A Heurística Lazy-Adaptativo para Determinar Itinerários de Agentes Móveis com Restrição Temporal

Luciana Rech[1], Alex Magalhães[1], Lau Cheuk Lung[1], Rômulo Silva de Oliveira[2]
[1]Departamento de Informática e Estatística – INE – CTC
[2]Departamento de Automação e Sistemas – DAS – CTC
Universidade Federal de Santa Catarina - Brasil

lrech@das.ufsc.br, {alex, lau.lung}@inf.ufsc.br, romulo@das.ufsc.br

Abstract. *Algorithms to determine the itinerary of agents are fundamental in the context of distributed applications based on mobile agents with time restriction. To establish the agent´s itinerary is necessary to consider the trade-offs between high-quality of results and meeting the deadline. In this paper we describe and evaluate adaptive heuristics able to choose the best behavior to be used for decision making in the itinerary definition. This decision-making is based on a log of benefits collected by the mobile agent in past executions. The paper's objective is to propose a new heuristic able to realize the change in the environment and adapt itself dynamically, changing it behavior in order to meet the deadline of the mission and achieve the greatest possible benefit for this mission.*

Resumo. *Algoritmos para determinar o itinerário de agentes são fundamentais no contexto de aplicações distribuídas baseadas em agentes móveis com restrição temporal. Para estabelecer o itinerário do agente é necessário considerar a relação entre qualidade dos resultados e cumprimento do deadline. Neste artigo serão descritas e avaliadas heurísticas adaptativas capazes de escolher o melhor comportamento a ser utilizado para a tomada de decisão na formação do itinerário. Esta tomada de decisão é baseada no uso de um histórico de execuções anteriores. O objetivo do artigo é propor uma nova heurística capaz de perceber a mudança no ambiente e adaptar-se dinamicamente, mudando seu comportamento com o intuito de cumprir o deadline da missão e alcançar o maior benefício possível para esta missão.*

1 Introdução

A mobilidade de código serve como uma alternativa às abordagens tradicionais (modelo cliente-servidor, computação concorrente e outros) focadas no modelo orientado à troca de mensagens ou compartilhamento de memória, e traz benefícios no contexto de vários domínios de aplicação. Entre os benefícios esperados da mobilidade de código pode-se citar [Baek 2002]: maior grau de flexibilidade, escalabilidade e customização; melhor utilização da infra-estrutura de comunicação e a provisão de serviços de maneira autônoma sem a necessidade de conexão permanente. Tais benefícios justificam o crescente interesse nesta área de pesquisa. Alguns domínios de aplicação para os quais a mobilidade de código tem sido aplicada na literatura são: recuperação de informação, serviços avançados de telecomunicações, controle de dispositivos remotos, gerência de fluxos de trabalho, comércio eletrônico, entre outros.

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 autonomamente através de uma rede. Agentes móveis reduzem a carga na rede, executam assincronamente e autonomamente, e adaptam-se dinamicamente, estabelecendo assim um novo paradigma para a programação em ambientes distribuídos. A seqüência de nodos visitados por um agente móvel, entre um

nodo origem e um nodo destino, é chamada itinerário. Uma questão relevante para a determinação do itinerário de um agente móvel é o seu conhecimento prévio ou não dos nodos que poderão ser visitados. Neste artigo é considerada ainda a restrição temporal na definição do itinerário.

No contexto de aplicações distribuídas, existem diversas abordagens que permitem às aplicações de tempo real se adaptarem dinamicamente às várias plataformas. Técnicas como *anytime algorithms* [Garvey 1994] e computação imprecisa [Liu 1994], por exemplo, podem ser empregadas com esse objetivo. Nessa técnica, cada tarefa de aplicação é capaz de gerar resultados com diferentes níveis de qualidade ou precisão. Com esse objetivo, as tarefas são divididas em partes obrigatórias e partes opcionais. A parte obrigatória é capaz de gerar um resultado com qualidade mínima, necessária para manter o sistema operando de maneira segura. A parte opcional refina este resultado, até que ele alcance a qualidade desejada.

Este trabalho trata da utilização da mobilidade de código aliada ao cumprimento de requisitos temporais de tarefas em um sistema distribuído. O conceito de agente móvel impreciso com restrição temporal foi originalmente apresentado em [Rech 2005] juntamente com algumas heurísticas simples para guiar o agente na definição do itinerário. Em [Rech 2006] foi descrita a idéia de se utilizar clones de agentes móveis imprecisos, onde sua avaliação de desempenho foi verificada em [Rech 2008a]. Esta avaliação de desempenho mostrou que embora heurísticas simples sejam apropriadas para intervalos de deadline específicos, a utilização de pares de clones constituiu uma abordagem mais robusta para o problema, mas com a constante preocupação com a possibilidade de overhead. Partindo desta avaliação de desempenho percebeu-se a necessidade em criar heurísticas que se adaptassem às necessidades do agente móvel. O primeiro passo em direção a este comportamento adaptativo foi a criação de heurísticas capazes de decidir o melhor comportamento a ser utilizado para a missão atual do agente móvel [Rech 2008]. Baseando-se num histórico de execuções anteriores, o agente móvel é capaz de selecionar o comportamento mais adequado para a missão atual. Em [Rech 2008] a adaptação acontece na partida, o agente assume o mesmo comportamento durante toda a missão.

A tomada de decisão utilizada pelas heurísticas é baseada apenas na observação dos possíveis próximos nodos que o agente poderá visitar. As heurísticas empregadas são simples (com mínimo esforço computacional) e míopes (trabalham sobre diagrama de recursos e diagramas de nodos parcialmente conhecidos). Diferentemente dos trabalhos anteriormente mencionados, neste artigo será apresentada uma nova heurística com capacidade de adaptar-se dinamicamente. Isto é, durante a trajetória do agente móvel poderão acontecer mudanças nas condições da rede ou dos nodos a serem visitados. Então, o agente móvel - preocupado não apenas em cumprir o deadline, mas também em alcançar o maior benefício possível para a missão - é capaz de mudar o comportamento inicialmente assumido. Será verificado também que a heurística proposta é eficiente para todas as faixas de deadline, principalmente, para os mais apertados.

