LUCIANA DE OLIVEIRA RECH

HEURÍSTICAS PARA DETERMINAÇÃO DO ITINERÁRIO DE AGENTES MÓVEIS SOB RESTRIÇÕES TEMPORAIS

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

HEURÍSTICAS PARA DETERMINAÇÃO DO ITINERÁRIO DE AGENTES MÓVEIS SOB RESTRIÇÕES TEMPORAIS

Tese submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Doutor em Engenharia Elétrica.

LUCIANA DE OLIVEIRA RECH

Florianópolis, Maio de 2007.

HEURÍSTICAS PARA DETERMINAÇÃO DO ITINERÁRIO DE AGENTES MÓVEIS SOB RESTRIÇÕES TEMPORAIS

Luciana de Oliveira Rech

Esta tese foi julgada adequada para obtenção do Título de Doutor em Engenharia Elétrica, Área de Concentração Automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

_____
Rômulo Silva de Oliveira, Dr.
Orientador

_____
Kátia Campos de Almeida, PH.D.
Coordenadora do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____
Luiz Affonso H. Guedes de Oliveira, Dr.
Presidente

_____
Carlos Barros Montez, Dr.

_____
Luiz Nacamura Júnior, Dr.

_____
Joni da Silva Fraga, Dr.

_____
Ricardo José Rabelo, Dr.

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais José Aldemar e Nelma e ao meu irmão Roberto, pessoas que representam para mim, a união nos momentos importantes. Vocês são a razão da minha energia, persistência e luta. Mãe, muito obrigada pelo colo e pelas intermináveis horas no telefone durante esses anos. Paizão, seu exemplo de força e sua confiança em mim, me tornaram mais forte e decidida. Betão sua garra e determinação contagiantes sempre me impulsionaram. Não consigo expressar em palavras o quanto eu AMO vocês!!!

Aos meus avós, Nelson e Zenita, exemplos de fé, coragem, amor e perseverança.

Aos meus grandes amigos Gabi, Lea, Guto, Márcia, Karenssita, Déia e Anika, pessoas importantes no conjunto que cerca minha vida. Amigos que me incentivaram em todos os momentos. Pelo "ombro" nos momentos difíceis e pelas risadas nos momentos felizes, o meu muito obrigada. À Terezinha e Seu Arnildo, e a tia Telma e tio Eraldo que proporcionaram que eu me sentisse em família aqui em Floripa.

Ao meu orientador, Rômulo, com quem interagi tantos anos e com quem participei de lutas que me trouxeram cada vez mais experiência e amadurecimento, sem dúvida, um professor no sentido profundo da palavra.

Ao meu co-orientador, Montez, pessoa que participou comigo em vários momentos importantes que superamos com determinação.

Finalmente, em especial, agradeço ao meu Lau, que mesmo entrando em minha vida no período final desse trabalho, sua presença foi essencial em momentos decisivos. Obrigada pelo apoio moral, incentivo e principalmente seu carinho.

Paizão, mãe, Betão, ao pequeno Otávio e Lau, a vocês dedico a alegria desta vitória!

Resumo da Tese apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

Heurísticas para a Determinação do Itinerário de Agentes Móveis sob Restrições Temporais

Luciana de Oliveira Rech

Maio/2007

Orientador: Rômulo Silva de Oliveira, Dr.
Co-Orientador: Carlos Montez, Dr.
Área de Concentração: Automação e Sistemas
Palavras-chave: Agentes Móveis, Tempo Real
Número de Páginas: xiv + 227

O presente trabalho aborda o desenvolvimento de um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrição temporal, considerando a definição dinâmica do itinerário e seu impacto no tempo de resposta da missão. Neste modelo computacional cada agente possui certa flexibilidade na definição de seu itinerário. Esta flexibilidade está relacionada com características dos recursos. Para auxiliar o agente na definição de seu itinerário são propostas heurísticas. Cada heurística confere ao agente um comportamento distinto que, baseado nas diferentes características de cada recurso, é utilizado pelo agente móvel na definição do itinerário. Essas heurísticas podem ser utilizadas individualmente ou em pares/trios (através do uso de clones). Heurísticas mais elaboradas também foram propostas, capazes de escolher seu comportamento considerando um histórico de benefícios conseguidos em execuções passadas do agente móvel. O agente usa adaptação na partida. Uma vez escolhido o comportamento para a missão em questão, ele prossegue com este comportamento até o final da missão. Para realizar a escolha do comportamento foi utilizada probabilidade condicional baseada nas características do ambiente (sistema distribuído) e nos últimos eventos (histórico). As heurísticas propostas foram avaliadas através de simulações.

Abstract of Thesis presented to UFSC as partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

Heuristics for Itinerary Determination of Mobile Agents with Timing Constraints

Luciana de Oliveira Rech

May/2007

Advisor: Rômulo Silva de Oliveira, Dr.
Co-Advisor: Carlos Montez, Dr.
Area of Concentration: Automation and Systems
Keywords: Mobile Agents, Real Time
Número de Páginas: xiv + 227

This work presents the development of a computational model for applications based on imprecise mobile agents with timing constraints. It is considered the dynamic definition of the itinerary and its impact on the response time of the mission. In this computational model each agent has some flexibility in the itinerary definition. This flexibility is related to resource characteristics. In order to assist the agent with its itinerary definition some heuristics are proposed. Each heuristic gives to the agent a distinct behavior that is based on different characteristics of each resource, it is used by the mobile agent for the itinerary definition. These heuristics can be used individually or with pairs/trios (using clones). Heuristics more elaborated are also proposed. These heuristics are able to choose their behavior considering a description of benefits obtained with the last executions of the mobile agent. The agent uses adaptation at the departure. First, the agent behavior is chosen and then the agent goes on (with its behavior) until the end of the mission. In order to make the behavior choice it was used conditional probability based on the characteristics of the environment (distributed system) and the last events (history). The heuristics are evaluated through simulation experiments.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

1.1. Motivação

A mobilidade de código é uma área emergente de pesquisa, a qual pode ser considerada como uma solução promissora para projetar e implementar aplicações distribuídas em larga escala. Ela serve como uma alternativa às abordagens tradicionais e traz benefícios no contexto de vários domínios de aplicação (DOTTI , 2001).

Entre os benefícios esperados da mobilidade de código pode-se listar: 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 (CARZANIGA, 1997). Tais benefícios justificam o crescente interesse nesta área de pesquisa. Alguns domínios de aplicação para os quais a mobilidade de código tem se mostrado promissora são: recuperação de informação, serviços avançados de telecomunicações, controle de dispositivos remotos, gerência de fluxos de trabalho, comércio eletrônico, entre outros.

Dentro do contexto da mobilidade de código, uma importante área de pesquisa atualmente é a tecnologia de agentes móveis. 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 um sistema distribuído. Um agente móvel deve ser capaz de interromper sua execução em um lugar, transferir-se para outro lugar e lá retomar sua execução. Agentes móveis podem reduzir a carga na rede, executam assincronamente e autonomamente, e adaptam-se dinamicamente, estabelecendo assim um novo paradigma para a programação em ambientes distribuídos (PICCO, 1998).

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.

Nesta tese, a questão temporal – aliada à efetiva utilidade da mobilidade de código – é um requisito de importante valor. Sistemas de tempo real são identificados como aqueles sistemas computacionais que devem reagir a estímulos vindos de seu ambiente em

prazos específicos. A computação em tempo real é uma tecnologia fundamental em importantes áreas de aplicação, como controle de processos físicos, plantas de geração de energia, controle de tráfego aéreo, multimídia, entre outras.

No contexto de aplicações distribuídas, existem diversas abordagens que permitem às aplicações de tempo real se adaptarem dinamicamente às várias plataformas. A técnica da computação imprecisa, por exemplo, pode ser empregada 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. Tem-se por objetivo propor um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrições de tempo real e desenvolver algoritmos para a definição do itinerário destes agentes. Esses algoritmos precisam lidar de maneira eficaz com as premissas e restrições assumidas pelo modelo computacional proposto. Define-se o agente móvel como impreciso pela sua flexibilidade na tomada de decisão na execução – ou não execução – de alguns recursos que pertencem a missão deste agente móvel.

A definição do itinerário é um problema clássico de agentes móveis, nesta tese vamos definir o agente móvel como *míope*, ou seja, ele só conhece os nodos que poderão ser visitados um passo a frente. Desta forma, o agente móvel necessita rever seu itinerário planejado após a visita de cada nodo. A questão da miopia é um assunto pouco tratado na literatura. Com relação à definição do itinerário de agentes móveis com requisitos temporais, objeto de estudo desta tese, diversas abordagens foram estudadas e serviram de inspiração ao desenvolvimento de novas heurísticas (novos algoritmos) que suprem algumas lacunas detectadas.

Apesar da literatura tratar dos assuntos relacionados a esta tese – agentes móveis, definição de itinerário e tempo real – individualmente, ou mesmo combinando requisitos destes, não foram identificados trabalhos na literatura que cobrissem completamente os requisitos levantados e modelados nesta tese.

1.2. Objetivos da Tese

O objetivo geral desta tese é propor um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrições de tempo real. Em especial, é considerada a questão da definição do itinerário e seu impacto no tempo de resposta da missão do agente.

Visando atender o objetivo geral desta tese, os seguintes objetivos específicos foram perseguidos:

- Apresentar um estudo da tecnologia de agentes móveis com restrição temporal.
- Desenvolver um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrição temporal.
- Elaborar heurísticas capazes de guiar o agente móvel míope na definição de seu itinerário, auxiliando-o no cumprimento de sua missão enquanto ele respeita o *deadline*.
- Avaliar o comportamento destas heurísticas em diferentes circunstâncias no sistema distribuído.

1.3. Metodologia

1.3.1. Classificação da Pesquisa

Segundo a classificação proposta por (SILVA, 2005), esta tese se enquadra da seguinte forma:

- Quanto à Natureza da Pesquisa: Aplicada, pois gera conhecimentos para aplicação prática que podem ser dirigidos à solução de problemas específicos. Por exemplo, o uso de agentes com restrição temporal em um sistema de geração e distribuição de energia elétrica.
- Quanto à Forma de Abordagem do Problema: Quantitativa. Foi feita uma análise do comportamento do agente através de simulação e também usados os resultados das simulações para classificar e analisar o desempenho das heurísticas propostas.

- Quanto aos Objetivos da Pesquisa: inicialmente Exploratória, porém predominantemente Explicativa, já que inicialmente foi feita uma pesquisa bibliográfica mas apresentando ao final características explicativas.

- Quanto aos Procedimentos Técnicos: Bibliográfica e Experimental. Em um primeiro momento, foi elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e de conferências. Em um segundo momento, a fase de avaliação da abordagem proposta ocorre através de experimentos conduzidos via simulação.

1.3.2. Roteiro da Pesquisa

A partir da classificação previamente apresentada, a pesquisa que culminou com a escrita deste trabalho seguiu os seguintes passos:

1) Revisão bibliográfica da literatura relacionada. Estas informações encontram-se nos capítulos iniciais da tese (Capítulos 2, 3 e 4).

2) Estudo detalhado de propostas que envolvessem assuntos em comum interesse aos desta tese. As informações obtidas estão sintetizadas na Seção 2.5 e no Capítulo 4.

3) Proposição, implementação e avaliação – por meio de simulações – do modelo computacional e das heurísticas para definição de itinerário propostos.

4) Escrita da tese, com base nos dados obtidos anteriormente.

1.4. Adequação às Linhas de Pesquisa do Curso

O trabalho descrito nesta tese está inserido no contexto da Área de Concentração em Automação e Sistemas do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina. Este trabalho está perfeitamente integrado com os demais trabalhos de pesquisa sobre tempo real e com as atividades do Curso de Pós-Graduação em Engenharia Elétrica desta Universidade.

1.5. Organização do Texto

O texto da tese reflete as diversas etapas que foram cumpridas durante o trabalho de doutorado. Uma primeira parte, composta pelos capítulos iniciais, espelha uma etapa de amadurecimento na qual foi necessário o levantamento do estado da arte nas disciplinas de agentes móveis, computação imprecisa e tempo real. Inicialmente é apresentado um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrição temporal, em seguida são apresentados e avaliados diversos algoritmos para definição de itinerário.

Este texto está dividido em 8 capítulos. Este primeiro capítulo descreveu o contexto geral no qual esta tese está inserida, destacando as questões da pesquisa, os objetivos da tese e sua adequação às linhas de pesquisa do curso.

O Capítulo 2 introduz uma série de conceitos sobre agentes móveis, os quais são necessários para o entendimento de todo o trabalho. O texto apresenta: uma caracterização de agentes móveis, os aspectos de sistemas e de ambientes de agentes móveis, e uma argumentação sobre potenciais vantagens no uso de agentes móveis. Também são discutidos alguns trabalhos que introduzem conceitos importantes para esta proposta.

O Capítulo 3 apresenta uma caracterização dos sistemas de tempo real. Algumas abordagens para o problema do escalonamento tempo real são descritas. Também, neste capítulo, a técnica de computação imprecisa é discutida.

O Capítulo 4 mostra um estudo de abordagens existentes na literatura que envolvem o uso da tecnologia de agentes móveis juntamente com requisitos temporais.

O Capítulo 5 contém a descrição do modelo computacional proposto e desenvolvido, sendo esta a primeira contribuição desta tese. São apresentados alguns tópicos que justificam a escolha do tema e também a formulação do problema.

O Capítulo 6 trata da questão da tomada de decisão do agente móvel na definição do itinerário. Este capítulo propõe algumas heurísticas simples capazes de guiar o agente móvel em sua viagem conforme o modelo computacional descrito no Capítulo 5. Cada heurística confere ao agente um comportamento distinto que, baseado nas diferentes características de cada recurso, é utilizado pelo agente móvel na definição do itinerário. Essas heurísticas podem ser utilizadas individualmente ou em pares/trios (uso de clones). A proposição e avaliação dessas heurísticas também é uma contribuição desta tese.

O Capítulo 7 propõe maneiras mais elaboradas de se tratar a questão da definição do itinerário do agente móvel. As heurísticas propostas neste capítulo são uma evolução daquelas apresentadas no Capítulo 6, pois são capazes de escolher seu comportamento considerando um histórico de benefícios conseguidos em execuções passadas do agente móvel. Estas heurísticas são chamadas adaptativas, e o agente móvel usa adaptação na sua partida. Para realizar a escolha do comportamento é utilizada probabilidade condicional baseada nas características do ambiente (sistema distribuído) e nos últimos eventos (histórico). A abordagem adaptativa para definição do itinerário proposta neste capítulo constitui outra contribuição desta tese.

Finalizando, o Capítulo 8 contém as considerações finais sobre o trabalho desenvolvido. São destacados os principais resultados alcançados e identificadas questões relevantes no contexto de agentes móveis e restrição temporal. São também apresentadas perspectivas futuras para este trabalho.

2. Agentes Móveis

2.1. Introdução

A mobilidade de código pode ser definida como a capacidade de uma aplicação distribuída realocar seus componentes em tempo de execução CARZANIGA (1997). Segundo THORN (1997), um código móvel em uma rede de larga escala pode ser visto como um software que viaja através de uma rede heterogênea, atravessando diversos domínios de segurança e sendo executado automaticamente em seu destino.

É importante esclarecer um conceito muito confundido com o de mobilidade de código, o de computação móvel (*mobile computing* ou *nomadic computing*). A computação móvel permite que os usuários, carregando dispositivos portáteis, tenham acesso a uma infra-estrutura compartilhada independente de sua localização física.

Em se tratando de mobilidade de código, uma importante área de pesquisa é a tecnologia de agentes móveis. Um agente é móvel se ele é capaz de se mover de nodo para nodo em uma rede heterogênea, atravessando diversos domínios de segurança e sendo automaticamente executado no seu destino.

Neste capítulo, o paradigma de agentes móveis é apresentado. Na Seção 2.2 suas características, vantagens e desvantagens de seu uso e exemplos de aplicações são discutidos. A Seção 2.3 descreve as definições básicas de agentes e suas funções. Na Seção 2.4 são mencionados e brevemente descritos alguns exemplos de plataformas para agentes móveis. Na Seção 2.5 são discutidas e comparadas abordagens para a definição do itinerário de agentes móveis. Finalizando este capítulo, na Seção 2.6 são feitos os comentários finais.

2.2. Introdução aos Agentes Móveis

Recentemente, devido à rápida expansão da Internet e ao aumento do uso de recursos em rede, um novo paradigma vem surgindo denominado agentes móveis. Por definição, agentes móveis são entidades de *software* autônomas, capazes de viajar através da rede e executar tarefas em nome do usuário (OMG, 2000, FIPA, 2003).

2.2.1. Características de um Agente de *Software*

Um agente de *software* pode ser definido como um processo de execução independente que escolhe autonomamente as ações a executar quando um evento esperado ou inesperado surge, sem a necessidade de intervenção externa de um operador (MARTINS, 2002).

Segundo FRANKLIN (1996), um agente não necessita ser um programa, mas agentes de *software* são, por definição, programas situados dentro de um ambiente de execução que necessitam ter algumas propriedades para serem agentes.

As características e propriedades principais dos agentes móveis ainda não foram completamente formalizadas. Por definição, todo agente deve ser (PICCO, 1998):

- Reativo: entende seu ambiente, percebe mudanças e atua de acordo com essas mudanças;
- Autônomo: tem o controle de suas próprias ações;
- Orientado a objetivos ou Pró-ativo: não responde simplesmente em resposta ao seu ambiente. É capaz de exibir um comportamento orientado por certos objetivos e tomar a iniciativa;
- Temporariamente contínuo: é um processo continuamente em execução.

Além dessas características, os agentes podem possuir as seguintes características (PICCO, 1998):

- Comunicativo ou Sociável: capacidade de se comunicar com outros agentes;
- Adaptativo: capacidade de aprender, muda de comportamento baseado em suas experiências anteriores;
- Mobilidade: habilidade para se transportar de uma máquina para outra;
- Caráter: possui personalidade e estado emocional;
- Agilidade: capacidade de aproveitar rapidamente novas oportunidades não previstas.

Dificilmente um agente possui simultaneamente todas as características descritas anteriormente (MARTINS, 2002), no entanto, vários agentes possuem grande parte delas. A importância de cada uma delas depende da comunidade em que o agente está inserido.

Por exemplo, na Robótica interessa o comportamento reativo e a interação com o mundo físico, para IA (Inteligência Artificial) interessam as características como o raciocínio e a aprendizagem, para Sistemas Distribuídos a mobilidade é a característica vital.

Como mobilidade é uma característica não obrigatória de agentes, um agente pode apenas se comunicar através de meios convencionais que incluem várias formas de chamadas de procedimento remoto RPC (*Remote Procedure Call*) e trocas de mensagens, esses agentes são conhecidos como estacionários. Segundo a OMG (*Object Management Group*) (OMG, 2000), um agente estacionário executa apenas no sistema onde este iniciou sua execução. Se o agente necessita de informações que não estão no seu sistema, ou necessita interagir com um agente que está em um sistema diferente, o agente utiliza mecanismo de comunicação, como por exemplo, o RPC. O suporte necessário para comunicação de agentes estacionários é fornecido pelos sistemas de objetos distribuídos correntes, como CORBA (*Common Object Request Broker Architecture*), e JAVA RMI (*Remote Method Invocation*).

Agentes móveis são utilizados em sistemas em rede com o objetivo de facilitar o controle e a realização de tarefas distribuídas e coordenadas entre indivíduos e sistemas. Um agente móvel é um agente de *software* que não se limita ao sistema no qual iniciou sua execução, seu fundamento principal está na capacidade de transferir o seu programa, código de execução, de um ponto a outro da rede e reiniciar ou continuar sua execução neste novo ponto. Geralmente são programados em linguagens interpretadas (como Java, TCL ou *Pearl*) devido à necessidade de portabilidade (RUBINSTEIN, 2001).

O fato de um agente móvel mover-se pela rede sobre seu próprio controle, migrando de um sítio para outro e interagindo com outros agentes, faz com que seja considerado um agente autônomo e com potencial para fornecer um paradigma de programação conveniente, eficiente e robusto para aplicações distribuídas (GRAY, 1996).

Agentes móveis são um paradigma efetivo para aplicações distribuídas, e são particularmente atrativos para computações conectadas parcialmente (GRAY, 1996). Dispositivos conectados parcialmente incluem computadores móveis como *laptops* e PDAs (*Personal Digital Assistants*), assim como computadores em casa ou escritório que ocasionalmente estão conectados à rede. Todos estes dispositivos são freqüentemente desconectados por longos períodos de tempo, freqüentemente possuem baixa largura de banda e as conexões nem sempre são confiáveis.

Um agente móvel, por exemplo, pode migrar de um nodo origem e navegar pela Internet para reunir informações para seu usuário. Pode acessar os recursos necessários eficientemente já que viaja pela rede ao invés de transferir múltiplos pedidos e receber respostas de conexões com baixa largura de banda. Por não permanecer continuamente em contato, o agente não é afetado se repentinamente a conexão for perdida; e pode continuar sua tarefa se o nodo origem for desligado ou desconectado. Quando o usuário reconectar, o agente retorna com o resultado.

Existem linguagens e sistemas que fornecem mecanismos habilitando e suportando mobilidade de código. As linguagens de código móvel diferem na forma com que suportam a mobilidade. Algumas características necessárias que devem ser consideradas pelas linguagens de programação de códigos móveis são (THORN, 1997): **portabilidade**; **segurança de funcionamento** como integridade, disponibilidade e garantia de confiabilidade; e **eficiência**. *Java Aglets* (IBM), *AgentTcl* ou *D'Agents* (*Dartmouth College*, EUA), TACOMA (*Troms And COrnell Mobile Agents*), *Telescrip* (*General Magic*) são exemplos de linguagens de código móvel.

2.2.2. Mobilidade

A mobilidade aparece como característica fundamental dos agentes móveis, sendo que estes podem iniciar sua transferência para outro local pela execução de uma instrução específica em meio ao seu próprio código levando consigo seu código e seus estados internos. Sua migração é feita através da infra-estrutura de rede. No destino, o programa é reiniciado e deve conservar as mesmas informações que tinha na origem, de forma que possa continuar a execução.

A movimentação pode ser realizada a partir de duas ações distintas: a primeira, chamada de **translação**, é a transferência com base na realização de uma cópia no destino e a destruição do agente móvel na origem; a segunda chama-se **retração**, consiste na solicitação de retorno de um agente móvel que já migrou para outro local (BRAGA, 2000).

A forma como o estado e as informações são transferidas (durante o processo de migração) e reativados pode assumir dois padrões (PICCO, 1998, MILOJICIC, 1999):

- **Mobilidade forte ou transparente**: quando o agente móvel reinicia a execução em seu novo destino exatamente com o mesmo estado e na mesma posição de código que

ele estava executando antes de sua migração. É a habilidade que permite a migração do código e de todo o estado de execução de uma unidade de execução para um ambiente computacional diferente;

- **Mobilidade fraca ou não-transparente**: o código do agente móvel é transferido juntamente com seus dados para o destino, porém o estado de execução não é transferido. O código pode ser acompanhado por algum dado de inicialização, mas não há migração de todo o estado de execução.

A mobilidade forte é suportada por dois mecanismos: migração e clone remoto. O mecanismo de migração suspende uma unidade de execução[1], a transmite para o ambiente computacional destino e então a retoma novamente. O mecanismo de clone remoto cria uma cópia de uma unidade de execução em um outro ambiente computacional. O que o diferencia do mecanismo de migração é que após a clonagem da unidade de execução para um novo ambiente computacional, a unidade de execução original não é mais gerenciada pelo ambiente computacional onde a unidade de execução clonada se encontrava (PICCO, 1998).

Mecanismos que suportam mobilidade fraca possuem a capacidade de transferir o código através de ambientes computacionais e ainda ligá-los dinamicamente com a unidade de execução ativa, ou usá-los como segmento de código para uma nova unidade de execução. Com relação à direção da transferência do código, uma unidade pode buscar (*fetch*) o código para ser dinamicamente ligado e/ou executado, ou pode ainda enviar (*ship*) o código para outro ambiente computacional. Este código migrado pode ser *stand-alone*, o código é autocontido e será usado para instanciar uma nova unidade de execução sobre o sítio destino, ou fragmentado, o código deve ser ligado ao contexto de um código que já está em execução e eventualmente executá-lo. A comunicação pode acontecer de forma síncrona ou assíncrona, dependendo se a unidade de execução que pede a transferência suspende ou não o código que está sendo executado (PICCO, 1998).

---

[1] Uma unidade de execução representa qualquer fluxo seqüencial de computação, como por exemplo, uma *thread* ou um agente.

2.2.3. Tipos de Agentes

Muitos fatores devem ser considerados para classificar corretamente os agentes. As características mencionadas no item anterior auxiliam nesta classificação.

Quanto à sua mobilidade, podem ser classificados como **móveis** ou **estáticos**. Os agentes móveis possuem habilidade de mover-se através de uma rede, isto é, não ficam restritos ao sistema onde iniciam sua execução. Agentes estacionários têm sua execução restrita ao sistema onde a execução foi iniciada.

Os agentes podem também ser **reativos** ou **deliberativos**. Caso sejam reativos, agem unicamente quando recebem algum estímulo alterando seu estado dentro do ambiente em que se encontram. Quando deliberativos, possuem um modelo simbólico interno e podem se engajar em um planejamento ou negociação junto com outros agentes, visando o cumprimento da tarefa (LOBO, 2001).

Podem ainda ser classificados de acordo com alguns parâmetros ideais ou primários que os agentes devam possuir, como: **autonomia**, **cooperação** e **aprendizagem**. De acordo com Nwana (NWANA, 1996), estas três características podem ser utilizadas para definir quatro tipos de agentes (Figura 2.1), descritos a seguir. Para cada combinação de duas características é definido um tipo de agente o que leva à definição de três tipos de agentes. Quando o agente possuir as três características é definido um quarto tipo.

- Autonomia e aprendizagem: **agentes de interface;**
- Cooperação e aprendizagem: **agentes colaborativos com capacidade de aprendizagem;**
- Autonomia e cooperação: **agentes cooperativos;**
- Autonomia, cooperação e aprendizagem: **agentes inteligentes**.

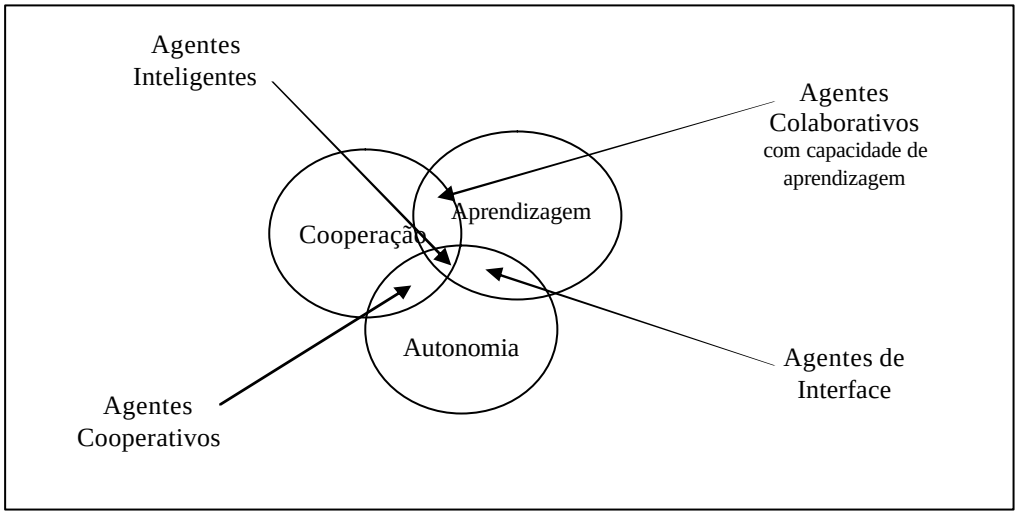


Figura 2.1: Visão Parcial de uma Tipologia de Agentes (NWANA,1996)

Considera-se **híbrido** o agente que utiliza características de dois ou mais tipos diferentes. Pode ser colaborativo, de interface, móvel, inteligente ou reativo (LOBO, 2001).

2.2.4. Vantagens e Desvantagens do Uso de Agentes Móveis

Em alguns casos os agentes móveis apresentam vantagens sobre as abordagens convencionais. Algumas dessas possíveis vantagens são (MARTINS, 2002, LANGE, 1998, ZHOU, 2005):

- **Redução do tráfego na rede**: agentes móveis permitem empacotar uma negociação e transferi-la para um sítio destino, para que as interações sejam realizadas localmente;

- **Diminuição da latência da rede**: agentes móveis são uma boa alternativa para interações com entidades de tempo real, onde latência e atrasos não podem ser tolerados. Uma vez que o agente está no mesmo sítio que os recursos invocados, esta interação pode assumir requisitos de tempo real;

- **Execução assíncrona e autônoma**: agentes móveis podem interagir de forma assíncrona e autônoma, independente do programa que o enviou;

- **Adaptação dinâmica**: agentes móveis têm a habilidade de sentir o seu ambiente de execução e reagir autonomamente às mudanças. Possuem habilidade de se distribuir

entre os sítios da rede de forma a manter uma configuração ótima para resolver um problema particular;

- **Robustez e tolerância a faltas**: devido à capacidade dos agentes móveis reagirem a situações e eventos desfavoráveis é facilitada a construção de sistemas distribuídos robustos tolerantes a faltas. Por exemplo, se um sítio estiver saindo do ar todos os agentes em execução na máquina podem ser alertados para serem despachados para outro sítio da rede para continuar a operação;

- **Lidar com grandes volumes de dados**: quando vastos volumes de dados são armazenados em localizações remotas, o processamento destes dados pode ser feito localmente em vez de serem transmitidos totalmente pela rede;

- **Naturalmente heterogêneos**: a execução de agentes móveis faz-se normalmente em ambientes baseados em máquinas virtuais o que facilita sua portabilidade. Como agentes móveis são geralmente independentes da camada de transporte e do computador, e dependem apenas de seu ambiente de execução, estes fornecem boas condições para a integração de sistemas.

Apesar dos benefícios, a utilização do paradigma de agentes móveis também possui desvantagens potenciais. A primeira é a necessidade de sistemas adicionais para o suporte ao paradigma. Para o processamento de agentes móveis geralmente são utilizadas plataformas de mobilidade. O uso de uma plataforma tem conseqüências tanto no uso individual de cada *host* como no aumento da complexidade das aplicações.

Outro fator bastante preocupante é a questão da segurança, não é confiável confiar irrestritamente em códigos convenientes de outras máquinas, sendo necessária alguma espécie de autenticação para garantir a confiabilidade do código ou restrições de acesso aos recursos da máquina. Em geral, as plataformas de mobilidade não tratam devidamente o requisito de segurança (KARNIK, 1998).

2.2.5. Aplicações de Agentes Móveis

Pode-se argumentar que agentes móveis não são uma tecnologia imprescindível já que poucas aplicações (se houver) são impossíveis de serem realizadas sem o uso de agentes móveis (HARRISON, 1995).

As vantagens dos agentes móveis conduzem à melhora do desempenho de aplicações distribuídas, onde desempenho está relacionado à utilização da rede, ao tempo de computação, a conveniência para o programador ou a capacidade de continuar interagindo com o usuário durante uma desconexão de rede. Agentes móveis são vistos como uma ferramenta geral para realizar aplicações distribuídas arbitrárias. Isto é refletido pela diversidade de aplicações nas quais agentes móveis são utilizados (GRAY, 1996).

Os agentes de *software* têm-se revelado extremamente úteis em áreas de aplicação como:

- Aplicações de caráter lúdico como jogos interativos ou teatros virtuais;
- Aplicações na área militar ou proteção civil como simulação de combates, desastres naturais e outros;
- Em sistemas de suporte de ensino, por exemplo, descrevendo um procedimento, respondendo a perguntas e guiando o aluno na execução do procedimento;
- Em aplicações de busca inteligente na Internet, tanto na investigação (SONG, 2005) como em aplicações comerciais (GELOWITZ, 2005);
- Gestão de redes e serviços, gerenciamento de controle de fluxo, gerenciamento de recursos distribuídos (RAIBULET, 2000), comércio eletrônico (ZHOU, 2005), gestão de informação pessoal;
- Recuperação de informação (GLITHO 2002, BREWINGTON 1999, YANG 1998);
- Roteamento e fusão de dados em redes de sensores sem fio (CHEN 2005, CHEN 2006, TONG 2003).

Também podem ser utilizados em Sistemas em Grid (Grades Computacionais), onde existe a necessidade de se executar tarefas complexas, mas ao mesmo tempo é necessária confiabilidade e baixo *overhead*. Em sistemas em Grid, a tecnologia de agentes tem sido utilizada de diferentes maneiras, como: previsão de desempenho, escalonamento e serviço de monitoramento (HE, 2005). Em monitoramento de Grid, os agentes móveis estão associados com a redução da carga na rede, execução independente e assíncrona das tarefas, interação de recursos heterogêneos, algoritmos adaptados para análise dinâmica de

dados executados sob demanda do usuário. Podem também ser utilizados para serviço de busca em ambiente em Grid (ROBLES, 2002).

2.3. Modelo Conceitual Comum da OMG para Agentes Móveis

A OMG (*Object Management Group*) publicou em 1998 um documento que trata da interoperabilidade entre sistemas de agentes móveis chamado MASIF (*Mobile Agent Specification Interoperability Facility*). A versão atualizada deste documento passou a se chamar MAF (*Mobile Agent Facility*) e foi publicada em 2000 (OMG, 2000). Neste documento, a OMG apresenta um modelo conceitual comum para definir conceitos básicos relativos ao paradigma de agentes móveis.

A especificação MAF (GRAY, 1996) não trata da interoperabilidade de linguagens, mas sim da interoperabilidade entre sistemas de agentes escritos na mesma linguagem, porém por desenvolvedores diferentes. A interoperabilidade se torna possível de ser alcançada se ações como transferência de agentes, transferência de classes e gerenciamento de agentes são padronizadas. Operações locais de agentes, como interpretação de agentes, serialização ou execução não são padronizadas pela especificação.

2.3.1. Definições Básicas de Agentes

- **Autoridade do Agente:** identifica a pessoa ou organização para quem o agente atua (proprietário do agente);

- **Nomes de Agentes:** em um sistema onde seja prevista a execução de vários agentes, é necessário identificar unicamente cada agente móvel, de forma a permitir operações de gerenciamento e localização;

- **Lugar ou Contexto:** quando um agente se transfere, ele viaja entre ambientes de execução chamados lugares. Um lugar é um contexto dentro de um sistema de agentes no qual um agente pode ser executado. O lugar fonte e o lugar destino podem residir em um mesmo sistema de agentes ou em sistemas de agentes diferentes que suportem o mesmo perfil de agente. Um sistema de agentes pode conter um ou mais lugares e um lugar pode conter um ou mais agentes (Figura 2.2);

• **Localização do Agente:** a localização de um agente é o endereço de um lugar. Um lugar reside dentro de um sistema de agente. Uma localização de agente deve conter o nome do sistema de agente onde o agente reside e o nome do lugar;

• **Sistemas de Agentes Móveis:** constituem-se de uma plataforma que possibilita criar, interpretar, executar, transferir e finalizar agentes, sendo associados a uma autoridade e identificados normalmente por um nome e endereço (PICCO, 1998a). Um sítio pode conter um ou mais sistemas de agentes. Um tipo de sistema descreve o perfil de um agente.

Um sistema de agentes é formado basicamente por duas entidades: os agentes propriamente ditos e os lugares (FRANKLING, 1996). A Figura 2.2 apresenta um sistema de agentes.

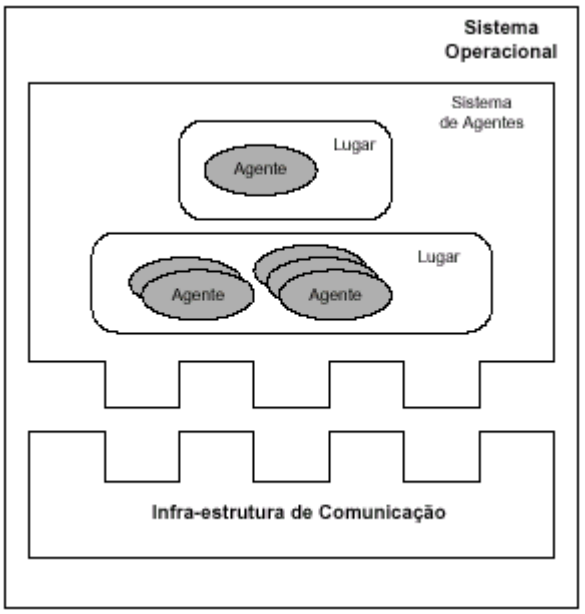


Figura 2.2: Sistema de Agentes (OMG, 2000).

• **Regiões:** uma região é um conjunto de sistemas de agentes que tem a mesma autoridade, mas não são necessariamente do mesmo tipo de sistemas de agentes. Uma região fornece um nível de abstração para comunicação com clientes de outras regiões. Cada região contém um ou mais pontos de acesso e é por estes pontos que as regiões são interligadas para formar uma rede (Figura 2.3).

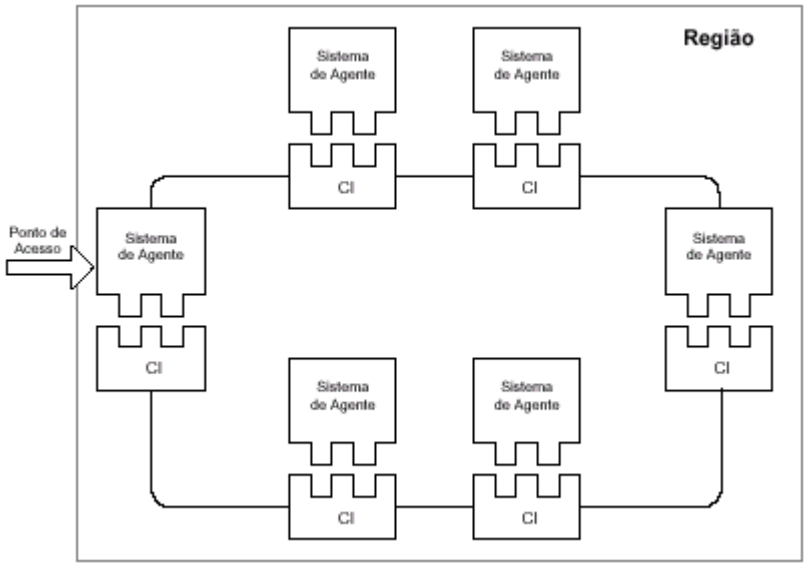


Figura 2.3: Arquitetura de uma Região (OMG, 2000).

As regiões são interconectadas via uma ou mais redes. Um sistema não baseado em agentes pode também se comunicar com sistemas de agentes dentro de qualquer região onde o sistema não-agente tem autorização para atuar.

- **Infra-estrutura de Comunicação:** uma infra-estrutura de comunicação (CI) fornece serviços de comunicação (por exemplo, RPC), serviço de nomes e de segurança para um sistema de agentes. É através dela que toda a comunicação entre sistemas de agentes acontece;

- **Serialização/Deserialização:** serialização é o processo de converter o agente em uma forma serializada, isto é, uma seqüência de bytes para transmitir pela rede. Deserialização é o processo de restaurar (colocar de volta no estado original) o agente a partir de sua forma serializada;

- ***Codebase*:** o *codebase* especifica a localização das classes usadas por um agente. Se um sistema de agentes é responsável por fornecer as classes necessárias, o *codebase* deve ter informação suficiente para localizar o sistema de agente.

2.3.2. Funções de um Sistema de Agentes Móveis

Ao longo da vida de um agente móvel acontecem três eventos principais: criação, migração ou transferência e finalização. Além desses eventos, um sistema de agentes móveis também executa algumas funções que serão descritas a seguir.

Criação de um Agente

A criação de um agente é caracterizada pela instanciação e identificação, isto é, o código do agente é carregado dinamicamente no lugar pretendido e é criada uma instância da classe do agente ao qual é atribuído, pelo próprio lugar, um identificador único (MARTINS, 2002).

O processo de criação de um agente móvel pode ser solicitado por um usuário, disparado por um evento externo ou até mesmo por outro agente móvel (MILOJICIC, 1999).

Transferência ou Migração de um Agente

O processo de transferência de um agente móvel inicia com o pedido, feito por parte do próprio agente móvel que deseja se transferir. A partir deste pedido, o lugar suspende a execução do agente e envia para o lugar de destino uma requisição de transferência. Durante o processo de transferência dos agentes, o lugar de destino e a qualidade do serviço de comunicação são identificados. Quando o sistema destino aceita a transferência, o estado do agente, juntamente com sua autoridade, credenciais de segurança e o código são transferidos para o destino, conforme ilustra a Figura 2.4.

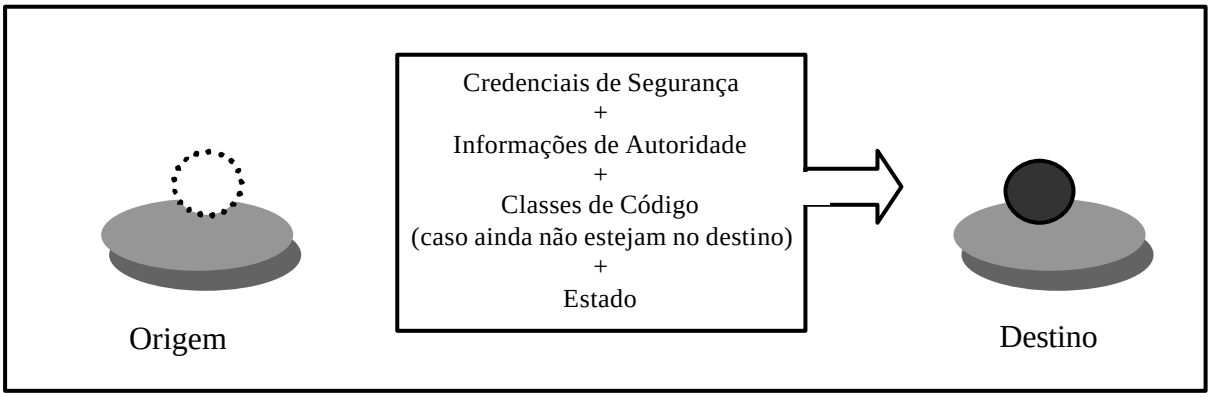


Figura 2.4: Elementos enviados no Processo de Transferência do Agente Móvel (REIS, 2001).

Esta transferência é realizada pela codificação destas informações em um protocolo de transferência pré-definido. Já no destino, os pacotes recebidos são decodificados, o processo do agente é reconstruído e sua execução é reiniciada

(MILOJICIC, 1999). Todo este processo pode ser resumido pelo esquema apresentado na Figura 2.5.

Finalização

A finalização de um agente móvel, que é o processo onde a execução do agente móvel é terminada e conseqüentemente seus dados são perdidos, pode acontecer pelos seguintes motivos (LANGE, 1998): término da tarefa; redundância, dois agentes móveis percebem que estão realizando a mesma tarefa no mesmo lugar, um deles pode decidir por sua finalização; aniquilação, um agente de *software* regulador detecta que um agente móvel deve ser finalizado com base em diretivas para evitar superpopulação ou execução com uso excessivo de recursos; iniciativa própria, quando por exemplo, o agente móvel detecta que seu objetivo não pode ser alcançado.

Assim como no processo de criação, a operação de finalização pode ser solicitada por um usuário, disparada por um evento externo ou até mesmo por outro agente móvel.

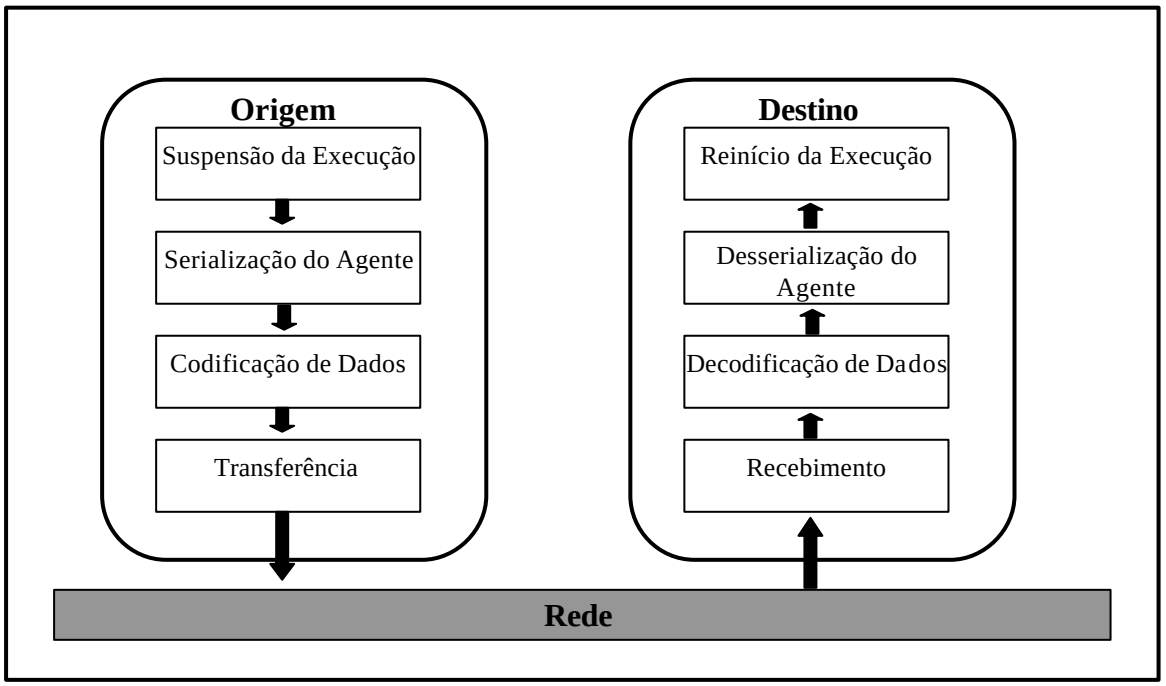


Figura 2.5: Arquitetura para Transferência de Agentes em um Sistema Multiagente (LANGE, 1998).

Suspensão

Durante todo o tempo de execução, um agente móvel pode submeter-se a procedimentos que suspendem temporariamente sua execução, que variam de acordo com o sistema utilizado. Os dados pertencentes aos agentes podem ser armazenados e são recuperados quando acontece o reinício da execução. Porém, sistemas mais simples podem implementar funcionalidades de suspensão que não preservam os dados (suspensão volátil) (LANGE, 1998).

Comunicação

A comunicação entre agentes é definida pela sua forma de transmissão (*unicast*, *broadcast*, *multicast* e *forward casting*), endereçamento (direta, indireta ou caixa postal), e conexão (síncrona ou assíncrona).

A comunicação entre agentes móveis é provida na maioria dos sistemas com métodos ou funções invocadas no nível de linguagem de programação. As mensagens podem ser transmitidas de quatro maneiras (BRAGA, 2000):

- ***Unicast****:* o agente móvel utiliza uma identificação para enviar a mensagem diretamente ao agente móvel desejado, Figura 2.6.

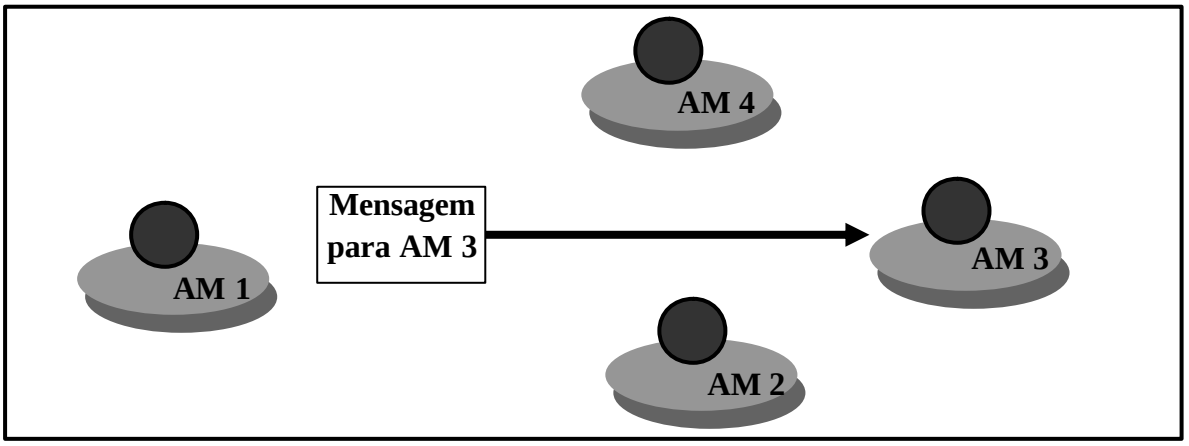


Figura 2.6: Comunicação *unicast* entre Agentes (MILOJICIC, 1999).

- ***Broadcast***: o agente móvel envia a mensagem para todo o grupo de agentes móveis, os quais irão interpretar e verificar se a mensagem é válida naquele contexto que se encontra, Figura 2.7.

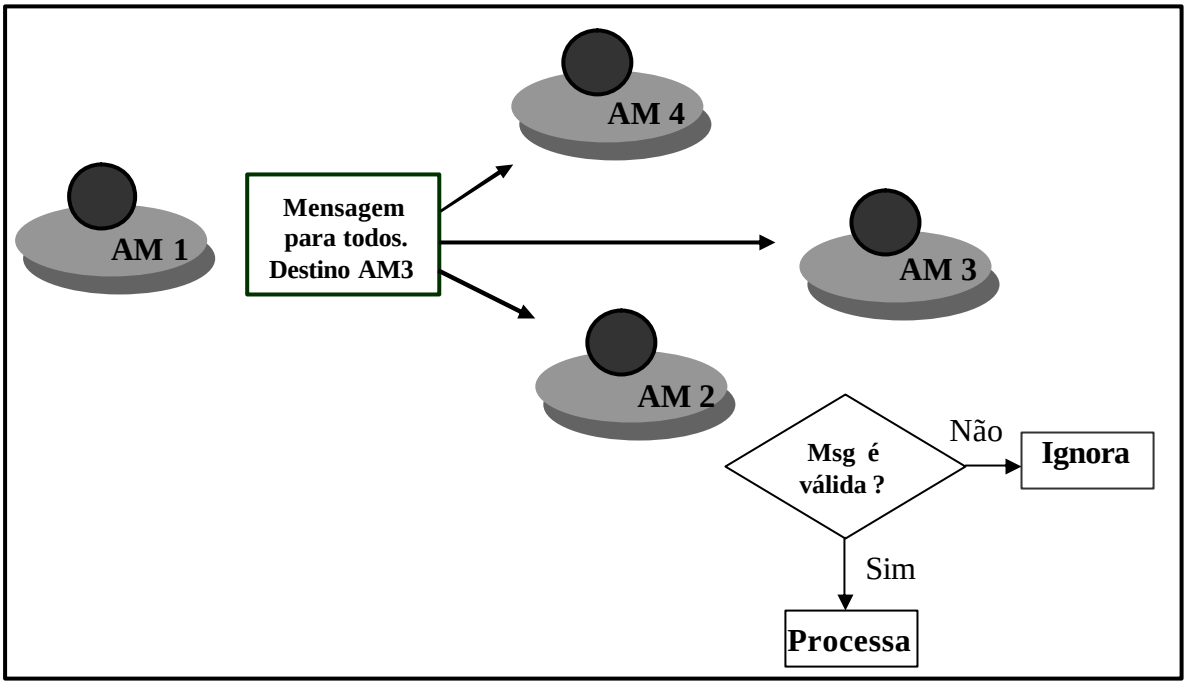


Figura 2.7: Comunicação *broadcast* entre Agentes (MILOJICIC, 1999).

• ***Multicast***: os agentes móveis são inscritos em grupos de serviços, que são o alvo das mensagens, Figura 2.8.

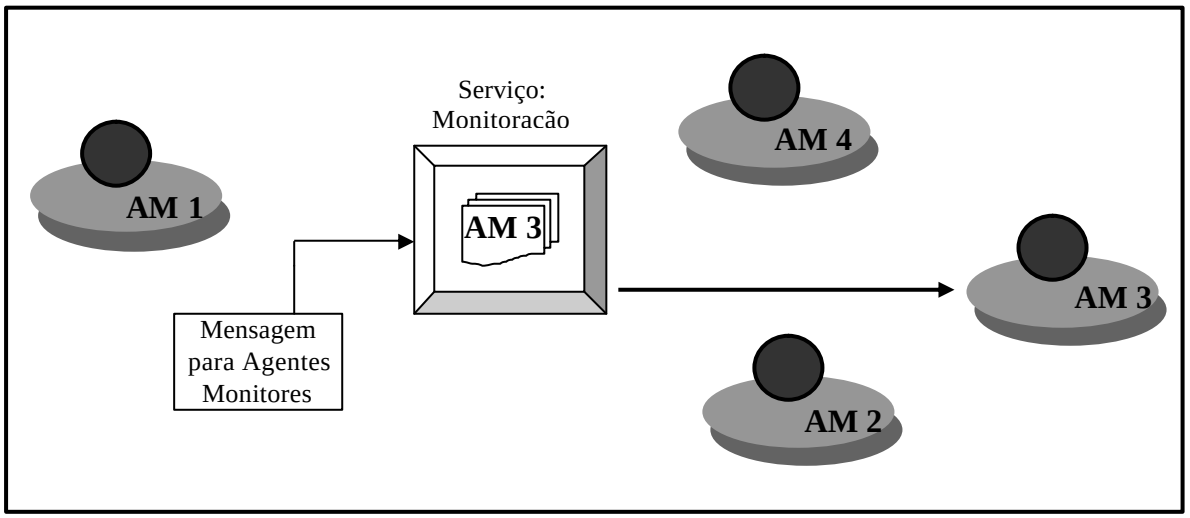


Figura 2.8: Comunicação *multicast* entre Agentes (MILOJICIC, 1999).

• ***Forward casting***: a mensagem é enviada para o agente móvel mais próximo, o qual por sua vez a repassa até que chegue ao seu destinatário, Figura 2.9.

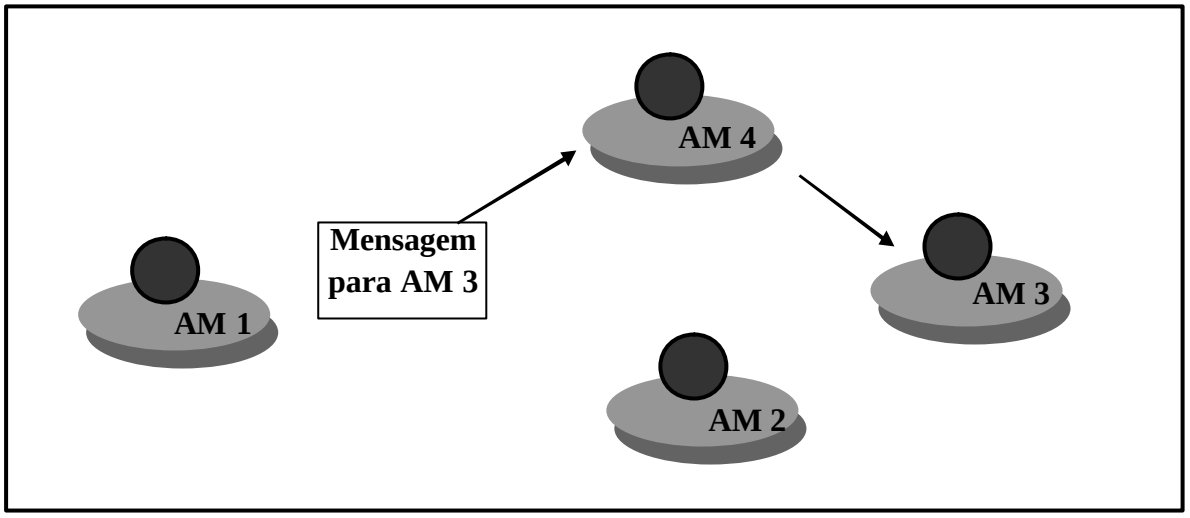


Figura 2.9: *Forward casting* entre Agentes (MILOJICIC, 1999).

A comunicação em um sistema de agentes móveis também pode variar de acordo com a forma de endereçamento, ou seja: direta (os agentes móveis enviam mensagens diretamente uns aos outros), indireta (as mensagens são enviadas às agências, ou locais, que se encarregam de repassá-las ao destinatário) ou através de mecanismos que funcionam como uma caixa postal, que são repositórios de mensagens periodicamente consultadas pelos agentes móveis (BRAGA, 2000).

Alguns ambientes implementam a funcionalidade de comunicação de forma síncrona, ou seja, o agente móvel obrigatoriamente deve suspender sua execução e aguardar a resposta de sua mensagem. Em contrapartida, outros ambientes permitem que os agentes móveis continuem executando suas tarefas enquanto não recebem sua resposta (LANGE, 1998).

2.4. Abordagens para a Definição do Itinerário de Agentes Móveis

Tomando como base o uso da tecnologia de agentes móveis e suas principais características como: capacidade de reduzir o tráfego na rede, superar latências, aumentar a confiança e a capacidade de tolerância a faltas para aplicações distribuídas (OSHIMA, 1998), serão apresentadas nesta seção alguns exemplos de abordagens para a definição do itinerário de agentes móveis em ambientes distribuídos.

2.4.1. Arquitetura de um Sistema de Planejamento de Agentes Móveis

Uma forma para se recuperar informação em um sistema distribuído seria enviar um agente móvel para cada nodo destino, assumindo que cada agente despachado poderia encontrar a informação cuja tarefa de recuperação lhe foi incumbida.

Um dos problemas na recuperação de informação é a possibilidade de um agente não ser capaz de encontrar a informação desejada no nodo destino. O planejamento proporciona a utilização de menos recursos de rede enviando o menor número de agentes móveis possíveis (número de agentes inferior ao número de nodos a serem visitados). Neste caso existe a necessidade de planejar a melhor seqüência (itinerário) dos nodos a serem visitados por cada agente a fim de que a informação desejada possa ser encontrada no menor tempo (BREWINGTON, 1999).

Em (BREWINGTON, 1999) é proposta uma arquitetura de sistema de planejamento de agentes móveis onde o itinerário dos agentes é determinado por planejamento e é baseado em três aspectos: uma lista de nodos onde um agente pode ser capaz de encontrar a informação desejada; a incerteza da qualidade dos dados disponíveis nestes nodos e as condições atuais da rede.

As condições da rede são obtidas através do módulo de sensoriamento da rede (*network-sensing module*), são elas: informações relacionadas à conectividade dos enlaces, à operabilidade dos nodos na rede, à latência e à disponibilidade de largura de banda entre os enlaces.

A arquitetura do sistema de planejamento de agentes proposta por (BREWINGTON, 1999) é descrita na Figura 2.10. O sistema de planejamento consiste de três componentes principais: o **módulo de planejamento**, o **módulo de sensoriamento da rede** e o **módulo páginas amarelas** (*yellow-pages module*).

Neste sistema, quando um agente é incumbido de buscar uma informação, ele consulta primeiramente o módulo de planejamento. O módulo de planejamento então consulta o módulo páginas amarelas para verificar as possíveis localizações (possíveis nodos) onde o agente móvel poderia encontrar a informação desejada.

Depois de obter a lista dos nodos e suas correspondentes probabilidades de sucesso, o módulo de planejamento passa a lista para o módulo de sensoriamento da rede o

qual retorna as latências e largura de banda entre os nodos e suas atuais cargas de CPU. O módulo sensitivo da rede mantém estas estatísticas da rede em intervalos fixos de tempo.

Assim que as estatísticas da rede são retornadas para o módulo de planejamento, a seqüência que os agentes visitarão os nodos (para minimizar o tempo de execução total) é calculada utilizando as estatísticas da rede e as probabilidades de sucesso.

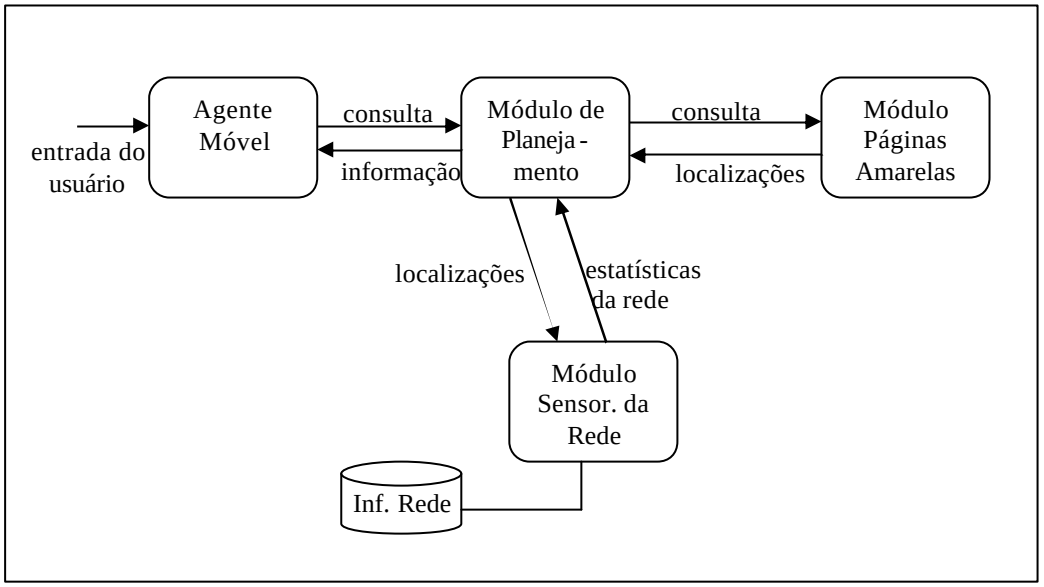


Figura 2.10: Arquitetura do Sistema de Planejamento (BREWINGTON, 1999).

Considerando que existem ***n+1*** nodos (s_i, onde $0 = i = n$). Cada nodo possui a probabilidade conhecida ($0 = p_i = 1$) de ser capaz de completar a tarefa do agente com sucesso, e um tempo $t_i>0$, solicitado para o agente obter a tarefa no nodo s_i indiferente de ter obtido sucesso. O tempo de viagem ou a latência para o agente mover-se entre os nodos são conhecidos por $l_{ij}=0$ para moverem-se do nodo **i** para o nodo **j**. Quando uma tarefa do agente for completada com sucesso em algum nodo, o agente deve retornar ao nodo *home* (nodo 0). Para o nodo 0, $p_0 = t_0 = 0$. O problema do agente viajante (*The Traveling Agent Problem*) é minimizar o tempo gasto para realizar uma tarefa com sucesso.

A solução para o problema do agente viajante consiste especificadamente da ordem na qual deve-se visitar os nodos, isto é, uma permutação $<i_1, i_2, ..., i_n>$ de **1** até **n**. A permutação será chamada de ***tour*** mantendo a nomenclatura tradicional para tais problemas.

O tempo gasto para completar a tarefa ou visitar todos os nodos em caso de falha, para uma *tour* $T=<i_1,i_2,...i_n>$ é (MOIZUMI, 1998a):

$$C_t = l_{0\,i1} + t_{i1} + p_{i1} l_{i1\,0} + ?_{k=2} \{(\mathbf{p}_{j=1} (1 - p_{ij}))\} (l_{ik-1\,ik} + t_{ik} + p_{ik}\, l_{ik\,0}) + \mathbf{p}_{j=1} (1-p_j)\, l_{n0} \qquad \textbf{[1]}$$

O primeiro nodo s_1, sempre visitado, requer que o tempo de viagem l_{0i1} seja alcançado. Na chegada, o tempo t_{i1} deve ser gasto independente de ter-se obtido sucesso. Com a probabilidade p_{i1} a tarefa é completada com sucesso e o agente pode retornar ao nodo 0 (*home*) com o custo l_{i10}. Entretanto, com a probabilidade $(1-p_{i1})$ houve falha e o agente prossegue para o nodo i_2. A contribuição para o tempo gasto por mover-se do nodo i_1 para o nodo i_2 é $(1-p_{i1})$ $(l_{i1\,i2} + t_{i2} + p_{i2}\, l_{i2\,0})$. Aqui o fator $(1- p_{i1})$ é a probabilidade de falha no nodo i_1. Similarmente, a contribuição para o tempo gasto para mover-se do nodo i_2 para i_3 é: $(1-p_{i1})$ $(1-p_{i2})$ $(l_{i2\,i3} + t_{i3} + p_{i3}\, l_{i3\,0})$, onde o termo $(1-p_{i1})$ $(1-p_{i2})$ é a probabilidade de falha em ambos os nodos i_1 e i_2. Em geral a contribuição para o tempo gasto atribuível ao nodo i_k é: (probabilidade de falha nos primeiros k-1 nodos) multiplicada pelo (tempo gasto para obter sucesso no nodo i_k). Todas estas contribuições formam a equação **[1]**, o último termo na equação **[1]** é executado quando a falha ocorre em todos os nodos visitados e o agente deve retornar ao nodo origem.

2.4.2. Planejamento de Agentes Móveis para Recuperação de Informação Distribuída

O número de agentes e o tempo de execução são dois significantes fatores de desempenho no planejamento de agentes móveis MAP (*Mobile Agent Planning*) (BAEK, 2001). Embora o planejamento de agentes móveis seja similar ao famoso problema do caixeiro viajante TSP (*Traveling Salesman Problem*) (GAREY, 1979), no TSP trata-se a definição do itinerário total ótimo com um dado número de agentes enquanto no MAP procura-se minimizar o tempo de execução de tarefas de recuperação de informação.

O objetivo do MAP é decidir: o número mínimo de agentes móveis necessários, a seqüência de nodos para cada agente móvel visitar, e o tempo de execução mínimo para uma tarefa.

As estatísticas da rede são conhecidas através de um serviço de monitoramento da rede. Este serviço capacita os agentes móveis para adquirir informação sobre a melhor rota para a tarefa.

Baseado no problema do agente viajante (MOIZUMI, 1998) descrito na seção anterior, o problema do planejamento de agentes móveis pode ser descrito desta forma: há ***n+1*** nodos, onde ***H*** é o nodo origem e os demais nodos são nodos indicados; cada nodo tem um tempo de computação necessário para o agente móvel executar a tarefa no nodo $\boldsymbol{h_i}$;

a latência para o agente mover-se entre os nodos ***hi*** e ***hj*** é também conhecida; para o nodo origem ***H***, o tempo de computação é igual a zero.

O problema do agente móvel é minimizar o tempo de execução (***T***) e o número de agentes móveis ***r*** para realizar com sucesso a tarefa.

O tempo de execução pode ser definido pelo maior tempo de roteamento (*TourT*) entre todos os nodos visitados pelo agente. Isso compreende latência de rede do nodo origem (***H***) ao nodo (***i***) mais o tempo de execução nele.

A solução do problema MAP é encontrar uma seqüência de agentes para visitar os nodos, esta solução consiste de um tempo de execução total mínimo e o número de agentes necessários.

Uma situação que ilustra o problema é apresentada na Figura 2.11, onde a configuração da rede foi particionada em duas partes. O peso de uma aresta representa a latência esperada para a conexão entre um par de nodos. O tempo de computação nos nodos são: ***h1***=30, ***h2***=5, ***h3***=5. O tempo de execução nunca é menor que 50 ms, onde o nodo ***h1*** tem o custo de roteamento máximo da configuração. O tempo de execução no nodo ***h1*** é 50 ms (=10+30+10), onde o custo dos nodos ***h2*** e ***h3*** são 21ms (=8+5+8) e 25ms (=10+5+10) respectivamente. Neste exemplo o número mínimo de agentes é dois. O primeiro agente cobre o nodo ***h1*** (Figura 2.11(b)), e o segundo agente cobre os nodos ***h2*** e ***h3*** (Figura 2.11(c)). O tempo de execução total é 88ms (=50+38). Note que a seqüência do itinerário do segundo agente é (***H***,***h3***,***h1***,***h2***,***H***), devido ao uso de um caminho secundário ao invés da conexão direta entre ***h2*** e ***h3*** em função da latência esperada entre eles.

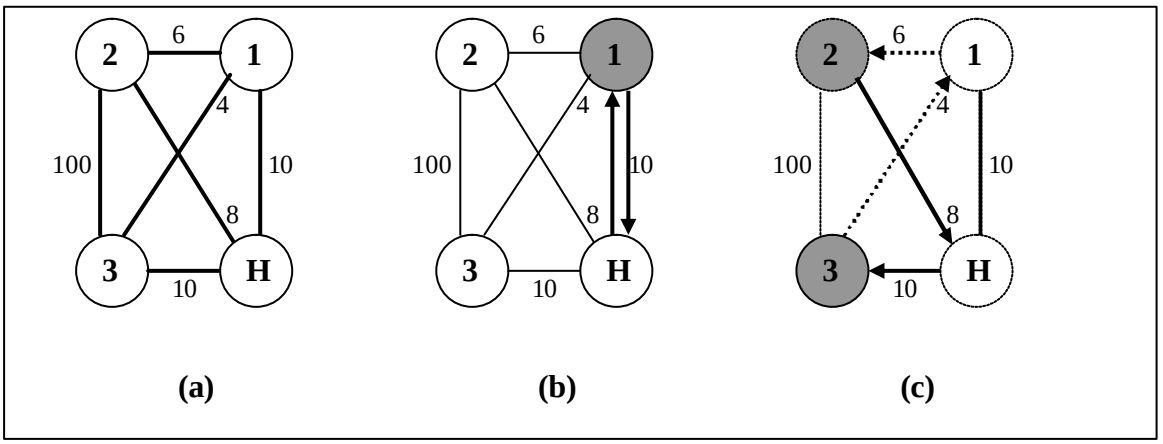


Figura 2.11: Um exemplo de configuração da rede.

Em (BAEK, 2001) foram propostos dois algoritmos heurísticos para encontrar o número mínimo de agentes móveis necessários para recuperar informações em um ambiente distribuído enquanto a latência mínima da rede é mantida.

O uso de poucos agentes gera menor tráfego na rede resultando em um consumo menor de largura de banda. Um itinerário de agentes mal escalonado pode causar maiores tempos de execução como resultado do custo de roteamento. Com o objetivo de reduzir o *overhead* na rede, os agentes devem ser escalonados antes de serem enviados.

Os algoritmos BYKY1 e BYKY2 (BAEK, 2001) podem ser usados em sistemas de recuperação de informação distribuídos para encontrar os fatores mencionados anteriormente (tempo de execução mais rápidos e menor tráfego na rede causado pelos agentes). Embora não seja o objetivo principal, o custo de roteamento total é importante nos problemas MAP, uma vez que ele decide o tempo de viagem de um agente alocado na rede local. Portanto, torna-se desejável gerenciar o custo do itinerário. Por esta razão o algoritmo 2OPT (um algoritmo simples TSP) é empregado nos algoritmos propostos. Ele otimiza cada caminho local do agente. Utilizando os algoritmos BYKY1 e BIKY2 o sistema pode manter o melhor desempenho enquanto alcança o menor custo de itinerário com o menor número de agentes possível.

```
Fase 1 //ordenando os nodos
    ordena os nodos em função do tempo de viagem(TourT) em ordem decrescente
    obtem a seqüência resultante (ha1,ha2,ha3,...,han)
    T = TourT(ha1)        // T= tempo de execução inicial
Fase 2  //planejamento dos agentes
  para j=1 até n  {
     Tour(j) = phi
     para k=j até n
     {
        se hak não é processado  e  TourtT (Union(Aj,(hak))  <  T
             Union(Aj, (hak))
             Marque hak como processado
     }
     se Aj = 0
        termina
  }
Fase 3  //otimização do caminho local do agente
  para cada agente Ai
      2OPT(Ai)
```

Figura 2.12: O Algoritmo de Planejamento BYKY1 (BAEK, 2001).

O BYKY2 é mais dinâmico que o BYKY1. O BYKY1 tenta encontrar a próxima partição possível calculando as latências sempre a partir do nodo *home*, enquanto que o BYKY2 procura o próximo nodo a partir do nodo atual, onde o agente reside. As Figuras 2.12 e 2.13 apresentam a descrição dos algoritmos BYKY1 e BYKY2 respectivamente.

```
Fase 1 //ordenando os nodos
   mesma da fase 1 do algoritmo BYKY1
Fase 2  //planejamento dos agentes
para j=1 até n
{
   Tour(Aj) = 0
   seleciona h_ak não processado, onde k é mínimo
   Union(Tourt(Aj),(h_ak))
   marque h_ak como processado
   While (true)
   {
     ordene os nodos em função de {Ls(h_ak,h_a1) | 1 <= l <= n e h_a1 é não processado
     dado que a seqüência (h_b1,h_b2,h_b3,...,h_bn)seja a seqüência resultante
     K_anterior = k;
     para x=1 até m
     {
       se TourT(Union(Aj, hbx)<= T {
           Union(Aj,h_bx)
           Marca h_bx como processado
           Encontra k, onde ak = bx
           break do loop para  }
     }
     se k_anterior = k
         break loop while
   }
   se Tour(Aj) = 0
         termina
}
Fase 3  //otimização do caminho local do agente
  mesma da fase 3 do algoritmo BYKY1
```

Figura 2.13: O Algoritmo de Planejamento BYKY2 (BAEK, 2001).

