agents"* IAT 2005, Compiègne, France, Setembro.

Ramamritham, K., Kwan, A. e Lam, K.Y. (2002) "*RTMonitor: Real-time Data Monitoring Using Mobile Agent Technologies"*, Proc. of 28th VLDB Conference, Hong Kong, China.

Rech, L., de Oliveira, R., and Montez, C. (2005) "*Dynamic Determination of the Itinerary of Mobile Agents with Timing Constraint"*. IAT 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, p. 45-50, Compiègne, France, Setembro.

Shin, M., Jung, M. e Ryu, K. (2001) "*A Novel Negotiation Protocol for Agent-based Control Architecture" 5th Int. Conference on Engineering & Automation.* Las Vegas. EUA.

Tolbert, L. M., et all. (2001) "Scalable Multi-Agent System for Real-Time Electric Power Mangement" IEEE Power Engineering Society Summer Meeting, pp. 1676-1679, Vancouver, Canadá, Julho.

Xu, J. e Pears, S. (2005) "*A Dynamic Shadow Approach to fault-tolerant Mobile Agents in an Autonomic Environment*" Real-time Systems, Springer Science + Business Media, Inc. Manufactured in the Netherlands, Dezembro.