Neste artigo, assumimos que as possibilidades de itinerário são ignoradas antes da partida do agente móvel. O desempenho das heurísticas é comparado seguindo diferentes métricas de qualidade. São descritos e analisados algoritmos para a determinação dinâmica do itinerário seguindo um diagrama de recursos específico. O

objetivo deste artigo é estudar o comportamento destas heurísticas adaptativas em diferentes cenários.

O texto está dividido em 6 seções. Na Seção 2 são apresentados alguns trabalhos relacionados. Na Seção 3 o modelo computacional utilizado é brevemente descrito. A Seção 4 contém as heurísticas adaptativas utilizadas na definição do itinerário. A Seção 5 apresenta uma comparação e uma avaliação entre as heurísticas de definição de itinerário, e finalmente, na Seção 6 são apresentas as conclusões.

2 Trabalhos Relacionados

Esta seção apresenta algumas abordagens que mostram agentes móveis sendo utilizados de forma a auxiliar tarefas distribuídas com restrição temporal.

Redes neurais e técnicas de sistemas especialistas são cada vez mais frequentes em aplicações no campo de diagnóstico de ferramentas de máquinas. Em [Ong e Sun 2003] os agentes móveis funcionam como programas de monitoramento, o controlador central pode despachá-los para os nodos remotos onde os sinais de máquina são fornecidos de máquinas reais. A integração destas duas técnicas de inteligência artificial é particularmente apropriada para diagnóstico de falhas em máquinas em tempo real.

Equipamentos de navegação de carros usualmente tratam o problema das rotas segundo os algorítimos clássicos de origem e destino. Em [Kanoh 2008] é proposta uma quebra de paradigma adicionando-se novos objetivos ao problema de buscar a menor rota: minimizar o tempo da viagem e obter a melhor dirigibilidade possível. A dirigibilidade é definida como o total de penalidades de uma rota. A rota com o menor número de penalidades é a rota com melhor dirigibilidade. As penalidades são calculadas em função das paradas e curvas que um motorista fará pela rota. Engarrafamentos também serão computados, assumindo-se que os informes de trânsito são recebidos em tempo real. Um algoritimo híbrido de inteligência artificial combinado a uma solução clássica de problemas de rotas é proposto pelo autor para resolver o problema. O principal problema da solução é a alta especialização da solução para o tipo de problema.

Em [Qu e Shen 2005] é apresentado um algoritmo de roteamento de agentes móveis que considera o custo dos enlaces. Para auxiliar na tomada de decisão do agente, existe uma distribuição de probabilidade para o agente móvel selecionar um dos nodos vizinhos e mover-se até ele. O custo do enlace entre dois nodos é definido com base nas informações de tráfego conhecidas e na distribuição de probabilidade encontrada. Com relação à forma como é definido o itinerário, diferentemente deste artigo, em [Qu e Shen 2005] vários agentes partem em busca do melhor itinerário, voltam ao nodo origem (servidor), é então selecionado o melhor itinerário, sendo que apenas um agente parte novamente para cumprir sua missão e este já conhece seu itinerário.

Em [Al-Kasassbeh e Adda 2008] é introduzido um modelo analítico para gerenciamento de rede cliente/servidor e o paradigma de agentes móveis, onde agentes móveis inteligentes são utilizados para auxiliar no gerenciamento da rede. O gerenciamento de falhas pode ajudar a aumentar a utilidade/disponibilidade da rede pelo fato de identificar falhas mais rapidamente e desta maneira iniciar um processo de recuperação antecipado. Diferentemente de [Al-Kasassbeh e Adda 2008], no presente artigo, agentes inteligentes são capazes de perceber uma possível folga no deadline da

missão, e então, o agente móvel poderá alterar sua estratégia de decisão - seu comportamento – na definição de seu itinerário.

Monitoramento de ambientes não é uma tarefa trivial. Combinando-se a necessidade de uma alta frequência de atualização de dados (tempo real) com o fato de que nem sempre os dados são visíveis à distância, a situação pode ser ainda mais desafiadora. Em [Ilarri 2008] é proposta uma técnica com agentes móveis escalonável e capaz de atender requisitos de tempo real. O problema apresentado é o monitoramento de dispositivos *bluetooth* em uma universidade e a estratégia dos agentes móveis é do tipo dividir-e-conquistar. A universidade é dividida em níveis e os agentes trabalham em hierarquias, respeitando os deadlines. Uma das críticas à solução apresentada no artigo é que apesar da flexibilidade e alta escalabilidade da solução, não é apresentada uma alternativa para problemas com deadlines muito apertados.

Em [Gutin e Punnen 2002] são analisadas diversas soluções para variações de problemas relacionados ao Caixeiro Viajante e a otimização combinatória. As buscas heurísticas locais estão entre as principais ferramentas de otimização e são principalmente usadas em casos em que a vizinhança é de cardinalidade polinomial. Há uma vasta literatura descrevendo os diferentes tipos de variações de problema do caixeiro viajante. Uma das técnicas usadas para explicar é a análise de dominação, as heurísticas com baixo número de dominação são consideradas sem utilidade prática. Apesar do artigo não tratar de ciclos completos no grafo – aqui representado pela rede em que o agente deve trafegar –, também foi escolhida a abordagem visando atender um maior número de soluções (alta dominância) e por isso as heurísticas apresentadas buscam ser eficientes para todas as faixas de deadlines.

Em [Koeing e Likhachey 2006] é apresentado um algoritmo para buscas em tempo real em um terreno desconhecido, o RTAA*. Esse algoritmo proposto utiliza os estados passados para manter o foco no objetivo final, sendo uma variação do algoritmo clássico de Inteligência Artificial A*. Para atender o requisito de tempo real, a heurística apresentada usa computação imprecisa. Diferente de [Koeing e Likhachey 2006], o presente artigo trabalha com um conceito de tempo real mais amplo, sendo o prazo aplicado à totalidade da missão e não apenas ao próximo trecho dela, sendo assim aplicável a uma gama maior de problemas.