2.4.3. Um VMAS (*Visual Mobile Agent System*) com Escalonamento de Itinerário

Outro trabalho que faz referência ao problema do planejamento do itinerário de agentes móveis objetivando o menor custo de migração possível é apresentado por CHANG e CHANG (2000). Foi desenvolvido um sistema visual de agentes móveis VMAS (*Visual Mobile Agent System*) e um método de escalonamento de itinerário híbrido com o objetivo de encontrar um caminho para migração de agentes com baixo custo.

O VMAS é um sistema, baseado em Java, o qual consiste de três principais componentes: ***Agent Manager***, ***Meta-Service Server*** e ***MABuilder***. O *Agent Manager* é encarregado pelo envio, recebimento, execução e comunicação entre os agentes. O *Meta-Service Server* oferece um serviço para o escalonamento do itinerário de agentes móveis (similar às páginas amarelas). O *MABuilder* fornece um ambiente de construção de agentes móveis. O *MABuilder* permite que as ações do agente móvel sejam definidas e seu itinerário seja manualmente escalonado de forma que os custos em tempo de projeto sejam reduzidos (CHANG, 2000).

No escalonamento manual do itinerário, a escala do itinerário dos agentes móveis é feita sem qualquer consideração sobre custos correspondentes a viagem total do agente móvel. Entretanto, com um conhecimento prévio e em tempo real do ambiente da rede, escalonar um caminho de migração ótimo é possível. Visando alcançar um itinerário ótimo, (CHANG, 2000) desenvolveu um esquema automático de escalonamento de itinerário.

A fim de desenvolver um método de escalonamento flexível, os nodos no caminho de migração dos agentes móveis foram classificados em: **nodos essenciais**, são os nodos que o agente móvel deve passar durante a viagem; e **nodos não-essenciais**, existe apenas a possibilidade deste tipo de nodo ser visitado pelo agente móvel durante sua viagem.

Para fornecer uma função de custo efetiva, todos os fatores de custo possíveis são classificados de acordo com a independência no tempo e a ordem de viagem entre os nodos, podem ser classificados em quatro tipos:

- TI-OI (Independente de tempo, independente de ordem): por exemplo, *login* no nodo;
- TI-OD (Independente de tempo, dependente de ordem): por exemplo, conexão entre dois nodos;
- TD-OI (Dependente de tempo, independente de ordem): por exemplo, número de agentes móveis residentes no nodo;
- TD-OD (Dependente de tempo, dependente de ordem): por exemplo, velocidade de transmissão entre os nodos.

Baseado nas definições e classificações anteriormente descritas, o método de escalonamento híbrido proposto é desenvolvido em dois passos:

- **Geração do caminho inicial**: Este passo acontece antes do agente móvel ser despachado. O objetivo deste passo é calcular um caminho inicial de menor custo de acordo com as funções custo atribuídas. O algoritmo para encontrar o caminho inicial ótimo é similar ao TSP. Devido aos agentes móveis terem que visitar os nodos essenciais, quando otimizado o caminho para os nodos essenciais, não é necessário considerar fatores não relacionados à ordem dos nodos. A função custo é simplificada;

- **Modificação incremental**: Este passo acontece depois do agente móvel ser transmitido. O objetivo deste passo é negociar como o itinerário do agente móvel (isto é, o caminho inicial) é modificado durante a viagem de acordo com o estado atual da rede. Por exemplo, assumido o caminho inicial 1.2.3.4.5.6.7.8, quando o agente móvel está deixando o nodo 3, se ele detectar que o estado do nodo 4 está diferente do estágio de geração do caminho inicial, então o caminho pode ser modificado. Assumido que o nodo 7 é selecionado como próximo candidato a nodo visitado, é tomada a decisão se a mudança no caminho inicial é adequada ou não. A decisão é feita comparando os custos do caminho antigo (3,4,5,6,7,8) e o novo caminho (3,7,5,6,4,8). Se o custo do novo caminho for menor que o original, o novo caminho substituirá o antigo. Caso contrário, o caminho continuará o mesmo.

2.4.4. Análise de Itinerário Ótimo para Agentes Móveis em Redes Sensores sem Fio Ad Hoc

Em (QI 2001) é descrito o uso de agentes móveis para fusão de dados em redes sensores sem fio, onde é apresentado um método para desenvolver um itinerário ótimo para os agentes móveis cumprirem a integração da tarefa enquanto consomem uma quantidade mínima de recursos, incluindo tempo e processamento.

Em tradicionais redes de sensores distribuídas (DSN *Distributed Sensor Networks)* os dados são coletados por sensores individuais, e então transmitidos para um elemento processador de alto nível no qual executa a fusão de dados. Durante este processo uma grande quantidade de dados é movida pela rede. QI e WANG (QI 2001) adotaram um novo paradigma, agentes móveis, e o referenciaram como redes sensores sem fio baseadas em agentes móveis MAWSN (*Mobile Agent-Based Wireless Sensor Network*).

No MAWSN, os dados permanecem no nodo local, enquanto o processo de integração (código) é movido para os nodos dos dados. Por transmitir a "máquina de computação" ao invés dos dados, o MAWSN apresenta alguns benefícios, como: a redução da largura de banda solicitada na rede (isto é importante para aplicações de tempo real e onde a comunicação acontece através de conexões sem fio com baixa largura de banda), e estabilidade (agentes móveis são enviados e podem retornar os resultados quando a conexão for restabelecida).

Uma rede de sensores distribuídas consiste de um conjunto de nodos sensores, um conjunto de elementos processadores (PEs) e uma rede de comunicação interconectando vários PEs. Um ou mais sensores podem estar associados com cada PE. Um sensor pode se comunicar com mais de um PE. Os dados são transferidos dos sensores para seus PEs associados onde a integração dos dados acontece. Um grupo de nodos sensores vizinhos que são comandados por um único PE forma um *cluster*.

O benefício introduzido pelo uso de agentes móveis no contexto de DSN dependerá do planejamento para o itinerário do agente, isto é, dependerá da escolha de um itinerário que consuma uma quantidade mínima de recurso (tempo e processamento) a fim de finalizar a tarefa da fusão.

O itinerário pode ser definido estaticamente ou dinamicamente, ou seja, ele pode ser definido antes da partida do agente ou enquanto o agente estiver migrando. O planejamento de itinerário dinâmico é mais flexível, e pode se adaptar às mudanças ambientais em tempo real. Entretanto, uma vez que o itinerário é calculado em tempo de execução, ele consome mais tempo de computação e exige um maior processamento no sensor local. Eficiência de computação, eficiência de processamento e flexibilidade são três características que não podem ser satisfeitas ao mesmo tempo (QI, 2001).

Dois algoritmos *ad hoc* podem ser usados para calcular o itinerário: *Local Closest First* (LCF) e *Global Closest First* (GCF). Os atributos do itinerário no agente móvel $\boldsymbol{AM_{i,j}}$ (onde $\boldsymbol{i}$ indica o número de identificação do despachante e $\boldsymbol{j}$ é o número serial do agente associado a seu despachante) são: $\boldsymbol{C_{i,j}}$ é o centro coordenativo de uma sub-área para ser percorrida pelo agente móvel; $\boldsymbol{r_{i,j}}$ é o raio desta sub-área; e $\boldsymbol{L_{i,j}}$ é a lista de destinos que o agente móvel precisa visitar em sua viagem. Assumindo que ambos os algoritmos iniciam no nodo sensor mais próximo ao centro $\boldsymbol{C_{i,j}}$, o algoritmo LCF busca o próximo nodo com a

menor distância do nodo atual, enquanto o algoritmo GCF busca para o próximo nodo aquele que estiver mais próximo do centro $C_{i,j}$.

Quando a agente móvel migra por uma rede de sensores, vai acumulando informação de leitura. Se a precisão do resultado alcançou os requisitos de certa tarefa, o agente pode retornar diretamente para o PE sem completar a viagem. Devido a esta complexidade, é sugerido que o itinerário seja gerado estaticamente.

Sendo $L_{i,j}$ a lista de nodos sensores que o agente deveria visitar, a medida que a migração acontece, os parâmetros de estimação $[a_k,b_k]$ devem ir reduzindo, enquanto a precisão $d_{i,j}$ se aproxima. Se a precisão foi alcançada, então o agente móvel não precisa visitar os nodos sensores restantes, ele pode retornar ao PE, assim economizando tempo de migração e largura de banda. Desta forma, seu desempenho é superior aos algoritmos descritos anteriormente (LCF e GCF).

O itinerário (lista de nodos) ótimo $L_{i,j}$ é formado tal que o custo do tempo de computação e o consumo de processamento com relação a cada nodo seja o mínimo possível. Uma função objetivo H indica o compromisso entre o H_t (tempo consumido) e o H_p (processamento consumido relativo):

$$H(L_{i,j}) = a\ H_t + (1 - a)\ H_p \qquad [2]$$

onde: $a = \hat{A}^+ < 1$

Obtido um itinerário ótimo, o algoritmo (Figura 2.14) descreve como ele é usado para subjugar o requisito de precisão. Este problema de otimização pode ser solucionado através de um algoritmo genético.

```
Dados: L_{i,j} (itinerário ótimo);
       δ_{i,j} (precisão requisitada);
       [a_k,b_k] (leitura dos sensores de um param. espec.)onde k=1,...,n;
       W_old (largura da leitura anterior);
W_old = ∞;  K = 1;
While k ≤ n do
   While L_{i,j}[k] não está ativo  do
       K = k + 1;
   end
   migrar para L_{i,j}[k];
   obter a leitura do sensor [a_k,b_k];
   if |(b_k-a_k)- w_old | < δ_{i,j}  then
          w_old
       break;
   else
       w_old = b_k – a_k;
       k = k + 1;
   end
end
retornar ao PE;
```

Figura 2.14: Planejamento do itinerário ótimo pela precisão requisitada.

2.4.5. Cálculo de Rotas de Agentes Móveis para Fusão de Dados em Redes Sensores Distribuídas

A ordem dos nodos visitados ao longo da rota possui um significante impacto na qualidade e no custo da fusão de dados, o que por sua vez impacta no objetivo principal de redes sensores, como classificação ou rastreamento do objetivo (alvo). WU et al. (2004) apresentam um modelo analítico para uma rede de sensores distribuída e formulam o problema de cálculo da rota em termos da maximização da função objetivo, o qual é diretamente proporcional à força do sinal recebido e inversamente proporcional à energia consumida e aos caminhos perdidos.

As motivações para usar agentes móveis em DSN tem sido extensivamente estudadas (QI, 2001a). A ordem e o número de nodos na rota do agente móvel determina o consumo de energia, os caminhos perdidos (nodos visitados desnecessariamente, por exemplo, um nodo com um sensor inativo) e a detecção da precisão, por isso possui um significante impacto no desempenho geral de uma MADSN (*Mobile Agent-based Distributed Sensor Networks*). Baixo consumo de energia, poucos caminhos perdidos e alta detecção de precisão são sempre os principais objetivos de uma DSN. O cálculo da rota adequada, na prática, envolve a relação entre o custo (consumo de energia e caminhos perdidos) e o benefício (detecção de precisão). A visita a mais sensores melhora a qualidade da fusão de dados, mas também aumenta os *overheads* de comunicação e computação. O objetivo geral do roteamento é maximizar a soma do sinal de energia recebido nos nodos visitados enquanto minimiza o processamento necessário para comunicação e caminhos perdidos.

O agente móvel migra entre os nodos sensores da rede, integra-se aos dados locais com uma resolução desejada seqüencialmente, e carrega o resultado final para o PE originário.

A Figura 2.15 apresenta um MADSN com uma configuração simples. Neste exemplo a rede de sensores possui um PE (S_0) e n nodos folhas, sendo $N = 10$ e conjunto de nodos sensores $S = S_1, S_2, S_3, ..., S_{10}$. Os nodos sensores estão espacialmente distribuídos em uma área de supervisão, cada um é responsável por coletar as situações atuais do ambiente. O *link* de comunicação *wireless* possui distância física de $d_{i,j}$ entre os nodos sensores Si e Sj. Alguns sensores podem ficar inativos temporariamente devido a falhas intermitentes, como por exemplo, o sensor S_9 (Figura 2.15).

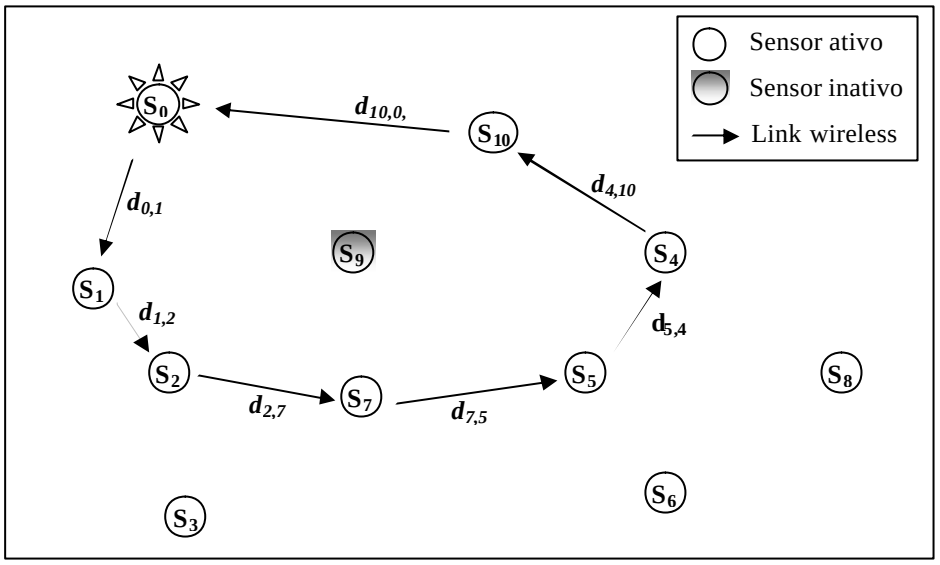


Figura 2.15: Exemplo de um simples MADSN com um PE e 10 nodos folhas.

Algoritmos de roteamento podem ser classificados como dinâmicos ou estáticos de acordo com o local onde a tomada de decisão acontece. Em (WU, 2004) é proposto um método adaptativo para implementar uma estratégia de roteamento semi-dinâmica baseada em um algoritmo genético de dois níveis (Figura 2.16).

Uma função de aptidão é utilizada para encontrar a precisão desejada enquanto minimiza o consumo de energia e os caminhos perdidos. Conseqüentemente, um algoritmo genético precisa garantir a relevância das informações do sistema para desenvolver os cálculos da rota. Já que o agente móvel sempre inicia sua viagem de um nodo PE, o qual usualmente é equipado com maior poder de processamento que os nodos regulares, ele é responsável por calcular a rota. O agente móvel simplesmente segue a rota calculada pelo algoritmo genético de acordo com a função de aptidão.

O código com o cálculo da rota reside exclusivamente no nodo PE a fim de manter o código do agente móvel o mais compacto possível. O agente apenas carrega a rota pré-calculada que determina a ordem dos nodos sensores a serem visitados. Entretanto, se o sistema da rede é modificado (por exemplo, alguns nodos são desativados ou desligados, não existe energia suficiente para transmitir o sinal através do link previamente designado), isto pode fazer com que o cálculo prévio da rota torne-se inválido. Nestes casos, o código de roteamento é recalculado no nodo PE e a nova rota é transmitida ao agente móvel. A Figura 2.16 apresenta o diagrama de atividades que ilustra o método de roteamento semi-dinâmico descrito.

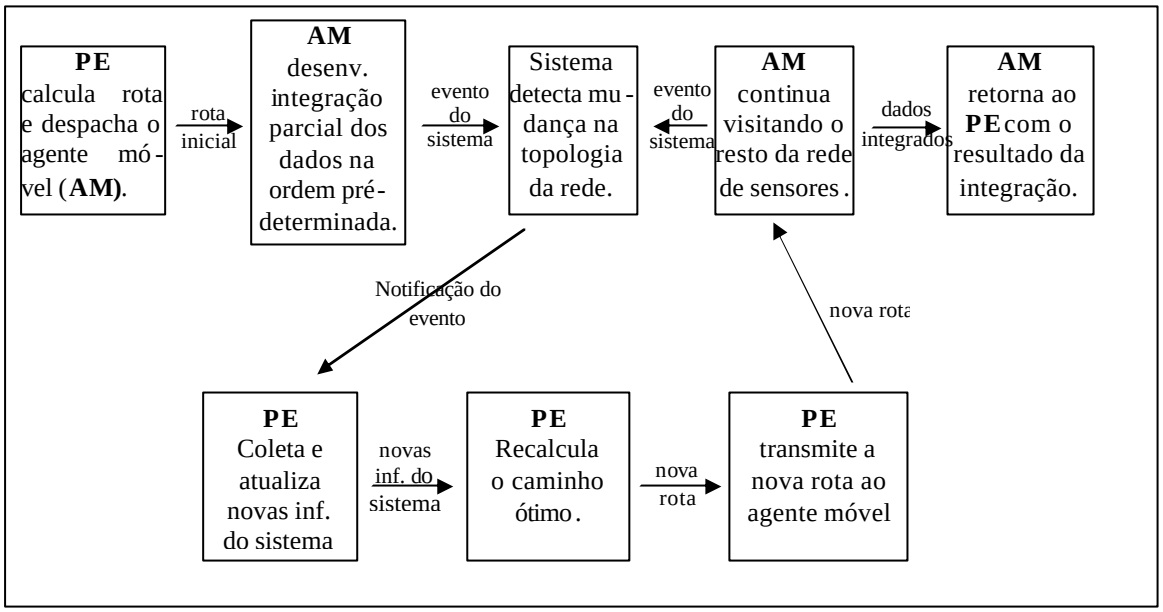


Figura 2.16: Diagrama de atividade do método de roteamento semi-dinâmico adaptativo (WU, 2004).

Foram executadas simulações para comparar o resultado do algoritmo genético (AG) proposto e as heurísticas LCF e GCF já existentes. Aspectos analisados foram: complexidade computacional, impacto da infra-estrutura da rede e a partida do nodo. Na maioria dos casos, o LCF foi capaz de entregar uma rota aceitável, quando comparado ao AG. O GCF apresentou rotas muito desordenadas, o que resultou em muitos caminhos perdidos. O algoritmo AG gerou rotas com o menor número de saltos (exceto quando o número de nodos sensores é muito pequeno) freqüentemente encontrando rotas econômicas, com baixo consumo de energia, assim apresentando um melhor desempenho quando comparado as outras duas heurísticas.

2.4.6. Algoritmo de Roteamento de Agentes Móveis para Recuperação de Consultas usando Algoritmo Genético

Em [SELAMAT 2004] é proposto um método para minimizar o tempo de roteamento da rede que os agentes móveis levam para coletar informações de diferentes nodos usando algoritmo genético. Os agentes móveis repetem suas viagens sobre rotas curtas e evitam as longas.

SELAMAT (2004) propõe um sistemas de busca de agentes móveis MaSS (*Mobile Agent Search System*) para trazer resultados de máquinas de busca onde os agentes

móveis utilizam algoritmo genético para selecionar as melhores rotas, tentando minimizar o custo enquanto mantém um atraso aceitável na definição do caminho.

A arquitetura do MaSS consiste de um cliente MaSS, um servidor MaSS e alguns servidores de busca locais. O cliente MaSS consiste de um agente (***W***) e uma interface de usuário simples. O usuário solicita uma palavra chave para ser pesquisada no navegador HTML. O agente cliente MaSS ***W*** enviará o pedido ao agente servidor MaSS (***X***) a fim de obter os resultados de sua consulta. O servidor MaSS consiste de um agente servidor (***X***) e um agente de busca (***Y***). O agente servidor MaSS ***X*** é um agente estacionário. No momento em que chega uma solicitação de ***W***, o agente servidor ***X*** delegará as tarefas de busca para o agente de busca ***Y***. Depois de receber os resultados da pesquisa, o agente servidor irá classificá-los. Os resultados são armazenados em uma base de dados no servidor MaSS. Quando o agente ***Y*** recebe o pedido vindo de ***X,*** ele inicia sua viagem percorrendo o conjunto de servidores de busca locais. Ele se comunica com os agentes locais em cada nodo visitado que fornecem os resultados ao agente móvel de busca ***Y***. Uma vez que as tarefas tenham sido completadas, o agente de busca ***Y*** retornará ao nodo origem e passará os resultados ao agente servidor MaSS (***X***).

Servidores de busca consistem de um servidor de base de dados local, por exemplo Yahoo ou Alta Vista. A idéia básica do servidor de busca local é armazenar os resultados das consultas localmente ao invés de buscá-los na WWW (*World Wide Web*). Isto reduz o tempo de busca e recuperação das consultas feitas pelo agente de busca MaSS ***Y***. Os servidores de busca enviam seus agentes locais para viajar pela WWW e coletar os resultados de máquinas de busca e armazenam os resultados em suas base de dados nos nodos.

Com o objetivo de reduzir o tempo de consulta, o agente de busca MaSS ***Y*** utiliza algoritmos genéticos para selecionar sua rota. A otimização da rota é dada por

$$\textit{Min Qrt(route) s.t.Delay(route)} \pounds \textit{MaxDelay} \qquad \textbf{[3]}$$

onde *Qrt(route)* é o tempo de consulta levado para recuperar um resultado, *route* é o caminho que foi utilizado pelo agente ***Y*** para enviar e recuperar os resultados dos servidores de busca locais. *Delay(route)* é o atraso devido a gargalos na rede. *MaxDelay* é o atraso máximo aplicado no caminho utilizado pelo agente ***Y***.

O método *crossover point* é utilizado pelo algoritmo genético. Primeiramente, um ponto no vetor é determinado (o número de elementos do vetor representa o número de nodos da rede, onde cada número indica o nodo visitado e 0 indica que o nodo não foi visitado pelo agente) e é trocada a parte posterior ao *cross point* para gerar duas novas soluções. O fato do número de nodos das rotas não ser fixo dificulta a determinação do *crossover point*. Por este motivo, um novo método *crossover point* dinâmico é usado: **[(A+B)/4]** onde ***A*** e ***B*** são o número de nodos que as duas rotas visitaram. Por exemplo, as duas rotas antes das operações de *crossover* eram: [1 2 3 4 5 0 0 0 0 0] e [6 7 8 0 0 0 0 0 0 0]. Utilizando o método proposto por [Selamat, 2004] somente parte da população de nodos executará a operação de crossover, resultando em [1 2 8 0 0 0 0 0 0 0] e [6 7 3 4 5 0 0 0 0 0].

Embora algoritmos genéticos possam ser utilizados para busca em um amplo espaço de soluções e obter uma solução ótima, eles podem levar muito tempo para chegar a esta solução ótima. Nestes casos, o algoritmo genético pode somente encontrar algumas soluções próximas a ótima. No método proposto em (SELAMAT, 2004), uma combinação de condições foi utilizada para determinar quando parar de calcular novas soluções.

Depois de um número mínimo de ***K*** execuções, se o algoritmo encontrou uma solução possível e não houve melhora por um período de tempo específico, então o algoritmo deve parar. No caso do algoritmo proposto, a "melhora" é apresentada pela taxa de custo da melhor solução de certa execução. Esta taxa é avaliada por:

$$\textbf{\textit{ChangeRate (K)}} = \frac{\textbf{\textit{costs (k-1)}}}{\textbf{\textit{costs (k)}}} * \textbf{\textit{100}} \qquad \textbf{[4]}$$

onde *ChangeRate(k)* é a média da taxa de troca no $\boldsymbol{k}^{\textit{ésimo}}$ passo. Uma vez que esse valor é maior que determinado limiar (por exemplo, *ChangeRate(k)* ≤ 90%), então os cálculos do algoritmo genético devem parar.

Desta forma, o algoritmo de roteamento de agentes móveis tenta minimizar o custo de recuperação de uma consulta enquanto mantém um atraso aceitável. Simulações mostraram que o número de execuções solicitadas para alcançar um bom resultado foram reduzidas significativamente pela escolha de rotas mais curtas e pela reutilização de soluções conhecidas como cromossomos iniciais para a nova busca.

2.4.7. Planejamento de Itinerário Autônomo para Agentes Móveis

No campo de computação distribuída inteligente em redes de computadores dinâmicas, uma questão que atrai interessante é o uso de agentes móveis e sistemas de agentes móveis, especialmente com relação ao problema do itinerário e da adaptação de agentes móveis.

Em um cenário típico, um agente móvel visita mais de um nodo na rede para cumprir sua tarefa. Por isso, ele precisa de um plano de viagem quando inicia sua missão. Em (ERFURTH, 2003) um plano pré-determinado é fornecido pelo programador (*owner*), no entanto, devido ao tamanho e ao comportamento dinâmico das redes *ad-hoc* - que mudam mais rápido que a percepção do agente (*owner*) da rede – os agentes móveis devem desenvolver habilidade para construir e adaptar seu próprio plano de viagem, o qual é chamado de itinerário. Durante a sua viagem o agente visita pontos de serviço de agências remotas, se comunica e coopera com outros agentes.

A estrutura proposta por (ERFURTH, 2003) consiste de três módulos: o módulo de mapeamento (*Map Module*), o módulo de planejamento de rota (*Route Planner Module*) e o módulo de planejamento de migração (*Migration Planner Module*).

O módulo de mapeamento é responsável pela coleta de dados para criar uma base de informação no qual é chamado de mapa. Cada agência gerencia seu mapa local. O módulo de mapeamento contém sensores monitorando a rede que coletam os dados. Existem dois tipos de dados: propriedades das agências, como serviço de lista, protocolos suportados, etc.; e características do enlace, como largura de banda, latência, etc. O mapa é dividido em mapa local e remoto. Seguindo intervalos regulares de tempo, os sensores atualizam as informações dentro de sua área local. Como conseqüência, cada agência possui um mapa com informações atualizadas da área local e possuem um esboço das áreas remotas (possivelmente desatualizadas).

O módulo de planejamento de rota é capaz de calcular o caminho mais curto. Este módulo usa, especialmente, os dados sobre as topologias de conexões e qualidade dos enlaces. O processo de planejamento de rota é basicamente o Problema do Caixeiro Viajante (GAREY, 1979), o qual é conhecidamente como um problema do tipo NP completo. Em geral, o cálculo do caminho mais curto é aplicado somente se os pesos das arestas diferem claramente. Portanto, se existe uma rede homogênea não existe razão para recalcular um novo plano de rota.

O módulo de planejamento de migração foi desenvolvido para recalcular o itinerário. O objetivo é transmitir apenas as partes de código necessárias para a execução da tarefa. Desta forma, a carga na rede causada pelo agente e o tempo de transmissão são reduzidos. A meta do módulo de migração é decidir o que é mais eficiente: transmitir todas as partes do código juntamente com o estado do agente e os dados, ou transmitir apenas as partes/fragmentos de código necessárias, ou até não enviar nenhuma parte.

ERFURTH e ROSSAK (2003) propuseram uma infra-estrutura que torna o agente móvel mais autônomo, onde o usuário preocupa-se apenas em 'o quê' o agente deve fazer e não 'como' fazê-lo.

2.4.8. Agentes Móveis Autônomos com Comportamento de Migração Emergente

Em uma tentativa de se solucionar o problema do aumento do tráfego produzido na rede pela grande comunicação entre os agentes em sistemas distribuídos de larga escala, SCHLEGEL (2006) propôs utilizar agentes móveis para reduzir este *overhead* na rede, permitindo que o agente encontre o mesmo nodo antes de iniciar o processo de comunicação.

Uma comunicação remota entre dois agentes poderia então ser substituída por uma ou duas migrações, seguidas de uma comunicação local.

A tomada de decisão entre migrar o agente móvel ou optar pela comunicação remota deve acontecer em tempo de execução, baseando-se nos parâmetros do ambiente e em experiências passadas do agente.

Em (SCHLEGEL, 2006) é apresentado um método adaptativo onde cada agente prevê a carga na rede do próximo passo de comunicação e utiliza um modelo matemático simples para decidir entre as duas alternativas em tempo de execução. Esse método não considera apenas a carga da rede mas também a carga do servidor (os agentes prevêem dinamicamente o número de agentes que já migraram para um servidor de agentes específico).

No modelo proposto por (SCHLEGEL, 2006) cada agente possui uma lista de passos de comunicações $\boldsymbol{C = \{ c_i \mid i : 1..p\}}$ como parte de sua tarefa. Cada passo de comunicação $\boldsymbol{c_i = \{A_a, A_b, m_k, m_j\}}$ define que uma mensagem de solicitação $\boldsymbol{m_k}$ é enviada do agente origem $\boldsymbol{A_a}$ ao agente destino $\boldsymbol{A_b}$, o qual responde com uma mensagem de

resposta m_j. Cada mensagem m_k pode ser vista como uma seqüência arbitrária de bytes de tamanho $B_m(m_k)$. O custo da rede para um passo de comunicação remota é calculado por:

$$B_{RC}(c_i) = \begin{cases} 0 & \text{se } A_a \text{ e } A_b \text{ encontram-se no mesmo nodo} \\ Bm(mk) + Bm(mj) & \text{se } A_a \text{ e } A_b \text{ encontram-se em nodos distintos} \end{cases} \quad [5]$$

Antes de cada passo de comunicação, os agentes envolvidos nesta comunicação (agentes móveis) devem decidir se migram de sua localização atual para a localização destino antes de iniciar a comunicação. Se ambos os agentes migram ao mesmo servidor destino a carga da rede somente inclui o custo de transferência do código do agente de tamanho B_c e o estado do agente B_e (no qual é igual ao tamanho da mensagem de solicitação), porém não há custos para a comunicação local. Após as comunicações, ambos agentes migram de volta a seus nodos servidores origem (sem o código – porque o código já esta disponível no seu nodo servidor origem). O agente A_a carrega a informação de estado e os resultados de tamanho $(1-?)\ B_m(m_j)$ com $0 \pounds ? \pounds 1$. O custo da rede de comunicação do agente móvel é calculado por:

$$B_{MA}(Ci) = \begin{cases} 0 & \text{se } A_a \text{ e } A_b \text{ encontram-se no mesmo nodo} \\ Bc(Aa) + Bm(mk) + & \\ Bc(Ab) + (1-?)Bm(mj) & \text{se } A_a \text{ e } A_b \text{ encontram-se em nodos distintos} \end{cases} \quad [6]$$

se ambos os agentes são móveis. Se somente o agente enviante for móvel, a transmissão do código do agente A_b deve ser desconsiderada.

A formula **[6]** é independente da localização do agente destino porque apenas são consideradas as cargas da rede e não o tempo de transmissão neste estágio. As fórmulas **[5]** e **[6]** são utilizadas para decidir entre as duas alternativas em tempo de execução, contanto que todos os tamanhos de mensagem sejam conhecidos.

A medida que o agente for executando suas tarefas forma-se um histórico que o auxiliará na tomada de decisão. Baseado em um conjunto de regras que refletem algumas preferências do usuário, o agente decide autonomamente de forma a oferecer o maior benefício para o usuário.

2.4.9. Comentários Gerais sobre as Abordagens

A introdução de agentes móveis no campo dos sistemas distribuídos provou sua utilidade quando os agentes viajam pela rede em busca de informações solicitadas pelos usuários. Como visto em (RAMAMRITHAM, 2002, SHIN, 2001, SELAMAT, 2004) agentes móveis podem cooperar com outros agentes afim de completar suas tarefas atribuídas.

Muitos pesquisadores têm investigado a adaptação de agentes móveis para o roteamento da rede. Por exemplo, em (BREWINGTON et al., 1999) é apresentado um método de recuperação de informação no qual é feito um planejamento do itinerário que o agente móvel percorrerá para obter a informação, utilizando probabilidade. Este planejamento proporciona a utilização de menos recursos de rede enviando o menor número de agentes móveis possíveis (número de agentes inferior ao número de nodos a serem visitados).

Já em (BAEK, 2001), além da preocupação com o número de agentes, considera-se também o tempo de execução da tarefa como significante fator de desempenho no planejamento de agentes móveis. A solução do problema MAP é encontrar uma seqüência de agentes para visitar os nodos. Esta solução consiste de um tempo de execução total mínimo e o número de agentes necessário. Em (BAEK, 2001), os agentes são escalonados antes de serem enviados com o objetivo de reduzir o *overhead* na rede. Um itinerário de agente mal escalonado pode causar maiores tempos de execução como resultado do custo de roteamento.

Estendendo as precauções descritas em (BREWINGTON, 1999) e (BAEK, 2001), em (CHANG, 2000) foi apresentado um sistema visual de agentes móveis e um método de escalonamento de itinerário flexível onde os nodos no caminho de migração dos agentes móveis foram classificados em nodos essenciais e nodos não-essenciais. Diferentemente de quando o itinerário é definido estaticamente, o método de escalonamento híbrido proposto é desenvolvido em dois passos: geração do caminho inicial, antes da partida do agente, e a modificação incremental que acontece após o agente móvel iniciar sua missão.

Na comunidade de redes sensores, algoritmos baseados nas heurísticas LCF e GCF são utilizados para calcular as rotas de agente móveis em MADSNs (QI, 2001a). Seus resultados são satisfatórios para pequenas DSN. Entretanto, seu desempenho piora quando o tamanho da rede cresce e a distribuição dos sensores torna-se mais complexa. Em alguns

casos, as rotas traçadas por estas heurísticas podem gerar soluções insatisfatórias uma vez que elas apenas consideram a distância espacial entre os nodos sensores. Nestas aplicações, muitos outros fatores, tais como detecção do nível do sinal e consumo de energia do enlace, são importantes considerações, sendo essenciais para o êxito da execução.

A ordem e o número de nodos na rota do agente móvel determina o consumo de energia, o caminho perdido e a detecção da precisão, por isso possui um significante impacto no desempenho geral de uma MADSN. Na tentativa em se obter o melhor caminho possível para uma DSN, foram propostas novas heurísticas. Em (WU, 2004) foi formulado o problema do cálculo da rota para o agente móvel em termos da maximização do sinal recebido com baixo consumo de energia e baixo índice de caminhos perdidos. Para resolver este problema foi implementada uma estratégia de roteamento semi-dinâmica baseada em um algoritmo genético de dois níveis.

Em (QI, 2001) foi apresentado um método para desenvolver um itinerário ótimo para agentes móveis para preencher a tarefa de integração (fusão de dados) enquanto consome a menor quantidade de recursos, incluindo tempo e processamento. Neste método, quando o agente móvel alcança a precisão estimada, ele pode retornar ao PE origem sem terminar seu itinerário, dessa forma economizando tempo de processamento e energia consumida. Em (QI, 2001) e (WU, 2004) foram apresentadas diferentes métricas com os mesmos objetivos.

Em (SELAMAT, 2004), assim como em (WU, 2004), foram apresentados o uso de algoritmos genéticos no roteamento de agentes móveis. SELAMAT (2004) propõe um sistema de busca de agentes móveis para trazer resultados de máquinas de busca onde os agentes móveis utilizam algoritmo genético para selecionar as melhores rotas, tentando minimizar o custo enquanto mantém um atraso aceitável na definição do caminho.

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. Nesta tese vamos definir o agente móvel míope, aquele que só conhece os nodos que poderão ser visitados imediatamente a seguir. As informações obtidas pelo agente móvel em um dado nodo, em função do objetivo da aplicação, geram o interesse em visitar outros nodos, assim como eliminam o interesse em visitar nodos que antes eram originalmente atrativos. Desta forma, o agente móvel míope necessita rever seu itinerário planejado após a visita de cada nodo.

Mesmo que algumas das abordagens acima descritas comportem a definição dinâmica do itinerário, em nenhuma delas foi tratada a questão da miopia do agente móvel.

Nas abordagens propostas por (BREWINGTON, 1999) e (BAEK, 2001) o itinerário do agente é definido estaticamente. Em (CHANG, 2000), (QI, 2001) e (WU, 2004) são utilizados métodos hídridos onde a definição do itinerário pode acontecer estática ou dinamicamente. Mas os nodos a serem visitados são conhecidos previamente.

Em (BREWINGTON, 1999) o agente móvel, antes de partir para sua missão, consulta alguns módulos de controle que contêm informações sobre a localização dos recursos (prováveis nodos a serem visitados) e também as condições da rede para se alcançar estes nodos. Somente depois de conhecer todo seu itinerário, o agente parte em busca do cumprimento da tarefa de recuperação que lhe foi incumbida.

O mesmo acontece em (BAEK, 2001), onde são apresentados dois algoritmos que definem a seqüência de nodos para cada agente móvel visitar e também estipulam o número mínimo de agentes possíveis para realizar a tarefa com o menor tempo de execução.

Em (CHANG, 2000), onde é utilizado um método de escalonamento híbrido (o itinerário pode ser modificado após o agente móvel iniciar sua missão), todos os nodos que devem ser visitados são previamente conhecidos.

Na abordagem proposta por (QI, 2001) o problema da definição do itinerário foi estudado dentro do contexto de fusão de dados, onde o tempo de computação e o poder de processamento são os principais interesses. Uma vez que o itinerário é calculado em tempo de execução, ele consome mais tempo de computação e exige um maior processamento no sensor local. Nesta proposta existe flexibilidade sobre em qual momento o agente atinge a precisão desejada. Ele não precisa visitar os nodos sensores restantes e pode retornar ao elemento processador.

Em (WU, 2004) o agente móvel simplesmente segue a rota calculada pelo algoritmo genético de acordo com a função de aptidão. O código do cálculo permanece no PE (nodo inicial).

Em (ERFURTH, 2003) é descrita uma estrutura para o planejamento do itinerário de agentes móveis. Analisa-se também o código que será transmitido (necessidade de transmissão), buscando assim minimizar o tempo de transmissão da rede. Nesta proposta

não existe uma preocupação com o *deadline* da tarefa a ser executada. A questão da definição do itinerário é tratada em função das possíveis mudanças do ambiente, procurando o menor caminho.

Assim como na presente tese, o modelo proposto por (SCHLEGEL, 2006) baseia-se no histórico de execuções passadas do agente móvel para a tomada de decisão dinâmica. Porém, nesta tese, a tomada de decisão do agente acontece de forma a escolher a heurística mais adequada para o agente para determinada missão, enquanto que em (SCHLEGEL, 2006), a tomada de decisão do agente acontece de forma a decidir se o agente migrará ou não.

Diferentemente do que será proposto nesta tese, nas propostas anteriormente refenciadas não foi tratada a questão da miopia do agente móvel, mesmo quando a definição do itinerário acontece dinamicamente. Pressupõe-se nos trabalhos citados que o agente possui um itinerário pré-definido e por alguma alteração nas características do sistema ou da rede optou-se pela redefinição de um novo itinerário. Um dos objetivos desta tese é capacitar o agente móvel a definir seu itinerário de forma dinâmica, sendo que nos trabalhos até o momento relacionados, o agente móvel já conhece seu itinerário na partida.

Em vista das abordagens descritas na Seção 2.5, percebe-se que a utilização de agentes móveis em sistemas onde a informação encontra-se distribuída é justificável. Os algoritmos apresentados para definição do melhor itinerário resultam em execuções que geram menor custo, menor tráfego na rede e conseqüentemente trazem maiores benefícios para os sistemas em questão.

2.5. Conclusões

Este capítulo tem por objetivo esclarecer os conceitos que envolvem mobilidade de código, enfatizando os agentes móveis.

Através das características que foram relacionadas, dos serviços disponíveis apresentados que um agente móvel pode oferecer e das vantagens exibidas no uso de agentes móveis, justifica-se o interesse em unir a tecnologia de agentes móveis a sistemas com requisitos de tempo real.

A escolha do itinerário se constitui em tema de interesse nesta tese e, por este motivo, foi enfatizado neste texto. Foram descritas várias formas de atacar o problema, com objetivos diversos.

3. Sistemas de Tempo Real

3.1. Introdução

Um sistema de tempo real pode ser definido como aquele que não depende apenas dos resultados lógicos da computação, mas também do tempo no qual os resultados são produzidos (STANKOVIC, 1992).

Em (BUTTAZZO, 1999) é acrescentado que sistemas de tempo real devem reagir, sob restrições de tempo precisas, a eventos do ambiente. Uma reação que ocorre muito tarde pode ser inútil ou mesmo perigosa. A palavra real (de tempo real) indica que a reação do sistema a eventos externos deve ocorrer durante sua evolução. Como conseqüência, o tempo no sistema (tempo interno) deve ser medido usando a mesma escala de tempo usada para medir o tempo no ambiente controlado (tempo externo).

A complexidade do desenvolvimento de sistemas de tempo real (STR) tem aumentado nas últimas décadas, exigindo um entendimento minucioso da tecnologia envolvida. A crescente exigência de precisão e confiabilidade desencadeou a necessidade de aprimorar as atuais formas de desenvolvimento destes sistemas, buscando mecanismos para facilitar a compreensão, delimitar o problema e gerenciar o tempo e os custos da solução (SCHMIDTKE, 2001).

A maior parte das aplicações de tempo real se comporta como sistemas reativos com restrições temporais (FARINES, 2000). Sistemas computacionais que interagem com seus ambientes são chamados de sistemas reativos. Estes reagem enviando respostas continuamente a estímulos de entrada vindos de seus ambientes.

Alguns exemplos de áreas onde sistemas de tempo real são necessários:

- Sistemas militares: controle da trajetória dinâmica de mísseis ou jatos automatizados;
- Biomédica: a vida de uma pessoa pode depender do funcionamento correto do sistema e de respostas rápidas e confiáveis;
- Controle de tráfego aéreo: utilizado, por exemplo, para controle de decolagem e aterrissagem de aeronaves em aeroportos;

- Sistemas de controle de processos: como aplicações industriais, robótica, controle de um processo químico ou nuclear.

O restante do capítulo está organizado da seguinte forma: a Seção 3.2 descreve a caracterização de um STR. A Seção 3.3 fala sobre escalonamento em STR e o modelo de tarefas. Na Seção 3.4 o problema da alocação de tarefas é brevemente discutido. A Seção 3.5 introduz o Conceito de Computação Imprecisa. Finalizando este capítulo, a Seção 3.6, contém as conclusões.

3.2. Caracterização de um Sistema de Tempo Real

Para que um sistema possa atender às restrições impostas é necessário que ele possua algumas características especiais. As restrições sobre o tempo de computação de tarefas num STR provêem do impacto físico que isso pode causar no ambiente. Esse tempo limite é chamado de *deadline* (STANKOVIC, 1988, STANKOVIC, 1992).

Uma propriedade básica que deve estar presente num STR é a **previsibilidade**, pois um STR deve ser apto a prever as conseqüências de qualquer ação de escalonamento, e deve tirar proveito do conhecimento disponível sobre o ambiente para tratá-lo da maneira mais adequada (STANKOVIC, 1990). Um STR é dito previsível quando, independente das variações ocorridas no ambiente, o comportamento do sistema pode ser antecipado, antes de sua execução.

O termo **previsibilidade** é utilizado para descrever a capacidade de se conhecer o comportamento temporal de um sistema antes de sua execução, em função do escalonamento empregado. Ou seja, saber em tempo de projeto se as tarefas serão executadas dentro dos *deadline*s. Com relação ao comportamento temporal, a previsibilidade pode ser determinista (todos os *deadlines* serão cumpridos) ou probabilista (qual a probabilidade de um *deadline* ser cumprido) (STANKOVIC, 1990).

Outra propriedade relevante é a **tolerância a faltas**. Simples faltas de *hardware* ou *software* não devem causar a queda do sistema, um STR deve ser projetado para ser tolerante a faltas (BUTTAZZO, 1999).

Os STR podem ser classificados em relação a criticalidade em:

- **Sistemas não Críticos de Tempo Real (*soft*)**: quando o cumprimento do *deadline* é desejável por motivos de qualidade, mas sua perda não causa sérios prejuízos ao ambiente. São exemplos de STR *soft*: sistema de comutação telefônica, sistema de processamento bancário, sistemas multimídia;

- **Sistemas Críticos de Tempo Real (*hard*)**: quando o não cumprimento do *deadline* puder causar conseqüências catastróficas no ambiente controlado. São exemplos de STR *hard*: sistemas de controle de vôo, sistemas de sinalização em ferrovias, sistemas de controle de planta nuclear.

O grupo de sistemas críticos de tempo real pode ser dividido em (KOPETZ, 1997): Sistemas de Tempo Real Críticos Seguros em Caso de Falha e Sistemas de Tempo Real Críticos Operacionais em Caso de Falha. O primeiro caso é quando um ou vários estados seguros podem ser atingidos em caso de falha, o sistema pára se for necessário. No último, na presença de falhas parciais, o sistema pode degradar fornecendo alguma forma de serviço mínimo. Como por exemplo, um sistema de controle de vôo que continue o seu funcionamento em menor escala, mas ainda de forma segura.

3.3. Escalonamento em Sistemas de Tempo Real

O escalonador é o componente do sistema operacional responsável pelo agendamento de processos, ou seja, implementa uma política de escalonamento na qual determina a ordem de execução dos processos prontos para execução. O algoritmo utilizado em sua programação é chamado algoritmo de escalonamento (KOPETZ, 1997).

Quando diversos processos disputam os recursos disponíveis do sistema, existe a necessidade de escalonar esses processos. Portanto, o sistema operacional necessita de um algoritmo que compartilhe o tempo entre todos os processos em execução. Em sistemas de tempo real, os algoritmos de escalonamento são um importante componente. É responsabilidade do escalonador assegurar que os *deadlines* das tarefas sejam cumpridos, para que o sistema alcance seus objetivos.

3.3.1. Modelo de Tarefas

Tarefas ou processos formam as unidades de processamento seqüencial que ocorrem sobre um ou mais recursos computacionais de um sistema. Uma tarefa de tempo

real, além da correção lógica (*correctness*), deve satisfazer seus prazos e restrições temporais (*timeliness*) (BURNS, 2001).

Uma tarefa pode ser habilitada para executar sendo ativada por outra tarefa (quando uma tarefa termina e como parte de sua saída ativa outra tarefa – *event-trigged*) ou pela passagem do tempo (*time-trigged*) (OLIVEIRA, 1997).

Em relação à periodicidade da ativação, as tarefas podem ser classificadas em periódicas ou aperiódicas. As tarefas periódicas são ativadas sempre a cada período. Já o momento da ativação de tarefas aperiódicas é imprevisível.

Tarefas de tempo real tipicamente são subordinadas ao seu *deadline*. As conseqüências de uma tarefa ser concluída após seu *deadline* definem dois tipos de tarefas (STANKOVIC, 1990): **tarefas críticas (*hard*)** e **tarefas não-críticas (*soft*)**. Assim como nos STR críticos, uma tarefa é crítica quando a sua não execução dentro do prazo pode causar falhas catastróficas no STR e em seu ambiente. Uma tarefa é classificada como não-crítica quando a sua não execução dentro do prazo provocam falhas sem um custo significativo, no máximo uma diminuição da qualidade do sistema.

Sistemas que possuem algum serviço composto de uma ou mais tarefas críticas são chamados de sistemas de tempo real críticos. Sistemas críticos devem possuir alta previsibilidade. Quando não há tarefas com *deadline* crítico no sistema, ele é conhecido como sistema de tempo real não-crítico. Descumprir *deadline*s neste caso é aceitável em termos de custos para a aplicação ou para o usuário, embora não seja desejável (KOPETZ, 1997).

Outras restrições temporais são importantes na definição do comportamento temporal de uma tarefa (BURNS, 2001):

- **Tempo de Computação (*computation time*)**: é o tempo necessário para a execução completa da tarefa;
- **Tempo de Início (*start time*)**: é o instante do início do processamento da tarefa em uma ativação;
- **Tempo de Término (*completion time*)**: é o instante de tempo em que se completa a execução da tarefa na ativação;

- **Tempo de Chegada (*arrival time*)**: é o instante em que o escalonador toma conhecimento de uma ativação dessa tarefa;

- **Tempo de Liberação (*release time*)**: é o instante que uma tarefa está pronta para ser executada (instante em que é inserida na fila de prontos);

- **Tempo de Resposta (*response time*):** é o intervalo de tempo entre a chegada da tarefa e o término de sua execução;

- **Variação da Liberação (*release jitter*):** representa a máxima variação dos tempos de liberação das instâncias da tarefa. É a variação entre o tempo de chegada de uma tarefa e seu tempo de liberação.

Em um STR, uma tarefa ativada por um estímulo do ambiente deve ser completada até seu prazo final (*deadline*). De um modo geral, uma tarefa de tempo real difere de uma tarefa não tempo real por apresentar três tipos de restrições (BURNS, 2001): **temporal**, a atividade deve completar sua execução antes de seu *deadline* para evitar danos ao sistema; **precedência**, um STR deve considerar as ordens de precedência entre tarefas para efeitos de escalonamento e sincronização; e a **alocação de recursos**, quando duas ou mais tarefas acessam recursos exclusivos, deve-se garantir a exclusão mútua, bloqueando outras tarefas quando uma delas estiver acessando esses dados (seção crítica).

Em muitas aplicações de sistemas multitarefa faz-se necessário que as tarefas estejam sincronizadas, estabelecendo algumas relações de precedência entre si, isto é, duas ou mais tarefas não podem executar de modo arbitrário mas sim em uma seqüência que respeite as precedências impostas pela sincronização. Essas relações de dependência podem ser representadas através de um grafo acíclico dirigido, onde os nodos são tarefas e as arestas são as relações de precedência (BUTTAZZO, 1997). O escalonador de tempo real deve levar em consideração a precedência de tarefas de tal modo a garantir o cumprimento de *deadline*s e demais restrições dos STR.

Além da precedência, muitas tarefas também necessitam de recursos de vários tipos para que possam ser executadas. Em (BUTTAZZO, 1997), do ponto de vista da tarefa, um recurso é qualquer estrutura de software que possa ser usada pela tarefa para avançar sua execução. Um recurso pode ser **compartilhado** se várias tarefas podem utilizá-lo simultaneamente, ou **exclusivo**, se puder ser usado por uma única tarefa a cada momento. As tarefas que aguardam um recurso exclusivo ser liberado são ditas

bloqueadas. Essa situação pode levar ao fenômeno da Inversão de Prioridade, onde uma tarefa fica bloqueada aguardando que outra tarefa menos prioritária libere um recurso necessário. Além disso, deve-se evitar *deadlocks* e também delimitar o tempo que uma tarefa permanece bloqueada, evitando indeterminismo no tempo de resposta.

Com a finalidade de resolver estes problemas, vários algoritmos de alocação de recursos foram propostos como o **Protocolo de Herança por Prioridade** (*Priority Inheritance Protocol* – PIP) e o **Protocolo de Limite de Prioridade** (*Priority Ceiling Protocol* – PCP) (SHA, 1990).

O escalonamento de tarefas periódicas geralmente é dirigido por prioridades. Escalonamentos baseados em prioridades apresentam bom desempenho e flexibilidade (CHENG, 1998). Os algoritmos clássicos de prioridade fixa **Taxa Monotônica** (*Rate Monotonic*) (LIU, 1973), ***Deadline* Monotônico** (*Deadline Monotonic*) (LEUNG, 1982) e o algoritmo ***Earliest Deadline First*** (EDF) (LIU, 1973) que apresenta atribuição dinâmica de prioridades, são tidos como ótimos para determinadas classes de problemas, sendo caracterizados por modelos de tarefas simplificados (FARINES, 2000).

3.4. Problema da Alocação de Tarefas

No contexto de sistemas distribuídos de tempo real, o escalonamento é geralmente resolvido em duas etapas. Na primeira etapa é feita a alocação das tarefas aos processadores. Na segunda é feita o escalonamento local de cada processador, individualmente. Este escalonamento local considera como carga as tarefas alocadas na primeira etapa.

O objetivo da alocação é garantir que as tarefas poderão ser executadas antes de seus *deadline*s. Para isso, o algoritmo de alocação deve tentar diversas soluções de alocação, verificando para cada uma delas a escalonabilidade das tarefas (OLIVEIRA, 1997).

Muitas vezes, uma tarefa já está associada a um processador em particular como requisito de projeto. Por exemplo, tarefas que precisam acessar recurso computacional disponível em apenas alguns dos processadores ou atuam diretamente sobre o ambiente através de um dispositivo físico. Neste caso, a tarefa deve ser necessariamente alocada a

um dos processadores que dispõe do recurso em questão ou onde se encontra o dispositivo físico.

Também é comum especificar no projeto que determinado subconjunto de tarefas deve ficar necessariamente no mesmo processador. Isto decorre da forte interação entre estas tarefas e da conseqüente vantagem em mantê-las no mesmo processador.

3.5. Computação Imprecisa

Em STR existe uma dificuldade intrínseca em compatibilizar dois objetivos fundamentais: garantir que os resultados serão produzidos no momento desejado e dotar o sistema de flexibilidade para adaptar-se a um ambiente dinâmico.

Uma das técnicas existentes na literatura para resolver o problema de escalonamento tempo real é a **Computação Imprecisa** (LIU, 1994). A Computação Imprecisa é uma abordagem promissora em STR por adaptar a qualidade das computações. Quando ocorre sobrecarga no sistema os requisitos mínimos (por exemplo, *deadline*) são garantidos pelo decréscimo da qualidade da computação (NAKANISHI, 2000).

A técnica de Computação Imprecisa flexibiliza o escalonamento de tempo real (OLIVEIRA, 1997). A dificuldade encontrada no escalonamento de tempo real levou à proposta desta abordagem que tem por objetivo a execução do trabalho possível dentro do tempo disponível, assim diminuindo a qualidade dos resultados para poder cumprir os requisitos temporais exigidos.

Na Computação Imprecisa cada tarefa da aplicação possui uma parte obrigatória (*mandatory*) e uma parte opcional (*optional*). A parte obrigatória da tarefa é capaz de gerar um resultado com a qualidade mínima necessária para manter o sistema operando de maneira segura. A parte opcional refina o resultado, até que se alcance a qualidade desejada. O resultado da parte obrigatória é dito **impreciso** (*imprecise result*), o resultado das partes obrigatória + opcional é dito **preciso** (*precise result*). Uma tarefa é chamada de **tarefa imprecisa** se for possível decompô-la em parte obrigatória e parte opcional (OLIVEIRA, 1997a).

3.5.1. Formas de Programação

Existem três formas básicas de programar tarefas imprecisas normalmente citadas na literatura (OLIVEIRA, 1997):

- **Funções Monotônicas**: são funções onde a qualidade do resultado aumenta na medida em que o tempo de execução da função aumenta. O processamento necessário para obter-se um nível mínimo de qualidade corresponde a parte obrigatória. Qualquer processamento a mais faz parte da parte opcional. O nível mínimo de qualidade deve garantir uma operação segura do sistema, enquanto a parte opcional refina progressivamente o resultado da tarefa. É a forma de programação que oferece maior flexibilidade ao escalonador.

Quando uma função monotônica é empregada, o tempo do processador alocado à parte opcional pode ser qualquer valor entre zero e o tempo máximo de execução da parte opcional. Qualquer tempo de processador fornecido ajuda a melhorar a qualidade de resultado (Figura 3.1 (a)).

- **Funções de Melhoramento**: tem por finalidade produzir saídas no mínimo tão precisas quanto as correspondentes entradas. O valor de entrada é melhorado de alguma forma. As funções de melhoramento normalmente formam partes opcionais que seguem algum processamento obrigatório. Não existe benefício em executar uma função de melhoramento parcialmente, o escalonador deve optar antes de iniciar a tarefa se irá executá-la completamente ou não a executará.

A função de melhoramento utiliza restrição 0/1, ou seja, uma vez iniciada a parte opcional ela deve ser executada até o final (Figura 3.1(b)).

- **Múltiplas Versões**: normalmente são empregadas duas versões. A versão primária gera um resultado preciso, porém possui um tempo de execução no pior caso desconhecido ou muito grande. A versão secundária gera um resultado impreciso, porém seguro para o sistema, em um tempo de execução menor e bem conhecido. O escalonador escolhe qual versão será utilizada a cada ativação da tarefa. Deve ser garantido, no mínimo, tempo de processador para a execução da versão secundária.

Múltiplas versões também apresentam restrição 0/1. Neste caso o escalonador é obrigado a executar completamente a parte opcional (escolhendo a versão primária) ou

descartá-la completamente (escolhendo a versão secundária). Esta decisão deve ser tomada antes do início da execução da parte obrigatória (Figura 3.1 (c)).

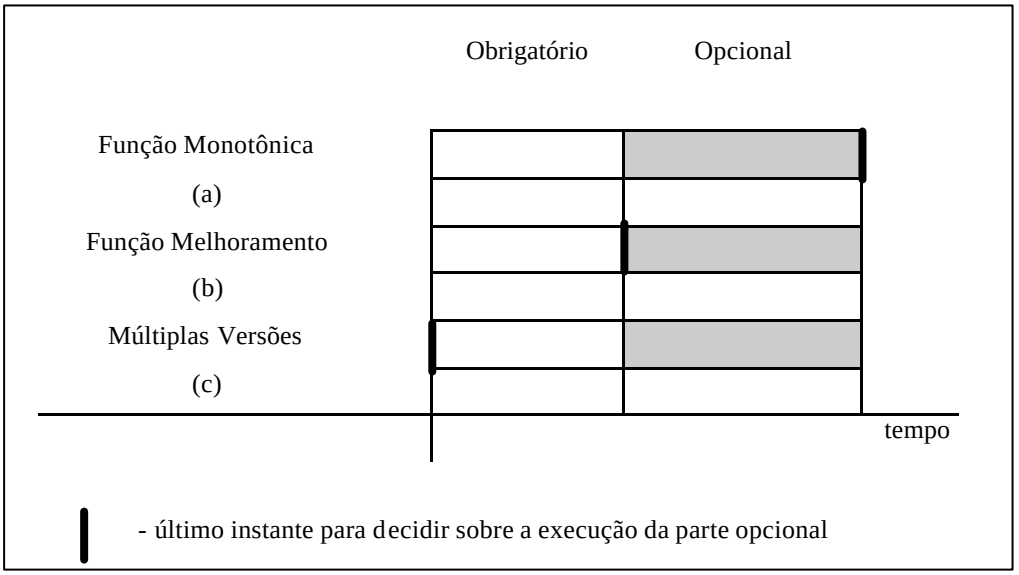


Figura 3.1: Último instante para decisão sobre execução da parte opcional (OLIVEIRA, 1997a).

3.5.2. Função Erro

Pode-se dizer que uma tarefa gerou um resultado de qualidade máxima quando a parte opcional desta tarefa foi executada completamente. Assim, quando a parte opcional não for executada completamente o resultado possui uma qualidade inferior.

Para efeito de escalonamento, algumas propostas associam um valor de erro a cada parte opcional não executada completamente. Este valor quantifica a diferença entre a qualidade do resultado preciso e a qualidade do resultado efetivamente gerado (LIU, 1994).

Quando uma parte opcional é executada completamente, ela gera um erro zero e um benefício máximo. Uma parte opcional completamente descartada gera um erro máximo e um benefício zero. Portanto, pode-se definir benefício com o sentido oposto ao erro.

Segundo (OLIVEIRA, 1997), o valor de erro pode ser proporcional ao tempo de execução que faltou para concluir a parte opcional em questão. É suposta uma linha reta para a relação “erro da parte opcional” x “tempo de execução” (Figura 3.2 (a)). Na prática, a função erro não tem sempre o formato de uma reta, podendo gerar uma curva linear, côncava, convexa ou ainda possuir um formato mais complexo, isso dependerá do

algoritmo em questão. Por exemplo, um algoritmo para cálculo numérico realiza iterações que convergem a um resultado na forma de uma curva côncava (Figura 3.2 (b)).

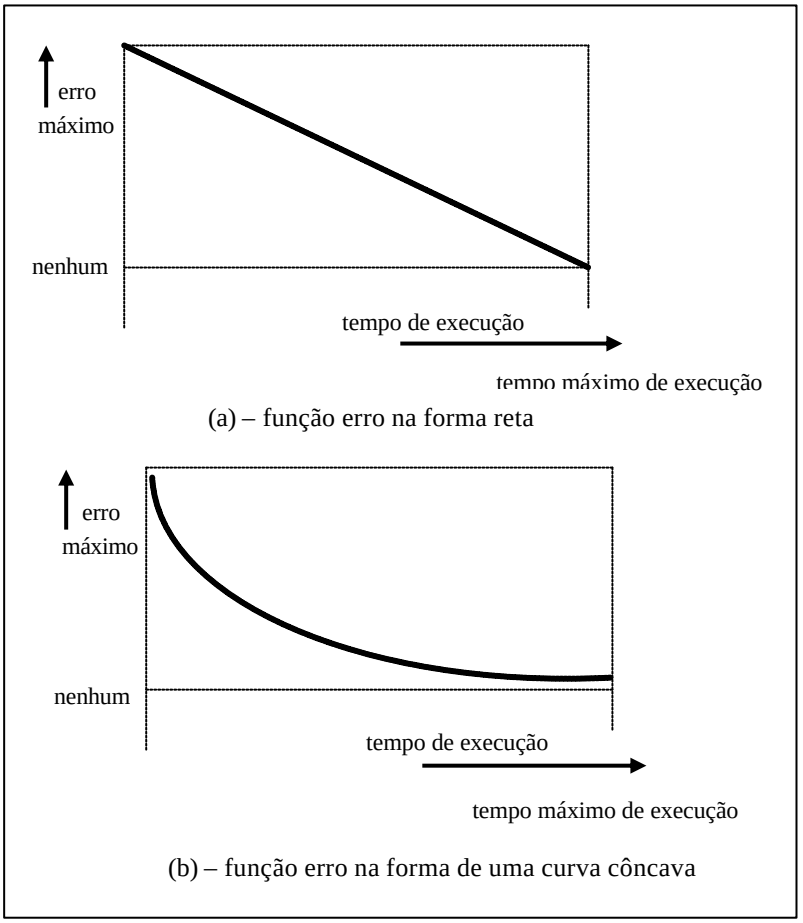


Figura 3.2: Representação da Função Erro (OLIVEIRA, 1997).

A combinação de tempo de execução e qualidade de dados de entrada como determinantes do benefício gerado pela parte opcional resultam num conjunto de retas, como ilustra a Figura 3.3.

Na Computação Imprecisa é necessário deixar claro qual o objetivo a ser considerado no escalonamento das partes opcionais. Esse objetivo será considerado em caso de sobrecarga. Segundo (OLIVEIRA, 1997), alguns dos objetivos geralmente encontrados na literatura com relação ao uso da função erro no escalonamento são: minimizar o erro total, minimizar o erro individual máximo, minimizar o número de partes opcionais descartadas e escalonar conforme a importância.

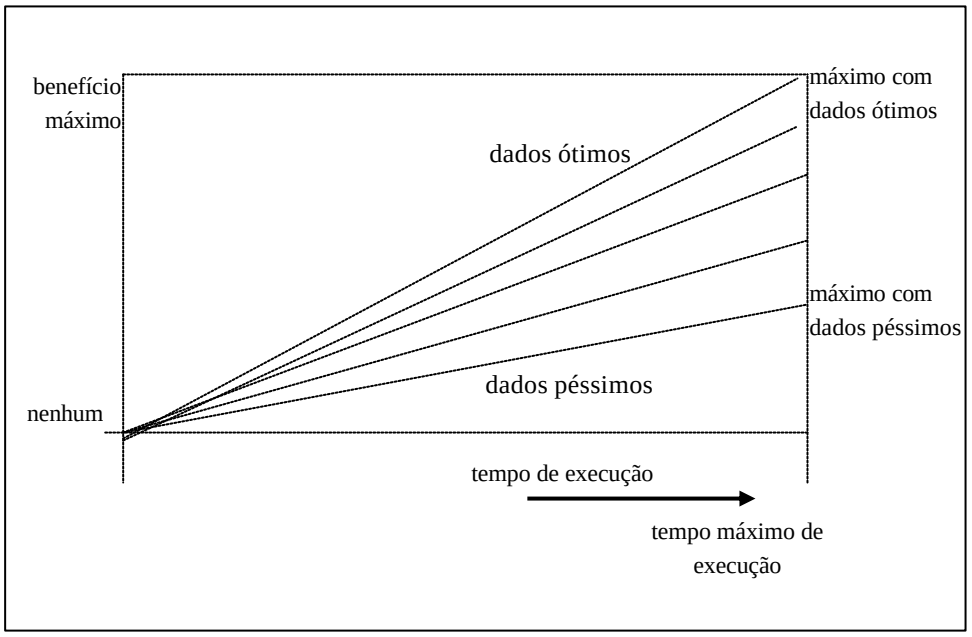


Figura 3.3: Função de Erro na Forma de um Conjunto de Retas (OLIVEIRA, 1997).

3.5.3. Escalonamento

Um dos principais aspectos da Computação Imprecisa é o escalonamento das tarefas, considerando que essas são compostas por partes opcional e obrigatória. A parte obrigatória de uma tarefa deve ter seu *deadline* garantido pelo escalonador (em tempo de projeto), enquanto que as partes opcionais devem ser escalonadas (ou não) visando maximizar a utilidade do sistema em tempo de execução ou minimizar o erro gerado pela não execução de algumas computações opcionais (OLIVEIRA, 1997).

O termo escalonamento preciso (*precise schedule*) é usado para descrever uma solução de escalonamento onde as partes opcionais são completamente executadas, todas as tarefas sempre geram um resultado preciso. Num escalonamento satisfatório (*feasible schedule*) as tarefas sempre executam sua parte obrigatória, as partes opcionais podem ou não ser executadas completamente. Todo escalonamento preciso é também satisfatório, mas nem todo escalonamento satisfatório é preciso (OLIVEIRA, 1997).

Exemplo de algoritmos de escalonamento para computação imprecisa podem ser encontrados em (CHEN, 1995), (NAKANISHI, 2000), (FENG, 1996), (LIU, 1994a), (NATARAJAN, 1995).

3.6. Conclusões

Um STR é um sistema computacional que deve reagir a estímulos do ambiente cumprindo prazos específicos. O comportamento correto de um STR não depende apenas da integridade dos resultados obtidos mas também dos valores de tempo em que são produzidos.

Em STR, os algoritmos de escalonamento são um importante componente, é responsabilidade do escalonador assegurar que os *deadline*s das tarefas sejam cumpridos, para que o sistema alcance seus objetivos.

A técnica de Computação Imprecisa flexibiliza o escalonamento de tempo real. Essa técnica adapta a qualidade das computações, quando ocorre sobrecarga no sistema os requisitos mínimos são garantidos pelo decréscimo da qualidade da computação.

Este capítulo apresentou a conceituação de STR, seguido pelos aspectos da teoria de escalonamento para STR. A idéia destes algoritmos aliada a definição de Computação Imprecisa serviram de base para o modelo que será posteriormente proposto.

4. Agentes Móveis e Restrição Temporal

4.1. Introdução

Alguns trabalhos na literatura fornecem estudos envolvendo a tecnologia de agentes móveis com requisitos de tempo real. Agentes móveis podem resolver o problema de latência na rede, pois podem ser despachados de um controlador central para agir localmente e executar diretamente as diretivas de controle. Sistemas de tempo real, por exemplo, robôs e sistemas de manufatura, não toleram atrasos na resposta a mudanças em seu ambiente. A eficiência nestas operações em tempo real é um requisito imprescindível.

Este capítulo apresenta um estudo de abordagens que envolvem a tecnologia de agentes e requisitos temporais. O objetivo deste capítulo é descrever políticas de controle de recursos para suportar código móvel tendo como resultado final a utilização dos recursos computacionais de acordo com as necessidades e características de cada tarefa.

A Seção 4.2. apresenta propostas envolvendo o uso da tecnologia de agentes móveis que apresentam restrições temporais. Na Seção 4.3 é discutida a questão da chegada da tarefa móvel nos nodos e sua escalonabilidade. Na Seção 4.4, o tema de interesse é o planejamento de agentes móveis com restrição temporal na definição de seu itinerário, abordagens que tratam sobre este assunto são discutidas nesta seção. A Seção 4.5. apresenta uma proposta interessante que não envolve o uso de agentes móveis, mas estão relacionadas a área de interesse desta tese. Os comentários gerais sobre as abordagens envolvendo agentes móveis e restrição temporal são expostos na Seção 4.6. E finalizando, a Seção 4.7 contém os comentários finais.

4.2. Aplicações de Agentes Móveis com Requisitos Temporais

Esta seção apresenta propostas que envolvem o uso da tecnologia de agentes móveis que apresentam restrições temporais. Justifica-se o estudo destas propostas por fazerem parte da área de interesse deste trabalho.

4.2.1. ARS: Um Sistema de Roteamento Baseado em Agentes para Garantir QoS

Em (OIDA, 2000) é descrito um sistema de roteamento baseado em agentes (ARS) o qual aloca recursos de rede para tarefas que solicitam comunicação ponto a ponto em tempo real. Com seu mecanismo de gerenciamento de recursos, agentes móveis coletam o atual estado dos enlaces do sistema para examinar a disponibilidade de possíveis caminhos em cada nodo.

Uma garantia de serviço para comunicações ponto a ponto pode ser fornecida através da alocação de uma quantidade de largura de banda na rede para as comunicações. A base matemática para tal se argumenta na idéia que se um fluxo origem é limitado por um mecanismo de controle de fluxo baseado em taxa, os limites superiores do atraso e do *jitter* do fluxo podem ser determinados pela largura de banda do fluxo (OIDA, 2000). O próximo passo para alcançar garantia de serviço é considerar como alocar os recursos da rede eficientemente para cada pedido da tarefa, onde um recurso é uma largura de banda dividida de acordo com o pacote escalonável.

Um roteamento baseado em QoS (*Quality-of-Service routing*) usualmente seleciona de forma periódica possíveis caminhos que satisfazem algumas restrições de QoS (por exemplo, todos os enlaces ao longo do caminho devem ter um número determinado de recursos não reservados). Este roteamento deve prover caminhos realmente possíveis, já que recursos são reservados para cada pedido ao longo do caminho que ele seleciona. Se a reserva de recurso é desenvolvida sobre um caminho que atualmente não está disponível (isto é, não encontra atendidos os seus requisitos de QoS), então a reserva para o fluxo falha.

Um roteamento baseado em QoS nem sempre satisfaz os requisitos solicitados, especialmente quando a taxa de chegada de pedidos do usuário é alta e o número de nodos na rede é grande. Isto provoca mudanças rápidas no número de recursos não reservados no enlace, enquanto que o número de enlaces que devem satisfazer as restrições de QoS aumenta. Nestes ambientes, a disponibilidade dos caminhos selecionados muda freqüentemente. Para fornecer um caminho realmente possível, um roteamento baseado em QoS deve atualizar caminhos possíveis mais freqüentemente, isto resulta numa excessiva utilização dos recursos da rede para o roteamento de mensagens.

Para resolver este problema foi proposto o ARS. O ARS consiste de roteamento baseado em QoS, reserva de recurso e controle de admissão global. O controle de admissão

global no ARS é baseado na seguinte idéia (OIDA, 2000): uma vez que os agentes movem-se através da rede procurando caminhos possíveis e reservando recursos, estes agentes carregam informações como o estado dos enlaces entre os nodos que eles devem visitar. Esta informação é utilizada para decidir se um caminho é realmente possível ou não.

O ARS utiliza agentes móveis no roteamento baseado em QoS para reunir informações. O algoritmo de roteamento é classificado como um algoritmo de roteamento baseado em agentes móveis, no qual utiliza uma colônia artificial de "operários" que trabalham cooperativamente para descobrirem caminhos possíveis. O AntNet (CARO, 1998) serviu como inspiração para o ARS.

Um agente usado para reservar ou liberar recursos carrega o estado dos enlaces que tenham mudado. Esta informação carregada pelos agentes tem alto impacto na disponibilidade de caminhos possíveis.

Na primeira etapa, cada nodo na rede mantém um caminho pré-estabelecido que inclui todos os caminhos possíveis - ***f-paths*** - para todos os pares que tenham por destino o nodo ***d,*** largura de banda mínima ***m***, e um conjunto de estado de todos os enlaces que compõem os caminhos possíveis. Dado que ***P(s,d,m)*** denota um ***f-path*** do nodo ***s*** para o nodo ***d*** com uma largura de banda não menor que ***m*** entre os dois nodos, e ***s(n1,n2,m)*** denota um estado que indica se o enlace ***(n1,n2)*** tem no mínimo ***m*** unidades de recursos não reservados, então o caminho pré-estabelecido no nodo ***s*** inclui:

$$\mathbf{P(s,d,m), \{ s(n1,n2,m) \mid (n1,n2) \in P(s,d,m)\}, \forall\ d \in S - \{s\}} \qquad \mathbf{[1]}$$

onde ***S*** **- {s}** representa o conjunto de todos os nodos menos o nodo ***s***.