Em [Zgaya 2008] é apresentada uma estratégia de migração de Agentes Móveis para a recuperação de informação para diversos clientes. Cada cliente solicita uma informação para o sistema e esta é interpretada como uma tarefa. Cada tarefa é então dividida em sub-tarefas. O principal objetivo da estratégia proposta é o ganho de escala e a minimização do tempo de computação. Isto é obtido otimizando-se o número de agentes e os nodos que cada agente precisa visitar, identificando as sub-tarefas iguais de cada solicitação.

3. Descrição do Modelo

Nesta seção é brevemente descrito o modelo computacional apresentado em [Rech 2005]. Este modelo descreve o comportamento de aplicações baseadas em agentes móveis imprecisos com restrição temporal, em um sistema distribuído formado por um conjunto de nodos conectados através de um serviço subjacente de comunicação.

Neste modelo computacional, cada agente móvel possui uma missão associada com a visita a um determinado grupo de nodos do sistema. Nestes nodos encontram-se

os recursos necessários para o agente completar sua missão. Será assumido que os agentes não se comunicam. Um recurso é uma abstração e pode corresponder a um processador, dispositivo, arquivo, estrutura de dados, etc. A missão é formada por um grupo de recursos que devem ser executados respeitando um deadline.

Um agente móvel impreciso é aquele capaz de reduzir a qualidade do resultado para conseguir atender o deadline da missão. Neste artigo é suposto que um diagrama de recursos indique as relações de precedência e as opções existentes entre os recursos a serem utilizados pelo agente em sua missão. Como vários recursos podem aparecer replicados no sistema, é construído um diagrama de nodos que, a partir do diagrama de recursos, mostra também estas opções.

Cada recurso representa um benefício adicional para a missão do agente, cujo objetivo é maximizar o somatório dos benefícios obtidos ao longo do itinerário, respeitando as precedências. Entretanto, o agente móvel é suposto míope [Rech 2006], ou seja, ele conhece apenas as opções imediatas do diagrama de recursos. A miopia do agente impede que o problema seja tratado através de técnicas clássicas de otimização.

Outro requisito que deve ser considerado pelo agente móvel é o deadline de cada missão, salientando a importância da escolha do melhor itinerário para cada situação apresentada. Esta seqüência de nodos visitados pelo agente móvel entre o nodo origem e o nodo destino (itinerário) é definida dinamicamente.

3.1 Formulação do Problema

Cada agente móvel possui uma missão $\boldsymbol{M}$ que define a função benefício $\boldsymbol{B}$ para cada tipo de recurso $\boldsymbol{R}$. Ou seja, esta função determina a quantidade de benefício $\boldsymbol{b_i}$ que cada tipo de recurso $\boldsymbol{ri}$ contribui para a missão do agente. Na prática, estabeleceu-se uma escala arbitrária para comparação dos diferentes tipos de recursos (dispositivo, arquivo, estrutura de dados, etc.) em relação à missão do agente, de forma que a função benefício $\boldsymbol{B}$ possa ser otimizada e sirva como parâmetro de comparação entre os diferentes itinerários possíveis.

O diagrama de recursos de uma missão corresponde a um diagrama de atividades UML (*Unified Modeling Language*) [Arlow e Neustadt 2005] que descreve as relações de precedência entre os recursos. Qualquer recurso em desacordo com este diagrama não traz nenhum benefício para a missão. O diagrama de nodos é baseado no diagrama de recursos da missão. Cada recurso é substituído pelo(s) nodo(s) onde eles aparecem no sistema.

O cumprimento de uma missão $\boldsymbol{M}$ está relacionado à escolha de um itinerário $\boldsymbol{I}$. Para definição do itinerário são considerados: o tempo de computação para benefício máximo de cada tarefa; a latência de comunicação entre os nodos; as dependências entre os recursos que aparecem na forma do diagrama de recursos da missão. Devem ser analisados os custos para que o agente chegue ao nodo (vindo do nodo anterior) e para adquirir aquele recurso ($\boldsymbol{C_i} + \boldsymbol{Q_i}$ onde: $\boldsymbol{C_i}$ é o tempo de execução no nodo e $\boldsymbol{Q_i}$ é o tempo na fila do processador local esperando para executar).

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

1) $\boldsymbol{r_i}$ do tipo *variable*

$$b_{i,t} = b_i * min(1, t/Ci) \qquad (1)$$

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

$$b_{i,t} = b_i \quad \text{se} \quad t \geq C_i \qquad (2)$$
$$b_{i,t} = 0 \quad \text{se} \quad t < C_i$$

O benefício efetivo B da missão é obtido através do somatório dos benefícios realizados a partir de cada recurso no itinerário:

$$B = \sum b_i(y) \quad \text{para todo} \quad r_i \in R. \qquad (3)$$

Sendo $b_i(y)$ o benefício realizado pela utilização do recurso r_i 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 objetivo do algoritmo de definição de itinerário é maximizar o valor de B, respeitando o deadline D da missão.

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 não é considerado no modelo computacional.

4. Descrição das Heurísticas Adaptativas

Este artigo apresenta um novo método para definir dinamicamente o itinerário de agentes móveis míopes com restrição temporal de acordo com o modelo computacional descrito na Seção 3. As heurísticas discutidas nesta seção consideram um histórico de benefícios conseguidos em execuções passadas do agente móvel. O agente consulta o histórico existente com o objetivo de escolher seu comportamento para iniciar a missão. O histórico armazena informações referentes a missões anteriores, como: o tempo de resposta, o benefício adquirido e a heurística utilizada**.**

O agente usará adaptação dinamicamente, uma vez escolhido o comportamento para a missão em questão, ele poderá assumir diferentes comportamentos até o final da missão. Para realizar a escolha do comportamento é utilizada probabilidade condicional baseada nas características do ambiente (sistema distribuído) e nos últimos eventos (histórico).

Em [Rech 2008] são descritas e avaliadas as heurísticas PG (Preguiçoso-Guloso) e MTR (Memória por Tempo de Resposta) ambas com adaptação na partida, onde uma vez escolhido o comportamento o agente permanecerá com ele até o final da missão. O método de definição de itinerário proposto no presente artigo pode ser considerado uma evolução das versões com adaptação na partida.

A definição de heurísticas adaptativas utiliza algumas das heurísticas simples que são minuciosamente descritas em [Rech 2005] e avaliadas em [Rech 2008a]. A seguir, será brevemente apresentada a tomada de decisão de cada heurística simples, informação necessária para a compreensão da descrição das heurísticas adaptativas:

- Preguiçoso (*Lazy*): este algoritmo tenta executar os recursos o mais rápido possível, ignorando o benefício de cada recurso. Os recursos opcionais não são executados e os recursos com estereótipo <<variável>> são executados com o menor tempo possível.

- Guloso (*Greedy*): sempre que existir uma rota alternativa, ele irá escolher a que representar em um maior benefício para a missão, sem nenhuma preocupação com o tempo de execução (deadline). Recursos opcionais sempre serão executados e recursos do tipo <<variável>> serão executados com o tempo de computação necessário para atingir o maior benefício possível.
- Ponderado (*Higher Density*): Este algoritmo escolhe como próximo recurso a ser executado aquele que apresentar a melhor relação custo/benefício. Recursos opcionais somente serão executados se sua densidade (relação entre o benefício adquirido e o custo) for superior à densidade média da missão até aquele momento.

4.1. Adaptação na Partida

Heurística Preguiçoso-Guloso (PG): Esta heurística é baseada na utilização de duas heurísticas simples: Preguiçoso e Guloso. A heurística constrói um histórico formado apenas pelos tempos de resposta das execuções anteriores (para a formação do histórico desta heurística, os deadlines das execuções anteriores e a informação se os deadlines foram cumpridos ou não são irrelevantes).

Na sua primeira execução, o agente móvel sempre escolhe o Algoritmo Guloso – esta missão é então salva no histórico. Quando surge uma nova missão, seu deadline é comparado com os tempos de resposta anteriores armazenados no histórico, e assim estima-se $P(R \leq D)$, ou seja, a probabilidade da missão atual ter o seu deadline atendido. Esta estimativa é comparada com um limiar e o resultado desta comparação decide o comportamento que o agente irá exibir. Quando a probabilidade de sucesso está abaixo do limiar, o agente assume o comportamento Preguiçoso. Mas se a probabilidade de sucesso estiver acima do limiar, o comportamento Guloso é adotado pelo agente. O valor de $P(R \leq D)$ é calculado simplesmente através da contagem do número de missões do histórico que teriam atendido o deadline atual e da divisão desta contagem pelo número total de missões do histórico. Neste cálculo é desconsiderado qual heurística foi utilizada no passado pelas missões registradas no histórico.

O valor escolhido como limiar para a troca de algoritmos entre Guloso e Preguiçoso é uma constante arbitrária. A qualidade dos resultados obtidos está diretamente relacionada à calibragem desta constante. Apesar dessa desvantagem, a complexidade computacional deste algoritmo é O(n), onde n é o tamanho do histórico. Além disso, esse algoritmo apresenta robustez com relação à escolha do algoritmo do agente, mesmo com poucas execuções e com histórico vazio. Caso uma escolha anterior tenha sido feita "erradamente" no algoritmo Preguiçoso, a média dos tempos de resposta armazenados no histórico diminui em função do comportamento Preguiçoso, e a probabilidade de escolha do Guloso aumenta na sua próxima execução. Fato similar ocorre quando há uma escolha "errada" no algoritmo Guloso.

4.2. Adaptação durante o Percurso (Dinâmica)

Heurística Lazy-Adaptativo: Este algoritmo é baseado na utilização da heurística simples Preguiçoso, porém com a capacidade de melhorar a escolha dos próximos nodos usando seu histórico. O objetivo principal desta heurística é atingir os maiores índices de benefícios permitidos para qualquer deadline, inclusive os deadlines muito apertados. A heurística constrói um histórico formado pelos deadlines alcançados para

comportamentos utilizados em cada missão, bem como os detalhes desse comportamento, a fim de poder repeti-los para cumprir deadlines semelhantes.

O histórico armazena apenas 50 comportamentos de missão e deadlines. Uma questão importante é o tamanho mínimo para o histórico poder ser considerado confiável. As heurísticas adaptativas foram avaliadas utilizando históricos de tamanhos variados entre 10 e 1000 com o objetivo de estabelecer um tamanho de histórico consistente, que seja confiável e evite armazenamento desnecessário, conseqüentemente economizando processamento. Partindo dos resultados obtidos pelas avaliações realizadas, percebeu-se que o histórico contendo de 50 à 60 execuções passadas, seria suficiente para garantir que as heurísticas adaptativas pudessem escolher o melhor comportamento para a missão atual.