Na segunda etapa, considera-se que o estado do enlace carregado pelo agente é representado pela tupla ***(k,l,r)***, onde ***l*** é o nodo vizinho do nodo ***k*** e ***r*** é o número de recursos não reservados do enlace ***(k,l)***. De tempos em tempos um agente *backward* chega no nodo ***s*** e atualiza os estados dos enlaces nos caminhos pré-estabelecidos. A atualização acontece da seguinte forma: para cada ***f-path P(s,d,m)*** no caminho pré-estabelecido, se ***(k,l)*** $\in$ ***P(s,d,m)***, então o correspondente estado do enlace ***s(k,l,m)*** é atualizado. Receberá **1** (caminho possível) somente se ***r*** **>=** ***m***, senão receberá **0** (caminho inalcançável). Assim, a combinação das etapas 1 e 2 torna alguns caminhos disponíveis ou não, dependendo das características atuais da rede.

4.2.2. Um Esquema de Geração de Oferta Baseado em Agentes Móveis para Escalonamento Tempo Real em Sistemas de Manufatura

Sistemas de manufatura tradicionais podem não ser adequados para ambientes que mudam rapidamente tais como aqueles que possuem alta variedade de produção e ciclo de vida curto. Empresas de manufatura devem ter habilidade para responder rapidamente a mudanças dinâmicas do ambiente. Então, um sistema de manufatura inteligente e distribuído é considerado uma escolha inevitável. A essência de um sistema de manufatura inteligente e distribuído é o controle baseado em negociação. No sistema proposto por (SHIN, 2001) todas as tarefas de tomada de decisão são implementadas através de negociação entre agentes inteligentes e autônomos.

Este sistema aplica a mobilidade de código para um processo de negociação entre sistemas isolados e distribuídos visando um controle baseado em negociação eficiente e inteligente. Um protocolo de negociação baseado em *agent*es móveis (MANPro) é aplicado em um sistema de escalonamento de tempo real. Este protocolo emprega a mobilidade de um *agent*e no processo de negociação entre os sistemas. O MANPro define a estrutura de controle baseada em agentes móveis e o módulo gerador de oferta (SHIN, 2001).

Este protocolo é composto por quatro módulos básicos: um **gerenciador de oferta** (*bid manager* – BM) que supervisiona todo processo de negociação, um **agente de tarefa** (*T-agent*), um **agente de negociação** (*N-agent*) e um **agente de recursos** (*R-agent*). A Figura 4.1 apresenta o protocolo MANPro. O BM e o *T-agent* estão localizados em um controlador virtual, o qual não está fisicamente fixo no sistema como um controlador centralizado. O *N-agent* é um agente móvel que viaja pela rede de controladores de recurso para coletar ofertas. O *R-agent* fornece ao *N-agent* informações sobre o estado do recurso através da interface com a base de dados local, isto é, de cada controlador de recurso. O *T-agent* é responsável pela preparação e criação do *N-agent*. Quando a viagem é completada o *N-agent* repassa a lista de ofertas para o *T-agent* que avalia as ofertas e envia a mensagem escolhida para o *R-agent* selecionado.

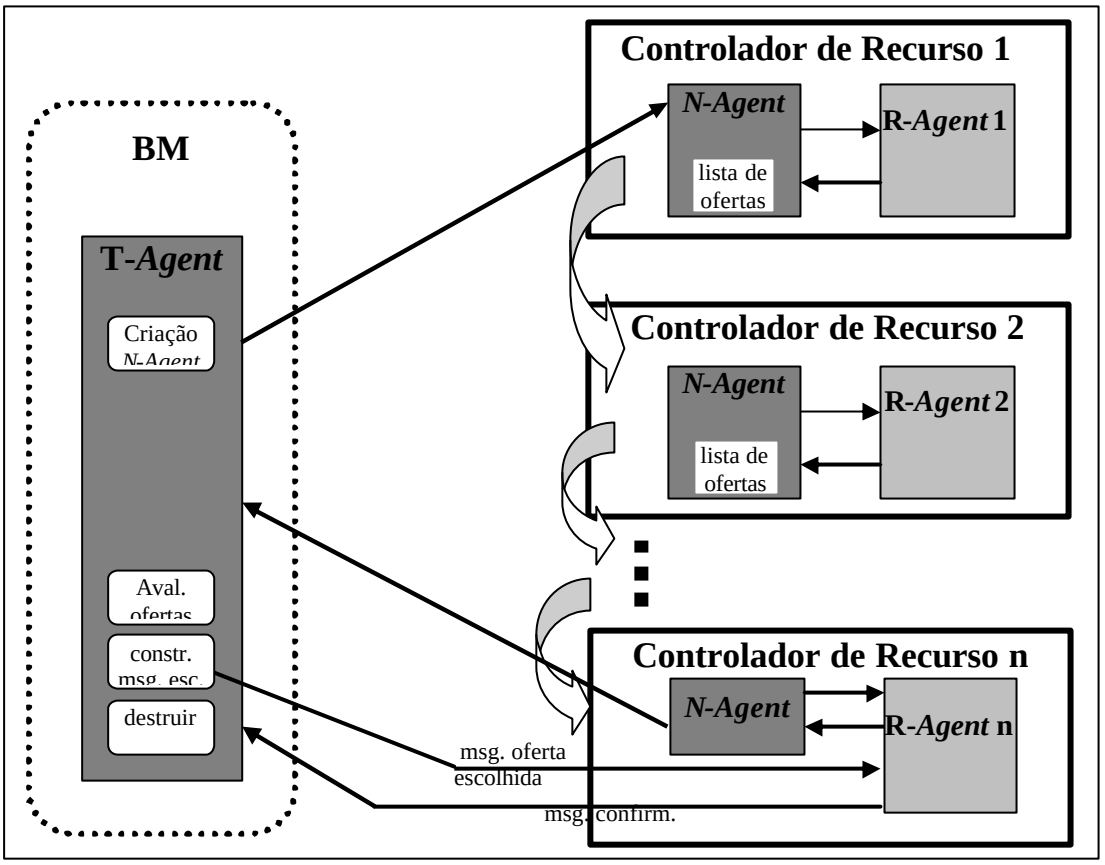


Figura 4.1: Processo de Negociação Baseado no MANPro (SHIN, 2002).

Quando o BM decide inicializar um processo de negociação, ele gera o *N-agent* que viaja através dos controladores de recurso. Em cada controlador de recurso, o *N-agent* recebe informação sobre o estado dos recursos através de uma interface fornecida pelo *R-agent* que gera uma oferta de acordo com a política oferecida (o *R-agent* não pode participar de mais de um processo de negociação ao mesmo tempo, pois isto o tornaria não confiável e ineficiente). O *N-agent* move-se então para outro controlador de recurso. Quando a viagem estiver completa, o *N-agent* informa a lista de ofertas para o *T-agent*. Depois que o *T-agent* avalia as ofertas, ele envia uma mensagem (*task offer*) para o *R-agent* selecionado. Quando o *T-agent* recebe a mensagem de confirmação de recebimento, ele destrói a si próprio e o processo de negociação está completo.

O processo de negociação é aplicado para escalonamento em tempo real de um sistema de manufatura. O *N-agent*, que é um tipo de agente móvel, viaja pela rede de controladores de recurso e gera ofertas para tornar escalonável uma dada tarefa. Ele considera um escalonamento pré-organizado de um recurso e faz uma nova escala. Esta escala torna-se a oferta que é submetida ao *T-Agent*. Visto que o *N-agent* possui regras

baseadas em inteligência e restrições para escalonamento, ele pode adaptar-se dinamicamente.

O BM e o *N-agent* devem ser desenvolvidos de acordo com o domínio da aplicação e o objetivo da negociação, determinando a seqüência na qual as tarefas serão escalonadas através do processo de negociação. O módulo gerador de oferta (embutido no *N-agent*) possui informações sobre a política de escalonamento e o algoritmo de escalonamento, gerando uma escala para a tarefa em um controlador local.

Em um sistema de fabricação distribuído, muitas tarefas solicitam ser escalonadas em tempo real, o sistema de escalonamento em tempo real deve considerar quando cada tarefa é escalonada. Para resolver este problema, o módulo escalonador de negociação proposto aplica o esquema de reserva de produção estendida EXPRES (KIM, 2000) para escalonar o processo de negociação. O EXPRES introduz uma reserva baseada em janela flexível (*sliding window*), onde o alvo não são tarefas que deveriam ser imediatamente escalonadas, mas tarefas que poderiam ser escalonadas com uma certa flexibilidade de tempo.

O módulo gerador de oferta usa um mecanismo de otimização com restrição de tempo para encontrar uma escala ótima. Ele é composto pela fase de geração da oferta inicial e a fase de melhora. Na primeira fase, o *N-agent* encontra um intervalo de tempo que satisfaça as restrições temporais da nova tarefa e esteja em harmonia com as demais restrições temporais das tarefas pré-arranjadas. Entretanto, a oferta pode ser melhorada para encontrar uma solução ótima. Isso acontece na fase de melhora, na qual a nova tarefa pode ser escalonada em um intervalo de tempo reservado para uma tarefa pré-arranjada, diminuindo assim a latência máxima. Essa troca somente acontece se não houver relação de precedência entre as tarefas.

4.2.3. RT Monitor – Monitoramento de Dados Tempo Real usando Tecnologia de Agentes Móveis

O RT Monitor é um sistema de gerenciamento de dados tempo real para tráfego de aplicações de navegação na rede (RAMAMRITHAM, 2002). Neste sistema, pedidos de navegação com restrições de tempo são originados e o RTMonitor calcula e comunica os melhores caminhos para as tarefas solicitantes baseando-se no caminho pela rede e nos

dados de tráfego tempo real. A precisão das rotas sugeridas dependerá da consistência dos dados de tráfego.

Para minimizar o *overhead*, o RTMonitor adota um método cooperativo e distribuído usando agentes móveis que pode reduzir a quantidade de comunicação e melhorar a escalabilidade do sistema. Foi projetado um esquema de grafos de tráfego com 2 níveis (global e regional) para organizar dados de tráfego tempo real e suportar os pedidos de navegação. Cada servidor de região mantém um grafo do tráfego local baseado nos caminhos das conexões e nos dados de tráfego de sua região. Um grafo global virtual também é mantido para o sistema completo.

Pela necessidade de exatidão, um esquema adaptativo para monitoramento (*Adaptive Push or Pull Scheme* – APoP) foi formulado, no qual o período monitorado é definido baseado na urgência do pedido, no estado do sistema e nas propriedades dinâmicas dos dados de tráfego. Este esquema é responsável por manter a consistência temporal dos dados de tráfego.

O sistema é composto por dois tipos de agentes: agentes estacionários e agentes móveis. Os agentes estacionários são principalmente usados para controlar os grafos de tráfego de forma que a consistência temporal dos dados de tráfego nos grafos seja mantida de acordo com os requisitos das tarefas solicitantes. Os agentes móveis são usados principalmente para atender os pedidos de navegação.

Algumas vantagens no uso de agentes móveis para resolver o problema do monitoramento de dados são (RAMAMRITHAM, 2002):

- agentes móveis podem minimizar o *overhead* de comunicação e a quantidade de dados a serem transmitidas entre os servidores para calcular e monitorar o melhor caminho para o cliente (aplicação que solicita informações sobre o melhor caminho);

- o sistema torna-se mais adaptativo para mudanças de situações no ambiente;

- já que um agente móvel pode ser executado assincronamente e autonomamente, uma conexão fixa não é necessária e o agente pode operar mesmo sobre ambientes desconectados.

Em cada servidor de região existe dois agentes estacionários (o *LocalServerAgent* e o *GraphAgent*) e dois agentes móveis (o *GraphMonitor* e o *QueryAgent*). O

LocalServerAgent representa um servidor regional para criar outros agentes em tempo de execução e o *GraphAgent*, situado no *LocalServerAgent*, mantém o grafo local de sua região e um grafo virtual. O *GraphMonitor* é gerado pelo *GraphAgent*. Este agente é despachado e estacionado em regiões remotas para coletar dados de tráfego de *GraphAgents* remotos para atualizar o grafo virtual. O *QueryAgent* é inicializado por clientes móveis, se movimentam com os clientes e geram pedidos para o *GraphAgent* a fim de encontrar informações sobre o melhor caminho em nome dos clientes. As Figuras 4.2 e 4.3 apresentam o modelo de agentes RT Monitor.

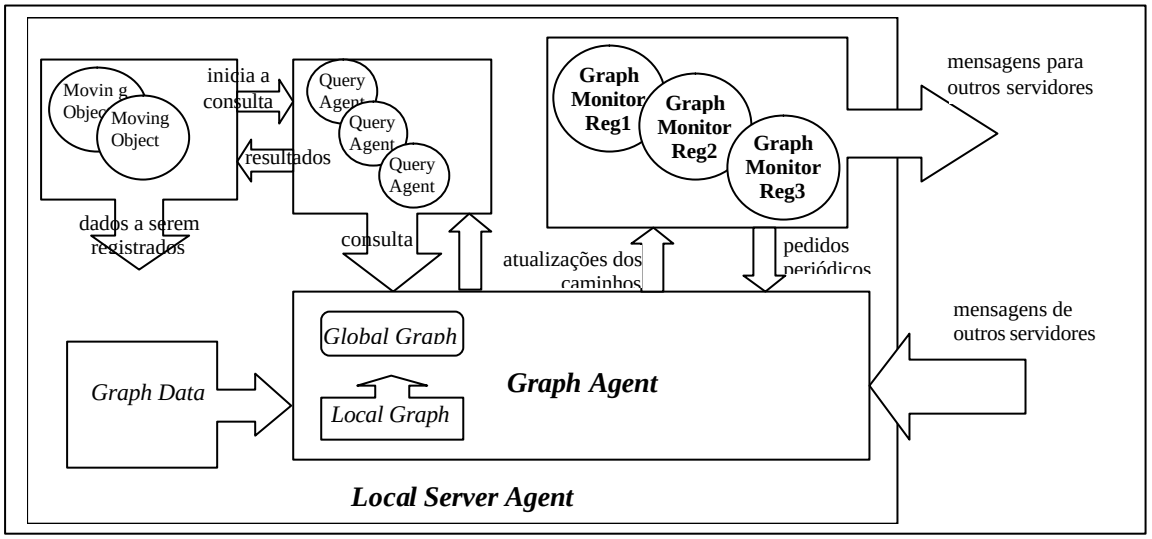


Figura 4.2: Modelo de Agentes do RT Monitor (RAMAMRITHAM, 2002).

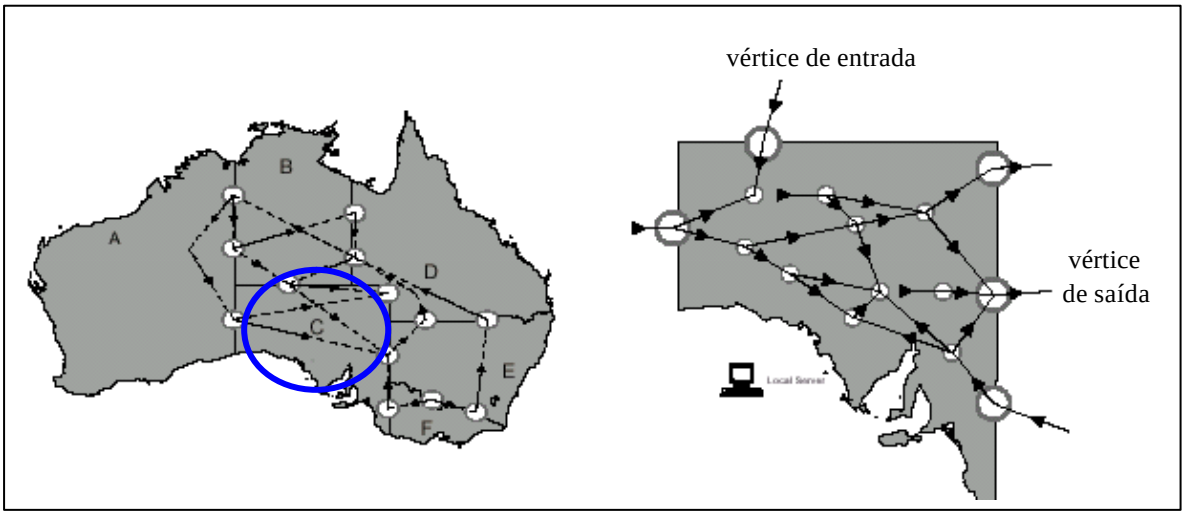


Figura 4.3: Grafo Virtual Global (à esquerda) e Grafo do Tráfego Local da Região C (à direita) (RAMAMRITHAM, 2002).

Para calcular o melhor caminho é importante utilizar os dados de tráfego mais recentes (os dados são consistentes temporariamente). Assume-se que existem sensores de tráfego para capturar estes dados. Uma atualização de tráfego deve ser enviada ao servidor de sua região sempre que um novo valor for significativamente diferente do valor anterior. Entretanto, freqüentes recálculos podem induzir a uma sobrecarga nos servidores. Para minimizá-la sem afetar a precisão (*correctness*) dos resultados de navegação, o esquema APoP é utilizado.

Quando o *LocalServerAgent* recebe um pedido de navegação, ele gera o *QueryAgent* para o cliente e gera um pedido para o seu *GraphAgent* calcular o melhor caminho utilizando o algoritmo do caminho mais curto (algoritmo de Dijkstra). O melhor caminho consiste de três partes: um caminho no grafo local na região origem, um caminho virtual para a região destino e um caminho local de um ponto de entrada (vértice de entrada da região) até a aplicação destino na região destino.

O *GraphAgent* também calcula o tempo de chegada estimado no destino e a folga do tempo do pedido. Depois de servir a requisição pela primeira vez, o *GraphAgent* monitorará o melhor caminho através (RAMAMRITHAM, 2002): do monitoramento do grafo local; do envio de pedidos para seus *MonitorAgents* no qual estão nas regiões remotas para monitorar o caminho virtual para estas regiões remotas; e do envio de uma requisição para o seu *MonitorAgent* (localizado na região destino) para monitorar o grafo local da região destino. Se a tarefa solicitante entrar em outra região, o trabalho de monitoramento será feito pelo *GraphAgent* da nova região e seus *MonitorAgents.*

A informação do novo grafo virtual será passada ao *MonitorAgents* da região e então o *MonitorAgents* passará a informação para seus *GraphAgents* para construir novos grafos virtuais em suas regiões. O *GraphAgent* de uma região determina o período de recálculo baseado na folga de tempo do conjunto atual de pedidos no qual está servindo, na sobrecarga de seu servidor e nas propriedades dinâmicas dos dados de tráfego. Se a folga de tempo é menor e os valores das arestas mudam rapidamente então um período de recálculo menor será usado para prover um monitoramento mais próximo ao grafo local.

4.2.4. *Agent Cloning*: Um método para Mobilidade de Agente e Alocação de Recursos

Sistemas multiagentes (SMA) estão sujeitos a apresentar gargalos quando os recursos são insuficientes. Soluções para este problema incluem desde troca de tarefas para outro agente (local) ou migração do agente para um nodo remoto. SHEHORY (1998) propôs clonagem de agentes como um método perspectivo para resolver o problema de sobrecarga no agente local. A clonagem de agente inclui transferência da tarefa e mobilidade do agente. De acordo como este paradigma, agentes podem ser clonados, passar tarefas a outros agentes, morrer ou se unirem.

A clonagem é uma possível resposta de um agente a sobrecarga. A sobrecarga do agente não implica no *overhead* da máquina, portanto, a clonagem local (na mesma máquina) pode ser possível.

Uma vez que a carga de um agente varia de forma indeterminística, a tomada de decisão de se ou quando clonar um agente não é trivial. Em (SHEHPORY, 1998) é apresentado um modelo estocástico de tomada de decisão baseado em programação dinâmica para determinar o tempo ótimo para clonagem do agente.

Supondo que um clone foi criado e ativado, muitas questões permanecem com relação a este clone, por exemplo, sua autonomia, tarefas, tempo de vida e acesso aos recursos. A autonomia referencia a independência versus um clone subordinado (um clone subordinado permanece sobre o controle de seu iniciador). Sendo uma vez criado e ativado, um clone independente não é controlado pelo seu criador. Entretanto, desta maneira um clone continua existindo depois de completar a execução da tarefa. Por isso, um mecanismo para decidir o que ele deveria fazer posteriormente é necessário. Este mecanismo deve decidir se o clone continuará trabalhando com outras tarefas (se for necessário ou se os recursos computacionais permitirem), unir-se-á com outros agentes ou morrerá.

Um agente deve considerar a clonagem se (SHEHORY, 1998):

- Não puder desenvolver todas as suas tarefas em tempo;
- Não existir nenhum agente com carga leve que possa receber e executar a tarefa excedida (ou sub-tarefa);
- Existirem recursos suficientes para criar e ativar um agente clone (remoto ou local);

- A eficiência do agente clone e do agente original é esperada ser maior que a eficiência do agente original sozinho.

As informações necessárias usadas pelo agente para decidir se e quando iniciar a clonagem consiste de parâmetros que descrevem os recursos locais e remotos. Por exemplo: carga da memória e da CPU; velocidade de execução da CPU; carga dos canais de comunicação e suas taxas de transmissão; fila atual das tarefas, os recursos solicitados para sua execução e seus *deadline*s; fluxo futuro de tarefas estimado.

O algoritmo de clonagem proposto por SHEHORY (1998) consiste dos seguintes componentes:

- Raciocínio antes da clonagem: inclui o raciocínio sobre a lista de tarefas com relação à restrição temporal, capacidade e requisitos dos recursos. As considerações da lista de tarefas assim como as habilidades do agente, a capacidade e carga do nodo resultam na decisão entre clonar ou transferir a tarefa para um agente já existente;

- Clonagem: inclui a criação e ativação do clone, a transferência das tarefas, e as atualizações resultantes entre os agentes. As ações básicas a serem tomadas são: criar uma cópia de seu código (esta cópia, entretanto, pode sofrer alguma modificação), e o agente deve transferir para seu clone apenas a sub-tarefa relevante e as informações necessárias para a tarefa passada ao clone. Caso contrário, o clone pode enfrentar o mesmo problema de sobrecarga de seu criador;

- Raciocínio depois da clonagem: coletar informações com relação aos benefícios da clonagem e propriedades do ambiente (tais como: distribuição da tarefa), e analisá-los estaticamente como forma de aprendizagem para clonagens futuras.

A clonagem é desenvolvida quando um agente percebe ou prevê uma sobrecarga, desta maneira crescendo a habilidade do SMA desenvolver tarefas. SHEHORY (1998) apresentou um modelo de clonagem de agentes para balanceamento de carga, assim melhorando o desempenho das tarefas do SMA em execução em muitas máquinas remotas.

4.2.5. Aplicações de Agentes Móveis em um Sistema de Monitoramento Tempo Real Baseado na Web

A tecnologia de agentes móveis pode ser utilizada a fim de superar as limitações da largura de banda da rede em tarefas de monitoramento em tempo real. Em (ONG, 2003) foi proposto uma arquitetura distribuída baseada em Web para o desenvolvimento de uma plataforma independente em um sistema de monitoramento em tempo real. Programas de monitoramento podem ser despachados de um controlador central como um agente móvel para nodos remotos onde sinais de máquina são fornecidos de máquinas reais. Um programa de monitoramento pode ser despachado como um agente móvel para o nodo que enviou os requisitos para solicitar o programa de monitoramento.

Redes neurais e técnicas de sistemas especialistas estão sendo cada vez mais encontradas em aplicações no campo de diagnóstico de ferramentas de máquinas. Em (HU, 2001) é proposto um sistema de diagnóstico inteligente baseado na combinação de redes neurais e técnicas de sistemas especialistas. A integração destas duas técnicas de inteligência artificial é particularmente apropriada para diagnóstico de falhas em máquinas em tempo real.

O sistema de monitoramento de tempo real proposto em (ONG, 2003) envolve aplicação de agentes móveis e consiste de um controlador central e suas máquinas remotas subordinadas. A estrutura deste sistema é ilustrada na Figura 4.4.

Nesta estrutura, diferentes tipos de máquinas podem ser remotamente conectadas pela rede. Estas máquinas distribuídas são conectadas por um controlador central através do qual elas podem solicitar programas de monitoramento específicos. Um *request agent* é responsável por carregar um pedido para o controlador central de um nodo remoto, encapsulando os requisitos correspondentes de uma máquina remota que está conectada ao nodo remoto. Isto é, um nodo remoto pode criar um agente para garantir um programa de monitoramento adequado de um centro de recurso através da rede. Um programa de monitoramento adequado pode ser encontrado e retornado (*agent returned*, como mostra a Figura 4.4) para o nodo remoto para monitorar a máquina localmente.

Em situações onde existem rotas entre múltiplos destinos, o nodo remoto pode também fornecer o *request agent* com um itinerário e enviá-lo para a primeira localização no itinerário. Um itinerário mantém um lista de destinos, define um esquema de roteamento e controla casos especiais em que o agente móvel sempre sabe qual é o

próximo nodo. Um itinerário também pode permitir que um agente seja salvo e reutilizado mais tarde por conveniência. O agente pode atuar autonomamente de acordo com a rota dada.

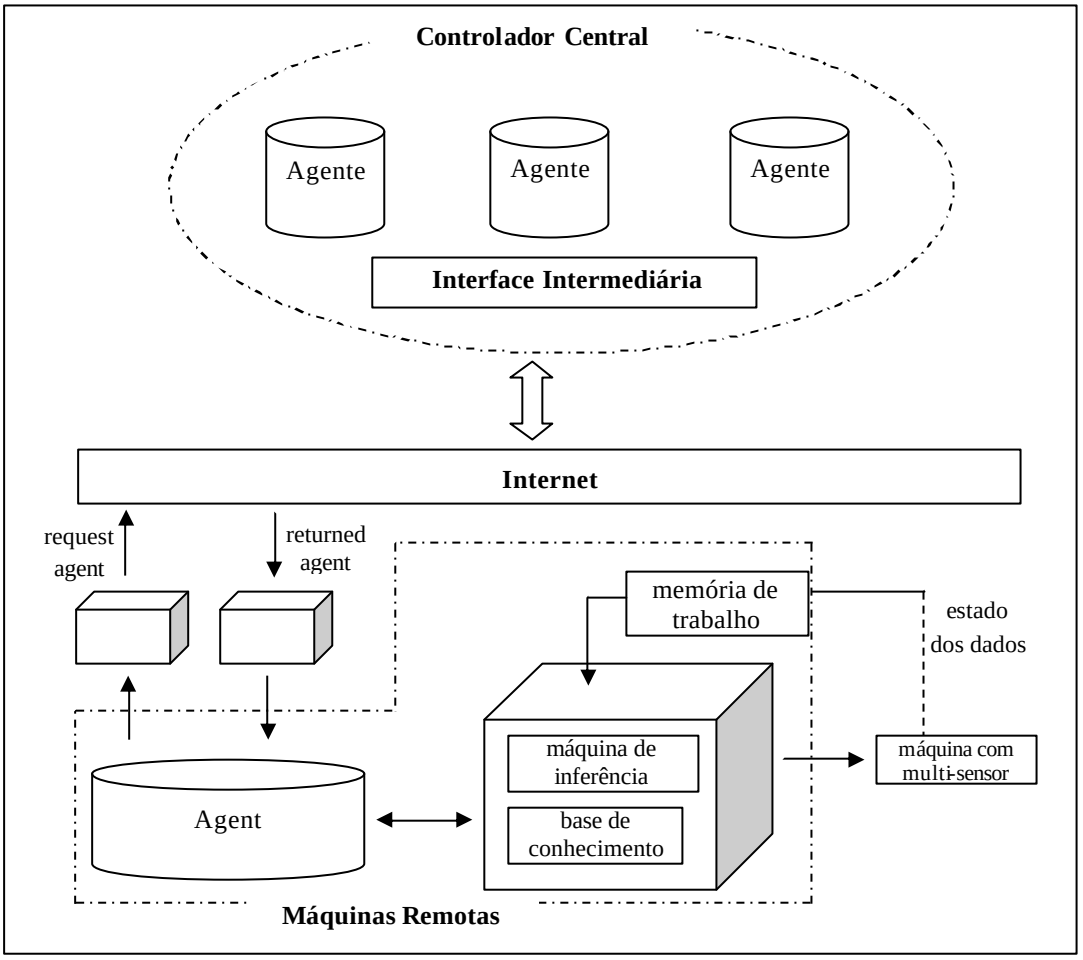


Figura 4.4: Arquitetura de um Sistema de Monitoramento Tempo Real baseado em Web.

O controlador central é o centro de recursos dos programas de monitoramento, e ele contém muitos agentes. Neste sistema, um agente é um programa de monitoramento específico. O controlador central consiste de duas partes: um módulo intermediário e um módulo de monitoramento. O módulo intermediário serve como uma ponte entre os *request agent*s que chegam e o módulo de monitoramento interno.

Depois de receber uma solicitação de um agente que chegou, o controlador central pesquisará por programas de monitoramento apropriados para a máquina remota. O módulo de monitoramento possui muitos programas de monitoração. Cada programa está embutido em um agente móvel. Neste artigo existem três tipos de máquinas: máquinas de torneamento, de perfuração e de corte. Existe uma base de dados para cada tipo de

máquina e cada aspecto a ser monitorado (tais como: força de corte, emissão acústica, temperatura, etc.) no controlador central.

Os programas de monitoramento podem ser implementados de várias maneiras usando técnicas de inteligência artificial, tais como sistemas especialistas baseados em regras e redes neurais. Existem três níveis distintos em um sistema especialista:

- Memória de trabalho (*working memory*): se referencia aos dados específicos da tarefa para o problema em consideração. Os dados dos multi-sensores são transferidos e processados aqui;

- Base de conhecimento (*knowledge base*): consiste das regras de solução dos problemas, procedimentos, e dados relevantes para o domínio do problema;

- Máquina de Inferência (*inference engine*): é um mecanismo de controle genérico que aplica o conhecimento evidente na base de conhecimento para uma tarefa específica chegar a uma solução ou conclusão.

Na arquitetura descrita acima, os agentes móveis possuem uma importante função. Eles funcionam como programas de monitoramento, o controlador central pode despachá-los para os nodos remotos onde os sinais de máquina são fornecidos por máquinas reais.

4.3. Escalonamento Local de Agentes Móveis que Chegam ao Nodo

Esta seção discute a questão da chegada da tarefa móvel nos nodos e sua escalonabilidade. As propostas apresentadas descrevem modelos de escalonamento baseados em cota fixa, onde as tarefas móveis e os nodos podem especificar restrições de como o processador deve ser alocado.

O principal objetivo de um esquema de escalonamento é construir uma escala que satisfaça um conjunto de restrições de uso de recursos. Essas restrições irão especificar como os recursos devem ser alocados (LAL, 1999).

A fim de atender os requisitos temporais impostos por tarefas de tempo real que executam juntamente com tarefas sem requisitos temporais, diferentes políticas de controle de recursos de processador vêm sendo estudadas e apresentadas.

4.3.1. Controle de Recurso do Processador para Tarefas Móveis

Para atender aos requisitos necessários para ambientes compostos por tarefas com ou sem requisitos temporais, algumas políticas híbridas têm sido propostas (LAL, 1999). Usualmente, tarefas de tempo real são separadas das tarefas não tempo real e, a cada um desses grupos, uma política diferente é aplicada.

Em (LAL, 1999), os autores focaram o problema da realocação de recursos para tarefas móveis utilizando a política de liberação de recursos baseada em cota. Um esquema de escalonamento que controla a alocação de recursos do processador para tarefas móveis é descrito. Dentro desta estrutura, as tarefas móveis e os nodos podem especificar restrições de como o processador deve ser alocado. Baseado nessas restrições, é construído um grafo de escalonamento no qual aplica-se algoritmos de escalonamento. No caso de conflitos entre restrições de tarefas móveis e dos nodos, o esquema implementa uma política que resolve os conflitos a favor do nodo, ou seja, priorizando as restrições/limitações do nodo.

As duas principais preocupações com relação às restrições específicas do nodo quanto ao uso de recursos são: segurança, por exemplo impedindo o acesso de programas externos não autorizados; e qualidade de serviço, provendo diferentes níveis de serviço para diferentes categorias de tarefas móveis. Por exemplo, um nodo pode querer alocar mais recursos para tarefas originárias de seus nós sócios.

O esquema inclui:

- Especificação das restrições de uso de recursos: como compartilhamento, prioridade, limite superior (quantidade máxima da capacidade do processador que a tarefa pode utilizar em cada unidade escalonável) e *deadline*;

- Agrupamento hierárquico: tarefas móveis são organizadas em grupos tendo por base domínio de rede ou recursos;

- Algoritmos de escalonamento: o esquema consiste de três algoritmos para impor restrições às tarefas móveis: um algoritmo para impor restrições de prioridade em tarefas não tempo real, um algoritmo para impor restrições de *deadline* em tarefas tempo real e um algoritmo para impor restrições de segurança;

- Política de composição do algoritmo: utilizada para resolver conflitos entre restrições de uso de recursos.

Quanto à alocação de recursos, existem duas visões: a visão das tarefas móveis (como pretendem consumir os recursos) e a visão do nodo hospedeiro (como pretende distribuir seus recursos). Na primeira, as tarefas móveis querem utilizar os recursos do processador o quanto for possível para que executem rapidamente. São especificados alguns requisitos: **limite mínimo** (quantidade mínima da capacidade do processador que a tarefa utiliza em cada unidade escalonável); **peso** (percentual da capacidade do processador que a tarefa pode alocar a cada unidade escalonável), **parte** (relacionado ao peso da tarefa, em quantidades relativas) e ***deadline***. Na perspectiva do nodo existem duas preocupações, atender as necessidades do cliente e controlar a alocação de recursos do servidor. Um nodo pode especificar as seguintes restrições: **prioridade**, **limite superior** (quantidade máxima da capacidade do processador que a tarefa pode utilizar em cada unidade escalonável), **restrição de tempo de vida** (limite de unidades tempo do processador para uso da tarefa sobre todas suas execuções).

O esquema particiona uma linha de tempo contínua em pequenos espaços de tempo QTC (*quantum time chunks*), dentro de cada QTC são escalonadas tarefas móveis do grupo de tarefas tempo real de acordo com suas reservas. O tempo restante fica disponível para tarefas não tempo real (Figura 4.5).

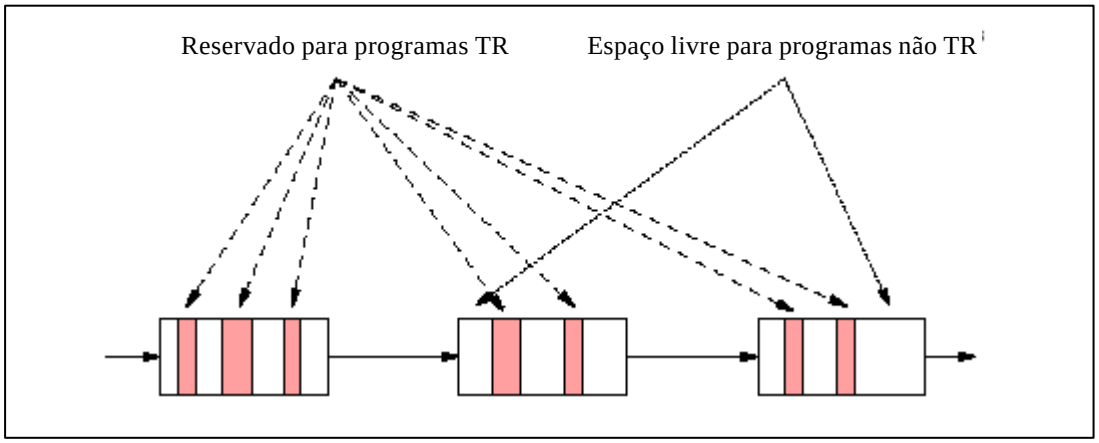


Figura 4.5: Lista de QTC com reservas para Programas Tempo Real (LAL, 1999).

O escalonamento de tarefas móveis é baseado em uma modificação do algoritmo de escalonamento Rialto (JONES, 1997). O Rialto originalmente utiliza um grafo de escalonamento pré-calculado continuamente, garantindo as reservas do processador com tarefas de períodos definidos. Os nodos no grafo de escalonamento do Rialto indicam o período de tempo reservado para a tarefa ou o tempo livre que não foi reservado para nenhuma tarefa. As tarefas fazem reservas no processador da seguinte forma: “reserve *x*

unidades de tempo a cada *y* unidades". Tarefas de tempo real solicitam recursos do processador especificando o tempo na forma <**S**, **E**, **T**>, onde é feita uma reserva dentro do intervalo de tempo entre **S** e **E** do tempo de computação **T**.

O esquema de escalonamento (LAL, 1999) é mais simples se comparado ao Rialto, pois no esquema não são consideradas reservas contínuas do processador na forma "reserve *x* unidades de tempo a cada *y* unidades" mas as reservas do processador são definidas sobre QTC. Com essa simplificação não é necessário calcular o grafo de escalonamento do Rialto. Dentro de cada QTC, as tarefas de tempo real são escalonadas de acordo com suas reservas. As reservas fixam os tempos no qual o processador é alocado para tarefas de tempo real. Quando uma nova tarefa chega, o algoritmo verifica se o pedido pode ser atendido dentro do *deadline*.

4.3.2. Qualidade de Serviço para Programas Móveis de Tempo Real

O trabalho desenvolvido por (MONTEIRO, 2002) apresenta uma proposta de qualidade de serviço para código móvel considerando a precedência e as restrições de execução de cada tarefa, levando em conta a característica dinâmica dos ambientes de código móvel. Esta política permite diferenciar o nível de serviço a ser oferecido conforme a classe da tarefa móvel. Para isso foi considerada uma estrutura que apresenta dois níveis: uma política que gerencia os requisitos de qualidade de serviço no nível do cliente e um esquema de escalonamento de tarefas para ambientes compostos por tarefas com ou sem requisitos temporais.

O algoritmo de escalonamento se aplica a um ambiente multitarefas, uniprocessado e preemptivo. Seu objetivo é manter a taxa de execução das tarefas de tempo real, do tipo *soft*, e também atender as tarefas não tempo real (MONTEIRO, 2002).

O esquema particiona uma linha de tempo contínua em pequenos espaços de tempo QTC (*quantum time chunks*), dentro de cada QTC são escalonadas tarefas móveis do grupo de tarefas tempo real de acordo com suas reservas. O tempo restante fica disponível para tarefas não tempo real O tempo virtual de referência (TVR) é a soma dos tempos TVC e TVP.

O valor do TVC só é incrementado quando a tarefa é selecionada para execução, assim seu valor permanece constante enquanto a tarefa estiver aguardando ser chamada

pelo escalonador. Quanto ao TVP, seu valor deve ser decrementado de uma unidade a cada novo *quantum*.

Um *quantum* é uma quantidade fixa de tempo dedicada a execução de uma tarefa sem que ocorra interrupção. O valor do *quantum*, definido pelo esquema de escalonamento, deve ser suficientemente pequeno a fim de garantir a granularidade desejável ao atendimento das tarefas de tempo real, porém não tão pequeno que resulte num chaveamento de contexto excessivo. Se uma tarefa terminar sua execução antes do término de seu *quantum* apenas será adiantado o início do processamento da próxima tarefa.

Nesse esquema de escalonamento, as tarefas são dispostas em duas filas: uma fila principal e uma fila secundária. Na fila principal são colocadas as tarefas de tempo real ativas, além de uma tarefa ociosa, chamada de *REST*, relacionada ao controle do tempo restante que deve apontar para a fila secundária. A fila secundária é destinada às tarefas não tempo real.

Ambas as filas são ordenadas pelo TVR. A tarefa ativa que possui o menor TVR ocupa a cabeça da fila e, portanto, será a próxima tarefa a ser selecionada para execução. O algoritmo opera primeiramente sobre a fila principal (composto por tarefas de tempo real). Quando a cabeça da fila é ocupada pela tarefa ociosa *REST*, o próximo *quantum* é destinado a tarefa que ocupa a cabeça de fila secundária, ou seja, destinado a uma tarefa não tempo real.

4.4. Definição do Itinerário de Agentes Móveis com Requisitos Temporais

A seqüência de nodos visitados por um agente móvel entre um nodo origem e um nodo destino é chamada itinerário. O itinerário estático é inteiramente definido antes da partida do agente e não muda durante sua execução. O itinerário dinâmico está sujeito a modificações do próprio agente (PLEISCH, 2004).

No Capítulo 2, na Seção 2.5, foram apresentadas propostas que envolvem o cálculo do itinerário para agentes móveis buscando o melhor caminho. A seguir serão apresentadas abordagens sobre o planejamento de agentes móveis com restrição temporal na definição de seu itinerário.

4.4.1. Planejamento de Agentes Móveis com Tempo Determinado (*timed*) para Recuperação de Informação Distribuída

Em (BAEK, 2001a) é proposto um método de planejamento de agentes móveis com restrição de tempo (TMAP - *Timed Mobile Agent Planning*) para encontrar o número mínimo de agentes e o melhor itinerário para os agentes realizarem a recuperação de informação em um ambiente de computação distribuído, respeitando as restrições de tempo enquanto mantém o tempo mínimo de roteamento total.

Os principais fatores do desempenho no planejamento de agentes móveis (MAP, *Mobile Agent Planning*) são (BAEK, 2001a): o número de agentes móveis, o tempo consumido pelos agentes participantes e as restrições de tempo.

Se agentes móveis conhecem as características da rede como: latência, largura de banda dos enlaces entre os nodos e a carga computacional de cada nodo pode-se fazer o planejamento do itinerário dos agentes.

Uma área de aplicação para agentes móveis é a recuperação de informação distribuída, onde grande quantidade de dados pode ser acessada através de várias redes. A informação é propagada através de muitos nodos. Em um sistema de recuperação de informação as restrições de tempo são (BAEK, 2001a): o *ready-line* e o *deadline*. O *ready-line* é o tempo o mais cedo possível que a informação esperada está disponível ou para ser atualizado seu conteúdo com uma informação mais nova. Por exemplo, se o agente enviado chega cedo (antes do tempo de *update* especificado) e coleta informações, o recebedor pode entregar alguma informação antiga ou corrompida.

Os algoritmos BYKY1 e BYKY2 (apresentados na Seção 2.6.2.) (BAEK 2001) negociam com os dois primeiros fatores no planejamento de agentes: o número mínimo de agentes e o tempo de roteamento total consumido. Entretanto, em função de situações inclinadas a qualquer tipo de restrição temporal, como recuperação de informação com *deadline*, o MAP precisou ser aperfeiçoado para suportar restrições de tempo.

Foi então desenvolvido o algoritmo TMAP. Informalmente, o TMA é descrito da seguinte maneira: há *n* nodos onde a informação está disponível. Cada nodo tem um tempo de computação e uma janela (*time window*). Um tempo de computação é solicitado para um agente móvel para executar uma tarefa no nodo. A tarefa do agente móvel é válida

somente durante a janela do nodo. A janela de um nodo é definida pela informação do *deadline* e *ready-line.* Assume-se que as latências para mover os agentes móveis são conhecidas. O objetivo do planejamento é encontrar o número de agentes móveis e o itinerário de cada agente para minimizar o tempo de execução total sob as restrições da janela de tempo. Para uma tarefa do agente ser validada dentro da janela de tempo, deve respeitar a seguinte regra:

$$Max\ \{ ready(hi), Ls(hi,H) \} + Comp(hi) <= Dead(hi),\ 1 = i = n \qquad [2]$$

onde ***ready(hi)*** é o tempo a partir do qual a informação já está "pronta" no nodo ***hi***, ***Ls(hi,H)*** representa a latência mínima entre os nodos ***hi*** e ***H*** (***H*** é o nodo origem), ***Comp(hi)*** é o tempo de computação no nodo ***hi***, ***Dead(hi)*** é o *deadline* da informação no nodo ***hi*** e ***n*** é o número de nodos menos o nodo origem.

Este método de planejamento melhora significativamente o uso da tecnologia de agentes móveis (BAEK, 2001a).

4.4.2. Análise de um Algoritmo de Roteamento Basedo em Tráfego de Agentes Móveis

Em (QU, 2005) é proposto um algoritmo de roteamento baseado em agentes móveis no qual o custo de tráfego é considerado. Em cada nodo, existe uma distribuição de probabilidade para o agente móvel selecionar um dos nodos vizinhos e mover-se até ele. Uma função de custo é definida para cada enlace baseado nas informações de tráfego conhecidas e na distribuição de probabilidade encontrada.

Uma vez que um pedido para enviar um pacote para o destino é recebido do servidor, o servidor gera um número de agentes móveis. Cada agente carrega o endereço de seu servidor, seu destino, seu nodo anterior e alguma informação de controle para o roteamento como: seu *life-span* e um contador do número de nodos visitados. Depois de gerados, esses agentes viajam pela rede.

Quando o agente alcança um nodo, ele verifica se é seu nodo destino, se não for ele escolherá o próximo nodo vizinho a visitar de acordo com o custo do enlace para os nodos vizinhos e o número de usuários naquele enlace. Um enlace com baixo custo e poucos usuários será selecionado com prioridade. Uma vez que um agente alcançou seu destino, ele voltará ao servidor (através do caminho utilizado) atualizará as tabelas de roteamento nos nodos ao longo do caminho e apresentará sua informação sobre o melhor

caminho ao servidor. Quando um determinado número de agentes volta, o servidor seleciona o caminho ótimo seguindo certos critérios e envia o pacote para o destino através do novo caminho. Ao mesmo tempo o servidor atualiza sua tabela de roteamento.

Seja ***G = {V, E}*** um grafo correspondente a uma rede fixa, onde ***V*** é o conjunto de vértices (nodos) e ***E*** é o conjunto de arestas (enlaces). ***NB(i)*** é o conjunto de nodos vizinhos do nodo ***vi*** e **|*NB(i)*|** é o número de nodos em ***NB(i)***.

Originalmente, cada nodo não possui informação sobre seus nodos vizinhos e enlaces. Portanto, cada vértice no conjunto ***NB(i)*** possui a mesma probabilidade de ser selecionado, isto é, ***1 / |NB(i)|***. Esta distribuição de probabilidade uniforme da seleção do nodo vizinho do agente será atualizada a medida que os agentes iniciem suas viagens.

As novas distribuições de probabilidades devem satisfazer duas restrições:

1) é feita uma inferência das informações de tráfego conhecidas;
2) é não tendencioso, isto é, a probabilidade deveria, na maioria das vezes, balancear o custo de tráfego em cada enlace.

QU et al. em (QU, 2005) modelaram matematicamente estas restrições. O efeito da informação de tráfego conhecida na tomada de decisão do agente pode ser expressa por:

$$\min_{x \in R^n} \; f^{(j)}_{max}(x) \qquad [3]$$

onde ***x*** é uma variável aleatória com ***n*** entradas, o qual indica ***n*** itens a serem considerados no custo de um enlace. A função de custo de tráfego é definida como:

$$f^{(j)}_{max}(x) = \max_{i \in NB(j)} \{f_{ji}(x)\}, \qquad [4]$$

onde $f_{ji}(x)$ é a função de custo de tráfego do nodo ***vj*** ao nodo ***vi***. Ao mesmo tempo, o requisito de imparcialidade é expresso pela função de entropia máxima (*maximum entropy function*). A entropia é uma medição do grau de incerteza. O problema da otimização combinatorial resulta em uma distribuição de probabilidade que pode ser expressa por:

$$p_{ji} = \frac{\exp\{q\, f_{ji}(x)\}}{\sum_{l \in NB(j)} \exp\{q\, f_{ji}(x)\}}, \quad j = 1,2,\ldots;\ i \in NB(j), \qquad [5]$$

onde p_{ji} é a probabilidade que um agente no nodo ***vj*** migre para o nodo ***vi***, $q \geq 0$ é um coeficiente de peso definido de acordo com o estado conhecido do tráfego da rede.

O propósito de utilizar a teoria da entropia máxima no processo de busca dos agentes é encontrar uma distribuição de probabilidade que satisfaça a informação de roteamento conhecida e na maioria das vezes se aproxime da distribuição balanceada.

4.5. Aplicação de Computação Móvel com Requisitos Temporais

Nesta seção é apresentado um trabalho que não utiliza agentes móveis. Neste trabalho é discutido um esquema de roteamento de pacotes em uma rede móvel, combinando o conceito de mobilidade (computação móvel) interagindo em tempo real. Este artigo foi incluído neste capítulo pois utiliza heurísticas semelhantes - em relação à idéia – às heurísticas simples que serão apresentadas no Capítulo 6.

A essência do problema apresentado por (MOSSE, 2006) não é muito diferente da que será apresentada no Capítulo 5. Uma das questões tratadas nesta tese, que será discutida nos próximos capítulos, é a escolha do comportamento mais adequado a ser utilizado pelo agente móvel para determinada situação, ou seja, o comportamento/tomada de decisão que o agente móvel deve manter para cumprir os objetivos de sua missão (a tomada de decisão na escolha do próximo recurso afim de formar o itinerário do agente móvel).

Na proposta apresentada por (MOSSE, 2006) as heurísticas são utilizadas para escolher o nodo mensageiro, que viajará pela rede, mas a questão da escolha do itinerário a ser utilizado por este nodo mensageiro não é tratada. Diferentemente de (MOSSE, 2006), na presente tese as heurísticas são utilizadas para selecionar o tipo de comportamento do agente móvel no decorrer de sua missão. É considerada uma missão um conjunto de tarefas a serem cumpridas respeitando um *deadline*.

4.5.1. Integração de Escalonamento e Roteamento em Redes Ad-hoc Móveis Tolerantes a Atraso

Em (MOSSE, 2006) é considerado o problema de como rotear pacotes em uma rede *ad hoc* móvel esparsa (*Mobile Ad-Hoc Networks* – MANETs). Em tais redes, um nodo é considerado **líder**, se este nodo atribui tarefas e recebe informações de outros nodos, os **trabalhadores**. MANETs são freqüentemente inapropriadas para sistemas de tempo real

hard, mas podem ser úteis em situações tempo real *soft*, como em situações de emergência e recuperação que possuam vários *deadline*s aceitáveis.

MANETs são redes sem fio cujo nodos são também roteadores. Além de processarem suas próprias aplicações, nodos MANET repassam pacotes destinados a outros nodos. MANETs habilitam comunicação quando não existe uma infra-estrutura de rede (por exemplo, comunicação tática militar) ou a rede foi danificada (por exemplo, devido ao ataque de *sites* maliciosos).

O método de roteamento descrito em (MOSSE, 2006) é baseado na observação da ordem na camada de aplicação de um nodo líder MANET, que também controla a mobilidade e a habilidade de nodos trabalhadores para transmitirem as mensagens. Portanto, um líder pode atribuir não apenas tarefas na camada de aplicação, mas também tarefas mensageiras, cujo propósito é fornecer na camada de rede a mobilidade necessária para a comunicação. Este método tenta minimizar os *deadline*s perdidos e o consumo de energia através do escalonamento das tarefas trabalhadoras considerando as necessidades das camadas de rede e de aplicação.

É assumido que a rede possui um nodo líder e ***n*** nodos trabalhadores. O nodo líder recebe informações e responde enviando/atribuindo tarefas aos nodos trabalhadores. Cada tarefa possui um tempo de processamento estimado e um *deadline.*

Após unir-se a uma rede com um nodo líder, um nodo trabalhador move ou executa uma tarefa somente como resultado de uma atribuição vinda do líder. O nodo líder sempre conhece, supostamente, a localização de seus nodos trabalhadores. É assumido que o líder e os trabalhadores possuem um mapa da área da rede. O mapa é marcado com pontos de referência no qual todos os nodos (líder e trabalhadores) podem identificar e informar uns aos outros, sua posição aproximada.

A chegada de atribuições vinda do líder pode causar o particionamento da rede. O líder pode enviar um ou mais trabalhadores para um sítio que, quando alcançado, necessitará de uma rota fim a fim com o líder. Neste caso, a rede é dividida entre a partição principal (contendo o líder) e partições subordinadas (contendo os nodos trabalhadores).

Quando o nodo líder precisa se comunicar com um nodo trabalhador ***Ws*** que está na partição subordinada, o líder envia um atribuição de mensageiro para o trabalhador ***Wm*** dentro da partição principal. O ***Wm*** move-se fisicamente entre as partições para transferir a

tarefa atribuída ao ***Ws***, ele recebe informações do ***Ws*** ou de outros trabalhadores e transfere-as de volta ao líder.

O líder utiliza um algoritmo de escalonamento de mensageiro para selecionar o ***Wm***. O desempenho da missão dependerá deste algoritmo. Foram consideradas duas métricas de desempenho, primeiro é medido o percentual de *deadline*s perdidos na camada de aplicação (o algoritmo deveria minimizar esta métrica), segundo é medida a distância viajada pelos mensageiros (estimando-se assim o gasto de energia no transporte).

Foram propostos quatro diferentes heurísticas que são utilizadas pelo líder quando torna-se necessária a comunicação com um nodo em uma partição subordinada. As heurísticas diferem em como o líder seleciona o mensageiro ***Wm*** (MOSSE, 2006):

- **Mensageiro Randômico** (*random courier*): ele escolhe o ***Wm*** randomicamente entre os trabalhadores na partição principal;

- **Mensageiro Dedicado** (*dedicated courier*): o algoritmo reserva n_c nodos da partição principal para usar como mensageiro. O líder seleciona como mensageiro aquele que estiver mais próximo de qualquer nodo trabalhador na partição destino (esta seleção acontece entre os n_c nodos reservados), em caso de empate, escolherá o mensageiro com menor identificador;

- **Mensageiro mais Próximo** (*closest courier*): escolhe o mensageiro ***Wm***, entre todos os nodos da partição principal, aquele que estiver mais próximo de qualquer trabalhador na partição destino;

- **Maior Fatia** (*highest slack*): este algoritmo adiciona otimização a heurística do Mensageiro mais Próximo, escolhendo como mensageiro, o ***Wm*** na partição principal que possuir a maior fatia. O líder classifica os trabalhadores na partição principal pelo tempo máximo que cada nodo trabalhador poderia ficar ocioso sem perder o *deadline* de sua próxima aplicação. Então, ele seleciona como mensageiro o ***Wm*** cujo tempo é maior.

Quando o ***Wm*** recebe a atribuição de mensageiro, ele imediatamente interrompe qualquer tarefa ***Tm*** que estiver executando, e move-se em direção ao nodo ***Wd*** que está mais próximo a ***Wm*** na partição destino. Um mensageiro pode carregar múltiplas tarefas de atribuição em uma única viagem.

Simulações compararam o desempenho das quatro heurísticas acima descritas. A heurística do Mensageiro Dedicado agrega mais mensagens por viagem e minimiza as distâncias viajadas a custas de um maior número de *deadline*s perdidos. A heurística da Maior Fatia minimiza a perda de *deadline*s, apresentando na maior parte das situações simuladas o melhor resultado. A heurística do Mensageiro Dedicado é competitiva com a da Maior Fatia somente quando o mensageiro move-se mais rápido que os trabalhadores. Os mensageiros selecionados pela heurística Mensageiro mais Próximo tendem a perder a conectividade com a partição principal mais cedo que os selecionados pela da Maior Fatia, portanto este algoritmo aumenta a média de distância viajada. O pior resultado foi apresentado pela heurística randômica.

4.6. Comentários Gerais Sobre as Abordagens Envolvendo Agentes Móveis e Restrição Temporal

Neste capítulo foram apresentadas abordagens que mostram agentes móveis sendo utilizados de forma a auxiliar e aperfeiçoar o escalonamento de tarefas distribuídas com restrição temporal, também foram apresentadas situações onde os agentes móveis auxiliam na atualização das informações sobre as condições atuais dos enlaces.

O objetivo maior deste capítulo foi apresentar propostas encontradas na literatura que tratassem a questão da mobilidade aliada à questão da restrição temporal. Nem todas as propostas aqui discutidas possuem características ou interesses em comum com os objetivos desta tese, mas o estudo destas abordagens teve significativa importância e serviu como base para o desenvolvimento da presente tese.

Partindo da idéia inicial do problema a ser tratado nesta tese, seguido pelo estudo de abordagens que tratam de áreas afins, foi percebida uma lacuna na literatura com relação às questões tratadas nesta tese, as quais serão discutidas nos próximos capítulos.

Por exemplo, no ARS (OIDA 2000), os agentes móveis são utilizados para reunir as informações da situação atual dos enlaces da rede, através destas informações é que são encontrados os possíveis caminhos para as tarefas que solicitam comunicação ponto-a-ponto em tempo real. Através da reserva de recursos e seu controle de admissão global, o ARS consegue decidir se um caminho é possível ou não.

Na abordagem acima referida, a questão da definição do itinerário é discutida de uma forma diferente a desta tese, no ARS (OIDA 2000) a função dos agentes é verificar a possibilidade de se utilizar um itinerário previamente definido sem afetar a qualidade da execução e cumprindo seu *deadline*. Os agentes móveis coletam o atual estado dos enlaces do sistema para examinar a disponibilidade de possíveis caminhos em cada nodo.

Em (RAMAMRITHAM, 2002) os agentes também são utilizados para supervisionar as condições atuais do tráfego. Em sua estrutura são utilizados agentes móveis operando em conjunto com agentes estacionários, que calculam e comunicam os melhores caminhos para as tarefas com restrição temporal solicitantes. A questão da miopia do agente não é tratada nesta abordagem, dado que todas as possibilidades de itinerário são conhecidas antes do agente móvel partir.

Em (SHIN, 2001) a mobilidade do agente é utilizada no processo de negociação de um sistema de escalonamento de tempo real. Este processo de negociação é aplicado para escalonamento em tempo real de um sistema de manufatura. No caso, o agente móvel viaja pela rede de controladores de recurso gerando ofertas para tornar escalonável uma dada tarefa. Ele considera um escalonamento pré-organizado de um recurso e faz uma nova escala. A questão da definição de itinerário não é tratada nesta proposta.

Tanto em (OIDA 2000), (SHIN 2001) quanto em (RAMAMRITHAM, 2002) existe a necessidade em se responder rapidamente a mudanças dinâmicas do ambiente.

Diferentemente das abordagens anteriormente descritas neste capítulo, em (SHEHORY, 1998) existe uma preocupação com a quantidade de carga no agente, evitando desta forma possíveis gargalos resultantes de uma sobrecarga. Para solucionar este problema é proposta a clonagem do agente. O agente somente será clonado se não puder desenvolver todas as suas tarefas em tempo e a eficiência do agente clone e do agente original for maior que a eficiência do agente original sozinho.

Nesta tese, o uso de agentes clones é proposto para aumentar a eficiência das heurísticas de definição de itinerário (que serão descritas no Capítulo 6), onde foi possível perceber as vantagens em se combinar – através de agentes clones – o uso das heurísticas para alcançar melhores desempenhos. A miopia do agente pode limitar a eficiência das heurísticas existentes, portanto, uma possível solução foi utilizar um agente clone que adota uma segunda heurística.

Redes neurais e técnicas de sistemas especialistas estão sendo cada vez mais encontradas em aplicações no campo de diagnóstico de ferramentas de máquinas. Em (ONG 2003) os agentes móveis possuem uma importante função, eles 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. Nesta proposta não são tratadas questões do itinerário do agente. Agentes móveis foram utilizados a fim de superar as limitações da largura de banda da rede em tarefas de monitoramento em tempo real.

Em (LAL, 1999) e (MONTEIRO, 2002) são apresentados diferentes esquemas de escalonamento de tarefas para ambientes compostos por tarefas com ou sem requisitos temporais. Seus esquemas de escalonamento particionam uma linha de tempo contínua em pequenos espaços de tempo, dentro de cada "fatia de tempo" são escalonadas tarefas móveis do grupo de tarefas tempo real de acordo com suas reservas. O tempo restante fica disponível para tarefas não tempo real. Nestas duas propostas foram discutidas as questões da chegada da tarefa móvel nos nodos e da sua escalonabilidade, o itinerário percorrido pela tarefa móvel para chegar ao nodo não foi discutido.

Algoritmos de roteamento para definição do itinerário dos agentes móveis com restrição temporal foram propostos em (BAEK, 2001a) e (QU, 2005).

Para BAEK (2001a) os principais fatores do desempenho no planejamento de agentes móveis são: o número de agentes móveis, o tempo consumido pelos agentes participantes e as restrições de tempo. Na tentativa de satisfazer estes requisitos, foi proposto um método de planejamento de agentes móveis com restrição de tempo para encontrar o número mínimo de agentes e o melhor itinerário. Os agentes são utilizados na recuperação de informação em um ambiente de computação distribuído, respeitando as restrições de tempo enquanto mantém-se o tempo mínimo de roteamento total. A miopia do agente não é discutida, visto que o caminho a ser percorrido já é conhecido no momento da partida do agente.

Em (QU, 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 dos enlaces entre dois nodos é definido baseado nas informações de tráfego conhecidas e na distribuição de probabilidade encontrada.

Assim como em (QU et.al., 2005), nesta tese os agentes móveis são míopes e a probabilidade é utilizada para auxiliar o agente móvel em suas tomadas de decisão. No caso de QU et.al. (2005), utiliza-se probabilidade para decidir qual o próximo nodo a visitar, enquanto que na presente tese utiliza-se probabilidade para decidir qual o comportamento que será adotado pelo agente móvel (heurísticas adaptativas, Capítulo 7).

Outra divergência é com relação à forma que é definido o itinerário, em (QU et.al., 2005) vários agentes partem em busca do melhor itinerário, e voltam ao nodo origem (servidor). Então é selecionado o melhor itinerário, sendo que apenas um agente parte novamente para cumprir sua missão e este já conhece seu itinerário. Nesta tese, a definição do itinerário acontece de forma dinâmica.

4.7. Conclusões

Este capítulo teve por objetivo discutir propostas que inter-relacionam mobilidade de código e restrição temporal em aplicações distribuídas, comprovando assim o expressivo interesse nestas áreas.

Buscou-se uma bibliografia que desse suporte aos objetivos desta tese que serão apresentados nos próximos capítulos.

Com relação à definição do itinerários para agentes móveis com requisitos temporais, objeto de estudo desta tese, algumas abordagens foram estudadas com o propósito de servir como inspiração ao desenvolvimento de novas heurísticas (novos algoritmos) que supram algumas lacunas detectadas. Estas questões serão detalhadamente tratadas nos capítulos futuros.

O próximo capítulo apresenta o modelo computacional desenvolvido que descreve o comportamento de aplicações baseadas em agentes móveis imprecisos com restrição temporal. Nos Capítulos 6 e 7, serão apresentadas e avaliadas as heurísticas que auxiliam o agente móvel na tomada de decisão para a definição de seu itinerário.

5. Agentes Móveis de Tempo Real

5.1.Introdução

Este capítulo descreve o modelo computacional para agentes móveis de tempo real proposto nesta tese. Inicialmente, são discutidos alguns tópicos que justificam as escolhas feitas. Em seguida, na Seção 5.2, é apresentado o modelo computacional propriamente dito. Finalmente, a Seção 5.3 apresenta as considerações finais do capítulo.

Dentro do contexto da mobilidade de código, uma importante área de pesquisa é a tecnologia de agentes móveis (PICCO 1998, BAEK 2001). 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. Um agente móvel deve ser capaz de interromper sua execução em um lugar, transferindo-se para outro, e lá retomar sua execução. 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.

Por migrar para a localização do recurso desejado, um agente móvel pode responder a estímulos rapidamente e pode continuar sua interação com o recurso se a conexão de rede cair temporariamente. Estas características fazem com que os agentes móveis tornem-se atrativos para o desenvolvimento de aplicações móveis, as quais freqüentemente devem tratar com baixa largura de banda, alta latência e enlaces de rede não confiáveis.

Devido aos agentes processarem os dados localmente e somente transferirem os resultados para uma central, o tempo de resposta e a largura de banda da rede necessária podem ser menores quando comparados a um sistema centralizado. Sistemas que trabalham com agentes também possuem boa escalabilidade com relação à adição de novos recursos ao sistema.

A maioria das aplicações distribuídas encaixa-se naturalmente no modelo de agentes móveis, já que um agente móvel pode migrar seqüencialmente através de um conjunto de nodos, enviar agentes filhos para visitar nodos em paralelo, permanecer estacionário e interagir com recursos remotamente, ou qualquer combinação destas três habilidades.

Considerando os aspectos acima relacionados é possível ainda justificar a aplicação de agentes móveis em sistemas grandes e complexos, pois mesmo o custo para manutenção de todos os softwares e dados locais em todos os nodos do sistema distribuído pode ser impraticável.

Exemplos da utilização de agentes podem ser encontrados em sistemas de telefonia, sistemas de defesa, sistemas de energia elétrica, ambientes fabris compostos por sistemas grandes e complexos, e sistemas onde não é possível pré-estabelecer todos os processamentos.

No interesse desta tese, aliado às vantagens da mobilidade de código, um requisito importante é o da questão temporal. Como os agentes migram em um sistema distribuído, o planejamento do itinerário para se alcançar o nodo destino possui grande relevância na correção temporal da tarefa (cumprimento de *deadline*).

Um possível cenário que ilustra o uso de agentes móveis com restrição temporal é um sistema de geração e distribuição de energia elétrica onde, seguindo uma estrutura hierárquica, os dados encontram-se fisicamente espalhados. No contexto da indústria de energia elétrica, uma larga escala geográfica e respostas em tempo real são apenas dois requisitos na implementação da supervisão e controle do sistema, o qual também é caracterizado pela heterogeneidade dos equipamentos. A escala deste tipo de sistema pode ser de milhares de nodos (TOLBERT, 2001).

Considerando uma usina, onde acontece a geração da energia, e as subestações que participam da transmissão de energia elétrica, o agente móvel pode substituir o técnico (ou engenheiro) com seu *laptop* visitando seção por seção do sistema. O agente tem como função: coletar dados, formar uma imagem do problema e decidir a próxima visita para coleta de dados. Neste caso, portanto, alcança-se o diagnóstico de uma falha no sistema elétrico percorrendo vários nodos e coletando dados para a tomada de decisão.

Por exemplo, o trabalho (HIGASHI, 2003) visa explorar as potencialidades dos sistemas multiagentes, no auxílio da recomposição do sistema elétrico após a ocorrência de grandes perturbações na rede elétrica que conduzam a desligamentos totais ou parciais. Embora esse trabalho não empregue mobilidade, é um exemplo do uso de agentes em sistemas de geração de energia elétrica.

Outro exemplo do uso de agentes móveis em sistemas industriais distribuídos complexos é proposto por (WU, 2005), que apresenta uma plataforma de automação baseada em multi-agentes (MAbAP), no qual adota a tecnologia de agentes inteligentes em computadores em rede. Para uma planta industrial complexa, esquemas de controle centralizados podem ser difíceis de implementar devido ao número de dispositivos de aquisição de dados e controladores envolvidos, e também a dificuldade em se conectar todos estes itens de equipamentos. O MAbAP foi projetado para integrar as funções de gerenciamento de informação, monitoramento das condições e controle de tempo real. Sua arquitetura é baseada em sistemas multi-agentes e visa permitir a implementação de diversas tarefas, com alto nível de inteligência e coordenação flexíveis entre os componentes do sistema.

Existem na literatura alguns trabalhos que colocam agentes móveis no contexto de tempo real, como foi descrito no Capítulo 4. Entretanto, não é possível encontrar na literatura a formalização de um modelo computacional para agentes móveis que inclua as premissas adotadas neste trabalho com respeito a tempo real e ao comportamento do agente. Desta forma, a próxima seção descreve um modelo, sendo esta uma das contribuições desta tese.

5.2. Modelo Computacional

O modelo computacional proposto descreve o comportamento de uma aplicação baseada em agentes móveis imprecisos, com restrições temporais, 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 conjunto de nodos do sistema, os quais hospedam os recursos necessários para o agente. É suposto que este agente não se comunica com outro agente. 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 agente móvel impreciso é aquele capaz de reduzir a qualidade do resultado para conseguir atender ao *deadline* da missão.

Um recurso é uma abstração e pode corresponder a um processador, dispositivo, arquivo, estrutura de dados, etc. A cada recurso é atribuído um benefício associado a sua importância funcional dentro da missão. Em cada nodo do sistema existe um conjunto de recursos, e podem existir instâncias de um mesmo tipo de recurso em nodos distintos. O processador é um recurso sempre necessário para o agente e está presente em todos os nodos, sendo tratado de forma implícita.

Os recursos podem ser classificados como obrigatórios ou opcionais. Um recurso obrigatório é aquele que deve ser executado para a missão ser cumprida com sucesso. Um recurso é opcional se sua execução pode ou não acontecer durante a missão. A execução de um recurso opcional refina o resultado, até que ele alcance a qualidade desejada.

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:

- ser utilizados em qualquer ordem;
- aparecer replicados;
- ser descartados;
- ou ser parcialmente utilizados (com respeito ao seu tempo de utilização).

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 até m nodos Ni , Ni Î N, onde $1 = i = m$.

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 pertencente ao conjunto R pode ser encontrado em um ou mais nodos, isto é, podem existir várias instâncias (réplicas) de um mesmo tipo de recurso, desde que em nodos distintos. A Figura 5.1 ilustra uma situação onde é possível observar que instâncias do tipo de recurso $r3$ aparecem tanto no nodo N_2 como no nodo N_6.

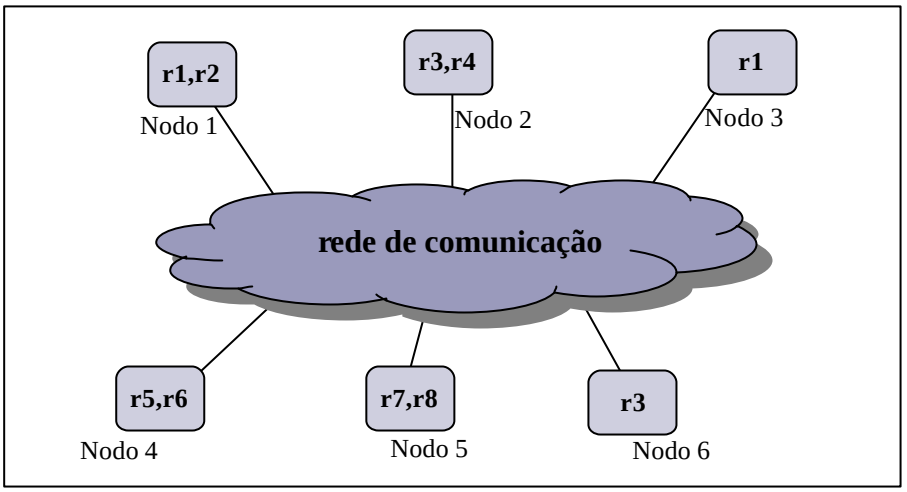


Figura 5.1: Recursos em nodos de um sistema distribuído.

Um itinerário ***I*** é definido como um caminho percorrido no sistema computacional, 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 da utilização 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 nenhuma instância dos tipos de recursos que formam o conjunto ***R***. É assumida uma sincronização de relógios no sistema; e toda missão ***M*** possui um *deadline firme* ***D*** associado. O agente ***Z*** deve concluir a missão (retornar ao nodo N_0) antes de ***D***, sob pena de perder o benefício coletado ao longo do itinerário.

O diagrama de recursos de uma missão ***M*** (Figura 5.2) corresponde a um diagrama de atividades UML (*Unified Modeling Language*) (ARLOW, 2005) que 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. A Figura 5.2 descreve uma missão com oito recursos utilizados, sendo o tipo de recurso ***r0*** um pseudo-recurso, encontrado somente no nodo N_0, usado para indicar ao agente a volta

ao nodo original. No diagrama, o esteriótipo <<*variable*>> indica que o tempo de uso do recurso ***r1*** é variável, podendo este ser liberado antes de seu benefício máximo.

As flexibilidades na composição do itinerário estão relacionadas a alguns dos recursos pertencentes ao conjunto ***R,*** que podem ser utilizados sem uma ordem pré-definida (ex. ***r5*** e ***r3*** na Figura 5.2), aparecer replicados (ex. ***r1*** em N_1 e N_3, na Figura 5.1), ser descartados (ex. ***r7*** na Figura 5.2) ou ser parcialmente utilizados (ex. ***r1***, na Figura 5.2).

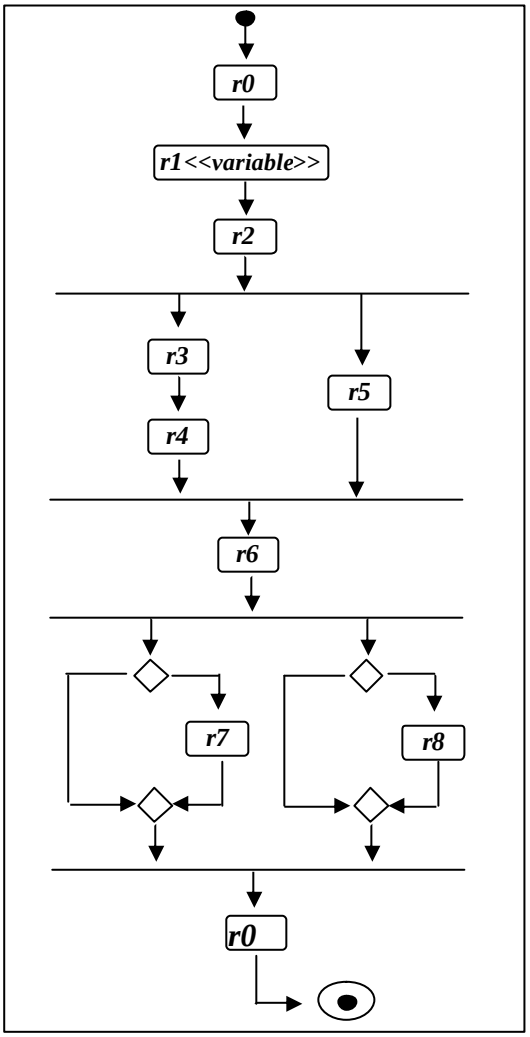


Figura 5.2: Diagrama de Recursos da Missão ***M.***

O diagrama de nodos de uma missão ***M*** (Figura 5.3) corresponde a um diagrama de atividades UML (*Unified Modeling Language*) (ARLOW, 2005) 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, supondo que a missão descrita na Figura 5.2 seja executada no

sistema descrito na Figura 5.1, a Figura 5.3 contém o diagrama de nodos da missão resultante, e representa a possibilidade de itinerários diferentes que o agente pode seguir para cumprir sua missão.

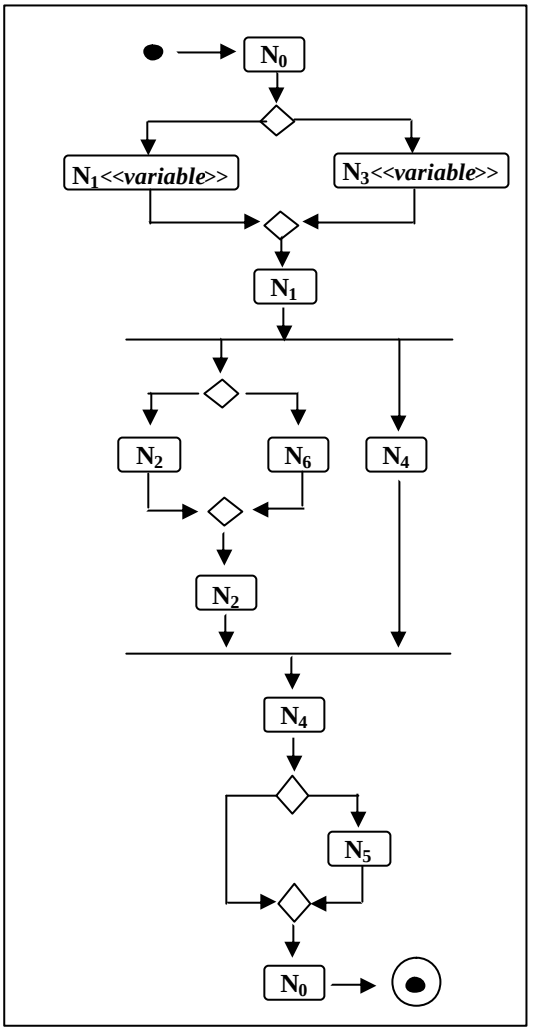


Figura 5.3: Diagrama de Nodos da Missão ***M***

A execução da missão ***M*** pode ser vista como uma atividade aperiódica ***A***. Considera-se uma atividade o conjunto de tarefas ***X*** e tempo de rede ***L*** que compõem um itinerário ***I*** em particular. Uma tarefa x_i representa o uso de um recurso r_i por um agente ***Z*** em um dado nodo N_i. Cada tarefa x_i pertence ao conjunto ***X***. As ativações dessas tarefas ($x_1, x_2, ..., x_n$, onde ***n*** é a cardinalidade do conjunto ***X***) ocorrem em intervalos irregulares de tempo, caracterizando o aspecto aperiódico da atividade e das tarefas. O *deadline* ***D*** da atividade ***A*** está associado com a conclusão de todas as tarefas pertencentes à atividade ***A***.

O cumprimento de uma missão ***M*** está relacionado à escolha de um itinerário ***I***. Para definição do itinerário são considerados:

- O tempo de computação para benefício máximo de cada tarefa x_i, denominado C_i;
- A latência de comunicação entre os nodos N_k e N_y, denominada L_{ky};
- 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 ($C_i + Q_i$ onde: C_i é o tempo de execução no nodo e Q_i é o tempo na fila do processador local esperando para executar).

Para todo r_i Î 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*

$$B_{i,t} = B_i * min (1, t/Ci) \qquad [1]$$

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

$$B_{i,t} = B_i \quad \textbf{se} \quad t = C_i \qquad [2]$$

$$B_{i,t} = 0 \quad \textbf{se} \quad t < C_i$$

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

- se r_i precede r_j, e r_i não foi utilizado, então $B_j = 0$;
- se r_i' e r_i" são réplicas de r_i, e r_i' já foi utilizado, então o benefício máximo de r_i" passa a ser zero.

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

$$B = \sum B_i(y) \quad \text{para todo} \quad r_i \ ? \ R. \qquad [3]$$

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.

O problema considerado neste trabalho é semelhante ao apresentado em (BALAS, 1989), chamado "*The Prize Collecting Traveling Salesman Problem*". No entanto, algoritmos naquele trabalho são computacionalmente pesados e voltados para determinação estática do itinerário, quando não existem requisitos de tempo real envolvidos.

Nesta tese é suposto que o diagrama de recursos vai sendo conhecido pelo agente móvel ao longo do caminho, em função dos dados que vão sendo 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 capítulo não pode ser resolvido utilizando técnicas clássicas de otimização, pois na origem o diagrama de recursos não é conhecido; e durante o percurso os nodos visitados podem possuir 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, 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 não é considerado no modelo computacional.

Com o objetivo de exemplificar a viagem do agente pelo sistema distribuído foi criado um grafo composto por 51 vértices e 60 arestas. Este grafo foi desenvolvido a partir do diagrama de nodos (Figura 5.3) que apresenta os possíveis itinerários da missão descrita pelo diagrama de recursos (Figura 5.2).

Vértices representam um estado no comportamento do agente (por exemplo, chegada no nodo, execução do recurso, transmissão via rede) e arestas representam as relações de precedência. Cada vértice que representa a execução de um recurso ***R*** em um nodo ***N*** é marcado como ***R/N***; um vértice que representa apenas a chegada em um nodo (sem a execução de um recurso) é marcado como *bypass*; e um vértice que representa o custo da transmissão via rede é marcado como *net*. O grafo da Figura 5.4, ilustra possíveis itinerários a serem percorridos pelo agente móvel que, baseado no próximo nodo (vértice), fará sua tomada de decisão fundamentada nos critérios do algoritmo utilizado para definição do itinerário.

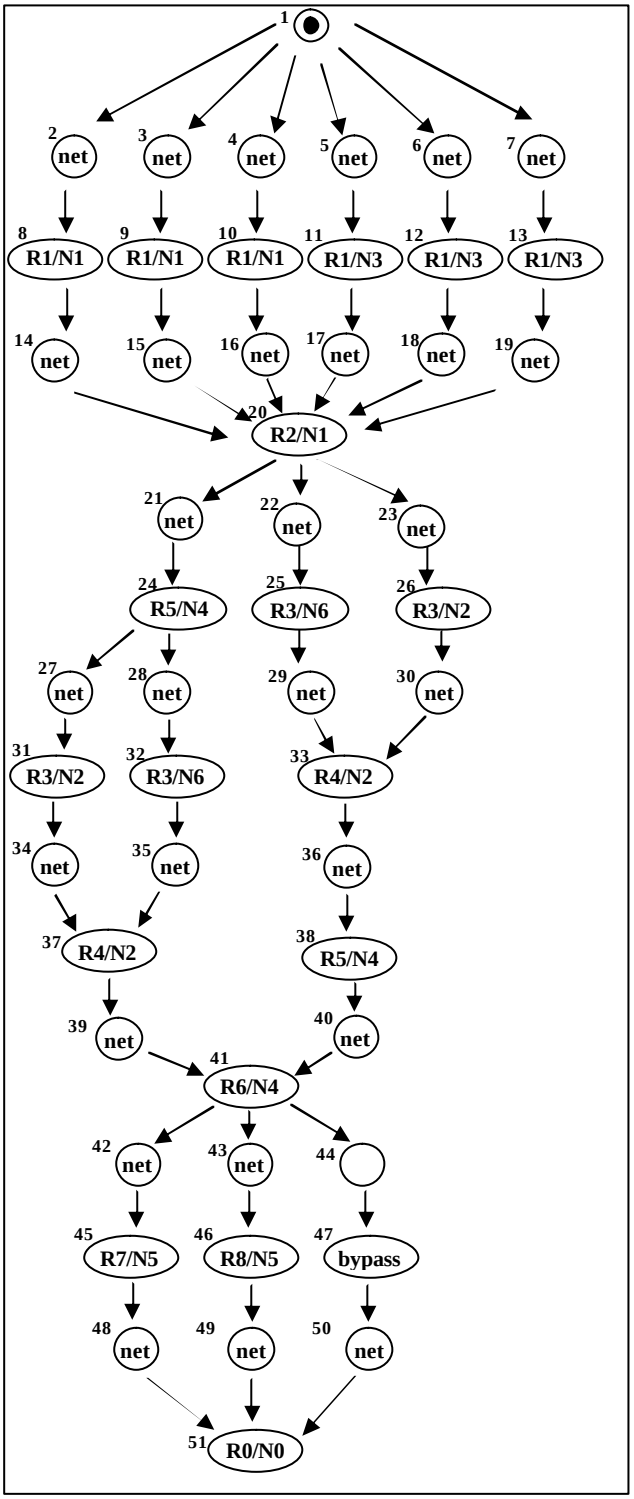


Figura 5.4: Grafo para a aplicação exemplo.

5.3. Conclusões

Neste capítulo foi apresentado o modelo computacional desenvolvido para aplicações baseadas em agentes móveis imprecisos com restrição temporal. Este modelo descreve um sistema distribuído formado por um conjunto de nodos conectados através de um serviço subjacente de comunicação.

Neste trabalho é suposto que um diagrama de recursos indica as relações de precedência e as opcionalidades 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, 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 é definida dinamicamente.

A proposição de um modelo computacional original para agentes móveis de tempo real, o qual inclui *deadline* para a missão, miopia do agente, níveis de precisão e diagramas de recursos é uma contribuição desta tese.

Por tratar-se de um novo modelo computacional, não existe na literatura soluções apropriadas já descritas para a definição do itinerário do agente móvel.

Os Capítulos 6 e 7 apresentarão as heurísticas que foram criadas, neste trabalho de doutorado, para a definição do itinerário dos agentes móveis imprecisos neste modelo computacional.

6. Descrição e Avaliação de Heurísticas Simples e Baseadas em Clones

6.1 Introdução

Neste capítulo são propostas heurísticas que foram desenvolvidas no decorrer deste doutorado para serem utilizadas na definição do itinerário conforme o modelo computacional descrito no Capítulo 5. Neste modelo computacional para agentes móveis imprecisos com restrições temporais, cada agente possui certa flexibilidade na definição de seu itinerário. Esta flexibilidade está relacionada com características dos recursos (descritas no Capítulo 5). Cada heurística confere ao agente um comportamento distinto que, baseado nas diferentes características de cada recurso, é utilizado pelo agente móvel na definição do itinerário. Essas heurísticas podem ser utilizadas individualmente ou em pares/trios (através do uso de clones).

As heurísticas, embora simples, são originais pela forma como são aplicadas ao problema descrito no capítulo anterior, o qual também é original. A utilização de clonagem de agentes móveis é prática bem conhecida. Porém, nesta tese a clonagem é utilizada em um novo cenário específico, ou seja, a definição do itinerário de agentes móveis míopes de tempo real. As heurísticas apresentadas neste capítulo são uma das contribuições desta tese.

O restante do capítulo está organizado da seguinte forma: na Seção 6.2 são descritas as características individuais de cada heurística proposta. A Seção 6.3 apresenta as condições da simulação, a maneira como foram feitas as simulações que avaliam o comportamento das heurísticas individualmente. Na Seção 6.4 são apresentados os resultados destas simulações. Na Seção 6.5 o método utilizando clones é descrito, e na Seção 6.6, a avaliação deste método é apresentada. Na Seção 6.7 são comparados os resultados apresentados nas Seções 6.4 e 6.6. Finalmente, na Seção 6.8, são feitos os comentários finais sobre o comportamento das heurísticas simples (individuais ou em clones).

6.2 Descrição das Heurísticas

A tomada de decisão utilizada pelas heurísticas simples (descritas a seguir) 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). Em função da miopia do agente móvel, não é possível considerar os desdobramentos futuros das decisões de curto prazo. Sempre que o agente móvel precisa decidir entre dois caminhos que, na perspectiva da heurística usada, são equivalentes, é tomada uma decisão aleatória.

Em função da informação obtida através do último recurso recém-usado, o código de aplicação do agente identifica quais seriam os possíveis próximos recursos a serem visitados. Essa lista de destinos candidatos para o agente deve incluir as réplicas dos recursos candidatos, se essas existirem, e também a possibilidade de não executar nenhum deles, caso todos sejam opcionais. Caso a instância de recurso selecionada para ser a próxima a ser usada esteja em um nodo da rede diferente daquele onde o agente encontra-se, isto implicará no deslocamento do agente até aquele nodo. Seja qual for a decisão do agente, ela deve estar de acordo com o Diagrama de Nodos da missão, o qual define as possibilidades de recursos a serem utilizados.

6.2.1. Algoritmo Preguiçoso (*Lazy Algorithm*)

O Algoritmo Preguiçoso procura executar os recursos o mais rapidamente possível, ignorando o benefício de cada recurso. Os recursos opcionais não são executados, e recursos com estereótipo <<*variable*>> são executados com o menor tempo possível (C=0). Sempre que existir uma alternativa, será escolhida aquela que possuir o menor tempo de uso, o qual inclui os tempos de comunicação e computação. Assim, o benefício obtido é mínimo, podendo ser zero. Uma descrição abreviada do algoritmo é apresentada na Figura 6.1.

```
Dado um conjunto S de ações imediatas possíveis
Para cada ação imediata possível Si ? S
   estima tempo total ti para executar Si, Si ? S
Seleciona Si com o menor ti
```

Figura 6.1: Algoritmo Preguiçoso.

A complexidade computacional deste algoritmo é ***O(x)***, onde ***x*** representa o número de opções que o agente móvel dispõe para a escolha de seu próximo movimento.

6.2.2. Algoritmo Guloso (*Greedy Algorithm*)

Sempre que existir uma alternativa, o Algoritmo Guloso escolherá aquela que possuir o maior benefício, isto é, quando não existir relação de precedência entre dois recursos 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*>>, o algoritmo guloso executa todo o tempo solicitado para obter o benefício máximo. A Figura 6.2 apresenta a tomada de decisão desta heurística.

```
Dado um conjunto S de ações imediatas possíveis
Para cada ação imediata possível Si ? S
   estima benefício bi da execução de Si, Si ? S
Seleciona Si com o maior bi
```

Figura 6.2: Algoritmo Guloso.

Novamente, a complexidade computacional é ***O(x)***, onde ***x*** representa o número de opções que o agente móvel dispõe para a escolha de seu próximo movimento.

6.2.3. Algoritmo Aleatório (*Random Algorithm*)

Este algoritmo escolhe aleatoriamente o próximo nodo a ser visitado pelo agente (Figura 6.3). 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*>>, o algoritmo *aleatório* decidirá aleatoriamente valores entre 0 e o tempo de computação necessário para alcançar o benefício máximo para este recurso. A complexidade computacional deste algoritmo é ***O(1)***.

```
Dado um conjunto S de ações imediatas possíveis
Seleciona Si aleatoriamente, Si ? S
```

Figura 6.3: Algoritmo Aleatório.

6.2.4. Algoritmo Ponderado (*Higher Density Algorithm*)

O Algoritmo Ponderado – 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 de um recurso ***i*** é obtida pela relação:

$$db_i = B_i / (C_i + L_{j,i} + Q_i) \qquad [1]$$

onde: 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 valores 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. A Figura 6.4 apresenta as tomadas de decisão do Algoritmo Ponderado. Novamente, a complexidade computacional do algoritmo é ***O(x)***, pois cada uma das ***x*** opções que o agente móvel dispõe precisam ser consideradas.

```
Atualiza db_medio
Dado um conjunto S de ações imediatas possíveis
Para cada ação imediata possível Si, Si ? S
   estima tempo total ti para executar Si
   estima benefício bi da execução de Si
   calcula a densidade do benefício dbi da execução Si, dbi = bi/ti
se existe a opção x de não fazer nada
   então  dbx = db_medio
Seleciona Si com a melhor relação dbi
```

Figura 6.4: Algoritmo Ponderado.

A Figura 6.5. ilustra o comportamento de um agente móvel (tomada de decisão) utilizando as heurísticas acima descritas.

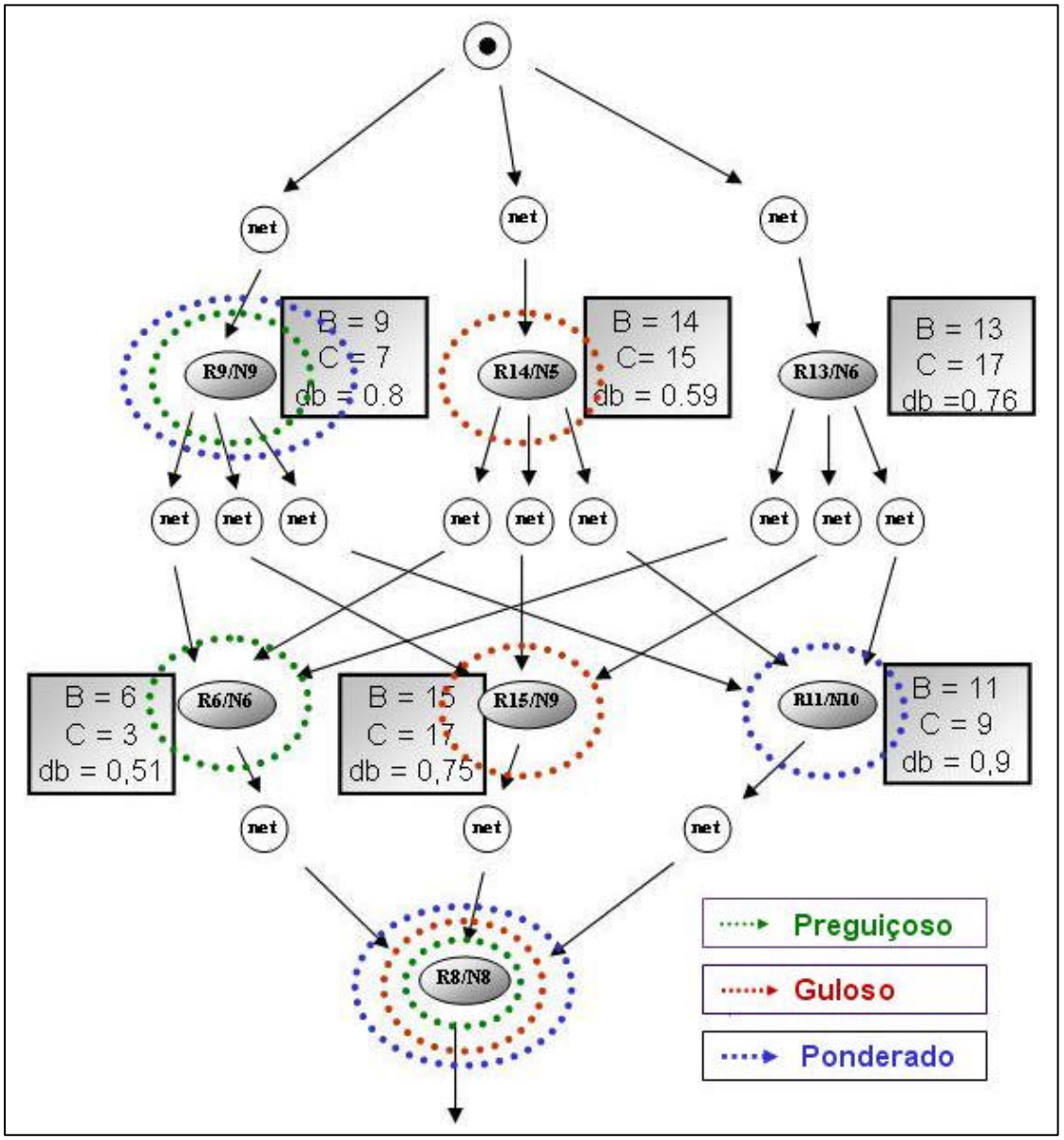


Figura 6.5: Exemplo da tomada de decisão do agente móvel conforme o algoritmo utilizado.

6.2.5. Versões com Relógio

Uma variação possível nos algoritmos apresentados é, após cada recurso ser executado, estimar o tempo disponível para executar os recurso restantes da missão. A folga existente é calculada pela diferença entre o *deadline* absoluto e o tempo atual, dividida pelo *deadline* relativo. Está é uma medida relativa de quanto tempo o agente móvel ainda dispõe.

As versões com relógio dos algoritmos são probabilistas. No início de suas execuções seguem seu comportamento original; à medida que o *deadline* da missão se aproxima, estes algoritmos ganham características semelhantes às 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 seu consumo de tempo. Para estas versões das heurísticas é necessária uma sincronização de relógios.

Os algoritmos que apresentam esta variação são: **Algoritmo Guloso com Relógio** e **Algoritmo Ponderado com Relógio**. A Figura 6.6 descreve o procedimento da escolha do próximo recurso a ser executado. A complexidade computacional não é alterada.

```
Dado um conjunto S de ações imediatas possíveis
Para cada ação imediata possível Si
    calcula percentual de tempo relativo até o deadline, Fi
    Fi = (deadline absoluto – tempo atual) / deadline relativo
    sorteia x a partir de uma distribuição uniforme entre 0 e 1
    se x > Fi
       meu_comportamento = Preguiçoso
    senão
       meu_comportamento = original (Guloso ou Ponderado)
```

Figura 6.6: Versões com relógio.

6.3. Simulação das Heurísticas Simples

Com o objetivo de avaliar o desempenho das heurísticas simples, foram realizadas simulações com uma grande quantidade de diferentes cenários. Na simulação foi utilizado o modelo computacional descrito no Capítulo 5, 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, uma parte deste total

foi utilizada, como será descrito posteriormente. O Anexo A apresenta os Diagramas de Recursos, de Nodos e os grafos dos 10 segmentos que podem formar a missão do agente.

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 uma 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;
- O tempo de comunicação necessário para transferir um agente móvel entre dois nodos quaisquer, fixo para todos os enlaces é igual a 1.

Espera-se que apenas alguns recursos sejam do tipo variável, desta forma iremos tratar esta questão através da discretização do recurso variável, criando 3 caminhos alternativos para fins de avaliação: execução completa, execução mediana e sem execução.

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 seguindo a seguinte equação:

$$G = ? \ (B\_DA / NT) \qquad [2]$$

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

Os recursos ***r1***, ***r2***, ***r3***, ***r4***, ***r5***, ***r6***, ***r7***, ***r8***, ***r9*** e ***r10*** são simples, isto é, são encontrados em um único nodo no sistema. Os recursos ***r11***, ***r12***, ***r13***, ***r14*** e ***r15*** são replicados: existem instâncias destes recursos em dois ou mais nodos no sistema distribuído. A disposição dos recursos nos nodos é a seguinte: ***r1*** em {***N1***}; ***r2*** em {***N2***}; ***r3***

em {***N3***}; ***r4*** em {***N4***}; ***r5*** em {***N5***}; ***r6*** em {***N6***}***; r7*** em {***N7***}***; r8*** em {***N8***}; ***r9*** em {***N9***}; ***r10*** em {***N10***}; ***r11*** em {***N1,N2***}; ***r12*** em {***N3, N4***}; ***r13*** em {***N5,N6***}; ***r14*** em {***N7, N8***}; ***r15*** em {***N9, N10***}.

6.4. Avaliação das Heurísticas Simples

Nesta seção será apresentada a avaliação – por meio de simulações – do modelo computacional proposto no Capítulo 5 e das heurísticas simples descritas na seção 6.2. Com o objetivo de facilitar a comparação do comportamento destas heurísticas, a avaliação será realizada em diferentes fases (descritas individualmente a seguir). O simulador utilizado foi desenvolvido em Java, e é utilizado pelo grupo de pesquisa. Este simulador mantém uma fila de eventos ordenada pelo tempo, estrutura clássica de um simulador de eventos discretos. O tempo simulado sempre avança até o instante do próximo evento de interesse o qual é então processado.

O objetivo desta avaliação em fases é observar e descrever distintamente o comportamento das heurísticas pelo intermédio de sucessivas simulações com diferentes configurações envolvendo diferentes situações.

6.4.1. Primeira Fase da Avaliação

Nesta primeira fase de avaliação, foram sorteadas uma missão e uma carga. Para esta missão foram lançados 100 agentes, o que significa que cada simulação foi repetida 100 vezes e os resultados apresentados são formados pela média dos valores obtidos. A carga do sistema distribuído inclui as condições da rede, a ocupação dos nodos – que reflete num maior ou menor tempo de fila – e os valores atribuídos a cada recurso pertencente a missão (o tempo de computação e o benefício adquirido).

A Tabela 6.1 indica os valores dos benefícios médios alcançados pelos algoritmos utilizando a mesma missão, a mesma carga e variados *deadline*s (apertados, justos e folgados). Por meio da observação destes valores foi possível descrever o comportamento de cada heurística para diferentes situações de restrição temporal e assim descobrir qual delas apresenta melhor comportamento para uma determinada situação.

Pode-se considerar um *deadline* apertado como aquele que tem grande possibilidade de ser perdido; *deadline* folgado, por outro lado, é aquele que tem grande possibilidade de ser alcançado. *Deadline*s justos são aqueles que se encontram em uma faixa intermediária de valores entre *deadline*s apertados e folgados (a Seção 6.4.3 apresenta uma definição mais detalhada desses tipos de *deadline*s).

A Figura 6.7, baseada nos dados descritos na Tabela 6.1, apresenta o gráfico do benefício global ***G*** obtido por cada algoritmo. Analisando este gráfico, percebe-se que a partir de um determinado momento (ponto em que todos os *deadline*s são atendidos) o benefício global obtido pelos algoritmos permanece estável, independente do tempo restante entre o *deadline* e o tempo gasto com a missão.

Os valores em negrito na Tabela 6.1 representam os maiores benefícios alcançados pelas heurísticas (com variação de 1%) para cada *deadline*. Estes valores formam o envelope superior do gráfico apresentado na Figura 6.1.

Tabela 6.1: Benefício Global das heurísticas simples.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	3,5	20,6	57,4	123,7	156,5	174,5	177,8	179,6	182,9	182,1
Preguiçoso	0	0	0	**1,7**	**8,3**	**31,4**	77,6	132	155,1	161,7	165	165	165	165
Ponderado_R	0	0	0	0	7,2	25,1	**95,3**	**142,8**	**173**	177,7	181,3	180,8	182,7	182,3
Ponderado	0	0	0	0	3,8	24,4	**95,9**	**142,9**	**172,9**	178,6	188	188	188	188
Guloso_Rel	0	0	0	0	1,8	18,5	72,9	120,6	145,6	**180**	190,1	187,9	191,8	192,7
Guloso	0	0	0	0	0	10,3	18,5	74,2	138	**181,3**	**203,9**	**201,9**	**206**	**206**

Com os *deadline*s apertados (por exemplo, 300) o Algoritmo Preguiçoso apresentou o melhor desempenho, a medida que o *deadline* vai se tornando menos apertado essa vantagem diminui (ver Tabela 6.1, os *deadline*s 350 e 400). No momento em que o *deadline* da missão é considerado justo, o melhor desempenho é alcançado pelo Algoritmo Ponderado (do *deadline* 450 ao 550) em suas versões com e sem relógio. É importante perceber que enquanto o *deadline* é apertado (por exemplo, 350) o Algoritmo Ponderado com Relógio alcança um benefício superior à sua versão sem relógio. Este resultado sugere que a utilização de relógio é vantajosa em situações nas quais o tempo disponível para se executar os recursos que formam a missão é limitado. O mesmo pode ser observado com o Algoritmo Guloso com Relógio quando o *deadline* é justo ou apertado. Se compararmos apenas estas duas heurísticas (Algoritmo Guloso e Algoritmo Guloso com Relógio)

percebe-se que a versão sem relógio encontra os melhores resultados somente quando o *deadline* é folgado.

Quando os *deadline*s são folgados (neste caso, a partir de 600) o Algoritmo Guloso, seguido de sua versão com relógio, obtiveram os melhores índices de benefício. O Algoritmo Aleatório não obteve em nenhuma circunstância o benefício mais alto.

Os algoritmos Preguiçoso e Guloso representam os dois extremos, sendo o Algoritmo Guloso o melhor para *deadline*s folgados e o Algoritmo Preguiçoso o melhor para *deadline*s apertados. Entretanto, no modelo computacional 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 aos algoritmos Guloso e Ponderado resultou em uma melhoria significativa em situações com menor tempo disponível para o cumprimento da missão.

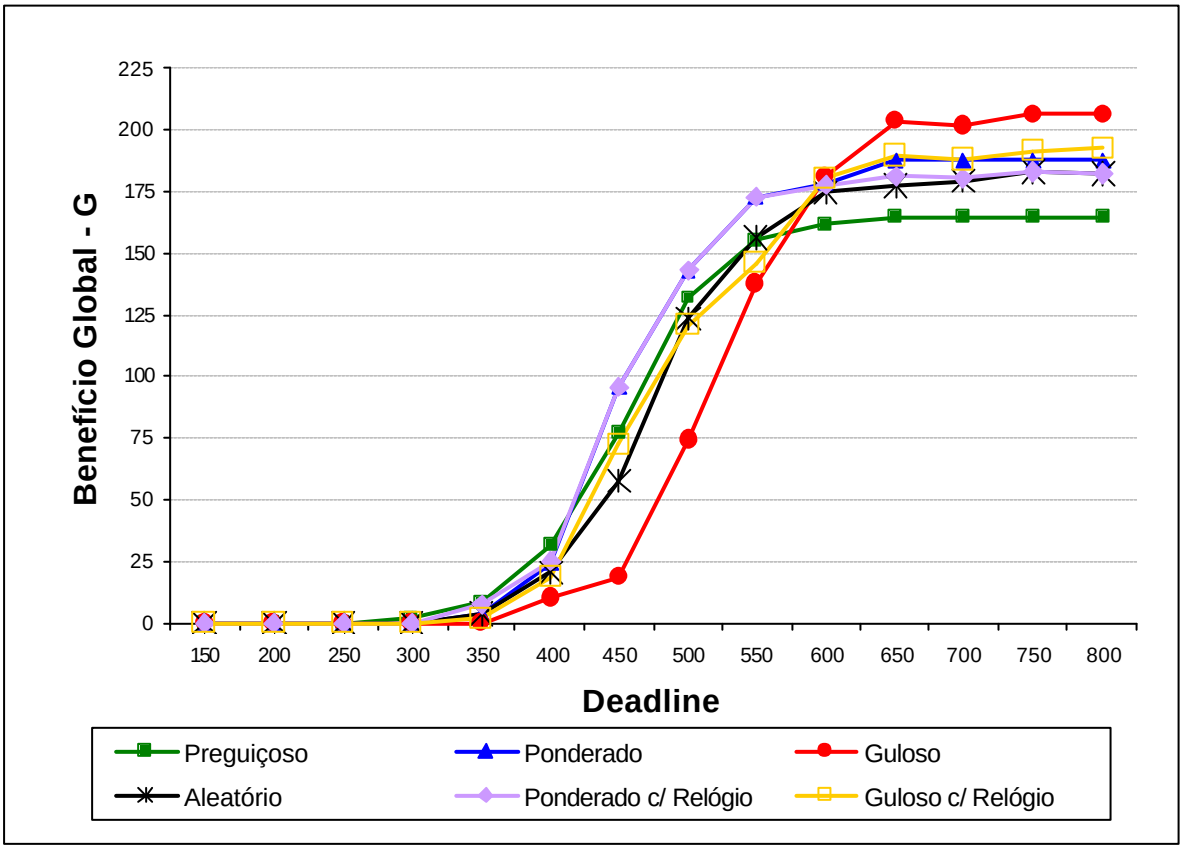


Figura 6.7: Benefício Global das heurísticas simples.

Os números apresentados na Tabela 6.2 definem um intervalo de valores para o benefício médio de cada heurística, com uma confiança de 95%. Isto significa que existe

uma confiança de 95% de que o benefício médio de cada heurística esteja dentro do intervalo (para as condições de simulação usadas):

$$\boldsymbol{Bm} \pm \boldsymbol{IC} \qquad \textbf{[3]}$$

onde: ***Bm*** = benefício médio da simulação e ***IC*** = valor do intervalo de confiança.

Com *deadline*s apertados ou folgados, os valores dos intervalos de confiança tendem a zero, isto acontece porque a maioria dos agentes tende a perder ou a ter seus *deadline*s atendidos, respectivamente. Algoritmos em suas versões com relógio e o Algoritmo Aleatório apresentam variação no seu crescimento do intervalo de confiança. Diferentemente, para os algoritmos Guloso, Preguiçoso e Ponderado, a medida que recebem um *deadline* cada vez mais folgado, o intervalo de confiança tende a zero. Por exemplo, quando o *deadline* é folgado (D ≥ 750), o Algoritmo Guloso sempre obtém o mesmo valor em todas as 100 execuções, por isso o intervalo ficou zerado. Quando o *deadline* é mais justo, existe maior variação no comportamento do Algoritmo Guloso, por isso o intervalo é maior, por exemplo, com ***D=450*** o valor de ***IC*** é igual a ***11,6***. Isto significa que o benefício médio da heurística estará dentro do intervalo: [6,9; 30,1]. Para encontrarmos estes valores utilizamos o ***Bm=18,5*** (ver Tabela 6.1).

Quanto maior o intervalo, maior a variação no comportamento da heurística. Todos os intervalos ficaram pequenos o bastante para simplesmente serem comparadas às médias.

Tabela 6.2: Intervalo de confiança dos benefícios medidos.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	± 4,8	± 11	±16,5	±16,4	±12,1	± 6,3	± 5,4	± 4	± 1,9	± 2
Preguiçoso	0	0	0	± 3,2	± 7,1	±12,8	±16,2	± 13	± 7,7	± 4,5	0	0	0	0
Ponderado_R	0	0	0	0	± 7	±12,3	±17,7	±14,5	± 7	± 5,1	± 1,1	± 3,7	± 0,9	± 1
Ponderado	0	0	0	0	± 5,2	±12,5	±18,5	±15,8	±10,1	± 8,1	0	0	0	0
Guloso_Rel	0	0	0	0	± 3,6	± 11	± 18	±17,9	±15,7	±8,3	± 1,6	± 5,4	± 1,7	± 1,5
Guloso	0	0	0	0	0	±8,8	±11,6	±19,5	±19,1	±13,2	± 4	± 5,7	0	0

A Tabela 6.3 e a Figura 6.8 apresentam o benefício médio alcançado pelos agentes com *deadline*s atendidos, isto é, o valor de quem atendeu os requisitos temporais desconsiderando os não atendidos. Estes valores representam o benefício médio das missões cumpridas.

Tabela 6.3: Benefício médio das missões cumpridas.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	174	171,4	179,3	179,3	179,9	179,9	181,4	181,4	183	182,1
Preguiçoso	0	0	0	165	165	165	165	165	165	165	165	165	165	165
Ponderado_R	0	0	0	0	180,8	179,5	179,9	180,8	180,3	181,3	181,3	182,6	182,7	182,4
Ponderado	0	0	0	0	188	188	188	188	188	188	188	188	188	188
Guloso_R	0	0	0	0	183	185,7	186,9	188,5	189,1	189,5	190,1	191,8	191,8	192,7
Guloso	0	0	0	0	0	206	206	206	206	206	206	206	206	206

Observando o gráfico da Figura 6.7 percebe-se que o benefício individual dos *deadline*s atendidos é constante para algoritmos sem relógio (com exceção do Algoritmo Aleatório). Algoritmos que utilizam relógio apresentaram um pequeno acréscimo no benefício obtido a medida que o *deadline* aumenta. Para a maior faixa de valores de *deadline*s, o Algoritmo Guloso apresentou o maior índice de benefício e o Algoritmo Preguiçoso apresentou o menor índice de benefício.

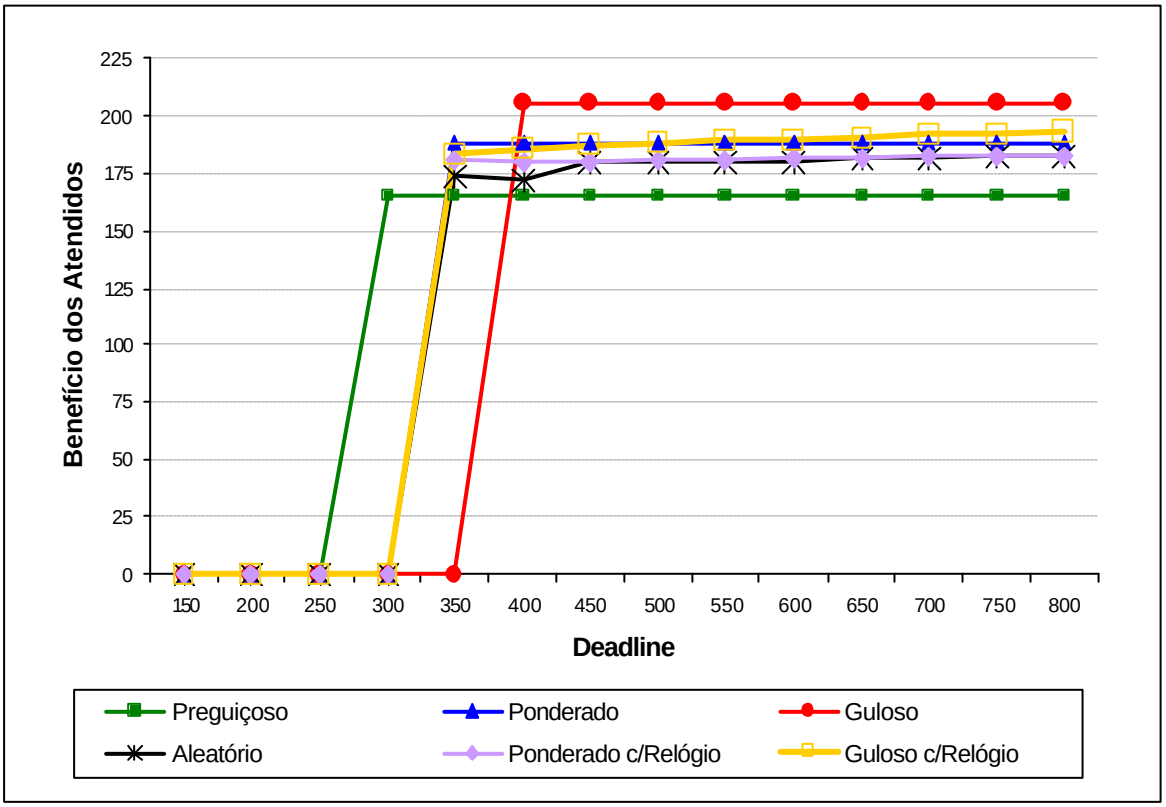


Figura 6.8: Benefício médio das missões atendidas.

O Algoritmo Preguiçoso foi a primeira heurística a conseguir completar a missão com pelo menos um agente (D=300). O Algoritmo Guloso exigiu o maior *deadline* para cumprir a missão com pelo menos um agente (D=400).

A Tabela 6.4 e a Figura 6.9 expõem os índices que indicam o percentual de agentes que obtiveram sucesso, referentes a cada *deadline*. Os algoritmos Aleatório e Guloso foram os que necessitaram de maiores *deadline*s para atingir 100% de atendidos.

Inicialmente, o Algoritmo Preguiçoso apresentou os maiores índices de atendimento, os algoritmos Ponderado e Ponderado com Relógio também obtiveram bons índices com *deadline*s justos, enquanto que o Algoritmo Guloso atingiu índices de atendimento mais baixos.

Tabela 6.4: Taxa média de *deadline*s atendidos das heurísticas simples.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	0,02	0,12	0,32	0,69	0,87	0,97	0,98	0,99	1	1
Preguiçoso	0	0	0	0,01	0,05	0,19	0,47	0,80	0,94	0,98	1	1	1	1
Ponderado_R	0	0	0	0	0,04	0,14	0,53	0,79	0,96	0,98	1	0,99	1	1
Ponderado	0	0	0	0	0,02	0,13	0,51	0,76	0,92	0,95	1	1	1	1
Guloso_R	0	0	0	0	0,01	0,1	0,39	0,64	0,77	0,95	1	0,98	1	1
Guloso	0	0	0	0	0	0,05	0,09	0,36	0,67	0,88	0,99	0,98	1	1

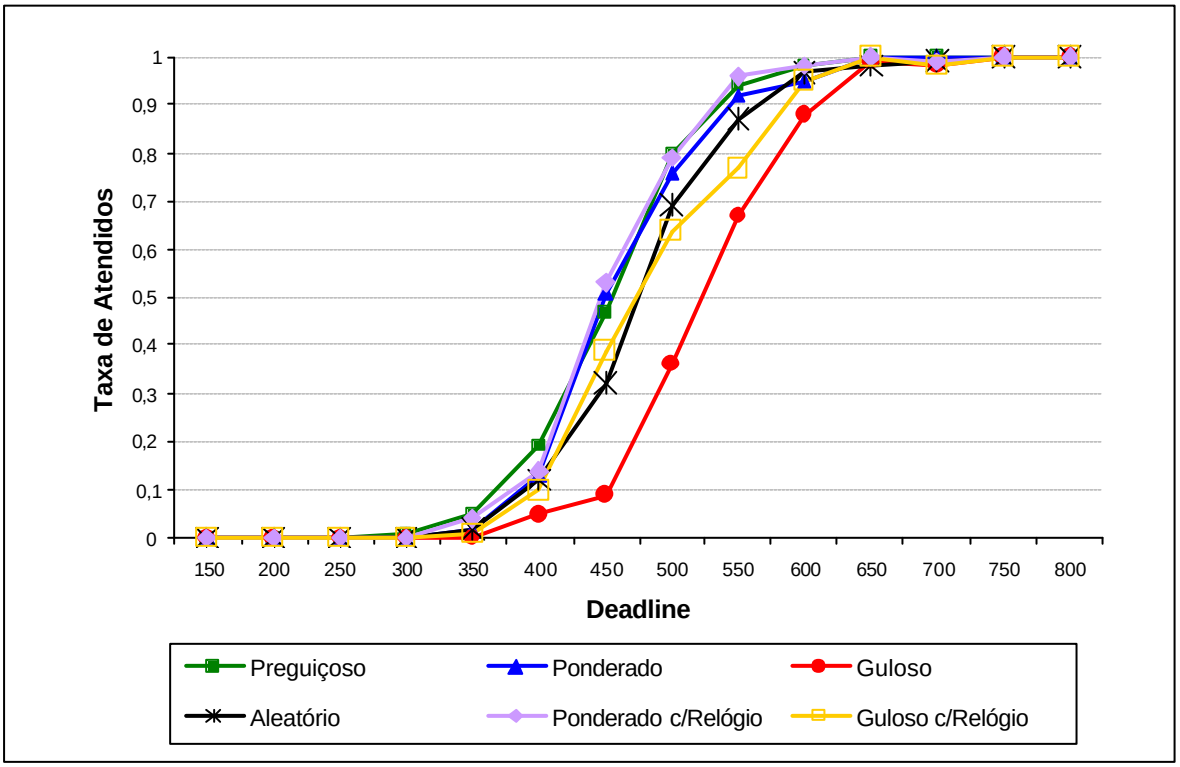


Figura 6.9: Taxa média de *deadline*s atendidos das heurísticas simples.

Versões com relógio podem conseguir 100% de *deadline*s atendidos com um determinado *deadline* e em outras tentativas não alcançarem este valor com *deadline*s maiores (ver na Tabela 6.4, D=650 e D=700), esta oscilação é característica destas versões.

Isto se deve ao fato das versões com relógio embutirem um elemento de aleatoriedade em seus comportamentos o qual aparece nos resultados das simulações.

A Figura 6.10 exibe o gráfico com os tempos médios de resposta utilizados pelas heurísticas. Analisando-se o gráfico, nota-se 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 movem-se alternadamente dentro de um intervalo de tempo com valores mais baixos. A Tabela 6.5 apresenta estes valores.

Tabela 6.5: Tempo médio de resposta das heurísticas simples.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	469,7	478,1	479,1	473	471,8	483,6	482,9	479,7	474,7	466,0	485,8	480,3	478,6	480,2
Preguiçoso	450,7	451,3	446,1	450,4	456	448	460,6	443,6	440,4	459,5	451,7	455,0	440,4	440
Ponderado_R	445,7	446,8	462,1	448,2	444,9	454	446,2	455,8	454,0	456,8	460,0	452,8	446,7	451,8
Ponderado	461,8	458,8	454,6	457,3	462,3	461,6	447,8	461,5	461,8	465,1	458,8	460,2	470,6	454,6
Guloso_R	454,6	467	461,9	466,4	487,2	480,6	470,7	481,6	487,8	491,3	482,7	497,4	495,1	488,3
Guloso	518,1	529,4	522,9	533,1	523,2	531,1	527,7	533,5	520,2	522,9	515,2	527,5	533,4	530,7

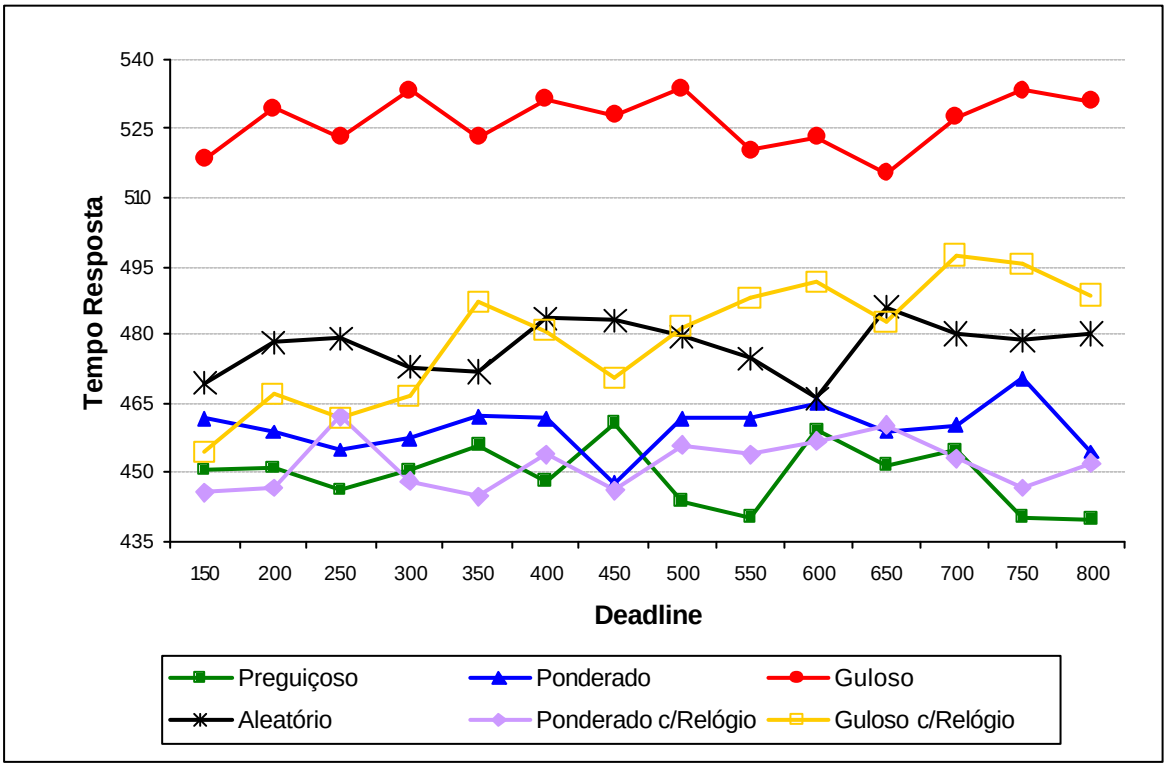


Figura 6.10: Tempo médio de resposta das heurísticas simples.

6.4.2. Segunda Fase da Avaliação

Diferentemente da primeira fase de avaliação, nesta etapa foram utilizados 7 diferentes grafos de recursos e nodos (onde os recursos aparecem em diferentes ordens). O objetivo desta fase de testes é verificar se o comportamento das heurísticas permanece similar mesmo quando os agentes executam diferentes missões.

Nesta segunda fase foram realizados 7 sorteios de segmentos de missão para uma mesma carga e foram lançados 100 agentes para cada missão. A Tabela 6.6 apresenta os benefícios alcançados pelos algoritmos, utilizando a mesma carga e variados *deadline*s. Os valores em negrito indicam as heurísticas que obtiveram os melhores índices para cada *deadline* (com uma variação de 1%).

Tabela 6.6: Benefício Global das 7 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Configuração I														
Aleatório	0	0	0	1	26	65	145	165	171	175	174	176	173	173
Preguiçoso	0	0	0	**32**	**131**	141	146	147	147	147	147	147	147	147
Ponderado_R	0	0	0	26	124	**149**	155	158	158	158	159	160	160	161
Ponderado	0	0	0	3	66	**150**	**168**	170	170	170	170	170	170	170
Guloso_R	0	0	0	0	33	97	161	169	178	177	182	182	183	185
Guloso	0	0	0	0	0	0	67	**175**	**192**	**204**	**204**	**204**	**204**	**204**
Configuração II														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	**28**	30	117	142	143	143	143	143	144	144	143	144
Preguiçoso	0	0	37	122	144	144	144	144	144	144	144	144	144	144
Ponderado_R	0	0	17	37	137	**161**	163	**163**	**163**	**163**	**163**	**163**	**163**	**163**
Ponderado	0	0	6	117	150	158	158	158	158	158	158	158	158	158
Guloso_R	0	0	3	73	**152**	157	**168**	161	160	161	**162**	161	161	161
Guloso	0	0	14	**129**	**153**	156	155	156	157	157	157	157	156	157
Configuração III														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	0	0	2	2	75	127	163	178	183	185
Preguiçoso	0	0	0	0	0	3	**60**	104	145	155	158	158	158	158
Ponderado_R	0	0	0	0	0	**4**	52	**127**	**178**	188	190	196	199	199
Ponderado	0	0	0	0	0	0	2	66	162	**203**	**223**	**228**	**228**	**228**
Guloso_R	0	0	0	0	0	**4**	36	94	140	172	185	191	194	192
Guloso	0	0	0	0	0	0	0	7	36	109	171	210	221	**228**

Configuração IV														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	3	35	106	151	168	170	168	167	165	167	168	168
Preguiçoso	0	**7**	**109**	146	146	146	146	146	146	146	146	146	146	146
Ponderado_R	0	1	94	153	156	155	155	156	156	157	156	156	157	157
Ponderado	0	3	100	**156**	**158**	158	158	158	158	158	158	158	158	158
Guloso_R	0	0	6	61	145	**172**	181	183	184	184	186	190	188	191
Guloso	0	0	0	4	60	125	**183**	**203**	**203**	**203**	**203**	**203**	**203**	**203**
Configuração V														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	1	10	63	110	149	148	153	153	151	155	154
Preguiçoso	0	0	0	**19**	**70**	**118**	135	139	139	139	139	139	139	139
Ponderado_R	0	0	0	3	34	106	144	163	164	164	166	166	169	170
Ponderado	0	0	0	0	14	65	**166**	**170**	**175**	177	177	177	177	177
Guloso_R	0	0	0	0	19	76	137	156	165	165	169	170	171	171
Guloso	0	0	0	0	0	24	118	155	171	**182**	**182**	**182**	**182**	**182**
Configuração VI														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	7	21	74	99	148	166	177	176	177	180
Preguiçoso	0	0	0	7	70	123	156	171	171	171	171	171	171	171
Ponderado_R	0	0	**2**	**16**	**75**	**134**	**171**	**173**	174	174	174	174	174	174
Ponderado	0	0	0	12	60	123	168	**173**	175	175	175	175	175	175
Guloso_R	0	0	0	2	22	68	126	168	**178**	187	189	190	190	190
Guloso	0	0	0	0	0	6	38	103	159	**195**	**201**	**201**	**201**	**201**
Configuração VII														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	3	26	72	143	160	171	169	169	171	169	170	171
Preguiçoso	0	1	34	93	138	141	141	141	141	141	141	141	141	141
Ponderado_R	0	0	**38**	**118**	149	153	155	155	155	156	156	156	156	157
Ponderado	0	0	**38**	**118**	**153**	157	159	159	159	159	159	159	159	159
Guloso_R	0	0	0	55	127	**159**	171	173	176	177	178	178	182	182
Guloso	0	0	0	4	33	137	**178**	**196**	**196**	**194**	**196**	**196**	**196**	**196**

Dados os índices apresentados na tabela acima, foi examinado o comportamento individual de cada heurística para os diferentes *deadline*s em cada configuração sorteada. Analisados esses resultados, constatou-se que o comportamento das heurísticas permaneceu, em geral, semelhante à bateria de testes da primeira fase de avaliação.

No caso da Configuração III, observa-se que o Algoritmo Ponderado foi igual ou melhor do que o Algoritmo Guloso em *deadline*s folgados. Isto acontece em função da existência de uma armadilha no diagrama de recursos da missão, ou seja, um ponto no diagrama de recursos onde o Algoritmo Guloso toma uma decisão míope (maior benefício

local) que na verdade o afasta do itinerário de maior benefício global. Em função da premissa da miopia é impossível evitar completamente as armadilhas. Porém serão apresentadas mais adiante nesta tese técnicas que procuram reduzir este problema (clones e algoritmos adaptativos).

Na Tabela 6.7 são apresentados os valores do intervalo de confiança, com confiança de 95%, para o benefício global de todas as heurísticas referente a cada *deadline* (para as 7 configurações).

Tabela 6.7: Intervalos de confiança das 7 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Configuração I														
Aleatório	0	0	0	±2,6	±11,6	±16,1	±12,6	± 7,1	± 4,2	± 2,3	± 2,2	± 2,1	± 2,2	± 2,6
Preguiçoso	0	0	0	± 12	± 9,1	± 5,7	± 2,9	0	0	0	0	0	0	0
Ponderado_ R	0	0	0	±11,1	±11,5	± 5,3	± 1,5	± 1,4	± 1,6	± 1,4	± 1,7	± 1,4	± 1,6	± 1,5
Ponderado	0	0	0	± 4,7	±16,3	±10,9	± 3,3	0	0	0	0	0	0	0
Guloso_R	0	0	0	0	±13,1	±16,7	± 8,9	± 7,1	± 2	± 2	± 2,1	± 2,1	± 2,1	± 2,1
Guloso	0	0	0	0	0	0	±18,9	±13,9	± 9,5	0	0	0	0	0
Configuração II														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	± 3,8	±11,3	±11,2	± 4,4	± 1,7	± 1,8	± 1,7	± 1,8	± 1,7	± 1,6	± 2	± 1,7
Preguiçoso	0	0	±12,4	±10,1	0	0	0	0	0	0	0	0	0	0
Ponderado_R	0	0	± 8,7	±11,1	± 3,2	± 1	± 1,1	± 0,9	± 0,8	± 0,7	± 0,7	± 0,8	± 1	± 0,8
Ponderado	0	0	± 6,1	±13,7	± 6,8	0	0	0	0	0	0	0	0	0
Guloso_R	0	0	± 4,4	±15,5	± 6,9	± 4,5	± 0,9	± 0,9	± 0,9	± 1	± 0,7	± 0,8	± 0,8	± 0,9
Guloso	0	0	± 3,2	±13,6	±11,8	± 3,2	0	0	0	0	0	0	0	0
Configuração III														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	0	0	± 3,2	±12,5	±17,3	±16,9	±12,7	± 7,8	± 4,7	± 2,8
Preguiçoso	0	0	0	0	0	± 4,4	±15,1	±14,7	± 8,4	± 4,4	0	0	0	0
Ponderado_R	0	0	0	0	0	± 5,1	±16,5	±17,7	±10,5	± 5,7	± 4,2	± 2,2	± 2,4	± 2
Ponderado	0	0	0	0	0	0	± 4,5	±20,4	±20,4	±14,1	± 6,3	0	0	0
Guloso_R	0	0	0	0	0	± 4,8	±14,3	±18,2	± 16	±10,8	± 6,7	± 2	± 1,7	± 1,9
Guloso	0	0	0	0	0	0	0	± 7,7	±16,5	±22,4	±19,5	±12,1	± 7,7	0
Configuração IV														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	± 3,2	±13,3	±15,8	± 9,6	± 4,1	± 2,7	± 2,7	± 2,8	± 2,5	± 2,6	± 2,4	± 2,7
Preguiçoso	0	± 6,3	±12,5	0	0	0	0	0	0	0	0	0	0	0
Ponderado_R	0	± 2,9	±14,8	± 3,2	± 1	± 1	± 1	± 0,8	± 0,9	± 0,6	± 0,8	± 0,7	± 0,6	± 0,7
Ponderado	0	± 4,4	± 15	± 3,1	0	0	0	0	0	0	0	0	0	0

Guloso_R	0	0	± 6,2	±16,2	±13,1	± 7,2	± 4,1	± 2,3	± 2,1	± 2,2	± 2,4	± 2	± 1,9	± 2
Guloso	0	0	0	± 5,6	±18,3	±19,4	± 12	0	0	0	0	0	0	0
Configuração V														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	± 2,8	± 7,4	±14,6	±13,4	± 4,8	± 4,8	± 2,1	± 2,1	± 2,6	± 2,4	± 2,5
Preguiçoso	0	0	0	± 9,5	±13,7	± 9,8	± 4,7	0	0	0	0	0	0	0
Ponderado_R	0	0	0	± 4,2	±12,6	±14,7	±10,1	± 3,7	± 1,9	± 1,9	± 1,8	± 1,9	± 1,6	± 1,5
Ponderado	0	0	0	0	± 9,5	±16,8	± 8,3	± 6,8	± 3,5	0	0	0	0	0
Guloso_R	0	0	0	0	±10,2	±15,8	±11,9	± 7,3	± 1,9	± 1,8	± 1,7	± 1,9	± 1,7	± 1,7
Guloso	0	0	0	0	0	±12,1	±17,1	±12,8	± 8,5	0	0	0	0	0
Configuração VI														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0	0	± 6,5	±11,2	±17,2	±17,3	±12,8	± 9,2	± 3,9	± 1,9	± 1,9	± 1,9
Preguiçoso	0	0	0	± 6,6	±16,6	±15,1	± 9,6	0	0	0	0	0	0	0
Ponderado_ R	0	0	± 3,4	± 9,8	± 17	±14,4	± 4,8	± 3,4	± 0,3	± 0,3	± 0,3	± 0,3	± 0,3	± 0,3
Ponderado	0	0	0	± 8,8	±16,3	±15,8	± 6,8	± 3,4	0	0	0	0	0	0
Guloso_R	0	0	0	± 3,6	±11,7	±17,4	± 17	±10,5	± 8,1	± 3,9	± 1,4	± 1,3	± 1,4	± 1,3
Guloso	0	0	0	0	0	± 6,8	±15,5	±19,8	±16,1	± 6,8	0	0	0	0
Configuração VII														
	150	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	± 4,3	±11,9	±16,4	±12,9	± 8,9	± 2,4	± 2,3	± 2,2	± 2,2	± 2,2	± 2,6	± 2,4
Preguiçoso	0	± 2,8	±11,9	±13,1	± 3,9	0	0	0	0	0	0	0	0	0
Ponderado_R	0	0	±12,9	±12,4	± 5,3	± 1,2	± 1	± 1	± 1,1	± 0,9	± 1	± 0,9	± 0,9	± 0,8
Ponderado	0	0	±13,4	±13,7	± 6,1	± 3,1	0	0	0	0	0	0	0	0
Guloso_R	0	0	0	±15,2	±13,8	± 7,4	± 2,1	± 2	± 2,1	± 1,9	± 1,8	± 2	±1,8	±1,8
Guloso	0	0	0	± 5,4	±14,5	±17,7	±11,1	0	0	± 3,8	0	0	0	0

Os gráficos da Figura 6.11 apresentam os benefícios globais encontrados com a simulação das 7 diferentes missões da segunda fase: (a) configuração I, (b) configuração II, (c) configuração III, (d) configuração IV, (e) configuração V, (f) configuração VI, (g) configuração VII; e também o benefício global encontrado na simulação da primeira fase: (h) configuração da primeira fase.

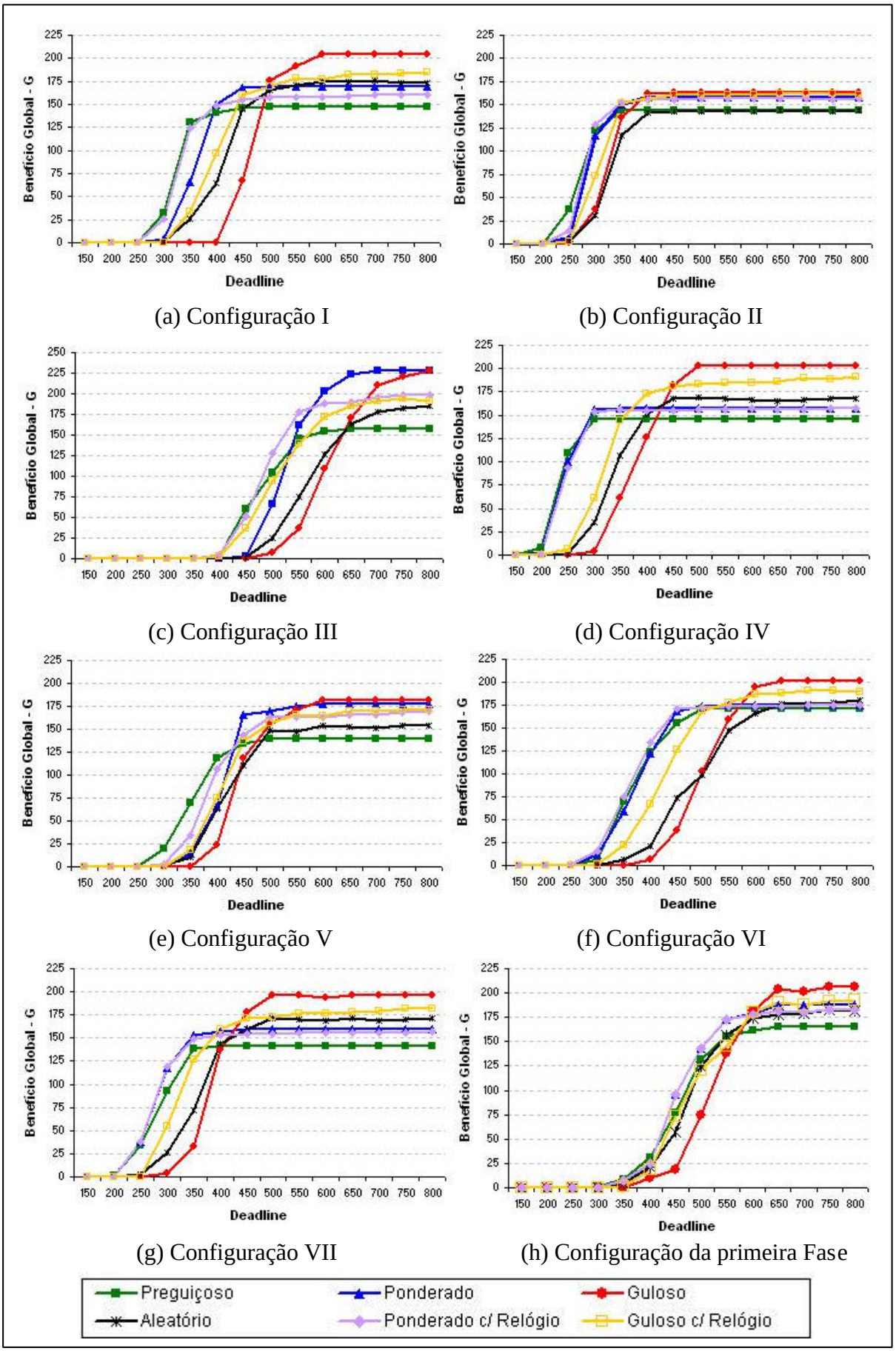


Figura 6.11: Benefícios globais para cada configuração.

Observando estes 8 gráficos, nota-se que o comportamento das heurísticas mantém, em geral, as características anteriormente descritas (na primeira fase de simulação, seção 6.4.1.). Devido a este comportamento similar apresentado, algumas afirmações podem ser feitas:

- O Algoritmo Preguiçoso sempre obteve os melhores índices com o menor *deadline*. Em todas as situações testadas, o Algoritmo Preguiçoso foi a primeira heurística a apresentar algum benefício (mesmo pequeno) quando o *deadline* era muito apertado;

- O Algoritmo Guloso sempre obteve o maior índice de benefício global em todas as situações testadas com *deadline* muito folgado;

- O Algoritmo Preguiçoso não apresentou bons resultados com *deadline*s folgados, seus índices de benefício global quando comparados as demais heurísticas foram pouco significativos;

- O Algoritmo Aleatório não apresentou um resultado satisfatório, o que já era esperado, pois seu fator de tomada de decisão é apenas um sorteio entre as próximas possíveis posições. Esta heurística, diferentemente das demais, não utiliza nenhum critério para sua tomada de decisão. Na configuração II, por exemplo, ele obteve o pior desempenho com todos os *deadline*s testados (com exceção de D = 250);

- O Algoritmo Ponderado e o Algoritmo Ponderado com Relógio apresentaram bons resultados com *deadline*s justos (maiores índices de benefício). Apresentaram bons resultados também com os demais tipos de *deadline*, geralmente resultados próximos aos melhores: em sua versão com relógio, alcançou valores próximos ao do Algoritmo Preguiçoso; em sua versão sem relógio, alcançou valores próximos ao Algoritmo Guloso e Guloso com Relógio. Na configuração III, por exemplo, estas heurísticas obtiveram os maiores índices de desempenho, sendo que a versão com relógio obteve os melhores resultados com *deadline*s apertados e justos e a versão sem relógio obteve os melhores resultados com *deadline*s folgados;

- Cumprindo as expectativas, a versão com relógio do Algoritmo Guloso apresentou índices melhores, com *deadline*s apertados, quando comparada com a sua versão original. Nos gráficos (a), (b), (d), (f), (g) e (h) podemos observar que o Algoritmo Guloso com Relógio, em situações com *deadline* folgado, obteve o segundo melhor índice, perdendo apenas para sua versão original.

6.4.3. Terceira Fase da Avaliação

Para os testes da terceira fase da avaliação 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 6.8 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* (com uma variação de 1%). A Tabela 6.9 mostra os intervalos de confiança de 95% para todas as heurísticas e para todos os *deadline*s.

Assim como nas fases de avaliação anteriores, o objetivo desta terceira fase de avaliação é observar o comportamento das heurísticas, desta vez simuladas com várias diferentes configurações.

A Figura 6.11, baseada nos dados descritos na Tabela 6.8, apresenta o gráfico do benefício global ***G*** obtido por cada algoritmo. As conclusões obtidas a partir deste gráfico são semelhantes àquelas das fases anteriores. Neste caso, a amostra utilizada envolve maior variação quanto à configuração das missões.

Tabela 6.8: Benefício Global das heurísticas simples para 100 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0,1	0,9	8,9	32,1	69,3	108,9	134,1	147,7	153,5	155,9	156,3	156,3	156,4
Preguiçoso	0	**2,2**	**13,1**	**40,6**	**81,3**	114,4	129,1	134,2	135,8	136,1	136,1	136,1	136,1	136,1
Ponderado_R	0	1,2	10,7	33,6	75,1	**117,4**	**139,1**	**149,9**	153,6	154,9	155,5	156,3	156,5	156,9
Ponderado	0	0,7	7,9	28,6	67	109,8	**140,6**	156,1	**160,9**	162,6	162,9	163,0	163,0	163,0
Guloso_Relógio	0	0,1	2,7	15,6	46	90,8	127,2	**149**	159,4	163,7	165,4	166,5	167,5	168,4
Guloso	0	0	0,4	4,2	19,4	52,4	95,0	136,6	**161,6**	**174,9**	**179,2**	**180,5**	**181,1**	**181,2**

Tabela 6.9: Intervalo de confiança das heurísticas simples para 100 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	±0,05	±0,21	±0,66	±1,18	±1,48	± 1,4	± 1,1	±0,79	±0,59	±0,47	±0,45	±0,44	±0,44
Preguiçoso	±0,01	± 0,3	±0,73	±1,17	± 1,3	±1,04	±0,74	±0,57	± 0,5	±0,48	±0,48	±0,48	±0,48	±0,48
Ponderado_R	±0,02	±0,23	±0,69	±1,17	±1,45	±1,27	±0,93	±0,64	± 0,5	±0,47	±0,46	±0,45	±0,46	±0,45
Ponderado	±0,02	±0,19	±0,62	±1,14	±1,51	±1,47	±1,14	±0,76	±0,57	±0,48	±0,46	±0,45	±0,45	±0,45
Guloso_R	0	±0,06	±0,37	±0,87	±1,37	±1,54	± 1,3	±0,93	±0,64	±0,48	±0,43	±0,42	±0,41	±0,41
Guloso	0	0	±0,14	± 0,5	±1,05	±1,56	±1,74	±1,52	±1,13	±0,74	±0,53	±0,44	± 0,4	±0,39

Analisando o gráfico da Figura 6.12, percebe-se (assim como nas fases de avaliação anteriores) que a partir do momento em que os *deadline*s 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.

Confirmando as características previamente descritas, o Algoritmo Guloso apresentou os melhores resultados para *deadline*s folgados e os menores índices com os *deadline*s apertados. O comportamento inverso é verificado pelo Algoritmo Preguiçoso, que apresentou os melhores resultados com *deadline*s apertados e os menores índices de benefício com *deadline*s folgados.

O Algoritmo Aleatório continuou apresentando resultados pouco significativos. Entretanto, por seu critério de decisão ser apenas um sorteio dos possíveis próximos passos, pode acontecer desta heurística em uma determinada missão apresentar melhores resultados.

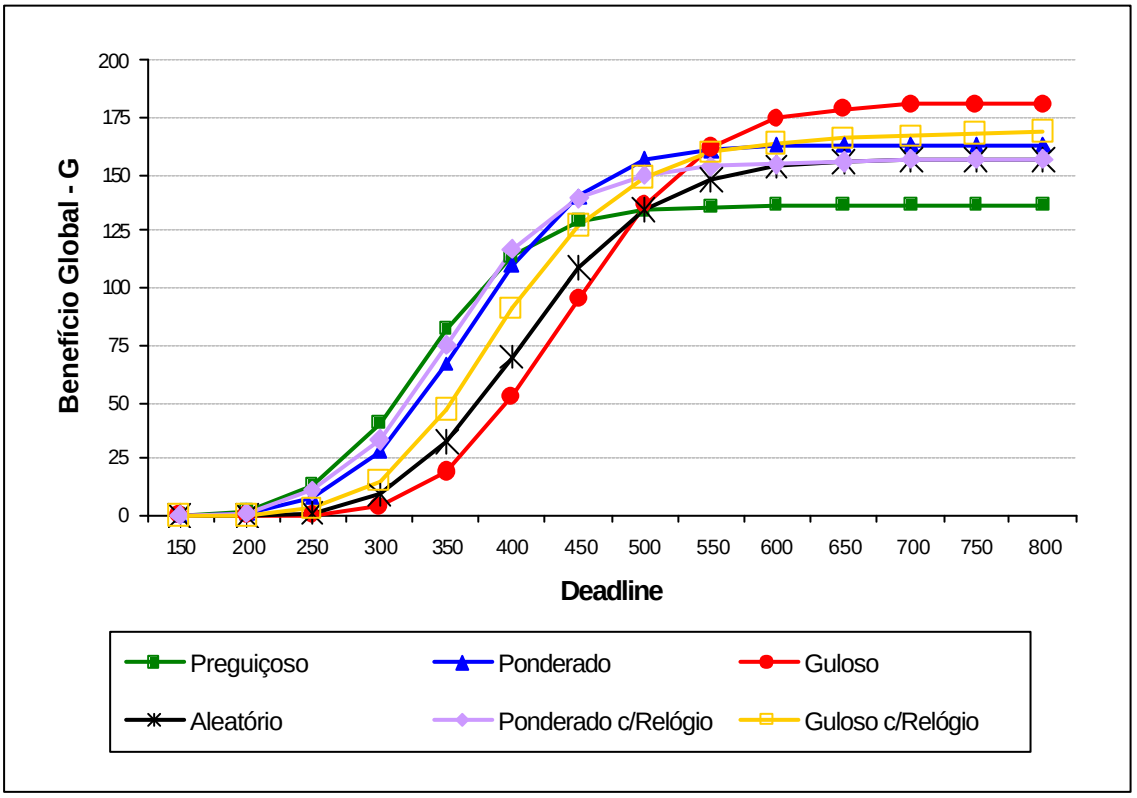


Figura 6.12: Benefício Global das heurísticas simples para 100 configurações.