Inicialmente, o agente sempre escolhe o comportamento do algoritmo Preguiçoso e à medida que percebe a possibilidade de mudar seu comportamento para obter mais benefícios, utiliza o histórico das missões para melhorar sua escolha em um nodo determinado em missão anterior. Ao alcançar esse nodo, ele não irá utilizar o comportamento Preguiçoso e sim um comportamento mais agressivo para obter mais benefício naquele ponto. Esta troca de comportamento é prevista no início da missão, dependendo do deadline, porém apenas no decorrer do itinerário é que será possível confirmar se houve chance de se utilizar esse comportamento mais agressivo. A Figura 1 descreve o algoritmo da heurística Lazy-Adaptativo.

```
{Agente}
Variáveis compartilhadas entre agentes:
 1: initialStage = 0;           { nodo inicial da missão }
 2: untestedBehavior = lazy; { comportamento agressivo nao testado }

Variávies locais de cada agente:
 3: behavior = null;          { comportamento padrão do agente }
 4: nextStage = null;         { proximo nodo }
 5: futureStage = null;    {nodo escolhido p/futuro comportamento agressivo}

procedure on ArrivalCalcNextStage ( currentStage, stageGraph,
                                              missionDeadline, history)
 6: if isNewMission(currentStage, initialStage) then
 7:   if isDeadlineReachable(missionDeadline, history) then
 8:     behavior <- getHistoricalBehavior(history)
 9:   else
10:     behavior <- getUntestedMoreAgressiveBehavior(untestedBehavior)
11:   end if
12: end if

13: nextStage <- calcNextStage(behavior, stageGraph)
14: futureStage <- calcFutureStage(nextStage, stageGraph)

15: if isMissionFinished(nextStage, initialStage) then
16:   updateHistory(behavior, history);
17:   untestedBehavior <- createMoreAgressiveUntestedBehavior
                                                  (behavior,futureStage)
18: end if
```

Figura 1. Pseudocódigo da Heurística Lazy-Adaptativo.

Em comparação com a heurística preguiçoso-guloso (Seção 4.1), existe aqui um pequeno *overhead* de processamento no decorrer da missão devido ao cálculo do comportamento do agente efetuado a cada nodo, porém este *overhead* é mínimo, não

afetando o resultado global da heurística e nem o objetivo do agente.

O método *ArrivalCalcNextStage* verifica sempre se uma nova missão está sendo iniciada para assim estabelecer o comportamento que será adotado pelo agente para cumprir a missão: comportamento Preguiçoso, comportamento presente no histórico ou comportamento mais agressivo.

Após estabelecer o comportamento inicial da missão, a cada novo estágio do itinerário, o método calcula o próximo estágio (*calcNextStage*) e também verifica se deseja marcar algum estágio do itinerário para um comportamento mais agressivo no futuro (*calcFutureStage*). Em seguida, acontece uma nova verificação para saber se a missão terminou. Caso tenha terminado, ele atualiza o histórico com o novo comportamento e deadline, e também monta um novo comportamento mais agressivo, com o objetivo de alcançar mais benefícios nas missões futuras, utilizando o comportamento da missão corrente, mais o estágio marcado como *futureStage*, o estágio que representa o melhor avanço em busca de mais benefícios.

5. Avaliação

As heurísticas Preguiçoso, Guloso, Ponderado, PG e Lazy-Adaptativo são avaliadas por meio de simulações. O desempenho da heurística Lazy Adaptativo será comparado às demais heurísticas com o objetivo de confirmar o seu desempenho perante deadlines apertados/curtos. O simulador utilizado para as simulações foi desenvolvido em Java pelo grupo de pesquisa. As condições de simulação são similares às apresentadas em [Rech 2008].

5.1. Condições das Simulações

Foi considerando um sistema composto por 10 nodos e 15 recursos. Os recursos estão alocados de forma fixa nos nodos, sendo que alguns são replicados em outros nodos. A partir disso, foram construídos os diagramas de nodos. Foram definidos 10 segmentos de missão (Diagrama de Recursos) contendo, cada um, relações de precedência e alguns recursos. Para cada segmento foi construído seu diagrama de nodos respectivo. A missão do agente é definida pela composição de 6 segmentos sorteados com reposição a partir deste conjunto de 10 diagramas. Este sorteio oferece 1.000.000 (10^6) possíveis Diagramas de Recursos diferentes. A Figura 2 mostra um exemplo de 1 dos 6 segmentos que poderão ser sorteados.

Para a simulação do tempo total da missão foram considerados: o tempo de fila do processador (com distribuição exponencial cujo valor médio varia de processador para processador com distribuição uniforme de 0 a 20); o tempo de uso do recurso, valor fixo por recurso (porém entre os recursos existe uma distribuição uniforme entre 1 e 15); o tempo de fila nos enlaces de comunicação (com distribuição exponencial, cujo valor médio varia de enlace para enlace entre dois nodos. O atraso médio dos enlaces varia entre 2 e 5 com distribuição uniforme).

Após a execução de cada recurso, algum benefício é recebido e somado ao benefício total da missão. O objetivo da missão é alcançar o maior benefício respeitando o *deadline* especificado. Os benefícios adquiridos pela execução de cada recurso são fixos, sendo $B_0 = 0$, $B_1=1$, $B_2=2$, $B_3=3$, $B_4=4$, $B_5=5$, $B_6=6$, $B_7=7$, $B_8=8$, $B_9=9$, $B_{10}=10$, $B_{11}=11$, $B_{12}=12$, $B_{13}=13$, $B_{14}=14$, $B_{15}=15$.

A qualidade do algoritmo (benefício global do algoritmo *G*) é calculada através da equação:

$$G = \sum (B\_DA / NT) \qquad (4)$$

onde: *B_DA* é o benefício obtido pela execução dos recursos com *deadline* atendidos e *NT* é o número de tentativas.