As versões com relógio mantiveram as características que as distinguem de suas versões originais, apresentando melhores resultados com *deadline*s menores e a partir de um ponto mantiveram desempenho inferior, mas próximo, a suas versões originais. No

gráfico abaixo, essas movimentações dos índices de benefícios adquiridos a cada *deadline* são facilmente perceptíveis. Por exemplo, com *deadline*s apertados, o Algoritmo Ponderado com Relógio apresentou melhores resultados que sua versão original, esta vantagem diminui até que o Algoritmo Ponderado (sem relógio) supera os valores de beneficio de sua versão com relógio. O mesmo pode ser observado com o comportamento dos Algoritmos Guloso e Guloso com Relógio.

Após efetuadas as simulações, pode-se relacionar cada tipo de *deadline* a uma determinada heurística. Partindo do *deadline* maior para o menor: o *deadline* ***folgado*** é definido como aquele onde o Algoritmo Guloso, versões com e sem relógio, superam os demais; o *deadline* ***justo*** é definido como aquele onde o Algoritmo Ponderado, versões com e sem relógio, superam os demais; e a partir deste ponto, em direção a *deadline*s cada vez menores, temos os *deadline*s ***apertados***, onde o Algoritmo Preguiçoso obteve o melhor desempenho.

A Tabela 6.10 apresenta o desempenho das heurísticas segundo o benefício adquirido para cada tipo de *deadline*. Esta tabela indica o número de vezes que cada heurística alcançou os maiores índices de benefício por tipo de *deadline*. Cada linha indica as heurísticas que obtiveram os melhores resultados para cada faixa de *deadline*. Das 1400 execuções: 621 apresentaram *deadline* folgado, 269 apresentaram *deadline* justo e 510 apresentaram *deadline* apertado.

Observando a primeira entrada da tabela, pode-se notar que das 621 execuções com *deadline*s folgado, em 583 execuções o Algoritmo Guloso obteve o melhor desempenho e em 38 execuções o Algoritmo Guloso com Relógio obteve o melhor resultado. As demais heurísticas não alcançaram o melhor desempenho em nenhuma execução para esta faixa de *deadline*, pela própria definição usada aqui para *deadline* apertado.

Na terceira entrada da Tabela 6.10, que ilustra os resultados com *deadline* apertado, é importante ressaltar que, das 510 execuções, em 241 execuções nenhuma heurística obteve êxito, portanto empataram em zero. Isto significa que nenhuma delas conseguiu atender o *deadline*, por ser muito apertado. Estas execuções não foram incluídas na tabela.

Tabela 6.10: Desempenho das heurísticas simples por tipo de *deadline.*

***Deadline*/ Heurística**	**Aleatório**	**Preguiçoso**	**Ponderado c/Relógio**	**Ponderado**	**Guloso c/Relógio**	**Guloso**
Folgado	0	0	0	0	38	583
Justo	0	0	66	230	0	0
Apertado	0	216	33	10	9	1

A Tabela 6.10 resume o que foi observado ao longo de todas as fases de avaliação das heurísticas simples, isto é, a especialização das heurísticas com respeito ao tipo de *deadline*.

Na continuidade desta tese serão criadas técnicas que visam exatamente explorar esta especialização das heurísticas com o propósito de obter-se melhores resultados em qualquer situação.

6.5. Utilização de Agentes Clonados

Analisando os resultados das simulações descritas na seção 6.4, é possível perceber as vantagens de se combinar o uso das heurísticas simples para alcançar melhores desempenhos. A miopia do agente pode limitar a eficiência das heurísticas simples, portanto, uma possível solução é o uso de um agente clone que adota uma segunda heurística.

Muitos fatores influenciam sobre a determinação de qual é a melhor heurística simples. Estes fatores estão diretamente relacionados com o *deadline* da missão. Nesta seção é proposto que, no momento que a missão inicia, um par de agentes usando heurísticas diferentes seja lançado. O par de agentes (composto pelo agente móvel original e pelo agente móvel clone) deixa o nodo origem e segue diferentes itinerários em busca do mesmo objetivo. No final da missão, ambos agentes retornam ao nodo origem e então é feita uma comparação e uma seleção do melhor resultado individual do par.

Este método provavelmente aumentará o benefício global da missão e certamente não diminuirá seu valor global. No caso de um dos agentes não retornar ao nodo origem dentro do *deadline*, os dados obtidos pelo agente que já chegou serão utilizados. Considera-se que o *overhead* do sistema é controlado devido ao número de clones ser limitado.

Como a capacidade computacional dos nodos do sistema pode ser pequena, a clonagem é feita uma única vez, no nodo origem. A partir deste momento os agentes viajam de forma independente e não existe uma relação de hierarquia entre eles.

O comportamento que cada agente deverá exibir ao longo de todo itinerário também é definido no nodo origem antes de sua partida.

Na tentativa de encontrar um algoritmo que apresente um comportamento satisfatório, busca-se uma heurística flexível que seja capaz de cumprir seus requisitos com qualquer situação de carga no sistema e diferentes *deadline*s. O primeiro passo em busca desta heurística flexível, após feita a análise das heurísticas simples na seção anterior é propor e avaliar o uso de pares/trios de agentes clones.

O número de agentes clones por missão foi limitado a, no máximo, três para reduzir *overhead* na rede e nos nodos visitados, ou seja, evitar qualquer processamento extra para executar uma missão, tendo como conseqüência piorar o desempenho dos demais agentes móveis que utilizam a rede.

O uso de pares de agentes só foi possível porque as heurísticas simples apresentaram um comportamento distinto para cada faixa de *deadline*. Por exemplo, pode-se observar o desempenho obtido individualmente pelo Algoritmo Guloso através dos resultados das simulações da seção 6.4. Esta heurística, em suas duas versões, alcança os melhores índices de benefício em situações onde o *deadline* é grande o suficiente para o agente executar todos os recursos opcionais, assim aprimorando a qualidade da missão. Porém, quando comparado seu desempenho com as demais heurísticas, em situações onde o *deadline* definido para a missão não permite tamanha flexibilidade na execução de recursos opcionais, o uso desta heurística não é aconselhado visto que nestes casos apresentou os piores resultados.

Analisando o desempenho do Algoritmo Preguiçoso, notou-se que o comportamento é inverso ao apresentado pelo Algoritmo Guloso. Este algoritmo obteve baixos índices de desempenho com *deadline*s folgados, mas em situações em que o *deadline* era considerado apertado, seus resultados foram satisfatórios. É importante lembrar que o Algoritmo Preguiçoso, por não executar nenhum tipo de recurso opcional, alcança índices mais baixos de benefício e a medida que o *deadline* vai se estendendo seu benefício adquirido permanece o mesmo.

Como anteriormente já discutido, essas heurísticas representam pontos extremos sob qualquer forma de comparação efetuada. A heurística que obteve resultados intermediários foi o Algoritmo Ponderado, em suas duas versões.

6.6. Avaliação dos Agentes Clonados

Nesta etapa de simulações foram lançados 100 agentes com 100 diferentes configurações testadas com 14 *deadline*s (que classificam-se como: apertados, justos e folgados – como na seção 6.4). Os resultados discutidos a seguir são referentes a média destas execuções.

Baseados no conhecimento do comportamento individual de cada heurística simples foram avaliados algumas duplas/trios de agentes no cumprimento de algumas missões. A Tabela 6.11 apresenta estas duplas/trios de agentes.

Tabela 6.11: Siglas dos agentes clonados.

Sigla	Descrição
A-A	Dois clones com o Algoritmo Aleatório
Pg-G	Combinação do Algoritmo Preguiçoso com o Algoritmo Guloso
Pg-Pd	Combinação do Algoritmo Preguiçoso com o Algoritmo Ponderado
Pd-G	Combinação do Algoritmo Ponderado com o Algoritmo Guloso
Pg-GR	Combinação do Algoritmo Preguiçoso com o Algoritmo Guloso com Relógio
Pg-PdR	Combinação do Algoritmo Preguiçoso com o Algoritmo Ponderado com Relógio
PdR-G	Combinação do Algoritmo Ponderado com Relógio com o Algoritmo Guloso
PdR-GR	Combinação do Algoritmo Ponderado com Relógio com o Algoritmo Guloso com Relógio
Pg-Pd-G	Combinação dos Algoritmos Preguiçoso, Ponderado e Guloso
Pg-PdR-G	Combinação dos Algoritmos Preguiçoso, Ponderado com Relógio e Guloso

Oito pares e dois trios de heurísticas foram testados e seus resultados foram comparados aos desempenhos individuais obtidos. A Tabela 6.12 apresenta o resultado das simulações. Se traçada uma linha imaginária na Tabela 6.12, dividindo os *deadline*s em 3 grupos, é facilmente percebido que, para cada faixa de *deadline* (apertado, justo e folgado),

uma das heurísticas simples acima referenciadas obtém os melhores resultados. Os valores em negrito representam os maiores índices de benefício global alcançados para cada *deadline* (com variação de 1%).

Tabela 6.12: Benefício Global das heurísticas simples e baseadas em clones.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	1,2	9,1	29,1	62,6	98,6	128,6	146,8	155,5	159,7	161,1	161,9	161,7
Preguiçoso	**0,4**	**2,7**	**11,8**	**39,3**	**75,9**	105,3	124,6	134,6	138,7	140,6	141	141	141,1	141,1
Ponderado_R	0,3	1,9	7,9	33,6	71	**106,9**	134,1	150,8	156,4	159,5	160,7	161,4	161,7	162,5
Ponderado	0	1,7	6,7	27,8	65,8	104,4	**136,1**	**155,9**	**163,5**	166,5	167,4	167,6	167,6	167,6
Guloso_Rel.	0	0,3	3,7	16	41,8	78,4	117,5	145,4	160,3	168	171,4	173,2	174,1	175,1
Guloso	0	0	0,2	5,5	21,1	45,3	82,9	123,1	155,1	**173,6**	**181,9**	**186,2**	**187,7**	**188,2**
A-A	0	0	1,2	9,1	29,1	62,6	98,6	128,6	146,8	155,5	159,7	161,1	161,9	161,7
Pg-G	0,4	2,7	11,8	41,4	82,5	118,7	147,6	167,6	178,9	**184,5**	**186,7**	**187,8**	**188,2**	**188,4**
Pg-Pd	0,4	3,1	14,9	50,7	**96,1**	**131**	**153,4**	164	166,6	167,5	167,6	167,6	167,6	167,6
Pd-G	0	1,7	6,8	30,1	71,4	113,8	149,8	**172,7**	**181,8**	**185,9**	**187,4**	**188,0**	**188,3**	**188,4**
Pg-GR	0,4	2,8	13,5	45	87,4	123,8	148,8	162,6	168,2	170,9	172,5	173,5	174,3	175,1
Pg-PdR	**0,6**	**3,2**	**15,8**	**53**	**95,3**	128	147,4	156,7	159,1	160,4	160,9	161,4	161,7	162,5
PdR-G	0,3	1,9	8	35,8	76,4	117,5	150,6	**171,4**	**181**	**185,5**	**187,1**	**188**	**188,3**	**188,5**
PdR-GR	0,3	2	9,7	39,8	81,9	121,9	150,2	164,9	170,3	172,7	174,1	175	175,7	176,5
Pg-Pd-G	0,4	**3,1**	15	52,1	**99,4**	**137,3**	**163,5**	**177,9**	**183,4**	**186,3**	**187,4**	**188**	**188,3**	**188,4**
Pg-PdR-G	**0,6**	**3,1**	**15,9**	**54,3**	**99,4**	**136,2**	161	175,6	**182,3**	**185,8**	**187,2**	**188**	**188,3**	**188,4**

A dupla de algoritmos Aleatório/Aleatório (A-A) foi o único par lançado utilizando a mesma heurística devido ao seu comportamento irregular. O par de algoritmos Preguiçoso/Guloso (Pg-G) apresentou resultados satisfatórios, mas apenas compôs o envelope superior com *deadline*s folgados. Nos demais *deadline*s, seus resultados na maioria das vezes estiveram próximos aos índices mais altos de benefício adquirido.

O par de algoritmos Preguiçoso/Ponderado com Relógio (Pg-PdR), conquistou o melhor desempenho em situações nas quais o *deadline* estava apertado (até D=300). Pode–se afirmar que, para *deadline*s apertados, foi o par que apresentou melhores resultados, mas a medida que o *deadline* foi aumentado esta vantagem desapareceu.

É importante notar que o desempenho das duplas, em todas as situações, foi superior ao desempenho alcançado com a execução das heurísticas individualmente. As duplas que contiveram o Algoritmo Guloso garantiram altos índices em seus desempenhos com *deadline*s folgados.

O par de algoritmos Ponderado/Guloso (Pd-G) obteve destaque entre os demais para *deadline*s folgados e justos, mas apresentou resultados mais baixos para *deadline*s apertados. Na tentativa de suprir esta lacuna, foi lançado o par de algoritmos Ponderado com Relógio/Guloso (PdR-G). Se comparados os resultados entre estes dois pares, percebe-se a vantagem em se utilizar o segundo par, no qual a versão com relógio para o Algoritmo Ponderado trás um melhor desempenho para *deadline*s apertados e o bom desempenho apresentado pelo primeiro par com *deadline*s justos e folgados foi mantido.

O par Ponderado com Relógio/Guloso com Relógio (PdR-GR) atingiu resultados medianos, não apresentando destaque em nenhuma faixa de *deadline*.

Enfim, baseado nos valores das simulações indicados pela Tabela 6.12, é possível definir o melhor par para cada *deadline*. Se existir a necessidade de sugerir um par (ou até mesmo uma heurística apenas) para *deadline*s com valores intermediários entre os apresentados nas simulações, a indicação deve ser feita baseada no *deadline* imediatamente inferior. Por exemplo, para se indicar o melhor par para o *deadline* 475, deve ser considerado o resultado do *deadline* 450 – assim indicando o uso do par Preguiçoso/Ponderado.

A Figura 6.12 ilustra o comportamento das heurísticas (individualmente ou em pares), com seus respectivos benefícios globais obtidos. O envelope superior é formado pelos pares de algoritmos: Preguiçoso/Ponderado com Relógio, Preguiçoso/Ponderado e Ponderado/Guloso. A Figura 6.13 (a) apresenta todas as heurísticas simples e em clones que foram testadas. Para facilitar a visualização do comportamento das heurísticas, na Figura 6.13 (b) foram mantidas apenas as heurísticas que apresentaram os melhores desempenhos.

Em vista dos resultados encontrados, considerando o benefício global, pode-se concluir que o melhor par para *deadline*s apertados foi o par Preguiçoso/Ponderado com Relógio; para *deadline*s justos foi o par Preguiçoso/Ponderado; e para *deadline*s folgados foram os pares Ponderado/Guloso e Preguiçoso/Guloso.

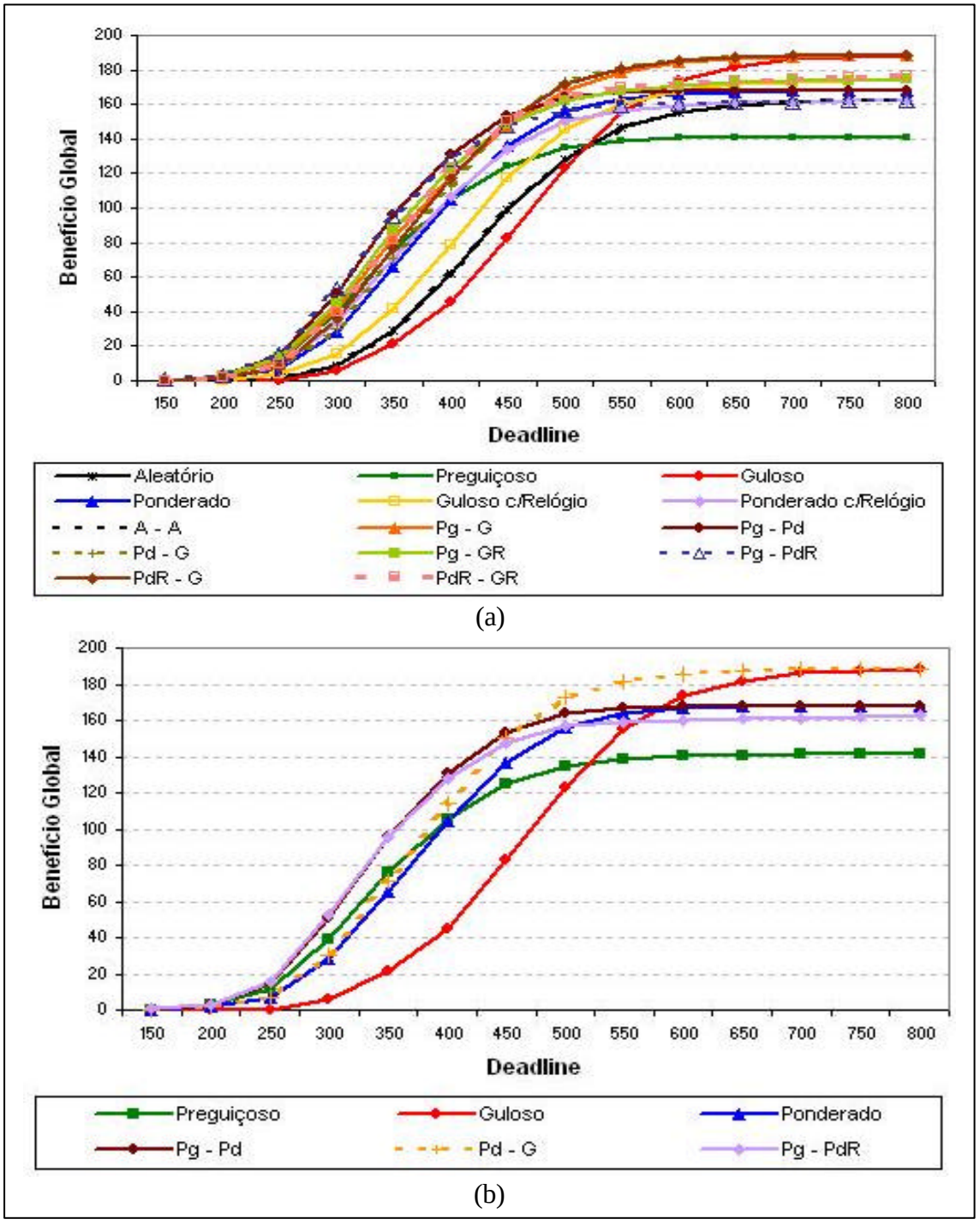


Figura 6.13: Benefício Global das heurísticas simples e baseadas em clones.

A dupla de heurísticas que obteve o maior número de índices mais altos (dos 14 níveis de *deadline*s testados) foi o par Ponderado/Guloso. É importante ressaltar que alguns pares obtiveram resultados com valores muito próximos (a Tabela 6.12 indica estes resultados). Portanto, a escolha do melhor par deve ser feita baseando-se nos resultados durante todo o transcorrer da tabela, por intermédio de uma análise comparativa do

comportamento do par para cada *deadline*, e considerando não apenas o índice mais alto de cada coluna.

A Tabela 6.13 indica o benefício global relativo, ou seja, parte-se do índice com valor mais alto para cada *deadline* – este valor recebe 1 e os demais benefícios são calculados proporcionalmente em relação ao este maior índice. Desta forma, torna-se possível conhecer a heurística e o par de heurísticas que apresentaram os melhores resultados de uma forma geral.

A taxa de atendidos com *deadline* igual a 150 e 200 é menor que 0,5%, portanto as médias de valor são pouco significativas porque tratam de eventos raros, onde uma única ocorrência pode alterar completamente o resultado. Para o cálculo do benefício médio relativo foram considerados os *deadline*s a partir de 250, onde a taxa de cumprimento de *deadline* alcança pelo menos 10%.

A última coluna da tabela abaixo apresenta o benefício global relativo médio obtido por cada heurística. Entre as execuções individuais das heurísticas, a que apresentou melhor desempenho foi o Algoritmo Ponderado, seguido de sua versão com Relógio.

Tabela 6.13: Benefício global relativo das heurísticas simples e baseadas em clones.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**	**Md**
Aleatório	0	0,01	0,11	0,23	0,38	0,59	0,72	0,83	0,9	0,9	0,88	0,86	0,86	0,86	**0,67**
Preguiçoso	1	1	1	1	1	0,99	0,92	0,86	0,85	0,81	0,77	0,76	0,75	0,75	**0,87**
Ponderado_R	0,71	0,69	0,67	0,85	0,94	1	0,99	0,97	0,96	0,92	0,88	0,87	0,86	0,86	**0,89**
Ponderado	0,11	0,63	0,57	0,71	0,87	0,98	1	1	1	0,96	0,92	0,9	0,89	0,89	**0,89**
Guloso_R	0	0,11	0,31	0,41	0,55	0,73	0,86	0,93	0,98	0,97	0,94	0,93	0,93	0,93	**0,79**
Guloso	0	0	0,01	0,14	0,28	0,42	0,61	0,79	0,95	1	1	1	1	1	**0,68**
A-A	0	0,01	0,08	0,17	0,29	0,46	0,6	0,72	0,8	0,83	0,85	0,86	0,86	0,86	**0,61**
Pg-G	0,64	0,85	0,74	0,76	0,83	0,86	0,9	0,94	0,98	0,99	1	1	1	1	**0,91**
Pg-Pd	0,71	1	0,94	0,93	0,97	0,95	0,94	0,92	0,91	0,9	0,89	0,89	0,89	0,89	**0,92**
Pd-G	0,07	0,54	0,43	0,55	0,72	0,83	0,92	0,97	0,99	1	1	1	1	1	**0,86**
Pg-GR	0,64	0,88	0,85	0,83	0,88	0,9	0,91	0,91	0,92	0,92	0,92	0,92	0,93	0,93	**0,90**
Pg-PdR	1	1	1	0,97	0,96	0,93	0,9	0,88	0,87	0,86	0,86	0,86	0,86	0,86	**0,90**
PdR-G	0,45	0,59	0,5	0,66	0,77	0,86	0,92	0,96	0,99	1	1	1	1	1	**0,88**
PdR-GR	0,45	0,63	0,61	0,73	0,82	0,89	0,92	0,93	0,93	0,93	0,93	0,93	0,93	0,94	**0,87**
Pg-Pd-G	0,71	1	0,94	0,96	1	1	1	1	1	1	1	1	1	1	**0,99**
Pg-PdR-G	1	1	1	1	1	0,99	0,98	0,99	0,99	1	1	1	1	1	**1,0**

Quando comparados os valores referentes aos clones, o par de algoritmos que alcançou maior índice foi Preguiçoso/Ponderado (Pg-Pd), seguido pelo par Preguiçoso/Guloso (Pg-G). O desempenho obtido por estes dois pares confirmam as expectativas iniciais.

Uma vez que o Algoritmo Preguiçoso supre os objetivos da missão para os menores *deadline*s, o Algoritmo Ponderado supre os objetivos da missão para *deadline*s justos e o Algoritmo Guloso provê os melhores resultados para *deadline*s folgados, imaginou-se que a combinação entre eles obtivesse um comportamento que resultasse no cumprimento dos requisitos da missão para diferentes *deadline*s.

Dois trios de agentes foram lançados e, conforme esperado, obtiveram os maiores índices de benefício global relativo médio, uma vez que para compor os trios, foram utilizadas as heurísticas que geralmente formavam o envelope superior dos gráficos de benefício global. O gráfico de barras da Figura 6.14 ilustra os resultados já discutidos.

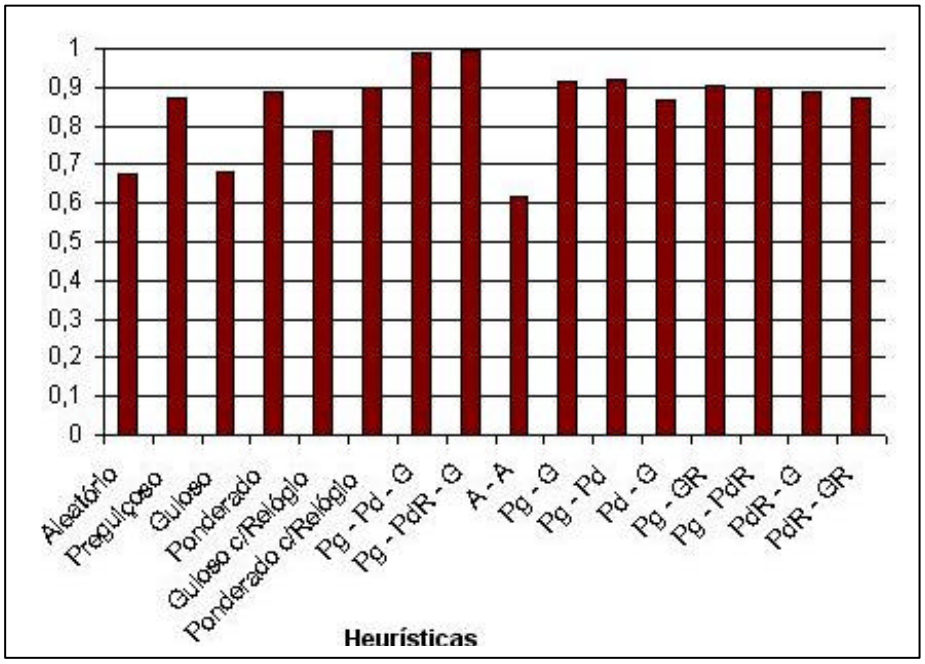


Figura 6.14: Benefício global relativo médio das heurísticas simples e baseadas em clones.

Os números apresentados na Tabela 6.14 definem um intervalo de valores para o benefício médio de cada heurística, com uma confiança de 95%. Quanto maior o intervalo, maior a variação no comportamento da heurística.

Nos algoritmos que não utilizam relógio, a medida que utilizam um *deadline* cada vez mais folgado, o intervalo de confiança tende a zero. No caso da Tabela 6.14, esta situação não está representada porque não foram efetuados testes com *deadline*s folgados o suficiente para alcançar zero no intervalo de confiança, o que indicaria a não variação nos valores de benefício de cada heurística.

Observando, por exemplo, o par de algoritmos Preguiçoso/Guloso (Pg-G) percebe-se que a variação no intervalo de confiança inicia com valores muito baixos, esta variação aumenta (com *deadline* em torno de 350) e gradativamente volta a diminuir, tendendo a zero. O que significa que este par apresenta maior variação em seu comportamento com *deadline*s próximos a 350. Quando comparado ao *deadline* 800, por exemplo, o grau de determinismo em relação ao benefício médio obtido é maior.

Tabela 6.14: Intervalo de confiança das heurísticas simples e baseadas em clone.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	±0,03	±0,25	±0,68	±1,15	±1,47	±1,49	±1,26	±0,95	±0,71	±0,53	±0,47	±0,44	±0,44
Preguiçoso	±0,13	±0,33	±0,71	±1,18	±1,32	±1,18	±0,94	±0,73	±0,63	±0,56	±0,55	±0,55	±0,55	±0,55
Ponderado_R	±0,11	±0,29	±0,62	± 1,2	±1,47	± 1,4	±1,13	±0,78	± 0,6	±0,49	±0,45	±0,43	±0,43	±0,42
Ponderado	±0,05	±0,29	± 0,6	±1,16	±1,54	±1,54	±1,27	±0,89	±0,63	±0,49	±0,43	±0,42	±0,42	±0,42
Guloso_R	0	±0,12	±0,45	±0,91	±1,35	±1,58	±1,48	±1,16	±0,86	±0,62	± 0,5	±0,44	±0,43	±0,42
Guloso	0	0	±0,11	±0,6	±1,12	±1,52	±1,78	±1,72	±1,4	±1,02	±0,75	±0,54	±0,45	±0,41
A-A	0	±0,03	±0,25	±0,68	±1,15	±1,47	±1,49	±1,26	±0,95	±0,71	±0,53	±0,47	±0,44	±0,44
Pg-G	±0,13	±0,33	±0,72	±1,23	±1,42	± 1,3	±1,04	±0,76	±0,57	±0,45	±0,41	± 0,4	± 0,4	± 0,4
Pg-Pd	±0,14	±0,37	±0,82	±1,34	±1,44	± 1,2	±0,84	±0,55	±0,46	±0,42	±0,42	±0,42	±0,42	±0,42
Pd-G	±0,05	±0,29	±0,61	±1,22	± 1,6	±1,57	±1,25	±0,81	±0,55	±0,44	± 0,4	± 0,4	± 0,4	± 0,4
Pg-GR	±0,13	±0,34	±0,78	±1,27	±1,41	±1,22	± 0,9	±0,61	±0,49	±0,44	±0,42	±0,42	±0,42	±0,42
Pg-PdR	±0,16	±0,36	±0,84	±1,33	±1,38	±1,12	± 0,8	±0,55	±0,47	±0,44	±0,43	±0,43	±0,43	±0,42
PdR-G	±0,11	±0,29	±0,63	±1,25	±1,54	±1,46	±1,15	±0,77	±0,55	±0,43	±0,41	± 0,4	± 0,4	± 0,4
PdR-GR	±0,11	± 0,3	± 0,7	±1,29	±1,52	±1,36	± 1	±0,64	±0,48	±0,42	±0,41	± 0,4	± 0,4	± 0,4
Pg-Pd-G	±0,14	±0,37	±0,83	±1,37	±1,48	±1,24	±0,86	±0,53	±0,43	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4
Pg-PdR-G	±0,16	±0,36	±0,84	±1,37	±1,43	±1,18	±0,84	±0,54	±0,45	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4

A Tabela 6.15 mostra os índices que indicam o percentual de agentes que foram atendidos referentes a cada *deadline*. Pode-se observar que estes índices tendem a 1, o que representa 100% de agentes atendidos. O algoritmo Guloso exigiu *deadline*s maiores para atingir 100% de *deadline*s atendidos. Na primeira coluna, relativa ao *deadline* 150, todas as

heurísticas apresentam valor zero como percentual de agentes atendidos, já que o cálculo de sua média gerou números muito pequenos, irrelevantes para a análise.

Vale ressaltar que, quando efetuado o uso de clones, obtém-se 100% de atendidos com *deadline*s menores que em execuções individuais do agente. Por exemplo, com *deadline* 550 com os pares Preguiçoso/Ponderado, Preguiçoso/Guloso com Relógio e Preguiçoso/Ponderado com Relógio. Enquanto que o Algoritmo Preguiçoso individualmente alcançou 100% de atendidos com *deadline* 600, o Algoritmo Ponderado com Relógio precisou do *deadline* 650 e o Algoritmo Guloso com Relógio precisou de *deadline* 700.

Tabela 6.15: Taxa média de *deadline*s atendidos das heurísticas simples e baseadas em clones.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Aleatório	0	0	0,01	0,06	0,2	0,42	0,64	0,81	0,92	0,97	0,99	1	1	1
Preguiçoso	0	0,03	0,1	0,31	0,58	0,78	0,9	0,96	0,99	1	1	1	1	1
Ponderado_Rel	0	0,02	0,06	0,24	0,48	0,71	0,86	0,95	0,98	0,99	1	1	1	1
Ponderado	0	0,01	0,05	0,18	0,42	0,65	0,83	0,94	0,98	0,99	1	1	1	1
Guloso_Relogio	0	0	0,03	0,11	0,27	0,5	0,72	0,87	0,95	0,98	0,99	1	1	1
Guloso	0	0	0	0,03	0,12	0,26	0,46	0,67	0,84	0,93	0,97	0,99	1	1
A-A	0	0	0,01	0,06	0,2	0,42	0,64	0,81	0,92	0,97	0,99	1	1	1
Pg-G	0	0,03	0,1	0,32	0,59	0,79	0,92	0,97	0,99	1	1	1	1	1
Pg-Pd	0	0,03	0,12	0,37	0,65	0,84	0,95	0,99	1	1	1	1	1	1
Pd-G	0	0,01	0,05	0,19	0,44	0,68	0,86	0,96	0,99	1	1	1	1	1
Pg-GR	0	0,03	0,11	0,34	0,61	0,82	0,93	0,98	1	1	1	1	1	1
Pg-PdR	0	0,03	0,13	0,39	0,66	0,85	0,95	0,99	1	1	1	1	1	1
PdR-G	0	0,02	0,06	0,25	0,5	0,73	0,88	0,97	0,99	1	1	1	1	1
PdR-GR	0	0,02	0,07	0,27	0,54	0,77	0,91	0,98	0,99	1	1	1	1	1
Pg-Pd-G	0	0,03	0,12	0,37	0,65	0,84	0,95	0,99	1	1	1	1	1	1
Pg-PdR-G	0	0,03	0,13	0,39	0,67	0,86	0,95	0,99	1	1	1	1	1	1

A Tabela 6.16 indica os pares que obtiveram os melhores benefícios médios. Para cada configuração é encontrado um benefício médio máximo, todos os pares que possuírem este benefício médio somam 1. Desta forma o par que alcançar o maior número a cada coluna (na Tabela 6.16) é o melhor par para aquele *deadline*. Os valores em negrito indicam os índices de desempenho mais altos para cada *deadline*.

Por exemplo, com *deadline* 400, em 37 execuções de 100 configurações o par Preguiçoso/Ponderado (Pg-Pd) apresentou o melhor valor. Com este mesmo *deadline*, pode-se observar que o par Preguiçoso/Guloso com Relógio (Pg-GR), das 100 configurações em 5 vezes atingiu o maior índice de benefício. A soma de cada coluna, referente a cada *deadline*, pode ultrapassar 100 devido aos possíveis empates. A soma da coluna analisada (D=400) é 111, o que indica que 11 pares de heurísticas tiveram seus valores empatados, assim sendo contabilizado para ambas.

Nas ocasiões em que o trio de heurísticas alcançou valor igual ou superior aos pares em determinada coluna, lhe era acrescida 1 unidade. Seguindo o exemplo com *deadline* igual a 400, das 100 execuções em 73 o trio de algoritmos Preguiçoso/Ponderado/Guloso (Pg-Pd-G) obteve o benefício global médio igual ou superior ao melhor índice alcançado pelos pares.

Assim como na tabela que apresentou o benefício global relativo (Tabela 6.13), para análise da Tabela 6.16 foram considerados apenas os valores referentes aos *deadline*s a partir de 250, onde a taxa de cumprimento de *deadline* alcança pelo menos 10%. As taxa de atendidos com *deadline* igual a 150 e 200 é menor que 0,5%, portanto pouco significativas porque tratam de eventos raros.

O par Ponderado/Guloso (Pd-G), dos 12 *deadline*s considerados, obteve os melhores índices com os 8 maiores *deadline*s. O que confirma sua eficiência em situações onde a missão possui *deadline* folgado ou justo.

O par que obteve os mais altos índices de benefício médio com *deadline*s apertados foi o Preguiçoso/Ponderado com relógio (Pg-PdR). Em situações onde o *deadline* tende do apertado ao justo, a dupla que apresentou melhor desempenho foi Preguiçoso/Ponderado (Pg-Pd).

Tabela 6.16: Desempenho das heurísticas (simples ou baseadas em clones) conforme os maiores benefícios adquiridos.

Heurística/ *Deadline*	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
A-A	46	19	5	0	0	0	0	0	0	0	0	0
Pg-G	59	27	8	4	8	14	26	45	64	78	87	95
Pg-Pd	68	45	**46**	**37**	25	13	3	7	6	6	6	6
Pd-G	46	23	17	23	**48**	**74**	**87**	**94**	**100**	**99**	**100**	**100**

Pg-GR	62	29	11	5	3	1	2	0	0	0	0	0
Pg-PdR	**87**	**73**	38	25	9	3	0	0	0	0	0	0
PdR-G	47	21	9	6	15	24	39	54	75	85	94	96
PdR-GR	48	23	9	11	11	5	2	0	0	1	1	1
Pg-Pd-G	68	51	**63**	**73**	**88**	**94**	**96**	**99**	**100**	**100**	**100**	**100**
Pg-PdR-G	**87**	**81**	52	45	42	36	45	57	76	85	94	96

Com relação aos trios de agentes clones, o resultado esperado mais uma vez foi confirmado, onde o índice de obtenção dos melhores resultados foi muito próximo a 100% em todos os *deadline*s analisados. O trio composto pelas heurísticas Preguiçoso/Ponderado/Guloso (Pg-Pd-G) obteve o maior índice em 10 dos 12 períodos considerados.

6.7. Comparações entre as Heurísticas

Em vista dos resultados apresentados nas diferentes fases de avaliação, pode-se claramente perceber o perfil de cada heurística estudada.

O Algoritmo Aleatório não apresentou resultados significativos em nenhuma situação simulada. Deve-se isto por sua forma de tomada de decisão, isto é, depende de fatores incertos, favoráveis ou não a um determinado evento.

As versões com relógio mantiveram as características que as distinguem de suas versões originais, apresentando melhores resultados com *deadline*s menores e a partir de um ponto mantiveram desempenho inferior, mas próximo, a suas versões originais. Na maioria das situações simuladas, mesmo apresentando resultados pouco expressivos com relação a suas versões originais, justificou-se o uso do relógio para situações em que a restrição temporal apresentava-se um pouco mais rígida (*deadline*s um pouco mais justos). Onde estas heurísticas, conhecendo as situações atuais da missão em execução, e percebendo a necessidade em se concluir a missão de uma forma mais rápida possível – por questão da proximidade do *deadline* – tenderam a mudar seu comportamento, adotando um comportamento mais rápido na tentativa em se cumprir o *deadline* da missão em execução.

Uma importante constatação é que os Algoritmos Guloso, Preguiçoso e Ponderado apresentaram resultados nítidos que permitem descrever seus comportamentos

para cada tipo de *deadline*. Sendo possível assim estimar se sua utilização para determinada situação seria/ou não recomendada.

O Algoritmo Guloso tem como característica principal obter a maior quantidade de benefício possível a cada recurso executado, para isso desconsidera os tempos de computação exigidos. Portanto, em situações onde os *deadline*s das missões foram folgados, esta heurística apresentou os maiores índices de benefício alcançados. Em contrapartida, em situações onde os *deadline*s das missões eram apertados, apresentou baixo desempenho.

O Algoritmo Preguiçoso apresenta um comportamento oposto ao Algoritmo Guloso. Esta heurística considera o tempo de execução de cada recurso para sua tomada de decisão na definição do itinerário, priorizando sempre os menores tempos de execução. Este algoritmo tem como prioridade concluir a missão da maneira mais rápida possível, não enfatizando o benefício adquirido. Para *deadline*s apertados foi a heurística que obteve o melhor desempenho. Porém, mesmo em situações onde o *deadline* da missão apresentou maior folga, o Algoritmo Preguiçoso sempre obteve o mesmo benefício.

Em função das características apresentadas, entre as heurísticas simples, o Algoritmo Ponderado mostrou-se mais interessante. Porque, diferentemente das heurísticas Guloso e Preguiçoso, não apresentou resultados tão extremos. Esta heurística apresentou os maiores índices de benefício para situações onde o *deadline* da missão era justo e mesmo não fazendo parte do envelope superior com *deadline*s apertados e folgados, apresentou um desempenho satisfatório para estas faixas de *deadline*.

Com respeito aos clones, pode-se afirmar que a meta de se encontrar um algoritmo que apresentasse um comportamento satisfatório para todos os tipos de *deadline* foi atingida com sucesso. O uso de clones foi o primeiro passo em busca de uma heurística flexível.

É interessante encontrar uma relação satisfatória entre o ganho do maior de benefício e o aumento de processamento exigido para se obter este ganho. Por isso, foi limitado o número de agentes lançados por missão. A decisão de se utilizar pares ou trios de agentes clonados vai depender das condições dos nodos e da rede.

Entre os pares de agentes testados, destacaram-se os pares de algoritmos Ponderado/Guloso (Pd-G) e Preguiçoso/Guloso (Pg-G). Comparando apenas estes dois

pares, percebe-se que ambos obtiveram os maiores índices de benefício com *deadline*s folgados. Com *deadline*s justos o primeiro par supera o segundo, mas é importante ressaltar que os índices alcançados pelo par Pg-G são próximos aos do par Pd-G. Com *deadline*s apertados, o par Pg-G obtém os melhores resultados com uma vantagem maior que a apresentada pelo par Pd-G para a outra faixa de *deadline*.

Dois trios de agentes foram testados. Conforme esperado, obtiveram os maiores índices de benefício global médio relativo, uma vez que para compor os trios, foram utilizadas as heurísticas que geralmente formavam o envelope superior dos gráficos de benefício global.

É importante destacar que as heurísticas propostas neste capítulo são sempre de baixo esforço computacional. Esta é uma característica muito importante, pois em sistemas de automação industrial por vezes os nodos visitados pelos agentes possuem pequena capacidade computacional e não suportariam algoritmos complexos para determinação do itinerário.

6.8. Conclusões

Este capítulo propõe heurísticas simples capazes de guiar o agente móvel através do sistema distribuído, conforme as premissas definidas no modelo computacional do capítulo anterior. Todas as heurísticas simples oferecem um baixo custo computacional, algo necessário pois alguns nodos a serem visitados pelo agente móvel podem possuir baixa capacidade de processamento. Além disso, todas as heurísticas simples respeitam a premissa da miopia dos agentes móveis.

No Capítulo4, foi apresentado um estudo da literatura onde foram discutidas propostas que tratassem à questão da mobilidade aliada a questão da restrição temporal. Em (BAEK, 2001a) e (QU, 2005), por exemplo, foram desenvolvidos algoritmos de roteamento para definição do itinerário dos agentes móveis com restrição temporal.

Entretanto, em (BAEK, 2001a) a miopia do agente não é discutida, visto que o caminho a ser percorrido já é conhecido no momento da partida do agente. Em (QU, 2005) em um primeiro momento – no qual vários agentes partem em busca do melhor itinerário –

os agentes móveis podem ser definidos como míopes, contudo, no instante em que o agente móvel parte em busca do cumprimento da tarefa (missão), seu itinerário já está definido.

A avaliação das heurísticas simples foi dividida em três partes. Na primeira fase foram sorteadas uma missão e uma carga com o objetivo de se conhecer e analisar o comportamento das heurísticas simples. Na segunda fase foram sorteadas 7 diferentes missões utilizando a mesma carga. E finalmente, na terceira fase, foram sorteadas 100 diferentes configurações utilizando a mesma carga. Após a análise das várias simulações foi possível descrever o comportamento de cada heurística simples.

Após efetuadas as simulações, pôde-se relacionar cada tipo de *deadline* a uma determinada heurística. Todas as heurísticas simples apresentaram um comportamento bem definido. As heurísticas que formaram o envelope superior, por ordem crescente de agressividade foram: Algoritmo Preguiçoso, Algoritmo Ponderado e Algoritmo Guloso. Entre estas três heurísticas simples que alcançaram os maiores benefícios com *deadline*s distintos, é possível afirmar que a heurística Ponderado destacou-se por apresentar um comportamento menos extremista. Obtendo resultados satisfatórios para as demais faixas de *deadline* – nas quais não obteve os índices mais altos.

Também neste capítulo foi considerada a utilização de agentes clonados, porém com comportamentos distintos. Esta abordagem procura variar o comportamento entre os clones para compensar a miopia do agente. A avaliação realizada mostrou que o desempenho das duplas, em todas as situações, foi superior ao desempenho alcançado com a execução das heurísticas individualmente.

Existe a constante preocupação com o possível *overhead* provocado pela execução dos agentes clones, para isso o número de agentes clones por missão foi limitado. Assim, diminuindo a possibilidade que processamento extra para executar uma missão tenha como conseqüência piorar o desempenho dos demais agentes móveis que utilizam a rede.

Apesar das heurísticas discutidas neste capítulo serem simples, pôde-se observar que elas apresentaram características bem distintas com relação ao seu comportamento frente aos diferentes tipos de *deadline*s. Isto indica que o algoritmo a ser escolhido para uma nova missão deve estar de acordo com o *deadline* da missão.

Já que as heurísticas simples apresentaram um comportamento definido para cada faixa de *deadline*, é possível sugerir o comportamento mais adequado para uma nova missão, uma vez que o *deadline* da missão atual é conhecido. Considerando a possibilidade de se escolher a heurística a ser utilizada a cada nova missão, antes da partida do agente a heurística a ser utilizada pode ser cuidadosamente selecionada levando em conta apenas o *deadline* da missão atual e a experiência adquirida nas missões anteriores.

As heurísticas propostas neste capítulo podem servir de ponto de partida para algoritmos mais complexos. No próximo capítulo são propostas algumas heurísticas adaptativas que são capazes de mudar seu comportamento considerando um histórico de benefícios conseguidos em execuções passadas do agente móvel.

7. Abordagem Adaptativa

7.1. Introdução

Este capítulo apresenta novas maneiras de se tratar a questão da definição do itinerário do agente móvel míope de tempo real. As heurísticas discutidas a seguir são mais elaboradas do que as apresentadas no capítulo anterior, pois são capazes de escolher seu comportamento considerando um histórico de benefícios conseguidos em execuções passadas do agente móvel. Essas heurísticas foram desenvolvidas no decorrer do doutorado e também são uma contribuição desta tese.

O agente usará adaptação na partida. Uma vez escolhido o comportamento para a missão em questão, ele prosseguirá com este comportamento até o final da missão. Para realizar a escolha do comportamento foi utilizada probabilidade condicional baseada nas características do ambiente (sistema distribuído) e nos últimos eventos (histórico).

A seguir, na Seção 7.2, estas heurísticas serão descritas detalhadamente. Na Seção 7.3 será apresentada a avaliação de todas as heurísticas estudadas, dando ênfase às novas abordagens adaptativas. A Seção 7.4 traz as comparações quanto ao desempenho apresentado por estas abordagens. E finalizando este capítulo, na Seção 7.5, são apresentadas as considerações finais.

7.2. Descrição das Heurísticas Adaptativas

A abordagem adaptativa proposta neste capítulo é composta por quatro novas heurísticas, descritas a seguir. Estas heurísticas são baseadas no histórico de execuções anteriores. Por intermédio de consultas a este histórico, o agente escolhe o comportamento para uma determinada missão.

No histórico é mantido, para cada missão passada, o tempo de resposta da missão, o benefício potencial daquela missão (benefício desconsiderando o *deadline*) e a heurística que foi usada. Algumas das heurísticas adaptativas não utilizam todas estas informações. A partir dos dados do histórico vamos definir 3 valores médios de benefícios, os quais podem ser calculados para uma heurística em particular ou para o histórico inteiro. São eles:

•***B*** - benefício médio realmente obtido, considerando o *deadline* de cada missão quando ela foi executada;

•***B^*** - benefício hipotético médio, supondo que todas as missões passadas possuiam um *deadline* igual ao *deadline* da missão corrente;

•***B**** - benefício potencial médio, supondo que todas as missões passadas possuiam um *deadline* infinito.

7.2.1. Heurística Preguiçoso-Guloso - PG

Esta heurística é baseada na utilização de outras duas heurísticas descritas no capítulo anterior: Preguiçoso e Guloso. A heurística constrói dinamicamente um histórico formado apenas pelos tempos de resposta das execuções anteriores (para a formação do histórico desta heurística, os *deadline*s das execuções anteriores e a informação se os *deadline*s foram cumpridos ou não são irrelevantes).

Na sua primeira execução, o agente móvel sempre escolhe o Algoritmo Guloso. Quando surge uma nova missão, seu *deadline* é comparado com os tempos de resposta anteriores armazenados no histórico, e assim estima-se ***P(R £ 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 está acima do limiar, o comportamento Guloso é adotado pelo agente. O algoritmo é apresentado na Figura 7.1.

O valor de ***P(R £ 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.

```
const LIMIAR = 0,9;

percorre o histórico e calcula P(R ≤ D);

se P(R ≤ D) > LIMIAR
   meu_comportamento = guloso;
senão
   meu_comportamento = preguiçoso;

probabilidade_atendidos = 0;
```

Figura 7.1: Algoritmo para tomada de decisão da heurística PG.

Tabela 7.1 apresenta um exemplo de tomada de decisão para a escolha da heurística a ser usada na missão do agente. Na Tabela 7.1, para facilitar a visualização, o histórico é apresentado de forma ordenada, e uma seta representa a posição do *deadline* atual frente ao histórico de tempos de resposta. No entanto, essa ordenação não é necessária para a heurística PG apresentada na Figura 7.1.

Tabela 7.1: Exemplo de tomada de decisão na escolha da heurística na abordagem PG (limiar 0,9).

Execuções	***Deadline* especificado**	**Histórico ordenado**	**Probabilidade de atendimento do *deadline***	**Heurística Usada**	**Tempo de resposta obtido**
1ª	qualquer		-	Guloso	55
2ª	60	55 ↑	1	Guloso	70
3ª	50	↑ 55 70	0	Preguiçoso	35
4ª	45	35 ↑ 55 70	0,33	Preguiçoso	40
5ª	60	35 40 55 ↑ 70	0,75	Preguiçoso	45
6ª	60	35 40 45 55 ↑ 70	0,8	Preguiçoso	40
7ª	70	35 40 40 45 55 70 ↑	1	Guloso	...
...	...	...	...	...	...

O valor escolhido como limiar para a troca de algoritmos entre Guloso e Preguiçoso é uma constante arbitrária (um “número mágico”). 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.

7.2.2. Heurística Memória do Tempo de Resposta - MTR

Esta abordagem pode ser considerada uma extensão da anterior, pois utiliza o mesmo princípio, porém com cinco heurísticas: Guloso, Guloso com Relógio, Ponderado, Ponderado com Relógio e Preguiçoso.

Nesta heurística, a troca de comportamento é novamente baseada nas probabilidades de atendimento do *deadline* atual. As heurísticas estão organizadas respeitando uma hierarquia, onde estão classificadas em ordem decrescente de agressividade, de acordo com como estas obtêm o benefício máximo para uma missão (Figura 7.2). Os comportamentos destas heurísticas foram analisados individualmente na Seção 6.4.

De forma similar à abordagem PG, no MTR o histórico armazena os tempos de resposta observados anteriormente. Os *deadline*s das missões passadas são irrelevantes. Nesta nova heurística, no entanto, o histórico armazena também o tipo de algoritmo que resultou naquele tempo de resposta. Isso permite, dado um *deadline*, determinar a probabilidade de cada algoritmo atender os requisitos temporais da missão atual, ou seja, $P(Ri \leq D \mid Hi)$, $1 \leq i \leq 5$.

Em sua primeira execução, o agente escolherá sempre o Algoritmo Guloso. Nas próximas execuções, o agente analisará no histórico os tempos de resposta obtidos por esta heurística, os comparará com o *deadline* da missão atual e decidirá se pode continuar com o comportamento atual ou se deve mudar para um comportamento menos agressivo. O critério para mudar para um comportamento menos agressivo é a probabilidade de atender o *deadline* atual ser inferior a um limiar arbitrário. O processo é repetido, podendo chegar até a escolha da heurística menos agressiva disponível, ou seja, a heurística Preguiçoso. Assim como na heurística PG descrita anteriormente, a heurística atual utiliza uma constante arbitrária (um limiar) para efetuar as escolhas de heurísticas. O valor escolhido neste trabalho é 90%, e seu valor foi escolhido a partir de experimentos anteriores.

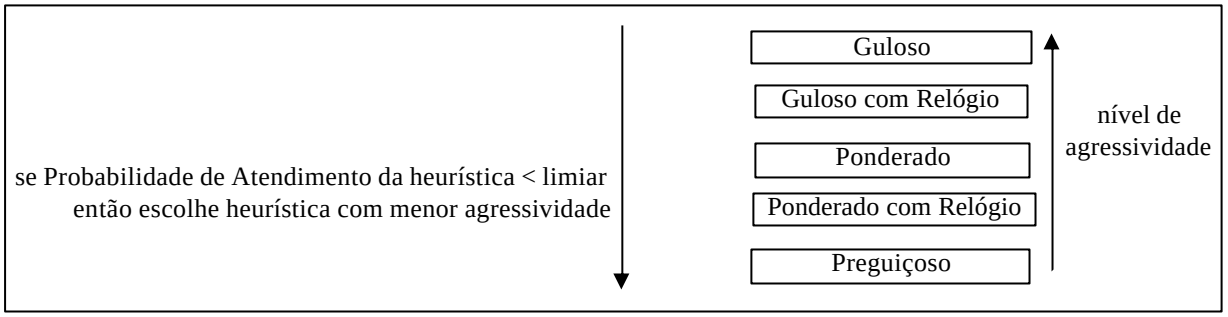


Figura 7.2: Tomada de decisão da heurística MTR.

A Figura 7.3 descreve o algoritmo da abordagem em questão. A complexidade computacional deste algoritmo é ***O(n)+O(H)*** onde ***n*** é o tamanho do histórico e ***H*** é o número de heurísticas. Como ***n*** >> ***H*** temos uma complexidade aproximadamente ***O(n)***.

```
const  LIMIAR = 0,9;
vet[5]
heurística={Guloso,GulosoComRelógio,Ponderado,PonderadoComRelógio,Pregui-
çoso};

percorre o histórico e calcula P(Ri ≤ D | Hi), 1 ≤ i ≤ 5

i = 0;   // começa com Guloso
enquanto i < 4  e  P(Ri ≤ D | Hi) <=  LIMIAR
    i++;    // escolhe comportamento menos agressivo

meu_comportamento =  heurística[i];
```

Figura 7.3: Procedimento para escolha do comportamento a ser utilizado na heurística MTR.

A Tabela 7.2 exemplifica a tomada de decisão do agente móvel na escolha do comportamento a ser seguido. O agente assumirá o comportamento mais agressivo possível para o *deadline* da missão atual. Por exemplo, na 5ª entrada da Tabela 7.2, a missão possui *deadline* 75, e consultando o histórico das execuções passadas, percebe-se que todas as heurísticas obtiveram tempos de resposta menores que 75, portanto estima-se que todas as heurísticas têm chance de cumprir o *deadline*. Seguindo as regras da heurística MTR, o agente opta pelo comportamento Guloso por ser o mais agressivo, com o objetivo de alcançar o maior benefício possível para a situação atual.

Outro exemplo interessante acontece na 9ª execução, onde o *deadline* da missão atual é muito apertado (30) e não é encontrado nenhum registro no histórico de execuções anteriores de alguma heurística que obteve sucesso no passado com este *deadline*. Este

caso ilustra a escolha da heurística menos agressiva e, conseqüentemente, mais rápida, na tentativa de se cumprir o *deadline* atual.

Tabela 7.2: Exemplo de tomada de decisão na escolha do comportamento na heurística MTR (limiar 0,9).

Execuções	***Deadline***	**Probabilidade de atendimento do *deadline***					**Heurística Utilizada**	**Tempo de Resposta**
		G	**GR**	**P**	**PR**	**Pg**		
1ª	50	-	-	-	-	-	Guloso	70
2ª	30	0	-	-	-	-	Guloso c/Relógio	60
3ª	45	0	0	-	-	-	Ponderado	55
4ª	50	0	0	0	-	-	Ponderado c/Relógio	50
5ª	75	1	1	1	1	-	Guloso	68
6ª	55	0	0	1	1	-	Ponderado	60
7ª	55	0	0	0,5	1	-	Ponderado c/Relógio	52
8ª	45	0	0	0	0	-	Preguiçoso	45
9ª	30	0	0	0	0	0	Preguiçoso	40
10ª	60	0	1	1	1	1	Guloso c/Relógio	58
11ª	50	0	0	0	0,5	1	Preguiçoso	...
...	...	...	...	...	...	...	...	...

A análise comportamental desta heurística adaptativa mostra sua robustez. No momento que um comportamento agressivo é escolhido erroneamente, provocando uma perda de *deadline*, seu tempo de resposta é incluído no histórico. O mesmo acontece quando um comportamento não-agressivo é escolhido erradamente. Por conseguinte, na próxima execução com o mesmo *deadline*, será indicado um comportamento menos ou mais agressivo que o anterior, respectivamente.

7.2.3. Heurística Memória do Valor - MV

Esta heurística – da mesma forma que as anteriores – utiliza heurísticas estudadas no Capítulo 6: Preguiçoso, Ponderado com Relógio, Ponderado, Guloso com Relógio e Guloso (podendo também utilizar apenas três heurísticas, descartando as versões com relógio). Nesta abordagem, assim como nas abordagens já vistas neste Capítulo, busca-se maximizar o benefício da missão. No entanto, diferentemente das duas heurísticas adaptativas anteriormente descritas, que utilizavam um valor pré-determinado como limite

("número mágico") para troca de heurística, nesta abordagem utiliza-se probabilidade condicional para determinar o comportamento mais adequado para cada nova missão.

Conforme o modelo computacional definido no Capítulo 5, no passado, cada heurística H_i, $1 \leq i \leq 5$ gerou um benefício hipotético médio $B\hat{}i$, calculado a partir de n_i missões. Se, para todas as heurísticas, n_i for muito grande (por exemplo, $n_i > 100$), então esse problema pode ser resolvido de forma trivial, simplesmente escolhendo-se a heurística H_j com o maior $B\hat{}_j$ entre os comportamentos disponíveis.

No entanto, essa solução simples não pode ser adotada quando o histórico de uma heurística está incompleto ou em formação. Ou ainda, com dados antigos, os quais não refletem mais a realidade das missões executadas, das cargas ou da topologia do sistema computacional distribuído. Ao serem descartados os dados mais antigos, o histórico remanescente pode conter uma pequena quantidade de amostras. Para estes casos, cada $B\hat{}_i$ tem um grau de confiança associado, não sendo possível comparar diretamente as médias.

O benefício obtido por um dado comportamento do agente móvel é uma variável aleatória com média, variância e distribuição desconhecidas. O histórico contém um conjunto de amostras dessa variável. O tamanho da amostra disponível para cada comportamento do agente define um intervalo de confiança para sua média de benefícios, o qual, por sua vez, será usado na heurística MV para definir qual comportamento será adotado pelo agente.

Nesta seção, os métodos estatísticos não estão sendo utilizados para fazer afirmações estatísticas sobre uma população, algo que exigiria rigor na aplicação de tais métodos. Métodos estatísticos estão sendo usados para compor uma heurística para a formação do histórico. O fato de serem utilizados métodos estatísticos apenas para compor uma heurística nos dá liberdade para executar certos passos do método estatístico que seriam questionáveis no caso de amostras muito pequenas (BAR 2004, VIN 1998). Entretanto, mesmo com apenas amostras pequenas disponíveis, é necessário tomar decisões na formação do histórico e a análise estatística é utilizada como base para uma heurística capaz de fazer isso.

É desejável que a heurística usada na formação do histórico apresente as seguintes propriedades:

- Mesmo durante a formação do histórico é dada alguma preferência para comportamentos de agente que tenham gerado o maior benefício global médio até então;

- Com um histórico suficientemente grande, a escolha será sempre daquele comportamento de agente que gera o maior benefício global médio;

- Enquanto o histórico for ainda pequeno, todos os comportamentos de agente têm uma probabilidade não nula de serem escolhidos para uma dada missão;

- A probabilidade de usar um dado comportamento de agente em uma dada missão cresce à medida que aumenta a confiança na hipótese de que aquele comportamento de agente gera o maior benefício hipotético médio entre todos os comportamentos disponíveis.

A definição exata de "histórico pequeno" e "histórico grande" vai depender do sistema em questão. A própria heurística usada na formação do histórico deve ser capaz de perceber o crescimento do histórico e considerar isso em suas decisões.

A partir do histórico existente, essa heurística define para cada comportamento de agente H_i conhecido (isto é, para cada heurística de definição de itinerário conhecida) uma probabilidade $P(H_i)$ daquele comportamento, em particular, ser usado na próxima missão. As cinco diferentes heurísticas que podem ser usadas na MV geram cinco valores usados na escolha do comportamento do agente: ***P(G), P(GR), P(Pd), P(PdR) e P(Pg)***. Essas siglas correspondem as probabilidades de se usar as heurísticas Guloso, Guloso com Relógio, Ponderado, Ponderado com Relógio ou Preguiçoso como o comportamento do agente na próxima missão, respectivamente. Esses valores de probabilidades são calculados de tal forma que:

$$P(G) + P(GR) + P(Pd) + P(PdR) + P(Pg) = 1 \qquad [1]$$

Sempre que for necessário decidir o comportamento de um agente que sai em missão, calculam-se os valores das probabilidades de usar cada comportamento e, então, escolhe-se um deles através do sorteio de um número aleatório com distribuição uniforme entre 0 e 1, respeitando as probabilidades de cada comportamento.

Tabela 7.3: Histórico de Execuções Anteriores.

Heurística utilizada	**Tempo de Resposta**	**Benefício independente do *deadline***
G	70	100
GR	60	100
Pd	55	100
PdR	50	100
G	68	100
Pd	60	100
PdR	52	100
Pg	45	100
Pg	40	100
GR	58	100
Pg	50	100

É definido ***B^i*** como sendo o benefício hipotético médio que a heurística ***Hi*** obteria, caso todas as suas execuções registradas no histórico tivessem sido feitas com o *deadline* atual. Por exemplo, usando os dados da Tabela 7.3 e supondo que o *deadline* atual é 50, tem-se:

B^Guloso = 0/2 = 0

B^Guloso com Relógio = 0/2 = 0

B^Ponderado = 0/2 = 0

B^Ponderado com Relógio = 100/2 = 50

B^Preguiçoso = 300/3 = 100

A determinação da probabilidade de uso para cada comportamento é feita em duas etapas. Inicialmente, os comportamentos são ordenados conforme o benefício hipotético médio obtido por cada um, conforme os dados do histórico e o *deadline* atual. Sem perda de generalidade, é definido como ***H1*** o comportamento com maior benefício hipotético médio, ***H2*** aquele com o segundo maior benefício hipotético médio, e assim por diante. Casos de empate devem ser resolvidos de forma arbitrária.

Os comportamentos são comparados dois a dois conforme a ordem definida anteriormente. Esta comparação estabelecerá entre cada dois comportamentos uma relação de probabilidade de uso entre eles. Tem-se assim a definição dos seguintes valores: ***a=P(H1)/P(H2)***, ***b=P(H2)/P(H3)***, ***c=P(H3)/P(H4)*** e ***d = P(H4)/P(H5).***

A relação entre as probabilidades de uso de duas heurísticas quaisquer ***Hx*** e ***Hy***, onde ***Hx*** com benefício hipotético médio maior que ***Hy***, é definida através de um teste de hipótese. Define-se como hipótese nula que o benefício hipotético médio de ***Hx*** é igual ao benefício hipotético médio de ***Hy***. Testa-se esta hipótese com graus de significância de 0.6000, 0.6500, 0.7000, 0.7500, 0.8000, 0.8500, 0.9000, 0.9500, 0.9600, 0.9750, 0.9800, 0.9900, 0.9950, 0.9975, 0.9990, e 0.9995 (VIN 1998).

Determina-se o valor de ***p***, definido como o maior grau de significância com o qual pode-se rejeitar a hipótese nula. O valor de ***P(Hx)/P(Hy)*** é definido em função de ***p***, ou seja:

P(Hx) / P(Hy) = p / (1-p) caso seja possível rejeitar a hipótese nula com grau de significância ***p***;

ou

P(Hx) / P(Hy) = 1 caso não seja possível rejeitar a hipótese nula.

Uma vez definidos os valores ***{a, b, c, d}*** e considerando a equação **[1],** tem-se um sistema linear com 5 variáveis e 5 equações, o qual pode ser resolvido com facilidade. Assim são determinados os valores de ***P(G)***, ***P(GR)***, ***P(Pd)***, ***P(PdR)*** e ***P(Pg)*** a serem utilizados no sorteio do comportamento que o agente deverá assumir na sua próxima missão.

Assim como nas duas abordagens anteriores, o *deadline* das missões passadas é irrelevante. O histórico da abordagem MV é baseado no tempo de resposta e nos benefícios adquiridos nas missões anteriores. Quando surge uma nova missão, seu *deadline* é comparado aos tempos de resposta anteriores e assim estima-se qual o benefício hipotético médio de cada heurística.

Esta abordagem é robusta porque, ao escolher um comportamento errado, aumenta-se a quantidade dos dados sobre ele, aumentando a probabilidade de se escolher o melhor comportamento no futuro. A Figura 7.4 apresenta o algoritmo para a tomada de decisão da heurística a ser adotada pelo agente.

```
percorre todo o histórico {
   para cada heurística i entre 1 e 5
      calcula o número Ni de aparições de Hi no histórico
      calcula o Benefício hipotético médio B^i
}
percorre todo o histórico {
   para cada heurística i entre 1 e 5
      calcula a variância Si dos benefícios hipotéticos de Hi
}
ordena as heurísticas na ordem decrescente de B^i

// Comparação duas a duas
para j de 1 até 4 faça {
   n1 = Nj;
   n2 = Nj+1;

   S1 = Sj;
   S2 = Sj+1;

   Sa = ((n1-1)*S1 + (n2-1)*S2)/(n1+n2-2);       // variância agregada

   gl = n1 + n2 - 2;       // graus de liberdade

   t_num = B^j – B^j+1;
   t_denum = sqrt(Sa * sqrt( 1/n1 + 1/n2 );
   t = t_num / t_denum;    // Calcula a estatística t

   se( n1 < 2  ||  n2 < 2 ) {
      relation[j] = 1;
      proximo j;
   }
   cl = TStudent.getConfidence( gl, t);
   pH1 = cl + (1-cl)/2.0;
   pH2 = 1.0 - pH1;
   relation[j] = pH1 / pH2;
}
relation1234 = relation[1] * relation[2] * relation[3] * relation[4];
relation234  = relation[2] * relation[3] * relation[4];
relation34   = relation[3] * relation[4];
relation4    = relation[4];

P[H5] = 1.0 /(relation1234 + relation234 + relation34 + relation4 + 1.0);
P[H4] = relation[3] * P[H5];
P[H3] = relation[2] * P[H4];
P[H2] = relation[1] * P[H3];
P[H1] = relation[0] * P[H2];

sorteia x com distribuição uniforme entre 0 e 1
escolhe h conforme as probabilidades P[Hk], 1 ≤ k ≤ 5
meu_comportamento = h;
```

Figura 7.4: Procedimento para escolha do comportamento a ser utilizado na heurística MV.

A heurística apresenta complexidade ***O(n)+O(H.logH)*** onde ***n*** é o tamanho do histórico e ***H*** é o número de heurísticas. Como ***n*** >> ***H*** temos uma complexidade aproximadamente ***O(n)***. A complexidade computacional deste algoritmo é similar àquela da seção anterior, porém sua execução é mais demorada porque exige mais cálculos.

7.2.4. Heurística Chance de Reversão - CR

A abordagem por chance de reversão (CR) utiliza cinco heurísticas: os Algoritmos Preguiçoso, Ponderado com Relógio, Ponderado, Guloso com Relógio e Guloso. Assim como nas abordagens anteriores, apenas o *deadline* da missão e o histórico são conhecidos. Esta abordagem também pode utilizar apenas as heurísticas nas versões sem relógio, desta forma economizando processamento, e diminuindo o número de equações e variáveis.

Esta abordagem parte da idéia da abordagem MV, onde a escolha da melhor heurística a ser utilizada considera o histórico de benefícios alcançados nas execuções anteriores. Primeiramente, são calculados os benefícios hipotéticos médios para cada uma das heurísticas, considerando-se o *deadline* atual, e as mesmas são classificadas em ordem decrescente. Inicialmente, a heurística escolhida é aquela que apresentar o maior benefício hipotético médio para o *deadline* atual. Estes valores são comparados e então é verificada a possibilidade desta classificação ser revertida.

Sem perda de generalidade, as heurísticas são enumeradas baseando-se no benefício hipotético médio, onde ***H1*** representa o comportamento com maior benefício hipotético médio, ***H2*** aquele com o segundo maior benefício hipotético médio, e assim por diante. Caso de empate devem ser resolvidos de forma arbitrária.

Para estimar a chance da heurística que está com maior benefício hipotético médio no histórico $P(H_1)$ ser ultrapassada num futuro próximo por uma das outras heurísticas candidatas $P(H_k)$ – onde k varia de 2 à 5 – os passos são os seguintes:

1° passo) calcular $B^{\wedge}i$, $B^{*}i$ e N_i do histórico;

Onde: $B^{\wedge}i$ é o benefício hipotético médio da heurística i, calculado a partir dos tempos de resposta armazenados no histórico e o *deadline* da nova missão. $B^{*}i$ é o benefício potencial médio da heurística i sem considerar as perdas de *deadline* e N_i é o número de vezes que a heurística i aparece no histórico.

2° passo) classificar em ordem decrescente conforme $B^{\wedge}i$;

Neste passo, a idéia é utilizar a heurística que possua em seu histórico o maior benefício hipotético médio para o *deadline* em questão.

3° passo) comparar cada heurística H_k com $B^{\wedge}_1$, sendo que $B^{\wedge}_1$ representa a heurística que obteve o maior benefício hipotético médio;

- se $N_1 = N_k$? $\frac{P(H_k)}{P(H_1)} = 0$

Se o histórico de H_1 tiver tamanho menor ou igual ao histórico de H_k, definiu-se que a probabilidade de se usar H_k é nula. Caso $B^{\wedge}_1$ esteja muito alto por ser um histórico muito pequeno, a escolha de H_1 fará naturalmente o histórico referente a H_1 crescer.

- se $N_1=0$ ou $Nk=0$? $\frac{P(H_k)}{P(H_1)} = 1$

Se um dos históricos estiver zerado, definimos como igual a probabilidade de usar uma ou outra heurística.

- se $B^{\wedge}_1 = B^{\wedge}_k$? $\frac{P(H_k)}{P(H_1)} = 1$

Se os benefícios hipotéticos médios são iguais, as chances das heurísticas serem escolhidas são iguais.

- se $B^{\wedge}_1 > B^{*}_k$? $\frac{P(H_k)}{P(H_1)} = 0$

Se o benefício hipotético médio $B^{\wedge}_1$ da primeira heurística for maior que o benefício potencial médio (aquele que não considera as perdas de *deadline*) de Hk, isto significa que a probabilidade de usar Hk é nula neste momento. É importante ressaltar que a cada execução o histórico é modificado.

4° passo) estimar qual a taxa de acertos necessária para H_k reverter;

Para reverter a situação (onde o benefício hipotético médio de H_k torna-se superior ao benefício hipotético médio de H_1), assume-se que:

$$FB^{\wedge}_k = B^{*}_1 \,.\, P(R \pounds D \mid H_1)$$

O benefício hipotético médio futuro $FB^{\wedge}_k$ da heurística k é igual ao benefício potencial médio da heurística H_1 multiplicado pela probabilidade desta heurística apresentar valores de tempo de resposta R menores que o *deadline* D. Ou seja, no futuro $B^{\wedge}_k$ será igual ao B^{*}_1 atual. Por outro lado, também é suposto que:

$$FB^{\wedge}_k = B^{*}_k \,.\, FP\,(R \leq D \mid H_k)$$

O benefício hipotético médio futuro da heurística ***k*** é igual ao benefício potencial médio da heurística ***k*** multiplicado pela probabilidade desta heurística apresentar valores de tempo de resposta ***R*** menores que o *deadline* ***D***. Ou seja, no futuro $B^{\wedge}_k$ será igual ao B^{*}_k multiplicado pela probabilidade futura de H_k cumprir ***D***.

Manipulando as equações obtêm-se:

$$FP\,(R \leq D \mid H_k) = \frac{B^{*}_1 \,.\, P(R \leq D \mid H_1)}{B^{*}_k}$$

Como, em geral, $B^{*}_i \,.\, P(R \leq D \mid H_i) > B^{\wedge}i$, este valor é usado como $FB^{\wedge}_i$. Isso acontece porque a probabilidade de cumprir o *deadline* não é independente do benefício da missão. A missão com benefício maior tende a ser mais longa, o que aumenta a chance de perda de *deadline*.

- se $FP(R \leq D \mid H_k) > 1 \;\Rightarrow\; \frac{P(H_k)}{P(H_1)} = 0$ //H_k não tem chance
- se $FP\,(R \leq D \mid H_k) \leq \frac{P(R \leq D \mid H_k)}{P(H_1)} \;\Rightarrow\; \frac{P(H_k)}{P(H_1)} = 1$ //chances iguais

5° passo) calcular o número de acertos em seqüência ***Nx*** necessários para passar de $P(R \leq D \mid H_k)$ para $FP\,(R \leq D \mid H_k)$;

Neste passo é feita a suposição de não ser H_1 a melhor heurística e é dada uma chance às demais heurísticas. Os valores de P guiam a escolha da heurística.