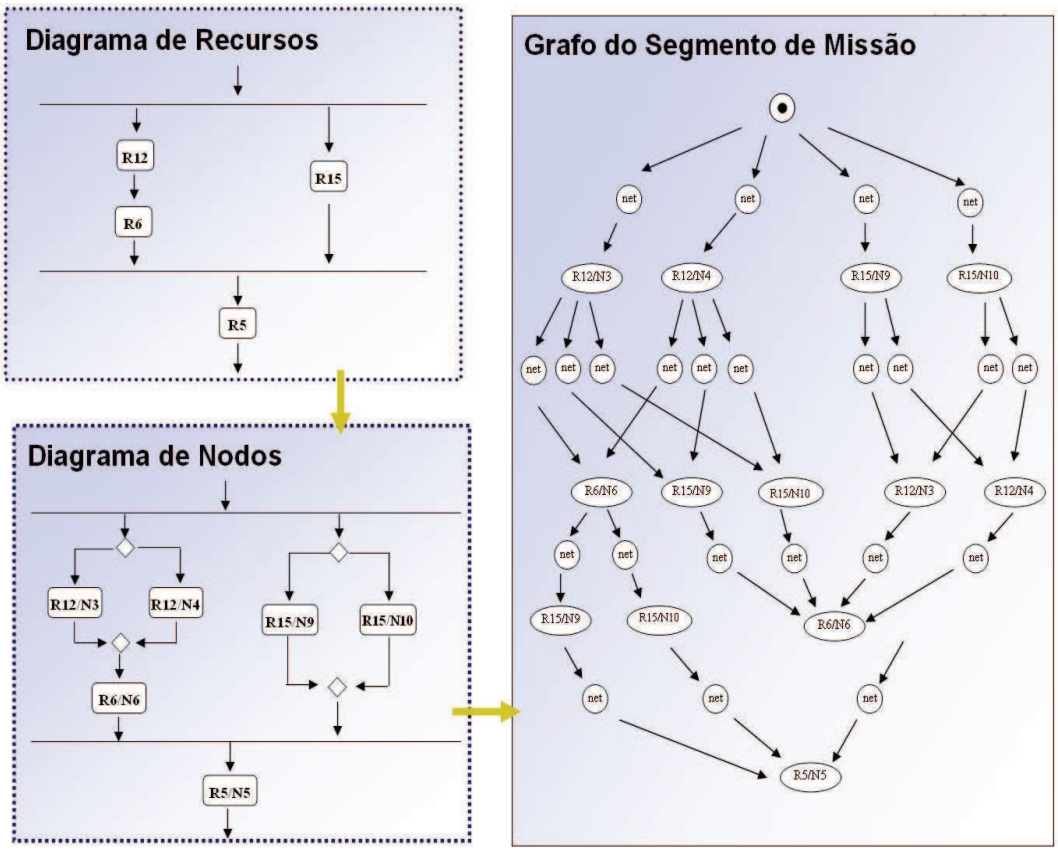

Figura 2. Exemplo do Diagrama de Recursos, Diagrama de Nodos e Grafo de um segmento de missão.

5.2. Resultados das Simulações

Foram simulados 100 diferentes Diagramas de Nodos, cada um com 14 deadlines diferentes. Foram lançados 100 agentes, com 100 diferentes configurações e 14 *deadline*s diferentes, o que resultou em 1400 execuções de cada configuração. Isto representa a execução de 140.000 missões usando cada uma das heurísticas.

A Tabela 1 apresenta os valores do Benefício Global alcançado pelos algoritmos utilizando as 100 diferentes configurações e os 14 diferentes *deadline*s testados. Os valores em negrito indicam as heurísticas que alcançaram os maiores índices de desempenho para cada faixa de *deadline*.

A Figura 3, baseada nos dados descritos na Tabela 1, apresenta o gráfico do Benefício Global G obtido por cada Algoritmo. A amostra utilizada nestas simulações envolve uma ampla variação de configurações de missões dos agentes móveis, fundamentando assim a credibilidade dos resultados apresentados.

Tabela 1. Benefício Global.

	150	200	250	300	350	400	450	500	550	600	650	700	750	800
Preguiçoso	0	0	0	94,55	148,8	155	155	155	155	155	155	155	155	155
Guloso	0	0	0	0	17,28	103,68	**167,04**	**186,24**	**192**	**192**	**192**	**192**	**192**	**192**
Ponderado	0	0	9,3	86,8	148,8	155	155	155	155	155	155	155	155	155
Lazy Adaptativo	0	0	**12,4**	**114,7**	144	135,32	153,61	170,33	**192**	**192**	**192**	**192**	**192**	**192**
P-G	0	0	4,65	89,16	**151,42**	**160,12**	162,88	182,44	**192**	**192**	**192**	**192**	**192**	**192**

Analisando o gráfico da Figura 3, percebe-se que a Heurística Lazy-Adaptativo alcançou os mais altos índices de benefício para situações em que o deadline da missão é considerado apertado (faixa onde o algorítimo guloso não consegue alcançar nenhum resultado), cumprindo assim seu objetivo inicial. Percebe-se também que esta heurística forma o envelope superior do gráfico com deadline folgados.

A partir do instante em que os deadlines de 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.

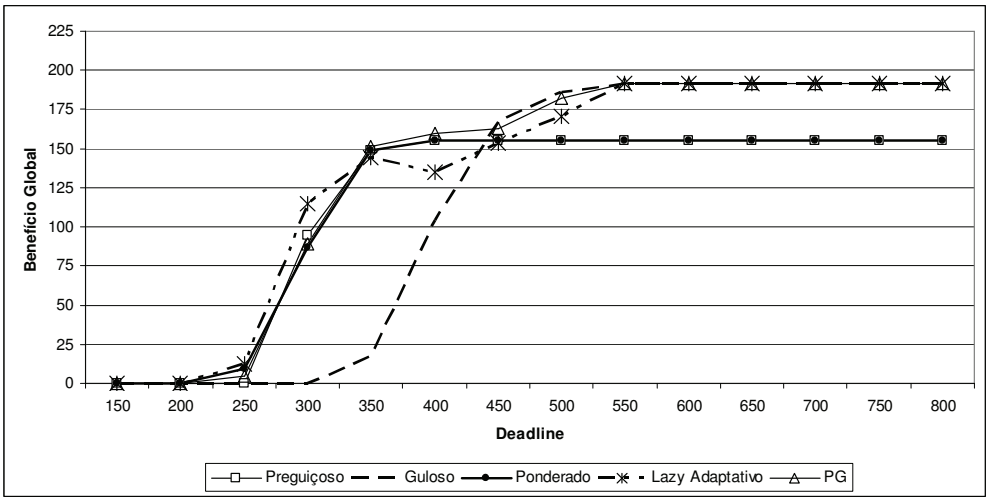

Figura 3. Benefício Global.

Concluindo a análise do gráfico da Figura 3, percebe-se claramente a evolução das heurísticas com relação ao desempenho. As heurísticas simples Preguiçoso, Ponderado e Guloso apresentam um bom resultado apenas para faixas de deadline específicas, respectivamente, deadlines apertados, justos e folgados. A heurística PG apresentou uma melhora significativa quanto ao seu desempenho, apresentando bons resultados para mais de uma faixa de deadline, esta vantagem é conseqüência de sua característica de adaptação na partida. É importante lembrar que uma vez escolhido o comportamento do agente para a missão atual ele permanecerá com este até o final da missão.

A Heurística Lazy-Adaptativo suporta adaptação dinâmica, ou seja, é capaz de mudar seu comportamento em tempo de execução, assim na mesma missão é possível que o agente móvel assuma comportamentos diferentes. Por esta razão, esta heurística mostrou uma curva de evolução mais rápida que a Heurística PG para deadlines apertados.

Outra situação interessante a ser discutida é a questão do tempo de resposta. A Figura 4 exibe o gráfico com os tempos médios de resposta utilizados pelas heurísticas. Observando o gráfico, nota-se que as heurísticas simples gastam o mesmo tempo de resposta para todas as faixas de deadline, sendo que no Algoritmo Guloso há um maior consumo de tempo, o que comprova que esta heurística necessita de *deadline*s mais folgados para obter benefícios relevantes. As demais heurísticas simples movem-se alternadamente dentro de um intervalo de tempo com valores mais baixos.

No gráfico da Figura 4 é possível confirmarmos outra vantagem no uso de heurísticas adaptativas. Para faixas de deadlines apertados estas heurísticas consomem menos tempo, percebendo uma mudança na faixa de deadline, as heurísticas com característica adaptativa procuram aproveitar essa "folga" no deadline, e buscando maiores índices de benefício, permitem-se aumentar o consumo de tempo para concluir a missão. A Tabela 2 apresenta estes valores. A Heurística Lazy-Adaptativo consome o tempo que lhe é permitido, oferecendo bom resultados desde deadlines muito apertados à deadlines mais folgados.

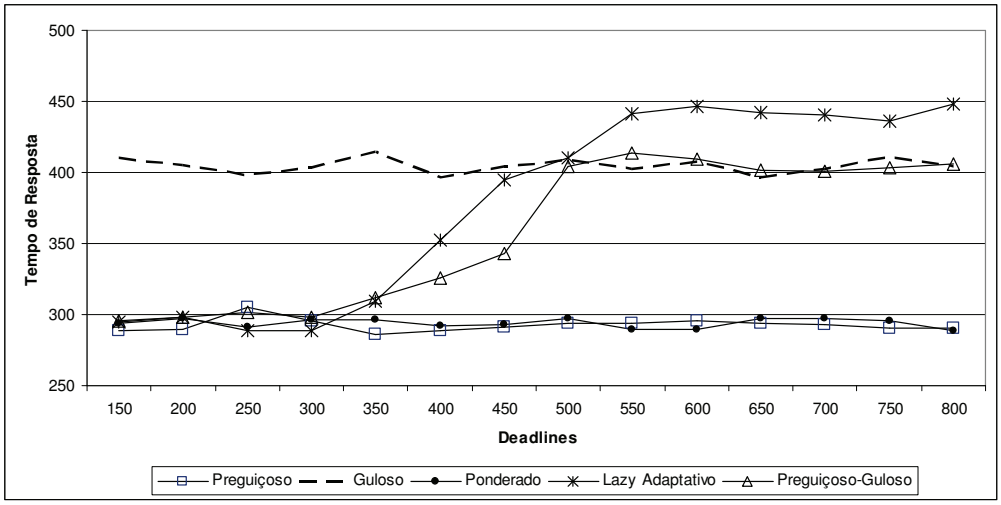

Figura 4. Tempo médio de Resposta das Heurísticas.

Baseando-se nos dados das tabelas e nos gráficos apresentados, pôde-se notar que o algoritmo Lazy-Adaptativo apresentou um desempenho bastante eficiente para deadlines apertados (característica desejável para grande parte das aplicações tempo-real), mostrando um desempenho superior aos algoritmos estáticos (o Preguiçoso, o Ponderado e o Guloso) nesses cenários. O comportamento da heurística mostrou uma característica peculiar de oscilação de resultados, causada principalmente pela limitação de memória (implementado dessa forma para atender cenários de baixa disponibilidade de recursos e melhorar o tempo de viagem pela rede dos agentes móveis).

Tabela 2. Tempo Médio de Resposta

	150	200	250	300	350	400	450	500	550	600	650	700	750	800
Preguiçoso	288,46	289,27	304,78	295,61	286,53	288,85	291,23	293,95	294,22	295,72	294,13	292,78	290,1	290,19
Guloso	410,04	405,56	398,25	403,37	414,67	396,84	404,44	408,99	402,6	407,34	396,97	402,58	411,39	404,04
Ponderado	294,17	297	291,56	296,55	296,59	292,45	293,53	297,23	289,59	290,06	297,77	297,78	295,59	288,53
L.Adaptativo	295,09	298,26	288,99	288,77	309,64	352,89	394,49	410,47	440,96	446,7	442,53	440,16	436,42	448,21
PG	295,51	298,25	302,02	298,28	312,33	325,73	343,45	404,3	413,65	409,15	401,46	400,99	403,17	406,23

Os agentes utilizando a Heurística Lazy-Adaptativo seguem a linha de aprendizado contínuo, visto o tamanho do histórico (armazena as 50 últimas execuções). O algoritmo descarta o histórico de execução mais antigo para guardar novos resultados, causando esse comportamento de oscilação ao se deparar com novas faixas de deadline mais folgadas.