Já existem $N_k * P(R \leq D \mid H_k)$ sucessos em N_k tentativas, sendo que N_k representa o número de vezes que a heurística H_k aparece no histórico. Logo, pode-se ter:

$$FP(R=D|H_k) = \frac{N_k * P(R=D|H_k) + Nx}{N_k + Nx}$$

Sendo ***Nx*** o número de sucessos consecutivos que será preciso para chegar ao valor $FP(R=D|H_k)$. Portanto, N_k+Nx é o número total de tentativas.

Isolando ***Nx***:

$$(N_k+Nx).FP(R=D|H_k) = N_k\,P(R=D|H_k)+Nx$$

$$N_k.FP(R=D|H_k)+Nx.FP(R=D|H_k) = N_k.P(R=D|H_k)+Nx$$

$$Nx - Nx.FP(R=D|H_k) = N_k.FP(R=D|H_k) - N_k.P(R=D|H_k)$$

$$Nx.(1-FP(R=D|H_k)) = N_k.(FP(R=D|H_k) - P(R=D|H_k))$$

$$Nx = N_k . \frac{FP(R=D|H_k) - P(R=D|H_k)}{1-FP(R=D|H_k)}$$

Obviamente, quanto maior a diferença de ***P*** para ***FP,*** maior o ***Nx***.

6° passo) calcular a chance de ***Nx*** acertos em seqüência dado que a real taxa de acertos é ***FP(R£D| H_k)***;

$$P(Nx \text{ sucessos consecutivos}) = FP(R \pounds D \mid H_k)^{Nx}$$

Dado que a probabilidade de um *deadline* ser cumprido é ***FP (R£D | H_k)***, a probabilidade de ***Nx*** *deadline*s cumpridos em seqüência é dada por $FP(R \pounds D \mid H_k)^{Nx}$.

7° passo) calcular a chance de observar o número de acertos em N_k tentativas dado que a verdadeira probabilidade é $FP(R \pounds D \mid H_k)$;

Neste caso, é necessário saber o quanto ***FP*** é realmente possível pois se ***FP*** for muito maior que ***P***, ele será improvável. O objetivo é encontrar a probabilidade de observar o que já está no histórico supondo que ***FP*** seja verdadeiro.

Considerando uma distribuição binomial com parâmetros $p = FP(R \pounds D \mid H_k)$ e $n = N_k$, e utilizando a expressão geral da distribuição binomial, é possível saber a probabilidade de ***x*** sucessos, ***p(x)***, a qual é dada por:

$$p(x) = \binom{n}{x} . P^x . (1-p)^{n-x}$$

onde $x = N_k * P(R \pounds D \mid H_k)$ e indica o número de acertos, ***p*** e ***n*** são os parâmetros da distribuição binomial, e N_k representa o número de tentativas.

O número de combinações entre o número de tentativas ***n*** e o número de acertos ***x*** pode ser calculado pela combinação de ***n***, tomados ***x*** a ***x***:

$$\binom{n}{x} = \frac{n!}{(n-x)!\ x!} \qquad e \qquad x = round(N_k . P(R \pounds D \mid H_k))$$

8° passo) calcular a chance de reverter a situação. Para isso, calcula-se a seguinte equação:

$$P_{reverter} = P\ (Nx\ sucessos\ consecutivos \mid FP(R \leq D \mid H_k)\ .$$

$$P\ (observar\ P(R \leq D \mid H_k) \mid FP(R \leq D \mid H_k))$$

Onde a probabilidade de reverter é definida pela probabilidade do número de acertos consecutivos dada a probabilidade futura de cumprir o *deadline* para a heurística ***k,*** multiplicada pela probabilidade de observar uma certa taxa de sucessos no histórico dada a probabilidade futura de cumprir o *deadline* para a heurística ***k***.

Calcula-se essa relação para cada heurística ***k*** em relação a heurística H_1, que é aquela com $B^{\wedge}$ maior, conhecendo-se assim a chance de utilizá-la. Partindo desta relação são encontradas as probabilidades de utilizar cada heurística disponível:

$$\frac{P(H_k)}{P(H_1)} = \frac{P_{reverter}}{P_{não\ reverter}} = \frac{P_{reverter}}{1 - P_{reverter}}$$

Para o caso de 3 heurísticas, teremos 3 equações e 3 variáveis. Abaixo está ilustrado a resolução do sistema para 3 variáveis.

$$\frac{P(H_2)}{P(H_1)} = x_{21} \qquad \frac{P(H_3)}{P(H_1)} = x_{31} \qquad P(H_1) + P(H_2) + P(H_3) = 1$$

$$P(H_1) + x_{21} * P(H_1) + x_{31} * P(H_1) = 1$$

$$P(H_1) * (1 + x_{21} + x_{31}) = 1$$

$$P(H_1) = \frac{1}{1 + x_{21} + x_{31}}$$

O comportamento desta heurística inclui uma certa robustez. No momento que é escolhida erroneamente uma heurística com histórico pequeno, naturalmente ele será corrigido. Se for escolhida erroneamente uma heurística com histórico grande, acontece um processo decisório onde a heurística de histórico pequeno mantém uma possibilidade de ser escolhida. O algoritmo usado para a tomada de decisão sobre o comportamento a ser adotado pelo agente é apresentado na Figura 7.5. Assim como na heurística adaptativa descrita anteriormente, a complexidade computacional deste algoritmo é $O(n)+O(H.logH)$.

```
percorre todo o histórico {
   para cada heurística i entre 1 e 5
     calcula Ni, B^i, B*i;
}
ordena as heurísticas na ordem decrescente de B^i;

para k entre 2 e 5 faça {
   se  N1 == 0  ||  Nk == 0 {
      relation[k] = 1;
      proximo k;
   }
   se B^1 == B^k {
      relation[k] = 1;
      proximo k;
   }
   se N1 <= Nk {
      relation[k] = 0;
      proximo k;
   }
   se B^1 > B*k {
      relation[k] = 0;
      proximo k;
   }

   FP(R<=D|Hk) = B*1 . P(R<=D|H1) / B*k;

   se  FP(R<=D|Hk) > 1 {
       relation[k] = 0;
       proximo k;
   }
   se FP(R<=D|Hk) <= P(R<=D|Hk) {
       relation[i] = 1;
       proximo k;
   }
   Nx =( Nk . FP(R<=D|Hk) - P(R<=D|Hk) ) / ( 1 - FP(R<=D|Hk) );
   PNx = FP(R<=D|Hk) ** Nx;
   pObs = binomial com n=Nk, x=P(R<=D|Hk), p=FP(R<=D|Hk);
   Pinversão = PNX * pObs;
   relation[k] = pInversão / (1.0 - pInversão);
} //fim do para

P[H1] = 1.0/(1.0+relation[1]+relation[2]+relation[3]+relation[4])
P[H2] = relation[1] * P[H1];
P[H3] = relation[2] * P[H1];
P[H4] = relation[3] * P[H1];
P[H5] = relation[4] * P[H1];

sorteia x com distribuição uniforme entre 0 e 1;
escolhe h conforme as probabilidades P(Hi), 1 ≤ i ≤ 5;
meu_comportamento = h;
```

Figura 7.5: Procedimento para escolha do comportamento a ser utilizado na heurística CR.

7.3. Avaliação das Heurísticas Adaptativas

Nesta seção será apresentada a avaliação – por meio de simulações – do modelo computacional proposto no Capítulo 5 e das heurísticas descritas na seção anterior. Assim como no capítulo anterior, a avaliação das heurísticas será realizada em diferentes fases. Neste capítulo são comparados os desempenhos das abordagens adaptativas em conjunto com as heurísticas simples.

O objetivo desta divisão em fases é observar e descrever separadamente o comportamento das heurísticas adaptativas por meio de sucessivas simulações com diferentes configurações envolvendo situações distintas. As condições para as simulações são as mesmas já descritas no capítulo anterior (Seção 6.3).

A forma como as missões são geradas, através da escolha aleatória do diagrama de recursos e da carga do sistema, dentro de certos limites, caracteriza o histórico como uma amostra aleatória com reposição. O termo "com reposição" significa que existe uma probabilidade não nula de o mesmo diagrama de recursos e a mesma carga na rede serem utilizados em mais de uma missão que compõe o histórico.

7.3.1. Primeira Fase de Avaliação – Tipo Fixo de Missão

Nesta primeira fase de simulação foram sorteadas 8 diferentes missões para a mesma carga, ou seja, para as mesmas condições do sistema e da rede. Para cada missão foram lançados 100 agentes. Isto significa que cada simulação foi repetida 100 vezes para cada *deadline* e o resultado médio foi apresentado. Foram utilizados 14 *deadline*s diferentes, portanto para cada missão ocorreram 1400 execuções por heurística.

O objetivo desta fase de testes com 8 diferentes possíveis caminhos (onde os recursos aparecem em diferentes ordens) é verificar se o comportamento das heurísticas permanece similar mesmo quando os agentes executam diferentes missões. Assim como no capítulo anterior, cada missão do agente é composta por 6 segmentos sorteados a partir de um conjunto de 10 com reposição. Os diagramas de recursos e os grafos dos segmentos de missão podem ser vistos no Apêndice A.

As abordagens MV e CR foram simuladas em suas versões com 3 heurísticas. Esta decisão foi tomada após testes que avaliaram a sensibilidade destas duas abordagens

com relação ao número de heurísticas (comportamentos) utilizadas. Os testes de sensibilidade serão descritos posteriormente (seção 7.3.4).

A Tabela 7.4 mostra o benefício global alcançado por cada algoritmo utilizando as 8 diferentes configurações, com uma mesma carga e variados *deadline*s (apertados, justos e folgados). Diferentemente das simulações do capítulo anterior, a cada missão executada foi se formando um histórico de execuções passadas possibilitando a descrição do comportamento de cada heurística para diferentes situações de restrição temporal. Para essa fase de avaliação o histórico armazenou as últimas 100 execuções. Baseadas neste histórico, as abordagens adaptativas – cada uma seguindo suas métricas e premissas – escolhem a heurística que consideram adequada para uma nova missão que será iniciada.

Tabela 7.4: Benefício Global das heurísticas simples e adaptativas – 8 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Configuração I														
Preguiçoso	0	0	0	0	21	116	169	176	176	176	176	176	176	176
Ponderado_R	0	0	0	0	30	137	175	185	188	189	193	193	194	195
Ponderado	0	0	0	0	0	44	159	203	212	212	212	212	212	212
Guloso_R	0	0	0	0	14	92	162	177	187	188	190	190	193	192
Guloso	0	0	0	0	0	8	89	169	205	212	209	212	212	212
PG	0	0	0	0	21	125	160	168	200	212	212	212	212	212
MTR	0	0	0	0	26	108	167	180	206	212	212	212	212	212
MV	0	0	0	0	12	127	149	196	201	211	211	211	211	211
CR	0	0	0	2	18	117	156	179	179	186	211	211	211	211
Configuração II														
Preguiçoso	0	0	0	19	53	125	138	145	147	147	147	147	147	147
Ponderado_R	0	0	0	0	35	98	148	160	173	173	175	176	176	176
Ponderado	0	0	0	0	15	89	152	178	183	185	185	185	185	185
Guloso_R	0	0	0	0	23	50	119	153	167	180	180	179	183	184
Guloso	0	0	0	0	0	4	54	97	157	181	195	199	199	199
PG	0	0	0	28	68	112	133	136	146	173	193	199	199	199
MTR	0	0	0	21	80	122	141	156	168	182	199	197	199	197
MV	0	0	0	19	62	120	133	155	172	181	192	198	199	199
CR	0	0	0	10	64	108	141	167	177	182	182	194	198	198
Configuração III														
Preguiçoso	0	0	1	16	79	117	135	144	144	143	144	144	144	144
Ponderado_R	0	0	0	7	38	112	147	167	173	171	172	174	174	175

Ponderado	0	0	0	6	42	101	153	165	178	182	182	182	182	182
Guloso_R	0	0	0	0	24	78	140	158	168	175	178	179	180	179
Guloso	0	0	0	0	6	41	109	153	186	186	190	194	194	194
PG	0	0	4	17	55	117	131	148	182	186	194	194	194	194
MTR	0	0	0	20	81	113	131	160	173	192	194	194	194	194
MV	0	0	1	21	56	90	148	170	176	179	190	193	193	191
CR	0	0	0	16	60	90	140	174	182	188	193	193	193	193
Configuração IV														
Preguiçoso	0	0	0	7	84	118	133	133	133	133	133	133	133	133
Ponderado_R	0	0	0	6	60	133	152	158	155	158	157	161	160	159
Ponderado	0	0	0	3	37	122	150	165	167	167	167	167	167	167
Guloso_R	0	0	0	0	4	37	100	136	158	163	164	164	167	167
Guloso	0	0	0	0	0	2	17	68	114	155	169	182	184	184
PG	0	0	0	17	77	118	121	121	127	165	178	178	184	184
MTR	0	0	0	12	75	124	128	163	164	160	163	184	184	182
MV	0	0	0	4	71	121	158	162	161	165	172	178	182	166
CR	0	0	0	7	69	115	154	162	164	165	172	183	183	181
Configuração V														
Preguiçoso	0	0	0	6	83	149	177	188	188	188	188	188	188	188
Ponderado_R	0	0	0	4	45	129	174	207	208	210	207	209	211	213
Ponderado	0	0	0	0	16	113	176	224	224	226	226	226	226	226
Guloso_R	0	0	0	2	18	84	167	196	209	210	211	212	211	213
Guloso	0	0	0	0	5	70	149	192	222	224	226	226	226	226
PG	0	0	0	4	70	147	179	189	215	213	226	226	226	226
MTR	0	0	0	6	78	141	190	196	222	226	226	226	226	226
MV	0	0	0	0	67	139	183	204	222	223	225	225	225	224
CR	0	0	0	4	62	151	172	188	191	226	226	225	226	226
Configuração VI														
Preguiçoso	0	0	5	41	114	147	156	156	156	156	156	156	156	156
Ponderado_ R	0	0	0	13	118	155	160	160	159	159	159	159	159	160
Ponderado	0	0	5	53	114	152	159	160	160	160	160	160	160	160
Guloso_R	0	0	0	0	220	105	149	172	182	188	186	187	188	190
Guloso	0	0	0	0	0	30	100	150	191	199	203	203	203	203
PG	0	0	2	44	119	140	143	162	185	203	203	203	203	203
MTR	0	0	2	33	100	153	160	173	193	183	186	203	203	203
MV	0	0	2	46	111	147	152	160	192	199	201	201	201	202
CR	0	0	0	50	100	151	158	155	192	202	200	202	202	202
Configuração VII														

Preguiçoso	0	0	2	57	138	160	162	162	162	162	162	162	162	162
Ponderado_R	0	0	2	68	132	164	165	164	166	166	166	166	166	166
Ponderado	0	0	3	61	150	161	166	166	166	166	166	166	166	166
Guloso_R	0	0	0	4	32	111	158	178	190	191	194	194	194	194
Guloso	0	0	0	0	0	35	103	161	192	200	206	206	206	206
PG	0	0	0	50	130	146	148	177	199	204	206	206	206	206
MTR	0	0	3	60	136	161	165	180	186	204	206	206	206	206
MV	0	0	3	57	141	160	166	165	184	200	205	204	205	205
CR	0	0	5	61	128	162	163	165	197	203	202	205	205	205
Configuração VIII														
Preguiçoso	0	0	1	11	38	81	93	97	99	99	99	99	99	99
Ponderado_R	0	0	0	4	39	79	109	117	126	129	129	130	130	131
Ponderado	0	0	0	3	23	84	120	134	141	144	144	144	144	144
Guloso_R	0	0	0	0	14	32	90	116	130	136	141	143	140	145
Guloso	0	0	0	0	0	5	33	86	119	150	163	163	165	165
PG	0	0	3	19	50	70	88	96	110	150	163	162	165	165
MTR	0	0	1	15	47	79	98	115	141	153	163	163	165	163
MV	0	0	0	17	35	73	114	95	134	141	157	163	164	142
CR	0	0	1	2	48	79	111	132	136	146	163	164	163	164

Os índices apresentados na tabela acima descrevem o comportamento individual de cada heurística para os diferentes *deadline*s em cada configuração sorteada. Em um primeiro momento, foram analisadas as heurísticas simples. O desempenho apresentado pelas heurísticas simples foram semelhantes à bateria de testes do capítulo anterior, onde para cada faixa de *deadline* uma heurística distinta destaca-se sobre as demais. Novamente, para *deadline*s apertados o Algoritmo Preguiçoso consistentemente mostrou os maiores índices de benefício, seguido pelo Algoritmo Ponderado com Relógio; para *deadline*s justos, o Algoritmo Ponderado representa ser a melhor opção; e para *deadline*s folgados, o Algoritmo Guloso obteve os maiores ganhos.

Em um segundo momento, foram comparados os resultados das heurísticas simples e das abordagens adaptativas. Confrontados esses resultados constatou-se que as abordagens adaptativas procuram se ajustar ao *deadline* da missão, buscando alcançar o maior benefício possível para cada faixa de *deadline*. Por exemplo, na configuração VII o envelope superior (ver gráfico (g) na Figura 7.6) foi composto pelas abordagens

adaptativas em praticamente todas as faixas de *deadline*, com exceção dos *deadline*s 300 à 400.

Analisando ainda a Tabela 7.4, considerando apenas os resultados obtidos pelas abordagens adaptativas, por exemplo, na Configuração II a abordagem PG apresentou os melhores índices de benefício com os *deadline*s apertados e os mais folgados, enquanto que a abordagem MTR apresentou os maiores índices com os *deadline*s justos e em algumas situações com *deadline* folgado. A Abordagem CR também faz parte do envelope superior quando os *deadline*s são justos. Para esta configuração a abordagem MV apresentou os maiores benefícios apenas com *deadline*s muito folgados.

Na Tabela 7.5 são apresentados os valores do intervalo de confiança, com grau de confiança de 95%, para o benefício global de todas as heurísticas referente a cada *deadline* (para as 8 configurações).

Tabela 7.5: Intervalos de confiança das 8 configurações.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Configuração I														
Preguiçoso	0	0	0	0	±11.3	±16.4	±6.8	0	0	0	0	0	0	0
Ponderado_R	0	0	0	0	±13.2	±15.6	± 8.1	± 4	± 2	±1.9	± 1.7	± 1.8	± 1.8	± 1.9
Ponderado	0	0	0	0	0	±17	±18.1	±8.2	0	0	0	0	0	0
Guloso_R	0	0	0	0	±9.5	±17.8	±11.3	± 7.3	± 1.9	± 2	± 2	± 2	±1.9	± 2
Guloso	0	0	0	0	0	±8.2	±20.6	±16.7	±7.1	0	±4.2	0	0	0
PG	0	0	0	0	±11.3	±15.7	±10.6	±12.1	± 8.3	0	0	0	0	0
MTR	0	0	0	0	±12.4	±17	± 8.4	± 6.6	±7.1	0	0	0	0	0
MV	0	0	0	0	± 8.8	±15.7	± 14	± 10	± 8.3	± 1	± 1	± 1	± 1	± 1
CR	0	0	0	± 3.5	±10.4	±16.5	±13.7	± 4.2	± 4.2	± 4.9	± 0.7	± 1.2	± 0.7	± 1
Configuração II														
Preguiçoso	0	0	0	± 9.7	±13.9	±10.3	± 6.9	± 2.9	0	0	0	0	0	0
Ponderado_R	0	0	0	0	±13.3	±16.2	±11.4	± 8.1	± 1.7	± 1.6	± 1.8	± 1.7	± 1.7	±1.5
Ponderado	0	0	0	0	± 9.9	±18.2	± 14	± 7.1	± 3.6	0	0	0	0	0
Guloso_R	0	0	0	0	±11.6	±15.1	±15.8	±11.2	± 7.8	± 1.9	± 1.9	± 1.8	± 1.7	± 1.8
Guloso	0	0	0	0	0	± 5.6	± 17	±19.6	± 16	±11.2	± 5.5	0	0	0
PG	0	0	0	±11.4	±14.4	±12.3	± 8.9	±9.56	±10.6	±10.3	± 5.8	0	0	0
MTR	0	0	0	±10.1	±14.5	±11.4	± 7.2	± 11	± 7	± 5.2	0	± 4	0	± 4
MV	0	0	0	± 9.7	±14.3	±11.5	± 9.4	±12.6	± 8.8	± 5.2	± 5.9	± 1.5	± 0.4	± 0.4
CR	0	0	0	± 7.4	±14.4	± 13	± 9.2	± 11	± 7.2	± 5.2	± 5.3	± 5.6	± 1.5	± 1

Configuração III														
Preguiçoso	0	0	± 2.8	± 8.9	±14.1	±11.1	±6.7	0	0	±2.8	0	0	0	0
Ponderado_R	0	0	0	± 6.6	±13.9	±15.8	±11.8	± 6	± 1.5	± 1.6	± 1.6	± 1.6	± 1.5	±1.6
Ponderado	0	0	0	± 6.1	±15.1	±17.8	±13.1	±10.3	± 5	0	0	0	0	0
Guloso_R	0	0	0	0	±11.6	±16.7	±13.4	±10.5	± 6.9	± 2	± 1.8	± 1.9	± 2	±1.9
Guloso	0	0	0	0	± 6.5	±15.6	± 19	±15.7	± 7.5	± 7.5	± 5.3	0	0	0
PG	0	0	± 4.8	± 9.2	±13.7	±11.1	± 8.8	±10.7	± 9.1	± 7.5	0	0	0	0
MTR	0	0	0	± 9.8	±14.1	±12.2	± 9.4	± 8.8	± 5.3	± 3.8	0	0	0	0
MV	0	0	± 2.8	±10.1	±13.8	±17.4	± 13	± 8.1	± 6.4	± 3.8	± 4	± 1	± 1	± 2.4
CR	0	0	0	± 8.9	±14.6	±17.8	±11.7	± 7.9	± 8.3	± 4	± 1.0	± 1	± 1	± 1
Configuração IV														
Preguiçoso	0	0	0	± 5.7	±12.6	± 8.2	0	0	0	0	0	0	0	0
Ponderado_R	0	0	0	± 5.7	±14.6	±10.3	± 4.6	± 1.6	± 1.5	± 1.5	± 1.6	± 1.4	± 1.4	± 1.4
Ponderado	0	0	0	± 4.6	±13.6	±14.6	± 9.9	± 3.3	0	0	0	0	0	0
Guloso_R	0	0	0	0	± 5	±13.1	±14.9	±11	±5.8	±3.9	±1.7	±1.8	±1.9	±2.1
Guloso	0	0	0	0	0	± 3.6	±10.4	±17.5	±17.6	±13.3	± 9.8	± 3.6	0	0
PG	0	0	0	± 8.8	±12.9	± 8.2	± 8.0	± 8.1	± 9	± 10	± 6.2	± 6.2	0	0
MTR	0	0	0	± 7.5	±13.1	± 6.7	± 6.6	± 4.6	± 4.6	± 5.9	± 3.7	0	0	± 3.6
MV	0	0	0	± 4.5	±13.1	±10.7	± 6.6	± 5.6	± 5.8	± 6.1	± 7.2	± 5.3	± 1.7	± 0.9
CR	0	0	0	± 5.7	±13.3	±12.1	± 8.9	± 5.6	± 4.7	± 4.8	± 6.3	± 1.2	± 1.1	± 2.4
Configuração V														
Preguiçoso	0	0	0	± 6.3	±18.5	±15.1	± 8.8	0	0	0	0	0	0	0
Ponderado_R	0	0	0	± 5..5	±16.7	±19.5	±14.5	± 1.5	± 1.6	± 1.6	± 1.7	± 2	± 1.9	± 1.6
Ponderado	0	0	0	0	±11.3	±22.3	±18.4	±4.4	±4.4	0	0	0	0	0
Guloso_R	0	0	0	±3..9	±11.3	±19.9	± 16	± 8.9	± 1.7	± 1.6	± 1.4	± 1.6	± 1.7	± 1.7
Guloso	0	0	0	0	± 6.2	±20.6	±21.1	±15.9	± 6.2	±4.4	0	0	0	0
PG	0	0	0	± 5.2	±17.9	±15.4	±12.2	±12.9	± 9.7	± 9.7	0	0	0	0
MTR	0	0	0	±6.3	±18.3	±16.4	±10.4	± 9.9	± 6.2	0	0	0	0	0
MV	0	0	0	0	± 18	±16.2	±10.3	± 8.9	± 4.7	± 2.1	± 1	± 1.1	± 1.3	± 1.8
CR	0	0	0	± 5.2	±17.5	± 15	±12.2	± 8.1	± 4.5	± 0.7	± 0.7	± 1.1	± 0.7	± 0.7
Configuração VI														
Preguiçoso	0	0	± 5.2	±13.5	±13.6	± 7.3	0	0	0	0	0	0	0	0
Ponderado_R	0	0	0	± 8.6	±13.7	± 5.5	± 4.8	± 0.9	± 0.4	± 0.5	± 0.4	± 0.5	± 0.4	± 0.8
Ponderado	0	0	± 5.4	±14.8	± 14	± 6.8	± 3.2	± 0.5	± 0.5	± 0.5	± 0.5	± 0.5	± 0.5	±0.5
Guloso_R	0	0	0	0	± 11	±17.5	±13.8	± 9.4	± 3.8	± 1.6	± 1.5	± 1.4	± 1.6	± 1.5
Guloso	0	0	0	0	0	±14.3	± 20	±17.5	± 9.5	± 5.6	0	0	0	0

PG	0	0	± 3	±13.8	±13.1	± 9.2	± 9.6	±11.5	±10.8	0	0	0	0	0
MTR	0	0	± 3.1	±12.5	±14.8	± 6.9	± 6.7	± 8.7	± 8.7	± 5.4	± 4	0	0	0
MV	0	0	± 3.1	±14.2	±14.3	± 7.3	± 5.3	± 0.5	± 8.0	± 4.4	± 1.8	± 1.8	± 2	± 1.3
CR	0	0	0	±14.5	±14.8	± 6.8	± 5.8	± 7.3	± 8.7	± 1	± 4.1	± 1	± 1	± 1
Configuração VII														
Preguiçoso	0	0	± 3.2	±15.2	±11.4	± 3.2	0	0	0	0	0	0	0	0
Ponderado_R	0	0	± 3.2	± 16	± 13	± 3.3	± 0.4	±3.26	± 0.5	± 0.4	± 0.6	± 0.2	± 0.4	± 0.3
Ponderado	0	0	± 4.6	±15.8	± 9.8	± 5.6	± 0.3	± 0.3	± 0.2	± 0.3	± 0.3	± 0.3	± 0.3	± 0.3
Guloso_R	0	0	0	± 4.9	±13.8	±17.8	±13.4	± 9	± 1.4	± 4	± 1.5	± 1.5	± 1.4	± 1.6
Guloso	0	0	0	0	0	±15.2	±20.3	±16.8	±10.3	± 6.9	0	0	0	0
PG	0	0	0	±14.7	±12.7	± 9.6	± 9.9	±12.1	± 7	± 4	0	0	0	0
MTR	0	0	± 4.5	±15.4	±11.7	± 6.5	± 4.8	± 8.2	± 7.7	± 4	0	0	0	0
MV	0	0	± 4.5	±15.3	±11.6	± 5.7	± 0.3	± 3.4	± 9.7	± 5.9	± 1.3	± 1.7	± 1.3	±1.5
CR	0	0	± 5.5	±15.7	±13.3	± 5.6	± 4.6	±10.2	± 8	± 4.1	± 4.3	± 1	± 1.5	± 1
Configuração VIII														
Preguiçoso	0	0	± 1.9	± 6.1	± 9.5	± 7.5	± 4.6	± 2.7	0	0	0	0	0	0
Ponderado_R	0	0	0	± 4.2	±10.9	±11.2	± 7.7	± 5.7	± 2.4	± 2.2	± 2	± 2.1	± 1.9	± 2
Ponderado	0	0	0	± 3.9	±10.4	± 14	± 10	± 7.2	± 4	0	0	0	0	0
Guloso_R	0	0	0	0	± 7.7	±10.8	±11.8	± 8.7	± 4.2	± 2.2	± 2.2	± 2.5	± 2.5	± 2.4
Guloso	0	0	0	0	0	± 5.5	± 13	±16.2	±14.6	± 9.3	± 3.2	± 3.2	0	0
PG	0	0	± 3.3	± 7.6	± 9.7	± 8.8	± 6.1	± 7.5	± 9.3	± 8.2	± 3.2	± 4.5	0	0
MTR	0	0	± 1.9	± 6.9	± 9.7	± 8.3	± 6.5	± 7.0	± 4	± 8.3	± 3.2	± 3.2	0	± 3.2
MV	0	0	0	± 7.3	± 9.5	± 9.7	±11.6	± 4.4	± 6.9	± 4.1	± 5.8	± 1.9	± 1.4	± 1.9
CR	0	0	± 1.9	± 2.7	± 9.9	± 8.7	±11.9	± 7.7	± 6.9	±10.2	± 1.8	± 1.3	± 2.2	± 1.3

Os gráficos da Figura 7.6 apresentam os benefícios globais encontrados com a simulação das 8 diferentes missões: (a) Configuração I, (b) Configuração II, (c) Configuração III, (d) Configuração IV, (e) Configuração V, (f) Configuração VI, (g) Configuração VII; (h) Configuração VIII.

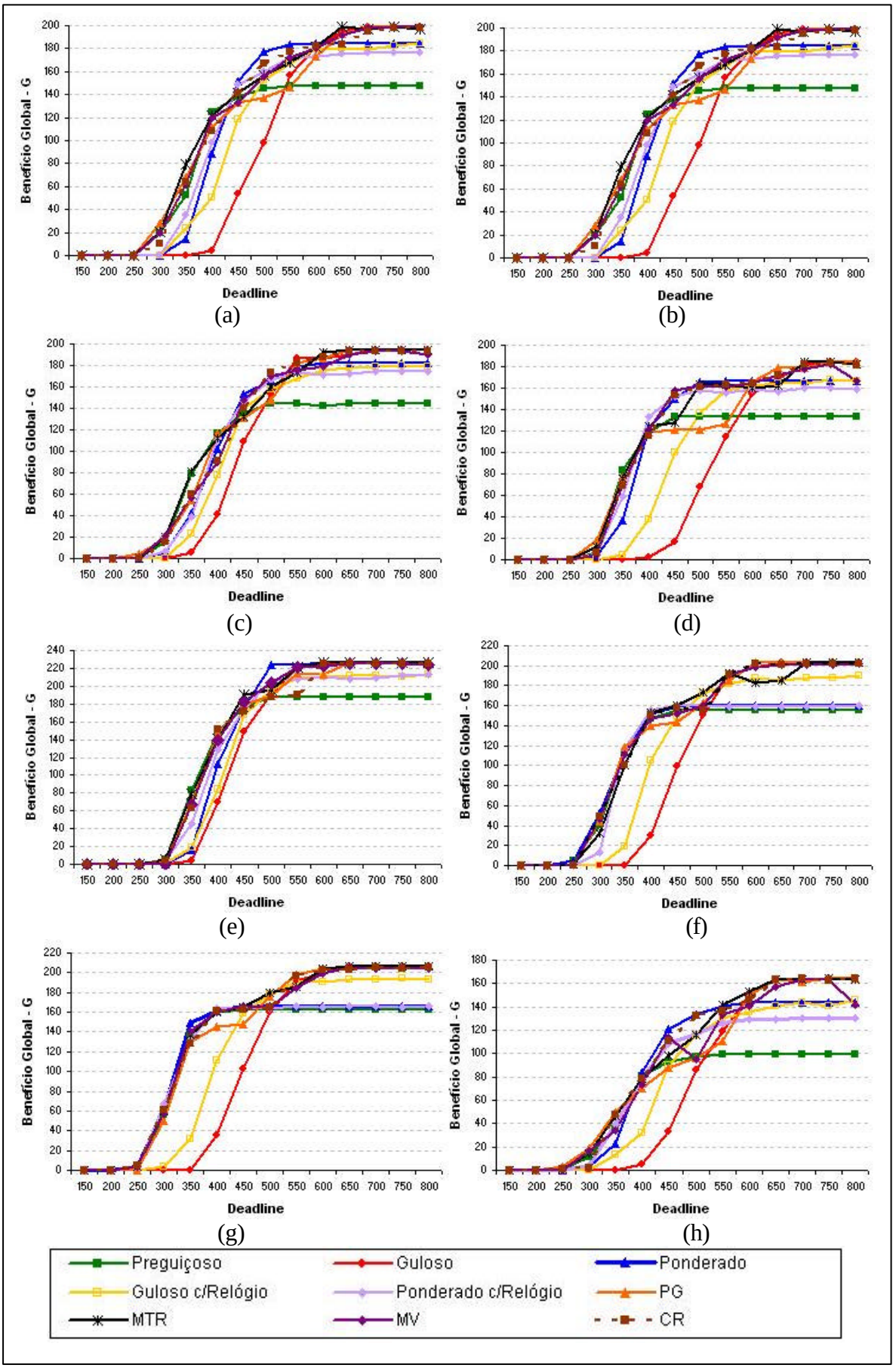


Figura 7.6: Benefício Global das heurísticas simples e adaptativas – 8 configurações.

Observando estes 8 gráficos, percebe-se que as abordagens adaptativas apresentaram bons resultados para qualquer faixa de *deadline*, enquanto que as heurísticas simples apresentaram um comportamento distinto para cada faixa de *deadline*. Este é o maior mérito das abordagens adaptativas.

Os gráficos ilustram os índices mostrados na Tabela 7.4. Algumas considerações relevantes com relação ao comportamento destas heurísticas são:

- As heurísticas simples, individualmente, obtiveram um bom resultado apenas em uma faixa de *deadline*. Por exemplo, o Algoritmo Guloso obteve ótimos resultados com *deadline* folgados e péssimos resultados para *deadline*s apertados;

- A heurística MTR apresentou ótimos resultados fazendo parte, na maioria das configurações simuladas, do envelope superior dos gráficos para todas as faixas de *deadline*;

- A heurística MV mostrou-se flexível e, mesmo não atingindo os índices de desempenho mais altos quando comparadas com as demais abordagens adaptativas, apresentou resultados satisfatórios com relação ao comportamento escolhido para cada situação;

- A abordagem PG, como esperado, não apresentou bons resultados com *deadline*s justos em nenhuma das configurações.

A Tabela 7.6 expõe o percentual de agentes que foram atendidos referentes a cada *deadline*. Entre as heurísticas simples, o Algoritmo Guloso necessitou de um maior *deadline* para atingir 100% de atendidos. As abordagens adaptativas apresentaram resultados próximos, atingindo 100% de atendidos praticamente com o mesmo *deadline*.

Tabela 7.6: Taxa média de *deadline*s atendidos – Configuração I.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	0	0	0	0	0,12	0,66	0,96	1	1	1	1	1	1	1
Ponderado_R	0	0	0	0	0,17	0,75	0,95	0,99	1	1	1	1	1	1
Ponderado	0	0	0	0	0,0	0,21	0,75	0,96	1	1	1	1	1	1
Guloso_R	0	0	0	0	0,08	0,51	0,89	0,96	1	1	1	1	1	1
Guloso	0	0	0	0	0,0	0,04	0,42	0,8	0,97	1	0,99	1	1	1
PG	0	0	0	0	0,12	0,71	0,9	0,89	0,96	1	1	1	1	1
MTR	0	0	0	0	0,15	0,61	0,94	0,97	0,97	1	1	1	1	1

MV	0	0	0	0	0,07	0,72	0,82	0,94	0,96	1	1	1	1	1
CR	0	0	0	0	0,1	0,66	0,84	0,99	0,99	0,99	1	1	1	1

A Tabela 7.7 apresenta os tempos médios de resposta obtidos pelas heurísticas para a Configuração I. A Figura 7.7 exibe o gráfico destes valores.

Analisando o gráfico, nota-se que dentre as heurísticas simples, o Algoritmo Guloso apresentou um maior consumo de tempo, o que comprova que esta heurística necessita de *deadline*s folgados para obter benefícios relevantes. As demais heurísticas simples movem-se alternadamente dentro de um intervalo de tempo com valores de tempo mais baixos. As heurísticas simples, versões com relógio, apresentam um pequeno crescimento no tempo de resposta a medida que o *deadline* aumenta. Esta variação é reflexo da mudança no comportamento destas heurísticas em função do tamanho do *deadline*.

Tabela 7.7: Tempo médio de resposta – Configuração I.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	386,1	379,7	384,2	389	391	391,9	390,2	380,7	390,7	381	388,2	388,8	389,0	381,4
Ponderado_R	383,5	382,0	380,9	388,9	384,7	385,3	400,0	396,7	396,7	403,7	402,9	406,6	412,2	411,7
Ponderado	434,6	432,7	435,4	435,9	432,4	433,6	437,6	436,7	437,1	436,5	432,7	438,3	431,7	434,5
Guloso_R	390,2	393,1	393,3	391,7	397,3	407,4	398,7	412,0	409,6	421,5	415,5	420,4	421,9	422,1
Guloso	456,1	463,3	458,9	455	457,7	466,9	462,2	458,2	462,1	453,7	463,8	462	462,1	462,5
PG	388,4	381,1	382,0	391,4	391,6	383,7	396,2	416,5	460,3	461,3	462,1	454,1	450,3	468,1
MTR	387,9	389,5	388,7	382,6	383,3	394,4	389,8	405,4	458,6	460,9	462,1	466,4	463,8	454,7
MV	400,5	399,5	396,1	406,3	404,7	386,7	402,7	434	453,5	433,1	442,6	438,4	446,2	439,6
CR	421,1	430,3	422,3	428,9	396,3	387,3	405,6	392,5	388,5	404,7	444,6	447,4	453,5	449,8

As abordagens adaptativas apresentaram um tempo de resposta mais baixo para situações com *deadline* apertado e tempo de respostas maiores quando o *deadline* permitia. A heurística Chance de Reversão apresentou os tempos de resposta mais baixos com *deadline*s justos. É importante ressaltar que o tempo de resposta é proporcional ao benefício máximo obtido por cada heurística. Por exemplo, observando a coluna que representa o *deadline* 600 nas Tabelas 7.4 e 7.7, a heurística em questão apresentou o menor tempo de execução e também um benefício mais baixo quando comparada às demais heurísticas adaptativas. Através dos gráficos 7.6(a) e 7.7 é facilmente percebida esta relação.

O gráfico da Figura 7.7 reflete a robustez das abordagens adaptativas onde, diferentemente das heurísticas simples, são capazes de ajustarem-se às condições presentes em busca do melhor benefício possível pra cada situação.

Por exemplo, a abordagem MTR apresentou o tempo médio de resposta R= 387,9 com *deadline* D=150 e apresentou o tempo médio de resposta R= 466,4 com *deadline* D= 700. O que significa que as abordagens adaptativas, percebendo um *deadline* mais folgado para sua missão, permitiram-se utilizar um comportamento que resultasse num tempo de resposta maior, afim de alcançar maiores benefícios. Enquanto que, por exemplo, a heurística simples Preguiçoso utilizou com *deadline*s folgados tempos de reposta muito próximos aos que utilizou em situações em que o *deadline* era apertado. A Tabela 7.7 apresenta estes valores.

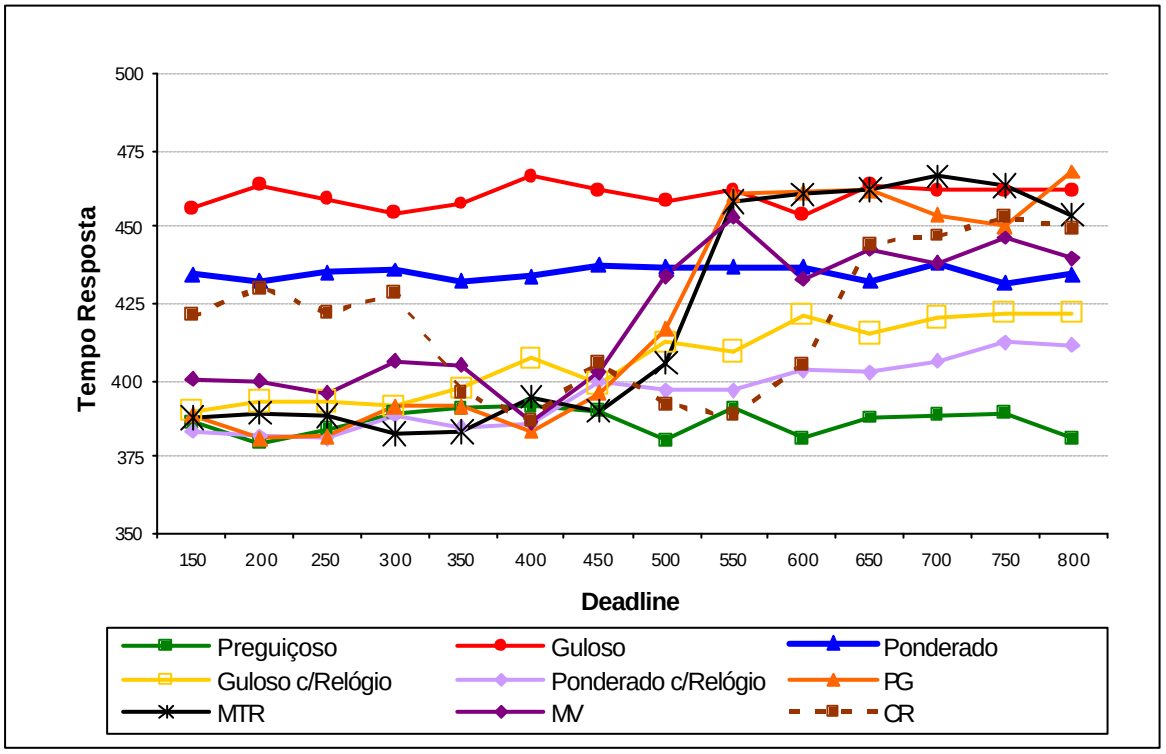


Figura 7.7: Tempo médio de resposta – Configuração I.

Esta seção descreveu o comportamento apresentado pelas abordagens adaptativas para 8 diferentes configurações. É possível que exista sensibilidade das heurísticas quanto ao tipo de grafo. Por este motivo, nas próximas seções (6.4.2, 6.4.3, 6.4.4 e 6.4.5), o desempenho das heurísticas simples e adaptativas será avaliado em cenários mais gerais para que se possa comparar o comportamento destas heurísticas.

7.3.2. Segunda Fase de Avaliação: Vários Tipos de Missão (Histórico Separado)

Nesta fase de avaliação foram simulados 100 diferentes tipos de missão. Em cada missão foram lançados 100 agentes para 14 *deadline*s diferentes, o que resultou em 1400 execuções para cada missão e um total de 140.000 missões utilizando cada heurística.

Foram utilizados históricos separados, ou seja, a cada configuração sorteada, um novo histórico foi formado a medida que apenas missões daquele tipo foram sendo executadas. O histórico é composto pelas últimas 100 execuções efetuadas. Basicamente, repete-se para 100 diferentes tipos de missão o que, na seção anterior, já foi feito para 8 diferentes tipos de missão. Os resultados obtidos para os 100 tipos de missão são apresentados em conjunto.

A Tabela 7.8 apresenta o benefício médio adquirido por cada heurística a cada *deadline* testado. Os valores em negrito indicam os maiores índices de benefício (com variação de 1%) para cada *deadline.* Estes valores formam o envelope superior do gráfico apresentado na Figura 7.8.

A abordagem MTR faz parte do envelope superior na maioria das situações. Examinando a tabela abaixo é possível notar a vantagem em se utilizar uma das abordagens adaptativas, pois elas apresentam bons resultados para qualquer faixa de *deadline*.

Tabela 7.8: Benefício Global das heurísticas simples e adaptativas – 100 configurações (histórico separado).

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	**0,3**	**5,3**	**21,3**	55,8	95,5	121	133,1	136,3	137,8	138,1	138,1	138,2	138,2	138,2
Ponderado_R	0,2	4,2	17,5	50,9	93,9	129	145,5	152,5	155,1	156,4	156,9	157,3	157,9	158,3
Ponderado	0	2,7	14,4	46,1	91,2	128,1	150,5	159,2	162,4	163,4	163,7	163,7	163,7	163,7
Guloso_R	0	0,8	7,2	27,4	65,4	108,1	136,9	155,1	163,3	166,3	168	169,1	170,2	170,6
Guloso	0	0	0,6	8,5	31,5	73,6	117,1	149,4	169,4	**178,8**	**182**	**183**	**183**	**183**
PG	**0,3**	**5,4**	20,9	55,6	93,3	122,4	142,2	158,2	170,6	**178,1**	**181,9**	**182,9**	**183,2**	**183,4**
MTR	**0,3**	**5,4**	**21,2**	**57,2**	**99**	**132,1**	**151,9**	**164,1**	**172,8**	**178,3**	**180,7**	**182,6**	**183,1**	**183,4**
MV	0,2	4,8	20,8	56,5	**98,4**	130,1	**152,3**	**163,9**	**171,2**	175,2	177,9	180,2	179,9	180,7
CR	**0,3**	5,1	20,1	53,6	96,7	129,8	148,6	**162,9**	**172,6**	**177,8**	**181,1**	**182,2**	**182,6**	**182,6**

As abordagens adaptativas têm por objetivo mudar seu comportamento conforme as condições da missão atual. Analisando os índices dos Algoritmos Preguiçoso e Guloso

individualmente, percebe-se que eles obtiveram bons resultados para *deadline*s apertados e folgados, respectivamente. Mas deixam a desejar quando comparados às demais heurísticas simples nas outras faixas de *deadline*. Com o intuito de suprir essa lacuna foi proposta a abordagem Preguiçoso-Guloso, na qual o comportamento do agente é similar ao Preguiçoso nas missões em que o *deadline* é apertado, e nas missões que apresentam um *deadline* folgado, seu comportamento é similar ao Guloso. Os resultados apresentados na tabela acima apontam que o objetivo desta abordagem foi atingido.

Examinando o resultado da abordagem Preguiçoso-Guloso, observou-se que os resultados obtidos com *deadline* justos não são satisfatórios. Percebe-se então, a importância em se elaborar uma abordagem que envolva todas as principais heurísticas simples, assim cobrindo eventuais lacunas. Para isto foram então desenvolvidas as abordagens MTR, MV e CR. Estas abordagens apresentaram bons resultados em situações com *deadline* justo também (ver Tabela 7.8 *deadline*s 350 à 550).

Os números apresentados na Tabela 7.9 definem um intervalo de valores para o benefício médio de cada heurística, com grau de confiança de 95%. Com *deadline*s apertados ou folgados, os valores dos intervalos de confiança tendem a zero, isto acontece porque a maioria dos agentes tende a perder ou a ter todos os seus *deadline*s atendidos, respectivamente.

Tabela 7.9: Intervalos de confiança - 100 configurações (histórico separado).

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	± 0,1	± 0,5	± 0,9	± 1,3	± 1,2	± 0,9	± 0,7	± 0,6	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5
Ponderado_R	± 0,1	± 0,5	± 0,9	± 1,3	± 1,4	± 1,1	± 0,8	± 0,6	± 0,5	± 0,5	± 0,4	± 0,4	± 0,4	± 0,4
Ponderado	± 0,1	± 0,4	± 0,9	± 1,4	± 1,5	± 1,3	± 0,9	± 0,6	± 0,5	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4
Guloso_R	± 0	± 0,2	±0,6	± 1,1	± 1,5	± 1,5	± 1,2	± 0,8	± 0,6	± 0,5	± 0,5	± 0,5	± 0,4	± 0,4
Guloso	± 0	± 0	± 0,2	± 0,7	± 1,3	± 1,7	± 1,7	± 1,4	± 1	± 0,7	± 0,5	± 0,4	± 0,4	± 0,4
PG	± 0,1	± 0,5	± 0,9	± 1,3	± 1,3	± 1,2	± 1	± 0,9	± 0,8	± 0,6	± 0,5	± 0,4	± 0,4	± 0,4
MTR	± 0,1	± 0,5	± 0,9	± 1,3	± 1,3	± 1,1	± 0,9	± 0,7	± 0,6	± 0,5	± 0,5	± 0,4	± 0,4	± 0,4
MV	± 0,1	± 0,4	± 0,9	± 1,3	± 1,4	± 1,2	± 0,9	± 0,7	± 0,6	± 0,5	± 0,5	± 0,4	± 0,4	± 0,4
CR	± 0,1	± 0,5	± 0,9	± 1,3	± 1,4	± 1,2	± 1	± 0,9	± 0,7	± 0,6	± 0,5	± 0,4	± 0,4	± 0,4

O gráfico (a) da Figura 7.8 apresenta o benefício médio adquirido por cada heurística. Sendo que o gráfico (b) apresenta um detalhe dos resultados já apresentados no

gráfico (a). Esta ampliação nesta faixa do gráfico (com B ≥ 100) visa facilitar a visualização do desempenho obtido pelas abordagens adaptativas.

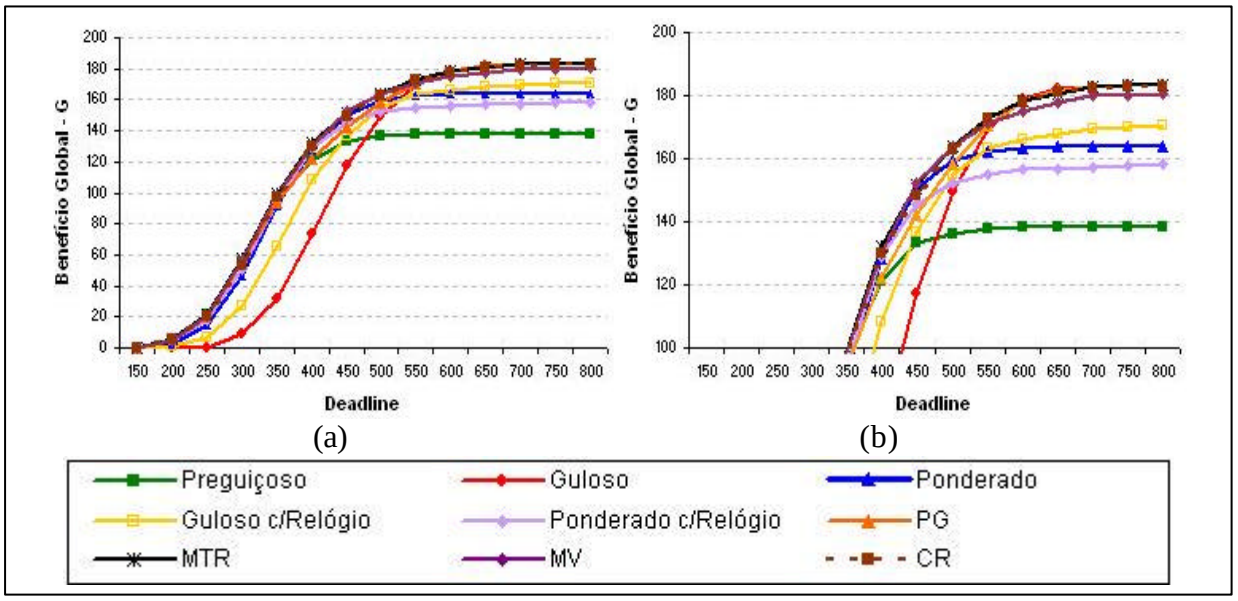


Figura 7.8: Benefício Global das heurísticas simples e adaptativas – 100 configurações (histórico separado).

No gráfico (b), o desvio no comportamento da abordagem PG com *deadline*s justos (neste caso, do *deadline* 350 ao 550) torna-se mais visível. Deve-se considerar também o intervalo de confiança. Para *deadline*s folgados, a abordagem MV obteve índices um pouco mais baixos quando comparados às demais heurísticas adaptativas.

Novamente, como no capítulo anterior (seção 6.4.2), após efetuadas as simulações, pode-se relacionar cada tipo de *deadline* a uma determinada heurística simples. Partindo do *deadline* maior para o menor: o *deadline* ***folgado*** é aquele onde o Algoritmo Guloso, versões com e sem relógio, superam os demais; o *deadline* ***justo*** é definido como aquele onde o Algoritmo Ponderado, versões com e sem relógio, superam os demais; e a partir deste ponto, em direção a *deadline*s cada vez menores, temos os *deadline*s ***apertados***, onde o Algoritmo Preguiçoso obteve o melhor desempenho.

Este comportamento se reflete no uso das abordagens adaptativas, onde a heurística PG, apresentou os índices mais altos de desempenho com *deadline*s apertados e folgados. Enquanto que as heurísticas MTR e MV mantiveram os maiores valores com *deadline* justo, sendo que na faixa de *deadline* de 350 a 550, estas duas abordagens superaram os resultados obtidos com as heurísticas originais.

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*.

O objetivo maior é que se consiga definir uma abordagem capaz de adaptar seu comportamento às diferentes condições, cumprindo os requisitos necessários para se realizar a missão com sucesso.

7.3.3. Terceira Fase de Avaliação: Vários Tipos de Missão (Histórico Misturado)

Na terceira fase de avaliação, foram simuladas 100 diferentes configurações com 14 *deadline*s diferentes. Para cada missão foram lançados 100 agentes. Portanto para cada missão ocorreram 1400 execuções e um total de 140.000 missões utilizando cada heurística.

Foi utilizado o mesmo histórico para todas as configurações. O histórico é composto pelas últimas 100 missões efetuadas.

A Tabela 7.10 apresenta os valores dos benefícios adquiridos para os 14 *deadline*s testados. O uso de um único histórico para todas as configurações refletiu no desempenho das heurísticas adaptativas. Os números em negrito, na tabela abaixo, indicam as heurísticas que formam o envelope superior do gráfico.

Tabela 7.10: Benefício Global das heurísticas simples e adaptativas – 100 configurações (histórico misturado).

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	0,7	**3**	**19,5**	52,4	**85,6**	112,5	125,1	131,1	132,7	133,6	133,1	133,2	132,8	133,3
Ponderado_R	0,7	2,5	14,8	46,6	84,7	**117,8**	**137,1**	147,7	151,9	154,2	155,1	155,6	155,9	156,8
Ponderado	0,6	1,6	10,7	36,8	78,3	115,9	141	**153,4**	**159,4**	162,4	162,7	162,8	162,8	163,2
Guloso_R	0	0,3	5,5	24,4	57,8	96,9	126,2	146,5	156,6	161,4	164,4	165,8	166,7	167,8
Guloso	0	0	1,8	9,2	28,4	61,6	98,8	134,3	155,6	**169,9**	**176,6**	**179,3**	**180,6**	**181,1**
PG	**0,8**	**3,1**	**19,2**	**53,8**	**86,4**	111,9	124,5	132,3	155,2	**170**	**176,5**	**179,5**	**180,4**	**181,1**
MTR	**0,8**	**3**	18,9	53,1	85	112,2	133,4	146,8	155,3	**168,4**	**175,7**	**179,4**	**180,5**	**181**
MV	0,6	2,4	16,9	49,2	83,2	**115,1**	**136,6**	150,3	**158,6**	167,8	**175,8**	**179,3**	**180,4**	**181,1**
CR	0,5	2,1	17,3	50	83,3	114,2	**137,9**	151,5	**158,9**	**168,7**	**176,6**	**179,7**	**180,3**	**181**

Embora o histórico separado tenha apresentado índices de benefícios mais altos, é um cenário menos realista para o modelo computacional em questão, o qual pressupõe o desconhecimento da configuração por parte do agente.

Para análise dos índices acima deve-se considerar o intervalo de valores para o benefício médio de cada heurística. Os números apresentados na Tabela 7.11 indicam essa possível variação nos valores dos benefícios encontrados, com um grau de confiança de 95%.

Tabela 7.11: Intervalos de confiança - 100 configurações (histórico misturado).

Heurística / Dealine	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	± 0,2	± 0,3	± 0,9	± 1,2	± 1,2	± 1	± 0,8	± 0,6	± 0,6	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5
Ponderado_R	± 0,2	± 0,3	± 0,8	± 1,3	± 1,4	± 1,2	± 1	± 0,7	± 0,5	± 0,4	± 0,5	± 0,4	± 0,4	± 0,4
Ponderado	± 0,1	± 0,3	± 0,7	± 1,3	± 1,5	± 1,4	± 1,1	± 0,8	± 0,6	± 0,5	± 0,4	± 0,4	± 0,4	± 0,4
Guloso_R	± 0	± 0,1	± 0,5	± 1,1	± 1,5	± 1,5	± 1,3	± 1	± 0,7	± 0,6	± 0,5	± 0,4	± 0,4	± 0,4
Guloso	± 0	± 0	± 0,3	± 0,7	± 1,2	± 1,6	± 1,7	± 1,5	± 1,2	± 0,9	± 0,6	± 0,5	± 0,4	± 0,4
PG	± 0,2	± 0,3	± 0,9	± 1,2	± 1,2	± 1	± 0,8	± 0,9	± 1,2	± 0,9	± 0,6	± 0,5	± 0,4	± 0,4
MTR	± 0,2	± 0,3	± 0,9	± 1,2	± 1,2	± 1	± 1	± 0,9	± 0,8	± 0,8	± 0,6	± 0,5	± 0,4	± 0,4
MV	± 0,1	± 0,3	± 0,8	± 1,3	± 1,4	± 1,3	± 1,1	± 0,9	± 0,9	± 0,8	± 0,6	± 0,5	± 0,4	± 0,4
CR	± 0,1	± 0,3	± 0,8	± 1,2	± 1,3	± 1,3	± 1,1	± 0,9	± 0,9	± 0,8	± 0,6	± 0,5	± 0,4	± 0,4

Com relação aos resultados apresentados pela Tabela 7.10 e pelos gráficos (a) e (b) da Figura 7.9, onde o gráfico (b) representa um detalhamento nos resultados já apresentados no gráfico (a), algumas considerações interessantes são:

- Se comparados o envelope superior formado com o uso de históricos separados e o envelope superior formado com o uso de um histórico único, percebemos que o uso de um histórico único fez com que as heurísticas simples compusessem o envelope superior. Isto é, as heurísticas adaptativas apresentaram valores próximos aos índices mais altos para todas as faixas de *deadline* (como era esperado), mas não superaram os índices alcançados pelas heurísticas simples, como quando utilizado históricos separados para cada configuração;

- O uso de um histórico único não refletiu no comportamento da abordagem PG, que manteve seu desempenho no mesmo padrão das simulações da fase anterior. Já a abordagem MTR, que com o uso de históricos separados formou o envelope superior em aproximadamente 85% dos *deadline*s, apresentou um desempenho mais baixo com

*deadline*s justos (fazendo parte de aproximadamente 42% dos *deadline*s). Este comportamento é facilmente percebido se observado o gráfico (b) da Figura 7.9;

•As heurísticas adaptativas não fazem parte do envelope superior do gráfico com *deadline*s justos. Os benefícios mais altos, para esta faixa de *deadline*, foram obtidos pelas heurísticas originais. Diferentemente da fase de avaliação anterior (com históricos separados), onde apresentaram índices de benefício mais altos que as heurísticas originais. É importante ressaltar que estes valores são próximos;

•As heurísticas CR e MV apresentaram um melhor desempenho, quando comparadas com as demais heurísticas adaptativas, com o uso do histórico misturado. Entre as abordagens adaptativas, estas duas obtiveram os melhores índices com *deadline*s justos;

•Com *deadline*s mais folgados (D = 700, 750, 800) aconteceu uma homogeneidade entre os benefícios adquiridos pelas heurísticas adaptativas. Todas alcançaram o mesmo benefício empatando com o Algoritmo Guloso. Quando utilizado históricos separados, para esta mesma faixa de *deadline* as abordagens PG e MTR obtiveram os maiores índices;

•Os resultados apresentados e discutidos mostram que as abordagens CR e MV podem ser consideradas mais apropriadas quando for utilizado um único histórico. É importante salientar que existe uma sutil diferença entre os benefícios adquiridos e, neste caso, a abordagem CR apresentou melhores índices na maior parte dos *deadline*s.

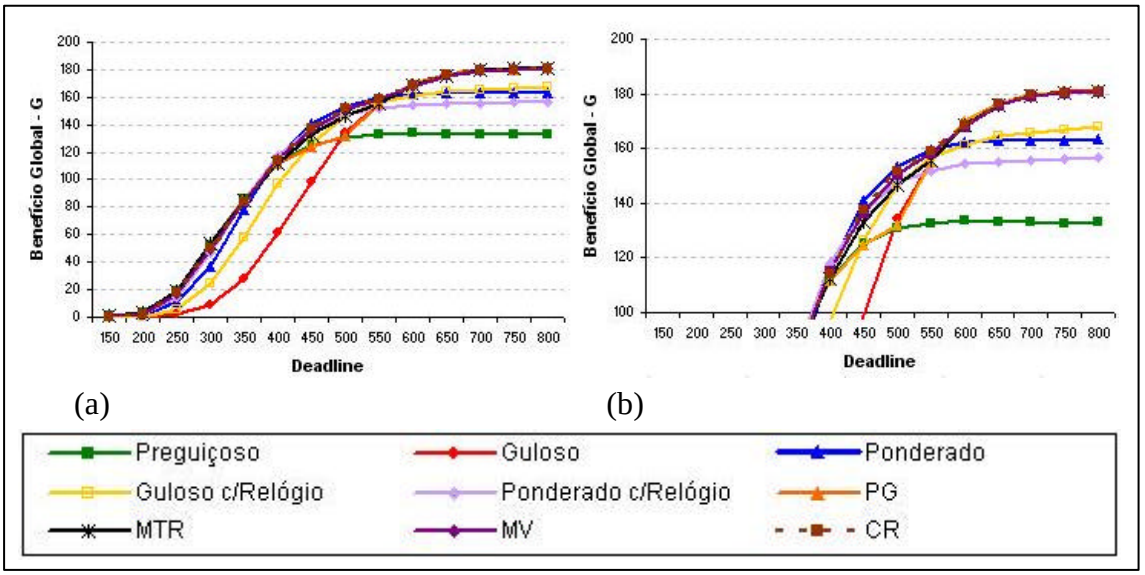


Figura 7.9: Benefício Global das heurísticas simples e adaptativas – 100 configurações (histórico misturado).

7.3.4. Quarta Fase de Avaliação: Análise de Sensibilidade

Nesta quarta fase de avaliação foram examinadas possíveis mudanças no comportamento apresentado pelas abordagens adaptativas com relação à sensibilidade a algumas questões, como: ao número de heurísticas simples que compõe as heurísticas adaptativas, ao tamanho do histórico, à concentração de benefício em um único recurso, e ao tipo e ao tamanho do grafo formado a cada missão.

Para as 4 subseções a seguir, foram simuladas 100 diferentes configurações (missões), com 14 *deadline*s, para cada simulação foram lançados 100 agentes. Portanto para cada missão ocorreram 1400 execuções e um total de 140.000 missões utilizando cada heurística. Foi utilizado um único histórico para todas as configurações. O histórico é composto pelas últimas 100 execuções efetuadas, com exceção da subseção B, onde é verificada a sensibilidade ao tamanho do histórico.

a)Sensibilidade ao Número de Heurísticas que Compõem a Abordagem Adaptativa

Nesta subseção, o objetivo das simulações é comparar o desempenho das abordagens MV e CR utilizando as suas duas variações, ou seja, utilizar 3 ou 5 heurísticas.

Baseados nos resultados apresentados na Tabela 7.12 e na Figura 7.10 pode-se assegurar que as versões com 3 heurísticas são tão eficientes quanto as versões com 5. As heurísticas versões com relógio de certa forma são adaptações limitadas, e neste caso, não somam valor às três outras heurísticas já disponíveis, para adaptação na partida do agente.

Os valores em negrito, na Tabela 7.12, indicam os benefícios mais altos para cada *deadline*. As comparações aconteceram entre as 1ª e 2ª linhas e 3ª e 4ª linhas. As duas primeiras linhas da tabela indicam os valores para a abordagem MV e as duas últimas indicam os valores para a abordagem CR.

Quando confrontados os resultados, percebeu-se que a diferença entre os valores apresentados pelas abordagens em suas diferentes versões foi muito pequena, mas apresentando em ambos os casos, vantagem para a versão com 3 heurísticas. A abordagem CR, por exemplo, atingiu índices mais altos (mesmo que próximos) em todos os *deadline*s.

Tabela 7.12: Benefício Global das abordagens MV e CR utilizando 3 ou 5 heurísticas.

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
MV_3	0,6	**2,9**	15,3	**50,8**	**86,8**	117,4	**143,1**	**155,4**	**164,1**	**175,8**	**181,2**	**183,6**	**184,3**	**184,7**
MV_5	**0,7**	2,5	**15,4**	48,1	85,4	**117,9**	141,6	153,9	163,2	174,8	180,6	182,9	183,9	184,3
CR_3	**0,3**	**2,3**	**16,4**	**51,6**	**87,4**	**117,7**	**144,7**	**156**	**164,6**	**176,8**	**182**	**183,7**	**184,5**	**184,8**
CR_5	**0,3**	1,8	13,9	46,8	84,6	116,8	142,4	155,3	163,7	175,9	181,1	183,4	184,2	184,7

A Tabela 7.13 mostra os valores que compõem o intervalo dos benefícios, com grau de confiança de 95%. Pelo intervalo de confiança pode-se verificar o intervalo dos possíveis benefícios, confirmando assim que é vantajoso utilizar as versões com 3 heurísticas, já que os intervalos são praticamente os mesmos.

Tabela 7.13: Intervalos de confiança das abordagens MV e CR utilizando 3 ou 5 heurística.

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
MV_3	± 0.1	± 0.3	± 0.8	± 1.2	± 1.3	± 1.2	± 1.1	± 0.9	± 0.9	± 0.8	± 0.6	± 0.4	± 0.4	± 0.4
MV_5	± 0.1	± 0.3	± 0.8	± 1.2	± 1.3	± 1.3	± 1	± 0.9	± 0.9	± 0.8	± 0.5	± 0.5	± 0.4	± 0.4
CR_3	± 0.1	± 0.3	± 0.8	± 1.3	± 1.4	± 1.3	± 1	± 0.9	± 0.9	± 0.7	± 0.5	± 0.5	± 0.4	± 0.4
CR_5	± 0.1	± 0.3	± 0.8	± 1.2	± 1.4	± 1.3	± 1	± 0.8	± 0.9	± 0.8	± 0.6	± 0.5	± 0.4	± 0.4

A Figura 7.10 evidencia a proximidade dos resultados discutidos anteriormente. A vantagem em se utilizar as versões com 3 heurísticas (além de terem obtido resultados levemente melhores) está na economia de cálculos (efetuando cálculos mais simples) no algoritmo, por possuir apenas 3 opções de troca para tomada de decisão sobre qual comportamento seguir, enquanto as demais possuem 5 opções.

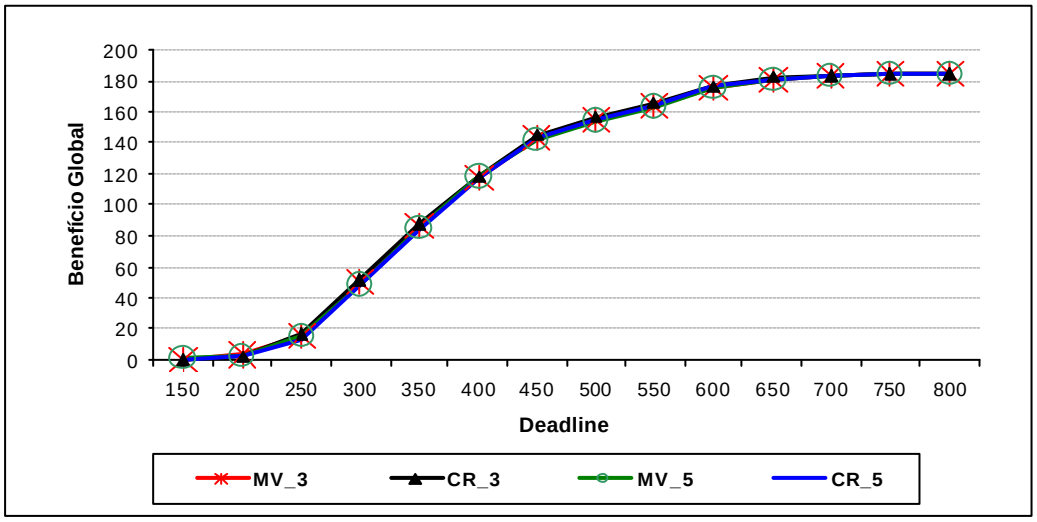


Figura 7.10: Benefício Global das abordagens MV e CR utilizando 3 ou 5 heurísticas.

b) Sensibilidade ao Tamanho do Histórico

Uma questão importante é o tamanho mínimo para o histórico poder ser considerado confiável. Um histórico maior implica em um maior gasto de memória para armazená-lo, e também em um maior tempo de execução para todas as heurísticas (para percorrer o histórico), tanto pior quanto mais complexa for a heurística.

Do ponto de vista do *overhead*, quanto menor o histórico tanto melhor. Se um histórico maior resultar em benefícios maiores, valerá a pena esse *overhead* associado.

A Tabela 7.14 apresenta os benefícios adquiridos pelas heurísticas adaptativas. Foram verificados o desempenho das heurísticas adaptativas utilizando os seguintes tamanhos de históricos: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, e 1000.

Tabela 7.14: Benefício Global com relação à sensibilidade ao tamanho.