Eventualmente, próximo aos cenários extremamente folgados, o Algoritmo Guloso (de baixo valor de dominância) mostra seu potencial. A Heurística Lazy-Adaptativo também conseguiu se aproximar rapidamente dele e obteve de forma semelhante os resultados máximos.

A Heurística Lazy-Adaptativo apresentou um comportamento positivo para deadlines apertados alcançando os melhores resultados nessa faixa. Para deadlines intermediários (faixa onde o algorítimo guloso alcança resultados positivos, mas estes ainda não são máximos), demonstrou sua capacidade de aprendizado. E por fim, para deadlines folgados, voltou a atingir os resultados com máximo benefício.

6. Conclusões

No contexto de aplicações distribuídas baseadas em agentes móveis, é possível encontrar agentes móveis que devam cumprir um deadline tendo um grau de flexibilidade na definição do itinerário. Neste artigo foi proposta e avaliada uma nova heurística de definição de itinerário. Uma minuciosa análise comparativa permitiu confirmar o bom desempenho desta heurística mais elaborada. A adaptação no decorrer do percurso trouxe maior robustez ao algoritmo. Esta heurística utiliza um histórico de execuções anteriores para mudar o comportamento sugerido na partida da missão, conforme perceba alterações no ambiente.

Todas as heurísticas aqui discutidas oferecem um baixo custo computacional, algo necessário, pois alguns nodos a serem visitados pelo agente móvel podem possuir baixa capacidade de processamento.

O comportamento das heurísticas propostas foi analisado por meio de simulações. Os resultados foram comparados com o intuito de se eleger a heurística que melhor se adaptasse a diferentes situações no sistema distribuído. Todas as heurísticas respeitam a premissa da miopia dos agentes móveis.

O fato das heurísticas simples apresentarem um comportamento definido para cada faixa de deadline tornou possível sugerir o comportamento mais adequado para cada nova missão, uma vez que o deadline desta nova missão fosse conhecido. Portanto, para cada nova missão, levando em conta apenas o deadline da missão atual e a experiência adquirida nas missões anteriores, tornou-se possível escolher o comportamento mais adequado para aquela situação. Os resultados apresentados após as simulações das abordagens adaptativas mostraram desempenho bastante satisfatório, fazendo com que fossem alcançados os maiores índices de benefício dentro das condições de cada missão. A possibilidade de mudar o comportamento em tempo de execução aliado à flexibilidade permitida pelas características de alguns dos recursos que formam a missão proporcionou bons resultados.

O objetivo do uso das abordagens adaptativas é criar uma heurística capaz de ajustar-se conforme a situação do sistema e as necessidades impostas pela missão a ser executada. Portanto, elas não têm por meta superar as heurísticas originais em todas as

faixas de *deadline*s e sim manter bons resultados (próximos ao melhor resultado obtido por cada heurística em cada faixa de *deadline*) para todas as faixas de *deadline*. Este objetivo foi alcançado com a heurística Lazy-Adaptativo.

Referências

Arlow, J., and Neustadt, I. (2005). UML 2 and the Unified Process. Second Edition, *Practical Object-Oriented, Analysis and Design. Object Technologies Series, Addison-Wesley – Series Editors*.

Garvey, A. and Lesser V. (1994). A survey of research in deliberative real-time artificial intelligence. The Journal of Real-Time Systems, 1994.

Gutin, G., and Punnen, A., P. (2002) "The Traveling Salesman Problem and Its Variations", Kluwer Academic Publishers, 2002.

Ilarri, S., Mena, E., Illarramendi, A., (2008) "Using cooperative mobile agents to monitor distributed and dynamic environments", Information Sciences: an International Journal, Volume 178, Issue 9.

Kanoh, H., Hara, K., (2008) "Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network", Proceedings of the 10th annual conference on Genetic and evolutionary computation, Atlanta, GA, USA.

Kasassbeh, A. and Adda, M. (2008) "Analysis of mobile agents in network fault management", Journal of Network and Computer Applications, Volume 35, Issue 4, November, pages 699-711.

Koenig, S. and Likhachev, M. (2006) "Real-time adaptive A*", Int. Conf. on Autonomous Agents. Japão., pag 281-288.

Lange, D. and Oshima, M. (1999). "Seven Good Reasons for Mobile Agents. Communication of the ACM" Vol.42, pag. 88-89.

Liu, J. W. S.; SHIH, W. –k.; LIN, k. –J. et al. "Imprecise Computations". Proceedings of the IEEE. Volume 82. n°1. pp. 83-94. Janeiro. 1994.

Ong, S. K., and Sun, W. W. (2003) "Aplication of Mobile Agents in a Web-based Real-Time Monitoring System", Int. J. Adv. Manuf. Technol. pp.33-40.

Qu, W., Shen, H., and Jin, Y. (2005) "Theoretical Analysis on a Traffic-Based Routing Algorithm of Mobile Agents". IAT 2005 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology. Compiègne. França. pag 520-526.

Rech, L., Oliveira, R. and Montez, C. (2005) "Dynamic Determination of the Itinerary of Mobile Agents with Timing Constraints". IAT 2005 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology. Compiègne. França. pag 45-50.

Rech, L., de Oliveira, R. and C. Montez, (2006) "A Clone-Pair for the Dynamic Determination of the Itinerary of Imprecise Mobile Agents with Firm Deadlines", ETFA 2006 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation, Praga, República Tcheca.

Rech, L., de Oliveira, R. and C. Montez, et. al. (2008) "Determination of the Itinerary of Imprecise Mobile Agents using an Adaptive Approach", ETFA 2008 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Alemanha.

Rech, L., de Oliveira, R. and C. Montez, (2008a) "Itinerary Determination of Imprecise Mobile Agents with Firm Deadlines", Web Intelligence and Agent Systems, An International Journal, Volume 6, n° 4, pages 421-439.

Zgaya, H., Hammadi, S., Ghédira, K., (2008) "A migration strategy of mobile agents for the transport network applications", Mathematics and Computers in Simulation, Volume 76, Issue 5-6.