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Tamanho Histórico = 10														
PG	0,1	3,6	13,1	41	75,3	103,9	123	133,4	148	164,5	176,6	182	184,1	184,9
MTR	0,1	3,7	13,5	40	75,7	103,6	126	143,4	158,4	171	178,7	183,3	184,9	185,1
MV	0	2,4	9,4	31,1	63,2	99,2	126,1	145,4	157,5	168,4	174	176,1	177,2	177,6
CR	0	1,5	8,1	31,8	68,1	102,7	128,1	146,7	160	169,5	176,2	179,2	180,5	180,8
Tamanho Histórico = 20														
PG	0,1	3,6	13,2	41,4	75,9	104,6	124,1	133	151,2	170,4	179,7	183,4	184,8	185,2
MTR	0,1	3,9	12,6	41,4	75,4	105,7	128,3	147,1	158,9	169,8	178,7	182,8	184,3	185,1
MV	0	2,6	10,9	34,5	69,3	103,5	128,7	148,7	160,2	170,1	177,1	180,3	181,9	182,1
CR	0,1	1,8	9,3	35,4	71,7	105,7	129,9	149,6	160,1	170,8	177,9	181,4	183,4	183,5
Tamanho Histórico = 30														
PG	0,1	3,4	12,9	40,9	75,6	105,1	123,9	132,9	152	172,9	180,5	183,7	185	185,1
MTR	0,1	3,7	13,2	41,5	75,6	105,5	130,5	147,5	160,2	171,1	178,4	182,6	184,7	185
MV	0,1	2,6	10,7	36,2	69,9	105,4	131,9	149,7	160,3	170,4	177,3	181,2	183,1	183,5
CR	0	1,9	10,2	37,2	71,7	105,6	133,4	150,8	161,3	171,2	178,4	182,1	183,8	184,4
Tamanho Histórico = 40														
PG	0,1	3,7	12,3	41	76,3	106	124,8	132,8	153,7	172	180,6	183,6	185	185,1
MTR	0,1	3,6	12,9	41,7	76,3	105,9	129,9	149	159,1	170,4	179,1	183,1	184,7	185
MV	0,1	2,8	10,7	35,1	70,8	104,6	132,6	150,6	161	170,3	177,6	181,9	183,6	184
CR	0	2,2	10,6	37	72,4	104,7	134,4	152,2	161,9	171,8	179,1	182,6	184,3	184,8

Tamanho Histórico = 50														
PG	0,1	3,7	12,7	41,4	75,8	105,2	124,7	132,8	154,2	173,9	180,6	183,8	185	185,1
MTR	0,2	3,8	12,7	41,9	76	106,6	129,7	149,5	159,1	171,1	179,9	183,2	184,7	185,1
MV	0,1	3	10,8	36,8	71,9	106	133,4	151,4	161,7	170,5	178	181,8	183,8	184,1
CR	0,	2,2	10,7	37,7	72,4	106,6	134,8	153,6	161,9	171,7	179,1	183,3	184,6	184,9
Tamanho Histórico = 60														
PG	0,1	3,7	13,1	41,6	76,4	105,8	124,5	132,6	155,2	172,6	180,3	183,5	185,1	185,2
MTR	0,2	3,6	12,5	41,1	76,6	104,8	130	148,9	159,3	170	179,6	183,5	185	185,2
MV	0,1	2,8	10,9	36,7	71,9	106,7	133,9	152,4	161	171,7	178,3	182,2	184,2	184,5
CR	0	2,3	10,5	38,3	72,8	106,8	135,6	153,5	163	171,9	180	183,2	184,7	185
Tamanho Histórico = 70														
PG	0,1	3,6	13,2	41,5	75,9	105,3	124,7	132,7	155	173,5	180,3	183,7	184,9	185,1
MTR	0,1	3,6	13,3	41	75,4	105,8	130,1	149,6	159,4	170,5	179,2	183,5	184,9	185,1
MV	0	3,2	10,8	37,2	72,1	106,4	134,7	152,4	161,4	171,8	179	182,5	184,5	184,4
CR	0	2,2	11,5	38,8	73,8	107,8	136,4	154,2	161,7	171,7	179,5	183,5	1851	185,2
Tamanho Histórico = 80														
PG	0,2	3,7	13,2	41,5	75,7	105,3	124,1	133,4	156,2	173	181	184	184,9	185,2
MTR	0,2	3,4	12,9	41,1	75,4	105,4	129,5	149,3	158,9	170,2	179,3	183,4	184,8	185
MV	0,1	2,9	11,2	38,2	71,4	106,4	135,9	152,9	161,6	172,1	178,8	182,6	184,3	184,8
CR	0	2,4	11,5	38,8	72,2	105,7	136,1	154,8	162,3	171,4	179,9	183,3	184,5	185,1
Tamanho Histórico = 90														
PG	0,1	3,4	13	40,7	75,2	104,5	124,3	133,1	157,7	173,4	180,4	183,6	184,9	185,3
MTR	0,1	3,3	13	41,2	75,4	104,8	130,1	150	159,1	170,6	179,4	183,5	184,7	185,1
MV	0,1	2,7	11,5	37,8	71,9	106,5	135,1	153,7	162,1	171	179,5	183,2	184,4	184,8
CR	0	2,6	11,4	38,4	73,6	106,8	136,6	154,9	162,1	171,3	180,1	183,7	184,8	185,2
Tamanho Histórico = 100														
PG	0,1	3,3	12,6	41,8	75,7	104,5	125,1	132,8	155,8	173,6	180,4	183,5	184,9	185,2
MTR	0,1	3,5	13,2	40,8	75,6	105,4	130,3	148,5	159,1	170,5	179,7	183,6	184,7	185,2
MV	0,1	2,8	11,5	38,8	72,3	106,9	136,2	153,1	161,7	171,4	179,8	182,8	184,5	184,9
CR	0,1	2,5	11,4	39,4	73,2	106,7	136,4	155,6	162,4	171,7	179,9	183,5	184,9	184,9
Tamanho Histórico = 200														
PG	0,1	3,6	12,9	41,3	76	104,8	124,5	133	158,8	173,5	180,4	184	185	185,1
MTR	0,1	3,6	13	40,8	75,6	105,2	130,1	149,5	159	171	179,7	183,3	184,9	185,1
MV	0,1	3,1	11,7	38,1	73,7	107	137,6	156	162,6	171,1	179,3	183,5	185,1	185,2
CR	0,1	3,2	12,3	39,6	74,8	106,1	139,3	157,2	163,3	172,6	180,8	183,4	185	185,1
Tamanho Histórico = 300														
PG	0,1	3,7	12,7	42	75,9	105,4	124,6	132,7	158,7	174,2	180,6	183,6	185	185,2
MTR	0,1	3,7	13,2	41,2	75,2	104,5	129,3	149,2	158,8	171,6	179,9	183,6	184,7	185,1

MV	0,1	3,1	12	39,8	73,9	106,4	138,5	156,5	162,4	172,2	180,1	183,4	184,8	185,3
CR	0	3,2	12,5	40,5	73,5	107,7	139	157,5	162,7	173,4	180,5	183,6	185,1	185,1
Tamanho Histórico = 400														
PG	0,1	3,5	12,8	41,8	76,2	105,3	124,6	132,9	159	172,7	180,5	183,8	185,1	185,2
MTR	0,1	3,9	13,2	41,8	76,2	105,2	128,4	148,5	158,6	171,9	179,8	183,5	184,8	185,2
MV	0,1	3,1	12	40,4	74,9	106,9	138,9	156,6	162,9	172,3	180,4	183,6	185	185,1
CR	0,1	3,3	12,4	41	74,6	107,3	138,8	157,2	162,7	172,8	180,4	183,6	184,8	185,2
Tamanho Histórico = 500														
PG	0,1	3,7	12,9	41	75,5	105,4	124,9	132,9	158,1	174	180,1	183,6	184,8	185,1
MTR	0,1	3,7	13,34	41	75,2	105	128,6	149,2	158,1	171,7	180,2	183,5	185	185,1
MV	0,1	3,2	12,4	40,2	74,4	106,4	139,5	157,1	162,7	172,6	180,4	183,6	185,2	185,1
CR	0	3,3	12	40,8	73,8	107	139,7	157,2	162,9	172,7	180,7	183,8	184,9	185,1
Tamanho Histórico =600														
PG	0,1	3,7	13	40,8	76,5	104,8	124,2	133,1	158,7	173,2	180,8	183,8	185	185,2
MTR	0,1	3,6	13,1	41,5	76	104,9	128,1	150	159	171,9	179,7	183,5	185	185,3
MV	0,1	3,3	12,7	40,1	74,1	106,9	139,3	157,2	162,7	173	180,4	183,7	185,1	185,2
CR	0	3,6	13	41,5	74,1	106,6	139,1	157,9	163,5	173,7	180,5	183,5	184,9	185,2
Tamanho Histórico = 700														
PG	0,1	3,3	12,9	41,4	75,9	106	123,6	133,2	158,4	173,2	180,3	183,2	185,2	185,2
MTR	0,2	3,5	13,4	41,3	75,3	104,3	127,9	148,7	159,2	171,9	180,4	183,3	184,8	185,1
MV	0,1	3,3	13	41,5	74,3	107,4	138,7	157	162,6	173,1	180,4	183,8	184,9	185,2
CR	0,1	3,5	13,1	41,4	76	107,9	140,7	157,3	163,3	172,7	180,5	183,6	185,1	185,2
Tamanho Histórico = 800														
PG	0,2	3,6	12,8	42,2	75,8	104,5	124	132,9	158,9	173,5	180,2	183,9	184,9	185,3
MTR	0	3,5	13,1	41,9	76	104,1	127,3	149,9	157,7	172,4	180,4	183,5	184,8	185,2
MV	0,1	3,5	12,7	41	74,6	107,1	139,4	156,8	163,1	173,3	182	183,4	184,9	185,1
CR	0,1	3,4	12,6	41,9	74,4	107,1	139,3	157,3	163,2	172,9	180,6	183,6	185,1	185,2
Tamanho Histórico = 900														
PG	0,1	3,4	12,8	41	76	106	125	133,5	157,6	172,8	180,6	183,4	1851	185,1
MTR	0,1	3,3	12,8	41,4	75,6	104,8	127,7	149,3	158,5	172,9	180,8	183,5	184,9	185,2
MV	0,1	3,4	12,8	41	75,9	107,9	139,6	157	163	172,8	180,3	183,6	185	185,2
CR	0,1	3,4	12,7	41	74,6	107,3	138,9	156,5	163,7	173,6	180,6	183,5	185	184,9
Tamanho Histórico = 1000														
PG	0,1	3,7	13,4	41,9	75,5	105,7	124,2	133,5	158,1	173,5	180,7	183,9	185	185,1
MTR	0,1	3,4	13,5	41,6	74,9	105,7	127,2	150	158,4	171,9	179,1	183	184,9	185,1
MV	0,1	3,4	13,2	42,2	74,7	106,4	139,9	157,4	163	173	180,7	183,8	184,9	185,2
CR	0,1	3,6	12,7	42	75,3	107,4	139,1	157,5	163,7	173,3	180	183,6	184,9	185,3

A Tabela 7.15 apresenta os benefícios médios obtidos por cada heurística adaptativa para a faixa de *deadline*s de 250 à 700. Os demais *deadline*s 150, 200, 750 e 800 foram desconsiderados por apresentarem valores sempre próximos de zero ou próximos do máximo para todas as heurísticas. *Deadline*s muito apertados, como o 150 e 200, tenderiam a uma média menor que a real, e o inverso acontece com *deadline*s muito folgados, como o 750 e 800.

Tabela 7.15: Média do benefício alcançado com relação à sensibilidade ao tamanho do histórico.

Heurística / Tam Hist	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
PG	116,1	117,7	117,91	118,3	118,5	118,6	118,6	118,7	118,6	118,6
MTR	119,4	120,1	120,6	120,7	121	120,6	120,8	120,5	120,7	120,7
MV	115	118,3	119,2	119,5	120,2	120,6	120,8	121,1	121,2	121,4
CR	117	119,2	120,2	120,7	121,2	121,6	121,9	121,6	121,9	122
	100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
PG	118,6	118,9	119	118,9	118,8	118,9	118,8	118,8	118,8	119
MTR	120,7	120,7	120,6	120,7	120,6	120,8	120,6	120,6	120,7	120,5
MV	121,4	122	122,5	122,9	122,9	123,1	123,2	123,2	123,4	123,4
CR	122	122,9	123,1	123,1	123,1	123,3	123,6	123,3	123,2	123,4

Baseando-se nos dados apresentados na Tabela 7.15, os gráficos da Figura 7.11 apresentam o desempenho médio alcançado pelas heurísticas adaptativas. O gráfico (a) apresenta o benefício médio adquirido por cada heurística com históricos de tamanho 100 à 1000. O gráfico (b) apresenta mais detalhadamente os resultados obtidos com históricos de tamanho até 100. Esta ampliação nesta faixa do gráfico (com tamanho do histórico ≤ 100) visa facilitar a visualização do desempenho obtido pelas abordagens adaptativas.

Analisando o gráfico (a) da Figura 7.11 e os dados apresentados na Tabela 7.15, pode-se perceber que as heurísticas PG e MTR apresentaram uma variação nos valores de benefício alcançados menor que 1%. A heurística MV apresentou, para as mesmas situações avaliadas, uma variação de aproximadamente 1,5%, enquanto que a heurística CR também obteve uma variação de aproximadamente 1%. Está análise indica que um histórico de tamanho superior a 100 é desnecessário, causando um possível *overhead* que pode ser evitado.

Com o objetivo de estabelecer um tamanho de histórico consistente que seja confiável e evite armazenamento desnecessário, conseqüentemente economizando processamento, as heurísticas adaptativas foram avaliadas utilizando históricos de tamanho 10 à 90. Possibilitando assim que fosse definido com precisão o tamanho ideal para o histórico de execuções anteriores. Observando o gráfico (b) da Figura 7.11 percebe-se que existe um crescimento no benefício à medida que o tamanho do histórico aumenta até determinado tamanho para todas as heurísticas. Partindo dos resultados obtidos pelas avaliações realizadas, percebeu-se que o histórico com tamanho entre 50 e 60, ou seja, contendo de 50 à 60 execuções passadas, foi suficiente para garantir que as heurísticas adaptativas pudessem escolher o melhor comportamento para a missão atual. Lembrando que a cada nova missão somente é conhecido seu *deadline*.

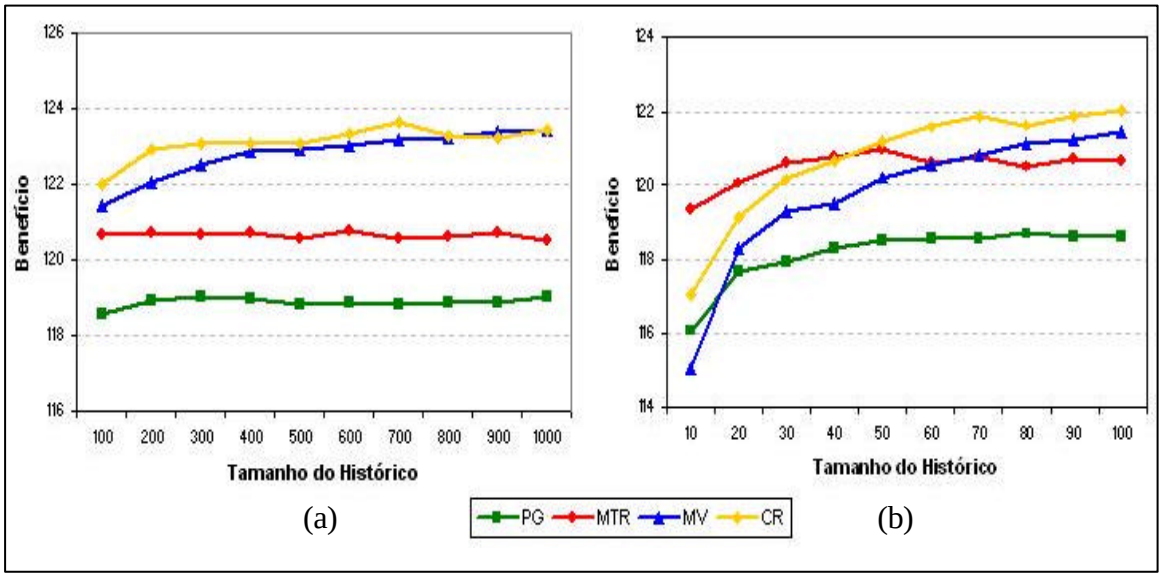


Figura 7.11: Sensibilidade ao tamanho do histórico.

c)Sensibilidade à Concentração de Benefício

Como já descrito no capítulo 6, os benefícios adquiridos pela execução de cada recurso pertencente a missão são fixos. Nesta fase de avaliação, a cada simulação efetuada, um recurso que compunha a missão tinha seu valor alterado para 150. Lembrando que, originalmente, os benefícios variavam de 1 à 15. O objetivo deste estágio de avaliação é verificar se a concentração de benefício em um determinado recurso pode refletir no benefício global da missão e no comportamento relativo das várias heurísticas.

A Tabela 7.16 apresenta os valores dos benefícios adquiridos pelas heurísticas adaptativas para os 14 *deadline*s testados. Os números em negrito, na tabela abaixo, indicam as heurísticas que formam o envelope superior para cada um dos *deadline*s. Está sendo considerada a variação de 1% do valor máximo alcançado.

A concentração de benefício em um único recurso da missão resultou, na maioria dos testes, no seguinte comportamento:

- Se comparados os índices de desempenho das heurísticas adaptativas desta fase de avaliação (análise de sensibilidade à concentração de benefício) com as demais fases de avaliação, percebe-se que a existência de um recurso com valor de benefício muito superior aos demais recursos da missão resulta em uma diferença maior entre os índices de benefício obtidos pelas heurísticas. Anteriormente, as 4 heurísticas adaptativas apresentaram resultados muito próximos. Esta diferença apresentada nos índices de desempenho era esperada, pois dependendo do comportamento que o agente tiver adotado para a missão, a execução ou não de um recurso com concentração de benefício refletirá de forma muito mais visível;

- A concentração de benefício evidencia os tipos de comportamento de cada heurística adaptativa, por exemplo com *deadline*s justos, onde a heurística PG perdeu por diferenças muito altas para as heurísticas CR e MV. Isto deve-se ao fato de que PG não possui uma opção de comportamento apropriada para este tipo de *deadline*, e a concentração de benefício evidenciou esta lacuna no desempenho da heurística PG. Por exemplo, com concentração de benefício nos recursos ***r5***, ***r9***, ***r11***, ***r12***, ***r14***. Isto acontece principalmente na faixa de *deadline* de ± 400 à ± 500, nas demais faixas de *deadline* as diferenças nos valores de benefício são mais próximas;

- Ainda considerando *deadline*s justos, uma particularidade importante é que as heurísticas que utilizam um limiar (“número mágico”) para efetivar a troca de comportamento, obtiveram índices de desempenhos inferiores (em alguns casos apresentaram uma diferença bastante considerável – como descrito anteriormente). Isto evidencia a vantagem do uso das heurísticas adaptativas que não utilizam constantes arbitrárias para tomada de decisão, estando elas melhor preparadas para mudanças que possam acontecer no sistema ou na missão;

•Com *deadline*s apertados, a heurística PG seguida pela heurística MTR obtiveram os maiores índices. Com *deadline*s justos o melhor desempenho foi apresentado pela heurística CR seguido pela heurística MV. Com *deadline*s folgados, novamente as heurísticas PG e MTR alcançaram os melhores índices de desempenho;

•A heurística MV apresentou resultados próximos à heurística CR, mas se desconsiderada a variação de 1% no benefício alcançado, a heurística CR obteve os maiores índices na maior parte das vezes;

•A heurística CR foi a que melhor se adaptou a esta mudança na carga (concentração de benefício) porque mesmo não fazendo parte do envelope superior (perdendo com *deadline*s apertados e folgados) foi por uma diferença pequena que não prejudicou sua eficácia e quando formou o envelope superior (com *deadline*s justos) foi com uma vantagem muito superior.

Tabela 7.16: Benefício Global com relação à concentração de benefício

Heurística/ *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
B1 = 150														
PG	**0,2**	9,4	**33,1**	**87**	**148,7**	**194,1**	221	230,7	242,7	265,8	273,7	278,5	280,5	**281,6**
MTR	**0,2**	**9,6**	**33**	86,6	147,1	**193,9**	229,5	251,8	257,4	265,3	272,8	279,1	280,3	**281,5**
MV	0,1	6,9	27,9	78,41	139,6	**192,4**	**234,1**	**259,5**	**270,4**	**276,1**	**279,4**	**281,5**	**282,2**	**282,9**
CR	0,1	5,8	27,4	80,8	142,1	**192,8**	**236,2**	**261,3**	**271,9**	**277,9**	**281,4**	**283,1**	**283,4**	**284,2**
B2 = 150														
PG	**0,3**	**11,6**	**39,3**	**113,1**	**197**	254,4	296,6	309,5	308,6	337,2	349,4	**356,9**	**358,6**	**360,7**
MTR	0,1	10,9	**39**	**113,2**	**197**	**258,8**	298,6	326,7	331,7	340,5	349,9	356,2	**358,9**	**361,1**
MV	0,2	9,2	33,7	105,1	187,7	251,7	**300,8**	**332,9**	**344,5**	**352,5**	**357,2**	**359,6**	**359,7**	**361,2**
CR	0,2	7,7	32,6	105,4	188,3	250,9	**303,1**	**333,1**	**346,1**	**353,1**	**358,2**	**360,1**	**360,5**	**361,6**
B3 = 150														
PG	**0,1**	**5,3**	**20,8**	76,3	145,5	208,1	247,9	264,6	300,4	**337,7**	**352**	**358,6**	**362,7**	**363,4**
MTR	**0,1**	5,2	**20,6**	**78**	**148,3**	208,3	261,7	300,7	314,6	335,9	**350,8**	**358,7**	**362,7**	**363,5**
MV	**0,1**	3,7	17,9	69,3	141,3	216,1	**275,2**	**311**	**324,3**	**339,7**	347,2	350,9	353,1	354,2
CR	0	3,3	17,8	70,3	143	**217,8**	**277,8**	**312,2**	**323,8**	**339,1**	**348,6**	353,4	355,4	355,8
B4 = 150														
PG	**0,7**	**17,1**	**46,8**	120,1	197,8	**266,5**	**308,2**	321,1	325,6	**356**	**370,6**	**378**	**383,5**	**383,5**
MTR	**0,7**	16,2	**46,9**	**123,5**	**200,9**	**265,6**	**307,5**	**333,5**	**345,4**	**357,4**	**372,7**	**378**	**383,5**	**383**
MV	0,6	13,8	40,4	111,6	189	256,1	**308**	**330,4**	**345,4**	**354,6**	363,1	368	371,8	371
CR	0,3	13,2	40	115,4	191,5	259,4	**307,7**	**332,7**	**345,1**	**356,9**	366,0	369,6	374,3	373,9

B5 = 150														
PG	**0,1**	3,9	17,4	66,2	130,6	195	241,3	267,5	426,1	**486,7**	**506,4**	**512,6**	**517,3**	**517,4**
MTR	**0,1**	**4,1**	17,6	64,1	133	198	295,8	387,3	430	469,1	**502,9**	**511,8**	**517,3**	**517,5**
MV	**0,1**	3,8	**18,6**	**67,9**	**160,2**	276,2	**380,1**	427,6	**457**	475,7	496,5	502,5	508	508,5
CR	**0,1**	2,6	17,5	67,1	157,9	**282,2**	**383,7**	**433,3**	**455**	477	497,5	504,7	510,1	510,1
B6 = 150														
PG	0,4	**11**	**36,7**	111,5	**208,1**	**290,6**	343,6	359,7	398	**430,1**	**447,5**	**453,8**	**460,8**	**458,6**
MTR	0,4	**10,9**	34,3	**114,9**	**208,8**	288,8	352,6	**389,2**	408,3	425	**446,8**	**454,2**	**459,9**	**458,7**
MV	**0,5**	8,5	29,9	101,6	195,8	**291**	354,6	**390,2**	**412,6**	425,6	437,9	441,3	447,7	445,8
CR	0,1	7,5	31,7	104,4	197,5	**291,8**	**357,9**	**392,8**	**414,5**	**427,5**	441,1	444,2	450,4	447,9
B7 = 150														
PG	0,3	**7,8**	**25,3**	70,7	**123,6**	169,9	196,6	212,4	264,6	**296,6**	**310,5**	**315,4**	**313,6**	**315,5**
MTR	0,3	**7,7**	**25,5**	**72**	122,3	**168,5**	212,7	254,7	271,9	**292,1**	**308,5**	**315**	**313,5**	**315,6**
MV	**0,4**	6,4	21,5	65,6	118,1	**180,3**	232,7	265,8	**283,1**	**292,9**	302,7	306,9	304,9	306,3
CR	0	4,9	21,4	65,7	118,2	**181,8**	**236**	**269,5**	**284,5**	**294,7**	303,1	307,8	305,6	307,4
B8 = 150														
PG	**0,1**	**5,7**	**24,6**	**80,5**	**150**	203,2	236	252,5	297,6	**324,5**	**338,4**	**345,3**	**345,2**	**345,5**
MTR	**0,1**	4,9	24,2	81,2	**149,7**	205	248	287,5	303,4	320,1	**337**	**345,2**	**344,9**	**345,5**
MV	**0,1**	4,2	20,3	72,9	141,6	204,9	258,2	292,3	**313,1**	320,7	331,5	337,7	337,1	336,7
CR	**0,1**	3,7	20,9	77,8	144,2	**207,7**	**262,8**	**295,5**	**313,9**	**323**	333,2	338,6	338,2	337,9
B9 = 150														
PG	**0,2**	**6,5**	**24,3**	**82,6**	152,3	211,4	249,8	278,9	438,4	**491**	**510,9**	**521,5**	**522,9**	**523,2**
MTR	0,1	**6,4**	**24,3**	80,2	150	210,5	286	378,1	422,6	475,4	**507,1**	**519,3**	**521,8**	**523,9**
MV	0,1	5,3	21,6	75,4	153,7	251,9	346,6	**416,4**	**453,2**	485	501,2	509	511,9	513,1
CR	0,1	4,7	21,3	77,8	**156,5**	**258,1**	**354**	**419,4**	**450,8**	**486,8**	505,4	512,6	514,8	515,7
B10 = 150														
PG	**0,3**	**9,1**	**30,2**	**89,6**	**159,6**	211,1	247,3	265,8	335,6	**380,8**	395,9	**403,2**	**403,5**	**406,7**
MTR	0,2	8,8	29	**88,3**	**159,9**	211,6	270,8	322,2	342,6	374,5	395,6	**403,1**	**403,7**	**406,8**
MV	0,2	6,5	26	83,3	158	**234,9**	300,3	**332,7**	**352,1**	372,4	**386**	394,7	392,7	396,9
CR	0	6,5	25	82,4	160,2	**236,1**	**304,1**	**335,7**	**349,7**	373,1	**388**	396,4	395,7	398,7
B11 = 150														
PG	0,2	10,5	31,2	87,7	151,4	205,7	239,7	286,3	429,6	**465,3**	**475,6**	**480**	**482**	**484,8**
MTR	**0,4**	**10,9**	31,7	87,1	151,1	209,1	310,2	389,5	416	458	**474,2**	**479,8**	**482,1**	**484,9**
MV	0,2	10,2	36,5	108,6	220,8	331,5	398,7	434	446,3	**462,2**	**471,5**	473,5	474,5	477,7
CR	0,1	8	**38,3**	**110,7**	**223,8**	**337,5**	**404,6**	**435,9**	**448**	**461,7**	470,4	473,7	475,2	478
B12 = 150														
PG	0,2	**7,9**	28,5	**97,4**	175	247,5	294,1	330,1	427,5	**461,6**	**475,4**	**482,4**	**481,3**	**482,8**
MTR	**0,3**	7,5	**30,2**	94,6	176	248,4	335,2	397,8	424,5	456,2	**473,9**	**481,4**	**481,7**	**482,8**

MV	0,2	6,1	25,7	90,5	**194**	299,2	381,9	422,6	**444,4**	**460,1**	468	473,6	472,9	473,4
CR	0,1	5,1	25,8	92,4	**194,5**	**305,8**	**390**	**428,3**	**443,6**	**459**	467,5	474,5	473,2	474,1
B13 = 150														
PG	**0,1**	**7**	**29,5**	**107,9**	**205,9**	**294,2**	349,1	370,8	391,6	**433,6**	**457**	**465,9**	**471,2**	**470**
MTR	**0,1**	6,7	**29,4**	103,4	**206,5**	**293,1**	357,3	399,2	413,4	**434,3**	**454,6**	**464,6**	**470,8**	**470,3**
MV	**0,1**	5,4	25,7	97,7	195,4	**292,2**	361,2	**401,3**	**417,6**	**432,6**	445,2	450,2	453,7	452,7
CR	**0,1**	5	25,3	100,2	201,4	**292,9**	**366**	**404,3**	**418,3**	**435,4**	447,3	453,5	458,1	455,4
B14 = 150														
PG	**0,1**	6,9	26,4	82,2	152,7	206	245,9	269,7	398,7	453,9	**471,9**	**480,8**	**482,5**	**485,5**
MTR	**0,1**	**7,6**	25,4	80,6	153	206,4	298,6	379,3	404,8	442,5	**470,5**	**480,1**	**481,6**	**485,5**
MV	**0,1**	7,1	**29,2**	**84,8**	**188,9**	294,7	384,1	**431,2**	**446,9**	**459,7**	**467,7**	472,4	473,6	476,9
CR	**0,1**	5,3	28,3	**87,5**	185,9	**302**	**388,6**	**434,1**	**448,2**	**461,3**	467,1	473,3	474,5	477,2
B15 = 150														
PG	0,1	**5**	**23,9**	84,1	160,8	227	274,7	304	441,7	**490,2**	**508,8**	**515,8**	**521,9**	**520,1**
MTR	0,1	**5**	23,2	83,1	160	230,2	321,5	409,7	442,7	478,7	**506,4**	**514,4**	**521,6**	**519,9**
MV	**0,2**	**4,8**	22,2	79,9	**191**	307,3	401,4	454,9	**481,3**	**492,7**	502,9	508,2	512,7	510,7
CR	0	3,6	23,4	**86,2**	**189,5**	**317,1**	**412,7**	**462**	**481,2**	**491,2**	**504,1**	507,3	513,3	511,1

d) Sensibilidade ao Número de Segmentos da Missão

Nesta etapa de simulações, a missão é composta por 4 segmentos sorteados a partir de um conjunto de 10 segmentos diferentes com reposição, ao contrário dos 6 segmentos utilizados até o momento em todas as simulações. O objetivo deste estágio de avaliação é verificar de que forma uma mudança no tamanho do grafo da missão pode refletir no comportamento das heurísticas.

Para cada missão foram lançados 100 agentes. Foi utilizado um único histórico para todas as configurações. O histórico é composto pelas últimas 100 execuções efetuadas.

Após as simulações verificou-se que o comportamento apresentado pelas heurísticas foi similar ao comportamento observado em missões compostas por 6 segmentos. Isto demonstra a robustez da adaptação frente a uma mudança não trivial no tamanho do grafo.

Conforme podemos observar na Tabela 7.17, as heurísticas simples apresentaram um comportamento similar ao apresentado com grafos maiores, onde cada heurística manteve seus níveis de desempenho conforme a faixa de *deadline* testada.

Tabela 7.17: Benefício Global – missão composta de 4 segmentos.

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	**7,3**	**30**	**58,8**	76,9	86	88,5	88,9	89	89,6	89,1	89,3	89,1	89,3	89,2
Ponderado_R	5,6	27,1	**58,9**	**84,6**	98,4	103	104,9	105,5	106,6	106,6	106,8	107,1	107,4	107,3
Ponderado	3,9	24	55,7	83,8	**101,2**	107,8	109,5	110,1	110,9	110,5	110,6	110,5	110,6	110,5
Guloso_R	2	14,3	44,3	75,3	96,5	107	110,9	112,9	114,4	114,6	115,6	115,9	116,5	116,7
Guloso	0,6	5,5	25	59,1	89,1	**109,3**	**118,2**	**121,9**	**123,3**	**123**	**123,5**	**123,3**	**123,4**	**123,4**
PG	7,3	29,4	**58,2**	77,1	86,05	105,8	**118,8**	**121,7**	**123,2**	**123,1**	**123,4**	**123,3**	**123,4**	**123,4**
MTR	**7,9**	**30,1**	57,7	77,9	96	106,8	117,2	**121,3**	**123,3**	**123,1**	**123,5**	**123,3**	**123,4**	**123,4**
MV	6,1	28,2	56,6	81	98,9	**108,1**	117,1	**121,1**	**123**	**122,8**	**123,2**	**123,1**	**123,2**	**123,1**
CR	6,3	28,7	57,3	**82**	**99,4**	**108,2**	117,2	**121,4**	**123,3**	**123**	**123,4**	**123,3**	**123,4**	**123,3**

A Tabela 7.18 apresenta os intervalos de confiança, com grau de confiança de 95%, para os benefícios médios alcançados pelas heurísticas.

Tabela 7.18: Intervalos de confiança – missão composta de 4 segmentos.

Heurística / *Deadline*	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Preguiçoso	± 0,4	± 0,8	± 0,8	± 0,7	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5	± 0,5
Ponderado_R	± 0,4	± 0,8	± 1	± 0,8	± 0,6	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4
Ponderado	± 0,3	± 0,8	± 1	± 0,9	± 0,7	± 0,5	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4
Guloso_R	± 0,2	± 0,7	± 1	± 1	± 0,7	± 0,5	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4
Guloso	± 0,1	± 0,5	± 0,9	± 1,2	± 1,1	± 0,8	± 0,6	± 0,4	± 0,4	± 0,4	± 0,4	± 0,3	± 0,3	± 0,4
PG	± 0,4	± 0,8	± 0,8	± 0,7	± 0,6	± 0,8	± 0,5	± 0,4	± 0,4	± 0,3	± 0,4	± 0,3	± 0,3	± 0,4
MTR	± 0,4	± 0,8	± 0,9	± 0,7	± 0,7	± 0,6	± 0,6	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,3	± 0,4
MV	± 0,4	± 0,8	± 0,9	± 0,9	± 0,7	± 0,7	± 0,6	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,3	± 0,4
CR	± 0,4	± 0,8	± 0,9	± 0,9	± 0,7	± 0,7	± 0,6	± 0,4	± 0,4	± 0,4	± 0,4	± 0,4	± 0,3	± 0,4

As abordagens adaptativas apresentaram resultados próximos e satisfatórios acompanhando o envelope superior do gráfico para todas as faixas de *deadline*. Os valores em negrito na Tabela 7.17 representam o envelope superior para as heurísticas simples e o envelope superior formado pelas abordagens adaptativas. Desta forma, pode-se facilmente perceber a atuação de cada heurística conforme a faixa de *deadline*. Por exemplo, com *deadline*s apertados (D < 300) o melhor desempenho foi apresentado pelo Algoritmo

Preguiçoso e pela abordagem MTR. A medida que o *deadline* vai aumentando, o Algoritmo Ponderado com Relógio supera as demais heurísticas simples, e a abordagem PG, as demais abordagens adaptativas. Com *deadline*s justos ($300 \leq D < 450$), o Algoritmo Ponderado apresentou os melhores resultados, assim como a abordagem CR, dentre as adaptativas, destacou-se. Com *deadline*s folgados ($D \geq 450$), inicialmente o melhor desempenho foi conquistado pela PG, e nos demais *deadline*s que pertecem a esta faixa, o Algoritmo Guloso apresentou os melhores benefícios. As abordagens adaptativas apresentaram, nesta faixa de *deadline*, resultados próximos e satisfatórios acompanhando os resultados alcançados pelo Algoritmo Guloso.

Os *deadline*s utilizados neste estágio de avaliação foram os mesmos simulados para grafos maiores (missões compostas por 6 segmentos – nas demais fases de avaliação). Na Tabela 7.16 e no gráfico da Figura 7.12 pode-se observar que a faixa de *deadline*s folgados é composta por um maior número de *deadline*s.

Dentre as abordagens que utilizam probabilidade (MV e CR) na sua tomada de decisão, a CR apresentou índices de desempenho mais altos em todos os *deadline*s verificados (mesmo que próximos).

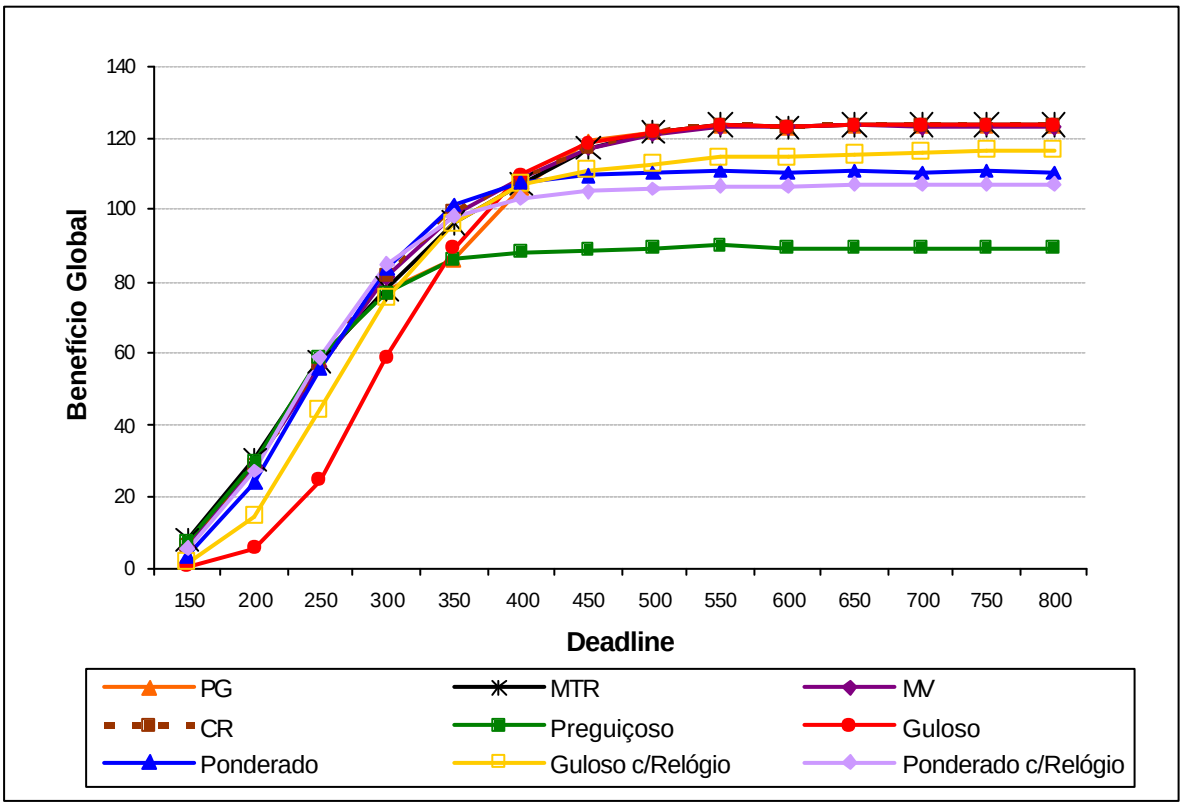


Figura 7.12: Benefício Global – missão composta de 4 segmentos.

7.3.4. Quinta Fase de Avaliação: Previsão de Falhas

As abordagens adaptativas contam com um serviço de diagnóstico que, baseado nas informações contidas no histórico e nas condições atuais da missão, é capaz de prever se a missão em andamento irá cumprir seu *deadline*. Para o uso de previsores, o histórico foi ampliado armazenando também o número de saltos (estágios) realizados pelo agente.

O uso de previsores não gera nenhuma alteração no benefício adquirido por cada heurística e a missão não é interrompida caso exista uma previsão de perda de *deadline*. O objetivo do uso de previsores é diagnosticar e avisar a aplicação da possibilidade de se realizar (ou não) a missão com êxito.

Foram desenvolvidos 4 diferentes tipos de previsores e cada um deles possui 3 medidas que indicam o percentual de acerto e de falha, são elas:

- Acerto: quando previu cumprir ou perder o *deadline* e acertou;
- Falso positivo: quando previu cumprir o *deadline* e no final da missão perdeu;
- Falso negativo: quando previu perder o *deadline* e no final da missão cumpriu.

Estas medidas podem ter uma importância diferente dependendo do tipo de aplicação. Por exemplo, o falso negativo pode lançar um tratamento de exceção para perda de *deadline* embora este não fosse necessário, pois o *deadline* não vai ser realmente perdido, algo que o falso positivo não gera. Dependendo do sistema, o falso positivo ou o falso negativo pode ser muito pior que o outro.

Os previsores não mudam o resultado da missão (benefícios alcançados), apenas prevêem possíveis falhas no cumprimento do *deadline*, o comportamento do agente não muda em função disto.

Os tipos de previsores são os seguintes:

1°) $\boldsymbol{P\,(\,R \pounds D \mid D = d)}$

O primeiro previsor calcula a probabilidade do agente cumprir o *deadline* da missão baseado apenas no valor do *deadline* do agente (apenas o *deadline* da missão é conhecido).

2°) ***P(R £ D | D = d, H = h)***

O segundo previsor calcula a probabilidade de se cumprir o *deadline* da missão conhecendo o *deadline* e a heurística que está sendo utilizada pelo agente.

3°) ***P (R £ D | D = d, EAMD = x)***

O terceiro previsor calcula a probabilidade de se cumprir o *deadline* da missão conhecendo o *deadline* do agente e o número de estágios visitados até o meio do *deadline*. Este previsor compara o número de estágios da missão atual (até o tempo D/2) com o número de estágios percorridos do histórico (no tempo D/2).

4°) ***P (R £ D | D = d, H=h, EAMD=x)***

O quarto previsor calcula a probabilidade de se cumprir o *deadline* da missão conhecendo o *deadline* e a heurística sendo utilizada pelo agente, e o número de estágios visitados até a metade do *deadline*.

A Tabela 7.19 apresenta os índices de desempenho dos 4 previsores acima descritos para cada heurística adaptativa.

Tabela 7.19: Taxa de acertos - Previsores de falha.

Heurística / Previsores		**P1**	**P2**	**P3**	**P4**
PG	**Acerto**	0,88	0,9	0,9	0,9
	Falso Positivo	0,09	0,04	0,04	0,04
	Falso Negativo	0,03	0,06	0,06	0,06
MTR	**Acerto**	0,88	0,9	0,9	0,9
	Falso Positivo	0,09	0,04	0,04	0,04
	Falso Negativo	0,03	0,06	0,06	0,06
MV	**Acerto**	0,88	0,9	0,9	0,89
	Falso Positivo	0,08	0,03	0,04	0,04
	Falso Negativo	0,03	0,07	0,06	0,07
CR	**Acerto**	0,88	0,9	0,9	0,89
	Falso Positivo	0,09	0,03	0,04	0,04
	Falso Negativo	0,03	0,07	0,06	0,07

Observando a Tabela 7.19, percebe-se o alto índice de acerto nas previsões para todas as heurísticas. A diferença do desempenho dos previsores é pequena, mas deve ser

considerada. O primeiro previsor apresentou índices de acertos mais baixos quando comparado aos demais. O segundo e o terceiro previsores apresentaram os melhores índices para todas as heurísticas, sendo que o quarto previsor não apresentou melhora significativa com relação a estes 2 previsores, mas apresentou melhores índices quando comparado ao primeiro previsor. O terceiro previsor de falhas não é tendencioso, pois os falsos negativos e falsos positivos são semelhantes, ele não tende a errar mais para um lado nem para o outro.

Baseado nos resultados das simulações concluiu-se que o uso do segundo ou do terceiro previsor é suficiente para se garantir uma previsão satisfatória, porém, o terceiro previsor apresentou um percentual de erro (falso positivo e falso negativo) mais balanceado que o segundo (na Tabela 7.19, ver a entrada das heurísticas MV e CR linhas 2 e 3, segunda e terceira colunas). Como estes índices podem ter uma importância diferente dependendo do tipo de aplicação, o terceiro previsor foi o mais eficaz.

Entretanto, os primeiro e segundo previsores fornecem a previsão no início da missão, enquanto que os terceiro e quarto previsores somente fornecem sua previsão após ter sido percorrido ***D/2*** da missão. Além disso, o primeiro e o segundo fornecem a previsão no nodo base do agente, onde o histórico está, ao passo que o terceiro e o quarto necessitam que o agente carregue o histórico e a previsão é disponível no nodo onde o agente está em ***D/2*** da missão.

7.4. Comparações entre as Heurísticas

Em vista dos resultados apresentados e discutidos nas fases de avaliação, foi possível perceber várias características das heurísticas adaptativas. Como os resultados (benefícios) obtidos foram próximos, realizou-se um grande número de simulações.

As heurísticas adaptativas definem seu comportamento conforme a necessidade da missão. A robustez destas abordagens foi comprovada pela capacidade de manter um índice de aproveitamento adequado para diferentes situações de testes.

Nesta seção de comparação do desempenho apresentado pelas heurísticas adaptativas foram consideradas as simulações que utilizaram um histórico único. Embora o uso de um histórico separado por missão tenha apresentado índices de benefícios mais

altos para estas abordagens, este tipo de histórico forma um cenário menos realista para o modelo computacional em questão o qual pressupõe o desconhecimento da configuração por parte do agente.

As abordagens MTR e PG, mesmo apresentando bons índices de benefício, utilizam como fator de tomada de decisão (no momento de troca de heurística) uma constante arbitrária (número mágico). A calibragem deste número é de vital importância para o sucesso desta heurística. O uso da constante 0,9 funcionou bem para as simulações testadas, mas não necessariamente funcionará bem para qualquer sistema, é um ponto fraco destas heurísticas. Uma escolha mal feita resultaria em decisões errôneas, conseqüentemente baixando a qualidade da execução da missão.

As heurísticas MV e CR não utilizam número mágico, elas empregam probabilidade para a tomada de decisão da heurística a ser utilizada para determinada missão. Esta característica aumenta a vantagem do uso destas abordagens.

Inicialmente, as heurísticas MV e CR foram desenvolvidas para utilizar 5 heurísticas simples como opção de comportamento para o agente. Com o intuito de poupar processamento (diminuindo o número de variáveis e equações) foram realizadas simulações para comparar o desempenho destas abordagens sem o uso das heurísticas que utilizam relógio. Os resultados obtidos mantiveram o mesmo padrão, por isso decidiu-se utilizar as versões com 3 heurísticas facilitando assim a tomada de decisão do agente.

A heurística PG não apresentou bons resultados com *deadline*s justos, onde obteve os índices mais baixos entre as abordagens adaptativas. A heurística MTR supre esta lacuna.

As heurísticas MV e CR apresentaram resultados satisfatórios. Quando comparados – entre si – seus índices de desempenho, a abordagem CR sobressaiu-se alcançando os maiores benefícios para maior parte dos *deadline*s testados.

Em relação às heurísticas simples do capítulo anterior, pode-se perceber a nítida vantagem em se utilizar uma heurística adaptativa, mostrando sua flexibilidade e capacidade de adaptação a diferentes situações de sistema e diferentes exigências de cada missão. Já as heurísticas simples apresentaram um comportamento distinto para cada faixa de *deadline*, satisfazendo os requisitos esperados, geralmente, para apenas uma faixa de *deadline*.

Em relação ao uso de clones, o uso de uma única heurística adaptativa supriu as expectativas de se obter bons resultados para diferentes situações. Foram alcançados resultados semelhantes aos encontrados com o uso de dois ou três agentes, economizando assim tempo de processamento nos nodos do caminho, embora gaste mais processamento no nodo inicial para a tomada de decisão da heurística adaptativa e com a formação do histórico.

7.5. Conclusões

Neste capítulo foram propostas, implementadas, avaliadas e discutidas abordagens mais elaboradas para a definição do itinerário do agente móvel. Estas abordagens consideram um histórico de execuções passadas e são capazes de eleger o comportamento adequado para o agente a cada nova missão. As informações conhecidas a cada nova missão são: o *deadline* da missão atual e o histórico das missões passadas.

No método adaptativo apresentado em (SCHLEGEL, 2006), não é tratada a definição do itinerário, a tomada de decisão tem o propósito de determinar se um agente deve se comunicar remotamente ou migrar até o nodo onde se encontra a aplicação destino. Assim como nesta tese, em (SCHLEGEL, 2006) a tomada de decisão do agente móvel - que acontece dinamicamente - baseia-se nas condições atuais do sistema e na experiência passada do próprio agente.

Assim como em (QU, 2005), nesta tese os agentes móveis são míopes e a probabilidade é utilizada para auxiliar o agente móvel em suas tomadas de decisão. O que os diferencia, em relação ao uso de probabilidade, é que em (QU, 2005) utiliza-se probabilidade para decidir qual o próximo nodo a visitar, enquanto que na presente tese utiliza-se probabilidade para decidir qual o comportamento que será adotado pelo agente móvel na próxima missão (heurísticas adaptativas, Capítulo 7). Outra divergência é com relação a determinação do itinerário, em (QU, 2005) primeiramente acontece a seleção do melhor itinerário (já descrito no Capítulo 4) e então um agente parte para cumprir sua missão e este já conhece seu itinerário. Nesta tese, a definição do itinerário acontece de forma dinâmica.

As abordagens adaptativas utilizam as heurísticas simples, avaliadas no capítulo anterior. Em função destas avaliações, foi possível descrever um comportamento distinto

para cada uma das heurísticas simples frente a diferentes faixas de *deadline*s. A combinação de heurísticas simples com diferentes comportamentos permitiu a construção de heurísticas combinadas mais eficazes, capazes de lidar com ampla variação no valor do *deadline* do agente.

Foram estudadas as sensibilidades das abordagens adaptativas quanto: ao tamanho do histórico, a concentração de benefício em um único recurso, ao número de heurísticas simples utilizadas na tomada de decisão e ao tipo/tamanho do grafo formado a cada missão.

Também foram propostos, descritos e avaliados 4 previsores de falha, os quais permitiriam à aplicação detectar a provável perda do *deadline* antecipadamente e tomar medidas de tratamento de exceção apropriados.

As simulações das abordagens com adaptação na partida mostraram desempenho bastante satisfatório, fazendo com que fossem alcançados os maiores índices de benefício dentro das condições de cada missão. Assim, os objetivos deste capítulo foram atingidos, uma vez que ficou comprovada a capacidade de adaptação das abordagens estudadas.

Em função das características apresentadas, a heurística adaptativa CR mostrou-se a mais interessante, pois apresentou a flexibilidade e a autonomia necessária para o cumprimento de diferentes tipos de missão sob diferentes situações do sistema e de *deadline*. A heurística CR alcançou bons índices de desempenho para todas as faixas de *deadline* utilizando probabilidade e seu histórico de execuções passadas para escolha do comportamento a utilizar a cada nova missão.

8. Conclusões

8.1. Revisão das Motivações e Objetivos

O desenvolvimento tecnológico vem tornando possível a materialização de conceitos e abstrações que já existem há algum tempo, mas que somente agora começam a sair do papel. Agentes móveis são um desses exemplos, graças à evolução da miniaturização de dispositivos, das redes de comunicação, fontes de energia e capacidade de processamento.

No entanto, um levantamento bibliográfico efetuado no decorrer desta tese mostrou que ainda há muitas lacunas nas pesquisas que tratam sobre a definição do itinerário de agentes móveis com requisitos temporais. Foram estudadas diversas abordagens relacionadas com as áreas de interesse deste trabalho. São elas: agentes móveis, definição de itinerário, computação imprecisa e tempo real. Este estudo serviu como base para a proposição de alguns aspectos do modelo computacional e das heurísticas apresentadas nesta tese.

O objetivo geral foi apresentar um modelo computacional para aplicações baseadas em agentes móveis imprecisos com restrições de tempo real. Em especial, foi considerada a questão da definição do itinerário e seu impacto no tempo de resposta da aplicação.

Visando atender a este objetivo geral, os seguintes objetivos específicos foram perseguidos:

- Desenvolvimento do modelo computacional para aplicações baseadas em agentes móveis imprecisos e míopes com restrição temporal;
- Elaboração de heurísticas capazes de guiar o agente móvel na definição de seu itinerário auxiliando-o no cumprimento de sua missão respeitando o *deadline*;
- Análise e avaliação do comportamento destas heurísticas em diferentes circunstâncias no sistema distribuído, por meio de simulações.

8.2. Visão Geral do Trabalho

Nesta seção é feita uma revisão do trabalho realizado e de como este trabalho atacou os problemas listados na seção anterior. O Capítulo 1 da tese descreveu o contexto onde este trabalho está inserido e seus objetivos.

Nos Capítulos 2, 3 e 4 foi apresentada uma revisão da literatura. O Capítulo 2 teve por objetivo esclarecer os conceitos que envolvem agentes móveis e evidenciar o interesse em se unir a tecnologia de agentes móveis a sistemas com requisitos de tempo real. Também foram discutidas algumas abordagens que tratavam da questão da definição do itinerário, onde foram descritas diferentes formas de atacar o problema, segundo aspectos e objetivos diversos.

O Capítulo 3 apresentou a conceituação de Sistemas de Tempo Real, foi discutida também a técnica de Computação Imprecisa. Combinados estes conceitos serviram de base para o modelo proposto.

Propostas que inter-relacionam mobilidade de código e restrição temporal em aplicações distribuídas foram discutidas no Capítulo 4, comprovando assim o expressivo interesse nestas áreas. Buscou-se uma bibliografia que desse suporte aos objetivos desta tese.

No Capítulo 5 foi apresentado o modelo computacional proposto por este trabalho. Este modelo descreve um sistema distribuído através do qual o agente móvel viaja em busca do cumprimento de sua missão. Uma missão é composta por um conjunto de recursos, e cada recurso representa um benefício adicional para a missão do agente, cujo objetivo é maximizar os benefícios obtidos ao longo do itinerário, respeitando as relações de precedência e as restrições temporais.

O itinerário é definido dinamicamente, uma vez que o agente móvel conhece apenas as opções imediatas do diagrama de recursos. Por esta característica que lhe é peculiar, ele é classificado como míope. Outro requisito que deve ser considerado pelo agente móvel é o *deadline* de cada missão, evidenciando assim a importância da escolha do melhor itinerário para cada situação apresentada.

Os Capítulos 6 e 7 abordaram a questão da definição do itinerário. Foram descritas, respectivamente, seis heurísticas simples e quatro heurísticas adaptativas utilizadas para determinar o comportamento do agente móvel.

Todas as heurísticas 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.

No Capítulo 6, baseando-se nos resultados apresentados pelas simulações efetuadas, foi possível relacionar cada tipo de *deadline* a uma determinada heurística simples. Todas as heurísticas simples apresentam um comportamento bem definido, cuja eficácia depende do *deadline* imposto ao agente móvel.

Também foi considerada a utilização de agentes clonados, onde buscou-se compensar a miopia do agente através da utilização de comportamentos distintos para a mesma missão. As avaliações mostraram que o desempenho das duplas (limitação do número de agentes clones imposta com o intuito de evitar *overhead* exagerado), em todas as situações, foi superior ao desempenho alcançado com a execução das heurísticas individualmente. Ao mesmo tempo, algumas duplas cobrem melhor o espectro de *deadline*s do que outras.

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.

Partindo destas premissas, no Capítulo 7 foram propostas, implementadas, avaliadas e discutidas abordagens mais elaboradas para a definição do itinerário do agente móvel. Estas abordagens consideram um histórico de execuções passadas e são capazes de eleger o comportamento adequado para o agente a cada nova missão.

Foram estudadas as sensibilidades das abordagens adaptativas quanto: ao tamanho do histórico, a concentração de benefício em um único recurso, ao número de heurísticas simples utilizadas na tomada de decisão e ao tipo/tamanho do grafo formado a cada

missão. Também foram descritos e avaliados previsores de falha, os quais permitiriam à aplicação detectar a provável perda do *deadline* antecipadamente e tomar medidas de tratamento de exceção apropriados.

Os resultados apresentados após as simulações das abordagens com adaptação na partida mostraram desempenho bastante satisfatório, fazendo com que fossem alcançados os maiores índices de benefício dentro das condições de cada missão.

8.3. Principais Contribuições desta Tese

Dentro dos objetivos traçados para este trabalho e das atividades desenvolvidas durante este período, pode-se enumerar os seguintes pontos de destaque desta tese:

Proposição do modelo computacional. Este modelo é inovador ao associar restrição de tempo e o conceito de recursos opcionais à tecnologia de agentes móveis;

Implementação de heurísticas apropriadas para o modelo computacional definido, de tal sorte a estabelecer um compromisso entre valor da missão e tempo de execução:

1) Proposição e validação, através de simulações, de abordagens – simples e em clones – para determinação dinâmica do itinerário de agentes móveis imprecisos com restrições temporais;

2) Proposição e validação, através de simulações, de abordagens mais elaboradas – adaptativas (baseadas em probabilidade condicional) – para determinação dinâmica do itinerário de agentes móveis imprecisos com restrições temporais.

De forma a divulgar os conceitos e resultados obtidos com o modelo computacional proposto e, principalmente, de submeter seus resultados para uma avaliação crítica da comunidade científica que se ocupa das questões discutidas nesta tese, alguns documentos na forma de artigos foram produzidos. As revisões e discussões provenientes que resultaram destas publicações contribuíram para elucidar algumas limitações e lacunas dos resultados preliminares e para motivar a superação destes problemas no modelo computacional final.

Conforme listados a seguir, os documentos científicos produzidos e publicados nesta tese foram: três artigos em eventos internacionais e um artigo em evento nacional.

1. RECH, Luciana de Oliveira; OLIVEIRA, Rômulo de; MONTEZ, Carlos. *Dynamic Determination of the Itinerary of mobile Agents with Timing Constrains*. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Compiègne, França, 2005, IEEE Computer Society, V. 1, p. 45-50.

2. RECH, Luciana de Oliveira; MONTEZ, Carlos; OLIVEIRA, Rômulo de. *Determinação Dinâmica do Itinerário de Agentes Móveis Imprecisos com Deadline Firme.* In: 8TH BRAZILIAN WORKSHOP ON REAL-TIME SYSTEMS, Anais do WTR 2006 - VIII Workshop de Tempo Real. Curitiba, Brasil, 2006, Gráfica Univ. Champaguat, V. I, p. 97-104.

3. RECH, Luciana de Oliveira; MONTEZ, Carlos; OLIVEIRA, Rômulo de. *A Clone-Pair for the Dynamic Determination of the Itinerary of Imprecise Mobile Agents with Firm Deadlines*. In: ETFA - 11th IEEE International Conference on Emerging Technologies and Factory Automation, Praga, República Tcheca, 2006, V. Único, p. 9-15.

4. RECH, Luciana de Oliveira; MONTEZ, Carlos; OLIVEIRA, Rômulo de. *A New Model for the Itinerary Definition of Real-Time Imprecise Mobile Agents.* In: The 2006 IEEE International Conference on Information Reuse and Integration - IEEE IRI 2006: Heuristic Systems Engineering, Waikoloa, Hawaii, EUA, 2006, V. Único. p. 123-126.

8.4. Trabalhos Futuros

Considerando o atual estágio deste projeto, algumas possibilidades para sua continuidade são apresentadas a seguir:

- Utilizar técnicas de IA para gerar os grafos de recursos e fazer uma análise de heurísticas propostas para definição de itinerário utilizando agentes inteligentes;

- Concepção de um modelo onde todos os recursos são do tipo “variável”. Isto exigiria novas heurísticas. No trabalho atual, a decisão do agente é discreta, caso todos os recursos fossem do tipo “variável”, a decisão seria de natureza contínua dentro do espectro de variação do tempo de uso de cada recurso;

- Integrar a essência do problema apresentada nesta tese – definição do itinerário de agentes móveis com restrição temporal – no contexto de redes sensores sem fio, salientando a questão da restrição de energia;

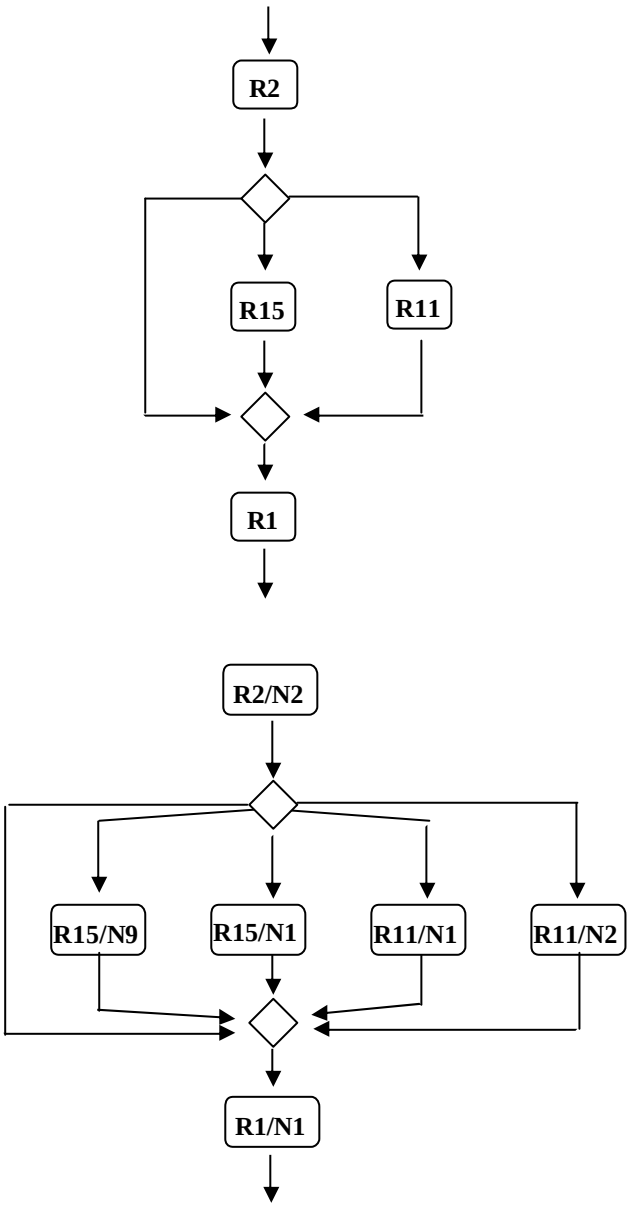
- Elaborar heurísticas adaptativas capazes de guiar o agente móvel na definição de seu itinerário dinamicamente, auxiliando-o no cumprimento de sua missão enquanto ele

respeita o *deadline*. Estas heurísticas utilizariam o histórico contendo os caminhos e resultados obtidos em execuções passadas;

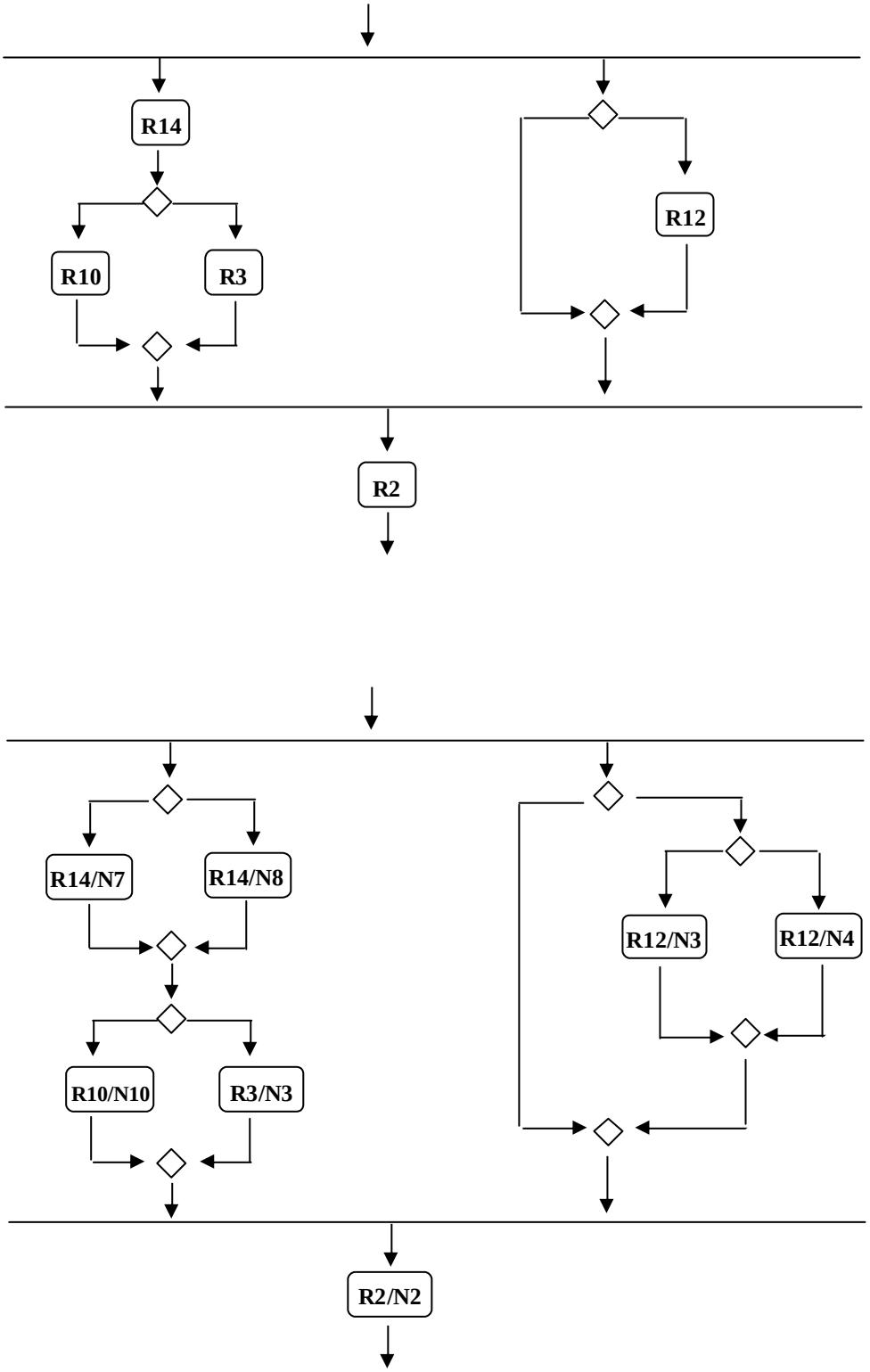
- Utilização da técnica de Aprendizado por Reforço onde um agente aprendiz, a partir da interação com o ambiente que o cerca, aprende de maneira autônoma uma política ótima de atuação. Utilizando as informações referentes aos caminhos anteriormente percorridos, o agente móvel terá menores possibilidades de erro na escolha do próximo itinerário. Pois a cada escolha "errada" é diminuída a chance de ser sorteado novamente este caminho;
- Uso de previsores de falhas (índices para basear a decisão adaptativa no futuro). O objetivo do uso de previsores na presente tese foi apenas diagnosticar e avisar a aplicação da possibilidade de se realizar (ou não) a missão com êxito. Propõe-se o estudo da possibilidade destes previsores de falhas refletirem na decisão de abortar a missão durante sua execução, caso seja detectada uma possível perda de *deadline*. Assim, evitando o processamento necessário até a finalização da missão;
- Versões com Relógio - onde após a execução de cada recurso é estimado o tempo disponível para executar os recursos restantes da missão - utilizando o histórico de execuções anteriores para sortear o comportamento do agente entre Preguiçoso e Guloso.

ANEXO A : Diagramas de Recursos e Diagramas de Nodos dos Segmentos de Missão

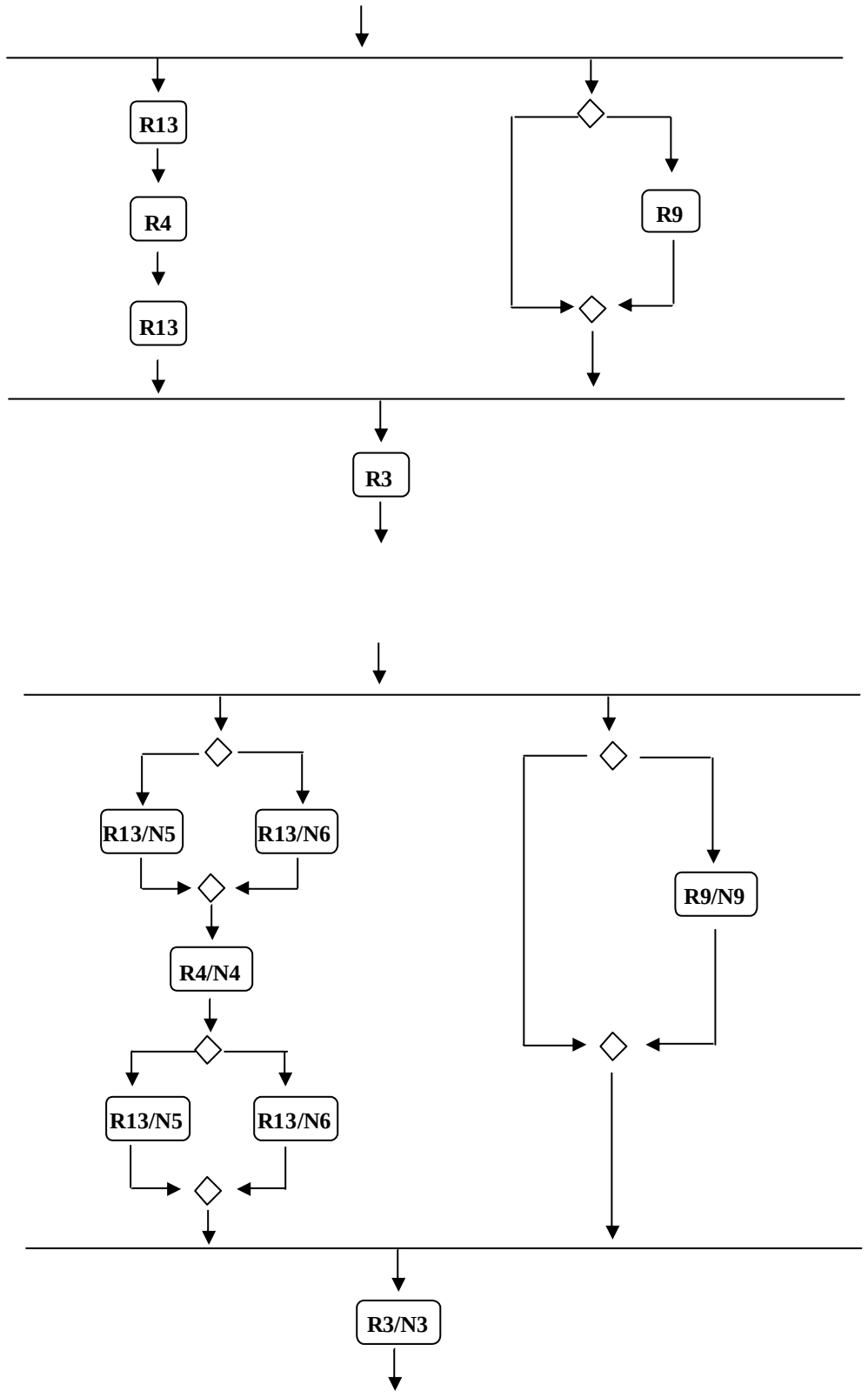
A.1 Segmento 1



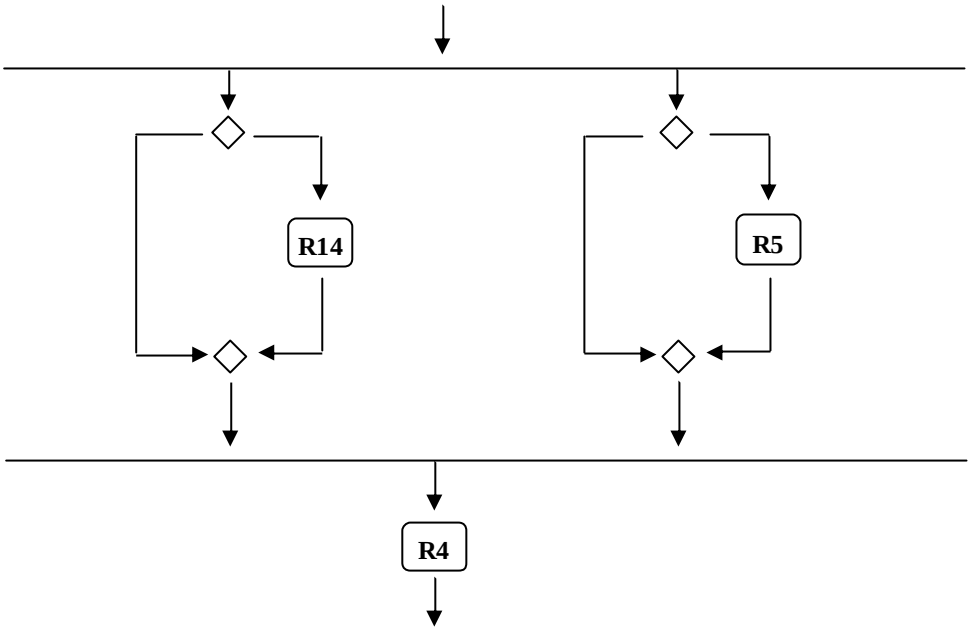
A.2 Segmento 2

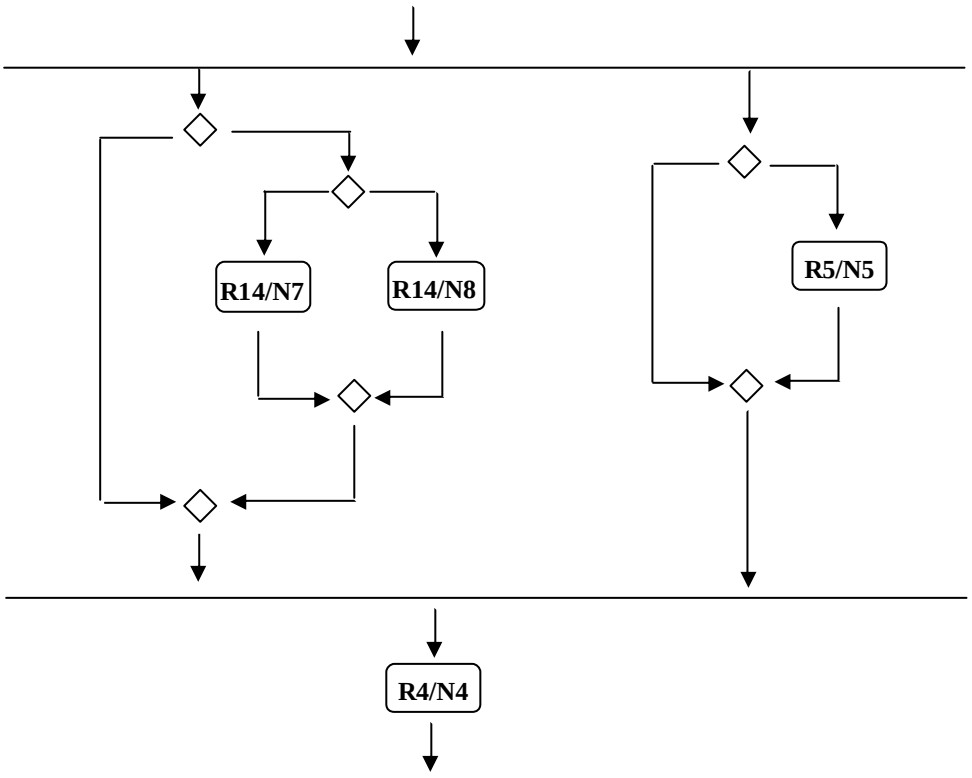


A.3 Segmento 3

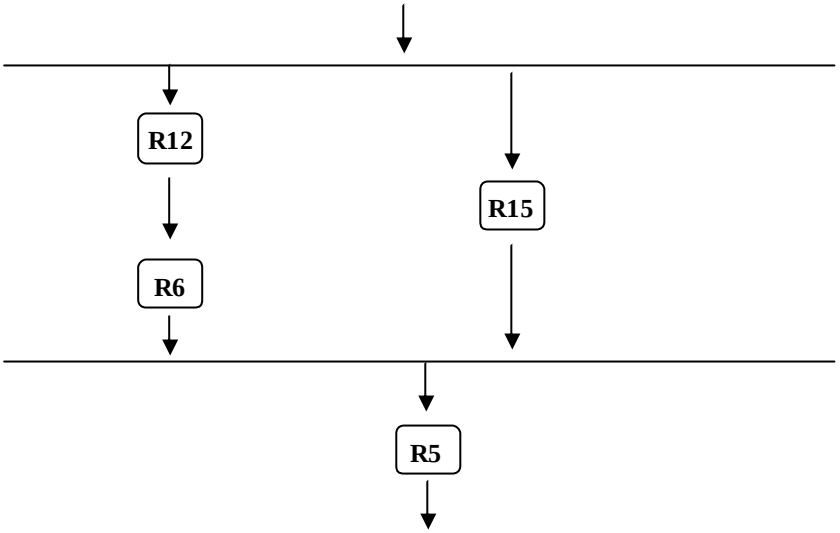


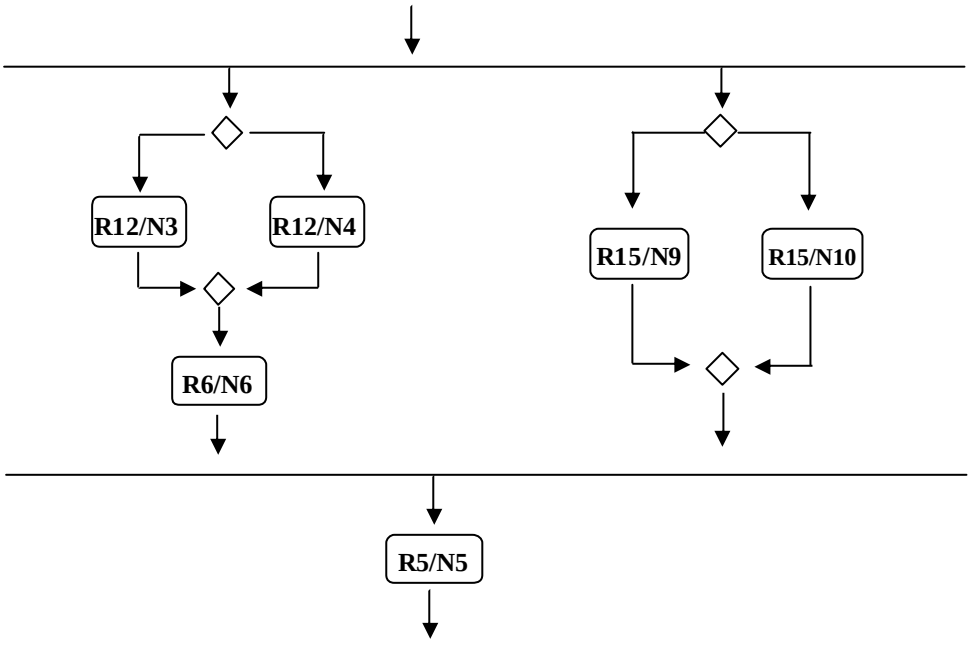
A.4 Segmento 4



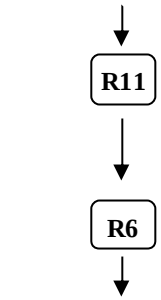


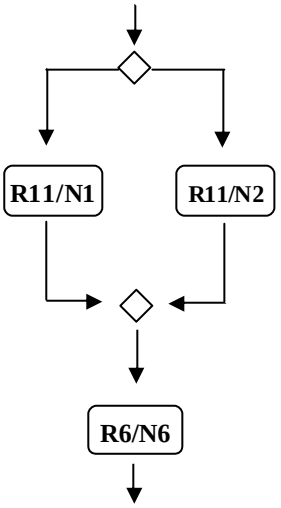
A.5 Segmento 5



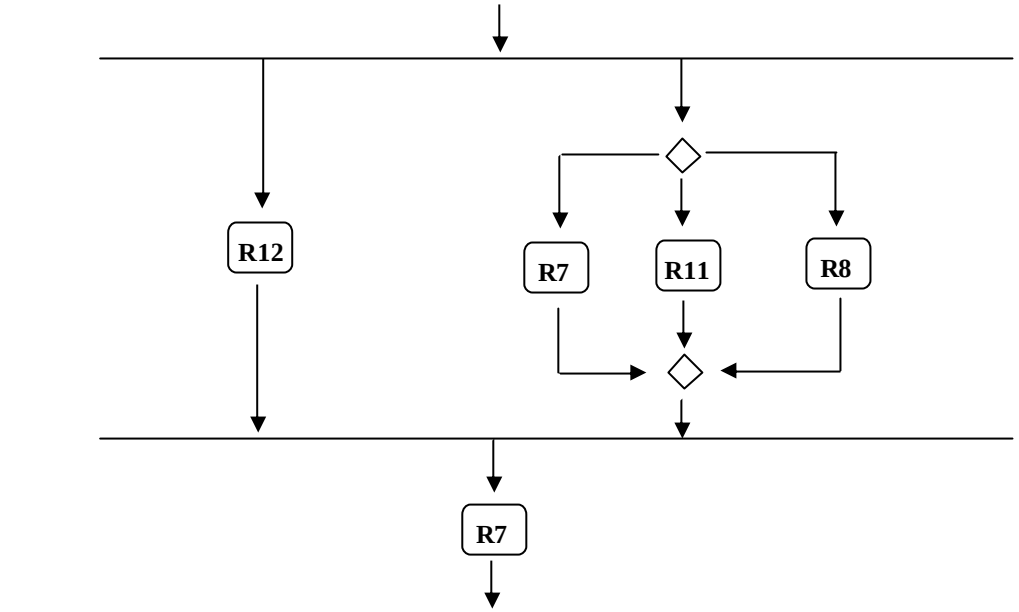


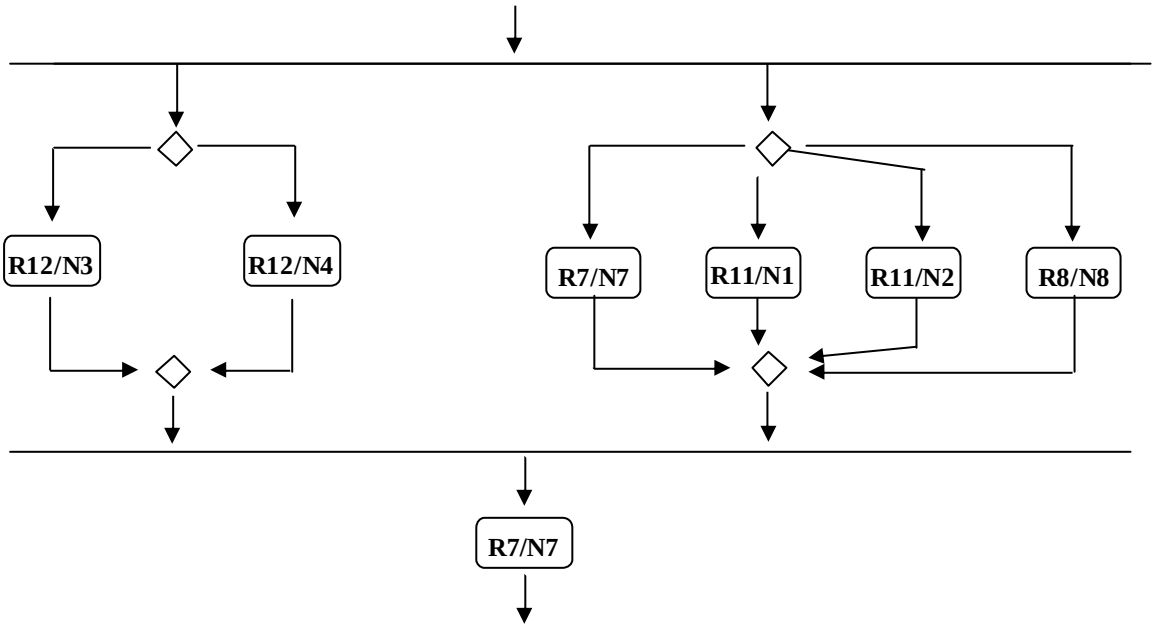
A.6 Segmento 6



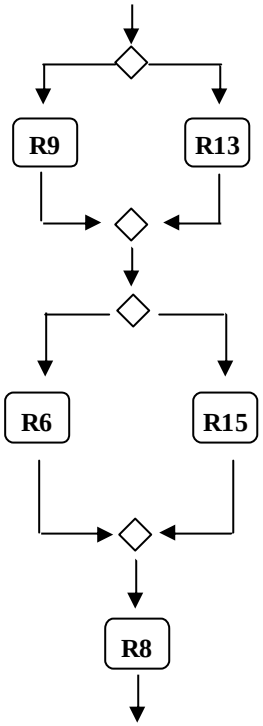


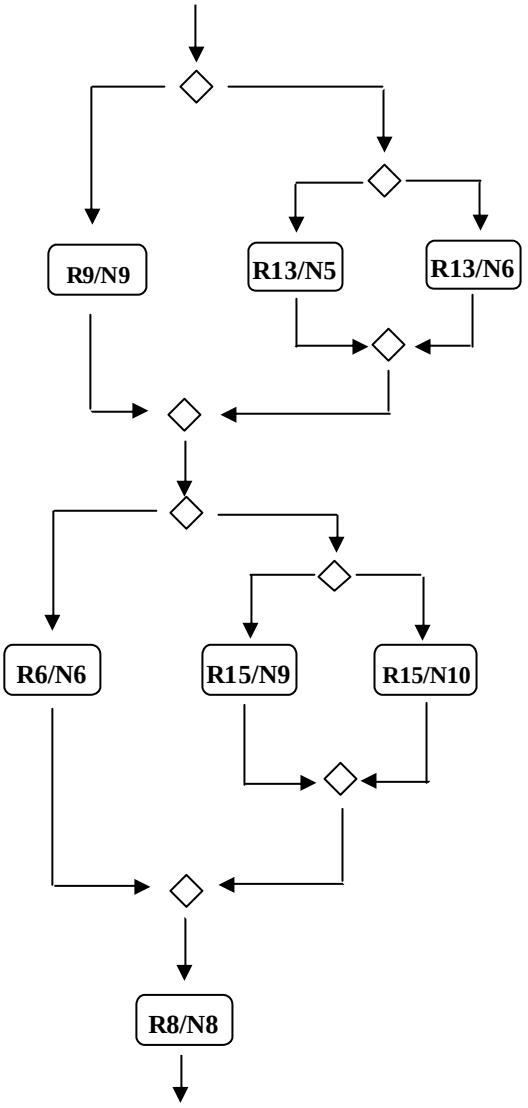
A.7 Segmento 7



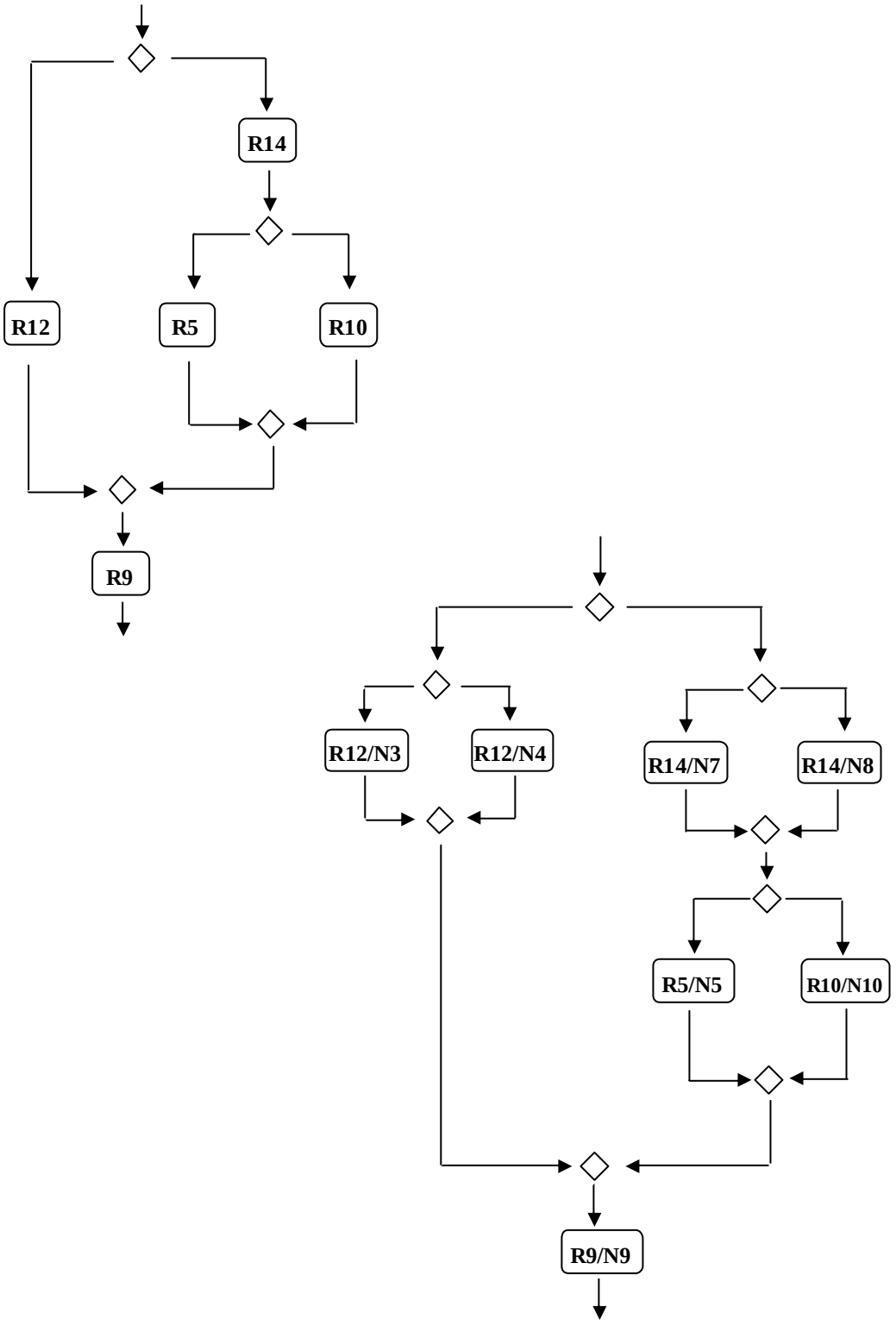


A.8 Segmento 8

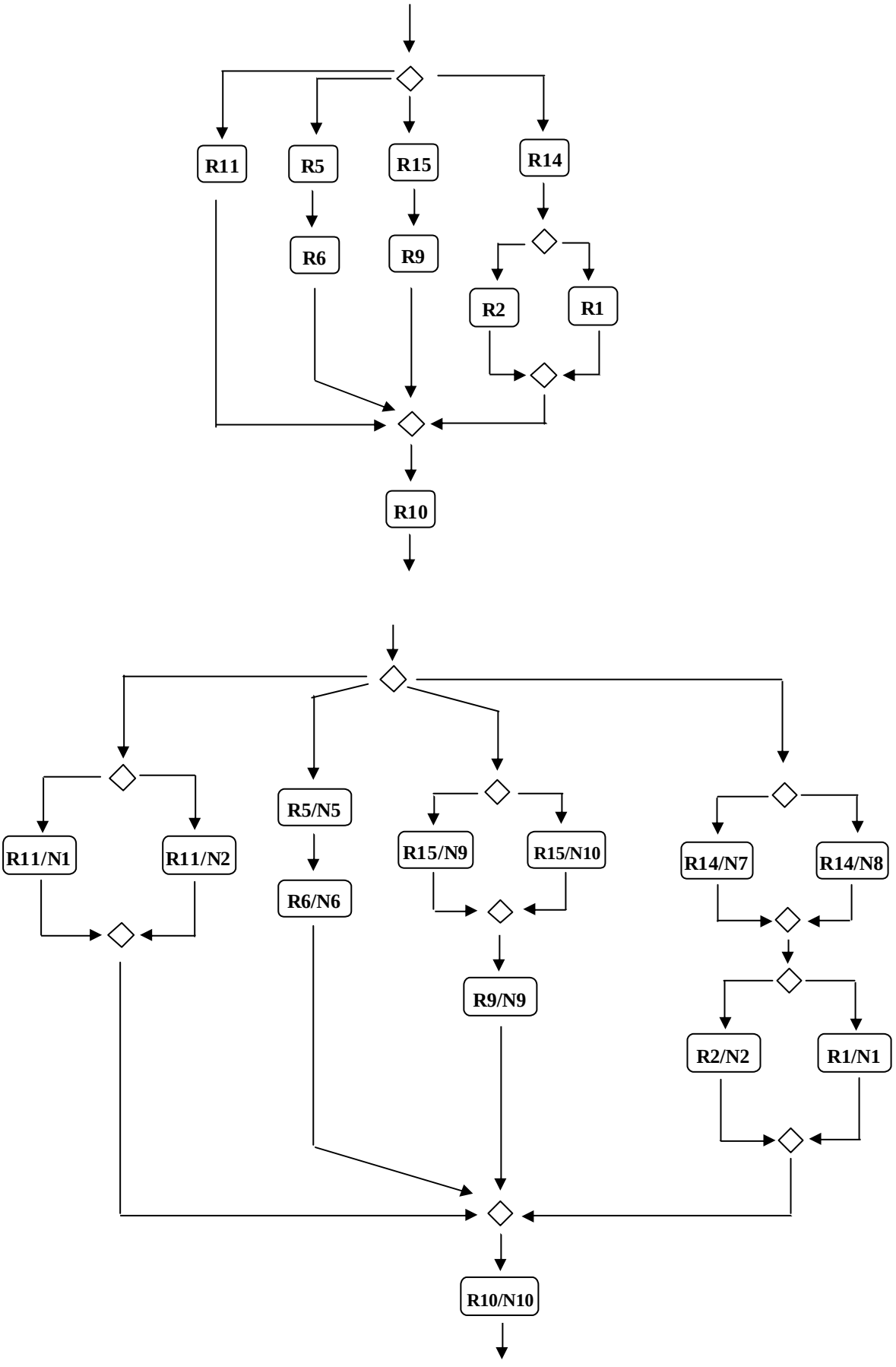




A.9 Segmento 9

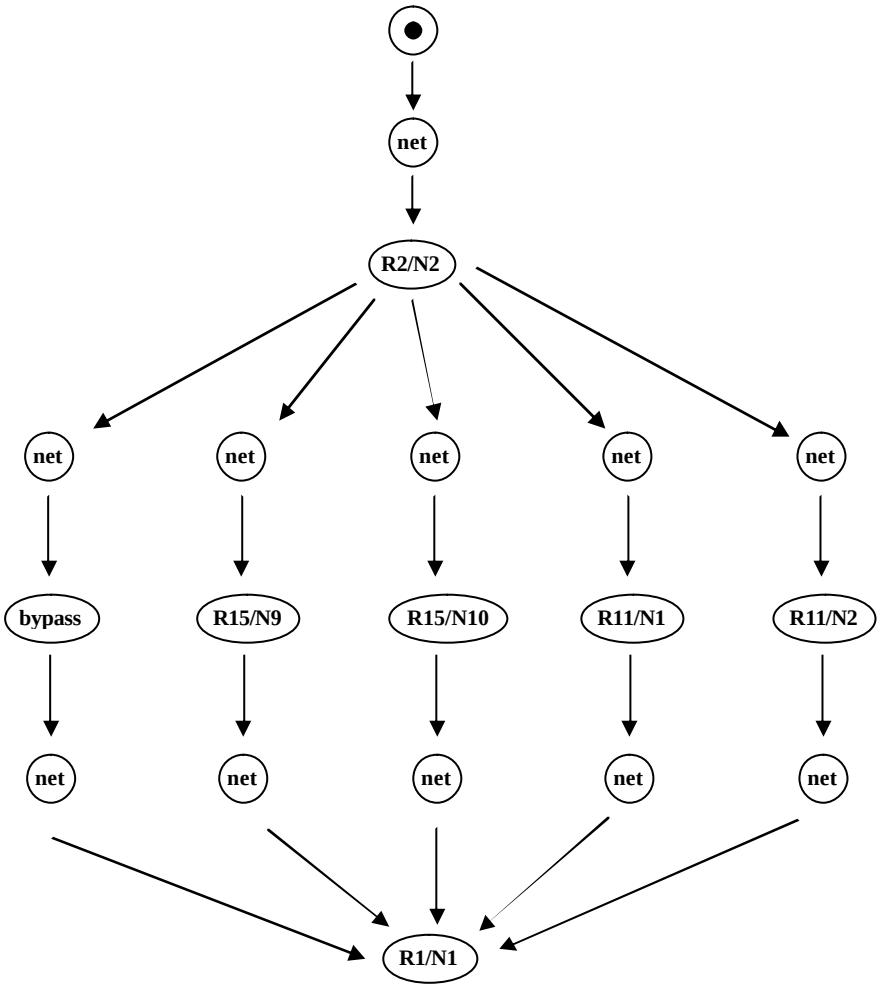


A.10 Segmento 10

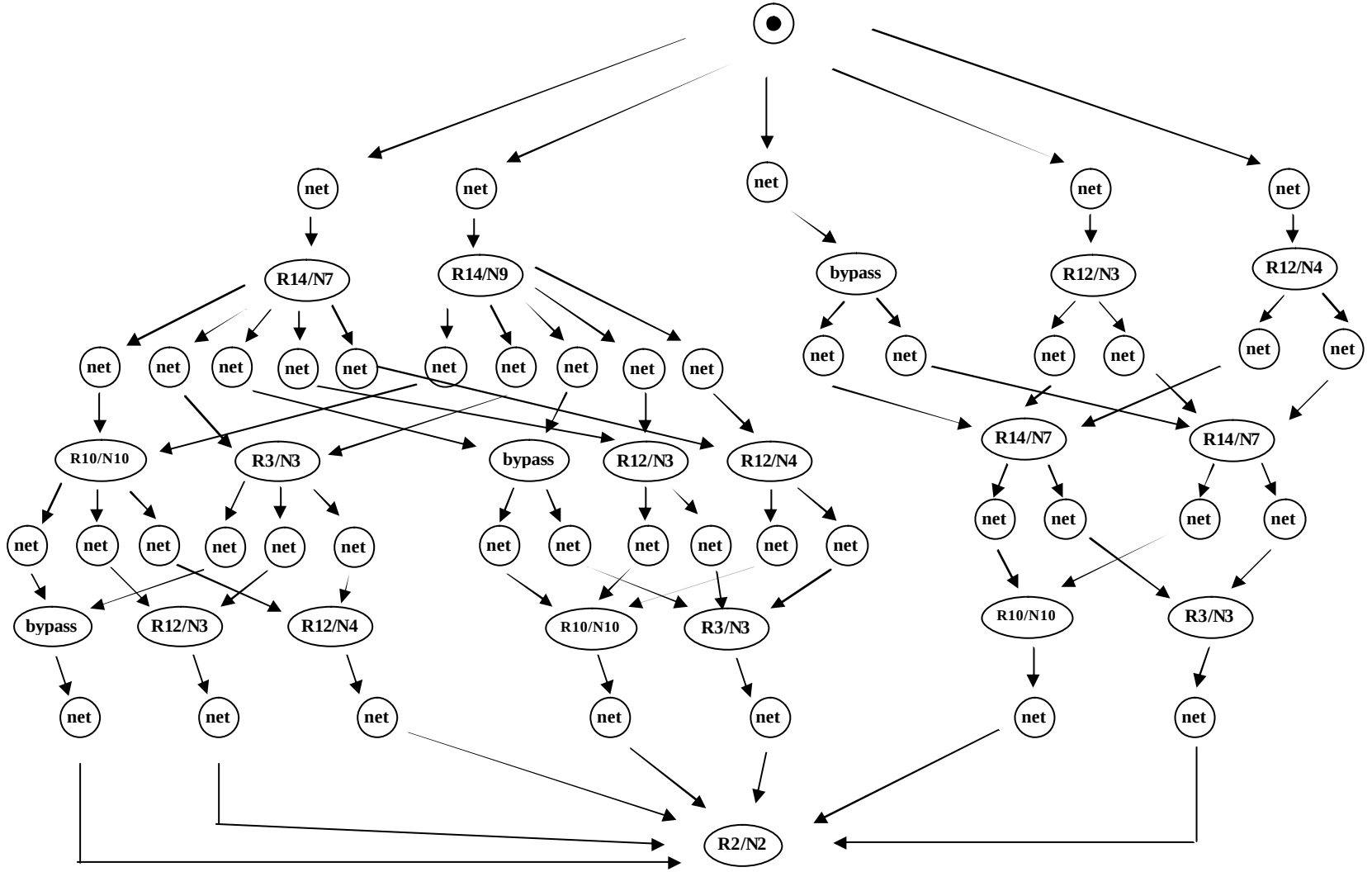


ANEXO B: Grafos dos Segmentos de Missão

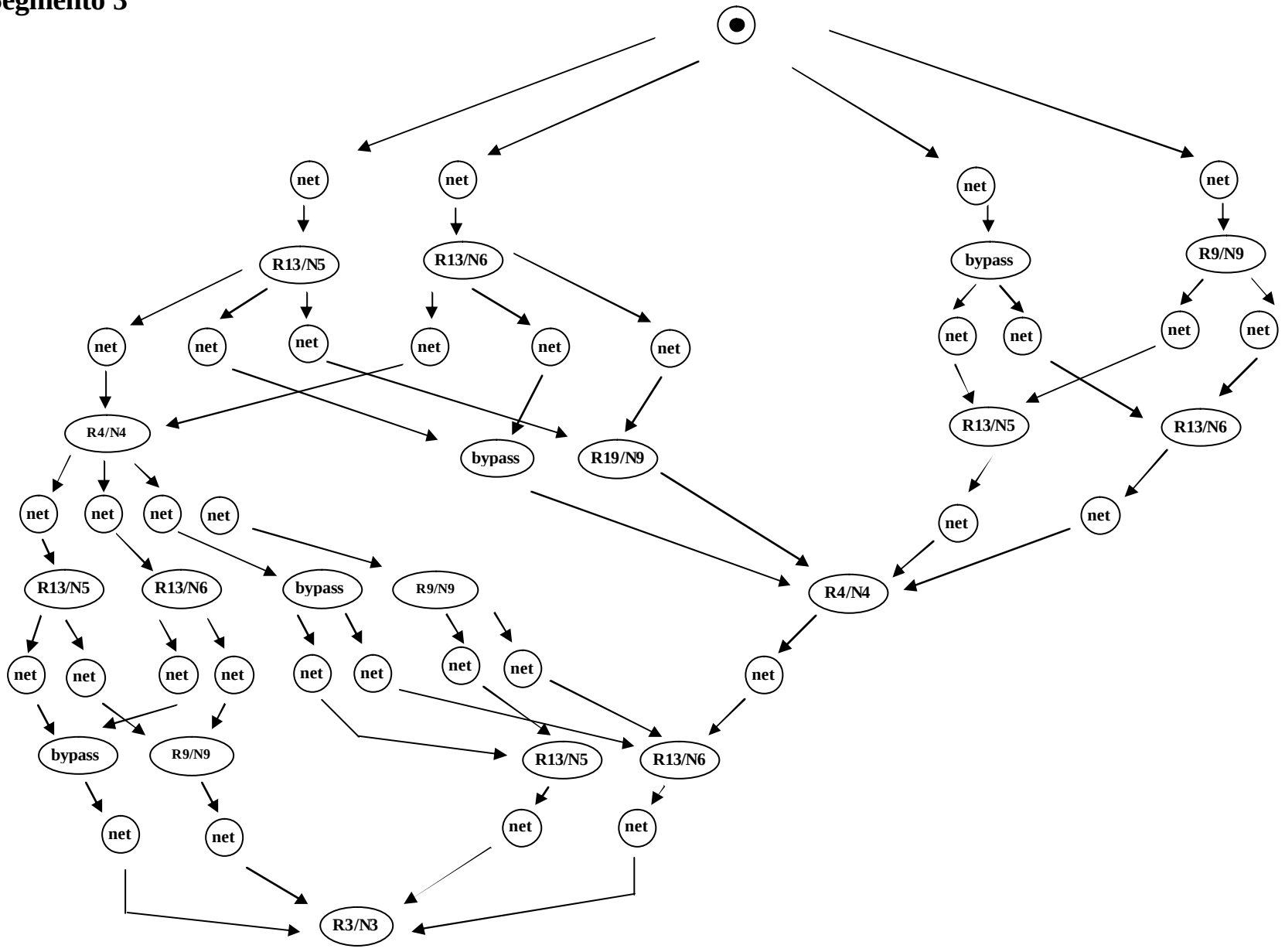
B.1 Segmento 1



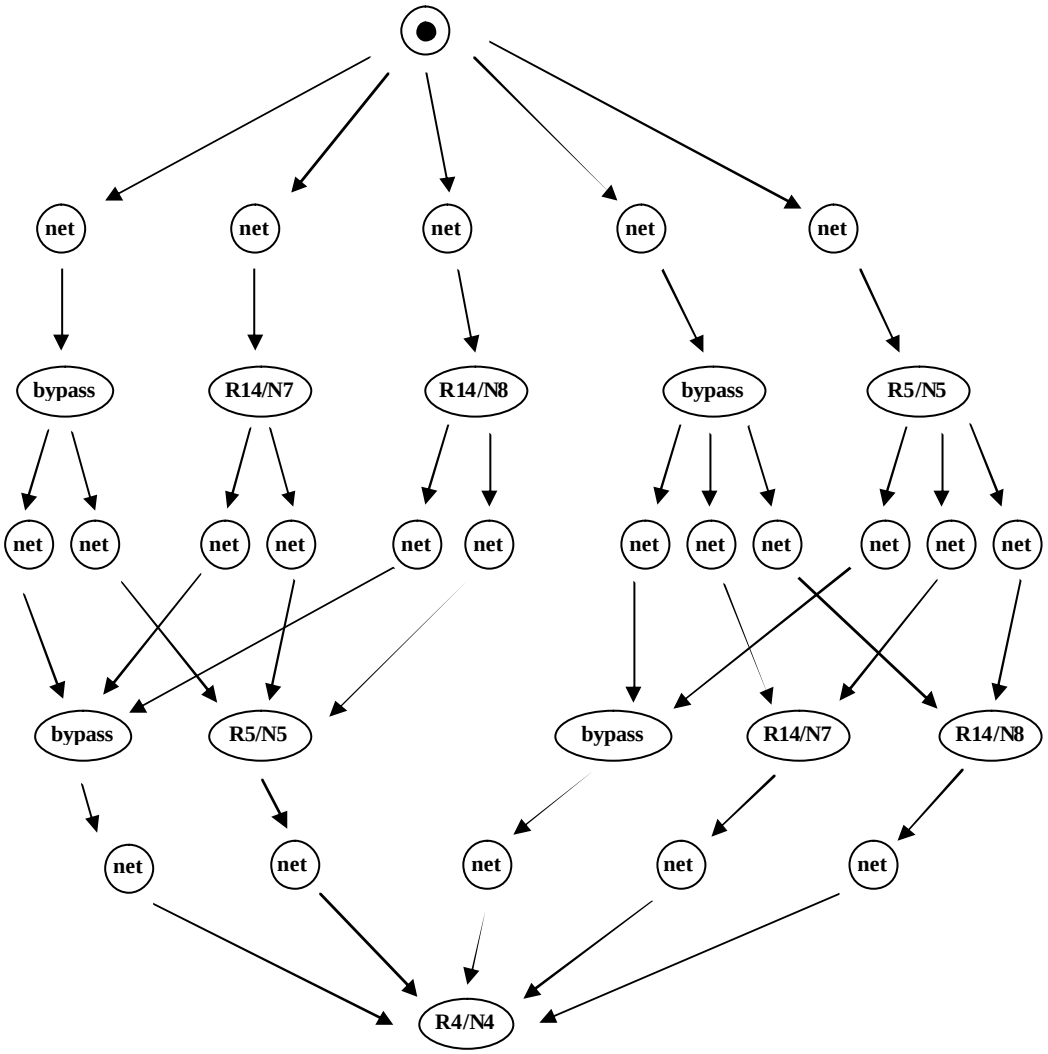
B.2 Segmento 2



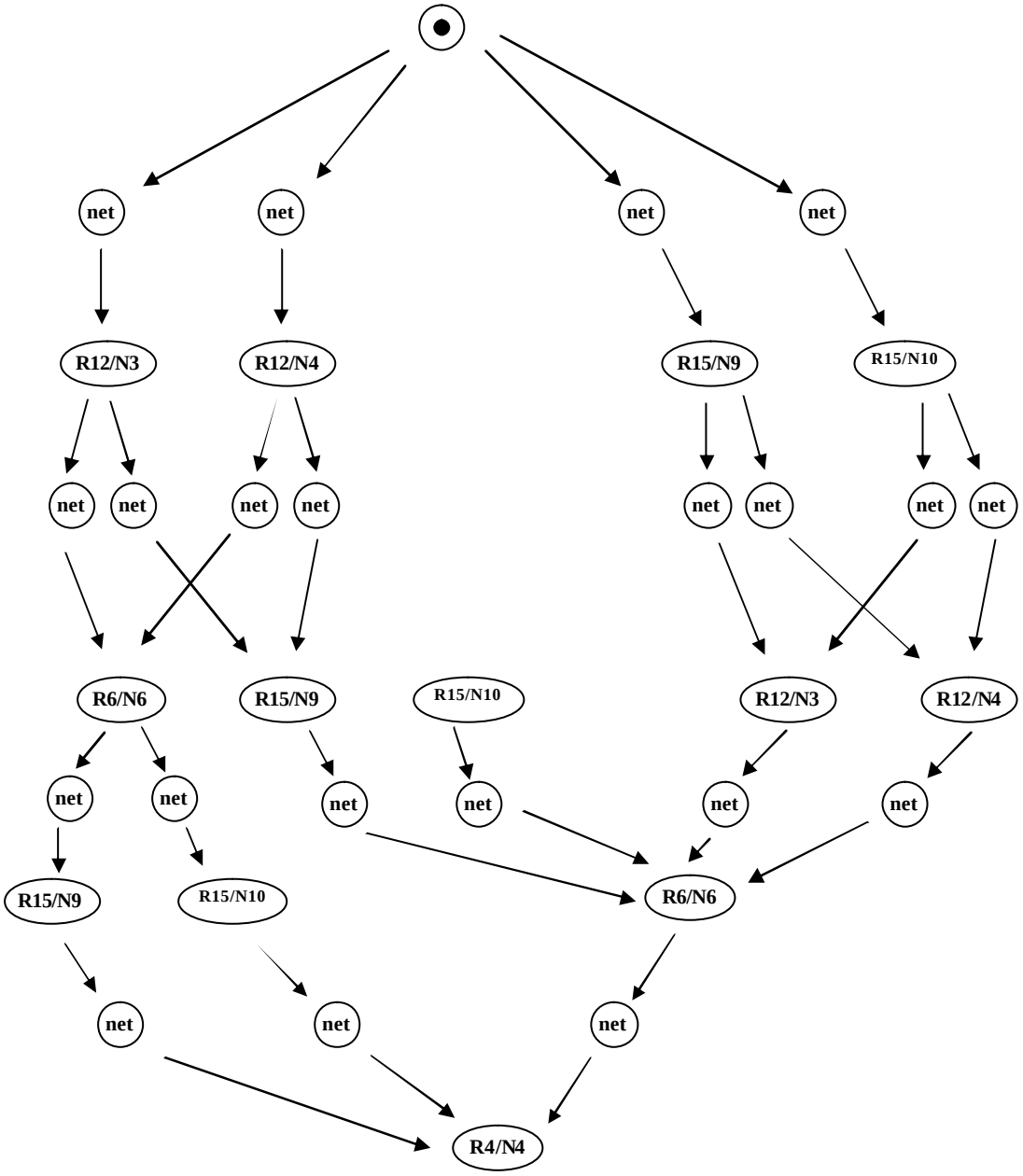
B.3 Segmento 3



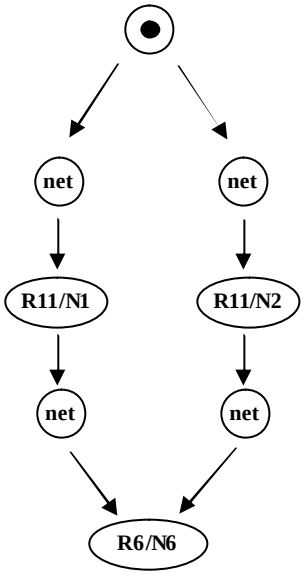
B.4 Segmento 4



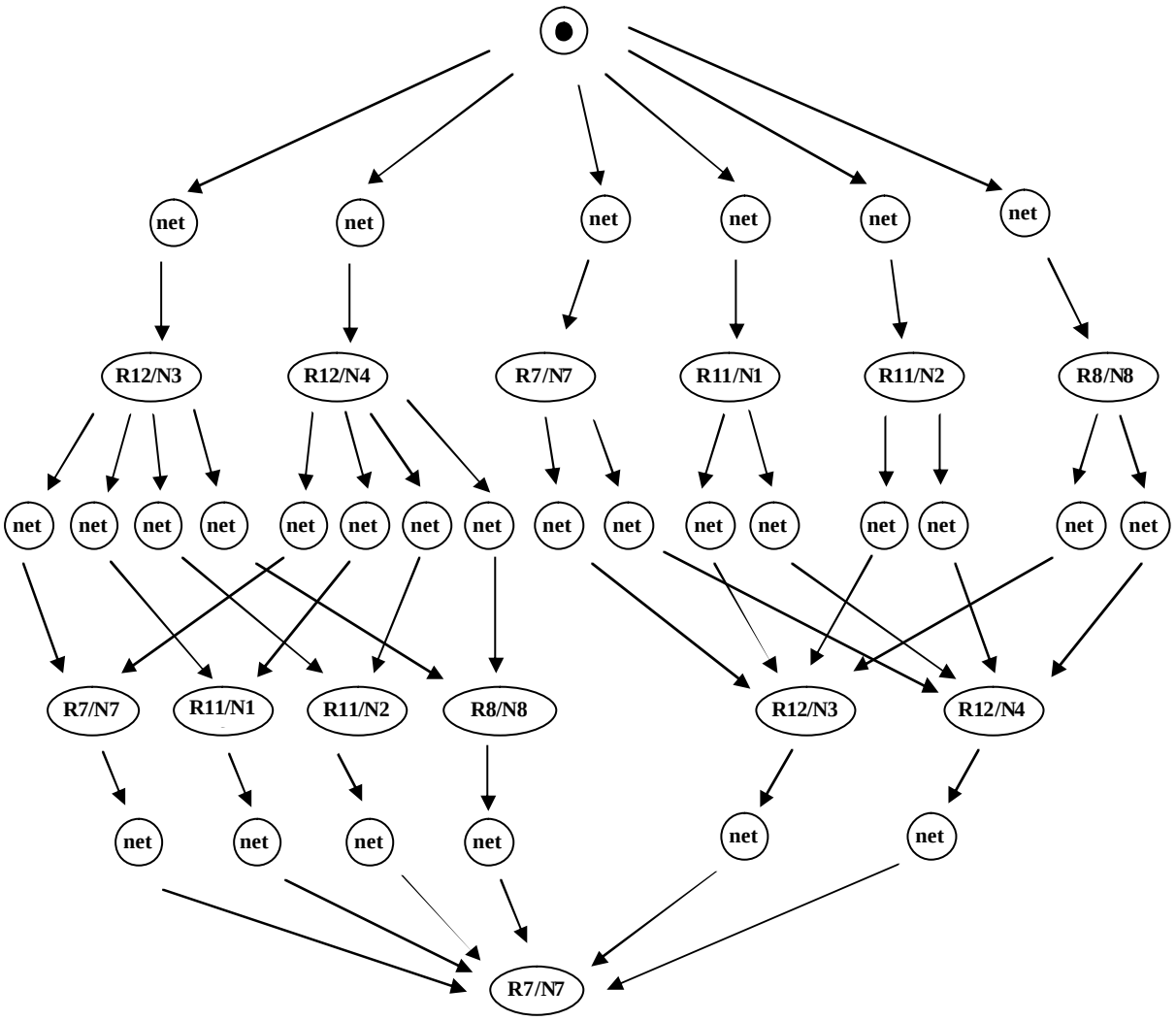
B.5 Segmento 5



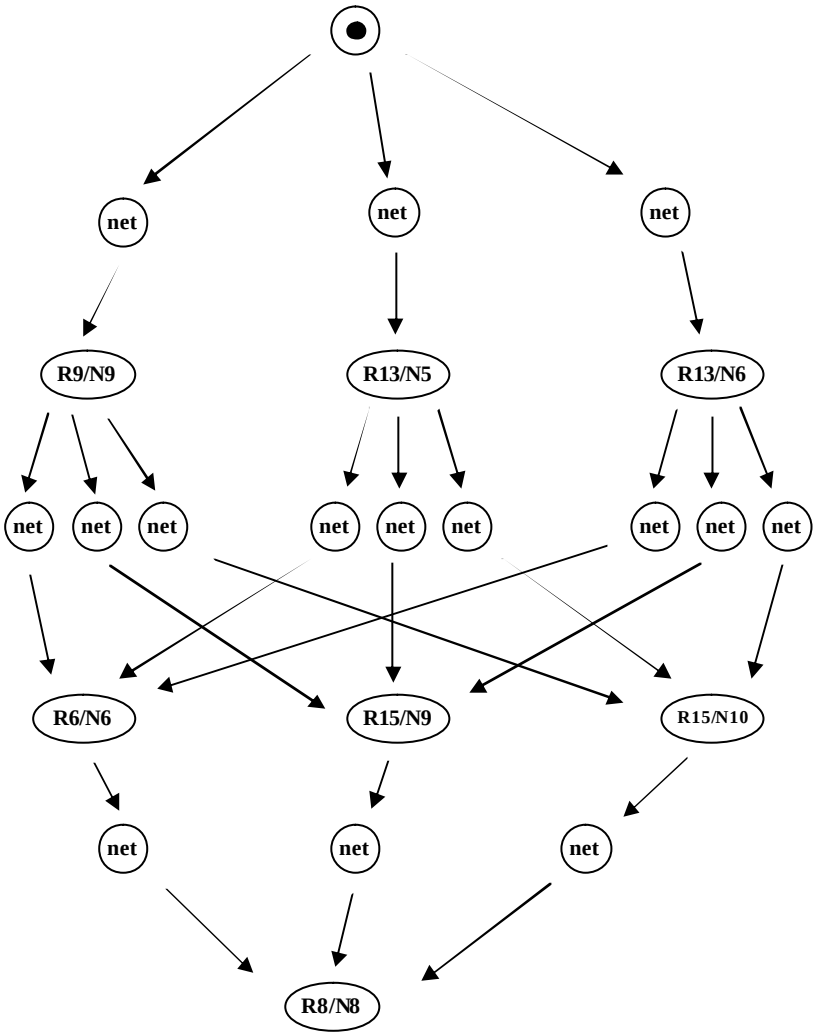
B.6 Segmento 6



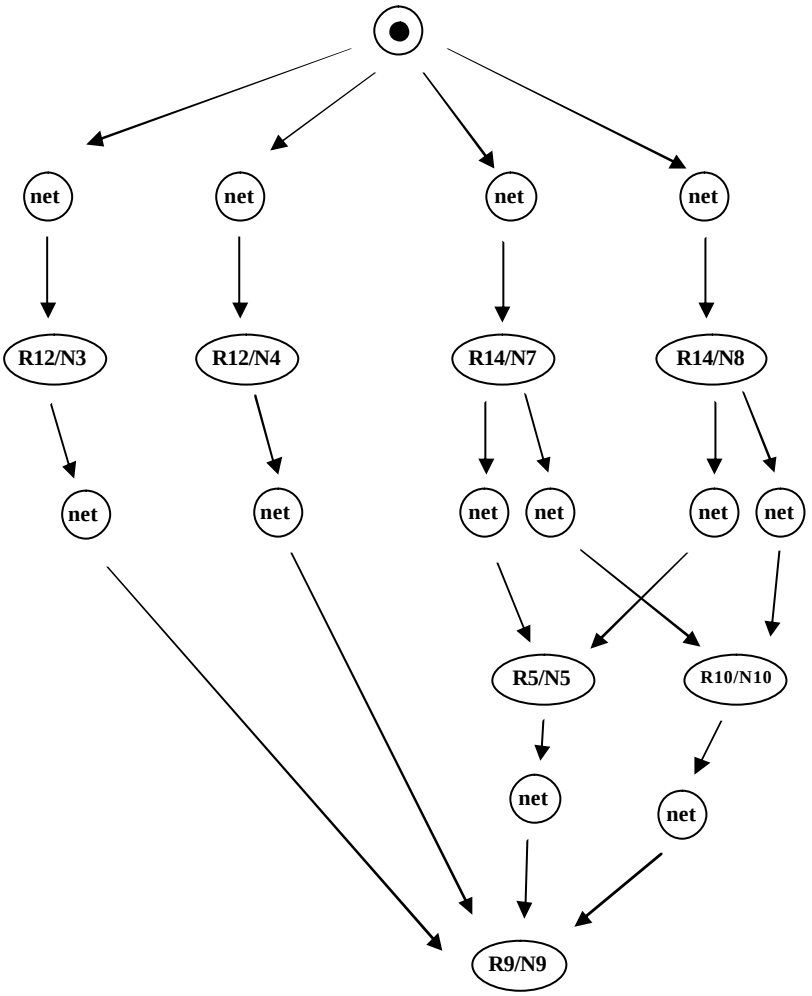
B.7 Segmento 7



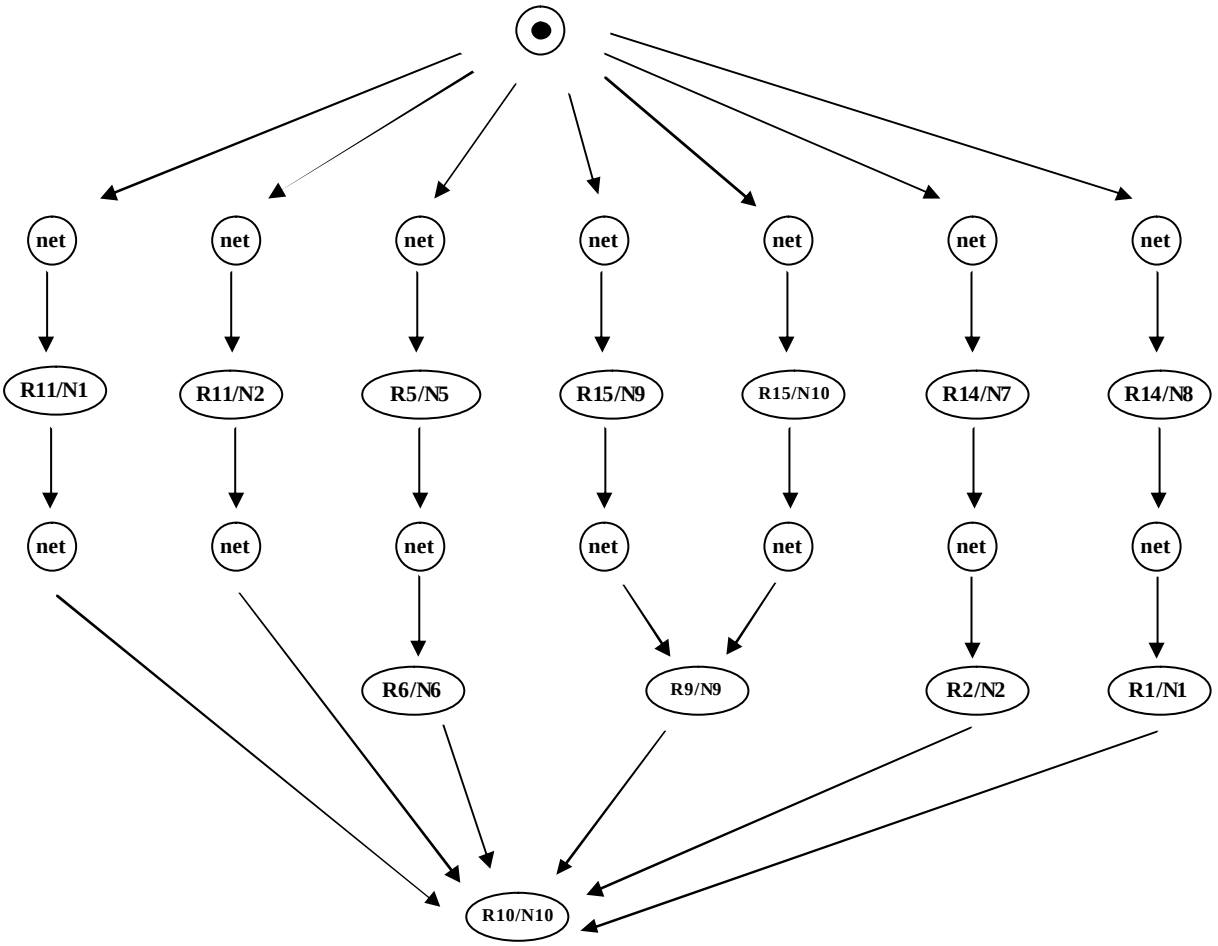
B.8 Segmento 8



B.9 Segmento 9



B.10 Segmento 10



Glossário

AG - Algoritmo Genético

APoP - *Adaptive Push or Pull Scheme*

ARS - Agent-based Routing System

BM - *Bid Manager*

CORBA - *Common Object Request Broker Architecture*

CR – Heurística Chance de Reversão

DSN - *Distributed Sensor Networks*

EDF - *Earliest Deadline First*

GCF - *Global Closest First*

IA - Inteligência Artificial

LCF - *Local Closest First*

MADSN - *Mobile Agent-based Distributed Sensor Networks*

MAF - *Mobile Agent Facility*

MANETs - *Mobile Ad-Hoc Networks*

MAP - *Mobile Agent Planning*

MASIF - *Mobile Agent Specification Interoperability Facility*

MaSS - *Mobile Agent Search System*

MAWSN - *Mobile Agent-Based Wireless Sensor Network*

MEITCA - *Mitsubishi Electric Information Technology Center America*

MTR – Heurística Memória do Tempo de Resposta

MV – Heurística Memória do Valor

OMG - *Object Management Group*

ORB - *Object Request Broker*

PCP - Priority Ceiling Protocol

PDAs - *Personal Digital Assistants*

PEs - Elementos Processadores

PG – Heurística Preguiçoso-Guloso

PIP - *Priority Inheritance Protocol*

QoS - *Quality-of-Service routing*

QTC – *QuantumTime Chunks*

RMI - *Remote Method Invocation*

RPC - *Remote Procedure Call*

SMA – Sistemas Multiagentes

STR - Sistemas de Tempo Real

TACOMA - *Troms And COrnell Mobile Agents*

TMAP - *Timed Mobile Agent Planning*

TSP - *Traveling Salesman Problem*

TVR - Tempo Virtual de Referência

UML - *Unified Modeling Language*

VMAS - *Visual Mobile Agent System*

WWW - *World Wide Web*

Referências Bibliográficas

[ARLOW 2005] ARLOW, J.; NEUSTADT, I. *UML 2 and The Unified Process*. Practical Object-Oriented. Segunda Edição. Analysis and Desing. Object Technology Series. Addison-Wesley Series Editors. 2005.

[BAEK 2001] BAEK, J.W.; KIM, G.T.; YEO, J.H.; et al. *Cost-Effective Mobile Agent Planning for Distributed Information Retrieval*. 21st *International Conference on Distributed Computing Systems* (ICDCS-21). Phoenix. Abril. 2001.

[BAEK 2001a] BAEK, J.W.; KIM, G.T.; YEOM, H.Y. *Timed Mobile Agent Planning for Distributed Information Retrieval* Escola de Computação e Engenharia. Seoul National University. Proceedings of AGENTS´01. Montreal. Quebec. Canadá. Junho. 2001.

[BALAS 1989] BALAS, E. "The Prize Collecting Traveling Salesman Problem" John Wiley & Sons, Inc. Networks, Vol.19, pp.621-636, 1989.

[BALAS 1995] BALAS, E. "The Prize Collecting Traveling Salesman Problem: II. Polyhedral Results", John Wiley & Sons, Inc. Networks, Vol.25, pp.199-216, 1995.

[BAR 2004] BARBETTA, P. A.; REIS, M. M.; BORNIA, A. C. Estatísticas para Cursos de Engenharia e Informática. Editora Atlas. ISBN: 85-224-3765-3. São Paulo. 2004.

[BRAGA 2000] BRAGA, A.L. *Utilização de Agentes Móveis em Recuperação e Troca de Dados*. Exame de Qualificação. Programa de Engenharia de Sistemas e Computação. Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2000.

[BREWINGTON 1999] BREWINGTON, B.; GRAY, R.; MOIZUMI, K.; et al. *Mobile Agents in Distributed Information Retrieval*. Springer Verlag, Capítulo 15, p.p.355-395, In: "Intelligent Information Agents". Ed. Klusch. 1999.

[BURNS 2001] BURNS, A.; WELLINGS, A. *Real-Time Systems and Programming Languages. Ada 95, Real-Time Java and Real-Time POSIX*. Terceira Edição. Addison-Wesley Editors. 2001.

[BUTTAZZO 1997] BUTTAZZO,G.; SENSINE, F.; ANCILOTTI,P. *GHOST: A Tool for Simulation and Analysis of Real-Time Scheduling Algoritms.* Proceedings of the IEEE Real-Time Educational Workshop (RTEM'97). Montreal, Canadá. 1997.

[BUTTAZZO 1999] BUTTAZZO, G. *Adaptive Bandwidth Reservation for Multimedia Computing*. Proceedings of the IEEE Real-Time Systems Symposium. Hong Kong. China. Dezembro, 1999.

[CARO 1998] CARO, C.D.; DORIGO, M. *Mobile Agents for Adaptative Routing*. Proceedings of 31st Hawaii International Conference on System Sciences (HICSS). 1998.

[CARZANIGA 1997] CARZANIGA, A.; PICCO, G.P.; VIGNA, G. *Designing Distributed Applications with Mobile Code Paradigms.* Proceedings of the 19th International Conference on *Software* Engineering (ICSE´97). Ed. ACM Press, p.p.22-32. 1997.

[CHANG 2000] CHANG, J.S.; CHANG, C.Y. *A Visual Mobile Agent System with Itinerary Scheduling.* Agents 2000. Barcelona, Espanha. 2000.

[CHEN 1995] CHEN, I.R. *On Applying Imprecise Computation to Real Time IA Systems.* Instituto de Engenharia da Informação. Universidade Nacional de Cheng Kung. Tainan, Taiwan. 1995.

[CHEN 2005] CHEN, M.; KWON, T.; CHOI, Y. *Data Dissemination based on mobile Agent in Wireless Sensor Networks.* Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN´05). IEEE Computer Society. 2005.

[CHEN 2006] CHEN, M.; KWON, T.; YUAN,Y.; et al. *Mobile Agent Based Wireless Sensor Networks.* Journal of Computers. Volume 1. N° 1. Abril. 2006.

[CHENG 1998] CHENG, S.; STANKOVIC,J.A.; RAMAMRITHAM, K. *Scheduling Algorithms for Hard Real-Time Systems: A Brief Survey*. Hard Real-Time Systems: Tutorial. Ed. J.A. Stankovic e K. Ramamritham. IEEE CS Press. p.p.150-173. 1988.

[CONCORDIA 1997] *Concordia: An Infrastructure for Collaborating Mobile Agents.* Proc. First Int. Workshop on Mobile Agents. AAAI Press. Menlo. Prk. Califórnia. <http://www.meitca.com/HSL/Projects/Concordia>. 1997.

[DAVIS 1995] DAVIS, A. et.al. Flexible Schaduling for Adaptable Real-Time Systems. Proceedings IEEE Real-Time Technologies and Application Symposium. pp. 230-239. Maio. 1995.

[DOTTI 2001] DOTTI, F. L.; DUARTE, L. M. Desenvolvimento de Aplicações Móveis Corretas. III Workshop de Comunicação Sem Fio 2001, Anais do WCSF 2001, Recife, Brasil, pp. 10-17, 2001.

[ERFURTH 2003] ERFURTH, C.; ROSSAK, W. *Autonomous Itinerary Planning for Mobile Agents.* Relatório Técnico. Friedrich-Schiller University Jena. Instituto de Informática. 2003.

[FARINES 2000] FARINES, J. M.; FRAGA, J. S.; OLIVEIRA, R. S. *Sistemas de Tempo Real*. Escola de Computação 2000. Departamento de Ciência da Computação USP. São Paulo. Brasil. 2000.

[FENG 1996] FENG, W.C. *Applications and Extensions of the Imprecise Computation Model*. Tese de Doutorado. Departamento de Ciência da Computação. Universidade de Illinois. 1996.

[FIPA 2003] Foundation for Intelligent Physical Agents. IEEE Computer Society. Disponível por www em: <http://www.fipa.org>. 2003.

[FRANKLING 1996] FRANKLING, S. *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.* Proceedings of Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag.1996.

[GAREY 1979] GAREY, M.; JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company. 1979.

[GELOWITZ 2005] GELOWITZ, Craig M. *Real- Time Acoustic Source Localization as a Mobile Agent Resource.* Dissertação de Mestrado. University of Regina. Canadá. <http://wwwlib.umi.com/dissertations/preview/MR03976>. 2005.

[GLITHO 2002] GLITHO, R.H.; OLOUGOUNA, E. *Mobile Agents and Their Use for Information Retrieval: A Brief Overview and an Elaborate Case Study*. IEEE Network. Janeiro/Fevereiro. pp.34-41. 2002.

[GRASSHOPPER 1999] IKV ++. *Grasshopper 2: The Agent Plataform*. <http://www.ikv.de/products/grasshopper2.index.html>. 1999.

[GRAY 1996] GRAY, R.; KOTZ, D.; NOG, S.; et al. *Mobile Agentes for Mobile Computing*. Relatório Técnico PCS-TR96-285. Departamento de Ciência da Computação. Dartmouth College. Hanover. E.U.A. 1996.

[GSCHWIND 2000] GSCHWIND, T. *Comparing Object Oriented Mobile Agent Systems.* 6th European Conference on Object-Oriented Programming (ECCOP) - Workshop on Mobile Object Systems. 2000.

[HARRISON 1995] HARRISON, C. G.; CHESS, D.; KERSHENBAUM, A. *Mobile Agents: Are they a good idea*? Relatório Técnico. IBM T. J. Watson Research Center. 1995.

[HE 2005] HE, Y.; WEN, W.; JIM, H.; LIU, H. *"Agent-based Mobile Service Discovery in Grid Computing*". Proceedings of the Fifth International Conference on Computer and Information Technology (CIT´05). IEEE Computer Society. 2005.

[HIGASHI 2003] HIGASHI, S.S.S.C. *Uma Abordagem Multiagente para o Auxílio à Recomposição de Sistemas Elétricos na Fase Coordenada*. Dissertação de Mestrado. CPGEEL. Departamento de Automação e Sistemas. UFSC. 2003.

[HU 2001] HU, W.; STARR, A.; ZHOU, Z.; LEUNG, A. *An Intelligent Integrated System Scheme for Machine Toll Diagnostics*. International Journal Adv. Manuf. Technology. Volume 18. pp. 836-841. 2001

[JONES 1997] JONES, M. B.; ROSU, D.; ROSU, M.-C. *CPU reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities*. 16th ACM Symposium on Operating Systems Principles. St. Malo. França. Outubro. 1997.

[KARNIK 1998] KARNIK, N. *Security in Mobile Agent System*. Tese de Doutorado. Universidade de Minnesota. <http://www.cs.umn.edu/Ajanta>. 1998.

[KIM 2000] KIM, D. *EXPRES: An event driven dynamic scheduling for distributed manufacturing system*. Dissertação de mestrado. Industrial Engineering, Pohang University of Science e Technology. POSTECH. Coréia. Fevereiro. 2000.

[KOPETZ 1997] KOPETZ H. *Real-Time Systems Design for Distributed Embedded Applications*. Kluwer Academic Publishers. 1997.

[LAL 1999] LAL, M. e PANDEY, R. *CPU Resource Control for Mobile Programs. First International Symposium on Agent Systems and Applications.* Outubro. 1999.

[LANGE 1998] LANGE, D.B. *Mobile Objects and Mobile Agents: The future of Distributed Computing*. European Conference on Object-Oriented Programming. Springer-Verlag Berlin Heidelberg. LNCS 1445, pp.1-12. 1998.

[LANGE 1998a] LANGE, D. B.; OSHIMA, M. *Programming and Deploying Java Mobile Agents with Aglets.* ISBN 0-201-32582-9. Addison-Wesley. 1998

[LEUNG 1982] LEUNG, J.Y.T.; WHITEHEAD, J. *On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks.* Performance Evaluation, 2(4). Dezembro. 1982.

[LIU 1973] LIU, C.L.; LAYLAND, J.W. *Scheduling Algoritms for Multiprogramming in a Hard-Real-Time Environment*. Jornal da ACM. Volume 20. Nº 1. Janeiro. 1973.

[LIU 1994] LIU, J. W. S.; SHIH, W.-K.; LIN, K.-J., et al. *Imprecise Computations*. Proceedings of the IEEE. Volume 82. Nº 1. pp. 83-94. Janeiro. 1994.

[LIU 1994a] LIU, J., et.al. *Use of Imprecise Computation to Enhance Dependability of Real-Time Systems.* Foundations of Dependable Computing: Paradigms for Dependable Applications. Editora Klumer. 1994.

[LOBO 2001] LOBO, E. A. O. *Avaliando a Interoperabilidade de Plataformas de Agentes Movéis Através da Padronização OMG MASIF*. Dissertação de Mestrado. Curso de Pós-Graduação em Ciência da Computação. UFSC. Florianópolis. Brasil. 2001.

[MARTINS 2002] MARTINS, P. B.; SILVA, N. R. V. *Agentes Móveis: Redes Inteligentes e Aplicações.* ISEP - Instituto Superior de Engenharia do Porto. Departamento de Engenharia Eletrônica. Portugal. 2002

[MILOJICIC 1999] MILOJICIC, D.; DOUGLIS, F.; WHEELER, R. *Mobility: Process, Computers and Agents.* Prentice-Hall. ISBN: 0-201-37928-7. Massachusetts. 1999.

[MOIZUMI 1998] MOIZUMI, K. e CYBENKO, G. *The Traveling Agent Problem*. Mathematics of Control, Signals and Systems. Janeiro. 1998.

[MOIZUMI 1998a] MOIZUMI, K. *Mobile Agent Planning Problems.* Tese de doutorado. Dartmouth College. 1998.

[MONTEIRO 2002] MONTEIRO, E.G.; LEITE, J.C.B. *Qualidade de Serviço para Programas Móveis de Tempo Real*. XXVIII Latin American Conference on Informatics (CLEI). Montevidéu. Uruguai. Novembro. 2002.

[MOSSE 2006] MOSSE, D.; BRUSTONOLI, J.; KHATTAB, S. Et al. *Integration of Application-Layer Scheduling and Routing in Delay-Tolerant MANETs*. . Relatório Técnico. CSD. University of Pittsburgh. EUA. 2006

[NAKANISHI 2000] NAKANISHI, F., et.al. *A Flexibel Scheduling for Automobile Control Using Imprecise Computation and its Fundamental Evaluation*. Departamento de Engenharia da Computação. Hiroshima, City University. 6th International Conference on Complex Computer Systems (ICECCS 00). Tokyo, Japão. 2000.

[NATARAJAN 1995] NATARAJAN, S. *Imprecise and Approximate Computation*. Kluwer Academic Publishers. Massachussetts. EUA. 1995.

[NWANA 1996] NWANA, H. S. *Software Agents: An Overview*. In: Knowledge Engineering Review. Cambridge University Press. V.11, n.3, pp1-40. 1996.

[OIDA 2000] OIDA, K.; SEKIDO,M. *ARS: an efficient agent-based routing system for QoS guarantees.* Elsevier Science. *Computer Communications* 23. pp. 1437-1447. 2000.

[OLIVEIRA 1997] OLIVEIRA, R. S. *Escalonamento de Tarefas Imprecisas em Ambiente Distribuído*. Tese de Doutorado. PGEEL. UFSC. Florianópolis. Brasil. 1997.

[OLIVEIRA 1997a] OLIVEIRA, R.S. *Computação Imprecisa*. Revista de Informática Teórica e Aplicada – RITA. Volume IV. Nº 1. Agosto. 1997.

[OMG 2000] OMG. *MAF - Mobile Agent FacilitySpecification*. Documento da OMG (*Object Management Group*). <http://www.omg.org/technology/documents/spec_catalog.htm>. 2000.

[ONG 2003] ONG, S. K.; SUN W. W. *Aplication of Mobile Agents in a Web-based Real-Time Monitoring System*. Int. J. Adv. Manuf. Technol. pp.33-40. 2003

[OSHIMA 1998] OSHIMA, M.; ARIDOR, Y. *Infraestruture for mobile agents: requirements and design*. Proceedings of International Workshop on Mobile Agents. 1998.

[PICCO 1998] PICCO, G.P. *Undersatanding, Evaluating, Formalizing, and Exploiting Code Mobility.* Tese de Doutorado. Politecno di Torino, Italia. 1998.

[PICCO 1998a] PICCO, G. P.; FUGGETTA, A.; VIGNA, G. *Understanding Code Mobility.* IEEE Transaction on Software Engineering. Janeiro. 1998.

[PLEISCH 2004] PLEISCH, S., SCHIPER, A. *Approaches for Fault-Tolerant and Transactional Mobile Agent Execution – An Algoritmic View*. *École Polytechnique Fédérale de Lausanne* (EPFL). Suíça. ACM Computing Surveys. Volume 36. N° 3. pp. 219-262. Setembro. 2004.

[QI 2001] QI, H.; WANG, F. Optimal Itinerary Analysis for Mobile Agents in Ad Hoc Wireless Sensor Networks. Proceedings of 15th International Conference on Wireless Communications. Julho. 2001.

[QI 2001a] QI, H.; IYENGAR, S.S.; CHAKRABARTY,K. *Multi-Resolution Data Integration Using Mobile Agents in Distributed Sensor Networks.* IEEE Transactions Systems, Man, and Cybernetics Part C: Applications and Rev. Volume 31. N° 3. pp. 383-391. Agosto. 2001.

[QU 2005] QU, W.; SHEN, H.; JIN, Y. *Theoretical Analysis on A Traffic-Based Routing Algorithm of Mobile Agents.* Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT). pp.520-526. Compiègne. França. Setembro. 2005.

[RAIBULET 2000] RAIBULET, C.; DEMARTINI C. *Mobile Agent Technology for the management of distributed systems-a case study*. Computer Network. Volume 34. pp. 823-830. 2000.

[RAMAMRITHAM 2002] RAMAMRITHAM, K.; KWAN, A.; LAM, K.Y. *RTMonitor: Real-time Data Monitoring Using Mobile Agent Technologies.* Proceedings of 28th VLDB Conference. Hong Kong. China. 2002.

[REIS 2001] REIS, A.L.A. *Comparação SNMP x Agentes Móveis para Gerência de Redes*. Dissertação de Mestrado. Curso de Pós-Graduação em Ciência da Computação. UFSC. Florianópolis. Brasil. 2001.

[ROBLES 2002] ROBLES, S.; NAVARRO, G.; PONS, J. et all. *Mobile Agents Supporting Secure GRID Environments.* E*uroweb 2002 Conference*. *British Computer Society*, Oxford, B. Matthews, B. Hopgood, M. Wilson, pp. 195-197. World Wide Web Consortium. Dezembro. 2002.

[RUBINSTEIN 2001] RUBINSTEIN, M.G. *Avaliação do Desempenho de Agentes Móveis no Gerenciamento de Redes.* Tese de Doutorado. Programa de Engenharia Elétrica. UFRJ. Rio de Janeiro. Brasil. 2001.

[SCHLEGEL 2006] SCHLEGEL, T.; BRAUN, P.; KOWALCZYK, R. *Torwards Autonomous Mobile Agents with Emergent Migration Behavior*. In Proceedings of Autonomous Agents and Multiagent Systems (AAMAS). pp.585-592. Maio. Hakodate. Japão. 2006.

[SCHMIDTKE 2001] SCHMIDTKE, S. L. *Uso de Computação Imprecisa e Reflexão Computacional como Mecanismo de Adaptação para Aplicações de Tempo Real*. Dissertação de Mestrado. CPGCC. Departamento de Informática e Estatística. UFSC. Florianópolis. Brasil. 2001.

[SELAMAT 2004] SELAMAT, A.; SELAMAT, H. *Routing Algorithm of Mobile Agents for Query Retrieval using Genetic Algorithm*. Malaysian Journal of Computer Science. Volume 17. Nº 2. pp. 1-10. Dezembro. 2004.

[SHEHORY 1998] SHEHORY, O.; SYCARA, K.; CHALASANI, P. et al. *Agent Cloning: An Approach to Agent Mobility and Resource Allocation*. Carnegie Mellon University. IEEE Commnications Magazine. pp. 58-67. Julho.1998

[SHIN 2001] SHIN, M.; JUNG, M.; RYU, K. *A Novel Negotiation Protocol for Agent-based Control Architecture.* Advanced of Mechanical and Industrial Engineering. Pohang University of Science and Technology, POSTECH. *5th International Conference on Engineering & Automation.* Las Vegas. USA. 2001.

[SHIN 2002] SHIN, M.; JUNG, M. *Mobile Agent-based bid generation scheme for real-time scheduling. Advanced of Mechanical and Industrial Engineering. Pohang University of Science and Technology* (POSTECH), Korea. Proceedings of the 30th International Conference on Computers and Industrial Engineering. Volume 2. pp. 809-814. Grécia. Junho/Julho.2002.

[SILVA 2005] SILVA, E. L.; MENEZES, E. M. *Metodologia da Pesquisa e Elaboração de Dissertação*. Quarta Edição. Universidade Federal de Santa Catarina. 138 p. Florianópolis. Brasil. 2005.

[SONG l2005] SONG, Lei. *Locating Services for Mobile Agents.* Dissertação de Mestrado. University of Guelph. Canadá. <http://wwwlib.umi.com/dissertations/preview/MR11164>. 2005.

[STANKOVIC 1988] STANKOVIC, J.A. *Misconceptions About Real-Time Architecture: A Serious Problem for Next Generation Systems.* IEEE Computer. Volume 21, Nº 10. pp.10-19. Outubro. 1988.

[STANKOVIC 1990] STANKOVIC, J.A. e RAMAMRITHAM, K. *What is Predictability for real-time Systems?*. The Journal of Real-Time Systems. 2. pp. 247-254. 1990.

[STANKOVIC 1992] STANKOVIC, J. A. *What is a Real-Time System*. Departamento de Ciência da Computação. Universidade de Massachussets. Amherst, Massachusetts. Abril. 1992.

[SUN 2003] SUN. Java™ 2 Platform, V.1.4.2 *Security Documentation*. Applets <http://java.sun.com/>. 2003.

[THORN 1997] THORN, J.; VALENTE, L. *Programming Languages for Mobile Code*. ACM Computing Surveys 29. 1997.

[TOLBERT 2001] TOLBERT, L. M.; QI, H.; PENG, F. Z. *Scalable Multi-Agent System for Real-Time Electric Power Mangement*. *IEEE Power Engineering Society Summer Meeting.* pp. 1676-1679. Vancouver. Canadá. Julho. 2001.

[TONG 2003] TONG, L.; ZHAO, Q.; ADIREDDY, S. *Sensor Networks with Mobile Agents.* Proceedings of IEEE MILCOM´03. Boston. MA. EUA. Outubro. 2003.

[VIN 1998] VINIOTIS, Y. *Probability and Random Process for Electrical Engineers*. Ed. Mc Graw Hill International Editions. Electrical Engineering Series. ISBN: 0-07-115872-3. 1998.

[VOYAGER 2003] Object Space. *Objectspace Voyager*. Guia do Desenvolvedor. Relatório Técnico. <http://www.objectspace.com/voyager.htm>. 2003.

[WANGHAM 2004] WANGHAM, M. *Esquema de Segurança para Agentes Móveis em Sistemas Abertos*. Tese de Doutorado. Curso de Pós-Graduação em Engenharia Elétrica. Departamento de Automação e Sistemas. Universidade Federal de Santa Catarina. Florianópolis. Brasil. 2004.

[WU 2004] WU, Q.; RAO, N.S.V.; BARHEN, J.; et al. *On Computing Mobile Agent Routes for Data Fusion in Distributed Sensor Networks.* IEEE Transactions on Knowledge and Data Engineering. Volume 16. N° 6. pp. 740-752. Junho. 2004.

[WU 2005] WU, Q.H.; FENG, J.Q.; TANG, W.H. *Multi-Agent Based Automation Plataform for Distributed Industrial Systems.* The IEE Control & Automation Professional Network. Autonomous Agents in Control. pp.19-26. 2005.

[YANG 1998] YANG, J.; HONAVAR, V.; MILLER, L. et al. *Intelligent Mobile Agents for Information Retrieval and Knowledge Discovery from Distributed Data and Knowledge Sources*. Proceedings of the IEEE Information Technology Conference. pp.99-102. Syracuse. NY. 1998.

[ZHOU 2005] ZHOU, T. *A fault tolerant architecture for mobile agent-Based e-commerce system*. Dissertação de Mestrado. DalHousie University. Canadá. <http://wwwlib.umi.com/dissertations/preview/MR00944>. 2005.